On the Solution of Nonlinear Semidefinite Programs by Augmented Lagrangian Methods

Den Naturwissenschaftlichen Fakultäten
der Friedrich-Alexander-Universität Erlangen-Nürnberg
zur
Erlangung des Doktorgrades

vorgelegt von

Michael Stingl

aus Waldsassen

Als Dissertation genehmigt von den Naturwissenschaftlichen
Fakultäten der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung:

Vorsitzender der Prüfungskommission:	Prof. Dr. D.-P. Häder
Erstberichterstatter:	Prof. Dr. G. Leugering Friedrich-Alexander-Universität Erlangen-Nürnberg
Zweitberichterstatter:	RNDr. M. Kočvara, DrSc. Academy of Sciences of the Czech Republic, Prague Prof. Dr. F. Jarre Heinrich-Heine-Universität Düsseldorf

Acknowledgements

This thesis has been growing over a period of several years. It is only natural that the thesis was influenced by a lot of people during that time. I would like to take the time to thank those who contributed directly or indirectly. In particular I want to thank

- my wife Andrea and my sons Jonas and Nikola, for their love and patience.
- my parents, who gave me the opportunity to start and finish my diploma studies in mathematics at the university of Erlangen.
- my former supervisor Prof. Dr. J. Zowe who gave me the opportunity to start my PhD studies at the chair of Applied Mathematics II in Erlangen. Unfortunately he could not continue his work after a serious accident shortly after the beginning of my PhD studies.
- Prof. Dr. G. Leugering who offered me to finish my PhD studies as a member of his chair and kindly took over the supervision of my thesis during the last year.
- Dr. M. Kočvara, who kindly took over the supervision of my PhD studies after the accident of Prof. Dr. J. Zowe. He always had an open ear for my questions. In innumerably many discussions, he shared his knowledge with me. He was always able to give me new inspirations and useful hints for further investigations.
- Dr. R. Sommer and Dr. R. Werner, – both former members of the chair of Applied Mathematics II in Erlangen – who gave me a lot of support especially in the first phase of my PhD studies.
- the colleagues at the Chair of Applied Mathematics II in Erlangen for the kind and helpful atmosphere during the time of my PhD studies.

Abstract

This thesis presents a method for the solution of nonlinear semidefinite programming problems. The goal of nonlinear semidefinite programming is to solve optimization problems subjected to nonlinear matrix inequalities. The method described can be interpreted as a generalization of the modified barrier method for the solution of nonlinear programs subjected to vector-valued inequality constraints. As such, the method belongs to the general category of augmented Lagrangian type methods. Along with a comprehensive convergence analysis, including local and global convergence aspects, emphasis is given to the efficient implementation of the method. A detailed algorithm is presented and numerical aspects of the computer code PENNON are described. The thesis concludes with reports on extensive numerical studies performed with the computer code PENNON . Moreover, two classes of nonlinear semidefinite programming problems arising from structural optimization and control theory are discussed in detail.

The *first chapter* contains an overview about related work in the area of nonlinear semidefinite programming. Further, examples of applications of nonlinear semidefinite programming are given, and a motivation for the development of the method described in this thesis is presented.

The *second chapter* introduces concepts from the area of matrix analysis. In particular, so called primary matrix functions and (directional) derivatives of matrix functions are defined.

In the *third chapter* of this thesis, the nonlinear semidefinite programming problem is specified in more detail. Several assumptions among them optimality conditions of first and second order and constraint qualifications are formulated. Furthermore the impact of these assumptions is discussed.

The *fourth chapter* introduces a class of matrix penalty functions. Concrete examples of this class are presented and matrix penalty functions, which are uniquely defined by associated real-valued penalty functions, are investigated.

Using the class of penalty functions defined in the preceding chapter, a class of augmented Lagrangian functions is introduced in the *fifth chapter*. This class is the basis of the algorithm for the solution of nonlinear semidefinite programming problems presented later in the thesis. Various important properties of these functions are proven.

At the beginning of the *sixth chapter*, a (local) algorithm for the solution of nonlinear semidefinite programs is introduced. It is shown that the local algorithm is well defined. A special version of the implicit function theorem plays an important role here. Contractivity of the algorithm is established under the assumption that the penalty parameter is kept constant. Finally, it is explained why in a certain neighborhood of the optimum, the solution of a non-convex semidefinite programming problem can be replaced by the solution of a sequence of unrestricted convex problems.

In the *seventh chapter*, two globalization techniques for the algorithm introduced in the preceding chapter are proposed. For each of the approaches certain convergence properties are established. At the end of this chapter, an algorithm combining the local and global convergence properties is presented.

The *eighth chapter* deals with the computational complexity of the algorithm. Complexity formulas are derived, which indicate serious problems of the (general) algorithm, when applied to large scale problems. A remedy is given by the choice

of a special penalty function. It is shown that using this special penalty function, the computational complexity can be reduced significantly and various sparsity types in the problem data can be exploited.

In the *ninth chapter* of the thesis, several sub-aspects of the algorithm are discussed. Among them are: the solution of the (generally non-convex) unrestricted subproblems, multiplier update formulas, penalty parameter update schemes, solution techniques for linear systems, initialization formulas and stopping criteria.

In *chapter ten*, nonlinear semidefinite programming problems arising in structural optimization and control theory are presented.

The *eleventh chapter* describes and discusses results of comprehensive numerical experiments. Apart from case studies in structural optimization and benchmarks with problems from control theory, comparisons of PENNON to alternative (linear) semidefinite programming solvers are given.

Finally, the *twelfth chapter* of this thesis offers an outlook on future research and possible improvements of the method.

Zusammenfassung

In Rahmen dieser Arbeit wird eine Methode zur Lösung nichtlinearer semidefiniter Programme eingeführt. Ziel der nichtlinearen semidefiniten Programmierung ist es, Optimierungsprobleme mit nichtlinearen Matrixungleichungen als Nebenbedingungen zu lösen. Die beschriebene Methode kann als Verallgemeinerung der modifizierten Barriere-Methode, die zur Lösung nichtlinearer Programme mit reellen Ungleichheitsnebenbedingungen entwickelt wurde, betrachtet werden und lässt sich als solche in den allgemeinen Kontext der Augmented-Lagrange-Methoden einordnen. Neben einer umfassenden Konvergenzanalyse, in deren Rahmen sowohl lokale als auch globale Konvergenzaussagen erarbeitet werden, liegt ein weiterer Schwerpunkt der Arbeit in einer effizienten Umsetzung der Methode. Ein detaillierter Algorithmus wird vorgestellt, und numerische Aspekte des Computerprogrammes PENNON werden erörtert. Im letzten Teil der Arbeit wird die Durchführung umfangreicher numerischer Experimente mit dem Computerprogramm PENNON beschrieben und deren Ergebnisse diskutiert. Desweiteren werden nichtlineare semidefinite Programme aus dem Bereich der Strukturoptimierung und der Steuerungstheorie genauer erläutert.

Im *ersten Kapitel* wird ein Überblick über weitere Arbeiten im Bereich der nichtlinearen semidefiniten Programmierung gegeben. Desweiteren werden wichtige Anwendungen der nichtlinearen semidefiniten Programmierung genannt. Schliesslich werden die Hintergründe, die zur Idee der im Rahmen dieser Arbeit vorgestellten Methode führten, erläutert.

Im *zweiten Abschnitt* werden wichtige Begriffe aus dem Bereich der Matrixanalysis bereitgestellt. Insbesondere werden sogenannte primäre Matrixfunktionen definiert und (Richtungs-)ableitungen von Matrixfunktionen diskutiert.

Im *dritten Kapitel* dieser Arbeit wird das Problem der nichtlinearen semidefiniten Programmierung genauer erläutert. Desweiteren werden verschiedene Voraussetzungen, darunter Optimalitätsbedingungen erster und zweiter Ordnung, sowie Constraint Qualifications gemacht und deren Bedeutung diskutiert.

Im *vierten Kapitel* wird eine Klasse von Penaltyfunktionen für Matrixnebenbedingungen eingeführt, die anhand von Beispielen konkretisiert wird. Matrix-Penaltyfunktionen, deren Definition auf Penaltyfunktionen basiert, die aus dem Bereich der nichtlinearen Programmierung (mit reellwertigen Nebenbedingungen) bekannt sind, werden auf Ihre Eignung untersucht.

Mit Hilfe der im vorangegangenen Abschnitt eingeführten Penaltyfunktionen wird im *fünften Kapitel* eine Klasse von Augmented-Lagrange-Funktionen definiert. Die hier beschriebene Klasse von Augmented-Lagrange-Funktionen stellt die Grundlage des später eingeführten Algorithmus zur Lösung nichtlinearer semidefiniter Programme dar. Es werden verschiedene entscheidende Eigenschaften nachgewiesen.

Im *sechsten Kapitel* wird ein (lokaler) Algorithmus zur Lösung nichtlinearer semidefiniter Programme definiert. Hier liegt der theoretische Schwerpunkt der Arbeit. Zunächst werden Wohldefiniertheit des Algorithmus und Existenzaussagen untersucht. Hierbei spielt eine spezielle Version des Satzes über implizite Funktionen eine besondere Rolle. Anschliessend wird Kontraktivität des Algorithmus bei fest gehaltenem Penaltyparameter bewiesen. Ferner wird erläutert, dass unter den gegeben Voraussetzungen in einer lokalen Umgebung des Optimums, die Lösung eines nichtkonvexen

semidefiniten Optimierungsproblems auf die Lösung einer Folge unrestringierter konvexer Optimierungsprobleme zurückgeführt werden kann.

Das *siebte Kapitel* befasst sich mit der Globalisierung des im sechsten Kapitel vorgestellten lokalen Algorithmus. Es werden zwei Globalisierungsansätze vorgestellt und entsprechende globale Konvergenzaussagen nachgewiesen.

Im *achten Abschnitt* wird die Rechenkomplexität unseres Algorithmus untersucht. Die errechneten Komplexitätsformeln legen die Wahl einer speziellen Penaltyfunktion nahe, mit deren Hilfe die Rechenkomplexität des Algorithmus erheblich reduziert werden kann. Desweiteren wird erläutert, wie spezielle Datenstrukturen (dünnbesetzte Matrizen) ausgenutzt werden können.

Im *neunten Teil* der Arbeit werden verschiedene Unteraspekte des Algorithmus genauer erläutert. Dazu gehören: Lösung von (unter Umständen nicht-konvexen) unrestringierten Unterproblemen, Multiplikatorupdateformeln, Penaltyparameterupdatestrategien, die effiziente Lösung linearer Gleichungssysteme, sowie die Initialisierung des Algorithmus und Abbruchkriterien.

Im *zehnten Kapitel* werden nichtlineare semidefinite Programme aus den Bereichen der Strukturoptimierung und der Steuerungstheorie genauer erläutert.

Im *elften Abschnitt* der Arbeit werden die Ergebnisse umfangreicher numerischer Experimente, darunter Fallstudien aus dem Bereich der Strukturoptimierung, Benchmarks mit Problemen aus dem Bereich der Steuerungstheorie und Vergleiche mit alternativer Software zur Lösung (linearer) semidefiniter Programme berichtet und diskutiert.

Abschliessend werden im *zwölften Abschnitt* Ausblicke auf mögliche Weiterentwicklungen der vorgestellten Methode gegeben und offene Punkte dargelegt.

Contents

List of Figures

List of Tables

Notation

$\mathbb{N}$	the natural numbers
$\mathbb{R}$	the real numbers
$\mathbb{R}^n$	real n-dimensional vectors over $\mathbb{R}$
$\mathbb{M}^{m,n}$	$m \times n$-matrices over $\mathbb{R}$
$\mathbb{S}^m$	symmetric $m \times m$-matrices over $\mathbb{R}$
$\mathbb{S}^m_+$	positive semidefinite $m \times m$-matrices over $\mathbb{R}$
$\mathbb{S}^m_{++}$	positive definite $m \times m$-matrices over $\mathbb{R}$
(a, b)	open interval in $\mathbb{R}$
$H^m(a, b)$	symmetric $m \times m$-matrices over $\mathbb{R}$ with spectrum in (a, b)
$\mathcal{C}(x^*)$	cone of critical directions
$\mathbb{O}_m$	zero matrix of order m
I_m	unit matrix of order m
$[a_i]_{i=1}^n$	column vector with entries a_i in i-th component
$[a_{i,j}]_{i,j=1}^m$	$m \times m$-matrix with entries $a_{i,j}$ in i-th row and j-th column
$A_{i,j}$	matrix entry in i-th row and j-th column
$\|v\|$	Euclidian norm of a real vector
$\|A\|$	Frobenius norm of a matrix
$\langle v, w \rangle$	standard inner product in $\mathbb{R}^n$
tr	trace operator
$\langle A, B \rangle$	standard inner product on $\mathbb{S}^m$
f	real valued objective function
$\mathcal{A}$	matrix valued constraint function
φ	real valued penalty function
Φ	matrix penalty function
Ω	set of feasible points
Ω_p	relaxed set of feasible points
$s_k(A)$	k-th eigenvector of matrix A
$\lambda_k(A)$	k-th eigenvalue of matrix A
$E_0, E_\perp, P_0, P_\perp$	projection matrices
$P_k(A)$	Frobenius covariance matrices

$F'_x(x,U,p)$	partial derivative of F w. r. t. x
$F''_{xx}(x,U,p)$	second order partial derivative of F w. r. t. x
$\mathcal{A}'_i$	partial derivative of $\mathcal{A}$ w. r. t. x_i
$\mathcal{A}''_{ij}$	second order partial derivative of $\mathcal{A}$ w. r. t. x_i and x_j
$\mathcal{A}_i$	partial derivative of $\mathcal{A}$ w. r. t. x_i, evaluated at x^*
$\mathcal{A}_{ij}$	second order partial derivative of $\mathcal{A}$ w. r. t. x_i and x_j, evaluated at x^*
$D\Phi(A)$	derivative of Φ w. r. t. A
$D\Phi(A)[B]$	directional derivative of Φ w. r. t. A in direction B
$D^2\Phi(A)[B,C]$	second order directional derivative of Φ w. r. t. A in directions B and C
$\Delta\varphi$	divided difference formula of first order
$\Delta^2\varphi$	divided difference formula of second order

Chapter 1

Introduction

In the recent years (linear) semidefinite programming problems have received more and more attention. One of the main reasons is the large variety of applications leading to semidefinite programs (see [5], [40], for example). As a consequence, various algorithms for solving linear semidefinite programs have been developed, as for instance interior point or dual scaling methods (see, for example, [40],[79], [82] and the references therein). However, mathematical models in several applications lead to problems, which can not be formulated as *linear*, but more generally as *nonlinear* semidefinite programming problems. For example, such an observation can be made in control theory: There are many relevant control problems, which boil down to linear semidefinite programming problems (see [20] for a long list), however, there are still fundamental problems for which no linear semidefinite formulation has been found yet. This was the reason why so called BMI formulations (problems with bilinear matrix inequalities) of control problems became popular in the mid 1990s [39]. As there were, however, no computational methods for solving (non-convex) BMIs, or more generally speaking nonlinear semidefinite programs, several groups of researchers started to develop algorithms and software for the solution of BMI problems. For example, interior-point constrained trust region methods were proposed in [57] for a special class of BMIs. Further approaches were presented in [33] and [34]. In [34], sequential semidefinite programming, as an extension of quadratic programming, was used to solve LMI problems with additional nonlinear matrix equality constraints, while in [33] the augmented Lagrangian method was applied to a similar class of problems.

Later, more applications, as, for example, robust optimization (see [5]) or structural optimization (see, e.g., [4]), appeared, where nonlinear semidefinite programming played an important role. Consequently the interest in general purpose nonlinear semidefinite programming methods and software grew. Nowadays, there are several approaches for the solution of nonlinear semidefinite programs: For example, in [28], the method proposed in [34] is generalized to general purpose nonlinear semidefinite programming problems and a proof of global convergence is given. Another promising approach for the solution of general purpose nonlinear semidefinite programming problems is presented in [46]. The method is an extension of a primal predictor corrector interior method to non-convex programs, where the corrector steps are based on quadratic subprograms that combine aspects of line search and trust region methods. Just recently a smoothing type algorithm for the solution of nonlinear semidefinite programming problems was introduced in [47] and further explored in [48].

Our main motivation for the development of an own general purpose semidefinite programming solver was given by nonlinear semidefinite programming problems arising in structural optimization. The difficulty with these problems is that depending on the type of constraints which are taken into consideration (as, for example, compliance constraints, strain constraints, constraints on global stability, etc.), one obtains large scale semidefinite programs with entirely different sparsity structures. On the other hand there exist examples, where the main difficulty is not in the structure or size, but in the nonlinearity of the problem. Consequently we came to the conclusion that there are at least two basic requirements to be satisfied by a solver, which should be applicable to a wide class of (nonlinear) semidefinite optimization problems: First, the solver should be able to cope with serious non-linearities and, second, the solver should be able to exploit various types of sparsity in the problem data. At the time when we started with the development of our algorithm, there was – to our best knowledge – no such solver available.

As a consequence of the special requirements, it seemed to be a natural idea to search for a concept, which combined in a certain way the features of highly developed (standard) nonlinear programming algorithms with the abilities of (large-scale) linear semidefinite programming solvers. Our first and probably the most obvious idea was to generalize interior point algorithms which were successfully applied to both, nonlinear programming problems and linear semidefinite programming problems. However, we encountered some difficulties in the generalization of the interior point idea, as, for example, the combination of the symmetrization issue with the requirement of preserving sparsity in the problem data. Consequently we drew our attention to another promising method, the so called Penalty-Barrier-Multiplier method (PBM), which had been recently invented by Ben-Tal and Zibulevsky for the solution of convex programming problems (see [7]) and later adapted by Zibulevsky and Mosheyev for the application to linear semidefinite programming problems (see [63] and [87]). Nevertheless, also the Zibulevsky/Mosheyev approach turned out to have some serious drawbacks. First of all, the convergence theory was done solely for convex programming problems and, second, the semidefinite version of the PBM method was based on eigenvalue decompositions of the constraint matrices, which are known to be critical numerical operations not only in terms of computational complexity, but also in terms of robustness. The first issue was not that crucial, because there were already successful attempts to generalize the PBM idea to non-convex problems (see, for example, [21], where a global convergence result is given) and there was a rich theory for the so called modified barrier function method (MBF), invented by R. Polyak (see, for example, [68], [69] and [70]), which is in certain respect the basis of the PBM methods. A remedy for the second issue was discovered, when we investigated the concepts of PBM and MBF methods: First we made the observation that a generalized PBM function applied to a matrix constraint could result in a non-convex function, even in the case of convex problem data. Second, we found a way how to avoid eigenvalue decompositions, when using a (generalized version of a) special modified barrier function instead. A nice consequence of this observation was that with the special choice of the modified barrier function, we saw a clear way, how to preserve sparsity patterns in the problem data. Consequently we decided to base our algorithm on the concept of the MBF method. Nevertheless, we did not want to restrict our theoretical considerations (convergence proofs, etc.) to an algorithm, which is based on one specific penalty function. Therefore we decided to develop

- an augmented Lagrangian algorithm based on the MBF method combined with a wide class of penalty functions,
- a comprehensive local and global convergence theory, and
- a specialized code which allows for the solution of large scale nonlinear semidefinite programs by extensive utilization of problem inherent sparsity structures.

Chapter 2

Functions of Matrices

In the course of this section we introduce a class of matrix functions defined on the space of symmetric $m \times m$-matrices and henceforth denoted by $\mathbb{S}^m$. The definition of these functions is based on a spectral decomposition of a symmetric matrix and a real-valued function.

Definition 2.1 *(Primary matrix function) Let $\varphi : \mathbb{R} \to \mathbb{R}$ be a given function and $A \in \mathbb{S}^m$ be a given symmetric matrix. Let further $A = S\Lambda S^\top$, where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)^\top$, be an eigenvalue decomposition of A. Then the primary matrix function Φ corresponding to φ is defined by*

$$\begin{aligned} \Phi \quad : \quad & \mathbb{S}^m \to \mathbb{S}^m \\ & A \longmapsto S \begin{pmatrix} \varphi(\lambda_1) & 0 & \ldots & 0 \\ 0 & \varphi(\lambda_2) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & \varphi(\lambda_m) \end{pmatrix} S^\top . \end{aligned}$$

Definition 2.1 is a version of a more general definition for hermitian matrices presented in [45].

Example 2.1 Consider the real-valued function

$$\varphi : \left\{ \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & x^2 \end{array} \right. .$$

Then the primary matrix function $\Phi : \mathbb{S}^m \to \mathbb{S}^m$ is given by

$$\Phi(A) = S \begin{pmatrix} \lambda_1^2 & 0 & \ldots & 0 \\ 0 & \lambda_2^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & \lambda_m^2 \end{pmatrix} S^\top = S\Lambda^2 S^\top = A^2,$$

where $S\Lambda S^\top$ is an eigenvalue decomposition of A.

By the following definition we introduce a notation for the set of symmetric matrices with spectrum in a certain interval.

Definition 2.2 *Let $(a,b) \subset \mathbb{R}$ and $m \in \mathbb{N}$ be given. Then by $H^m(a,b)$ we define the set of all symmetric matrices $A \in \mathbb{S}^m$ with $\lambda_1(A) > a$ and $\lambda_m(A) < b$, where $\lambda_1(A), \lambda_2(A), \ldots, \lambda_m(A)$ are the increasingly ordered eigenvalues of A.*

Next we want to discuss monotonicity and convexity of primary matrix functions. Before we are able to do this, we need to introduce a partial order on the space of symmetric matrices. We start with the following definition:

Definition 2.3 *A matrix $A \in \mathbb{S}^m$ is said to be* positive semidefinite*, if all eigenvalues of A are nonnegative.*

In analogy to definition 2.3 a matrix $A \in \mathbb{S}^m$ is said to be *negative semidefinite*, if all eigenvalues of A are non-positive. It is well known that the set of all positive semidefinite $m \times m$-matrices, denoted by $\mathbb{S}^m_+$ is a pointed convex cone. This cone induces a partial order on $\mathbb{S}^m$, the so called Loewner (partial) order: For two given matrices $A, B \in \mathbb{S}^m$, we define

$$A \preceq B :\Leftrightarrow B - A \text{ is positive semidefinite}$$

and

$$A \succeq B :\Leftrightarrow A - B \text{ is positive semidefinite.}$$

In accordance with this definition we use the notation $A \succeq 0$ to indicate that the matrix A is positive semidefinite and $A \preceq 0$ for negative semidefinite matrices.

Now we are prepared for the following definition:

Definition 2.4 *A given matrix function $\Phi : \mathbb{S}^m \to \mathbb{S}^m$ is said to be* monotone*, if for any two matrices A, $B \in \mathbb{S}^m$ with $A \succeq B$ the inequality $\Phi(A) \succeq \Phi(B)$ is satisfied. Furthermore Φ is said to be* convex *if the inequality*

$$\Phi(\lambda A + (1-\lambda)B) \preceq \lambda\Phi(A) + (1-\lambda)\Phi(B)$$

is valid or all $A, B \in \mathbb{S}^m$ and any $0 \leq \lambda \leq 1$.

The following example demonstrates that it is generally wrong to conclude from the monotonicity/convexity of φ to the monotonicity/convexity of the corresponding primary matrix function:

Example 2.2 Let φ be given as in example 2.1. Obviously φ is monotone on the interval $[0,\infty)$. Nevertheless, the same property does not hold for the corresponding primary matrix function. To see this, consider the following counter example:

Let $A = \begin{pmatrix} 20 & 2 \\ 2 & 1 \end{pmatrix}$ and $B = \begin{pmatrix} 2 & 1 \\ 1 & 0.5 \end{pmatrix}$. Then the matrix $A - B = \begin{pmatrix} 18 & 1 \\ 1 & 0.5 \end{pmatrix}$ has two strictly positive eigenvalues. On the other hand the matrix

$A^2 - B^2 = \begin{pmatrix} 399 & 39.5 \\ 39.5 & 3.75 \end{pmatrix}$ has one positive and one negative eigenvalue.

A useful criterion for the determination of monotonicity and convexity of a primary matrix function is based on the following definition and formulated in Lemma 2.1:

Definition 2.5 *Let (a,b) be a given real interval, let $t_1, t_2, \dots, t_m$ be m real values and let $\varphi : \mathbb{R} \to \mathbb{R}$ be a twice continuously differentiable function. Then we define*

$$\Delta\varphi(t_i,t_j) = \begin{cases} \dfrac{\varphi(t_i)-\varphi(t_j)}{t_i-t_j} & , \text{ for } i \neq j \,, \\ \varphi'(t_i) & , \text{ for } i = j \,, \end{cases}$$

$$\Delta^2\varphi(t_i,t_j,t_k) = \begin{cases} \dfrac{\Delta\varphi(t_i,t_k)-\Delta\varphi(t_j,t_k)}{t_i-t_j} & , \text{ for } i \neq j \,, \\ \dfrac{\Delta\varphi(t_i,t_j)-\Delta\varphi(t_k,t_j)}{t_i-t_k} & , \text{ for } i = j \neq k \,, \\ \varphi''(t_i) & , \text{ for } i = j = k. \end{cases}$$

Lemma 2.1 *Let (a,b) be a given real interval. Let further $\varphi : (a,b) \to \mathbb{R}$ be a twice continuously differentiable function. Then:*

a) The primary matrix function Φ corresponding to φ is monotone on $H^m(a,b)$, if and only if the matrix $\left[\Delta\varphi(t_i,t_j)\right]_{i,j=1}^m$ is positive semidefinite for each set of real values $t_1, \dots, t_m \in (a,b)$.

b) The primary matrix function Φ corresponding to φ is convex on $H^m(a,b)$, if and only if the matrix $\left[\Delta^2\varphi(t_i,t_j,t_1)\right]_{i,j=1}^m$ is positive semidefinite for each set of real values $t_1, \dots, t_m \in (a,b)$.

Proof. See, for example, [45, p. 537ff]. □

Lemma 2.1 allows us to conclude from certain properties of φ to the monotonicity and convexity of the corresponding primary matrix function. This motivates the following definition:

Definition 2.6 *A real-valued function φ defined on an interval $(a,b) \subseteq \mathbb{R}$ is*

a) operator monotone *on (a,b), if $\left[\Delta\varphi(t_i,t_j)\right]_{i,j=1}^m \succeq 0$ for all $t_1, \dots, t_m \in (a,b)$ and all $m \in \mathbb{N}$.*

b) operator convex *on (a,b), if $\left[\Delta^2\varphi(t_i,t_j,t_1)\right]_{i,j=1}^m \succeq 0$ for all $t_1, \dots, t_m \in (a,b)$ and all $m \in \mathbb{N}$.*

In the remainder of this section we want to discuss derivative formula for a mapping composed from a twice continuously differentiable mapping $\mathcal{A} : \mathbb{R}^n \to \mathbb{S}^m$ and a primary matrix function on $\mathbb{S}^m$. A useful tool in this context is provided by the following definition:

Definition 2.7 *(Frobenius covariance matrices) Let $\mathcal{A} : D \subset \mathbb{R}^n \to \mathbb{S}^m$ be a given mapping. Let $\lambda_1(x), \lambda_2(x), \dots, \lambda_{\mu(x)}(x)$ denote the $\mu(x)$ increasingly ordered distinct eigenvalues of $\mathcal{A}(x)$. Let further $\mathcal{A}(x) = S(x)\Lambda(x)S(x)^\top$ be an eigenvalue decomposition of $\mathcal{A}(x)$, with $\Lambda(x) = \operatorname{diag}(\lambda_1(x), \dots, \lambda_1(x), \dots, \lambda_{\mu(x)}, \dots, \lambda_{\mu(x)})$, where* *each eigenvalue occurs in its given multiplicity at x. Then the* Frobenius covariance matrices *are defined by*

$$P_{\mathcal{A},i}(x) = S(x)\mathrm{diag}(0,\dots,0,\dots,1,\dots,1,\dots,0,\dots,0)S(x)^\top, \quad i = 1,\dots,\mu(x),$$

where the non-zeros in the diagonal matrix occur exactly in the positions of $\lambda_i(x)$ in Λ.

Some important properties of Frobenius covariance matrices are summarized in the following Lemma:

Lemma 2.2 *Let $\mathcal{A}(x) \in \mathbb{S}^m$ be given.Then the following equations hold:*

1. $P_{\mathcal{A},1}(x) + \dots + P_{\mathcal{A},\mu(x)}(x) = I_m$;
2. $P_{\mathcal{A},i}(x)P_{\mathcal{A},j}(x) = 0$ *for all* $i \neq j,\ i,j = 1,\dots,\mu(x)$;
3. $P_{\mathcal{A},i}(x)^k = P_{\mathcal{A},i}(x)$ *for all* $k \geq 1,\ i = 1,\dots,\mu(x)$.

Proof . See, for example, [45, p. 403]. □

For simplicity of notation we omit the subscript $\mathcal{A}$ and henceforth use the abbreviation $P_k(x) = P_{\mathcal{A},k}(x)$ for all $i = 1,\dots,\mu(x)$ and all $x \in \mathbb{R}^n$.

In the scope of the following Theorem we want to present formulas for partial derivatives of the mapping $\Phi \circ \mathcal{A}\colon D \to \mathbb{S}^m$. In order to keep notation as brief as possible, we introduce the abbreviations

$$\mathcal{A}'_i(x) = \frac{\partial}{\partial x_i}\mathcal{A}(x) \text{ and } \mathcal{A}''_{i,j}(x) = \frac{\partial^2}{\partial x_i \partial x_j}\mathcal{A}(x)$$

for the first and second order partial derivatives of the mapping $\mathcal{A}$.

Theorem 2.3 *Let $(a,b) \subseteq \mathbb{R}$, $m \in \mathbb{N}$ and $\mathcal{A} : D \subset \mathbb{R}^n \to \mathbb{S}^m$ be a twice continuously differentiable mapping. Denote by $\lambda_1(x), \lambda_2(x), \dots, \lambda_{\mu(x)}(x)$ the $\mu(x)$ increasingly ordered distinct eigenvalues of $\mathcal{A}(x)$ and let $\lambda_1(x) \geq a$ and $\lambda_{\mu(x)}(x) \leq b$ for all $x \in D$. Let further $\varphi :\ (a,b) \to \mathbb{R}$ be a twice continuously differentiable function and Φ be the corresponding primary matrix function. Then $\Phi \circ \mathcal{A}$ is twice differentiable for all $x \in D$ and the following formulas hold:*

$$\begin{aligned}
\frac{\partial}{\partial x_i}\Phi(\mathcal{A}(x)) &= \sum_{k,l=1}^{\mu(x)} \Delta\varphi(\lambda_k(x),\lambda_l(x))P_k(x)\mathcal{A}'_i(x)P_l(x) \\
&= S(x)\left([\Delta\varphi(\lambda_k(x),\lambda_l(x))]_{k,l=1}^m \bullet \left[S(x)^\top \mathcal{A}'_i(x)S(x)\right]\right)S(x)^\top
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2}{\partial x_i \partial x_j}\Phi(\mathcal{A}(x)) &= \sum_{k,l,s=1}^{\mu(x)} \Delta^2\varphi(\lambda_k(x),\lambda_l(x),\lambda_s(x)) \cdot (M + M^\top)\ + \\
& \sum_{k,l=1}^{\mu(x)} \Delta\varphi(\lambda_k(x),\lambda_l(x))P_k(x)\mathcal{A}''_{i,j}(x)P_l(x),
\end{aligned}$$

where the matrix M is given by

$$M = P_k(x)\mathcal{A}_i'(x)P_l(x)\mathcal{A}_j'(x)P_s(x)$$

and '$\bullet$' denotes the Hadamard product, defined by $A \bullet B = [A_{i,j}B_{i,j}]_{i,j=1}^m$ for any pair of matrices $A, B \in \mathbb{S}^m$.

Proof . Theorem 2.3 is a direct consequence of Theorem 6.6.30 in [45]. □

Due to the construction of $\Phi \circ \mathcal{A}$ as a composition of two mappings, the first formula in Theorem 2.3 can be interpreted as a directional derivative of Φ at $\mathcal{A}(x)$ in direction $\mathcal{A}_i'(x)$. The following Corollary generalizes this observation:

Corollary 2.4 *Let the assumptions of Theorem 2.3 hold. Let further two matrices $B, C \in \mathbb{S}^m$ be given. Then the directional derivatives of $\Phi_p(\mathcal{A}(x))$ with respect to $\mathcal{A}(x)$ in direction B is given by:*

$$\begin{aligned} D\Phi_p\left(\mathcal{A}(x)\right)[B] &= \frac{\partial}{\partial A(x)}\Phi(A(x))[B] \\ &= \sum_{k,l=1}^{\mu(x)} \Delta\varphi(\lambda_k(x),\lambda_l(x))P_k(x)BP_l(x) \\ &= S(x)\left([\Delta\varphi(\lambda_k(x),\lambda_l(x))]_{k,l=1}^m \bullet \left(S(x)^\top BS(x)\right)\right)S(x)^\top. \end{aligned}$$

Furthermore the second order directional derivative of $\Phi_p(\mathcal{A}(x))$ with respect to $\mathcal{A}(x)$ in directions B and C is given by:

$$\begin{aligned} D^2\Phi_p\left(\mathcal{A}(x)\right)[B;C] &= \frac{\partial^2}{\partial^2 A(x)}\Phi(\mathcal{A}(x))[B;C] \\ &= D\left(D\Phi_p\left(\mathcal{A}(x)\right)[B]\right)[C] \\ &= 2\sum_{k,l,s=1}^{\mu(x)} \Delta^2\varphi(\lambda_k(x),\lambda_l(x),\lambda_s(x))\cdot(N+N^\top), \end{aligned}$$

where the matrix N is given by $N = P_k(x)BP_l(x)CP_s(x)$.

Chapter 3

Problem Formulation and Basic Assumptions

Throughout this section we briefly describe the class of semidefinite programming problems, we want to solve by our algorithm. Furthermore we formulate basic assumptions involving constraint qualifications and optimality conditions for this class of semidefinite programming problems. For a comprehensive discussion on optimality conditions and constraint qualifications for semidefinite programming problems we refer the reader, for example, to [17], [72] or [40].

3.1 Problem Formulation

We consider the finite dimensional Hilbert space $\mathbb{S}^m$ introduced in Chapter 2, equipped with the inner product

$$\langle A, B\rangle = \mathrm{tr} A^\top B = \mathrm{tr} AB \quad \text{for all } A, B \in \mathbb{S}^m,$$

where tr denotes the trace operator. As we have already seen in the first chapter of this thesis, $\mathbb{S}^m_+$ induces a partial order "$\preceq$" respectively "$\succeq$" on $\mathbb{S}^m$. Using this notation, the basic semidefinite programming problem can be written as

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \quad \mathcal{A}(x) \preccurlyeq 0\,. \end{aligned} \tag{SDP}$$

Here $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{A} : \mathbb{R}^n \to \mathbb{S}^m$ are twice continuously differentiable mappings. In the case when f and $\mathcal{A}$ are convex, we refer to the basic problem as (CSDP).

Remark . Problem (SDP) belongs to a wider class of optimization problems called conic programs. Other important representatives of this class are (standard) nonlinear programming problems of the type

$$\begin{aligned}&\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \quad g(x) \le 0\,,\end{aligned} \tag{NLP}$$

where g maps from $\mathbb{R}^n$ to $\mathbb{R}^m$ or problems involving so called second order cone constraints. An interesting fact is that the problem (NLP) can be considered as a sub-case of the problem (SDP), namely, when $\mathcal{A}$ is a diagonal matrix. Consequently, the results in this thesis can be directly applied to inequality constrained nonlinear programming problems. Vice versa it should be emphasized that many ideas presented in this thesis are motivated by existing results from the nonlinear programming literature, as, for example, [7], [68], [69], [70], [26], [25] and [21]. It should be further mentioned that it is beyond the scope of this thesis to generalize the algorithms and theory presented for semidefinite programs to general conic programs.

3.2 Basic Assumptions

Throughout this thesis the following assumptions on (SDP) are made:

$(A1)$ $x^* = \operatorname{argmin}\{f(x)|x\in\Omega\}$ exists, where $\Omega = \{x\in\mathbb{R}^n|\mathcal{A}(x) \preccurlyeq 0\}$.

$(A2)$ The Karush-Kuhn-Tucker necessary optimality conditions hold in x^*. That means there exists $U^*\in\mathbb{S}^m$ such that

$$\begin{aligned} f'(x^*) + [\langle U^*, \mathcal{A}_i\rangle]_{i=1}^m &= 0 \\ \operatorname{tr}(U^*\mathcal{A}(x^*)) &= 0 \\ U^* &\succeq 0 \\ \mathcal{A}(x^*) &\preceq 0, \end{aligned} \tag{3.1}$$

where we denote by $\mathcal{A}_i$ the $i{-}th$ partial derivative of $\mathcal{A}$ in x^* $(i=1,\dots,n)$. The second condition is called complementary slackness condition. Later we will prove that the complementary slackness condition can be given in the equivalent form

$$\lambda_i(U^*)\lambda_i(\mathcal{A}(x^*)) = 0 \ \text{ for all } \ i=1,\dots,m,$$

where $\lambda_i(U^*)$ and $\lambda_i(\mathcal{A}(x^*)),\ i=1,\dots,m$ denote the increasingly ordered eigenvalues of U^* and $\mathcal{A}(x^*)$, respectively. If in each of the above equations one factor is non-zero the complementary slackness condition is said to be strict. Throughout this thesis we assume the strict complementary slackness condition to hold.

$(A3)$ The nondegeneracy condition holds. This means that if for $1\le r<m$ the vectors $s_{m-r+1},\dots,s_m\in\mathbb{R}^m$ form a basis of the null space of the matrix $\mathcal{A}(x^*)$, then the following set of n-dimensional vectors is linearly independent:

$$v_{i,j} = (s_i^\top\mathcal{A}_1 s_j,\dots,s_i^\top\mathcal{A}_n s_j)^\top, \qquad m-r+1\le i\le\ j\le m.$$

The nondegeneracy condition is a well known constraint qualification for the problem (SDP) and implies that the matrix U^* is unique (see, for example, [17]).

$(A4)$ Define $E_0 = (s_{m-r+1}, \ldots, s_m)$, where $s_{m-r+1}, \ldots, s_m$ are the vectors introduced in assumption $(A3)$. Then the *cone of critical directions* at x^* is defined as

$$\mathcal{C}(x^*) = \left\{ h \in \mathbb{R}^n : \sum_{i=1}^{n} h_i E_0^\top \mathcal{A}_i E_0 \preceq 0, f'(x^*)^\top h = 0 \right\}.$$

Now we assume the following second order sufficient optimality condition to hold at (x^*, U^*): For all $h \in \mathcal{C}(x^*)$ with $h \neq 0$ the inequality

$$h^\top \left(L''_{xx}(x^*, U^*) + H(x^*, U^*) \right) h > 0,$$

is satisfied, where L is the classical Lagrangian for (SDP) defined as

$$L(x, U) = f(x) + \langle U, \mathcal{A}(x) \rangle ,$$

$H(x^*, U^*)$ is defined by

$$H(x^*, U^*)_{i,j} = -2 \left\langle U^*, \mathcal{A}_i [\mathcal{A}(x^*)]^\dagger \mathcal{A}_j \right\rangle \tag{3.2}$$

(see, for example, [17, p. 490]) and $[\mathcal{A}(x^*)]^\dagger$ is the Moore-Penrose inverse of $\mathcal{A}(x^*)$.

(A5) Define

$$\Omega_p = \{ x \in \mathbb{R}^n | \mathcal{A}(x) \preceq bpI_m \} ,$$

where b is a positive constant and p is a positive real value, which will play the role of a penalty parameter later. Then we assume the following growth condition to hold:

$$\exists \pi > 0 \text{ and } \tau > 0 \text{ such that } \max \{ \|\mathcal{A}(x)\| \mid x \in \Omega_\pi \} \leq \tau, \tag{3.3}$$

It is clear that the existence of such a π implies the validity of (3.3) for any $0 \leq \bar{\pi} < \pi$.

Later we will make use of the following lemma.

Lemma 3.1 *In case the strict complementarity condition holds the cone of critical directions simplifies to*

$$C(x^*) = \left\{ h \in \mathbb{R}^n : \sum_{i=1}^{n} h_i E_0^\top \mathcal{A}_i E_0 = 0 \right\}.$$

Proof . See, for example, [17]. □

Chapter 4

A Class of Penalty Functions

4.1 Idea and Definition

Using the concept of primary matrix functions introduced in Chapter 2 and recalling that the negative semidefiniteness of a matrix is equivalent to the non-positivity of its eigenvalues, the following idea is easy to understand: Let φ be a real-valued function defined on an interval $(a, b) \subseteq \mathbb{R}$, which penalizes positive arguments. Then the corresponding primary matrix function penalizes any matrix A violating the constraint $A \preceq 0$, since φ penalizes each positive eigenvalue of A. This gives rise to the following definition:

Definition 4.1 *Let $\varphi : (-\infty, b) \to \mathbb{R}$ be a function with the following properties:*

(φ_0)	*φ strictly convex, strictly monotone increasing and twice continuously differentiable*,
(φ_1)	$\mathrm{dom}\varphi = (-\infty, b)$ *with* $0 < b \leq \infty$,
(φ_2)	$\varphi(0) = 0$,
(φ_3)	$\varphi'(0) = 1$,
(φ_4)	$\exists\, C_{\varphi,-\infty}$ *such that* $\varphi(t) \geq C_{\varphi,-\infty}$ *for any* $t < 0$,
(φ_5)	$\exists\, C_{\varphi',\sigma}$ *such that* $\varphi'(\sigma/p) \leq pC_{\varphi',\sigma}$ *for any* $\sigma < 0$ *and* $p > 0$,
(φ_6)	$\exists\, C_{\varphi'',\sigma}$ *such that* $\varphi''(\sigma/p) \leq p^2 C_{\varphi'',\sigma}$ *for any* $\sigma < 0$ *and* $p > 0$,
(φ_7)	$\lim_{t \to b} \varphi'(t) = \infty$, $\lim_{t \to -\infty} \varphi'(t) = 0$, φ' *convex*,
(φ_8)	*φ is operator monotone on* $(-\infty, b)$,
(φ_8)	*φ is operator convex on* $(-\infty, b)$.

Define further $\varphi_p(t) = p\varphi(t/p)$ *for all* $t \in$ dom $\varphi_p = (-\infty, bp)$*, where p is a positive* penalty parameter. *Then a* matrix penalty function $\Phi_p : \mathbb{S}^m \to \mathbb{S}^m$ *is defined as*

$$\Phi_p : \mathcal{A}(x) \longmapsto S(x) \begin{pmatrix} \varphi_p\left(\lambda_1(x)\right) & 0 & \dots & 0 \\ 0 & \varphi_p\left(\lambda_2(x)\right) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \varphi_p\left(\lambda_m(x)\right) \end{pmatrix} S(x)^\top ,$$

where $\mathcal{A}(x), S(x), \lambda_i(x),\ i = 1, \dots, m$, are defined as in Definition 2.1.

Remark . Using the matrix penalty function Φ_p we are able to rewrite our initial problem (SDP) as

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) && (\text{SDP}_{\Phi_p}) \\ &\text{s.t. } \Phi_p(\mathcal{A}(x)) \preccurlyeq 0 . \end{aligned}$$

If we write down the Lagrangian for problem (SDP$_{\Phi_p}$) we obtain the function

$$L_{\Phi_p}(x, U) = f(x) + \langle U, \Phi_p(\mathcal{A}(x)) \rangle ,$$

which will be further considered in Section 5.

Remark . Let us briefly discuss the role of assumptions (φ_8) and (φ_9): As we will see later (compare Section 6) assumptions (φ_8) and (φ_9) are not used in the proofs of the main theoretical results presented in this thesis. Nevertheless they are important for the following reason: Φ_p is a convex and monotone matrix function for all $m \in \mathbb{N}$ if and only if assumptions (φ_8) and (φ_9) hold. Using this fact it is easy to see that for convex f and $\mathcal{A}$ assumptions (φ_8) and (φ_9) guarantee that problem (SDP$_{\Phi_p}$) is convex. In Section 5 we will see that in this case also the function L_{Φ_p} is convex. On the contrary, if assumptions (φ_8) and (φ_9) do not hold, problem (SDP$_{\Phi_p}$) may be non-convex even a for linear mapping $\mathcal{A}$. As a consequence it may happen that the augmented Lagrangian L_{Φ_p} is also non-convex for certain $U \succ 0$. After reading Section 6 we will understand that this would mean that we have to solve a series of potentially non-convex optimization problems in order to solve a problem, which is originally convex. This effect is avoided by the inclusion of assumptions (φ_8) and (φ_9) in Definition 4.1.

Remark . Replacing (φ_4) by the weaker assumption

$$(\varphi_4') \qquad \lim_{p \to 0} p\varphi(\frac{\sigma}{p}) \to 0 \text{ for any } \sigma < 0$$

enables us to use a wider class of penalty functions, as we will demonstrate below. Throughout this thesis we assume (φ_4) to hold and explain, how our results have to be modified, if the weaker condition (φ_4') is used instead.

4.2 Examples

The goal of this section is to present a collection of typical penalty– and barrier–type functions from the nonlinear programming literature (see, for example, [68], [69], [7] and [80]) and discuss their suitability with respect to the assumptions required in definition 4.1:

- The *logarithmic* penalty function is defined as

$$\varphi_{\log}(t) = -\log(1-t).$$

Looking at the derivative formulas

$$\varphi'_{\log}(t) = \frac{1}{1-t} \text{ and } \varphi''_{\log}(t) = \frac{1}{(1-t)^2}$$

it is obvious to see that (φ_0) to (φ_3), (φ'_4) and (φ_5) to (φ_7) are satisfied. The validity of (φ_8) and (φ_9) is proved, for example, in [75].

- The *hyperbolic* penalty function is defined as

$$\varphi_{\text{hyp}}(t) = \frac{1}{1-t} - 1.$$

Here the derivatives are given by

$$\varphi'_{\text{hyp}}(t) = \frac{1}{(1-t)^2} \text{ and } \varphi''_{\text{hyp}}(t) = 2\frac{1}{(1-t)^3}$$

and (φ_0) to (φ_7) are verified easily. Properties (φ_8) and (φ_9) have been established in [75]. Note that the functions $\varphi_{\log}$ and φ_{hyp} have been introduced in [68] as so called *modified barrier functions*.

- The *parabolic* penalty function is defined as

$$\varphi_{\text{par}}(t) = -2\sqrt{1-t} + 2.$$

The first and second derivatives of this function are

$$\varphi'_{\text{par}}(t) = (1-t)^{-\frac{1}{2}} \text{ and } \varphi''_{\text{par}}(t) = \frac{1}{2}(1-t)^{-\frac{3}{2}}.$$

In analogy to the hyperbolic penalty function one can show that (φ_0) to (φ_3), (φ'_4) and (φ_5) to (φ_9) are satisfied. I slight disadvantage of this function is that φ_{par} does not tend to infinity at the right boundary of its domain.

- The *exponential* penalty function is defined as

$$\varphi_{\exp}(t) = e^t - 1.$$

Again the first and second derivative formulas

$$\varphi'_{\text{par}}(t) = \varphi''_{\text{par}}(t) = e^t$$

imply the validity of properties (φ_0) to (φ_7). Unfortunately the corresponding matrix function is neither monotone nor convex in the sense of assumptions (φ_8) and (φ_9) (see, for example, [45]). Note that in [30] the exponential penalty function is used to construct an alternative algorithm for the solution of linear semidefinite programs.

- Another interesting class of functions was introduced in [6] and further studied in [7], [64] and [22]. The so called penalty/barrier functions are constructed from analytic penalty functions φ in the following way:

$$\widehat{\varphi}(t) = \begin{cases} \varphi(t) & \text{for } t \le r, \\ at^2 + bt + c & \text{for } t \ge r, \end{cases} \tag{4.1}$$

  where $r \in [-1, 1]$ and a, b and c are real numbers, which can be adjusted so that $\widehat{\varphi}$ is at least twice continuously differentiable in r. Obviously in this case, the right branch of $\widehat{\varphi}$ is nothing else as the quadratic C^2-extrapolation of the left branch φ. Two important advantages of such a choice are that

  – $\operatorname{dom}\widehat{\varphi} = (-\infty, \infty)$, which simplifies penalty parameter initializations and update strategies in penalty type methods,

  – the second derivative of $\widehat{\varphi}$ is bounded, which is a very useful property, when Newton type methods are applied.

  Numerical studies in nonlinear programming indicated that optimization methods based on penalty/barrier functions as defined in (4.1) are often superior to methods based on classical barrier or modified barrier functions. Moreover it is quite easy to see that any penalty/barrier function $\hat{\varphi}$ constructed from a function φ according to (4.1) satisfies the properties (φ_0) to (φ_7) if and only if the same properties hold for φ. However it can be shown that the matrix function corresponding to the right branch of the function $\widehat{\varphi}$ is – as a quadratic matrix function – non-monotone in the sense of definition 2.4 (compare Example 2.2).

An interesting fact is that all valid candidates of penalty functions we have found so far have two common properties:

- They are analytic on their full domain and
- they have a pole on the positive real half axis.

The following Theorem is a direct consequence of results presented in C. Loewner's paper on monotone matrix functions [29] and help to interpret these observations:

Theorem 4.1 *Let* $(a, b) \subseteq \mathbb{R}$.

a) If Φ is a monotone matrix function on $H^m(a, b)$, then φ has at least $2m - 3$ continuous derivatives.

b) φ is operator monotone on $H^m(a, b)$ if and only if φ is the restriction of an analytic function defined on the upper complex half plane to the real interval (a, b) . Furthermore each such function can be represented as

$$f(z) = \alpha z + \beta + \int_{-\infty}^{\infty} \left[\frac{1}{u - z} - \frac{u}{u^2 + 1} \right] d\mu(u),$$

where $\alpha \ge 0, \beta \in \mathbb{R}$ and $d\mu$ is a Borel measure that has no mass in (a, b).

An immediate consequence of Theorem 4.1 a) is that any operator monotone function φ must be analytic. Moreover if φ is such a function, if $(a, b) = \mathbb{R}$ and $\varphi(x) < \infty$ for all $x \in \mathbb{R}$, we conclude from part b) of Theorem 4.1 that φ is linear and therefore violates several assumptions required by Definition 4.1. In other words, it is not possible to find a valid penalty function in the sense of Definition 4.1 with the full real line as its domain. From this point of view the functions φ_{hyp}, $\varphi_{\log}$ and φ_{par} are "optimal" choices.

Remark . Any appropriately scaled version and any convex combination of the functions φ_{hyp}, $\varphi_{\log}$ and φ_{par} results in a valid penalty function.

Remark . The functions φ_{hyp}, $\varphi_{\log}$ and φ_{par} are special cases of the following formula:

$$\varphi_\alpha(t) = \delta_\alpha \int_0^t (1-s)^{\alpha-1} ds + \gamma_\alpha,$$

where $-1 \leq \alpha < 1$ and δ_α and γ_α are uniquely determined by assumptions (φ_2) and (φ_3). Depending on α we obtain the following classes of penalty functions:

$$\begin{aligned} 0 < \alpha < 1 &: \text{parabolic penalty functions,} \\ \alpha = 0 &: \text{logarithmic penalty function,} \\ -1 \leq \alpha < 0 &: \text{hyperbolic penalty functions.} \end{aligned}$$

So far it can be proven that all members of the family of parabolic and at least some members of the family of hyperbolic penalty functions have the same properties as the representatives φ_{hyp} and φ_{par} presented earlier in this section (see, for example, [45]).

Chapter 5

A Class of Augmented Lagrangians

At the beginning of this chapter we define a class of augmented Lagrangians – the heart of our algorithm. The definition is based on the class of matrix penalty functions introduced in the preceding chapter.

We start with some useful notations, which will be used throughout Chapter 5 and 6. Let $\lambda_1 \leq \ldots \leq \lambda_{m-r} < \lambda_{m-r+1} = \ldots = \lambda_m = 0$ denote the ordered eigenvalues of $\mathcal{A}(x^*)$ and $s_1, \ldots, s_m \in \mathbb{R}^m$ the corresponding eigenvectors. Further define

- $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_m) \in \mathbb{S}^m, S = (s_1, \ldots, s_m) \in \mathbb{M}^{m,m}$,
- $\Lambda_0 = \operatorname{diag}(\lambda_{m-r+1}, \ldots, \lambda_m) \in \mathbb{S}^r, \Lambda_\perp = \operatorname{diag}(\lambda_1, \ldots, \lambda_{m-r}) \in \mathbb{S}^{m-r}$,
- $E_0 = (s_{m-r+1}, \ldots, s_m) \in \mathbb{M}^{m,r}$, $E_\perp = (s_1, \ldots, s_{m-r}) \in \mathbb{M}^{m,m-r}$ and
- $P_0 = E_0 E_0^\top \in \mathbb{S}^m, P_\perp = E_\perp E_\perp^\top \in \mathbb{S}^m$.

Note that

- the columns of E_0 form a basis of $\operatorname{Ker}(\mathcal{A}(x^*))$,
- the columns of $E_\perp$ form a basis of $\operatorname{Im}(\mathcal{A}(x^*))$,
- $\mathcal{A}(x^*) = S\Lambda S^\top$,
- $P_0 + P_\perp = SD_0S^\top + SD_\perp S^\top = I_m$, where

$$\begin{aligned} D_0 &= \operatorname{diag}(\underbrace{0, \ldots, 0}_{m-r}, \underbrace{1, \ldots, 1}_{r}), \\ D_\perp &= \operatorname{diag}(\underbrace{1, \ldots, 1}_{m-r}, \underbrace{0, \ldots, 0}_{r}). \end{aligned}$$

- using the notation of Definition 2.7, we see that

$$P_0 = P_{\mu(x^*)}(x^*) \text{ and } P_\perp = \sum_{k=1}^{\mu(x^*)-1} P_k(x^*),$$

  where $P_k(x^*)$ for $k \in \{1, \ldots, \mu(x^*)\}$ are the Frobenius covariance matrices for the $\mu(x^*)$ distinct eigenvalues of $\mathcal{A}(x^*)$.

Next we define a class of augmented Lagrangians for the problem (SDP).

Definition 5.1 *Given a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, a twice differential matrix operator $\mathcal{A} : \mathbb{R}^n \to \mathbb{S}^m$ and a penalty function $\Phi_p : \mathbb{S}^m \to \mathbb{S}^m$ as given in Section 4, we define the following class of Augmented Lagrangians:*

$$\begin{array}{rcl} F_\Phi : \mathbb{R}^n \times \mathbb{S}^m \times \mathbb{R} & \to & \mathbb{R} \\ (x, U, p) & \mapsto & f(x) + \langle U, \Phi_p(\mathcal{A}(x)) \rangle . \end{array}$$

The following Theorem summarizes the most important properties of F_Φ:

Theorem 5.1 *The augmented Lagrangian $F_\Phi(x, U, p)$ has the following properties:*

a) $F_\Phi(x^*, U^*, p) = f(x^*)$.

b) $F'_{\Phi,x}(x^*, U^*, p) = f'(x^*) + [\langle U^*, \mathcal{A}_i \rangle]_{i=1}^m = 0$.

c) $F''_{\Phi,xx}(x^*, U^*, p) = L''_{xx}(x^*, U^*) + H_p(x^*, U^*) + p^{-1}M$, *where* $H_p(x^*, U^*)$ *and* M *are symmetric matrices and* $H_p(x^*, U^*) \to H(x^*, U^*)$ *for* $p \to 0$, $\mathrm{Ker}(M) = \mathcal{C}(x^*)$ *and* $y^\top M y > 0$ *for all* $y \notin \mathcal{C}(x^*)$.

If f and $\mathcal{A}$ are convex, then

d) $F_\Phi(x, U, p)$ *is convex in* x *for all* $x \in \Omega_p$.

Remark . Properties 5.1 a), b) and d) are shared by the classical Lagrangian function L associated with problem (SDP) and our augmented choice. However, as we will see later, property 5.1 c) is the reason for some advantages of F over L.

For simplicity of notation we let $F = F_\Phi$ in the remainder of this section. For the proof of Theorem 5.1 we make use of the following Lemmas.

Lemma 5.2 *(von Neumann – Theobald) Let $A, B \in \mathbb{S}^m$ be given. Denote the ordered eigenvalues of A, B by $\lambda_1(A) \leq \cdots \leq \lambda_m(A)$ and $\lambda_1(B) \leq \cdots \leq \lambda_m(B)$. Then*

$$\mathrm{tr}(AB) \leq \sum_{i=1}^m \lambda_i(A)\lambda_i(B),$$

where equality holds if and only if A and B have a simultaneous ordered spectral decomposition.

Proof . See, for example, [77]. □

Lemma 5.3 *Let $A \in \mathbb{S}^m_+$ and $B \in \mathbb{S}^m_-$ be given. Denote the ordered eigenvalues of A, B by $\lambda_1(A) \leq \cdots \leq \lambda_m(A)$ and $\lambda_1(B) \leq \cdots \leq \lambda_m(B)$. Then the following conditions are equivalent:*

a) $\langle A, B \rangle = 0$.

b) $AB = 0$.

c) *A and B are simultaneously diagonalizable and* $\lambda_i(A)\lambda_i(B) = 0$ *for all* $i \in \{1, \dots, m\}$.

Proof . Let $A = P^2$ and $B = -Q^2$ with $P, Q \in \mathbb{S}^m$ (existence of P and Q follows from the semidefiniteness of A and B).

<u>"a) ⇒ b)"</u> We have

$$\|PQ\|^2 = \langle PQ, PQ \rangle = \mathrm{tr}(QP^2Q) = \mathrm{tr}(Q^2P^2) = \left\langle Q^2, P^2 \right\rangle = -\langle B, A \rangle = 0.$$

Consequently $PQ = 0$ and $AB = -P(PQ)Q = 0$.

<u>"b) ⇒ c)"</u> From Lemma 5.2 we get

$$0 = \mathrm{tr}(AB) = \langle A, B \rangle \leq \sum_{i=1}^m \underbrace{\lambda_i(A)}_{\geq 0} \underbrace{\lambda_i(B)}_{\leq 0} \leq 0.$$

Immediately we see $\lambda_i(A)\lambda_i(B) = 0$ for all $i \in \{1. \dots, m\}$. Moreover we have

$$BA = B^\top A^\top = (AB)^\top = 0 = AB$$

and thus A and B are simultaneously diagonalizable.

<u>"c) ⇒ a)"</u> From the fact that A and B are simultaneously diagonalizable follows the existence of an orthonormal matrix S and diagonal matrices Λ_A, Λ_B such that

$$A = S\Lambda_A S^\top \text{ and } B = S\Lambda_B S^\top.$$

Thus we obtain

$$\langle A, B \rangle = \mathrm{tr}\left(S\Lambda_A\Lambda_B S^\top\right) = \sum_{i=1}^m \lambda_i(A)\lambda_i(B) = 0$$

and we have shown $\langle A, B \rangle = 0$. □

Lemma 5.4

a) $P_\perp \mathcal{A}(x^*) P_\perp = \mathcal{A}(x^*)$, $P_0 \mathcal{A}(x^*) P_0 = P_\perp \mathcal{A}(x^*) P_0 = P_0 \mathcal{A}(x^*) P_\perp = \mathbb{O}_m$.

b) $P_\perp U^* P_\perp = P_\perp U^* P_0 = P_0 U^* P_\perp = \mathbb{O}_m$, $P_0 U^* P_0 = U^*$.

Proof .

a) Using the spectral decomposition of $\mathcal{A}(x^*)$, we obtain

$$\begin{aligned} P_\perp \mathcal{A}(x^*) P_\perp &= SD_\perp S^\top S\Lambda S^\top SD_\perp S^\top = SD_\perp \Lambda D_\perp S^\top \\ &= S\Lambda S^\top = \mathcal{A}(x^*). \end{aligned}$$

Exemplary we show

$$\begin{aligned} P_0 \mathcal{A}(x^*) P_0 &= SD_0 S^\top S\Lambda S^\top SD_0 S = SD_0 \Lambda D_0 S^\top \\ &= S\mathbb{O}_m S^\top = \mathbb{O}_m. \end{aligned}$$

b) From the first order optimality conditions, Lemma 5.2 and Lemma 5.3 follows $U^* = S\Lambda_{U^*} S^\top$, where $\Lambda_{U^*} = \mathrm{diag}(\lambda_1(U^*), \dots, \lambda_m(U^*))$ and $0 = \lambda_1(U^*) = \dots = \lambda_{m-r}(U^*) \leq \lambda_{m-r+1}(U^*) \leq \dots \lambda_m(U^*)$. Now we obtain

$$\begin{aligned} P_0 U^* P_0 &= SD_0 S^\top S\Lambda_{U^*} S^\top SD_0 S^\top = SD_0 \Lambda_{U^*} D_0 S^\top \\ &= S\Lambda_{U^*} S^\top = U^*. \end{aligned}$$

Exemplary we show

$$\begin{aligned} P_\perp U^* P_\perp &= SD_\perp S^\top S\Lambda_{U^*} S^\top SD_\perp S = SD_\perp \Lambda_{U^*} D_\perp S^\top \\ &= S\mathbb{O}_m S^\top = \mathbb{O}_m. \end{aligned}$$

□

Now we are able to prove Theorem 5.1:

a) From the first order optimality conditions (3.1) and Lemma 5.3 we obtain

$$\lambda_i(U^*) = 0 \text{ for all } i \in \{1, \dots, m-r\}.$$

By definition of Φ_p we have

$$\varphi_p\left(\lambda_i\left(\mathcal{A}(x^*)\right)\right) = 0 \text{ for all } i \in \{m-r+1, \dots, m\}.$$

Now Lemma 5.3 implies $\langle U^*, \Phi_p\left(\mathcal{A}(x^*)\right)\rangle = 0$ and $F(x^*, U^*, p) = f(x^*)$.

b) The first equality follows from the fact that

$$U^* = D\Phi_p\left(\mathcal{A}(x^*)\right)[U^*], \tag{5.1}$$

which we will prove below. The second equality follows from (3.1). The following equation completes this part of the proof:

$$\begin{aligned}
& D\Phi_p\left(\mathcal{A}(x^*)\right)[U^*] \\
= \; & \sum_{k,l=1}^{\mu(x^*)} \Delta\varphi_p(\lambda_k,\lambda_l) P_k(x^*)\left(U^*\right) P_l(x^*) \\
= \; & \sum_{k,l=1}^{\mu(x^*)} \Delta\varphi_p(\lambda_k,\lambda_l) P_k(x^*)\left(P_0 U^* P_0\right) P_l(x^*) \\
= \; & \sum_{k,l=1}^{\mu(x^*)} \Delta\varphi_p(\lambda_k,\lambda_l) P_k(x^*) P_{\mu(x^*)}(x^*) U^* P_{\mu(x^*)}(x^*) P_l(x^*) \\
= \; & \Delta\varphi_p(\lambda_{\mu(x^*)},\lambda_{\mu(x^*)}) P_{\mu(x^*)}(x^*) U^* P_{\mu(x^*)}(x^*) \\
= \; & \varphi_p'(0) P_0 U^* P_0 = U^*.
\end{aligned}$$

c) Taking into account Theorem 2.3 and Lemma 5.4 and denoting

$$\mathcal{A}_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j}\mathcal{A}(x^*) \text{ for all } i,j = 1,\dots,n,$$

we have

$$\begin{aligned}
& F''_{xx}(x^*,U^*,p) \\
= \; & f''(x^*) + \left[\left\langle P_0 U^* P_0, \frac{\partial^2}{\partial x_i \partial x_j}\Phi_p\left(\mathcal{A}(x^*)\right)\right\rangle\right]_{i,j=1}^n \\
= \; & f''(x^*) + \Bigg[\left\langle U^*, P_0\left(\sum_{k,l=1}^{\mu(x^*)} \Delta\varphi_p(\lambda_k,\lambda_l) P_k(x^*) \mathcal{A}_{i,j} P_l(x^*)\right) P_0\right\rangle + \\
& \left\langle U^*, \sum_{k,l,s=1}^{\mu(x^*)} P_0 \Delta^2\varphi_p(\lambda_k,\lambda_l,\lambda_s) P_k(x^*) \mathcal{A}_i P_l(x^*) \mathcal{A}_j P_s(x^*) P_0\right\rangle + \\
& \left\langle U^*, \sum_{k,l,s=1}^{\mu(x^*)} P_0 \Delta^2\varphi_p(\lambda_k,\lambda_l,\lambda_s) P_k(x^*) \mathcal{A}_j P_l(x^*) \mathcal{A}_i P_s(x^*) P_0\right\rangle\Bigg]_{i,j=1}^n \\
= \; & f''(x^*) + \\
& \left[\left\langle U^*, \Delta\varphi_p(\lambda_{\mu(x^*)},\lambda_{\mu(x^*)}) P_{\mu(x^*)}(x^*) \mathcal{A}_{i,j} P_{\mu(x^*)}(x^*)\right\rangle\right]_{i,j=1}^n + \\
& \left[\left\langle U^*, P_{\mu(x^*)}(x^*) N_{i,j} P_{\mu(x^*)}(x^*)\right\rangle\right]_{i,j=1}^n + \\
& \left[\left\langle U^*, P_{\mu(x^*)}(x^*) N_{j,i} P_{\mu(x^*)}(x^*)\right\rangle\right]_{i,j=1}^n \\
= \; & \underbrace{f''(x^*) + \left[\left\langle U^*, P_0 \varphi_p'(0) \mathcal{A}_{i,j} P_0\right\rangle\right]_{i,j=1}^n}_{L''_{xx}(x^*,U^*)} + 2\left[\langle U^*, N_{i,j}\rangle\right]_{i,j=1}^n,
\end{aligned}$$

where

$$N_{i,j} = \mathcal{A}_i \left(\sum_{k=1}^{\mu(x^*)} \Delta^2 \varphi_p(\lambda_k, 0, 0) P_k(x^*) \right) \mathcal{A}_j \text{ for all } i, j = 1, \dots, n.$$

Now we calculate $\Delta^2 \varphi_p(\lambda_k, 0, 0)$. We consider two cases:

$\underline{k = \mu(x^*)}$:

$$\Delta^2 \varphi_p(\lambda_k, 0, 0) = \Delta^2 \varphi_p(0, 0, 0) = p^{-1} \varphi''(0) \to \infty \text{ for } p \to 0.$$

$\underline{k < \mu(x^*)}$:

$$\Delta^2 \varphi_p(\lambda_k, 0, 0) = \frac{\frac{p\varphi(\lambda_k/p)}{\lambda_k} - 1}{\lambda_k} \to -\frac{1}{\lambda_k} \text{ for } p \to 0. \tag{5.2}$$

The latter limit follows from $\lambda_k < 0$ and the properties of φ. Defining

$$H_p(x^*, U^*) = 2\big[\big\langle U^*, \mathcal{A}_i \big(\textstyle\sum_{k=1}^{\mu(x^*)-1} \Delta^2 \varphi_p(\lambda_k, 0, 0) P_k(x^*) \big) \mathcal{A}_j \big\rangle\big]_{i,j=1}^m \tag{5.3}$$

and $M = 2\left[\langle U^*, \mathcal{A}_i \varphi''(0) P_0(x^*) \mathcal{A}_j \rangle\right]_{i,j=1}^m$ we see that

$$F''_{xx}(x^*, U^*, p) = L''_{xx}(x^*, U^*) + H_p(x^*, U^*) + p^{-1} M,$$

where

$$H_p(x^*, U^*) \to -2 \left[\left\langle U^*, \mathcal{A}_i \left(\sum_{k=1}^{m-r} \frac{1}{\lambda_k} s_k s_k^\top \right) \mathcal{A}_j \right\rangle \right]_{i,j=1}^m = H(x^*, U^*)$$

for $p \to 0$. To complete the proof, we will show next that

$$\mathrm{Ker}(M) = \mathcal{C}(x^*).$$

This can be seen from

$$\begin{aligned}
M &= 2\varphi''(0) \left[\langle U^*, \mathcal{A}_i P_0(x^*) \mathcal{A}_j \rangle\right]_{i,j=1}^m \\
&= 2\varphi''(0) \left[\left\langle U^*, \mathcal{A}_i \left(\sum_{k=m-r+1}^{m} s_k s_k^\top \right) \mathcal{A}_j \right\rangle \right]_{i,j=1}^m \\
&= 2\varphi''(0) \left[\left\langle S \Lambda_{U^*} S^\top, \mathcal{A}_i \left(\sum_{k=m-r+1}^{m} s_k s_k^\top \right) \mathcal{A}_j \right\rangle \right]_{i,j=1}^m \\
&= 2\varphi''(0) \left[\sum_{l=m-r+1}^{m} u_l s_l^\top \mathcal{A}_i \left(\sum_{k=m-r+1}^{m} s_k s_k^\top \right) \mathcal{A}_j s_l \right]_{i,j=1}^m \\
&= 2\varphi''(0) \left[\sum_{k,l=m-r+1}^{m} u_l \left(s_l^\top \mathcal{A}_i s_k s_k^\top \mathcal{A}_j s_l \right) \right]_{i,j=1}^m \\
&= 2\varphi''(0) B U_r^* B^\top,
\end{aligned}$$

where $U_r^* = \operatorname{diag}(u_{m-r+1}, \ldots, u_m, \ldots, u_{m-r+1}, \ldots, u_m) \in \mathbb{S}^{r^2}$ and the i-th row of $B \in \mathbb{M}^{n,r^2}$ is defined by

$$\begin{aligned} b_i = (\; & s_{m-r+1}^\top \mathcal{A}_i s_{m-r+1}, \ldots, s_m^\top \mathcal{A}_i s_{m-r+1}, \ldots, \\ & s_{m-r+1}^\top \mathcal{A}_i s_m, \ldots, s_m^\top \mathcal{A}_i s_m \;)^\top. \end{aligned}$$

From the strict complementarity condition follows that $U_r^* \in \mathbb{S}_{++}^r$ and by Lemma 3.1 we obtain $\operatorname{Ker}(M) = \mathcal{C}(x^*)$ and the claim that $y^\top My > 0$ for all $y \notin \mathcal{C}(x^*)$.

d) We have to show that $\langle U, \Phi_p(\mathcal{A}(x))\rangle$ is convex for all $p > 0$, $U \succeq 0$ and $x \in \Omega_p = \{x \in \mathbb{R}^n | \mathcal{A}(x) \preceq bpI_m\}$. Given $\lambda \in [0;1]$ and $x, y \in \Omega_p$, the convexity of $\mathcal{A}$ assures

$$\lambda\mathcal{A}(x) + (1-\lambda)\mathcal{A}(y) - \mathcal{A}(\lambda x + (1-\lambda)y) \succeq 0. \tag{5.4}$$

By the monotonicity of Φ_p we get

$$\Phi_p(\lambda\mathcal{A}(x) + (1-\lambda)\mathcal{A}(y)) - \Phi_p(\mathcal{A}(\lambda x + (1-\lambda)y)) \succeq 0.$$

The latter inequality combined with the convexity of Φ_p show

$$\Phi_p(\lambda\mathcal{A}(x)) + (1-\lambda)\Phi_p(\mathcal{A}(y)) - \Phi_p(\mathcal{A}(\lambda x + (1-\lambda)y)) \succeq 0.$$

Since $U \succeq 0$, it follows that

$$\operatorname{tr}\left[U^\top \left[\Phi_p(\lambda\mathcal{A}(x)) + (1-\lambda)\Phi_p(\mathcal{A}(y)) - \Phi_p(\mathcal{A}(\lambda x + (1-\lambda)y))\right]\right] \geq 0.$$

Finally, the latter inequality and the linearity of $\operatorname{tr}(\cdot)$ imply the convexity of $\langle U, \Phi_p(\mathcal{A}(x))\rangle$. □

The following Corollary points out two important advantages of F over the classical Lagrangian L:

Corollary 5.5

a) *There exists $p_0 > 0$ such that $F(x, U^*, p)$ is strongly convex for all $p < p_0$ in a neighborhood of x^*.*

b) *There exist constants $\epsilon > 0$ and $p_0 > 0$ such that*

$$x^* = \operatorname{argmin}\{F(x, U^*, p) | x \in \mathbb{R}, \|x - x^*\| \leq \epsilon\} \; \forall p < p_0.$$

Moreover, if f and $\mathcal{A}$ are convex, then

$$x^* = \operatorname{argmin}\{F(x, U^*, p) | x \in \mathbb{R}^n\} \; \forall p > 0.$$

In other words, if the optimal Lagrangian multiplier U^* is known, problem (SDP) can be solved by solving one smooth optimization problem.

Proof .

a) Below we will show that there exists $\gamma > 0$ such that $y^\top F''(x^*, U^*, p)y > \gamma$ for all $y \in \mathbb{R}$ with $\|y\| = 1$, whenever p is small enough. By Theorem 5.1 we know that
$$F''_{xx}(x^*, U^*, p) = L''_{xx}(x^*, U^*) + H_p(x^*, U^*) + p^{-1}M,$$
where
$$H_p(x^*, U^*) \to H(x^*, U^*) \text{ for } p \to 0 \tag{5.5}$$
and
$$\text{Ker}(M) = \mathcal{C}(x^*),\ y^\top My > 0 \text{ for all } y \notin \mathcal{C}(x^*). \tag{5.6}$$
We consider two cases:

$\underline{y \in \mathcal{C}(x^*)}$: From the second order optimality conditions for (SDP) it follows that there exists $\gamma > 0$ such that
$$y^\top (L''_{xx}(x^*, U^*) + H(x^*, U^*))y > \gamma \text{ for all } y \in \mathcal{C}(x^*) \text{ with } \|y\| = 1. \tag{5.7}$$
Now, (5.5) implies
$$y^\top F''_{xx}(x^*, U^*, p)y = y^\top \left(L''_{xx}(x^*, U^*) + H_p(x^*, U^*)\right) y > \gamma$$
for p small enough.

$\underline{y \notin \mathcal{C}(x^*)}$: From (5.6) we see that
$$y^\top p^{-1}My \to \infty \text{ for } p \to 0.$$
On the other hand
$$y^\top \left(L''_{xx}(x^*, U^*) + H_p(x^*, U^*)\right) y$$
is bounded for all $p \le p_0$ for a $p_0 \in \mathbb{R}$ small enough. Thus we obtain
$$y^\top F''_{xx}(x^*, U^*, p)y > \gamma$$
for p small enough.

b) is a direct consequence of a) and Theorem 5.1. $\square$

Remark . Note that the assertion of Corollary 5.5 a) is generally wrong for the classical Lagrangian L. Even for convex problem data f and $\mathcal{A}$ strong convexity of L may not hold. Moreover it is a well known fact that the characterization in part b) of Corollary 5.5 may fail for the classical Lagrangian (see, for example, [71]). In case of the augmented Lagrangian, Corollary 5.5 a) guarantees that the last assertion of Corollary 5.5 b) holds even for non-convex f and $\mathcal{A}$, provided p is small enough and the optimization is started close enough to x^*

Chapter 6

A Locally Convergent Augmented Lagrangian Algorithm

At the beginning of this chapter we present a basic algorithm for the solution of problem (SDP). Then, in the main part, we deal with the (local) convergence properties of this algorithm.

6.1 Basic Algorithm

On the basis of Definition 5.1 we define the following algorithm:

Algorithm 6.1.1 *Let* $x^0 \in \mathbb{R}^n, U^0 \in \mathbb{S}^m_{++}$ *and* $p^0 > 0$ *be given. Then for* $k = 0, 1, 2, \ldots$ *repeat till a stopping criterium is reached:*

$$\begin{aligned} (i) \qquad & x^{k+1} = \arg\min_{x \in \mathbb{R}^n} F(x, U^k, p^k) \\ (ii) \qquad & U^{k+1} = D\Phi_p\left(\mathcal{A}(x^{k+1})\right)\left[U^k\right] \\ (iii) \qquad & p^{k+1} \leq p^k. \end{aligned}$$

Obviously Algorithm 6.1.1 consists of three steps, an unconstrained minimization step and two update formulas:

- In the first step we calculate the global minimum of F with respect to x, where the multiplier and the penalty parameter are kept constant. Of course, to find the global minimum of a generally non-convex function is a difficult task in practice, and in the worst case one may not be able to solve a single problem of type 6.1.1(i). We will return to this point at the end of Section 6, where we will show, how Algorithm 6.1.1 can be adapted for local minima.

- The second step of Algorithm 6.1.1 is the multiplier update formula. The multiplier update formula can be interpreted as the directional derivative of Φ_p in direction of the current multiplier, evaluated at $\mathcal{A}(x^{k+1})$.
- The third step describes the penalty parameter update formula, which is given in the most general form here. A possible update scheme will be developed in Section 9.1.

For a more detailed discussion on the particular steps of Algorithm 6.1.1 we refer to Section 9.

6.2 Local Convergence Properties

We start with the analysis of the local convergence properties of Algorithm 6.1.1. In particular we are interested in the behavior of the algorithm, when it is started with a penalty parameter p_0, which is small enough and an initial multiplier matrix U_0, which is close enough to U^*. These relations are formalized by the following definition:

Definition 6.1 *Let $\lambda_1(U^*), \ldots, \lambda_m(U^*)$ denote the eigenvalues of U^* in increasing order. Let further $0 < \epsilon < \lambda_{m-r+1}(U^*)$ and $\lambda_m(U^*) < \Theta$ be given. Then we define*

$$\begin{aligned}\mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta) &= \Big\{(U,p) \in \mathbb{S}^m_+ \times \mathbb{R} : \|U - U^*\| \le \delta p^{-1}, p < p_0\Big\} \cap \\ &\quad \Big\{(U,p) \in \mathbb{S}^m_+ \times \mathbb{R} : \|U\| \le \Theta\Big\} \cap \\ &\quad \Big\{(U,p) \in \mathbb{S}^m_+ \times \mathbb{R} : s_i^\top U s_i \ge \epsilon, i \in I_{\text{act}}\Big\},\end{aligned}$$

where $I_{\text{act}} = \{m - r + 1, \ldots, m\}$.

Motivation . The set $\mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ is constructed as an intersection of three sets. Below we give an interpretation for each of the sets:

i) The first set allows for any initial multiplier $U \in \mathbb{S}^m_+$ provided the parameter p is small enough. Thus we can start with multipliers arbitrarily far from the optimal multiplier U^* for the price of a small penalty parameter.

ii) The norm of the initial multiplier U should be restricted.

iii) The diagonal entries of the projection of the initial multiplier onto the nullspace of $\mathcal{A}(x^*)$ should be bounded away from zero.

Next we want to give a short overview about the goals we want to achieve in the remainder of this section:

(G1) First we show that Algorithm 6.1.1 is *well defined*. To this end we prove that for each pair $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ with appropriately chosen parameters $p_0, \delta, \epsilon, \Theta$, there exists a unique vector

$$\hat{x} = \hat{x}(U,p) = \operatorname{argmin}\{F(x,U,p) | x \in \mathbb{R}^n\}$$

such that $F'_x(\hat{x}, U, p) = 0$.

(G2) Then we prove the estimate

$$\max\left\{\|\hat{x}-x^*\|, \|\widehat{U}-U^*\|\right\} \leq Cp\,\|U-U^*\| \tag{6.1}$$

for the pair $\hat{x}$ and

$$\widehat{U} = \widehat{U}(U,p) = D\Phi_p\left(\mathcal{A}(\hat{x}(U,p))\right)[U],$$

where C is a constant which is independent of p. In other words we demonstrate that for appropriately chosen parameters $p_0, \delta, \epsilon, \Theta$ and for $p < p_0$ small enough the mapping $\widehat{U} : \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta) \to \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ is a *contraction* with fixed point U^*. Using this fact we conclude that Algorithm 6.1.1 converges and that the rate of convergence for the primal and dual iterates is determined by the contraction constant pC.

(G3) Finally we verify that the function $F(x, U, p)$ is strongly convex with respect to x in a neighborhood of $\hat{x}(U,p)$ for all $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$. This property guarantees fast convergence of Newton-type methods applied to the unconstrained minimization problems in step (i) of Algorithm 6.1.1.

As mentioned already earlier in this thesis our method is a generalization of Polyak's modified barrier function method, introduced in [68]. In the main part of this section we basically follow the ideas presented in [68] in order to prove assertions $(G1)$ to $(G3)$. However it will turn out during the proof that the generalization to nonlinear semidefinite programs is not always straight forward.

Before we start with the verification of (G1) to (G3) some preliminaries are needed. We start with the projection matrices P_0 and $P_\perp$ introduced at the beginning of Chapter 5 which are used to decompose the multiplier matrix $\widehat{U} = \widehat{U}(U,p)$ in the following way:

$$\widehat{U} = P_0\widehat{U}P_0 + \left(P_\perp\widehat{U}P_0 + P_0\widehat{U}P_\perp + P_\perp\widehat{U}P_\perp\right). \tag{6.2}$$

Note that the matrix $P_0\widehat{U}P_0$ is the orthogonal projection of $\widehat{U}$ onto the null space of $\mathcal{A}(x^*)$. Motivated by this fact we define the matrices

$$\begin{aligned}\widehat{U}_{\text{act}} &= P_0\widehat{U}P_0,\\ \widehat{U}_{\text{inact}} &= P_\perp\widehat{U}P_0 + P_0\widehat{U}P_\perp + P_\perp\widehat{U}P_\perp.\end{aligned}$$

Next we introduce a variable transformation $T = p(U - U^*)$ and matrices

$$\widehat{U}_0 = E_0^\top\widehat{U}E_0 \in \mathbb{S}^r,\ U_0^* = E_0^\top U^* E_0 \in \mathbb{S}^r.$$

Furthermore we define the mapping $\widehat{U}_\perp : \mathbb{R}^n \times \mathbb{S}^m \times \mathbb{R} \to \mathbb{S}^m$ by

$$\begin{aligned}\widehat{U}_\perp(x,T,p) &= P_\perp D\Phi_p\left(\mathcal{A}(x)\right)\left[p^{-1}T + U^*\right]P_\perp +\\ &\quad P_0 D\Phi_p\left(\mathcal{A}(x)\right)\left[p^{-1}T + U^*\right]P_\perp +\\ &\quad P_\perp D\Phi_p\left(\mathcal{A}(x)\right)\left[p^{-1}T + U^*\right]P_0.\end{aligned} \tag{6.3}$$

Using these definitions we observe

$$\widehat{U}_{\text{act}} = E_0 \widehat{U}_0 E_0^\top \quad \text{and} \tag{6.4}$$

$$\widehat{U}_{\text{inact}} = \widehat{U}_\perp(\hat{x}, p(U - U^*), p). \tag{6.5}$$

We will further make use of the mapping $h : \mathbb{R}^n \times \mathbb{S}^m \times \mathbb{R} \to \mathbb{R}^n$ defined by

$$h(x, T, p) = \left[\left\langle \widehat{U}_\perp(x, T, p), \mathcal{A}_i'(x) \right\rangle \right]_{i=1}^n . \tag{6.6}$$

Next we define local neighborhoods

- $S(\mathbb{O}_{\mathbb{S}^m}, \delta) = \{T : \|T\| \leq \delta\}$,
- $S(U_0^*, \epsilon_0) = \{V \in \mathbb{S}^r : \|V - U_0^*\| \leq \epsilon_0\}$ and
- $S(x^*, \epsilon_0) = \{x \in \mathbb{R}^n : \|x - x^*\| \leq \epsilon_0\}$ for given $\epsilon_0 > 0$,

and, using these neighborhoods, the following pair of mappings:

$$\begin{aligned} \Psi_1 : \quad & S(x^*, \epsilon_0) \times S(U_0^*, \epsilon_0) \times S(0, \delta) \times (0, \infty) \to \mathbb{R}^n \\ & (x, \widehat{U}_0, T, p) \mapsto f'(x)^\top + \left[\left\langle E_0 \widehat{U}_0 E_0^\top, \mathcal{A}_i'(x) \right\rangle \right]_{i=1}^n + h(x, T, p), \end{aligned} \tag{6.7}$$

$$\begin{aligned} \Psi_2 : \quad & S(x^*, \epsilon_0) \times S(U_0^*, \epsilon_0) \times S(0, \delta) \times (0, \infty) \to \mathbb{S}^r \\ & (x, \widehat{U}_0, T, p) \mapsto p E_0^\top \left(D\Phi_p \left(\mathcal{A}(x) \right) [p^{-1} T + U^*] \right) E_0 - p \widehat{U}_0. \end{aligned} \tag{6.8}$$

A frequently used isometry is provided by the following definition:

Definition 6.2 *Given a symmetric matrix $A \in \mathbb{S}^m$ we define the operator* **svec** : $\mathbb{S}^m \to \mathbb{R}^{m(m+1)/2}$ *as*

$$\textbf{svec}(A) = (a_{11}, \sqrt{2} a_{12}, a_{22}, \sqrt{2} a_{13}, \sqrt{2} a_{23}, a_{33}, \ldots)^\top \in \mathbb{R}^{m(m+1)/2}.$$

Further we define the operator **smat** : $\mathbb{R}^{m(m+1)/2} \to \mathbb{S}^m$ *as the inverse of* **svec**.

Using definition 6.2 we define $\tilde{r} = r(r+1)/2$, $\tilde{m} = m(m+1)/2$, $u_0^* =$ **svec**(U_0^*), the neighborhoods

$$S(u_0^*, \epsilon_0) = \{\hat{u} \in \mathbb{R}^{\tilde{r}} : \|\hat{u} - u_0^*\| \leq \epsilon_0\}$$

and

$$S(\mathbb{O}_{\mathbb{R}^{\tilde{m}}}, \delta_0) = \{t \in \mathbb{R}^{\tilde{m}} : \|t\| \leq \delta_0\}.$$

and a mapping

$$\Psi : S(x^*, \epsilon_0) \times S(u_0^*, \epsilon_0) \times S(0_{\mathbb{R}^{\tilde{m}}}, \delta_0) \times (0, \infty) \to \mathbb{R}^{n + \tilde{r}}$$

by

$$\begin{aligned} \Psi(x, \hat{u}_0, t, p) = & \Big(\Psi_1(x, \textbf{smat}(\hat{u}_0), \textbf{smat}(t), p), \\ & \textbf{svec}\left(\Psi_2(x, \textbf{smat}(\hat{u}_0), \textbf{smat}(t), p) \right) \Big). \end{aligned} \tag{6.9}$$

Now we are prepared to start with the proof of assertion (G1). The idea is to apply the following implicit function theorem, which is a slightly modified version of the Implicit Function Theorem 2 presented in [12, p. 12].

Theorem 6.1 *(Implicit Function Theorem) Let S be an open subset of $\mathbb{R}^{m+n}$, $\bar{X}$ be a compact subset of $\mathbb{R}^m$, and $h : S \to \mathbb{R}^n$ be a function such that $h \in C^1$ on S. Assume that $\nabla_y h(x, y)$ exists and is continuous on S. Assume $\bar{y} \in \mathbb{R}^n$ is a vector such that the matrix $\nabla_y h(\bar{x}, \bar{y})$ is nonsingular for all $\bar{x} \in \bar{X}$. Then there exist scalars $\epsilon > 0, \delta > 0$, and a function $\phi : S(\bar{X}; \epsilon) \to S(\bar{y}; \delta)$ such that $\phi \in C^1$ on $S(\bar{X}; \epsilon), \bar{y} = \phi(\bar{x})$ for all $\bar{x} \in \bar{X}$, and $h[x, \phi(x)] = 0$ for all $x \in S(\bar{X}; \epsilon)$. The function ϕ is unique in the sense that if $x \in S(\bar{X}; \epsilon)$, $y \in S(\bar{X}; \delta)$ and $h(x, y) = 0$, then $y = \phi(x)$.*

The following table identifies functions and sets in Theorem 6.1 with the corresponding functions and sets in the notation used in this thesis:

Theorem 6.1	our notation
m	$\hat{m} + 1$
n	$n + \hat{r}$
$S \subset \mathbb{R}^{m+n}$	$S(x^*, \epsilon_0) \times S(u_0^*, \epsilon_0) \times S(0_{\mathbb{R}^{\bar{m}}}, \delta_0) \times (0, \infty) \subset \mathbb{R}^{(n+\hat{r})+(\hat{m}+1)}$
$\bar{X}$	$\mathcal{K} = \{\mathbb{O}_{\mathbb{R}^{\bar{m}}}\} \times I,$ where $I \subset \mathbb{R}_+$ is a compact interval
$\bar{y}$	(x^*, u_0^*)
h	Ψ

In order to satisfy all assumptions of Theorem 6.1 we have to show that

- Ψ is continuous with respect to all variables,
- $\Psi(x^*, u_0^*, 0, p) = 0$,
- Ψ is continuously differentiable with respect to x and $\hat{u}_0$ and
- $\Psi'_{x,\hat{u}_0}(x^*, u_0^*, 0, p)$ is nonsingular for all p small enough.

We start with discussing some basic properties of h, Ψ_1 and Ψ_2.

Lemma 6.2 *If $\mathcal{A}$ is a twice continuously differentiable operator, then the function h defined in (6.6) is continuously differentiable with respect to x and*

a) $h(x^*, 0, p) = 0$.

b) $h'_x(x^*, 0, p) = H_p(x^*, U^*)$ *and* $h'_x(x^*, 0, p) \to H(x^*, U^*)$ *for* $p \to 0$*, where* $H(x^*, U^*)$ *is defined by formula (3.2).*

Proof . The differentiability of h is obvious.

a) Taking into account (5.1) and Lemma 5.4 we obtain

$$\begin{aligned}\widehat{U}_\perp(x^*, 0, p) &= P_\perp D\Phi_p\left(\mathcal{A}(x^*)\right)[U^*]\, P_\perp \\ &\quad + P_0 D\Phi_p\left(\mathcal{A}(x^*)\right)[U^*]\, P_\perp + P_\perp D\Phi_p\left(\mathcal{A}(x^*)\right)[U^*]\, P_0 \\ &= P_\perp U^* P_\perp + P_0 U^* P_\perp + P_\perp U^* P_0 = 0. \end{aligned} \qquad (6.10)$$

Thus we obtain $h(x^*, 0, p) = 0$.

b) Using Corollary 2.4 we get for any matrix $B \in \mathbb{S}^m$

$$\begin{aligned}
&\frac{\partial}{\partial x_i} D\Phi_p\left(\mathcal{A}(x)\right)[B] = D^2\Phi_p\left(\mathcal{A}(x)\right)[B; \mathcal{A}_i'(x)] \\
= &\sum_{j,k,l=1}^{\mu(x)} \Delta^2\varphi_p(\lambda_j, \lambda_k, \lambda_l) P_j(x)\mathcal{A}_i'(x)P_k(x)BP_l(x) + \\
&\sum_{j,k,l=1}^{\mu(x)} \Delta^2\varphi_p(\lambda_j, \lambda_k, \lambda_l) P_j(x)BP_k(x)\mathcal{A}_i'(x)P_l(x).
\end{aligned}$$

Now, performing the same steps as in the proof of Theorem 5.1 c), we easily conclude

$$\begin{aligned}
\frac{\partial}{\partial x_i} D\Phi_p\left(\mathcal{A}(x^*)\right)[U^*] &= D^2\Phi_p\left(\mathcal{A}(x^*)\right)[U^*; \mathcal{A}_i] \\
&= \sum_{k=1}^{\mu(x^*)} \Delta^2\varphi_p(\lambda_k, 0, 0) P_k(x^*)\mathcal{A}_i U^* + \\
&\quad \sum_{k=1}^{\mu(x^*)} \Delta^2\varphi_p(\lambda_k, 0, 0) U^*\mathcal{A}_i P_k(x^*).
\end{aligned}$$

Considering $U^*P_\perp = P_\perp U^* = 0$ we obtain

$$\begin{aligned}
\widehat{U}_{\perp,x}'(x^*, 0, p) &= \Big[\sum_{k=1}^{\mu(x^*)-1} \Delta^2\varphi_p(\lambda_k, 0, 0) P_k(x^*)\mathcal{A}_i U^* + \\
&\quad \sum_{k=1}^{\mu(x^*)-1} \Delta^2\varphi_p(\lambda_k, 0, 0) U^*\mathcal{A}_i P_k(x^*) \Big]_{i=1}^{n}
\end{aligned}$$

and using (6.10)

$$h_x'(x^*, 0, p) = 2\left[\left\langle U^*, \mathcal{A}_i \left(\sum_{k=1}^{\mu(x^*)-1} \Delta^2\varphi_p(\lambda_k, 0, 0) P_k(x^*) \right) \mathcal{A}_j \right\rangle\right]_{i,j=1.}^{n}$$

The latter matrix is equal to the matrix $H_p(x^*, U^*)$ defined in the proof of Theorem 5.1. Thus we can show that the right hand side of the latter equation converges to $H(x^*, U^*)$ as p tends to 0 using exactly the same arguments as in the proof of Theorem 5.1. □

Lemma 6.3

$$\Psi_1(x^*, U_0^*, 0, p) = \Psi_2(x^*, U_0^*, 0, p) = 0 \textit{ for all } p > 0.$$

Proof . Using (5.1) we can show that

$$\begin{aligned}\Psi_1(x^*, U_0^*, 0, p) &= f'(x^*)^\top + \left[\left\langle E_0 E_0^\top U^* E_0 E_0^\top, \mathcal{A}_i \right\rangle\right]_{i=1}^n + h(x^*, 0, p) \\ &= L'_x(x^*, U^*) = 0, \\ \Psi_2(x^*, U_0^*, 0, p) &= pE_0^\top \left(D\Phi_p \left(\mathcal{A}(x)\right) [U^*]\right) E_0 - pE_0^\top U^* E_0 \\ &= pE_0^\top U^* E_0 - pE_0^\top U^* E_0 = 0.\end{aligned}$$

for all $p > 0$. □

As a direct consequence of Lemma 6.3 we obtain

$$\Psi(x^*, u_0^*, 0, p) = 0 \text{ for all } p > 0. \tag{6.11}$$

Next, we investigate the differentiability of Ψ_1 and Ψ_2 and give formulas for partial derivatives.

Lemma 6.4 *Let $\mathcal{A}$ and f be twice continuously differentiable. Then the functions Ψ_1 and Ψ_2 defined in (6.7) and (6.8) are twice continuously differentiable and the following formulas for the derivatives of Ψ_1 and Ψ_2 hold true:*

$$\begin{aligned}\Psi'_{1,x}(x, \widehat{U}_0, T, p) &= f''(x) + \left[\left\langle E_0 \widehat{U}_0 E_0^\top, \mathcal{A}''_{i,j}(x)\right\rangle\right]_{i,j=1}^n + h'_x(x, T, p), \\ \Psi'_{2,x}(x, \widehat{U}_0, T, p) &= p\Big[\sum_{j,k,l=1}^{\mu(x)} \Delta^2(\lambda_j(x), \lambda_k(x), \lambda_l(x)) \cdot \\ &\qquad E_0^\top \Big(P_j(x)\mathcal{A}'_i(x)P_k(x)(p^{-1}T + U^*)P_l(x) + \\ &\qquad\qquad P_j(x)(p^{-1}T + U^*)P_k(x)\mathcal{A}'_i(x)P_l(x)\Big) E_0\Big]_{i=1}^n \\ &= p\Big[E_0^\top D^2\Phi_p(\mathcal{A}(x))[p^{-1}T + U^*; \mathcal{A}'_i(x)]E_0\Big]_{i=1}^n, \\ \Psi'_{1,\widehat{U}_0}(x, \widehat{U}_0, T, p) &= \left[E_0^\top \mathcal{A}'_i(x) E_0\right]_{i=1}^n, \\ \Psi'_{2,\widehat{U}_0}(x, \widehat{U}_0, T, p) &= -pE, \textit{ where } E_{i,j} = 1 \textit{ for all } i, j \in \{1, \ldots, r\}.\end{aligned}$$

Proof . The first and third formula follow easily from the linearity of the trace operator. For the second formula we only note that

$$\frac{\partial}{\partial x_i} D\Phi_p\left(\mathcal{A}(x)\right)[B] = D^2\Phi_p\left(\mathcal{A}(x)\right)[B; \mathcal{A}'_i(x)]$$

for any matrix $B \in \mathbb{S}^m$. The last equation can be seen directly. □

Corollary 6.5

$$\begin{aligned}
\Psi'_{1,x}(x^*, U_0^*, 0, p) &= L''(x^*, U^*) + H_p(x^*, U^*),\\
\Psi'_{2,x}(x^*, U_0^*, 0, p) &= \varphi''(0)\Big[E_0^\top \left(\mathcal{A}_i U^* + U^* \mathcal{A}_i\right) E_0\Big]_{i=1}^n,\\
\Psi'_{1,\widehat{U}_0}(x^*, U_0^*, 0, p) &= \left[E_0^\top \mathcal{A}_i E_0\right]_{i=1}^n,\\
\Psi'_{2,\widehat{U}_0}(x^*, U_0^*, 0, p) &= -pE, \text{ where } [E]_{i,j} = 1 \text{ for all } i, j \in \{1, \dots, r\}.
\end{aligned}$$

Proof . The first formula can be seen using Lemma 6.2 b). The second formula follows from Lemma 6.4 using the same arguments as in the proof of Theorem 5.1 c). The remaining formulas can be directly seen from Lemma 6.4. □

In the following Lemma and Corollary we derive the partial derivative of Ψ with respect to x and $\hat{u}_0$.

Lemma 6.6 *Let $\mathcal{A}$ and f be twice continuously differentiable. Then the function Ψ defined in (6.9) is continuously differentiable with respect to x, $\hat{u}_0$ and t and the following formula holds:*

$$\Psi'_{x,\hat{u}_0}(x, \hat{u}_0, t, p) =$$

$$\begin{pmatrix} f''(x) + \Big[\Big\langle E_0 \widehat{U}_0 E_0^\top, \mathcal{A}''_{i,j}(x)\Big\rangle\Big]_{i,j=1}^n + h'_x(x, T, p) & \left[\textbf{svec}\left(E_0^\top \mathcal{A}'_i(x) E_0\right)\right]_{i=1}^n \\ p\left(\left[E_0^\top D_2\Phi_p(\mathcal{A}(x)[\mathcal{A}'_i(x); p^{-1}T + U^*])E_0\right]\right)^\top & pI_{\tilde{r}} \end{pmatrix}.$$

Proof . Lemma 6.6 is a direct consequence of Lemma 6.4 and [76], where formulas for the derivatives of the functions **svec** and **smat** are presented. □

Corollary 6.7

$$\Psi'_{(p)} = \Psi'_{x,\hat{u}_0}(x^*, u_0^*, 0, p) =$$

$$\begin{pmatrix} L''(x^*, U^*) + H_p(x^*, U^*) & \left[\textbf{svec}\left(E_0^\top \mathcal{A}_i E_0\right)\right]_{i=1}^n \\ \left(\left[\textbf{svec}\left(\varphi''(0) E_0^\top \left(\mathcal{A}_i U^* + U^* \mathcal{A}_i\right) E_0\right)\right]_{i=1}^n\right)^\top & pI_{\tilde{r}} \end{pmatrix}.$$

Proof . Corollary 6.7 follows directly from Lemma 6.6 and Corollary 6.5. □

Along with $\Psi'_{x,\hat{u}_0}(x^*, u_0^*, 0, p)$ we define

$$\Psi'_{(0)} = \lim_{p\to\infty} \Psi'_{x,\hat{u}_0}(x^*, u_0^*, 0, p) = \tag{6.12}$$

$$\begin{pmatrix} L''(x^*, U^*) + H(x^*, U^*) & \left[\textbf{svec}\left(E_0^\top \mathcal{A}_i E_0\right)\right]_{i=1}^n \\ \left(\left[\textbf{svec}\left(\varphi''(0) E_0^\top \left(\mathcal{A}_i U^* + U^* \mathcal{A}_i\right) E_0\right)\right]_{i=1}^n\right)^\top & 0 \end{pmatrix}.$$

The following Lemma and Corollary provide information about the regularity of $\Psi'_{(0)}$ and $\Psi'_{(p)}$.

Lemma 6.8 $\Psi'_{(0)}$ *is nonsingular.*

Proof . For any pair $(y, v) \in \mathbb{R}^{n+\tilde{r}}$ with $\Psi'_{(0)}(y, v) = 0$ the following equations hold:

$$\begin{aligned} (L''(x^*, U^*) + H(x^*, U^*))y + \left[\textbf{svec}\left(E_0^\top \mathcal{A}_i E_0\right)\right]_{i=1}^n v &= 0 && (6.13)\\ \left(\left[\textbf{svec}\left(\varphi''(0) E_0^\top \left(\mathcal{A}_i U^* + U^* \mathcal{A}_i\right) E_0\right)\right]_{i=1}^n\right)^\top y &= 0. && (6.14)\end{aligned}$$

Now let $\lambda_1(U^*), \ldots, \lambda_m(U^*)$ denote the increasingly ordered eigenvalues of U^*, define $\Lambda^*_{U_0} = \operatorname{diag}(\lambda_{m-r+1}(U^*), \ldots, \lambda_m(U^*)) \in \mathbb{S}^r_{++}$ and recall that $\lambda_1(U^*) = \ldots = \lambda_{m-r}(U^*) = 0$ and $E_0 = (s_{m-r+1}, \ldots, s_m)$. Then we conclude

$$U^* = \sum_{k=m-r+1}^m \lambda_k(U^*) s_k s_k^\top = E_0 \Lambda^*_{U_0} E_0^\top$$

and

$$\begin{aligned}
(6.14) \quad &\Leftrightarrow \quad \sum_{i=1}^n y_i \left(E_0^\top \mathcal{A}_i \left(E_0 \Lambda^*_{U_0} E_0^\top\right) E_0 + E_0^\top \left(E_0 \Lambda^*_{U_0} E_0^\top\right) \mathcal{A}_i E_0\right) = 0\\
&\Leftrightarrow \quad \sum_{i=1}^n y_i \left(E_0^\top \mathcal{A}_i E_0 \Lambda^*_{U_0} + \Lambda^*_{U_0} E_0^\top \mathcal{A}_i E_0\right) = 0\\
&\Leftrightarrow \quad \sum_{i=1}^n y_i \left[s_k^\top \mathcal{A}_i s_l \lambda_l(U^*) + \lambda_k(U^*) s_k^\top \mathcal{A}_i s_l\right]_{k,l=m-r+1}^m = 0\\
&\Leftrightarrow \quad \sum_{i=1}^n y_i \left[(\lambda_k(U^*) + \lambda_l(U^*)) s_k^\top \mathcal{A}_i s_l\right]_{k,l=m-r+1}^m = 0\\
&\Leftrightarrow \quad [\lambda_k(U^*) + \lambda_l(U^*)]_{k,l=m-r+1}^m \bullet \left(\sum_{i=1}^n y_i \left[s_k^\top \mathcal{A}_i s_l\right]_{k,l=m-r+1}^m\right) = 0\\
&\Leftrightarrow \quad \sum_{i=1}^n y_i \left[s_k^\top \mathcal{A}_i s_l\right]_{k,l=m-r+1}^m = 0\\
&\Leftrightarrow \quad y^\top \left[\textbf{svec}\left(E_0^\top \mathcal{A}_i E_0\right)\right]_{i=1}^n = 0. \qquad (6.15)
\end{aligned}$$

Hence, multiplying (6.13) by $y^\top$ from left we get

$$y^\top (L''(x^*, U^*) + H(x^*, U^*))y = 0.$$

Now from the second order optimality conditions for (SDP) follows either $y \notin \mathcal{C}(x^*)$ or $y = 0$. Therefore from (6.15) and Lemma 3.1 we conclude $y = 0$. Finally (6.13) together with assumption $(A3)$ show $v = 0$ and therefore $\Psi'_{(0)}$ is nonsingular. □

Corollary 6.9 *There exist $p_0 > 0$ and $\rho > 0$ such that $\|\Psi'^{-1}_{(p)}\| < \rho$ for all $p < p_0$. Furthermore $\Psi'_{(p)}$ is nonsingular for all $p < p_0$.*

Proof . Lemma 6.8 guarantees that

a) there exists $\gamma_0 > 0$ such that $\|\Psi'^{-1}_{(0)}\| \le \gamma_0$ and

b) there exists $\mu_0 > 0$ such that $\|\Psi'_{(0)}w\|^2 > \mu_0\|w\|^2$ for all $w \in \mathbb{R}^{n+\tilde{r}}$.

From b) and the continuity of $\Psi'_{(p)}$ with respect to p, which can be easily derived from Theorem 5.1, it follows that we find $p_0 > 0$ such that

$$\|\Psi'_{(p)}w\|^2 > \frac{\mu_0}{2}\|w\|^2 \text{ for all } w \in \mathbb{R}^{n+\tilde{r}} \text{ and all } p \le p_0.$$

Consequently the smallest eigenvalue of $\Psi'_{(p)}$ is larger than $\frac{\mu}{2}$, $\Psi'_{(p)}$ is nonsingular and there exists $\rho > 0$ independent of p such that $\|\Psi'^{-1}_{(p)}\| < \rho$ for all $p < p_0$. $\square$

Now we are able to state the following proposition, which partly proves assertion (G1):

Proposition 6.10 *There exist $\delta > 0$ and $p_0 > 0$ small enough such that for given $\Theta > \lambda_m(U^*)$, any $0 < \epsilon < \lambda_{m-r+1}(U^*)$ and any $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ there exists a vector $\hat{x} = \hat{x}(U,p)$ such that the equation $F'_x(\hat{x}, U, p) = 0$ holds.*

Proof . We have shown that we find $p_0 > 0$ such that

- $\Psi(x^*, u_0^*, 0, p) = 0$,
- Ψ is continuously differentiable with respect to x and $\hat{u}_0$ and
- $\Psi'_{(p)}$ is nonsingular

for all $0 < p < p_0$. Now let $p_1 < p_0$ be arbitrary small and define the compact set $\mathcal{K} = \{\mathbb{O} \in \mathbb{R}^{\tilde{m}}\} \times [p_1, p_0]$. Then it follows from the implicit function Theorem 6.1 that there exist $\delta > 0$ and smooth functions $x(t,p)$ and $\hat{u}_0(t,p)$ defined uniquely in a neighborhood

$$\mathcal{S}(\mathcal{K}, \delta) = \big\{(t,p) \in \mathbb{R}^{\tilde{m}+1} : \|t\| \le \delta,\ p \in [p_1, p_0]\big\}$$

of the compact set $\mathcal{K}$ such that

- $\Psi(x(t,p), \hat{u}_0(t,p), t, p) = 0$ for all $(t,p) \in \mathcal{S}(\mathcal{K}, \delta_0)$ and
- $x(0,p) = x^*$, $\hat{u}_0(0,p) = \hat{u}_0^*$ for any $p \in [p_1, p_0]$.

Recalling that $\mathbf{smat}(t) = T = p(U - U^*)$ and $\|v\| = \|\mathbf{smat}(v)\|$ we conclude $\mathbf{svec}(p(U - U^*)) \in S(\mathcal{K}, \delta)$ for any $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$. Thus

$$\hat{x} = \hat{x}(U,p) = x(\mathbf{svec}(p(U - U^*)), p)$$

and

$$\widehat{U}_0 = \widehat{U}_0(U,p) = \mathbf{smat}(\hat{u}_0(\mathbf{svec}(p(U - U^*)), p)).$$

exist and the following equations hold true:

$$f'(\hat{x})^\top + \left[\left\langle E_0 \widehat{U}_0 E_0^\top, \mathcal{A}_i'(\hat{x})\right\rangle\right]_{i=1}^n + h(\hat{x}, p(U-U^*), p) \;=\; 0 \qquad (6.16)$$

$$pE_0^\top \left(D\Phi_p\left(\mathcal{A}(\hat{x})\right)[U]\right)E_0 - p\widehat{U}_0 \;=\; 0. \qquad (6.17)$$

From the latter equation we see that $\widehat{U}_0 = E_0^\top \left(D\Phi_p\left(\mathcal{A}(\hat{x})\right)[U]\right)E_0$ and after substitution of $\widehat{U}_0$ in (6.16) we obtain

$$\begin{aligned} 0 &= f'(\hat{x})^\top + \left[\langle P_0\left(D\Phi_p\left(\mathcal{A}(\hat{x})\right)[U]\right)P_0, \mathcal{A}_i'(\hat{x})\rangle\right]_{i=1}^n \\ &\quad + \left[\left\langle \widehat{U}_\perp(\hat{x}, p(U-U^*), p), \mathcal{A}_i'(\hat{x})\right\rangle\right]_{i=1}^n \\ &= f'(\hat{x})^\top + \left[\langle D\Phi_p\left(\mathcal{A}(\hat{x})\right)[U], \mathcal{A}_i'(\hat{x})\rangle\right]_{i=1}^n \\ &= f'(\hat{x})^\top + \left[\langle U, D\Phi_p\left(\mathcal{A}(\hat{x})\right)[\mathcal{A}_i'(\hat{x})]\rangle\right]_{i=1}^n \\ &= F_x'(\hat{x}, U, p). \end{aligned}$$

This completes the proof of Proposition 6.10. □

Later we will show that $\hat{x} = \mathrm{argmin}_{x\in\mathbb{R}} F(x,U,p)$ and therefore complete the proof of assertion (G1). Next we will prove assertion (G2). We start with the following Lemma.

Lemma 6.11 *There exist $\delta > 0$ and $p_0 > 0$ small enough such that for given $\Theta > \lambda_m(U^*)$, any $0 < \epsilon < \lambda_{m-r+1}(U^*)$ and any $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ the estimate*

$$\max\left\{\|\hat{x} - x^*\|, \|E_0^\top(\widehat{U} - U^*)E_0\|\right\} \le Cp\,\|U - U^*\|$$

holds, where C is a constant independent of p.

Proof. We start by rewriting equations (6.16) and (6.17) using $T = $ **smat**(t) and $\tilde{U}_0(t,p) = $ **smat**$(\hat{u}_0(t,p))$:

$$f'(x(t,p))^\top + \left[\left\langle E_0 \tilde{U}_0(t,p) E_0^\top, \mathcal{A}_i'(x(t,p))\right\rangle\right]_{i=1}^n + h(x(t,p), T, p) \;=\; 0 \quad (6.18)$$

$$p\,\mathbf{svec}\left(E_0^\top\left(D\Phi_p\left(\mathcal{A}(x(t,p))\right)[p^{-1}T + U^*]\right)E_0\right) - p\hat{u}_0(t,p) \;=\; 0. \quad (6.19)$$

Now, differentiating identity (6.18) with respect to t we get

$$\begin{aligned} &f''(x(t,p))\cdot x_t'(t,p) + \left[\left\langle E_0\tilde{U}_0(t,p)E_0^\top, \mathcal{A}_{i,j}''(x(t,p))\right\rangle\right]_{i,j=1}^n \cdot x_t'(t,p) + \\ &\left[\mathbf{svec}\left(E_0^\top \mathcal{A}_i'(x(t,p))E_0\right)^\top\right]_{i=1}^n \cdot \hat{u}_{0,t}'(t,p) + h_t'(x(t,p), T, p) = 0, \end{aligned} \qquad (6.20)$$

where

$$x_t'(t,p) = \frac{\partial}{\partial t}\left(x_j(t,p), j = 1,\ldots,n\right) \in \mathbb{R}^{n,\tilde{m}}$$

and

$$\hat{u}_{0,t}'(t,p) = \frac{\partial}{\partial t}\left(\hat{u}_{0,j}(t,p), j = 1,\ldots,\tilde{r}\right) \in \mathbb{R}^{\tilde{r},\tilde{m}}.$$

Differentiation of (6.19) with respect to t yields:

$$\begin{aligned}
& p\hat{u}'_{0,t}(t,p) \\
= \;& p\frac{\partial}{\partial t}\,\textbf{svec}\left(E_0^\top\left(D\Phi_p\mathcal{A}(x(t,p))\left[p^{-1}T+U^*\right]\right)E_0\right) \\
= \;& p\left[\textbf{svec}\left(E_0^\top\left(D^2\Phi_p\mathcal{A}(x(t,p))\left[p^{-1}T+U^*,\mathcal{A}'_i(x(t,p))\right]\right)E_0\right)^\top\right]_{i=1}^{n}\cdot x'_t(t,p) \\
& +\left(\left[\textbf{svec}\left(E_0^\top\left(D\Phi_p\mathcal{A}(x(t,p))\left[\textbf{smat}(e_j)\right]\right)E_0\right)^\top\right]_{j=1}^{\tilde{m}}\right)^\top, \qquad (6.21)
\end{aligned}$$

where e_j denotes the $j-th$ unit vector in $\mathbb{R}^{\hat{m}}$. Next we calculate $h'_t(x(t,p),T,p)$:

$$\begin{aligned}
& h'_t(x(t,p),T,p) \\
= \;& \frac{\partial}{\partial t}\left(\left[\left\langle \widehat{U}_\perp(x(t,p),T,p),\mathcal{A}'_i(x(t,p))\right\rangle\right]_{i=1}^{n}\right) \\
= \;& \left[\left\langle P_\perp\mathcal{A}'_i(x(t,p))P_\perp,\frac{\partial}{\partial t_j}D\Phi_p\mathcal{A}(x(t,p))[p^{-1}T+U^*]\right\rangle\right]_{(i,j)\in I\times J}+ \\
& \left[\left\langle P_0\mathcal{A}'_i(x(t,p))P_\perp,\frac{\partial}{\partial t_j}D\Phi_p\mathcal{A}(x(t,p))[p^{-1}T+U^*]\right\rangle\right]_{(i,j)\in I\times J}+ \\
& \left[\left\langle P_\perp\mathcal{A}'_i(x(t,p))P_0,\frac{\partial}{\partial t_j}D\Phi_p\mathcal{A}(x(t,p))[p^{-1}T+U^*]\right\rangle\right]_{(i,j)\in I\times J}+ \\
& \left[\left\langle \widehat{U}_\perp(x(t,p),T,p),\mathcal{A}''_{i,j}(x(t,p))\right\rangle\right]_{i,j=1}^{n}\cdot x'_t(t,p), \qquad (6.22)
\end{aligned}$$

where $I\times J=\{(i,j):i=1,\dots,n$ and $j=1,\dots,\hat{m}\}$ and

$$\begin{aligned}
& \frac{\partial}{\partial t_j}D\Phi_p\mathcal{A}(x(t,p))[p^{-1}T+U^*] \\
= \;& \textbf{smat}(\left[\textbf{svec}\left(D^2\Phi_p\mathcal{A}(x(t,p))\left[p^{-1}T+U^*,\mathcal{A}'_i(x(t,p))\right]\right)^\top\right]_{i=1}^{n}\frac{\partial}{\partial t_j}x(t,p))+ \\
& \frac{1}{p}D\Phi_p\mathcal{A}(x(t,p))\left[\textbf{smat}(e_j)\right]
\end{aligned}$$

for all $j=1,\dots,\hat{m}$. Using $B_j(t,p)=D^2\Phi_p\mathcal{A}(x(t,p))\left[p^{-1}T+U^*,\mathcal{A}'_j(x(t,p))\right]$ for $j=1,\dots,n$, recalling that

$$\begin{aligned}
h'(x(t,p),t,p) \;=\; & \left[\left\langle P_\perp A'_i(x(t,p))P_\perp,B_j(t,p)\right\rangle\right]_{i,j=1}^{n}+ \\
& \left[\left\langle P_0 A'_i(x(t,p))P_\perp,B_j(t,p)\right\rangle\right]_{i,j=1}^{n}+ \\
& \left[\left\langle P_\perp A'_i(x(t,p))P_0,B_j(t,p)\right\rangle\right]_{i,j=1}^{n}+ \\
& \left[\left\langle \widehat{U}_\perp(x(t,p),T,p),\mathcal{A}''_{i,j}(x(t,p))\right\rangle\right]_{i,j=1}^{n}
\end{aligned}$$

and defining

$$\begin{aligned} M(t,p) &= \frac{1}{p}\left[\langle P_\perp A_i'(x(t,p))P_\perp, D\Phi_p\mathcal{A}(x(t,p))\left[\textbf{smat}(e_j)\right]\rangle\right]_{(i,j)\in I\times J} \\ &\quad+\frac{1}{p}\left[\langle P_0 A_i'(x(t,p))P_\perp, D\Phi_p\mathcal{A}(x(t,p))\left[\textbf{smat}(e_j)\right]\rangle\right]_{(i,j)\in I\times J} \\ &\quad+\frac{1}{p}\left[\langle P_\perp A_i'(x(t,p))P_0, D\Phi_p\mathcal{A}(x(t,p))\left[\textbf{smat}(e_j)\right]\rangle\right]_{(i,j)\in I\times J} \end{aligned}$$

from (6.22) we obtain

$$h_t'(x(t,p),t,p) = h'(x(t,p),t,p)\cdot x_t'(t,p) + M(t,p). \tag{6.23}$$

Now defining

$$N(t,p) = \left[\textbf{svec}\left(E_0^\top\left(D\Phi_p\mathcal{A}(x(t,p))\left[\textbf{smat}(e_j)\right]\right)E_0\right)\right]_{j=1}^{\tilde{m}}$$

and combining Lemma 6.6, (6.20), (6.21) and (6.23) we obtain the system

$$\Psi'_{x,\hat{u}_0}(x(t,p),\hat{u}_0(t,p),t,p)\begin{pmatrix} x_t'(t,p) \\ \hat{u}_{0,t}'(t,p)\end{pmatrix} = -\begin{pmatrix} M(t,p) \\ N(t,p)\end{pmatrix}. \tag{6.24}$$

Next we consider the special case $t=0$. From (6.23) and Lemma 6.2 b) we see that

$$h_t'(x(0,p),0,p) = H_p(x^*,U^*)\cdot x_t'(0,p) + M_0, \tag{6.25}$$

where $M_0 = M(0,p)$ with

$$\begin{aligned} M_{0,i,j} &= \frac{1}{p}\Big(\textbf{svec}\Big(D\Phi_p(\mathcal{A}(x^*))\Big[\underbrace{P_0\mathcal{A}_iP_\perp + P_\perp\mathcal{A}_iP_0 + P_\perp\mathcal{A}_iP_\perp}_{=:M_1}\Big]\Big)\Big)_j \\ &= \frac{1}{p}\,\textbf{svec}\Big(\sum_{k,l=1}^{\mu(x^*)}\Delta\varphi_p(\lambda_k,\lambda_l)(P_k(x^*)M_1P_l(x^*) + P_l(x^*)M_1P_k(x^*))\Big)_j \\ &= \frac{1}{p}\,\textbf{svec}\Big(\sum_{k,l=1}^{\mu(x^*)-1}\Delta\varphi_p(\lambda_k,\lambda_l)P_k(x^*)\mathcal{A}_iP_l(x^*) \\ &\quad+\sum_{k=1}^{\mu(x^*)-1}\frac{\varphi_p(\lambda_k)}{p}\left(P_0\mathcal{A}_iP_k(x^*) + P_k(x^*)\mathcal{A}_iP_0\right)\Big)_j. \end{aligned}$$

Therefore we find the estimate

$$\|M_0\| \le \left(\left(2\frac{|C_{\varphi,-\infty}|}{\sigma} + C_{\varphi',\sigma}\right)\sum_{i=1}^n \|\,\textbf{svec}(\mathcal{A}_i)\|\right)^{\frac{1}{2}} =: C_M.$$

Now from (6.20) at $t=0$ and (6.25) we conclude

$$(L_{xx}'(x^*) + H_p(x^*,U^*))\cdot x_t'(0,p) + \left[\textbf{svec}\left(E_0^\top\mathcal{A}_iE_0\right)^\top\right]_{i=1}^n\cdot\hat{u}_{0,t}'(0,p) = -M_0. \tag{6.26}$$

Evaluating (6.21) at $t = 0$ and using Corollary 6.5 we obtain

$$\varphi''(0)\left[E_0^\top(\mathcal{A}_i U^* + U^*\mathcal{A}_i)E_0\right] \cdot x_t'(0,p) - pI_{\tilde{r}} \cdot \hat{u}_{0,t}'(0,p) = -N_0, \tag{6.27}$$

where $N_0 = N(0,p)$ with

$$N_{0,i,j} = \textbf{svec}\left(E_0^\top \textbf{smat}(e_j)E_0\right)_i$$

and thus

$$\|N_0\| \leq 1.$$

Combining (6.26) and (6.27) we obtain the system

$$\Psi'_{(p)}\begin{pmatrix} x_t'(0,p) \\ \hat{u}_{0,t}'(0,p) \end{pmatrix} = -\begin{pmatrix} M_0 \\ N_0 \end{pmatrix}$$

or equivalently

$$\begin{pmatrix} x_t'(0,p) \\ \hat{u}_{0,t}'(0,p) \end{pmatrix} = -\left(\Psi'_{(p)}\right)^{-1}\begin{pmatrix} M_0 \\ N_0 \end{pmatrix}. \tag{6.28}$$

Taking into account Corollary 6.9 and the estimates for $\|M\|$ and $\|N\|$ above, we see that

$$\max\left\{\|x_t'(0,p)\|, \|\hat{u}_{0,t}'(0,p)\|\right\} \leq \rho(C_M + 1).$$

Furthermore, for δ_0 small enough and any $(t,p) \in \left\{(t,p) \in \mathbb{R}^{\tilde{r}+1} | \|t\| \leq \delta_0, p \leq p_0\right\}$, the inequality

$$\left\|\Psi'^{-1}_{x,\hat{u}}(x(\tau t,p), \hat{u}_0(\tau t,p), \tau t, p)\begin{pmatrix} M(x(\tau t,p), \hat{u}_0(\tau t,p)) \\ N(x(\tau t,p), \hat{u}_0(\tau t,p)) \end{pmatrix}\right\| \leq 2\rho(C_M + 1) =: C_0$$

holds true for any $\tau \in [0,1]$. Also we have

$$\begin{aligned}
\begin{pmatrix} x(t,p) - x^* \\ \hat{u}_0(t,p) - u_0^* \end{pmatrix} &= \begin{pmatrix} x(t,p) - x(0,p) \\ \hat{u}_0(t,p) - \hat{u}_0(0,p) \end{pmatrix} \\
&= \int_0^t \Psi'^{-1}_{x,\hat{u}_0}(x(\nu,p), \hat{u}_0(\nu,p), \nu, p)\begin{pmatrix} M(x(\nu,p), \hat{u}_0(\nu,p)) \\ N(x(\nu,p), \hat{u}_0(\nu,p)) \end{pmatrix} d\nu \\
&= \int_0^1 \Psi'^{-1}_{x,\hat{u}_0}(x(\tau t,p), \hat{u}_0(\tau t,p), \tau t, p)\begin{pmatrix} M(x(\tau t,p), \hat{u}_0(\tau t,p)) \\ N(x(\tau t,p), \hat{u}_0(\tau t,p)) \end{pmatrix} t d\tau.
\end{aligned}$$

From the latter equation we obtain

$$\begin{aligned}
&\left\|\begin{matrix} x(t,p) - x^* \\ \hat{u}_0(t,p) - u_0^* \end{matrix}\right\| \\
\leq\ &\left\|\Psi'^{-1}_{x,\hat{u}_0}(x(\tau t,p), \hat{u}_0(\tau t,p), \tau t, p)\begin{pmatrix} M(x(\tau t,p), \hat{u}_0(\tau t,p)) \\ N(x(\tau t,p), \hat{u}_0(\tau t,p)) \end{pmatrix}\right\| \|t\| \int_0^1 d\tau \\
\leq\ &C_0\|t\|,
\end{aligned}$$

and therefore

$$\max\left\{\|x(t,p) - x^*\|, \|\hat{u}_0(t,p) - \hat{u}^*\|\right\} \leq C_0\|t\|.$$

Consequently for $U = p^{-1}\,\mathbf{smat}(t) + U^*$ there exists $C_1 > 0$ independent of p such that

$$\max\left\{\|\hat{x}(U,p) - x^*\|, \|E_0^\top\left(\widehat{U}(U,p) - U^*\right)E_0\|\right\} \le pC_1\|U - U^*\| \tag{6.29}$$

for all $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ and the proof of Lemma 6.11 is complete. □

Lemma 6.11 proves the assertion of (G2) for the primal variable and the component of the multiplier associated with the "active part" of $\mathcal{A}(x^*)$. Based on these results, we are going to show the remaining part of estimate (6.1) now. We start with the following inequality:

$$\|\widehat{U} - U^*\| \le \|P_0\left(\widehat{U} - U^*\right)P_0\| + 2\|P_0\left(\widehat{U} - U^*\right)P_\perp\| + \|P_\perp\left(\widehat{U} - U^*\right)P_\perp\|.$$

From (6.29) we conclude

$$\begin{aligned}
\|P_0\left(\widehat{U} - U^*\right)P_0\| &= \sqrt{\left\langle P_0\left(\widehat{U} - U^*\right)P_0, P_0\left(\widehat{U} - U^*\right)P_0\right\rangle} \\
&= \sqrt{\operatorname{tr}\left(P_0\left(\widehat{U} - U^*\right)P_0\left(\widehat{U} - U^*\right)\right)} \\
&= \sqrt{\operatorname{tr}\left(E_0E_0^\top\left(\widehat{U} - U^*\right)E_0E_0^\top\left(\widehat{U} - U^*\right)\right)} \\
&= \left\|E_0^\top\left(\widehat{U} - U^*\right)E_0\right\| \le pC_1\|U - U^*\|.
\end{aligned}$$

Again from estimate (6.29) we obtain

$$\max\left\{\|\hat{x}(U,p) - x^*\|, \|E_0^\top\left(\widehat{U}(U,p) - U^*\right)E_0\|\right\} \le \delta C_1 \tag{6.30}$$

for all $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$. From equation (6.30) we conclude that for given $\epsilon_1 > 0$ we find $\delta > 0$ small enough, such that

$$\begin{aligned}
\|\hat{x}(U,p) - x^*\| &\le \epsilon_1, &(6.31)\\
\|\widehat{U}_0(U,p) - U_0^*\| &\le \epsilon_1 &(6.32)
\end{aligned}$$

for all $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$. Using these inequalities we are able to prove the following Lemma:

Lemma 6.12 *There exists $\delta > 0$ small enough and a constant $0 < C_2 < b$ such that*

$$\frac{\lambda_i\left(\mathcal{A}(\hat{x}(U,p))\right)}{p} \le C_2$$

for all $i = 1, \dots, m$ and all $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$. Furthermore the constant C_2 does not depend on p.

Proof . From estimate (6.32) we see that we find $\delta > 0$ small enough such that

$$\|\widehat{U}_0(U,p)\| = \left\|E_0^\top\left(D\Phi_p\mathcal{A}(x(U,p))[U]\right)E_0\right\| \le \|U_0^*\| + \epsilon_1. \tag{6.33}$$

Since $\widehat{U}_0(U,p) \succeq 0$ we see from (6.33) that

$$0 \le \operatorname{tr}\left(\widehat{U}_0(U,p)\right) \le \sqrt{m}\,(\|U_0^*\| + \epsilon_1)\,. \tag{6.34}$$

Now let

$$\mathcal{A}(\hat{x}(U,p)) = S(\mathcal{A}\,(\hat{x}(U,p)))\Lambda(\mathcal{A}\,(\hat{x}(U,p)))S(\mathcal{A}\,(\hat{x}(U,p)))^\top$$

be an eigenvalue decomposition of $\mathcal{A}(\hat{x}(U,p))$, denote by $\lambda_k(\hat{x}) = \lambda_k(\mathcal{A}\,(\hat{x}(U,p)))$ the corresponding eigenvalues and by $s_k(\hat{x}) = s_k(\mathcal{A}\,(\hat{x}(U,p)))$ the corresponding eigenvectors. Then we conclude from inequality (6.31, the continuity of $\mathcal{A}$ and the continuity of the eigenvectors that we are able to find for any given $0 < \epsilon_2 < 1$ a $\delta > 0$ small enough such that the following estimates hold:

$$\left|s_i(\hat{x})^\top P_0 s_j(\hat{x})\right| \le \epsilon_2 \;\; \forall i,j = 1,\ldots,m, \quad i \ne j \tag{6.35}$$
$$\left|s_i(\hat{x})^\top P_0 s_i(\hat{x})\right| \le \epsilon_2 \;\; \forall i = 1,\ldots,m-r \tag{6.36}$$
$$\left|s_i(\hat{x})^\top P_0 s_i(\hat{x})\right| \ge 1-\epsilon_2 \;\; \forall i = m-r+1,\ldots,m. \tag{6.37}$$

Due to $(U,p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ we know that

$$s_i^\top U s_i \ge \epsilon \quad \text{for all} \quad i = m-r+1,\ldots,m.$$

Consequently we find $\delta > 0$ small enough such that the following inequalities hold:

$$s_i(\hat{x})^\top U s_i(\hat{x}) \ge \frac{3}{4}\epsilon \quad \text{for all} \quad i = m-r+1,\ldots,m. \tag{6.38}$$

Using the abbreviations

$$Q(\hat{x}) = [\Delta\varphi_p(\lambda_k(\hat{x}),\lambda_l(\hat{x}))]_{k,l=1}^m \quad \text{and} \quad S(\hat{x}) = S(\mathcal{A}\,(\hat{x}(U,p)))$$

we obtain

$$\begin{aligned}
\operatorname{tr}\left(\widehat{U}_0(U,p)\right) &= \operatorname{tr}\left(E_0^\top\,(D\Phi_p\mathcal{A}(\hat{x}(U,p))[U])\,E_0\right) = \\
&\operatorname{tr}\left(E_0^\top\left(S(\hat{x})\left[Q(\hat{x}) \bullet \left(S(\hat{x})^\top U S(\hat{x})\right)\right] S(\hat{x})^\top\right) E_0\right) \\
= \;& \operatorname{tr}\left(S(\hat{x})^\top E_0 E_0^\top S(\hat{x})\left[Q(\hat{x}) \bullet \left(S(\hat{x})^\top U S(\hat{x})\right)\right]\right) \\
= \;& \left\langle \left(S(\hat{x})^\top P_0 S(\hat{x})\right) \bullet \left(S(\hat{x})^\top U S(\hat{x})\right), Q(\hat{x})\right\rangle .
\end{aligned} \tag{6.39}$$

Next we define $Z(\hat{x}) = \left(S(\hat{x})^\top P_0 S(\hat{x})\right) \bullet \left(S(\hat{x})^\top U S(\hat{x})\right)$ and subdivide the matrices $Z(\hat{x})$ and $Q(\hat{x})$ in the following way:

$$Z(\hat{x}) = \begin{pmatrix} Z_1 & Z_2 \\ Z_2^\top & Z_3 \end{pmatrix} \text{ and } Q(\hat{x}) = \begin{pmatrix} Q_1 & Q_2 \\ Q_2^\top & Q_3 \end{pmatrix},$$

where $Z_1, Q_1 \in \mathbb{S}^{m-r}$, $Z_2, Q_2 \in \mathbb{M}^{m-r,r}$ and $Z_3, Q_3 \in \mathbb{S}^r$. Now from (6.39) we see that

$$\operatorname{tr}\left(\widehat{U}_0(U,p)\right) = \langle Z_1, Q_1\rangle + \langle Z_3, Q_3\rangle + 2\sum_{j=m-r+1}^{m}\sum_{i=1}^{m-r}(Z_2)_{i,m-r+j}(Q_2)_{i,m-r+j}.$$

From the positive semidefiniteness of $Q(\hat{x})$ and $Z(\hat{x})$ we conclude $\langle Z_1, Q_1\rangle \geq 0$, hence from (6.34) we see

$$\langle Z_3, Q_3\rangle + 2\sum_{j=m-r+1}^{m}\sum_{i=1}^{m-r}(Z_2)_{i,j}(Q_2)_{i,j} \leq \sqrt{m}\left(\|U_0^*\| + \epsilon_1\right). \tag{6.40}$$

Furthermore from the convexity of φ we get $0 \leq (Q)_{i,j} \leq (Q)_{j,j}$ for all $i \leq j, j = m-r+1,\dots,m$. Now from estimates (6.35) to (6.37) and the fact that

$$|s_i(\hat{x})^\top U s_j(\hat{x})| \leq \|U\| \leq \Theta \tag{6.41}$$

for all $i,j = 1,\dots,m$ we see that we are able to find $\delta > 0$ small enough such that

$$|Z_{i,j}| \leq \frac{\epsilon}{6(j-1)} \text{ for all } i < j, j = m-r+1,\dots,m$$

and we conclude

$$\left|2\sum_{j=m-r+1}^{m}\sum_{i=1}^{j-1}(Z)_{i,j}(Q)_{i,j}\right| \leq \frac{\epsilon}{3}\sum_{k=m-r+1}^{m}(Q)_{k,k}.$$

On the other hand from estimates (6.35) to (6.38) we obtain

$$(Z_3)_{i,i} \geq \frac{2}{3}\epsilon$$

for $\delta > 0$ small enough and thus

$$\begin{aligned}\langle Z_3, Q_3\rangle + 2\sum_{j=m-r+1}^{m}\sum_{i=1}^{m-r}(Z_2)_{i,j}(Q_2)_{i,j} &\geq \frac{2\epsilon}{3}\sum_{k=m-r+1}^{m}(Q)_{k,k} - \frac{\epsilon}{3}\sum_{k=m-r+1}^{m}(Q)_{k,k}\\ &= \frac{\epsilon}{3}\sum_{k=m-r+1}^{m}(Q)_{k,k}.\end{aligned}$$

Now we immediately obtain from (6.40)

$$\sum_{k=m-r+1}^{m}(Q)_{k,k} \leq \frac{3\sqrt{m}}{\epsilon}\left(\|U_0^*\| + \epsilon_1\right),$$

consequently

$$(Q)_{k,k} \leq \frac{3\sqrt{m}}{\epsilon}\left(\|U_0^*\| + \epsilon_1\right) \text{ for all } k = m-r+1,\dots,m$$

and finally

$$\frac{\lambda_{\max}}{p} \leq (\varphi')^{-1}\left(\frac{3\sqrt{m}}{\epsilon}\left(\|U_0^*\| + \epsilon_1\right)\right). \qquad \square$$

Remark . If we replace all estimates involving the Frobenius norm by the spectral norm in the proof of Lemma 6.12 and if we further choose ϵ close enough to $\lambda_{\min}(U_0^*)$ we obtain the estimate

$$\varphi_p'(\lambda_{\max}) \leq 2\frac{\lambda_{\max}(U_0^*)}{\lambda_{\min}(U_0^*)}.$$

This observation shows that the constant C_2 in Lemma 6.12 is closely related to the condition number of the matrix U_0^*.

Next we use Taylor expansion in a neighborhood of x^* for the calculation of the terms $\|P_0\left(\widehat{U}-U^*\right)P_\perp\|$ and $\|P_\perp\left(\widehat{U}-U^*\right)P_\perp\|$. Therefore we introduce the function

$$\begin{array}{rcl} \widetilde{U}:\mathbb{R}^n\times\mathbb{S}_+^m\times\mathbb{R} & \to & \mathbb{S}_+^m \\ (x,U,p) & \mapsto & D\Phi_p\left(\mathcal{A}(x)\right)[U], \end{array}$$

for which the equation

$$\widetilde{U}(\hat{x}(U,p),U,p)) = \widehat{U}(U,p)$$

holds. Now let $P_*\in\{P_0,P_\perp\}$ and define $\Delta x = \hat{x}(U,p)-x^*$. Taylors formula guaratees the existence of $\xi\in S(x^*,\epsilon_1)$ such that

$$\begin{aligned} P_*\left(\widetilde{U}(x,U,p)-U^*\right)P_\perp &= P_*\left(D\Phi_p\left(\mathcal{A}(x)\right)[U]-U^*\right)P_\perp \\ &= P_*\left(D\Phi_p\left(\mathcal{A}(x^*)\right)[U]-U^*\right)P_\perp + R(\xi)\Delta x, \end{aligned}$$

where $R(\xi) = \left[P_*D^2\Phi_p\mathcal{A}(\xi)[U,A_i'(\xi)]P_\perp\right]_{i=1}^n$. The following Lemma provides an upper bound for the norm of the remainder term $R(\xi)$.

Lemma 6.13 *$\|R(\xi)\|\leq C_3$, where $C_3\in\mathbb{R}$ is a constant independent of p.*

Proof . Let us denote the increasingly ordered eigenvalues of $\mathcal{A}(x)$ by $\lambda_i(x), i = 1,\ldots,n$, the corresponding Frobenius covariance matrices by $P_i(x), i=1,\ldots,n$ and the number of distinct eigenvalues of $\mathcal{A}(x)$ by $\mu(x)$.Then we obtain for all $i=1,\ldots,n$

$$P_*D^2\Phi_p\mathcal{A}(\xi)[U,A_i'(\xi)]P_\perp = \sum_{k_1,k_2,k_3=1}^{\mu(\xi)}\Delta^2(\lambda_{k_1}(\xi),\lambda_{k_2}(\xi),\lambda_{k_3}(\xi))P_*P_{k_1}(\xi)\left[M_{k_1,k_2,k_3}(\xi)+M_{k_1,k_2,k_3}^\top(\xi)\right],$$

where

$$M_{k_1,k_2,k_3}(\xi) = P_*P_{k_1}(\xi)UP_{k_2}(\xi)\mathcal{A}_i'(\xi)P_{k_3}(\xi)P_\perp$$

for all $k_1,k_2,k_3 = 1,\ldots,\mu(\xi)$ and all $i=1,\ldots,n$. Now we assume without loss of generality (see equation (6.31)) that the eigenvalues of $\mathcal{A}\left(x\right)$ are separated in the following sense: There exists

$$0<\sigma_0<\frac{1}{2}\min_{\substack{i,j=1,\ldots,\mu(x^*)\\ i\neq j}}|\lambda_i(x^*)-\lambda_j(x^*)|$$

such that for all $j = 1, \ldots, m$ there exists exactly one $i \in \{1, \ldots, \mu(x^*)\}$ with

$$|\lambda_j(x) - \lambda_i(x^*)| < \sigma_0.$$

Next we determine upper bounds for the terms

$$\Delta^2(\lambda_{k_1}(\xi), \lambda_{k_2}(\xi), \lambda_{k_3}(\xi)), \quad k_1, k_2, k_3 = 1, \ldots, \mu(\xi).$$

We define $\mu_1(x) = \max\{j \in \{1, \ldots, \mu(x)\} \mid \lambda_j(\mathcal{A}(x)) \leq -\sigma_0\}$, assume – again without loss of generality – that $\lambda_{k_1}(\xi) \leq \lambda_{k_2}(\xi) \leq \lambda_{k_3}(\xi)$ and distinguish three cases:

$\underline{k_i \leq \mu_1(\xi) \text{ for } i = 1, 2, 3}$: From the convexity of φ and φ' follows that

$$\Delta^2\varphi_p(\lambda_{k_1}(\xi), \lambda_{k_2}(\xi), \lambda_{k_3}(\xi)) \leq \varphi_p''(\lambda_{k_3}(\xi)) \leq \varphi_p''(-\sigma_0) \leq pC_{\varphi'', -\sigma_0}. \tag{6.42}$$

$\underline{\exists i \in \{1, 2, 3\} : k_i \leq \mu_1(\xi), \quad \exists j \in \{1, 2, 3\} : k_j > \mu_1(\xi)}$: Using the convexity of φ and Lemma 6.12 we obtain

$$\begin{aligned} \Delta^2\varphi_p(\lambda_{k_1}(\xi), \lambda_{k_2}(\xi), \lambda_{k_3}(\xi)) &= \frac{\Delta(\lambda_{k_3}(\xi), \lambda_{k_1}(\xi)) - \Delta(\lambda_{k_3}(\xi), \lambda_{k_2}(\xi))}{\lambda_{k_3}(\xi) - \lambda_{k_1}(\xi)} \\ \leq \quad \frac{\varphi_p'(\lambda_{k_3}(\xi)) - \varphi_p'(\lambda_{k_1}(\xi))}{\lambda_{k_3}(\xi) - \lambda_{k_1}(\xi)} &\leq \frac{\varphi'(C_2)}{\sigma_0}. \end{aligned} \tag{6.43}$$

$\underline{k_i > \mu_1(\xi) \text{ for } i = 1, 2, 3}$: Again from the convexity of φ and φ' and Lemma 6.12 we get the estimate

$$\Delta^2\varphi_p(\lambda_{k_1}(\xi), \lambda_{k_2}(\xi), \lambda_{k_3}(\xi)) \leq \varphi_p''(C_2) = \frac{1}{p}\varphi''(C_2). \tag{6.44}$$

Now we will show that in the third case when $k_i > \mu_1(\xi)$ for $i = 1, 2, 3$ the estimate

$$\|P_{k_i}(\xi)P_\perp\| \leq p\hat{C}$$

holds true, where $\hat{C}$ is a constant independent of p. First we obtain

$$\begin{aligned} \|P_{k_i}(\xi)P_\perp\| &= \operatorname{tr}(P_{k_i}(\xi)P_\perp) = \operatorname{tr}\left(E_\perp^\top P_{k_i}(\xi)E_\perp\right) \\ &\leq \sum_{k=\mu_1(\xi)+1}^{\mu(\xi)} \operatorname{tr}\left(E_\perp^\top P_k(\xi)E_\perp\right) \\ &= \operatorname{tr}\Big(E_\perp^\top\Big(\sum_{k=\mu_1(\xi)+1}^{\mu(\xi)} P_k(\xi)\Big)E_\perp\Big). \end{aligned} \tag{6.45}$$

Now, according to results presented in [73], the matrix $P_{x^*,0}(\xi) = \sum_{k=\mu_1(\xi)+1}^{\mu(\xi)} P_k(\xi)$ has the following properties:

- $P_{x^*,0}(\xi)$ is analytic in a neighborhood of x^*,
- $P_{x^*,0}(x^*) = P_0$,

- $\displaystyle \frac{\partial}{\partial x_i} P_{x^*,0}(x) = \sum_{k=1}^{\mu_1(x)} \sum_{l=\mu_1+1}^{\mu(x)} \frac{1}{\lambda_k(x) - \lambda_l(x)} \Big(B_{i,k,l}(x) + B_{i,k,l}^\top(x) \Big),$

  where $B_{i,k,l}(x) = P_k(x)\mathcal{A}'_i\,(x)\, P_l(x)$ for all $i = 1, \dots, n$.

From the last equation it is easy to see that $\left\| \frac{\partial}{\partial x} P_{x^*,0}(x) \right\|$ is restricted by a constant $\widetilde{C} > 0$ in a neighborhood of x^*. Using Taylor expansion again, we obtain:

$$\begin{aligned} \operatorname{tr}\left(E_\perp^\top P_{x^*,0}(\xi) E_\perp\right) &= \operatorname{tr}\left(E_\perp^\top P_0 E_\perp\right) + \left[\operatorname{tr}\left(E_\perp^\top \frac{\partial}{\partial x_i} P_{x^*,0}(\xi_2) E_\perp\right)\right]_{i=1}^{n\ \top} (\xi - x^*) \\ &\le \widetilde{C}\, \|\Delta x\| \le p\widetilde{C}C_1\, \|U - U^*\| \le p\widetilde{C}C_1(\Theta + \|U^*\|) \qquad (6.46) \end{aligned}$$

for some $\xi_2 \in \mathbb{S}(x^*, \epsilon_1)$. Finally the assertion of Lemma 6.13 follows from the combination of the estimates (6.42) to (6.46) and the fact that the norms of

- U,
- $P_k(x)$ for $k = 1, \dots, \mu(x)$ and
- $\mathcal{A}'_i(x)$ for $i = 1, \dots, n$

are restricted in a compact neighborhood of x^*. □

Now, due to (6.29) and Lemma 6.13, we can find a constant $C_3 > 0$ independent of p such that

$$\|P_*\left(\widehat{U} - U^*\right) P_\perp\| \le \|P_*\left(D\Phi_p\left(\mathcal{A}(x^*)\right)[U] - U^*\right) P_\perp\| + pC_3\|U - U^*\|. \qquad (6.47)$$

Next, using $U^* P_\perp = 0$ we see

$$\begin{aligned} & \|P_0\left(D\Phi_p\left(\mathcal{A}(x^*)\right)[U] - U^*\right) P_\perp\| \\ =\ & \left\| \sum_{k=1}^{\mu(x^*)-1} \frac{p\varphi(\lambda_k/p)}{\lambda_k} P_0 U P_k(x^*) \right\| \\ \le\ & p\frac{|C_{\varphi,-\infty}|}{\sigma}\|P_0 U P_\perp\| = p\frac{|C_{\varphi,-\infty}|}{\sigma}\|P_0(U - U^*)P_\perp\| \\ \le\ & pC_4\|U - U^*\|, \end{aligned} \qquad (6.48)$$

where $C_4 = m\frac{|C_{\varphi,-\infty}|}{\sigma}$. Analogously we obtain

$$\begin{aligned} & \|P_\perp\left(D\Phi_p\left(\mathcal{A}(x^*)\right)[U] - U^*\right) P_\perp\| \\ =\ & \left\| \sum_{k,l=1}^{\mu(x^*)-1} \Delta\varphi_p(\lambda_k, \lambda_l) P_k(x^*) U P_l(x^*) \right\| \\ \le\ & p\varphi'(\sigma/2)\|P_\perp U P_\perp\| = p\varphi'(\sigma/2)\|P_\perp(U - U^*)P_\perp\| \\ \le\ & pC_5\|U - U^*\|, \end{aligned} \qquad (6.49)$$

where $C_5 = mp\varphi'(\sigma/2)$. Finally estimate (6.1) follows from the combination of estimates (6.47) to (6.49) with estimate (6.29). We conclude the proof of assertion (G2) by the following observation: From Theorem 5.1 we get $F'_x(\hat{x}(U^*,p),U^*,p) = 0$ and due to Corollary 5.5 the function $F(x,U^*,p)$ is strongly convex in a neighborhood of $\hat{x}(U^*,p)$. Hence

$$\hat{x}(U^*,p) = \text{argmin}\{F(x,U^*,p)|x \in \mathbb{R}^n\} = x^*$$

and it immediately follows

$$\widehat{U}(U^*,p) = D\Phi_p\left(\mathcal{A}(x^*)\right)[U^*] = \sum_{k=m-r+1}^{m} \varphi'_p(0)P_0U^*P_0 = U^*.$$

It remains to verify (G3) and to complete the proof of (G1). In particular we have to show that

- $F(x,U,p)$ is strongly convex in a neighborhood of $\hat{x}(U,p)$ and
- $\hat{x} = \text{argmin}\{F(x,U,p)|x \in \mathbb{R}^n\}$.

Equations (6.16) and (6.17) show that $\hat{x}$ from Proposition 6.10 satisfies the equation $F'_x(\hat{x},U,p) = 0$. In Corollary 5.5 we have shown that $F(x,U^*,p)$ is strongly convex in a neighborhood of x^*. Therefore we find $\gamma > 0$ such that

$$y^\top F''_{xx}(x^*,U^*,p)y > \gamma\|y\|^2$$

for all $y \in \mathbb{R}^n$. On the other hand, estimate (6.1) shows that for small enough $p_0 > 0$ we have $\hat{x}(U,p)$ near x^* and $\widehat{U}(U,p)$ near U^* uniformly in $\mathcal{V}(U^*,p_0,\delta,\epsilon,\Theta)$. Consequently we find small enough $p_0 > 0$ and $\delta > 0$ such that the inequality

$$y^\top F''_{xx}(\hat{x}(U,p),U,p)y > \frac{1}{2}\gamma\|y\|^2$$

holds for any $(U,p) \in \mathcal{V}(U^*,p_0,\delta,\epsilon,\Theta)$. Hence $F(x,U,p)$ is strongly convex in a neighborhood of $\hat{x}(U,p)$ and the proof of assertion (G3) is complete. Now from (G3) follows that $\hat{x}(U,p)$ is a local minimum of $F(x,U,p)$ in a neighborhood of $\hat{x}(U,p)$ and due to (6.1) also in a neighborhood of x^*. It remains to show that the neighborhood of x^* can be extended to Ω_p and consequently to $\mathbb{R}^n$. We start with the estimate:

$$\begin{aligned} F(\hat{x}(U,p),U,p) &\leq F(x^*,U,p) = f(x^*) + \langle U, \Phi_p\left(\mathcal{A}(x^*)\right)\rangle \\ &= f(x^*) + \langle U, P_0\Phi_p\left(\mathcal{A}(x^*)\right)P_0\rangle + \langle U, P_\perp\Phi_p\left(\mathcal{A}(x^*)\right)P_\perp\rangle \\ &\leq f(x^*) + \langle U, P_0\Phi_p\left(\mathcal{A}(x^*)\right)P_0\rangle = f(x^*). \end{aligned} \tag{6.50}$$

Now suppose that there exists a constant $C > 0$ such that for each $p_0 > 0$ there exists some $(U,p) \in \mathcal{V}(U^*,p_0,\delta,\epsilon,\Theta)$ and some $\tilde{x} \in \Omega_p$ such that

$$F(\tilde{x},U,p) \leq F(\hat{x},U,p) - C.$$

Then from (6.50) we obtain

$$F(\tilde{x},U,p) \leq f(x^*) - C. \tag{6.51}$$

Now let $\Phi_p(\mathcal{A}(\tilde{x})) = S(\tilde{x})\Lambda(\tilde{x})S(\tilde{x})^\top$ and define $\Phi_p(\mathcal{A}(\tilde{x}))^- = S(\tilde{x})\Lambda^-(\tilde{x})S(\tilde{x})^\top$, where $\Lambda^-(\tilde{x}) = \text{diag}(\min(0, \lambda_1(\tilde{x})), \ldots, \min(0, \lambda_m(\tilde{x})))$. Then from (6.51) we see that

$$f(\tilde{x}) \leq f(x^*) - \left\langle U, \Phi_p(\mathcal{A}(\tilde{x}))^- \right\rangle - C.$$

Consequently from assumption $(A5)$ and $\|U\| < \Theta$ we get

$$f(\tilde{x}) \leq f(x^*) - \frac{C}{2}$$

for p_0 small enough and all $p \leq p_0$. On the other hand it is clear that

$$f(\tilde{x}) \geq \min\{f(x)|x \in \Omega_p\}$$

and from the continuity of f follows that

$$\min\{f(x)|x \in \Omega_p\} \geq f(x^*) - \frac{C}{4}$$

for p_0 small enough and thus we obtain

$$f(\tilde{x}) \geq f(x^*) - \frac{C}{4}.$$

This contradiction completes the proof of assertion (G1). Let us summarize our results in the following theorem:

Theorem 6.14 *Let $\mathcal{A}(x)$ be twice continuously differentiable and assumptions $(A1)$ to $(A5)$ hold. Let further $\Theta > \lambda_m(U^*)$ and $0 < \epsilon < \lambda_{m-r+1}(U^*)$. Then there exist $p_0 > 0$ and small enough $\delta > 0$ such that for any $(U, p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$*

a) There exists a vector

$$\hat{x} = \hat{x}(U, p) = \text{argmin}\{F(x, U, p)|x \in \mathbb{R}^n\}$$

such that $F'_x(\hat{x}, U, p) = 0$.

b) For the pair $\hat{x}$ and $\widehat{U} = \widehat{U}(U, p) = D\Phi_p\left(\mathcal{A}(\hat{x}(U, p))\right)[U]$ the estimate

$$\max\left\{\|\hat{x} - x^*\|, \|\widehat{U} - U^*\|\right\} \leq Cp\,\|U - U^*\| \tag{6.52}$$

holds, where C is a constant independent of p.

c) $\hat{x}(U^*, p) = x^*$ and $\widehat{U}(U^*, p) = U^*$.

d) The function $F(x, U, p)$ is strongly convex with respect to x in a neighborhood of $\hat{x}(U, p)$.

Remark . Let us briefly discuss in which way Theorem 6.14 has to be adapted, if we replace assumption (φ_4) in Definition 4.1 by the weaker assumption (φ_4'). It turns out that, while assertions a), b) and d) remain unchanged, the estimate in assertion c) becomes

$$\max\left\{\|\hat{x}-x^*\|, \|\widehat{U}-U^*\|\right\} \le Cp\varphi(-1/p)\,\|U-U^*\|\,. \tag{6.53}$$

Now assumption (φ_4') guarantees that we find p small enough, such that the contractive character of (6.53) can be maintained.

The following Corollary is a direct consequence of Theorem 6.14 c):

Corollary 6.15 *Let ϵ, Θ, δ and p_0 be given as in Theorem 6.14. Then for $p^1 < p_0$ small enough and $(U^1, p^1) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$, Algorithm 6.1.1 converges with a linear rate of convergence. If we choose $p^{k+1} < p^k$ in step (iii) of Algorithm 6.1.1 for all $k \ge 0$ and assume that $p_k \to 0$ for $k \to \infty$ the rate of convergence is superlinear.*

We conclude this chapter by two remarks.

Remark . Let x^+ be a local minimum of problem (SDP) satisfying assumptions (A2) to (A5) and denote by U^+ the corresponding (unique) optimal multiplier. Assume further that there exists $\nu > 0$ such that there is no first order critical point $\tilde{x} \ne x^+$ satisfying $\|\tilde{x}-x^*\| \le \nu$. Then all statements of Theorem 6.14 remain valid, if we replace (x^*, U^*) by (x^+, U^+) and the function $\hat{x}(U,p)$ by

$$\hat{x}_l(U,p) = \operatorname{argmin}\{F(x,U,p) | x \in \mathbb{R}^n, \|x-x^+\| \le \nu\}. \tag{6.54}$$

Moreover Theorem 6.14 d) guarantees the existence of $\eta > 0$ such that $F(x,U,p)$ is strongly convex in $S(x^+, \eta)$ for all $(U,p) \in \mathcal{V}(U^+, p_0, \delta, \epsilon, \Theta)$ and appropriately chosen parameters p_0, δ, ϵ and Θ. Consequently any local descent method applied to the problem

$$(i') \qquad \text{Find } x^{k+1} \text{ such that } \left\|F_x'(x, U^k, p^k\right\| = 0 \tag{6.55}$$

will automatically find a solution, which satisfies the additional constraint $\|x^{k+1} - x^+\| \le \nu$, if it is started with x^1 close enough to x^+. Thus, if the first step in Algorithm 6.1.1 is replaced by step (i') above, the resulting algorithm is guaranteed to converge to (x^+, U^+) with at least linear rate of convergence, provided that $(U^1, p^1) \in \mathcal{V}(U^+, p_0, \delta, \epsilon, \theta)$, $x_1 \in S(x^+, \min\{\eta, \nu\})$ and p^1 is small enough.

Remark . Whenever we solve a problem of type Algorithm 6.1.1(i) in practice, we have to replace the exact minimum by an approximation of certain precision, which can be determined by a finite procedure. In [70] this problem is analyzed for the so called nonlinear rescaling method. We have not explored this problem in the context of nonlinear semidefinite programming so far. Nevertheless we achieve results of high precision with a relatively moderate stopping criterion for the unconstrained minimization in practice (compare Section 9.3.2).

Chapter 7

Globally Convergent Algorithms

In the preceding section we have seen that Algorithm 6.1.1 converges provided the initial iterates are chosen appropriately. Unfortunately in many practical situations no such initial iterates are known and consequently Algorithm 6.1.1 is not applicable. A possible remedy is provided by the following hybrid strategy:

(i) Use a globally convergent algorithm (which is guaranteed to converge to a solution of (SDP) for arbitrary initial iterates) to find $(U, p) \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ with $p < \frac{C}{2}$.

(ii) Apply Algorithm 6.1.1.

In the framework of this section we are going to present two modified versions of Algorithm 6.1.1, which both turn out to be globally convergent under certain assumptions.

7.1 The Shifted Barrier Approach

In our first approach, we use a constant multiplier matrix during all iterations. The initial multiplier is chosen to be the identity matrix. This leads to the following algorithm, which we call shifted barrier approach below.

Algorithm 7.1.1 *Let $x^1 \in \mathbb{R}^n$ and $p^1 > 0$ be given. Then for $k = 1, 2, 3, \ldots$ repeat till a stopping criterium is reached:*

$$\begin{aligned}
(i) \qquad & x^{k+1} = \arg\min_{x \in \mathbb{R}^n} F(x, I_m, p^k) \\
(ii) \qquad & U^{k+1} = D\Phi_p(\mathcal{A}(x^{k+1}))[I_m] \\
(iii) \qquad & p^{k+1} < p^k.
\end{aligned}$$

Remark . Again we are searching for a global minimum of an unconstrained optimization problem in step (i) of Algorithm 7.1.1. From practical point of view such an algorithm is only recommendable for problems of moderate dimension (where global optimization techniques can be applied) or if the underlying problem is convex. In contrast to the preceding section, where we discussed the local convergence properties of Algorithm 6.1.1, the global minimization is essential in the analysis presented in this section.

Definition 7.1 *A class of Shifted Barrier functions $M_\Phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is defined by*

$$M_\Phi(x,p) = \begin{cases} f(x) + \langle I_m, \Phi_p\left(\mathcal{A}(x)\right)\rangle = F_\Phi(x, I_m, p) & \textit{for} \quad x \in \Omega_p \\ \infty & \textit{for} \quad x \notin \Omega_p. \end{cases} \tag{7.1}$$

In order to simplify notation we omit the subscript Φ in the remainder of this section. Along with $M(x,p)$ we define

$$M(x,0) = \lim_{p\to 0} M(x,p) = \begin{cases} f(x) & \text{for} \quad x \in \Omega \\ \infty & \text{for} \quad x \notin \Omega. \end{cases} \tag{7.2}$$

The latter limit can be seen from the following Lemma.

Lemma 7.1 *If Assumption $(A5)$ holds, then there exists $p_0 > 0$ such that*

$$f(x) \geq M(x,p) \geq f(x) - O\left(p\varphi\left(-\frac{1}{p}\right)\right)$$

for all $x \in \Omega$.

Proof . The first inequality follows from $\Phi_p\left(\mathcal{A}(x)\right) \preceq 0$ for all $x \in \Omega$. Assumption $(A5)$ guarantees the existence of $\pi > 0$ such that

$$\lambda_k\left(\mathcal{A}(x)\right) \geq -\pi \text{ for all } x \in \Omega_p \text{ and all } k = 1, \ldots, m.$$

Now the right inequality follows from the properties of φ. □

Next we introduce an additional assumption

$$(A6) \quad \text{There exists } p_0 > 0 \text{ such that } \Omega_p \text{ is a compact set for all } p \leq p_0 \tag{7.3}$$

and formulate the following Proposition.

Proposition 7.2 *Let f and $\mathcal{A}$ be twice continuously differentiable and assume that (A1) to (A4) and (A6) hold.*

a) Then for any $p \leq p_0$ there exists

$$x(p) = \operatorname{argmin}\{M(x,p) | x \in \mathbb{R}^n\}$$

such that

$$M_x'(x(p),p) = 0$$

and $\lim_{p\to 0} f(x(p)) = \lim_{p\to 0} M(x(p),p) = f(x^*)$.

b) The pair $x(p)$ and $U(p) = D\Phi_p\left(\mathcal{A}(x(p))\right)[I]$ converge to (x^*, U^*) as p converges to 0.

Proof . a) The existence of $x(p)$ for all $p \le p_0$ follows immediately from assumption $(A6)$. Moreover $M(x,p)$ is continuous in $\operatorname{int}(\Omega_p)$ and increases infinitely as x approaches the boundary of Ω_p, consequently $M'_x(x(p),p) = 0$. Now consider a convergent subsequence $\{x(p_s)\} \subset \{x(p)\}$ and let $\lim_{p_s \to 0} = \bar{x}$. Using (7.2) it is easy to see that $\bar{x} \in \Omega$. Now, from Lemma 7.1 we see that for any $\epsilon > 0$ we find p_s small enough that

$$f(x(p_s)) - \epsilon \le f(\bar{x}) \le f(\bar{x}) + \operatorname{tr}\Phi_{p_s}\left(\mathcal{A}(\bar{x})\right) + \epsilon \le M(x(p_s),p_s) + 2\epsilon$$

and we obtain:

$$f(x(p_s)) \le M(x(p_s),p_s) + 3\epsilon \le M(x^*,p_s) + 3\epsilon \le f(x^*) + 3\epsilon.$$

Since ϵ was chosen arbitrarily, we get

$$f(\bar{x}) = \lim_{p_s \to 0} f(x(p_s)) \le f(x^*) \text{ and thus } f(\bar{x}) = f(x^*).$$

Consequently for any convergent subsequence $\{x(p_s)\}$ with $p_s \to 0$ we have

$$\lim_{p_s \to 0} f(x(p_s)) = f(\bar{x}) = f(x^*),$$

hence

$$\lim_{p \to 0} f(x(p)) = f(x^*) \text{ and therefore } \lim_{p \to 0} M(x(p),p) = f(x^*).$$

b) Assumptions $(A2)$ to $(A4)$ guarantee that (x^*, U^*) is a unique KKT-pair. Therefore we conclude from a) that $\lim_{p\to 0} x(p) = x^*$. Next we will show that $\lim_{p \to 0} U(p) = U^*$. First we rewrite the multiplier update formula making use of the eigenvalue decomposition $\mathcal{A}(x(p)) = S(x(p))\operatorname{diag}(\lambda_1(x(p)), \ldots, \lambda_m(x(p)))S(x(p))^\top$:

$$\begin{aligned} U(p) &= D\Phi_p\left(\mathcal{A}(x(p))\right)[I] \\ &= S(x(p))\Big(\left[\Delta\varphi(\lambda_i(x(p)), \lambda_j(x(p)))\right]_{i,j=1}^m \bullet S(x(p))^\top I_m S(x(p))\Big) S(x(p))^\top \\ &= \Phi'_p\left(\mathcal{A}(x(p))\right), \end{aligned}$$

where

$$\Phi'_p : \mathbb{S}^m \to \mathbb{S}^m$$

$$\mathcal{A}(x(p)) \mapsto S(x(p)) \begin{pmatrix} \varphi'_p\left(\lambda_1(x(p))\right) & 0 & \ldots & 0 \\ 0 & & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & \varphi'_p\left(\lambda_m(x(p))\right) \end{pmatrix} S(x(p))^\top .$$

From the definitions of $x(p)$ and $U(p)$ and a) we obtain

$$0 = M'_x(x(p),p) = f'(x(p)) + \left[\left\langle \Phi'_p\left(\mathcal{A}(x(p))\right), \mathcal{A}_i(x(p))\right\rangle\right]_{i=1}^n$$

for all $p < p_0$, consequently

$$\begin{aligned} 0 &= \lim_{p\to 0}\Big(f'(x(p)) + \big[\big\langle \Phi_p'\left(\mathcal{A}(x(p))\right), \mathcal{A}_i(x(p))\big\rangle\big]_{i=1}^n\Big) \\ &= f'(x^*) + \left[\left\langle \lim_{p\to 0}\Phi_p'\left(\mathcal{A}(x(p))\right), \mathcal{A}_i\right\rangle\right]_{i=1}^n \end{aligned}$$

and $\lim_{p\to 0}\Phi_p'\left(\mathcal{A}(x(p))\right)$ exists. Furthermore

$$\begin{aligned} \lim_{p\to 0}\big\langle \mathcal{A}(x(p)), \Phi_p'\left(\mathcal{A}(x(p))\right)\big\rangle &= \lim_{p\to 0}\sum_{i=1}^m \lambda_i(x(p))\varphi_p'(\lambda_i(x(p))) \\ &= \sum_{i=1}^m \lambda_i(x^*)\lim_{p\to 0}\varphi_p'(\lambda_i(x(p))) \\ &= \sum_{i=1}^{m-r}\lambda_i(x^*)\underbrace{\lim_{p\to 0}\varphi_p'(\lambda_i(x(p)))}_{=0} = 0. \end{aligned}$$

Now from the uniqueness of the KKT-pair (x^*, U^*) we conclude $\lim\limits_{p\to 0} U(p) = U^*$. $\square$

Next we want to derive upper bounds for the error estimates $\|x(p) - x^*\|$ and $\|U(p) - U^*\|$. Let us therefore define $\Delta x = x(p) - x^*$, $\Delta U = U(p) - U^*$ and a function

$$\begin{aligned} \widetilde{U} : \mathbb{R}^n \times \mathbb{R}_+ &\to \mathbb{S}_+^m \\ (x,p) &\mapsto D\Phi_p\left(\mathcal{A}(x)\right)[I]. \end{aligned}$$

Now, using Taylor approximations in local neighborhoods of x^* and the notation introduced in Chapter 5 we obtain the following formulas:

$$\mathcal{A}'{}_i(x) = \mathcal{A}_i + \sum_{j=1}^n \mathcal{A}_{i,j}\Delta x_j + H_i^{\mathcal{A}}(\Delta x), \qquad \text{for all } i = 1,\dots,m, \tag{7.4}$$

where $H_i^{\mathcal{A}} : \mathbb{R}^n \to \mathbb{S}^m$, $H_i^{\mathcal{A}}(0) = 0$ and $\|H_i^{\mathcal{A}}\| = o\left(\|\Delta x\|\right)$ for all $i = 1,\dots,n$,

$$\begin{aligned} \widetilde{U}(x,p) &= \widetilde{U}(x^*,p) + \sum_{j=1}^n\left(\frac{\partial}{\partial x_j}\widetilde{U}(x^*,p)\right)\Delta x_j + H^{\widetilde{U}}(\Delta x) \qquad (7.5)\\ &= S\ \mathrm{diag}\left(\varphi_p'(\lambda_1(x^*)),\dots,\varphi_p'(\lambda_{m-r}(x^*)),1,\dots,1\right)S^\top + \\ &\quad \sum_{j=1}^n S\left(\left[\Delta\varphi_p'(\lambda_k(x^*),\lambda_l(x^*))\right]_{k,l=1}^m \bullet S^\top\mathcal{A}_jS\right)S^\top\Delta x_j + H^{\widetilde{U}}(\Delta x), \end{aligned}$$

where $H^{\widetilde{U}} : \mathbb{R}^n \to \mathbb{S}^m$, $H^{\widetilde{U}}(0) = 0$ and $\|H^{\widetilde{U}}\| = o\left(\|\Delta x\|\right)$ and finally

$$f'(x) = f'(x^*) + f''(x^*)\Delta x + H^f(\Delta x), \tag{7.6}$$

where $H^f : \mathbb{R}^n \to \mathbb{R}^n$, $H^f(0) = 0$ and $\|H^f\| = o\left(\|\Delta x\|\right)$.

Lemma 7.3

a) $P_0\widetilde{U}(x^*,p)P_\perp = 0.$

b) $\|P_\perp\widetilde{U}(x^*,p)P_\perp\| = O(p).$

c) $P_0\widetilde{U}(x^*,p)P_0 = P_0.$

Proof .

a) $P_\perp\widetilde{U}(x^*,p)P_0 = SD_\perp\mathrm{diag}(\underbrace{\varphi_p'(\lambda_1(x^*)),\dots,\varphi_p'(\lambda_{m-r}(x^*))}_{m-r},\underbrace{1,\dots,1}_{r})D_0S^\top = 0.$

b) $\left\|P_\perp\widetilde{U}(x^*,p)P_\perp\right\| = \left\|\sum_{k=1}^{m-r}\varphi_p'(\lambda_k(x^*))s_ks_k^\top\right\| \le \sum_{k=1}^{m-r}\varphi_p'(\lambda_k(x^*)) = O(p).$

c) $P_0\widetilde{U}(x^*,p)P_0 = \sum_{k=m-r+1}^{m}\varphi_p'(\lambda_k(x^*))s_ks_k^\top = P_0.$ □

Lemma 7.4

a) $\left\|P_\perp\frac{\partial}{\partial x_i}\widetilde{U}(x^*,p)P_\perp\right\| = O(p)$ *for all* $i=1,\dots,n.$

b) $\lim_{p\to 0}P_0\frac{\partial}{\partial x_i}\widetilde{U}(x^*,p)P_\perp = -\sum_{l=1}^{\mu(x^*)-1}(\lambda_l)^{-1}P_0\mathcal{A}_iP_l(x^*)$ *and*

$\lim_{p\to 0}P_\perp\frac{\partial}{\partial x_i}\widetilde{U}(x^*,p)P_0 = -\sum_{l=1}^{\mu(x^*)-1}(\lambda_l)^{-1}P_l(x^*)\mathcal{A}_iP_0$ *for all* $i=1,\dots,n.$

c) $P_0\frac{\partial}{\partial x_i}\widetilde{U}(x^*,p)P_0 = p^{-1}\varphi''(0)P_0\mathcal{A}_iP_0$ *for all* $i=1,\dots,n.$

Proof . For all $i=1,\dots,n$ we have:

a)
$$\begin{aligned}\left\|P_\perp\frac{\partial}{\partial x_i}\widetilde{U}(x^*,p)P_\perp\right\| &= \left\|D_\perp\left(\left[\Delta\varphi_p'(\lambda_l(x^*),\lambda_k(x^*))\right]_{l,k=1}^m \bullet S^\top\mathcal{A}_iS\right)D_0\right\| \\ &= \left\|\left[\Delta\varphi_p'(\lambda_l(x^*),\lambda_k(x^*))\right]_{l,k=1}^{m-r}\bullet\left[S^\top\mathcal{A}_iS\right]_{l,k=1}^{m-r}\right\| = O(p),\end{aligned}$$

since $\Delta\varphi_p'(\lambda_l(x^*),\lambda_k(x^*)) \le \varphi_p''(\lambda_{m-r}(x^*)) \le \varphi_p''(\sigma) = O(p).$

b) $P_0\frac{\partial}{\partial x_i}\widetilde{U}(x^*,p)P_\perp = P_0\left(\sum_{k,l=1}^{\mu(x^*)}\varphi_p'(\lambda_l(x^*),\lambda_k(x^*))P_k(x^*)\mathcal{A}_iP_l(x^*)\right)P_\perp$

$$= \sum_{l=1}^{\mu(x^*)-1}\frac{1-\varphi_p'(\lambda_l(x^*))}{0-\lambda_l(x^*)}P_0\mathcal{A}_iP_l(x^*) \overset{p\to 0}{\to} -\sum_{l=1}^{\mu(x^*)-1}(\lambda_l)^{-1}P_0\mathcal{A}_iP_l(x^*).$$

The second limit follows analogously.

c) $P_0 \dfrac{\partial}{\partial x_i} \widetilde{U}(x^*, p) P_0 = P_0 \left(\sum_{k,l=1}^{\mu(x^*)} \varphi_p'(\lambda_l(x^*), \lambda_k(x^*)) P_k(x^*) \mathcal{A}_i P_l(x^*) \right) P_0$
$= \varphi_p''(0) P_0 \mathcal{A}_i P_0.$ □

The following proposition provides us with error estimates for $x(p)$ and the "active part" of $U(p)$.

Proposition 7.5 *Let f and $\mathcal{A}$ be twice continuously differentiable and assume that (A1) to (A4) and (A6) are satisfied. Then the estimate*

$$\max\left\{ \|x(p) - x^*\|, \|E_0^\top U(p) E_0 - U_0^*\| \right\} \le C_6 p \tag{7.7}$$

holds, where C_6 is a constant independent of p.

Proof . Using the Taylor approximations (7.4) to (7.6), we obtain

$$\begin{aligned}
0 \quad = \quad & M(x(p), p) = f'(x^*) + f''(x^*)\Delta x + H^f(\Delta x) + \\
& \left[\left\langle P_0 (\underbrace{\Delta U + U^*}_{=U(p)}) P_0, \mathcal{A}_i + \sum_{j=1}^n \mathcal{A}_{i,j} \Delta x_j + H_i^{\mathcal{A}}(\Delta x) \right\rangle \right]_{i=1}^n + \\
& \left[\left\langle P_0 U(p) P_\perp + P_\perp U(p) P_0 + P_\perp U(p) P_\perp, \mathcal{A}_i'(x(p)) \right\rangle \right]_{i=1}^n \\
= \quad & f'(x^*) + \left[\left\langle P_0 U^* P_0, \mathcal{A}_i \right\rangle \right]_{i=1}^n + \left(f''(x^*) + \left[\left\langle P_0 U^* P_0, \mathcal{A}_{i,j} \right\rangle \right]_{i,j=1}^n \right) \Delta x + \\
& \left[\left\langle P_0 \Delta U P_0, \mathcal{A}_i \right\rangle \right]_{i=1}^n + H^f(\Delta x) + \left[\left\langle P_0 \left(\Delta U + U^* \right) P_0, H_i^{\mathcal{A}}(\Delta x) \right\rangle \right]_{i=1}^n + \\
& \left[\left\langle P_0 \Delta U P_0, \sum_{j=1}^n \mathcal{A}_{i,j} \Delta x_j \right\rangle \right]_{i=1}^n + \\
& \left[\left\langle P_0 U(p) P_\perp + P_\perp U(p) P_0 + P_\perp U(p) P_\perp, \mathcal{A}_i'(x(p))) \right\rangle \right]_{i=1}^n \\
= \quad & \left(f''(x^*) + \left[\left\langle P_0 U^* P_0, \mathcal{A}_{i,j} \right\rangle \right]_{i,j=1}^n \right) \Delta x + \left[\left\langle P_0 \Delta U P_0, \mathcal{A}_i \right\rangle \right]_{i=1}^n + \\
& H^f(\Delta x) + \left[\left\langle \Delta U + U^*, H_i^{\mathcal{A}}(\Delta x) \right\rangle \right]_{i=1}^n + \left[\left\langle \Delta U, \sum_{j=1}^n \mathcal{A}_{i,j} \Delta x_j \right\rangle \right]_{i=1}^n + \\
& \left[\left\langle P_0 U(p) P_\perp + P_\perp U(p) P_0 + P_\perp U(p) P_\perp, \mathcal{A}_i \right\rangle \right]_{i=1}^n \\
= \quad & \left(f''(x^*) + \left[\left\langle P_0 U^* P_0, \mathcal{A}_{i,j} \right\rangle \right]_{i,j=1}^n \right) \Delta x + \left[\left\langle P_0 \Delta U P_0, \mathcal{A}_i \right\rangle \right]_{i=1}^n + \\
& R_1(p) + R_2(p),
\end{aligned} \tag{7.8}$$

where

$$\begin{aligned} R_1(p) &= \Big[\Big\langle P_0 U(p) P_\perp + P_\perp U(p) P_0 + P_\perp U(p) P_\perp, \mathcal{A}_i \Big\rangle\Big]_{i=1}^n, \\ R_2(p) &= H^f(\Delta x) + \Big[\Big\langle \Delta U + U^*, H_i^{\mathcal{A}}(\Delta x)\Big\rangle\Big]_{i=1}^n + \Big[\Big\langle \Delta U, \sum_{j=1}^n \mathcal{A}_{i,j}\Delta x_j \Big\rangle\Big]_{i=1}^n \end{aligned}$$

and $\|R_2(p)\|$ is of order $o(\|\Delta x\|)$, which can be seen from the formulas (7.4) to (7.6) and the fact that

$$\Delta U = \sum_{j=1}^n \left(\frac{\partial}{\partial x_j}\widetilde{U}(x^*, p)\right)\Delta x_j + H^{\widetilde{U}}(\Delta x).$$

Further, using Lemma 7.3 and Lemma 7.4 we get

$$\begin{aligned} R_1(p) &= \Big[\Big\langle P_0 U(x^*,p) P_\perp + P_\perp U(x^*,p) P_0, \mathcal{A}_i \Big\rangle\Big]_{i=1}^n + \qquad (7.9) \\ & \Big[\Big\langle P_\perp U(x^*,p) P_\perp, \mathcal{A}_i \Big\rangle\Big]_{i=1}^n + \\ & \Big[\Big\langle P_\perp \Big(\sum_{j=1}^n \Delta x_j \frac{\partial}{\partial x_j} U(x^*,p)\Big) P_\perp, \mathcal{A}_i \Big\rangle\Big]_{i=1}^n + \\ & \Big[\Big\langle P_0 \Big(\sum_{j=1}^n \Delta x_j \frac{\partial}{\partial x_j} U(x^*,p)\Big) P_\perp, \mathcal{A}_i \Big\rangle\Big]_{i=1}^n + \\ & \Big[\Big\langle P_\perp \Big(\sum_{j=1}^n \Delta x_j \frac{\partial}{\partial x_j} U(x^*,p)\Big) P_0, \mathcal{A}_i \Big\rangle\Big]_{i=1}^n + \\ & \Big[\Big\langle H^U(\Delta x), (P_0 \mathcal{A}_i P_\perp + P_\perp \mathcal{A}_i P_0 + P_\perp \mathcal{A}_i P_\perp \Big\rangle\Big]_{i=1}^n \\ &= R_3(p) + R_4(p) + M(p)\Delta x, \qquad (7.10) \end{aligned}$$

where $\|R_3(p)\|$ is a term of order $o(\|\Delta x\|)$, $\|R_4(p)\|$ is a term of order $O(p)$ and

$$\begin{aligned} M(p) &= \Big[\Big\langle P_0 \frac{\partial}{\partial x_j} U(x^*,p) P_\perp + P_\perp \frac{\partial}{\partial x_j} U(x^*,p) P_0, \mathcal{A}_i \Big\rangle\Big]_{i,j=1}^n \\ &\overset{p\to 0}{\longrightarrow} \Big[-2 \sum_{l=1}^{\mu(x^*)-1} \Big(\lambda_l\Big)^{-1} \Big\langle P_0 \mathcal{A}_i P_l(x^*), \mathcal{A}_j \Big\rangle\Big]_{i,j=1}^n \\ &= -2\Big[\Big\langle P_0, \mathcal{A}_i \sum_{l=1}^{\mu(x^*)-1} \Big(\lambda_l\Big)^{-1} s_l s_l^\top \mathcal{A}_j \Big\rangle\Big]_{i,j=1}^n \\ &= -2\Big[\Big\langle P_0, \mathcal{A}_i \mathcal{A}^\dagger(x^*) \mathcal{A}_j \Big\rangle\Big]_{i,j=1}^n. \qquad (7.11) \end{aligned}$$

Combining (7.8), (7.10) and (7.11), defining $\Delta u = \textbf{svec}(\Delta U)$ and recalling that $\Delta x \to 0$ for $p \to 0$ we obtain the equation

$$\Big[L''_{xx}(x^*, U^*) + M(p)\Big]\Delta x + \Big[\textbf{svec}(P_0\mathcal{A}_i P_0)\Big]_{i=1}^{n}\Delta u = R(p), \tag{7.12}$$

where the right hand side term $R(p)$ is of order $O(p)$. On the other hand from the Taylor approximation for $\widetilde{U}(x,p)$ we see

$$\begin{aligned} E_0^\top U(p) E_0 &= E_0^\top \widetilde{U}(x^*, p)E_0 + \sum_{j=1}^{n} \Delta x_j E_0^\top \frac{\partial}{\partial x_j}\widetilde{U}(x^*, p)E_0 + E_0^\top H^U(\Delta x)E_0 \\ &= I_r + \frac{\varphi''(0)}{p}\sum_{j=1}^{n}\Delta x_j E_0^\top \mathcal{A}_j E_0 + E_0^\top H^U(\Delta x)E_0. \end{aligned} \tag{7.13}$$

Using the abbreviations $\Delta u_0 = \textbf{svec}(E_0^\top \Delta U E_0)$ and $N = \big(\big[\textbf{svec}\,\big(E_0^\top \mathcal{A}_i E_0\big)\big]_{i=1}^{n}\big)^\top$ we get from (7.13)

$$-p\Delta u_0 + \varphi''(0)N\Delta x = \underbrace{-pE_0^\top H^U(\Delta x)E_0 + p\,\textbf{svec}\,(U_0^* - I_r)}_{=Q(p)}, \tag{7.14}$$

where $Q(p)$ is of order $O(p)$. Finally, combining (7.12) and (7.14) we obtain

$$D(p)\begin{pmatrix}\Delta x\\ \Delta u\end{pmatrix} = \begin{pmatrix} L''_{xx}(x^*, U^*) + M(p) & N \\ N^\top & -pI_r \end{pmatrix}\begin{pmatrix}\Delta x\\ \Delta u\end{pmatrix} = \begin{pmatrix} R(p)\\ Q(p)\end{pmatrix}. \tag{7.15}$$

Recalling that $(M(p))_{i,j} = -2\langle P_0, \mathcal{A}_i\mathcal{A}^\dagger(x^*)\mathcal{A}_j\rangle$ and taking into account that P_0 has the same eigenvectors and the same non-zero structure of eigenvalues as U^*, we can prove exactly in the same way as it was done in the proof of Lemma 6.8 that there exists $\kappa_0 > 0$ such that

$$\big\|D^{-1}(p)\big\| \le \kappa_0, \text{ for all } p < p_0.$$

Now from (7.15) we obtain the estimate

$$\max\left\{\|x(p) - x^*\|, \|u_0(p) - u_0^*\|\right\} \le C_6 p \tag{7.16}$$

for a constant $C_6 > 0$ independent of p. □

The following Lemma provides an error estimate for the “inactive” part of U.

Lemma 7.6

$$\|P_0U(p)P_\perp + P_\perp U(p)P_0 + P_\perp U(p)P_0\| \le pC_7.$$

for a constant $C_7 > 0$ independent of p.

Proof . We start with the inequality

$$\|P_0U(p)P_\perp + P_\perp U(p)P_0 + P_\perp U(p)P_0\| \leq 2\,\|P_0U(p)P_\perp\| + \|P_\perp U(p)P_\perp\|\,.$$

From Lemma 7.3, Lemma 7.4 and (7.6) we know that $\|P_\perp U(p)P_\perp\|$ is of order $O(p)$. Now let $S(p)\,\mathrm{diag}(\lambda_1(p),\ldots,\lambda_m(p))S(p)^\top$ be an eigenvalue decomposition of $\mathcal{A}(x(p))$, denote by $s_l(p)$ the l-th column of $S(p)$ and consider a single entry of the matrix $P_0U(p)P_\perp$:

$$(P_0U(p)P_\perp)_{i,j} = s_i^\top U(p)s_j = \sum_{l=1}^{m} s_i^\top s_l(p)\varphi_p'(\lambda_l(p))s_l(p)^\top s_j. \tag{7.17}$$

Since $i \in \{m-r+1,\ldots,m\}$ and $j \in \{1,\ldots,m-r\}$ we find for each $l \in \{1,\ldots,m\}$ in the sum at the right hand side of equation (7.17) a product of type

$$s_i^\top s_l(p),\ \text{where } l \in \{1,\ldots,m-r\}$$

or

$$s_l(p)^\top s_j,\ \text{where } l \in \{m-r+1,\ldots,m\}.$$

Now, using the same arguments as in the proof of Lemma 6.13 we obtain

$$\left\|s_i^\top s_l(p)\right\| = s_i^\top s_l(p)s_l(p)^\top s_i \leq \mathrm{tr}\left(E_0^\top P_{x^*,\perp}E_0\right) = O(p)$$

and

$$\left\|s_j^\top s_l(p)\right\| = s_j^\top s_l(p)s_l(p)^\top s_j \leq \mathrm{tr}\left(E_\perp^\top P_{x^*,0}E_\perp\right) = O(p).$$

Taking further into account that $\varphi_p'(\lambda_l(p)) \to \lambda_l(U^*)$ for $p \to 0$ and all $l = 1,\ldots,m$ we get $\varphi_p'(\lambda_l(p)) \leq 2\lambda_l(U^*)$ for all $l = 1,\ldots,m$ and p small enough. Finally we conclude from (7.17) and the estimates above that $P_0U(p)P_\perp = O(p)$ and the assertion of Lemma 7.6 follows. □

We conclude this section with the following Theorem, which summarizes the convergence properties of Algorithm 7.1.1:

Theorem 7.7 *Let f and $\mathcal{A}$ be twice continuously differentiable and suppose that assumption (A6) holds. Then*

a) For any $p \leq p_0$ there exists

$$x(p) = \mathrm{argmin}\{M(x(p),p) | x \in \mathbb{R}^n\}$$

such that

$$M_x'(\hat{x},p) = 0$$

and $\lim\limits_{p\to 0} f(x(p)) = \lim\limits_{p\to 0} M(x(p),p) = f(x^*).$

If moreover conditions $(A3)$ and $(A4)$ are satisfied, then *b) the estimate*

$$\max\{\|x(p)-x^*\|, \|U(p)-U^*\|\} \le C_8 p \tag{7.18}$$

holds for the pair $x(p)$ and $U(p) = D\Phi_p(\mathcal{A}(x(p)))[I]$, where C_8 is a constant independent of p.

Proof . a) follows from Proposition 7.2.

b) follows from Proposition 7.5 and Lemma 7.6. □

7.2 The General Approach

Our second globalization approach deals with the general case, where we are faced with potential non-convexity of f and/or $\mathcal{A}$. In this situation we can not expect to be able to find global minima in step (i) of Algorithm 6.1.1. Thus the unconstrained minimization process is replaced by the (approximative) search for a first order critical point of problem (SDP). The result is the following Algorithm:

Algorithm 7.2.1 *Let $x^0 \in \mathbb{R}^n, U^0 \in \mathbb{S}^m_{++}, p^0 > 0$ and $\epsilon_0 > 0$ be given. Then for $k = 0, 1, 2, \ldots$ repeat till a stopping criterium is reached:*

$$\begin{aligned}
&(i) && \textit{Find } x^{k+1} \textit{ such that } \left\|F_x'(x^{k+1}, I_m, p^k)\right\| \le \epsilon_k\\
&(ii) && U^{k+1} = D\Phi_p(\mathcal{A}(x))\left[U^k\right]\\
&(iii) && p^{k+1} < p^k, \epsilon^{k+1} < \epsilon^k.
\end{aligned}$$

As a direct consequence of the modifications, which lead to Algorithm 7.2.1, we will no longer be able to guarantee that the sequence $\{x^k\}_{k\in\mathbb{N}}$ generated by Algorithm 7.2.1 converges. Therefore we have to focus on convergent subsequences: Let $\{x^{k_l}\}_{l\in\mathbb{N}}$ be a convergent subsequence of $\{x^k\}_{k\in\mathbb{N}}$. The existence of at least one convergent subsequence is guaranteed by assumption (A6), which we assume to be satisfied throughout this section. Let further $\tilde{x} = \lim_{l\to\infty} x^{k_l}$ and $\tilde{S}\tilde{\Lambda}\tilde{S}^\top = \mathcal{A}(\tilde{x})$,where

$$\tilde{\Lambda} = \tilde{\Lambda}(\mathcal{A}(\tilde{x})) = \operatorname{diag}\left(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_{m-r}, \tilde{\lambda}_{m-r+1}, \ldots, \tilde{\lambda}_m\right),$$

$\tilde{\lambda}_i$ denote the increasingly ordered eigenvalues of $\mathcal{A}(\tilde{x})$ for all $i = 1, \ldots, m$ and we assume without loss of generality that there exists $\sigma < 0$ such that

$$\tilde{\lambda}_1 \le \tilde{\lambda}_2 \le \ldots \le \tilde{\lambda}_{m-r} < \sigma < 0 \le \tilde{\lambda}_{m-r+1} \le \ldots \le \tilde{\lambda}_m.$$

Further, the i-th column of $\tilde{S}$ is denoted by $\tilde{s}_i$ for all $i = 1, \ldots, m$ and we define in analogy to Chapter 5:

$$P_0 = E_0E_0^\top = \tilde{S}D_0\tilde{S}^\top = \sum_{i=m-r+1}^{m} \tilde{s}_l\tilde{s}_l^\top \text{ and } P_\perp = E_\perp E_\perp^\top = \tilde{S}D_\perp\tilde{S}^\top = \sum_{i=1}^{m-r} \tilde{s}_l\tilde{s}_l^\top.$$

Using these definitions, we reformulate assumption $(A3)$ in the following way:

$(A3')$ The nondegeneracy constraint qualification holds at the limit point $\tilde{x}$ of the sequence $\{x^{k_l}\}_{l\in\mathbb{N}}$.

Moreover we introduce two additional assumptions on problem (SDP):

$(A7)$ The sequence of the Lagrange multiplier estimates generated by Algorithm 7.2.1 stays bounded.

$(A8)$ $U^1 \in \mathbb{S}^m_{++}$.

Assumption $(A7)$ is frequently used in the analysis of augmented Lagrangian type methods (see, for example, [12]). Next we prove that each cluster point of the sequence $(x^k, U^k)_{k\in\mathbb{N}}$ generated by Algorithm 7.2.1 converges to a first order critical point of problem (SDP). The proof is divided into several steps. We start with the following Proposition, which guarantees that Algorithm 7.2.1 is well defined:

Proposition 7.8 *Suppose that assumption (A6) holds. Let further f and $\mathcal{A}$ be continuously differentiable. Then there exists $p_0 > 0$ such that for all $p < p_0$ and all $U \succ 0$ there exists at least one $x \in \operatorname{int}(\Omega_p)$ with $F'_x(x, U, p) = 0$.*

Proof . The assertion follows immediately from the compactness of Ω_p and the fact that F increases infinitely as x approaches the boundary of Ω_p. $\square$

Next we want to show that the sequence of multipliers $\{U^{k_l}\}_{l\in\mathbb{N}}$ converges. We start with the following Lemma:

Lemma 7.9 $U^k \succ 0$ *for all* $k \geq 0$.

Proof . Assumption $(A8)$ guarantees that we find some $k_0 \geq 0$ such that

$$U^k \succ 0 \text{ for all } k \leq k_0. \tag{7.19}$$

Now, using the abbreviations $S(x^k) = S\left(\mathcal{A}(x^k)\right)$ and $\lambda_i(x^k) = \lambda_i\left(\mathcal{A}(x^k)\right)$ for all $i = 1, \ldots, m$ we obtain

$$U^k = S(x^k)\left(\left[\Delta\varphi(\lambda_i(x^k), \lambda_j(x^k))\right]_{i,j=1}^m \bullet S(x^k)^\top U^{k-1} S(x^k)\right) S(x^k)^\top.$$

The matrix $S(x^{k_l})^\top U^k S(x^{k_l})$ is positive definite due to assumption (7.19). The matrix $\left[\Delta\varphi(\lambda_i(x^{k_l}), \lambda_j(x^{k_l}))\right]_{i,j=1}^m$ is positive semidefinite due to the monotonicity of Φ. Now the positive definiteness of U^{k+1} follows for example from Theorem 5.3.6 in [45]. $\square$

Lemma 7.10 *Let $(B^k)_{k\in\mathbb{N}} \in (\mathbb{S}^m_+)^{\mathbb{N}}$ be a sequence of positive semidefinite matrices. Then the following implication holds true:*

$$\lim_{k\to\infty} \operatorname{tr}(B^k P_\perp) = 0 \Rightarrow \lim_{k\to\infty} \left\|B^k P_\perp\right\| = 0.$$

Proof . Let $S_{B^k}\Lambda_{B^k}S_{B^k}^\top$ be an eigenvalue decomposition of B^k and denote by $\lambda_{B^k,i}$ for all $i = 1,\dots,m$ the corresponding eigenvalues. Then we have

$$\begin{aligned} \mathrm{tr}(B^kP_\perp) &= \mathrm{tr}(E_\perp^\top B^k E_\perp) = \mathrm{tr}(E_\perp^\top S_{B^k}\Lambda_{B^k}S_{B^k}^\top E_\perp) \\ &= \sum_{i=1}^{m-r} s_i^\top S_{B^k}\Lambda_{B^k}S_{B^k}^\top s_i = \sum_{i=1}^{m-r}(s_i^\top S_{B^k})\Lambda_{B^k}(s_i^\top S_{B^k})^\top \\ &= \sum_{i=1}^{m-r}\sum_{j=1}^{m}\lambda_{B^k,j}(s_i^\top s_{B^k,j})^2. \end{aligned}$$

Since all terms are positive we get from $\lim_{k\to\infty}\mathrm{tr}(B^kP_\perp) = 0$ that for each pair $(i,j) \in \{1,\dots,m-r\}\times\{1,\dots,m\}$

$$\lambda_{B^k,j}(s_i^\top s_{B^k,j})^2 \to 0 \quad \text{for } k\to\infty$$

or equivalently

$$\lambda_{B^k,j} \to 0 \text{ for } k\to\infty \quad \vee \quad s_i^\top s_{B^k,j} \to 0 \text{ for } k \to \infty. \tag{7.20}$$

On the other hand we derive:

$$\begin{aligned} \left\|B^kP_\perp\right\|^2 &= \mathrm{tr}(P_\perp(B^k)^2P_\perp) = \mathrm{tr}(E_\perp^\top(B^k)^2E_\perp) = \mathrm{tr}(E_\perp^\top S_{B^k}\Lambda_{B^k}^2 S_{B^k}^\top E_\perp) \\ &= \sum_{i=1}^{m-r} s_i^\top S_{B^k}\Lambda_{B^k}^2 S_{B^k}^\top s_i = \sum_{i=1}^{m-r}(s_i^\top S_{B^k})\Lambda_{B^k}^2(s_i^\top S_{B^k})^\top \\ &= \sum_{i=1}^{m-r}\sum_{j=1}^{m}\lambda_{B^k,j}^2(s_i^\top s_{B^k,j})^2. \end{aligned}$$

Finally we conclude from (7.20) that $\left\|B^kP_\perp\right\|^2 \to 0$ for $k\to\infty$. □

Now we are able to prove the following Proposition.

Proposition 7.11 $U_\perp^{k_l} = P_\perp U^{k_l}P_\perp + P_0U^{k_l}P_\perp + P_\perp U^{k_l}P_0 \to \mathbb{O}_m$ *for* $l\to\infty$.

Proof . We consider the definition of U^{k_l} in the following form

$$U^{k_l} = S(x^{k_l})\left(\left[\Delta\varphi(\lambda_i(x^{k_l}),\lambda_j(x^{k_l})\right]_{i,j=1}^m \bullet S(x^{k_l})^\top U^{k_l-1}S(x^{k_l})\right)S(x^{k_l})^\top.$$

From assumption (A7) follows immediately that the norm of the matrix

$$M(x^{k_l}) := \left[\Delta\varphi(\lambda_i(x^{k_l}),\lambda_j(x^{k_l})\right]_{i,j=1}^m \bullet S(x^{k_l})^\top U^{k_l-1}S(x^{k_l})$$

is bounded from above. Furthermore for $i,j \le m-r$ we obtain for l large enough:

$$\Delta\varphi(\lambda_i(x^{k_l}),\lambda_j(x^{k_l})) \le \varphi'\left(\frac{\sigma}{2}\right) = O(p). \tag{7.21}$$

Now U^{k_l} can be written in the following abstract from:

$$U^{k_l} = S(x^{k_l})\begin{pmatrix} Z_1(x^{k_l}) & Z_2(x^{k_l}) \\ Z_2(x^{k_l})^\top & Z_3(x^{k_l}) \end{pmatrix} S(x^{k_l})^\top,$$

where $Z_1(x^{k_l}) \in \mathbb{S}^{m-r}$ tends to 0 for $l \to \infty$ and the norms of $Z_2(x^{k_l}) \in \mathbb{M}^{m-r,r}$ and $Z_3(x^{k_l}) \in \mathbb{S}^{r,r}$ are restricted. Due to $S(x^{k_l}) \to S$ for $l \to \infty$ we get

$$S^\top S(x^{k_l}) \to I_m \text{ and } S(x^{k_l})^\top S \to I_m \text{ for } l \to \infty$$

and

$$\begin{aligned} \langle P_\perp, U^{k_l} \rangle &= \operatorname{tr}\left(S D_\perp S^\top S(x^{k_l}) \begin{pmatrix} Z_1(x^{k_l}) & Z_2(x^{k_l}) \\ Z_2(x^{k_l})^\top & Z_3(x^{k_l}) \end{pmatrix} S(x^{k_l})^\top S D_\perp S^\top \right) \\ &\to \operatorname{tr}\left(D_\perp \begin{pmatrix} 0 & Z_2(x^{k_l}) \\ Z_2(x^{k_l})^\top & Z_3(x^{k_l}) \end{pmatrix} D_\perp \right) = 0 \end{aligned}$$

for $l \to \infty$. Now the assertion of Proposition 7.11 follows from Lemma 7.10. □

The following proposition states convergence of the sequence $\{U^{k_l}\}_{l\in\mathbb{N}}$.

Proposition 7.12 *The sequence* $\left\{U_0^{k_l}\right\}_{l\in\mathbb{N}} = \{P_0 U^{k_l} P_0\}_{l\in\mathbb{N}}$ *converges.*

Proof . We start with the following formula for the partial derivative of F with respect to x:

$$\begin{aligned} F'_x(x^{k_l}, U^{k_l-1}, p^{k_l-1}) &= f'\left(x^{k_l}\right) + \left[\langle P_0 \mathcal{A}'_i(x^{k_l}) P_0, U^{k_l} \rangle\right]_{i=1}^n + \\ &\quad \left[\left\langle \mathcal{A}'_i(x^{k_l}), P_\perp U^{k_l} P_0 + P_0 U^{k_l} P_\perp + P_\perp U^{k_l} P_\perp \right\rangle\right]_{i=1}^n \\ &= f'\left(x^{k_l}\right) + \left[\langle E_0^\top \mathcal{A}'_i(x^{k_l}) E_0, E_0^\top U^{k_l} E_0 \rangle\right]_{i=1}^n + R^{k_l} \\ &= f'\left(x^{k_l}\right) + R^{k_l} + \left(D^{k_l}\right)^\top \textbf{svec}\left(E_0^\top U^{k_l} E_0\right), \end{aligned} \tag{7.22}$$

where

$$R^{k_l} = \left[\left\langle \mathcal{A}'_i(x^{k_l}), P_\perp U^{k_l} P_0 + P_0 U^{k_l} P_\perp + P_\perp U^{k_l} P_\perp \right\rangle\right]_{i=1}^n$$

and

$$D^{k_l} = \left[\textbf{svec}\left(E_0^\top \mathcal{A}'_i(x^{k_l}) E_0\right)\right]_{i=1}^n .$$

Using assumption $(A3')$, we find l_0 large enough that D^{k_l} has maximal rank for all $l \geq l_0$. Hence we can rewrite equation (7.22) as

$$\textbf{svec}\left(E_0^\top U^{k_l} E_0\right) = \left[D^{k_l} D^{k_l T}\right]^{-1} D^{k_l} \left(f'\left(x^{k_l}\right) + R^{k_l} + F'_x(x^{k_l}, U^{k_l-1}, p^{k_l-1})\right).$$

Now from Proposition 7.11 and the fact that $\|F'_x(x^{k_l}, U^{k_l-1}, p^{k_l-1})\| \leq \epsilon^{k_l}$ and $\epsilon^{k_l} \to 0$ for $l \to \infty$ we obtain

$$\textbf{svec}\left(E_0^\top U^{k_l} E_0\right) \to -\left[D D^\top\right]^{-1} D f'(\tilde{x}) \text{ for } l \to \infty,$$

where $D = \lim_{l\to\infty} D^{k_l} = \left[\textbf{svec}\left(E_0^\top \mathcal{A}_i E_0\right)\right]_{i=1}^n$ and therefore

$$\lim_{l\to\infty} P_0 U^{k_l} P_0 = -E_0 \,\textbf{smat}\left(\left[D D^\top\right]^{-1} D f'(\tilde{x})\right) E_0^\top .$$

□

Now we are able to state the following convergence result:

Theorem 7.13 *Suppose that assumptions (A3'), (A4), (A5), (A6'), (A7) and (A8) hold. Let further f and $\mathcal{A}$ be twice continuously differentiable and assume that $\lim_{k\to\infty} p^k = \lim_{k\to\infty} \epsilon^k = 0$. Then the limit point of any convergent subsequence of the sequence $(x^k, U^k)_{k\in\mathbb{N}}$ generated by Algorithm 7.2.1 is a KKT-point of problem (SDP).*

Proof. Let $(x^{k_l}, U^{k_l})_{l\in\mathbb{N}}$ be a convergent subsequence of $(x^k, U^k)_{k\in\mathbb{N}}$ and let

$$\lim_{l\to\infty}(x^{k_l}, U^{k_l}) = (\tilde{x}, \widetilde{U}).$$

Then the stationarity of $(\tilde{x}, \widetilde{U})$ follows directly from step (i) of Algorithm 7.2.1 and the fact that $\epsilon^{k_l} \to 0$ for $l \to \infty$. The feasibility of $\tilde{x}$ follows from $F(x, U^{k_l}, p^{k_l}) = \infty$ for all $x \notin \Omega_{p^{k_l}}$ and $p^{k_l} \to 0$ for $l \to \infty$. It remains to prove that the complementary slackness condition is satisfied for the pair $(\tilde{x}, \widetilde{U})$. Recalling that $\|U^{k_l}\|$ and $\left\|\mathcal{A}(x^{k_l})\right\|$ are bounded and using Proposition 7.11 we obtain

$$\begin{aligned}\lim_{l\to\infty}\left\langle \mathcal{A}(x^{k_l}), U^{k_l}\right\rangle &= \lim_{l\to\infty}\left\langle P_0\mathcal{A}(x^{k_l})P_0, U^{k_l}\right\rangle + \lim_{l\to\infty}\left\langle \mathcal{A}(x^{k_l}), P_\perp U^{k_l} P_0\right\rangle + \\ &\quad \lim_{l\to\infty}\left\langle \mathcal{A}(x^{k_l}), P_0 U^{k_l} P_\perp\right\rangle + \lim_{l\to\infty}\left\langle \mathcal{A}(x^{k_l}), P_\perp U^{k_l} P_\perp\right\rangle = 0.\end{aligned}$$

Consequently $(\tilde{x}, \widetilde{U})$ is a KKT-point of problem (SDP) and the proof of Theorem 7.13 is complete. □

Let us conclude this section with the following discussion: Let x^+ be a local optimum of problem (SDP) and U^+ the corresponding optimal multiplier. Assume further that the pair (x^+, U^+) is a cluster point of the sequence $(x^k, U^k)_{k\in\mathbb{N}}$. Then we can stop the penalty parameter update for some $p_0 > 0$ and Algorithm 7.2.1 switches automatically to Algorithm 6.1.1. Of course we do not know this value p_0 a priori. Instead we use the KKT-error defined by

$$\Upsilon_{\mathrm{KKT}}(x, U) = \min\left\{\lambda_{\max}\left(\mathcal{A}(x)\right), \left|\langle \mathcal{A}(x), U\rangle\right|, \left\|\Delta_x F(x, U, p)\right\|\right\}, \tag{7.23}$$

in order to decide if a penalty update should be performed or not. This leads to the following algorithm

Algorithm 7.2.2 *Let $x^1 \in \mathbb{R}^n, U^1 \in \mathbb{S}^m_{++}, p^1 > 0$ and $\epsilon_0 > 0$ be given. Then for $k = 1, 2, \ldots$ repeat till a stopping criterium is reached:*

```
(i)     Find $x^{k+1}$ such that $\left\|F'_x(x^{k+1}, U^k, p^k)\right\| \le \epsilon_k$.
(ii)    Set $U^{k+1} = D\Phi_p\left(\mathcal{A}(x)\right)\left[U^k\right]$.
(iii)   If  $\Upsilon_{\mathrm{KKT}}(x^{k+1}, U^{k+1}) < \Upsilon_{\mathrm{KKT}}(x^k, U^k)$ go to $(iv)$. Otherwise
        set $p^{k+1} < p^k$.
(iv)    Set $\epsilon^{k+1} \le \epsilon^k$.
```

Note that Algorithm 7.2.2 will be further refined in Chapter 9.

Chapter 8

Complexity Issues and Consequences

In the first part of this chapter we derive a formula for the computational complexity of Algorithm 7.2.2. We further observe that the complexity formula is invariant with respect to sparsity in the problem data. In the second part we demonstrate how we can overcome this problem by the choice of a special penalty function.

8.1 Complexity Analysis

In order to guess the computational complexity of Algorithm 7.2.2 we have to analyze the computational complexity of its components – step (i) to step (iii). Obviously the penalty update, as an operation involving just one real number can be neglected in this context. Moreover we will see that the calculation of the multiplier estimate is implicitly performed during the solution of the unconstrained minimization problem in step (i). Consequently we can limit our complexity analysis to the problem

$$\min_{x \in \mathbb{R}^n} F(x, U, p). \tag{8.1}$$

As we will see in Chapter 9.1, our method of choice for the approximate solution of problem (8.1) is a second order method. The computational complexity of this method is dominated by the operations

(O1) assembling of $F'_x(x, U, p)$,

(O2) assembling of $F''_{xx}(x, U, p)$,

(O3) solution of linear systems of the form

$$(F''_x(x, U, p) + D)\Delta x = -F'_x(x, U, p), \tag{8.2}$$

where D is a diagonal matrix, whose computation is uncritical with respect to computational complexity.

Moreover we assume that

(S1) the number of evaluations of steps (O1) to (O3) is independent of the problem dimensions n and m.

Thus the computational complexity of Algorithm 6.1.1 is fully determined by the computational complexities of (O1) to (O3).

Now using the abbreviation $Q(x) = [\Delta\varphi(\lambda_k(x), \lambda_l(x))]_{k,l=1}^m$ we obtain from Theorem 2.3

$$\begin{aligned} F'_x(x,U,p) &= f'(x) + \left[\left\langle U, S(x)\left(Q(x) \bullet \left[S(x)^\top \mathcal{A}'_i(x) S(x)\right]\right) S(x)^\top \right\rangle\right]_{i=1}^n \\ &= f'(x) + \left[\left\langle S(x)\left(\left[S(x)^\top U S(x)\right] \bullet Q(x)\right) S(x)^\top, \mathcal{A}'_i(x) \right\rangle\right]_{i=1}^n \\ &= f'(x) + \left[\langle D\Phi_p(\mathcal{A}(x))[U], \mathcal{A}'_i(x)\rangle\right]_{i=1}^n , \end{aligned}$$

where the second equation follows from the properties of the trace operator and the third equation can be seen from Corollary 2.4. Note that the second formula has at least two significant advantages over the first one:

- We have to calculate just one directional derivative instead of n,
- the multiplier update is calculated implicitly.

Now the complexity formula for the computation of the gradient can be constructed from the steps listed below together with their complexity

- Compute an eigenvalue decomposition of $\mathcal{A}(x) \longrightarrow O(m^3)$.
- Compute the matrix $D\Phi_p(\mathcal{A}(x))[U] \longrightarrow O(m^3)$.
- For all i compute the inner products $\langle D\Phi_p(\mathcal{A}(x))[U], \mathcal{A}'_i(x)\rangle \longrightarrow O(m^2 n)$.

Consequently the gradient assembling takes $O(m^3)+O(m^2n)$ steps. Next we calculate the complexity of the Hessian assembling. The Hessian of the Augmented Lagrangian can be written as

$$\begin{aligned} F''_{xx}(x,U,p) &= f''(x) + \left[\left\langle D\Phi_p(\mathcal{A}(x))[U], \mathcal{A}''_{i,j}(x)\right\rangle\right]_{i,j=1}^n + \\ &\quad 2\left[\left\langle D^2\Phi_p(\mathcal{A}(x))[U, \mathcal{A}'_j(x)], \mathcal{A}'_i(x)\right\rangle\right]_{i,j=1}^n . \end{aligned} \tag{8.3}$$

Obviously we need $O(m^2n^2)$ time to compute the second term in formula (8.3), if we assume that the matrix $D\Phi_p(\mathcal{A}(x))[U]$ has already been calculated. Using Theorem 6.6.30 in [45] the last term in formula (8.3) can be reformulated as

$$\left[\sum_{k=1}^m s_k(x)^T \mathcal{A}'_i(x) \left[S(x)\Big(Q_k \bullet [S(x)^T U S(x)]\Big) S(x)^T\right] \mathcal{A}'_j(x) s_k(x)\right]_{i,j=1}^n, \tag{8.4}$$

where $Q_k(x)$ denotes the matrix $[\Delta^2\varphi(\lambda_r(x), \lambda_s(x), \lambda_k(x))]_{r,s=1}^m$ and s_k is the k-th row of the matrix $S(x)$. Essentially, the construction of (8.4) is done in three steps, shown below together with their complexity:

- For all k compute matrices $S(x)\Big(Q_k \bullet [S(x)^T U S(x)]\Big)S(x)^T \longrightarrow O(m^4)$.
- For all k, i compute vectors $s_k(x)^T \mathcal{A}_i'(x) \longrightarrow O(nm^3)$.
- Multiply and sum up expressions above $\longrightarrow O(m^3 n + m^2 n^2)$.

Consequently the Hessian assembling takes $O(m^4+m^3n+m^2n^2)$ time. Finally, taking into account that the complexity of the solution of one linear system of type (8.2) is of order $O(n^3)$, we are able to state the following Theorem:

Theorem 8.1 *Suppose that assumption $(S1)$ holds. Then the computational complexity of Algorithm 7.2.2 is of the order $O(m^4 + m^2n^2 + n^3)$.*

Proof . The assertion of the Theorem follows immediately from the complexity formulas above and the fact that $nm^3 \leq m^4$ for $n \leq m$ and $nm^3 \leq n^2m^2$ for $n \geq m$. $\square$

Next we want to discuss the ability of Algorithm 6.1.1 to exploit sparsity in the problem data. We assume that there are at most $K < m^2$ non-zero entries in the constraint matrix $\mathcal{A}(x)$ (and consequently in all partial derivatives $\mathcal{A}_i'(x)$). Unfortunately the complexity formula we obtain in this case, is just the formula derived in Theorem 8.1 for *dense* problem data. This is due to the fact that the matrices Q_k and $S(x)$ are generally dense, even if the matrix $\mathcal{A}(x)$ is very sparse. The disenchanting conclusion is that

Algorithm 7.2.2 is generally not able to exploit sparsity in the problem data.

Fortunately the situation can be improved significantly by a special choice of the penalty function Φ_p.

8.2 A Special Penalty Function

The problem that Algorithm 7.2.2 is not able to exploit sparsity in the problem data is mainly caused by the fact that it is based on eigenvalue decompositions. Subsequently we will show how we can avoid this drawback by a special choice of the penalty function φ. In particular, we are interested a function that allows for a "direct" computation of Φ_p and its first and second derivatives. The function of our choice is the hyperbolic penalty function φ_{hyp} introduced in Chapter 4.2.

Theorem 8.2 *Let $\mathcal{A} : \mathbb{R}^n \to \mathbb{S}^m$ be a convex operator. Let further be Φ_p^{hyp} the primary matrix function associated with φ_{hyp}. Then for any $x \in \mathbb{R}^n$ the following formulas hold:*

$$\Phi_p^{\text{rec}}(\mathcal{A}(x)) = p^2\mathcal{Z}(x) - pI \tag{8.5}$$

$$\frac{\partial}{\partial x_i}\Phi_p^{\text{rec}}(\mathcal{A}(x)) = p^2\mathcal{Z}(x)\mathcal{A}_i'(x)\mathcal{Z}(x) \tag{8.6}$$

$$\begin{aligned}\frac{\partial^2}{\partial x_i \partial x_j}\Phi_p^{\text{rec}}(\mathcal{A}(x)) = p^2\mathcal{Z}(x)(\mathcal{A}_i'(x)\mathcal{Z}(x)\mathcal{A}_j'(x) - \mathcal{A}_{i.j}''(x) \\ + \mathcal{A}_i'(x)\mathcal{Z}(x)\mathcal{A}_j'(x)\big)\,\mathcal{Z}(x)\end{aligned} \tag{8.7}$$

where

$$\mathcal{Z}(x) = (\mathcal{A}(x) - pI)^{-1}\,.$$

Proof . Let I_m denote the identity matrix of order m. Since $\mathcal{Z}(x)$ is differentiable and nonsingular at x we have

$$\begin{aligned} 0 &= \frac{\partial}{\partial x_i} I_m = \frac{\partial}{\partial x_i}\left[\mathcal{Z}(x)\mathcal{Z}^{-1}(x)\right] \\ &= \left[\frac{\partial}{\partial x_i}\mathcal{Z}(x)\right]\mathcal{Z}^{-1}(x) + \mathcal{Z}(x)\left[\frac{\partial}{\partial x_i}\mathcal{Z}^{-1}(x)\right], \end{aligned} \tag{8.8}$$

so the formula

$$\frac{\partial}{\partial x_i}\mathcal{Z}(x) = -\mathcal{Z}(x)\left[\frac{\partial}{\partial x_i}\mathcal{Z}^{-1}(x)\right]\mathcal{Z}(x) = -\mathcal{Z}(x)\left[\mathcal{A}_i'(x)\right]\mathcal{Z}(x) \tag{8.9}$$

follows directly after multiplication of (8.8) by $\mathcal{Z}(x)$ and (8.6) holds. For the proof of (8.7) we differentiate the right hand side of (8.9)

$$\begin{aligned} \frac{\partial^2}{\partial x_i \partial x_j}\mathcal{Z} &= -\frac{\partial}{\partial x_i}\left(\mathcal{Z}(x)\left[\mathcal{A}_j'(x)\right]\mathcal{Z}(x)\right) \\ &= -\left[\frac{\partial}{\partial x_i}\mathcal{Z}(x)\right]\mathcal{A}_j'(x)\mathcal{Z}(x) - \mathcal{Z}(x)\left[\frac{\partial}{\partial x_i}\left(\mathcal{A}_j'(x)\mathcal{Z}(x)\right)\right] \\ &= \mathcal{Z}(x)\mathcal{A}_i'(x)\mathcal{Z}(x)\mathcal{A}(x)_j'\mathcal{Z}(x) - \mathcal{Z}(x)\mathcal{A}_{i,j}''(x)\mathcal{Z}(x) \\ &\quad -\mathcal{Z}(x)\mathcal{A}_j'(x)\left[\frac{\partial}{\partial x_i}\mathcal{Z}(x)\right] \\ &= \mathcal{Z}(x)\mathcal{A}_i'(x)\mathcal{Z}(x)\mathcal{A}_j'(x)\mathcal{Z}(x) - \mathcal{Z}(x)\mathcal{A}_{i,j}''(x)\mathcal{Z}(x) \\ &\quad +\mathcal{Z}(x)\mathcal{A}_j'(x)\mathcal{Z}(x)\mathcal{A}_i'(x)\mathcal{Z}(x) \end{aligned}$$

and (8.7) follows. □

Using Theorem 8.2 we can compute the value of Φ_p^{hyp} and its derivatives directly, without the need of eigenvalue decomposition of $\mathcal{A}(x)$. The "direct" formulas (8.6)–(8.7) are particularly simple for an affine operator

$$\mathcal{A}(x) = A_0 + \sum_{i=1}^{n} x_i \mathcal{A}_i \quad \text{with } \mathcal{A}_i \in \mathbb{S}^m,\ i = 0,1,\ldots,n\,,$$

when $\dfrac{\partial \mathcal{A}(x)}{\partial x_i} = \mathcal{A}_i$ and $\dfrac{\partial^2 \mathcal{A}(x)}{\partial x_i \partial x_j} = 0$. If we replace the general penalty function by the hyperbolic function Φ_p^{hyp} then, according to Theorem 8.2, the Hessian of the augmented Lagrangian can be written as

$$\begin{aligned} F_{xx}''(x,U,p) &= f''(x) + \left[\left\langle \mathcal{Z}(x)U\mathcal{Z}(x), \mathcal{A}_{i,j}''(x)\right\rangle\right]_{i,j=1}^{n} + \\ &\quad 2\left[\left\langle \mathcal{Z}(x)U\mathcal{Z}(x)\mathcal{A}_j'(x)\mathcal{Z}(x), \mathcal{A}_i'(x)\right\rangle\right]_{i,j=1}^{n}\,. \end{aligned} \tag{8.10}$$

The assembling of (8.10) can be divided into the following steps:

- Calculation of $\mathcal{Z}(x) \longrightarrow O(m^3)$.
- Calculation of $\mathcal{Z}(x)U\mathcal{Z}(x) \longrightarrow O(m^3)$.
- Calculation of $\mathcal{Z}(x)U\mathcal{Z}(x)\mathcal{A}_i'(x)\mathcal{Z}(x)$ for all $i \longrightarrow O(m^3 n)$.
- Assembling the rest $O(m^2n^2)$.

Now it is straightforward to see that an estimate of the complexity of assembling of (8.10) is given by $O(m^3n + m^2n^2)$. Taking into account that the computational complexity of the calculation of F_x' is dominated by the computational complexity of the Hessian we can formulate the following Theorem.

Theorem 8.3 *Suppose that assumption $(S1)$ holds and let $\Phi = \Phi^{\rm hyp}$. Then the computational complexity of Algorithm 7.2.2 is of the order $O(m^3n + m^2n^2 + n^3)$.*

Of course, in the case of dense problem data, the complexity formula in Theorem 8.3 is not much better than the complexity formula in Theorem 8.1. However we will show in the following section that in contrast to the general situation the complexity formula reduces to $O(m^2 + n^3)$ for certain types of sparsity structures.

8.3 Various Ways of Exploiting Sparsity

Many optimization problems have very sparse data structure and therefore have to be treated by sparse linear algebra routines. We distinguish three basic types of sparsity.

The block diagonal case The first case under consideration is the block diagonal case. In particular we want to describe the case, where

(S2) the matrix $\mathcal{A}(x)$ consists of many (small) blocks.

In this situation the problem (SDP) can be rewritten as

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \quad \mathcal{A}_i(x) \preccurlyeq 0, \quad i = 1,\ldots,d\,, \end{aligned} \tag{SDP-bl}$$

where $\mathcal{A}_i(x) \in \mathbb{S}^{m_i}$ for all $i = 1,\ldots,d$. If we define $\bar{m} = \max\{m_i \mid i = 1,\ldots,d\}$ we can estimate the computational complexity of Algorithm 7.2.2 applied to problem (SDP-bl) by $O(d\bar{m}n^2 + \bar{m}^2n^2 + n^3)$. An interesting subcase of problem (SDP-bl), if

(S3) each of the matrix constraints $\mathcal{A}_i(x)$ involves just a few components of x.

If we denote the maximal number of components involved in any matrix constraint by $\bar{n}$ our complexity formula becomes $O(d\bar{m}\bar{n}^2 + d\bar{m}^2\bar{n}^2 + n^3)$. If we further assume that the numbers $\bar{n}$ and $\bar{m}$ are small compared to n and d and moreover independent of the actual problem size, then the complexity estimate can be further simplified to $O(d + n^3)$. Notice that

- The latter formula is independent from the choice of the penalty function, but we should mention that also in this case it is advantageous to make use of the penalty function Φ^{hyp}.
- The term $O(n^3)$ coming from the solution of the linear system (8.2) is clearly dominating.

Now we can go even further and assume that

(S4) the Hessian of the objective f is sparsely populated.

Then it follows from assumption (A10) that also the Hessian of the augmented Lagrangian is sparsely populated. Consequently, if we make use of specialized linear system solvers designed for sparsely populated matrices (compare Section 9.1.4), the computational complexity formula can be again improved. We summarize our considerations in the following Corollary.

Corollary 8.4 *Suppose that assumptions (S1) to (S4) hold for the problem (SDP-bl). Then the computational complexity of Algorithm 7.2.2 is of order* $O(d + n^2)$.

A typical example satisfying the assumptions of Corollary 8.4 will be presented in Section 11.5.2.

The case when $\mathcal{A}(x)$ is dense and $\mathcal{A}'_i(x)$ are sparse Let us first mention that for any index pair $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ the non-zero structure of the matrix $\mathcal{A}''_{i,j}(x)$ is given by (a subset of the) intersection of the non-zero index sets of the matrices $\mathcal{A}'_i(x)$ and $\mathcal{A}'_j(x)$. Now we want to find out, how the complexity estimate given in Theorem 8.3 improves, if we assume that

(S5) there are at most $O(1)$ non-zero entries in $\mathcal{A}'_i(x)$ for all $i = 1, \ldots, n$.

Then the calculation of the term

$$\left[\left\langle \mathcal{Z}(x)U\mathcal{Z}(x), \mathcal{A}''_{i,j}(x)\right\rangle\right]_{i,j=1}^{n}$$

can be performed in $O(n^2)$ time. In the paper by Fujisawa, Kojima and Nakata on exploiting sparsity in semidefinite programming [35] several ways are presented how to calculate a matrix of the form

$$D_1 S_1 D_2 S_2 \tag{8.11}$$

efficiently, if D_1 and D_2 are dense and S_1 and S_2 are sparse matrices. In the case when assumption (S5) holds, it follows that the calculation of the matrix

$$\left[\left\langle \mathcal{Z}(x)U\mathcal{Z}(x)\mathcal{A}'_j(x)\mathcal{Z}(x), \mathcal{A}'_i(x)\right\rangle\right]_{i,j=1}^{n}$$

can be performed in $O(n^2)$ time. Thus, recalling that for the calculation of $\mathcal{Z}(x)$ we have to compute the inverse of an $(m \times m)$-matrix, we get the complexity estimate:

Corollary 8.5 *Suppose that assumptions (S1) and (S5) hold and let* $\Phi = \Phi^{\text{hyp}}$. *Then the computational complexity of Algorithm 7.2.2 is of order* $O(m^3 + n^3)$.

Note that in our implementation we follow the ideas presented in [35]. Many linear SDP problems coming from real world applications have exactly the sparsity structure discussed in this paragraph. Several of them are collected in the test case library SDPLIB (compare Section 11.5).

The case when $\mathcal{A}(x)$ is sparse The third situation concerns the case when $\mathcal{A}(x)$ is a sparse matrix. Also here we can conclude that all partial derivatives of $\mathcal{A}(x)$ of first and second order are sparse matrices. Therefore it suffices to assume that

(S6) the matrix A(x) has at most $O(1)$ non-zero entries.

When using the hyperbolic penalty function $\Phi^{\rm hyp}$, we have to compute expressions of type

$$(\mathcal{A}(x) - pI)^{-1}U(\mathcal{A}(x) - pI)^{-1} \quad \text{and} \quad (\mathcal{A}(x) - pI)^{-1}.$$

Note that each of the matrices above can be calculated by maximally two operations of the type $(A - I)^{-1}M$, where M is a symmetric matrix. Now assume that not only $\mathcal{A}(x)$ but also its Cholesky factor is sparse. Then, obviously, the Cholesky factor of $(\mathcal{A}(x) - pI)$, denoted by L, will also be sparse. This leads to the following assumption:

(S7) Each column of L has at most $O(1)$ non-zero entries.

Now the i-th column of $C := (\mathcal{A}(x) - pI)^{-1}M$ can then be computed as

$$C^i = (L^{-1})^T L^{-1} M^i, \; i = 1, \ldots, n,$$

and the complexity of computing C by Cholesky factorization is $O(n^2)$, compared to $O(n^3)$ when computing the inverse of $(A(x) - pI)$ and its multiplication by U. The following corollary summarizes our observations:

Corollary 8.6 *Suppose that assumptions (S1),(S6) and (S7) hold and let $\Phi = \Phi^{\rm hyp}$. Then the computational complexity of Algorithm 7.2.2 is of order $O(m^2 + n^3)$.*

Chapter 9

Algorithmic Details as Implemented in PENNON

Algorithm 7.2.2 has been implemented in a computer code named PENNON . The code is written in the C-programming language. PENNON is equipped with several interfaces, among them are

- SDPA interface (see Section 11.5 for more details),
- MATLAB interface,
- `C/C++`-interface,
- `Fortran` interface.

Special versions of the code, namely PENBMI and PENSDP are integrated in YALMIP 3.0 [59], a comfortable toolbox based on MATLAB , which can be used to formulate semidefinite programs, subjected to linear, bilinear and in the newest version also general polynomial matrix constraints (compare sections 11.2 and 11.3). In the course of this chapter we present details of our implementation, as for example

- which tool we use for the unconstrained minimization,
- how we perform the multiplier and the penalty update,
- how the algorithm is initialized and when it is stopped.

9.1 The Unconstrained Minimization Problem

Throughout this section we consider the augmented Lagrangian function F as a function of x only. Using this, the unconstrained minimization problem in Step (i) of Algorithm 7.2.2 becomes

$$\min_{x\in\mathbb{R}^n} F(x). \tag{UNC}$$

We have implemented two algorithms for the (approximate) solution of the (possibly non-convex) problem (UNC).

9.1.1 Globalized Newton's Method

The first algorithm is a globalized version of the Newton's Method. The globalized Newton's method is defined as follows:

Algorithm 9.1 *Given an initial iterate x, repeat for all $k = 1, 2, 3, \ldots$ until a stopping criterion is reached*

1. *Compute the gradient g and Hessian H of F at x.*
2. *Try to factorize H by Cholesky decomposition. If H is factorizable, set $\widehat{H} = H$ and go to Step 4.*
3. *Compute $\beta \in [-\lambda_{\min}, -2\lambda_{\min}]$, where $\lambda_{\min}$ is the minimal eigenvalue of H and set*
$$\widehat{H} = H + \beta I.$$
4. *Compute the search direction*
$$d = -\widehat{H}^{-1} g.$$
5. *Perform line-search in direction d. Denote the step-length by s.*
6. *Set*
$$x_{\text{new}} = x + sd.$$

The step-length s in direction d is calculated by a gradient free line-search that tries to satisfy an Armijo condition. Obviously, for a convex F, Algorithm 9.1 is just the damped Newton's method, which is known to converge under standard assumptions (see, for example, [60]).

If, in the non-convex case, the Cholesky factorization in Step 2 fails, we calculate the value of β in Step 3 in the following way:

Algorithm 9.2 *For a given $\beta_0 > 0$*

1. *Set $\beta = \beta_0$.*
2. *Try to factorize $H + \beta I$ by the Cholesky method.*
3. *If the factorization fails due to a negative pivot element, go to step 4, otherwise go to step 5.*
4. *If $\beta \geq \beta_0$, set $\beta = 2\beta$ and continue with 2. Otherwise go to step 6.*
5. *If $\beta \leq \beta_0$, set $\beta = \frac{\beta}{2}$ and continue with step 2. Otherwise STOP.*
6. *Set $\beta = 2\beta$ and STOP.*

Obviously, when Algorithm 9.2 terminates we have $\beta \in [-\lambda_{\min}, -2\lambda_{\min}]$. It is well known from the nonlinear programming literature (see, for example, again [60]) that under quite mild assumptions any cluster point of the sequence generated by Algorithm 9.1 is a first order critical point of problem (UNC).

Remark . There is one exception, where we use a different strategy for the calculation of β. The exception is motivated by the observation that the quality of the search direction gets poor, if we choose β too close to $-\lambda_{\min}$. Therefore, if we encounter bad quality of the search direction, we use a bisection technique to calculate an approximation of $\lambda_{\min}$, denoted by $\lambda^{\mathrm{a}}_{\min}$, and replace β by $-1.5\lambda^{\mathrm{a}}_{\min}$.

9.1.2 The Trust Region Method

The second algorithm we apply to the unconstrained minimization problem in step (i) of Algorithm 7.2.2 is the standard Trust-Region method. In particular we have implemented a version of Algorithm 6.1.1 from [27], where we use the Euclidian norm to define the trust region and the standard second order model

$$m(x+s) = F(x) + \langle g, s\rangle + \frac{1}{2}\langle a, Hs\rangle$$

with $g = F'(x)$ and $H = F''(x)$ to approximate F within the trust region. The step calculation is performed exactly as described in Algorithm 7.3.4 of [27] and the free parameters in both algorithms are chosen as recommended in [27, p. 781ff]. A convergence result for the trust region algorithm is provided for example by Theorem 6.4.1 in [27].

9.1.3 Globalized Newton's Method versus Trust Region Method

Algorithm 9.1 turned out to be quite robust as long as the Hessian H of F is not too ill conditioned. In the ill conditioned case, we are still able to calculate approximations of KKT-points in many cases, but the precision we achieve is comparably low. The trust region variant on the other hand turned out to be often slower, but more robust in a neighborhood of first order points. Therefore we use as an alternative a combination of both approaches: At the beginning (typically during the first 10 to 15 outer iterations) of Algorithm 7.2.2 we use the first approach to solve problem (UNC). As soon as a certain stopping criterion is met or when running into numerical difficulties, the trust region variant is used instead. In many test cases very few (typically 3 to 5) iterations are sufficient to improve the precision of the solution.

9.1.4 How to Solve the Linear Systems?

In both algorithms proposed in the preceding sections one has to solve repeatedly linear systems of the form

$$(H + D)s = -g, \tag{9.1}$$

where D is a diagonal matrix chosen such that the matrix $H + D$ is positive definite. There are two categories of methods, which can be used to solve problems of type

(9.1): iterative and exact methods. Let us first concentrate on exact methods. Since the system matrix in (9.1) is forced to be positive definite, our method of choice is the Cholesky method. Depending on the sparsity structure of H, we use two different realizations:

- If the fill-in of the Hessian is below 20%, we use a sparse Cholesky solver which is based on an ideas of Ng and Peyton [66]. The solver makes use of the fact that the sparsity structure is the same in each Newton step in all iterations. The following steps are performed just once at the beginning of the optimization task:
  - At the very beginning the sparsity pattern of H is calculated and stored.
  - Then the rows and columns of H are reordered with the goal to reduce the fill-in in the Cholesky factor. This is done by the minimum degree algorithm described in [37] and [58].
  - A symbolic factorization of H is calculated.

  Then, each time the system (9.1) has to be solved, the numeric factorization is calculated based on the precalculated symbolic factorization. Note that we added stabilization techniques described in [86] to make the solver more robust for almost singular system matrices.

- Otherwise, if the Hessian is dense, we use the ATLAS implementation of the LAPACK Cholesky solver `DPOTRF`, which is (to our best knowledge) the fastest and most robust solver for dense symmetric positive definite systems, which is available free of charge.

Just recently we have started to use iterative methods for the solution of linear systems. A detailed description of this approach along with first numerical experiments are reported in [53]. For linear semidefinite programming problems, we use the following hybrid approach, whenever the number of variables (n) is large compared to the size of the matrix constraint (m): We try to solve the linear systems using the iterative approach as long as the iterative solver needs a moderate number of iterations. In our current implementation the maximal number of iterations allowed is 100. Each time the maximal number of steps is reached, we repeat the solution of the system by an exact method (as described above). As soon as the iterative solver fails three times in sequel, we completely switch to the exact method. Note that this strategy can certainly be improved significantly as we just recently started to use it. On the other hand already now we were able to improve the run time of several test cases significantly (compare Section 11.5). The main reason is that, when using the iterative approach, the Hessian of the Augmented Lagrangian has not to be calculated explicitly (again we refer to [53] for details).

9.2 Update Strategies

9.2.1 The Multiplier Update

First we would like to motivate the multiplier update formula in Algorithm 7.2.2.

Proposition 9.1 *Let x^{k+1} be the minimizer of the augmented Lagrangian F with respect to x in the k-th iteration. If we choose U^{k+1} as in Algorithm 7.2.2 we have*

$$\nabla_x L(x^{k+1}, U^{k+1}) = 0,$$

where L denotes the classical Lagrangian of problem (SDP).

Proof . The gradient of F with respect to x reads as

$$F'_x(x, U, p) = f'(x) + \begin{pmatrix} \langle U, D\Phi_p\left(\mathcal{A}(x)\right)[\mathcal{A}'_1(x)]\rangle \\ \vdots \\ \langle U, D\Phi_p\left(\mathcal{A}'_m(x)\right)\rangle \end{pmatrix}. \tag{9.2}$$

Now, since $U^{k+1} := D\Phi_p\left(\mathcal{A}(x^k)\right)[U^k]$, we immediately see that

$$F'_x(x^{k+1}, U^k, p^k) = L'_x(x^{k+1}, U^{k+1})$$

and we obtain $L'_x(x^{k+1}, U^{k+1}) = 0$. □

The following Proposition follows directly from formula 8.6 in Theorem 8.2.

Proposition 9.2 *For our special choice of the penalty function Φ_p^{hyp}, the multiplier update can be written as*

$$U^{k+1} = (p^k)^2 \mathcal{Z}(x^{k+1}) U^k \mathcal{Z}(x^{k+1}) \,,$$

where $\mathcal{Z}$ was defined in 8.2.

Next we want to discuss a modification of the multiplier update. The reason for the modification is twofold: First, numerical studies indicated that big changes in the multipliers often lead to a large number of Newton steps in the subsequent iteration. Second, it may happen that already after a few steps, the multipliers become ill-conditioned and the algorithm suffers from numerical troubles. To overcome these difficulties, we do the following:

Algorithm 9.3 *Given U^k in the k-th iteration*

1. *Calculate U^{k+1} using the update formula in Algorithm 7.2.2.*
2. *Choose a positive $\lambda^k \le 1$.*
3. *Update the current multiplier by*

$$\bar{U}^{k+1} = U^k + \lambda^k (U^{k+1} - U^k).$$

There are two different strategies how to choose λ^k. In our first strategy we use a fixed λ^k during all iterations. Typical values range between 0.1 and 0.7. Alternatively, we choose λ^k such that the norm $\left\|\bar{U}^{k+1} - U^k\right\|$ does not violate a certain upper bound.

Proposition 9.3 *Let $U^k \in \mathcal{V}(U^*, p_0, \delta, \epsilon, \Theta)$ and suppose that the parameter λ^k in Algorithm 9.3 is bounded away from 0, then there exists p small enough such that the estimate*

$$\|\bar{U}^{k+1} - U^*\| \leq \gamma(p) \left\|U^k - U^*\right\| \tag{9.3}$$

holds with $\gamma(p) < 1$

Proof . From Theorem 6.14 follows the existence of a constant C independent of p such that the estimate

$$\|U^{k+1} - U^*\| \leq Cp \left\|U^k - U^*\right\|$$

holds. Since λ^k is bounded away from 0, there exists $\underline{\lambda} > 0$ such that $\lambda^k > \underline{\lambda}$ for all $k = 1, 2, \ldots$ and we conclude

$$\begin{aligned}
\|\bar{U}^{k+1} - U^*\| &= \|\lambda U^{k+1} + (1-\lambda)U^k - U^*\| \\
&\leq \lambda\|U^{k+1} - U^*\| + (1-\lambda)\|\bar{U}^k - U^*\| \\
&\leq \lambda Cp \left\|U^k - U^*\right\| + (1-\underline{\lambda})\|\bar{U}^k - U^*\| \\
&= (\lambda Cp + (1-\underline{\lambda}))\|\bar{U}^k - U^*\|.
\end{aligned}$$

Now for p small enough the factor $(\lambda Cp + (1-\underline{\lambda}))$ can be driven arbitrarily close to $(1-\underline{\lambda}) < 1$. □

Proposition 9.3 shows that at least the convergence result for the local Algorithm 6.1.1 remains true, if we replace the original multiplier update formula by Algorithm 9.3. In the global situation (Algorithm 7.2.1) a similar result is difficult to achieve, since in general we can not expect that two subsequent iterates x^k and x^{k+1} are close to each other for large enough k, unless we assume that the sequence of iterates $\{x^k\}_{k\in\mathbb{N}}$ generated by the algorithm converges. On the other hand, if we replace the multiplier update formula in Algorithm 7.2.1 by Algorithm 9.3 and assume that the sequence $\{x^k\}_{k\in\mathbb{N}}$ generated by the modified algorithm converges, we can prove (under the assumption that $\{\lambda^k\}_{k\in\mathbb{N}}$ is bounded away from 0) that the sequence $\{x^k, \bar{U}^k\}_{k\in\mathbb{N}}$ converges to a first order critical point of the problem (SDP). For the proof we can use exactly the same argumentation as in Section 7.2 with the only difference that the assertions of Lemma 7.9, Proposition 7.11 and Proposition 7.12 have to be proven for two subsequent multiplier iterates U^k and U^{k+1} this time.

9.2.2 The Penalty Parameter Update

Let $\lambda_{\max}(\mathcal{A}(x^k)) \in \left(0, p^k\right)$ denote the maximal eigenvalue of $\mathcal{A}(x^k)$, $\kappa < 1$ be a constant factor (typically chosen between 0.3 and 0.6) and x_{feas} be a feasible point of problem (SDP). Then our strategy for the penalty parameter update can be described as follows:

Algorithm 9.4 *Given $0 < \kappa < 1$ perform the following steps*

1. *Calculate $\lambda_{\max}(\mathcal{A}(x^k))$.*

2. *If* $\kappa p^k > \lambda_{\max}(\mathcal{A}(x^k))$*, set* $\gamma = \kappa \;\; l = 1$ *and go to 5.*
3. *If* $l < 3$*, set* $\gamma = \left(\lambda_{\max}(\mathcal{A}(x^k)) + p^k\right)/2$*, set* $l = l + 1$ *and go to 5.*
4. *Let* $\gamma = \kappa$*, find* $\lambda \in (0, 1)$ *such that*
$$\lambda_{\max}\left(\mathcal{A}(\lambda x^{k+1} + (1-\lambda)x_{\text{feas}})\right) < \kappa p^k$$
and set $x^{k+1} = \lambda x^{k+1} + (1-\lambda)x_{\text{feas}}$.
5. *Update current penalty parameter by* $p^{k+1} = \gamma p^k$.

The redefinition of x^{k+1} in step 5 guarantees that the values of the augmented Lagrangian in the next iteration remain finite. Of course, if no feasible point x_{feas} is available, step 5 of Algorithm 9.4 is not applicable. In this case, the main Algorithm is restarted using a different choice of initial multipliers (compare Section 9.4 for details).

Note that the penalty parameter update is not necessarily performed in each step of the main algorithm. In fact, we use two different strategies:

- *Permanent strategy*: The penalty parameter is updated (using Algorithm 9.4) in each step of the main algorithm until a certain value $\underline{p}$ is reached. Afterwards we switch to the adaptive strategy in the hope that the penalty parameter can be kept constant.
- *Adaptive strategy*: The penalty parameter update is performed only if
$$\Upsilon(x^{k+1}, U^{k+1}) > \Upsilon(x^k, U^k).$$

The parameter $\underline{p}$ is typically chosen as 10^{-6}.

9.3 Initialization and Stopping Criteria

9.3.1 Initialization

As we have seen in Chapter 6.2, our algorithm can start with an arbitrary primal variable $x \in \mathbb{R}^n$. Therefore we simply choose $x^1 = 0$. For the description of the multiplier initialization strategy we rewrite problem (SDP) in the following form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \quad \mathcal{A}_i(x) \preccurlyeq 0, \qquad i = 1, \ldots, d\,. \end{aligned}$$

Here $\mathcal{A}_i(x) \in \mathbb{S}^{m_j}$ are diagonal blocks of the original constrained matrix $\mathcal{A}(x)$ and we have $d = 1$ if $\mathcal{A}(x)$ consists of only one block. Now the initial values of the multipliers are set to

$$U_j^1 = \mu_j I_{m_j}, \qquad j = 1, \ldots, d,$$

where I_{m_j} are identity matrices of order m_j and

$$\mu_j = m_j \max_{1\le \ell \le n} \frac{1 + \left|\frac{\partial f(x)}{\partial x_\ell}\right|}{1 + \left\|\frac{\partial \mathcal{A}(x)}{\partial x_\ell}\right\|} \,. \tag{9.4}$$

Given the initial iterate x^1, the initial penalty parameter p^1 is chosen large enough to satisfy the inequality

$$p^1 I - \mathcal{A}(x^1) \succ 0.$$

9.3.2 Stopping Criteria

Stopping criterion in the sub-problem In the first phase of Algorithm 6.1.1, the approximate minimization of F is stopped when $\|\frac{\partial}{\partial x}F(x,U,p)\| \le \alpha$, where $\alpha = 0.01$ is a good choice in most cases. In the second phase, after a certain precision is reached (compare Section 9.4 for details), α is reduced in each outer iteration by a constant factor, until a certain $\underline{\alpha}$ (typically 10^{-7}) is reached.

Stopping criterion for main algorithm We have implemented two different stopping criteria for the main algorithm.

- *First alternative:* The main algorithm is stopped if both of the following inequalities hold:

$$\frac{|f(x^k) - F(x^k, U^k, p)|}{1 + |f(x^k)|} < \varepsilon_1 \,, \qquad \frac{|f(x^k) - f(x^{k-1})|}{1 + |f(x^k)|} < \varepsilon_1 \,,$$

  where ε_1 is typically 10^{-7}.

- *Second alternative:* The second stopping criterion is based on the KKT-conditions. Here the algorithm is stopped, if

$$\Upsilon(x^k, U^k) \le \varepsilon_2.$$

Note that, in case the second stopping criterion is chosen, the first stopping criterion can used to define, when

- the update of the stopping criterion in the sub-problem is started,
- to switch from the globalized Newton's algorithm (compare Section 9.1.1) to the Trust Region algorithm (compare Section 9.1.2) as solver for the sub-problem.

Remark . In the case of linear semidefinite programs, we have additionally adopted the DIMACS criteria [61]. To define these criteria, we rewrite our problem (SDP) as

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} b^T x \\ &\text{subject to} \\ &\mathcal{C}(x) \preccurlyeq C_0 \end{aligned} \tag{9.5}$$

where $\mathcal{C}(x) - C_0 = \mathcal{A}(x)$. Recall that U is the corresponding Lagrangian multiplier and let $\mathcal{C}^*(\cdot)$ denote the adjoint operator to $\mathcal{C}(\cdot)$. The DIMACS error measures are defined as

$$\mathrm{err}_1 = \frac{\|\mathcal{C}^*(U) - f\|}{1 + \|f\|}$$

$$\mathrm{err}_2 = \max\left\{0, \frac{-\lambda_{\min}(U)}{1+\|f\|}\right\} \qquad \mathrm{err}_4 = \max\left\{0, \frac{-\lambda_{\min}(\mathcal{C}(x) - C_0)}{1+\|C_0\|}\right\}$$

$$\mathrm{err}_5 = \frac{\langle C_0, U\rangle - f^T x}{1 + |\langle C_0, U\rangle| + |f^T x|} \qquad \mathrm{err}_6 = \frac{\langle \mathcal{C}(x) - C_0, U\rangle}{1 + |\langle C_0, U\rangle| + |f^T x|}\,.$$

Here, err_1 represents the (scaled) norm of the gradient of the Lagrangian, err_2 and err_4 is the dual and primal infeasibility, respectively, and err_5 and err_6 measure the duality gap and the complementarity slackness. Note that, in our code, $\mathrm{err}_2 = 0$ by definition; also err_3 that involves the slack variable (not used in our problem formulation) is automatically zero. If the "DIMACS stopping criterion" is activated we require that

$$\mathrm{err}_k \le \delta_{\mathrm{DIMACS}}, \quad k \in \{1, 4, 5, 6\}\,.$$

9.4 The PENNON Algorithm

We conclude this chapter with a compact description of the PENNON -Algorithm:

Algorithm 9.4.1 *For given $\underline{p}$, κ, α, ς, ε_1, ε_2, ϑ perform the following steps:*

1. *Set $k = 1,\ l_1 = 1,\ l_2 = 1$ and*

$$\begin{aligned} x^1 &= (0, \dots, 0) \\ U^1 &= \mathrm{diag}(\mu_1, \dots, \mu_1,\ \dots\ , \mu_d, \dots, \mu_d), \\ p^1 &= 2\lambda_{\max}\left(\mathcal{A}(x^1)\right), \\ \epsilon^1 &= \alpha. \end{aligned}$$

2. *Repeat until $\|F'_x(x^{k+1}, U, p)\| < \epsilon^k$:*

   2.1 *Compute the gradient g and Hessian H of F at x^{k+1}.*

   2.2 *Try to factorize H by Cholesky decomposition. If H is factorizable, set $\widehat{H} = H$ and go to Step 2.4.*

   2.3 *Compute $\beta \in [-\lambda_{\min}, -2\lambda_{\min}]$ via Algorithm 9.2, where $\lambda_{\min}$ is the minimal eigenvalue of H and set*

$$\widehat{H} = H + \beta I.$$

   2.4 *Compute the search direction*

$$d = -\widehat{H}^{-1} g.$$

```
      2.5 Perform line-search in direction $d$. Denote the step-length by $s$.
      2.6 Set
              $x^{k+1} = x^k + sd.$
3.    3.1 Calculate $U^{k+1} = (p^k)^2 \mathcal{Z}(x^{k+1}) U^k \mathcal{Z}(x^{k+1})$.
      3.2 Choose a positive $\lambda^k \le 1$.
      3.3 Update the current multiplier by
              $U^{k+1} = U^k + \lambda^k (U^{k+1} - U^k).$
4. If $\Upsilon_{\rm KKT}(x^{k+1}, U^{k+1}) < \varepsilon_2 \longrightarrow$ STOP.
5. If $p > \underline{p}$ or $\Upsilon_{\rm KKT}(x^{k+1}, U^{k+1}) > \Upsilon_{\rm KKT}(x^k, U^k)$ do
      5.1 Calculate $\lambda_{\max}(\mathcal{A}(x^k))$.
      5.2 If $\kappa p^k > \lambda_{\max}(\mathcal{A}(x^k))$, set $\gamma = \kappa$, $l_1 = 1$ and go to 5.5.
      5.3 If $l_1 < 3$, set $\gamma = \left(\lambda_{\max}(\mathcal{A}(x^k)) + p^k\right) / 2p^k$, set $l_1 = l_1 + 1$ and go to
          5.5.
      5.4 If $x_{\rm feas}$ is not yet available, set
              $x^{k+1} = x^1,\ U^{k+1} = \vartheta\, U^1,\ p^{k+1} = p^1,\ \epsilon^{k+1} = \alpha,\ l_2 = l_2 + 1$
          and go to step 7. Otherwise set $\gamma = \kappa$, find $\lambda \in (0,1)$ such that
              $\lambda_{\max}\left(\mathcal{A}(\lambda x^{k+1} + (1-\lambda) x_{\rm feas})\right) < \kappa p^k$
          and set $x^{k+1} = \lambda x^{k+1} + (1-\lambda) x_{\rm feas}$.
      5.5 Set
              $p^{k+1} = \gamma p^k.$
   otherwise set
              $p^{k+1} = p^k.$
6. If
              $\max\left\{ \frac{|f(x^k) - F(x^k, U^k, p)|}{1 + |f(x^k)|}, \frac{|f(x^k) - f(x^{k-1})|}{1 + |f(x^k)|} \right\} < \varepsilon_1,$
   set
              $\epsilon^{k+1} = \varsigma \epsilon^k,$
   otherwise set
              $\epsilon^{k+1} = \epsilon^k.$
7. If $l_2 > 3 \longrightarrow$ STOP. Otherwise set $k = k + 1$ and go to step 2.
```

Remark . Optionally step 2 can be calculated by the Trust Region method, either from the very beginning, or after the condition in step 6 is met for the first time.

Remark . If we encounter penalty update problems in step 5.4 and no feasible point of problem (SDP) is available, the algorithm is restarted with larger initial multipliers. The idea is to put “more weight” to the constraints with the hope that the algorithm finds a feasible point in the next trial. If the algorithm is restarted several times without finding a feasible point it has no sense to go on and we give up (compare step 7).

Chapter 10

Applications

10.1 Structural Optimization

Structural optimization deals with engineering design problems with the aim of finding an optimal structure (as specified by a given cost function) that satisfies a number of given constraints. Typically, the designer is faced to the problem of finding optimal design parameters such that the resulting structure is light and/or stiff. A simple example is the problem of finding the stiffest structure with respect to a set of given loads under the constraint that the weight of the structure is restricted. In the following sections two types of structural optimization problems will be shortly introduced, namely the truss topology design and the material optimization problems. In both cases (linear) semidefinite problem formulations will be presented. In the third section two ways of including stability control to these problem formulations will be considered.

10.1.1 Material Optimization

In material optimization (MO) one tries to find a distribution of a given amount of a given elastic material in a given region, so that the resulting structure is optimal (in certain sense) with respect to given loads. The material can even vanish in certain areas, thus one often speaks of topology optimization. The single-load MO problem in the simplest form can be written as follows:

$$\max_{\substack{\rho\in L^\infty(\Omega):\\ \rho\geq 0,\int_\Omega \rho\, dx\leq V\\ 0\leq\rho\leq\overline{\rho}}} \min_{u\in U} \frac{1}{2}\int_\Omega \rho(x)\langle E(x)e(u(x)), e(u(x))\rangle \mathrm{d}x - \int_{\Gamma_2} f(x)\cdot u(x)\mathrm{d}x\,, \quad (10.1)$$

where Ω is a bounded domain with Lipschitz boundary $\Gamma = \overline{\Gamma_1}\cup\overline{\Gamma_2}$, E is the elasticity tensor of the given material, ρ is the design variable, $u(x)$ denotes the displacements in each point of the body, $e(u)$ is the small-strain tensor, $\overline{\rho}$ is an upper bound on ρ and f is an external force acting on Γ_2. Further, V is an upper bound on resources and $U \subseteq \mathcal{H}^1(\Omega)$ is a set defining boundary conditions and possible unilateral contact conditions. The design variable ρ can be interpreted as thickness in 2D problems or as a sort of density in 3D problems; see [9] for a detailed discussion. A categorization of resulting problems for different choices of E and dim is given in [52]. Here only two cases are considered:

- $$E = I$$
- $$E \ldots \text{elasticity matrix of an isotropic material}$$

We will concentrate on the first case, connected with the so-called Free Material Optimization (FMO) [88, 8, 84] at the moment. The second case, which is called variable thickness sheet (VTS) problem [8, 67] in the two-dimensional case, and can be considered as a sub-case of SIMP [8] in the three-dimensional situation will be further referred in Section 10.1.3. Free material optimization is a branch of structural optimization; its goal is to find the ultimately best structure able to carry given loads. The design variables are the material properties that can vary from point to point. The optimal structure/material can be interpreted by fiber-reinforced composites. A description of the single-load case, together with numerical techniques, is given, for example, in [88], where the single-load free material optimization problem is formulated as

$$\min_{\substack{E \in L^\infty(\Omega): \\ E \succeq 0, \int_\Omega \operatorname{tr}(E)\, dx \le V \\ 0 \le \operatorname{tr}(E) \le \overline{\rho}}} \max_{u \in U} \int_{\Gamma_2} f(x) \cdot u(x) \mathrm{d}x - \frac{1}{2} \int_\Omega \langle E(x) e(u(x)), e(u(x)) \rangle \mathrm{d}x \,. \tag{10.2}$$

The tensor E is written as a matrix in this formulation and we only require that it is physically attainable; i.e., E is symmetric and positive semidefinite at each point of Ω. At the first glance, problem (10.2) is much more difficult than the MO problem (10.1) due to the matrix variable E. However, it can be shown that after analytical reformulation E can be eliminated, and we indeed get a special case of problem (10.1) with $E = I$. In order to solve this (infinite-dimensional) problem numerically, discretized versions of material optimization problems have been derived. The discretization is done by the finite element method. More precisely, Ω is partitioned into m elements Ω_i, $i = 1, \ldots, m$, E is approximated by a function which is constant on each element Ω_i and the displacement vector u is approximated by a piece-wise polynomial function. If we stay with the notation of the original problem, the discretized version can be stated as follows:

$$\max_{\substack{\rho \in \mathbb{R}^m: \\ \sum_{i=1}^m \rho_i \le V, 0 \le \rho_i \le \overline{\rho}}} \min_{u \in U \subset \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^m \rho_i u^\top A_i u - f^\top u \,. \tag{10.3}$$

The matrices A_i are positive semidefinite matrices (more details will be given below), m is the number of finite elements and n the number of degrees of freedom of the (discretized) displacement vector. One can see that the matrix E was analytically reduced to a scalar variable ρ having the meaning of trace of E; the full matrix E can be, however, recovered from the optimal solution (ρ, u) of problem (10.3). There exist several equivalent formulations to problem 10.3. For example, one can rewrite problem (10.3) as linear SDP or as the following large-scale convex quadratically constrained NLP problem

$$\min_{\alpha\in\mathbb{R}, u\in\mathbb{R}^n} \left\{\alpha - f^\top u \,|\, \alpha \geq u^\top A_i u \text{ for } i = 1,\ldots,m\right\}. \tag{10.4}$$

The latter formulation is preferred in practice, due to its significantly lower computational complexity.

For illustration, we consider a three-dimensional single-load example. The results presented below were computed by MOPED , a software package for material optimization, which uses a special version of PENNON as optimization engine. The goal in this example was the design of a rear fuselage of a cargo airplane. Due to the large opening for the door, the fuselage was very weak with respect to bending. At the same time, there were huge bending forces at the very rear part, coming from the tail. The goal of the designer was to carry these forces to the closed part of the fuselage. We solved this problem using several discretizations ranging from 5000 to 50000 finite elements. One of these discretizations is depicted in Figure 10.1, at the left-hand side. The right-hand side of the same figure shows the density distribution $\rho(x), x \in \Omega$ of this problem, as computed by MOPED .

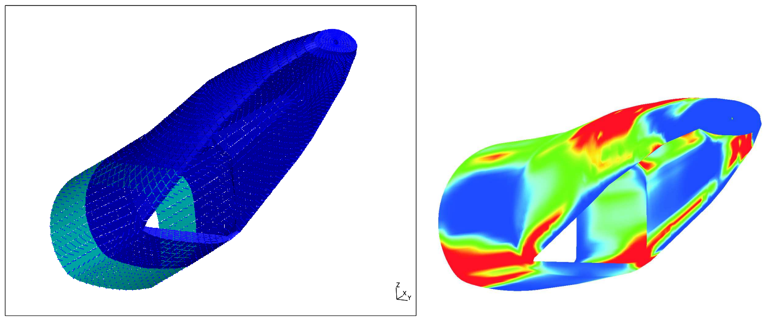


Figure 10.1: Cargo airplane; discretization and density plot

Next we consider the so called worst-case multiple-load problem. Here the optimal structure should be able to withstand a whole collection of L independent loads, acting at different times. Generalization of formula (10.2) leads to the following formulation of the multiple-load case:

$$\min_{\substack{E\in L^\infty(\Omega):\\ E\succeq 0\\ \int_\Omega \text{tr } dx \leq V\\ 0\leq \text{tr}\leq\overline{\rho}}} \max_{l=1,\ldots,L} \max_{u\in U^l} \int_{\Gamma_2} f^l(x)\cdot u(x)\mathrm{d}x - \frac{1}{2}\int_\Omega \langle E(x)e(u(x)), e(u(x))\rangle \mathrm{d}x \tag{10.5}$$

The situation in the multiple-load case is much more complicated than in the single-load case and was analyzed in detail in [3]. Since it is not the goal of this thesis to repeat this theory here, we restrict ourselves to the presentation of the discretized version of problem (10.5):

$$\min_{\substack{E=\{E_i\}_{i=1}^m:\\ E_i\succeq 0\\ \sum_{i=1}^m \omega_i \mathrm{tr}(E_i)\le V\\ 0\le \mathrm{tr}(E_i)\le\overline{\rho}}} \quad \max_{l=1,\dots,L}\max_{u^l\in U^l}\frac{1}{2}(f^l)^\top u - \sum_{i=1}^m \omega_i \mathrm{tr}\left(E_i\zeta_i(u^l)\zeta_i(u^l)^\top\right)\,, \tag{10.6}$$

where $\zeta_i(u),\ i=1,\dots,m$ are matrix valued functions calculated from the discretized strain tensor and positive weights coming from the Gauss integration formula (see [3] for further details). Again in [3] it was shown that under certain assumptions the following semidefinite program is the Lagrange dual to problem (10.6) and that the problems are equivalent in the sense that there is no duality gap:

$$\max_{u_1,\dots,u_l,\nu,\gamma} = -\alpha\nu + 2\sum_{l=1}^L (f^l)^\top u^l - \overline{\rho}\sum_{i=1}^m \gamma_m$$

subject to

$$\begin{pmatrix} (\nu+\gamma_m)I_d & \zeta_m(u^1) & \zeta_i(u^2) & \dots & \zeta_i(u^L)\\ \zeta_m(u^1) & \lambda_1 I_s & & & \\ \zeta_m(u^2) & & \lambda_2 I_s & & \\ \vdots & & & \ddots & \\ \zeta_m(u^L) & & & & \lambda_L I_s \end{pmatrix} \succeq 0,\ i=1,\dots,m, \tag{10.7}$$

$$\gamma_i \ge 0,\ i=1,\dots,m,$$

$$\nu\ge 0,$$

$$\sum_{i=1}^L \lambda_i = 1$$

Hereby α is a constant, d and s are integers defining the dimension of $\zeta_i(u),\ i = 1,\dots,m$, while ν and γ are additional variables introduced for the penalization of constraints in the primal formulation. Unlike in the single load case, from computational point of view, no formulation superior to problem (10.7) is known. Due to the relatively large number of matrix inequalities problem (10.6) has a strong block structure. In Section 11.5.2 we will demonstrate that the code PENNON is able to exploit this fact. A typical multiple-load example solved by FMO can be seen in Figure 10.2. Here the goal is to design a frame of a racing bicycle by means of fiber composites. Figure 10.2 top-left shows the design region Ω together with the loads—we consider two load-cases here. The top-right figure presents the strength of the optimal material—the variable ρ. Here the dense areas indicate stiff material, while the brighter areas stand for a weaker and weaker material. The final two figures show the optimal directions of the fibers in the composite material; one figure for each load case.

10.1.2 Truss Topology Design

In Truss Topology Optimization (TTO) we consider the problem of finding an optimal (stiffest, lightest) truss (pin-jointed framework) with respect to given loads. The

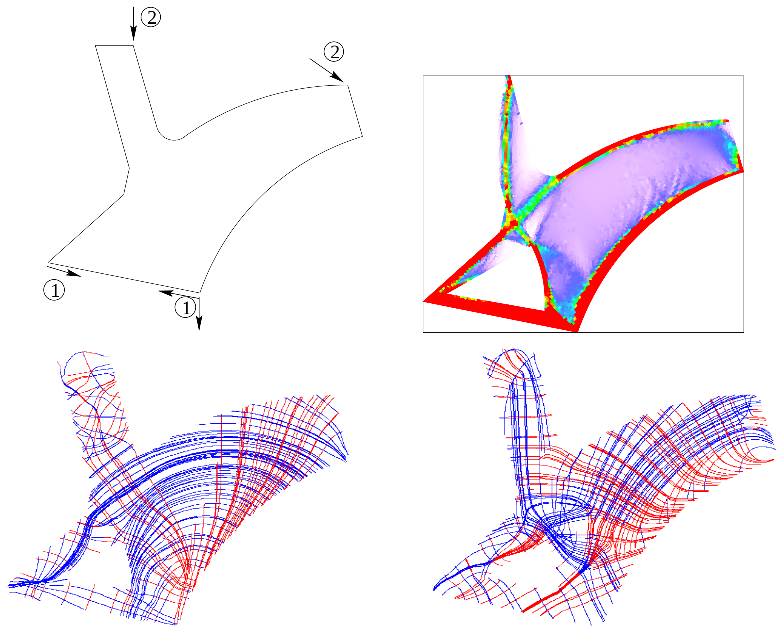


Figure 10.2: FMO design of a bicycle frame.

problem is studied in the so called ground-structure framework (see [31]). Here the truss is modelled by N nodal points in $\mathbb{R}^{\mathrm{dim}}$, $\mathrm{dim} \in \{2;3\}$ and each pair of nodes can be connected by a bar. The design variables are the bar volumes denoted by $t_i \geq 0$, $i = 1, \dots, m$ and $u \in \mathbb{R}^n$ denotes a vector of nodal displacements, where n is dependant on N. Further we can calculate for each bar a symmetric positive semidefinite matrix A_i, $i = 1, \dots, m$, called (in analogy to MO) the local stiffness matrix. Exact formulas are given, for example, in [4]. Under the assumption of linear elastic behavior, the single-load truss topology design problem can be written as

$$\max_{\substack{t \in \mathbb{R}^m: \\ \sum_{i=1}^m t_i \leq V, 0 \leq t_i \leq \bar{t}}} \min_{u \in U \subset \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^m t_i u^\top A_i u - f^\top u \,. \tag{10.8}$$

The multiple-load truss topology design problem is then formulated as

$$\max_{\substack{t \in \mathbb{R}^m: \\ \sum_{i=1}^m t_i \leq V, 0 \leq t_i \leq \bar{t}}} \min_{l=1,\dots,L} \min_{u^l \in U^l \subset \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^m t_i (u^l)^\top A_i u^l - (f^l)^\top (u^l) \,. \tag{10.9}$$

Note the strong analogy to the discretized versions of the material optimization problems introduced in the previous section. As in material optimization there exist various reformulations of the problems above, among them a convex nonlinear version of problem (10.8) and a linear semidefinite formulation of problem (10.9). For details refer, for example, [5] and [8].

10.1.3 Structural Optimization with Vibration and Stability Constraints

Arguably, the most serious limitation of the models presented in the preceding sections is that they do not count with possible instability of the optimal structure. Indeed, elastic instability is often the decisive point when designing a "real-world" structure, like a bridge or an aircraft. Experience showed that such structures may fail in some cases not on account of high stresses but owing to insufficient elastic stability ([78]). In this thesis we consider two ways of including stability control in the problem formulation. The first one is based on the so-called *linear buckling model*. This model is based on an assumption that leads to simplification of the nonlinear structural analysis and that is naturally satisfied for many real-world structures. The second one is based on the control of the minimal eigenfrequency of the structure. Both models lead to control of the minimal eigenvalue of a generalized eigenvalue problem

$$A(\rho)w = \lambda Q(\rho)w\,. \tag{10.10}$$

In the first case (linear buckling), Q is the so-called geometry stiffness matrix that depends in a nonlinear way on the design variable ρ. In the second case (self-vibrations), Q is the mass matrix of the structure and the dependence is linear. Several problem formulations are possible now (see [52]), from which we choose the following:

$$\begin{aligned}
&\min\ W(\rho)\\
&\text{subject to}\\
&\qquad C(\rho) \le \widehat{C}\\
&\qquad \lambda(\rho) \ge \widehat{\lambda}\\
&\qquad \text{equilibrium equation}
\end{aligned}$$

i.e., we minimize the weight of a structure under the constraints that

- the compliance of the structure is restricted,
- the structure is in equilibrium between internal and external forces,
- the minimal positive eigenvalue of problem (10.10) is bounded from below.

The primary goal is to solve problems with stability constraints based on the linear buckling phenomenon. As mentioned above, this approach leads to a non-convex matrix inequality constraint, involving the geometry stiffness matrix. It should be recalled here that it was exactly this problem which motivated the author to develop an algorithm and a computer program for the solution of nonlinear semidefinite programs. Later we will see that due to extremely high computational complexity of this problem, we can only solve model problems of relatively low dimension at the moment. Hence, as a viable alternative, we offer the control of self-vibrations of the optimal structure. This results in a formulation with linear matrix inequality constraints (involving the mass matrix) for which the complexity is much lower.

In the following we will try to explain these ideas in the framework of material optimization. As already mentioned in Section 10.1.1 we will use the assumption

$$E \text{ is an elasticity matrix of an isotropic material.} \tag{10.11}$$

We briefly sketch the discretization for this problem: Let m denote the number of finite elements and n the number of nodes (vertices of the elements). We approximate $\rho(x)$ by a function that is constant on each element, i.e., characterized by a vector $\rho = (\rho_1, \dots, \rho_m)$ of its element values. Further assume that the displacement vector $u(x)$ is approximated by a continuous function that is bi- or tri-linear (linear in each coordinate) on every element. Such a function can be written as $u(x) = \sum_{i=1}^n u_i \vartheta_i(x)$ where u_i is the value of u at i-th node and ϑ_i is the basis function associated with i-th node (for details, see [24]). Recall that, at each node, the displacement has dim components, so $u \in \mathbb{R}^{dim\cdot n}$. Further, element stiffness matrices A_i are defined by

$$A_i = \sum_{k=1}^{nig} B_{i,k}^\top E_{i,k} B_{i,k} \,,$$

where nig is the number of Gauss integration points and the matrices $B_{i,k} \in \mathbb{M}^{d,dim\cdot n}$ are composed from derivatives of the functions $\vartheta_i, i = 1, \dots m$. Now the (global) stiffness matrix A is defined as a linear combination of the element stiffness matrices as follows

$$A(\rho) = \sum_{i=1}^m \rho_i A_i$$

and the discretized version of problem (10.1) with assumption (10.11) becomes

$$\inf_{\substack{\rho \geq 0 \\ \sum_{i=1}^m \rho_i \leq 1}} \sup_{u \in \mathbb{R}^{dim\cdot n}} -\frac{1}{2} \sum_{i=1}^m \rho_i \langle A_i u, u\rangle + \langle f, u\rangle \,.$$

It is well-known that the above problem can be formulated as a minimum weight problem as follows (see again [52] and the references therein):

$$\begin{aligned} &\min_{u,\rho} \sum_{i=1}^m \rho_i \\ &\text{subject to} \\ &\qquad \rho_i \geq 0, \quad i = 1, \dots, m \\ &\qquad f^\top u \leq c \\ &\qquad A(\rho) u = f \,. \end{aligned} \tag{10.12}$$

The stability constraint, in the sense of critical buckling force, requires that all eigenvalues of problem (10.10) are either smaller than zero or bigger than one. The matrix Q in problem (10.10) is replaced by the so-called geometry stiffness matrix defined by

$$G(u,\rho) = P^\top \left(\sum_{i=1}^{m} G_i \right) P, \qquad G_i = \sum_{k=1}^{nig} Q_{i,k}^\top \, S_{i,k} \, Q_{i,k} \,,$$

where P is a permutation matrix, the matrices $Q_{i,k}$ are again composed from derivatives of the functions $\vartheta_i, i = 1,\ldots,m$, and the non-zero entries of the matrices $S_{i,k} \in \mathbb{S}^{dim}$ are items of the element "stress vector"

$$\sigma_{i,k} = (\sigma_1 \; \sigma_2 \; \sigma_3 \; \sigma_4 \; \sigma_5 \; \sigma_6 \;)_{i,k}^\top = \rho_i \, E_{i,k} \, B_{i,k} \, u$$

which can be written as

$$\sigma_{i,k} = \rho_i \, E_{i,k} \, B_{i,k} \, A(\rho)^{-1} f \,. \tag{10.13}$$

In the last formula we clearly see a nonlinear dependence on ρ. It has been proven that condition (10.10) is equivalent to the following matrix inequality (see, e.g., [51]):

$$A(\rho) + Q \succeq 0 \,,$$

so that we are able to combine the optimization problem (10.12) with the constraint

$$A(\rho) + G(u,\rho) \succeq 0 \,, \tag{10.14}$$

to get the minimum weight material optimization problem with stability constraint. Before writing down the full problem formulation, we rewrite, using the Schur complement Theorem, the compliance constraint and the equilibrium equation in one matrix inequality constraint

$$Z(\rho) := \begin{pmatrix} c & f^\top \\ f & A(\rho) \end{pmatrix} \succeq 0. \tag{10.15}$$

Using this, problem (10.12) can be written as:

$$\begin{aligned} &\min_{\rho} \sum_{i=1}^{m} \rho_i \\ &\text{subject to} \\ &\qquad Z(\rho) \succeq 0 \\ &\qquad \rho_i \geq 0, \quad i = 1,\ldots,m \,. \end{aligned} \tag{10.16}$$

We further eliminate the variable u from the stability constraint by assuming that $A(\rho)$ is nonsingular and setting

$$\widetilde{G}(\rho) = G(\rho, A^{-1}(\rho)).$$

The minimum weight problem with stability constraints reads as

$$\begin{aligned} &\min_{\rho} \sum_{i=1}^{m} \rho_i \\ &\text{subject to} \\ &\qquad Z(\rho) \succeq 0 \\ &\qquad A(\rho) + \widetilde{G}(\rho) \succeq 0 \\ &\qquad \rho_i \geq 0, \quad i = 1,\ldots,m \,. \end{aligned} \tag{10.17}$$

A critical issue of this approach is the computational complexity. For details we refer again to [52].

Therefore as a viable alternative we use vibration constraints instead of stability constraints. More precisely this means that we want to find the optimal structure/material such that the lowest eigenfrequency of the structure is bigger than or equal to a prescribed value $\overline{\lambda}$. Using similar arguments as above the vibration constraint can be formulated as a linear matrix inequality of the type

$$A(\rho) - \overline{\lambda} M(\rho) \succeq 0,$$

where M is the mass matrix and $\overline{\lambda}$ a given threshold vibration. Hence the optimization problem with vibration constraints can be than formulated as a linear semidefinite programming problem

$$\begin{aligned} &\min_{\rho} \sum_{i=1}^{m} \rho_i \\ &\text{subject to} \\ &\qquad Z(\rho) \succeq 0 \\ &\qquad A(\rho) - \overline{\lambda} M(\rho) \succeq 0 \\ &\qquad \rho_i \geq 0, \quad i = 1, \ldots, m\,. \end{aligned} \tag{10.18}$$

Due to the linearity and a different sparsity structure, the complexity of this linear SDP is much lower than the complexity of the nonlinear one presented in (10.17). A related problem is the following:

$$\begin{aligned} &\max_{\rho,\lambda} \lambda \\ &\text{subject to} \\ &\qquad A(\rho) - \lambda M(\rho) \succeq 0 \\ &\qquad \rho_i \geq 0, \quad i = 1, \ldots, m\,, \end{aligned} \tag{10.19}$$

where we try to maximize the minimal eigenfrequency of a given structure. Since no external forces are taken into consideration here, there is no compliance constraint. Of course, problem 10.19 makes only sense, if M is of the form

$$M_0 + \sum_{i=1}^{m} \rho_i M_i,$$

where M_0 is a predefined constant mass in the structure. The main advantage of this formulation is in the bilinear structure of the non-convex matrix constraint, which is often easier to solve. Let us conclude this section with the following remark.

Remark . Similar formulations for minimum weight design problems with stability and vibration constraints in the area of truss topology design have been developed already several years before stability constraints were considered in the context of material optimization (see e.g. [4] or [74]). Again there is a strong analogy between these formulations and the discretized problems presented above. Consequently, instead of repeating the problem formulations, we restrict ourselves to the presentation of numerical results in Section 11.1.

10.2 Nonlinear SDPs arising in Control Theory

Many interesting problems in linear and nonlinear systems control cannot be solved easily and efficiently with currently available software. Even though several relevant control problems boil down to solving convex linear matrix inequalities (LMI) - see [20] for a long list - there are still fundamental problems for which no convex LMI formulation has been found.

BMI formulation of the control problems was made popular in the mid 1990s [39]; there were, however, no computational methods for solving non-convex BMIs, in contrast with convex LMIs for which powerful interior-point algorithms were available [65]. Almost one decade later, this unsatisfactory state of the art in BMI solvers is almost unchanged, whereas LMI and semidefinite programming solvers now abound. There were, however, several attempts to solve BMI problems numerically, and the following list is far from being exhaustive:

- Global optimization algorithms based on branch-and-bound schemes [38] or the generalized Benders decomposition [11] were the first historically to be developed. More recently, concave minimization algorithms were described [2] but no software has been developed, and apparently no further research has been carried out in this direction;

- Various heuristics based on iteratively solving LMI subproblems were proposed, the most efficient of which seems to be the cone complementarity linearization algorithm [32]. Inefficiency of these methods in solving very basic BMI problems has been shown e.g. in [41], but because of their simplicity, these methods remain widely used in the control community;

- More recently, several researchers have been trying to apply non-convex programming techniques to BMI problems, with moderate success so far. Interior-point constrained trust region methods are proposed in [57] in the special case of static output feedback and low-order controller design BMIs. The method is a sequential minimization method of a logarithmic barrier function subject to a nonlinear matrix constraint. A similar approach, also based on logarithmic barrier function and using a sophisticated method to the minimization of the unconstrained sub-problems was proposed in [46]. Sequential semidefinite programming, as an extension of quadratic programming, is used in [34] to solve LMI problems with additional nonlinear matrix equality constraints. No publicly available software came out of these attempts to the best of our knowledge.

10.2.1 The Static Output Feedback Problem

A notorious example is the static output feedback control problem which admits a deceptively simple formulation, but for which no systematic polynomial-time algorithm has been designed so far. It is even unclear whether the static output feedback control problem belongs to the category of NP-hard problems.

Two basic static output feedback (SOF) control design problems, namely the SOF–$\mathcal{H}_2$ and SOF–$\mathcal{H}_\infty$ problem can be described like follows: We consider a LTI control system of the form

$$\begin{array}{rcl} \dot{x}(t) & = & Ax(t) + B_1 w(t) + Bu(t), \\ z(t) & = & C_1 x(t) + D_{11} w(t) + D_{12} u(t), \\ y(t) & = & Cx(t) + D_{21} w(t), \end{array} \tag{10.20}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, $y \in \mathbb{R}^{n_y}$, $z \in \mathbb{R}^{n_z}$, $w \in \mathbb{R}^{n_w}$ denote the state, control input, measured output, regulated output, and noise input, respectively. The goal of the SOF problem for a system of type (10.20) is to determine a matrix $F \in \mathbb{R}^{n_u \times n_y}$ of the SOF control law $u(t) = Fy(t)$ such that the closed loop system

$$\begin{array}{rcl} \dot{x}(t) & = & A(F)x(t) + B(F)w(t), \\ z(t) & = & C(F)x(t) + D(F)w(t), \end{array} \tag{10.21}$$

fulfills some specific control design requirements, where $A(F) = A + BFC$, $B(F) = B_1 + BFD_{21}$, $C(F) = C_1 + D_{12}FC$, $D(F) = D_{11} + D_{12}FD_{21}$. Now assuming that $D_{11} = 0$ and $D_{21} = 0$, the SOF–$\mathcal{H}_2$ problem reads as follows:

Find a SOF gain F such that $A(F)$ is Hurwitz and the $\mathcal{H}_2$–norm of (10.21) is minimal.

An equivalent $\mathcal{H}_2$–**BMI** formulation is provided by the following Theorem:

Theorem 10.1 *The SOF–$\mathcal{H}_2$ problem can be rewritten to the following $\mathcal{H}_2$–**BMI** problem formulation:*

$$\begin{gathered} \min \quad Tr(X) \quad \textit{s. t.} \quad Q \succ 0, \\ (A + BFC)Q + Q(A + BFC)^\top + B_1 B_1^\top \preceq 0, \\ \begin{bmatrix} X & (C_1 + D_{12}FC)Q \\ Q(C_1 + D_{12}FC)^\top & Q \end{bmatrix} \succeq 0, \end{gathered} \tag{10.22}$$

where $Q \in \mathbb{R}^{n_x \times n_x}$, $X \in \mathbb{R}^{n_z \times n_z}$.

Note that (10.22) is bilinear in F and Q. For a proof see e. g. [55].

$\mathcal{H}_\infty$ synthesis is an attractive model–based control design tool and it allows incorporation of model uncertainties in the control design. The optimal SOF–$\mathcal{H}_\infty$ problem can be formally stated in the following term:

Find a SOF matrix F such that $A(F)$ is Hurwitz and the $\mathcal{H}_\infty$–norm of (10.21) is minimal.

We consider the following well known $\mathcal{H}_\infty$–**BMI** version:

Theorem 10.2 *The SOF–$\mathcal{H}_\infty$ problem can be equivalently stated as:*

$$\begin{gathered} \min \quad \gamma \quad \textit{s. t.} \quad X \succ 0, \quad \gamma > 0, \\ \begin{bmatrix} A(F)^\top X + XA(F) & XB(F) & C(F)^\top \\ B(F)^\top X & -\gamma\, I_{n_w} & D(F)^\top \\ C(F) & D(F) & -\gamma\, I_{n_z} \end{bmatrix} \prec 0, \end{gathered} \tag{10.23}$$

where $\gamma \in \mathbb{R}$, $X \in \mathbb{R}^{n_x \times n_x}$.

Due to the bilinearity of the free matrix variables F and X, the BMI–formulation of the SOF–$\mathcal{H}_\infty$ is non–convex and nonlinearly constrained. Again, for a proof see e. g. [55].

10.2.2 Simultaneous Stabilization BMIs

Another example is the problem of simultaneously stabilizing a family of single-input single-output linear systems by one fixed controller of given order. This problem arises for instance when trying to preserve stability of a control system under the failure of sensors, actuators, or processors. Simultaneous stabilization of three or more systems was extensively studied in [15]. Later on, the problem was shown to belong to the wide range of robust control problems that are NP-hard, i.e. that are very unlikely to be solved in polynomial time [16].

In [44] a BMI formulation of the simultaneous stabilization problem was obtained in the framework of the polynomial, or algebraic approach to systems control [54]. This formulation is briefly summarized in this section:

Let the real rational functions

$$P_i(s) = \frac{n_i(s)}{d_i(s)}, \quad i = 1, 2, \dots, N$$

be coprime polynomial fraction descriptions for N linear plants. We seek a controller

$$C(s) = \frac{x_n(s)}{x_d(s)}$$

of fixed order simultaneously stabilizing plants $P_i(s)$ when placed in a standard negative feedback configuration. In other words, given polynomials $n_i(s)$, $d_i(s)$ of degree n_p the simultaneous stabilization problem amounts to finding polynomials $x_n(s)$, $x_d(s)$ of given degree n_x such that all the characteristic polynomials

$$p_i(s) = n_i(s)x_n(s) + d_i(s)x_d(s), \quad i = 1, 2, \dots, N \tag{10.24}$$

of degree $n = n_p + n_x$ have their roots in some specified stability region $\mathcal{D}$.

The location of the roots of a polynomial in region $\mathcal{D}$ is captured by the following well-known Hermite stability criterion, which is the symmetric counterpart of the standard Routh-Hurwitz or Schur-Cohn stability criteria.

Lemma 10.3 *The roots of a polynomial $p(s) = p_0 + p_1 s + \cdots + p_n s^n$ belong to region $\mathcal{D}$ if and only if the matrix*

$$H(p) = \sum_{j=0}^{n} \sum_{k=0}^{n} p_j p_k H_{jk}$$

is positive definite, where the Hermitian matrices H_{jk} depend on region $\mathcal{D}$ only.

For more details the interested reader is referred to [43]. Applying Lemma 10.3 to characteristic polynomials (10.24), we derive easily the following BMI formulation of the simultaneous stabilization problem.

Theorem 10.4 *The simultaneous stabilization problem for N plants of order n_p is solved with a controller of order n_x if and only if the $2n_x + 1$ controller coefficients x_k satisfy the N following BMIs of size $n = n_p + n_x$*

$$H(p_i) = \sum_{j=0}^{n}\sum_{k=0}^{n} x_j x_k H_{i,jk} \succ 0, \quad i = 1, 2, \ldots, N$$

where Hermitian matrices $H_{i,jk}$ depend only on the stability region and open-loop plant coefficients.

For explicit expressions for the matrices $H_{i,jk}$ and other details, the interested reader is referred to [44].

In other words our goal is to check, whether a system of BMIs is feasible. In a more general setting this problem can be reformulated by the following procedure: Assume we want to find a feasible point of the following system of BMIs

$$A_0^i + \sum_{k=1}^{n} x_k A_k^i + \sum_{k=1}^{n}\sum_{\ell=1}^{n} x_k x_\ell K_{k\ell}^i \prec 0, \qquad i = 1, \ldots, N \tag{10.25}$$

with symmetric matrices $A_k^i, K_{k\ell}^i \in \mathbb{R}^{d_i \times d_i}$, $k, \ell = 1, \ldots, n$, $i = 1, \ldots, N$, and $x \in \mathbb{R}^n$. Then we can check the feasibility of (10.25) by solving the following optimization problem

$$\min_{x\in\mathbb{R}^n, \lambda\in\mathbb{R}} \lambda \tag{10.26}$$

$$\text{s.t.} \quad A_0^i + \sum_{k=1}^{n} x_k A_k^i + \sum_{k=1}^{n}\sum_{\ell=1}^{n} x_k x_\ell K_{k\ell}^i \preccurlyeq \lambda I_n, \qquad i = 1, \ldots, N\,. \tag{10.27}$$

Problem (10.26) is a global optimization problem: we know that if its global minimum λ is non-negative then the original problem (10.25) is infeasible. On the other hand our algorithm can only find local optima (more precisely, critical points). Thus, when solving (10.26) by our algorithm, the only conclusion we can make is the following:

when $\lambda < 0$, the system is strictly feasible;
when $\lambda = 0$, the system is marginally feasible;
when $\lambda > 0$ the system may be infeasible.

During numerical experiments it turned out that the feasible region of (10.25) is often unbounded. We used two strategies to avoid numerical difficulties in this case: First we introduced large enough artificial bounds x_{bound}. Second, we modify the objective function by adding the square of the 2-norm of the vector x multiplied by a weighting parameter w. After these modifications problem (10.26) reads as follows:

$$\min_{x\in\mathbb{R}^n, \lambda\in\mathbb{R}} \lambda + w\|x\|_2^2 \tag{10.28}$$

$$\text{s.t.} \qquad -x_{\text{bound}} \le x^k \le x_{\text{bound}}, \qquad k = 1, \ldots, n$$

$$A_0^i + \sum_{k=1}^{n} x_k A_k^i + \sum_{k=1}^{n}\sum_{\ell=1}^{n} x_k x_\ell K_{k\ell}^i \preccurlyeq \lambda I_{n\times n}, \qquad i = 1, \ldots, N\,.$$

This is exactly the problem formulation we used in our numerical experiments (compare Section 11.3).

Chapter 11

Benchmarks and Numerical Experiments

11.1 Numerical Experiments in Structural Optimization

11.1.1 Global Stability and Self Vibration

Throughout this section we present results of several numerical studies on (partly nonlinear) semidefinite optimization problems arising from structural optimization, when global stability is taken into account (compare Section 10.1.3). Our studies will involve both – truss topology optimization problems (TTO) (compare section 10.1.2) and free material optimization problems (FMO) (compare section 10.1.1). In each of the test scenarios described below, we try to solve a structural optimization problem in the so called minimum weight formulation using different numbers of nodes in the ground structure (TTO case) respectively different levels of discretizations (FMO case) and different types of constraints. In particular we tried to solve the following problems for each ground structure/level of discretization:

- Problem (10.16): no stability is taken into account.
- Problem (10.18): stability is "approximated" by constraints on the self vibration of the system.
- Problem (10.17): stability is considered in the sense of a critical buckling force.

Note that the first two problems are linear semidefinite programs, whereas the third problem is highly nonlinear. Apart from the nonlinearity there is a second difficulty arising, when solving problems of type (10.17), namely the computational complexity of the Hessian assembling of the augmented Lagrangian. This is the reason, why we used for some of the larger FMO problems an iterative method to solve step 2.1 of Algorithm 9.4.1 (compare also Section 9.1.4), which does not require the explicit

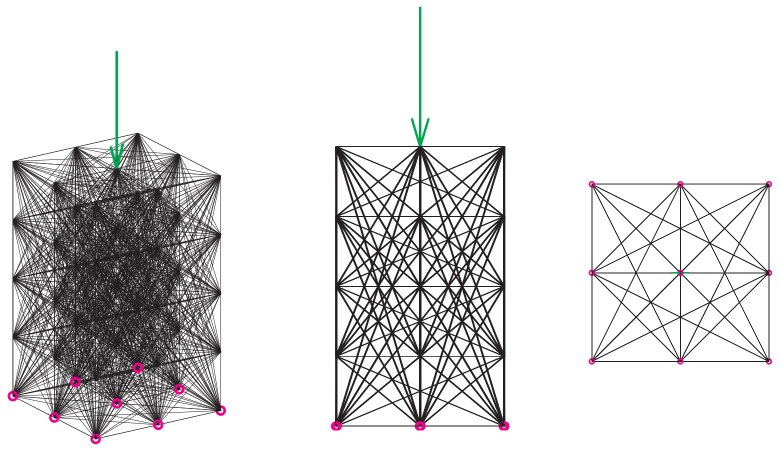


Figure 11.1: Scenario *Tower*; ground structure

calculation of the Hessian. Note that these results (marked by an asterisk in the tables below) where computed with reduced accuracy. Note further that the same approach could not be successfully applied to (large) TTO problems so far. For this reason we are not able to present results for the stability problems of the largest TTO instances below.

Scenario *Tower* In our first test scenario we consider a truss, which is fixed at the bottom nodes and subject to a single vertical force at the top-middle node (compare Figure 11.1). The result of standard TTO (with no stability constraint) is just a single vertical bar (as depicted in Figure 11.2) —an extremely unstable structure. Adding a constraint on the self vibration we obtain the truss shown in Figure 11.3. Finally using a constraint on the global stability, we obtain a truss presented in Figure 11.4. The latter two trusses are obviously stable.

Table 11.1: Scenario *Tower*; problem dimensions

#Nodes	#Vars	Matsize – compliance constraint	Matsize – stability constraint	#linear constraints
3x3x3	315	55	54	316
3x3x4	594	82	81	595
3x3x5	954	109	108	955
3x3x11	4825	271	270	4826
3x3x16	10260	406	405	10261

Table 11.1 shows the problem dimensions for increasing number of nodes in the ground structure. In Table 11.2 we present the corresponding computational results, which are in particular: number of outer/inner iterations and run time in seconds. Note that single-load problems are usually solved much more efficiently using a different formulation (compare section 10.1). However, we have chosen the linear SDP formulation for comparison purposes.

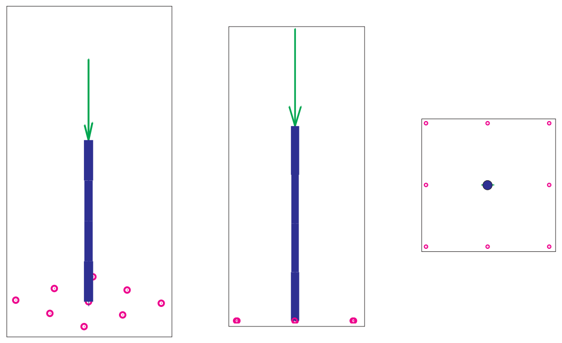


Figure 11.2: Scenario *Tower*; single-load result

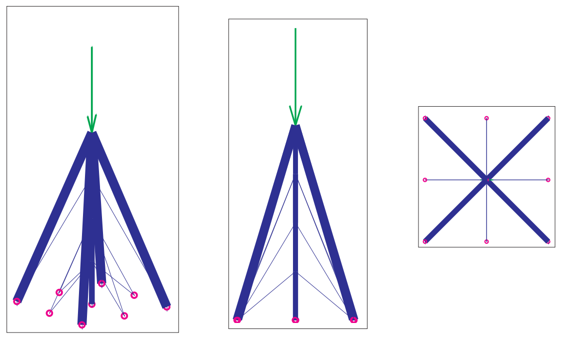


Figure 11.3: Scenario *Tower*; vibration constraint added

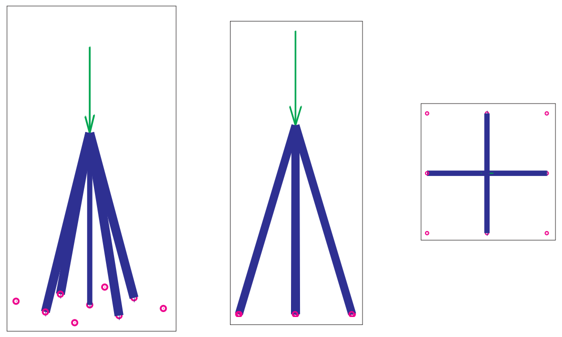


Figure 11.4: Scenario *Tower*; stability constraint added

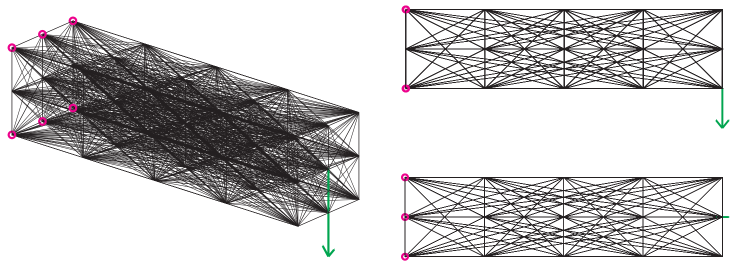


Figure 11.5: Scenario *Cantilever*; ground structure

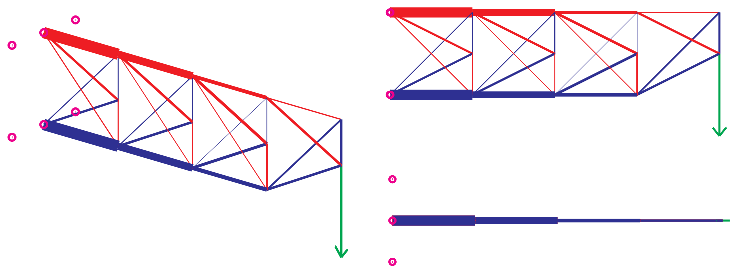


Figure 11.6: Scenario *Cantilever*; single-load result

Table 11.2: Numerical results for sample case *Tower*

#Nodes	Sl-Iter	Sl-time	Vib-Iter	Vib-time	Buck-iter	Buck-time
3x3x3	10/105	2	10//99	4	10//96	135
3x3x4	11/115	11	11/103	18	11//95	1041
3x3x5	12/121	36	12/118	58	12/116	10056
3x3x11	14/161	4268	14/183	6075	–	–
3x3x16	16/187	43594	15/225	58649	–	–

Scenario *Cantilever* In our second example we try to find an optimal truss, which is fixed at the upper and lower left nodes. Furthermore the truss is subject to a vertical force at the inner right node (compare Figure 11.5). The result of standard TTO, presented in Figure 11.6 is just a two-dimensional construction. It is easy to understand that such a construction must be unstable against forces having a non-zero component, lying not in the plane described by the single-load-truss. The stability can be significantly improved by adding a vibration constraint (see Figure 11.7) or a global stability

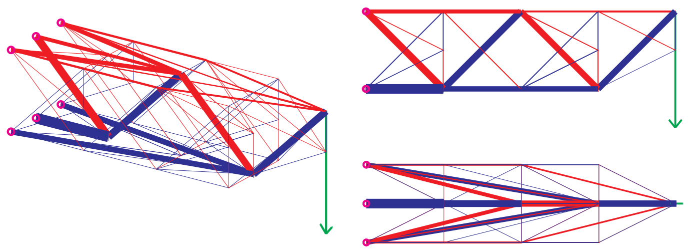


Figure 11.7: Scenario *Cantilever*; vibration constraint added

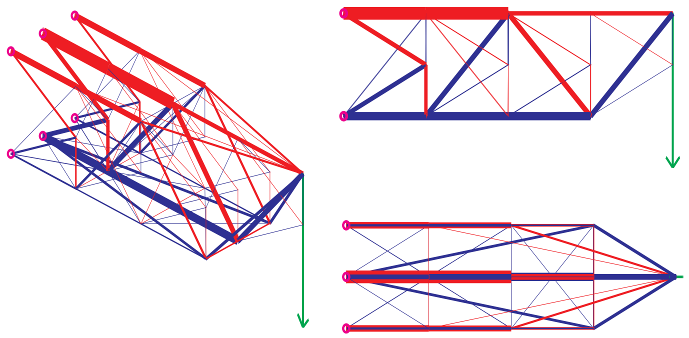


Figure 11.8: Scenario *Cantilever*; stability constraint added

constraint (see Figure 11.8).

Table 11.3: Scenario *Cantilever*; problem dimensions

#Nodes	#Vars	Matsize – compliance constraint	Matsize – stability constraint	#linear constraints
3x2x2	261	55	54	262
5x3x2	420	73	72	421
5x3x3	954	109	108	955
6x3x6	5625	271	270	5626
7x3x7	10521	379	378	10522

Table 11.3 shows the problem dimensions for all ground structures we have used for Test Scenario *Cantilever*.

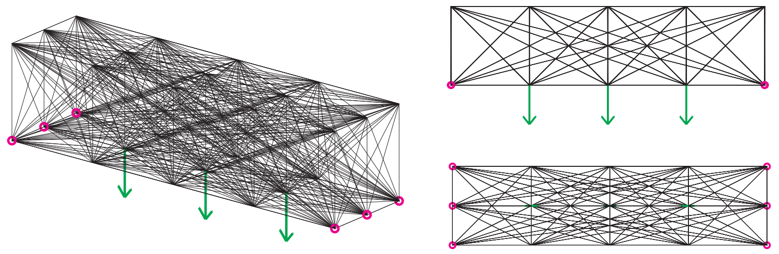


Figure 11.9: Scenario *Bridge*; ground structure

The computational results for this test case can be seen in Table 11.4.

Table 11.4: Scenario *Cantilever*; numerical results

#Nodes	Sl-Iter	Sl-time	Vib-Iter	Vib-time	Buck-iter	Buck-time
3x2x2	9/101	2	9//99	4	9/103	79
5x3x2	9/115	6	9/126	12	9/111	386
5x3x3	9/110	36	10/137	71	10/137	12120
6x3x6	11/189	7779	11/227	11286	–	–
7x3x7	12/223	54844	12/240	65897	–	–

Scenario *Bridge* In the third and last TTO example we consider a truss, fixed at the left-most and right-most bottom nodes. Moreover the truss is subject to vertical forces at all inner bottom nodes (compare Figure 11.9). As in the preceding example all bars in the standard TTO result, depicted in Figure 11.10, lie in a hyperplane parallel spanned by the initial forces. Thus the truss is again unstable. As before we can achieve "better" constructions by adding a constraint on the self vibration (compare Figure 11.11) or a global stability constraint (see Figure 11.12).

Table 11.5: Scenario *Bridge*

#Nodes	#Vars	Matsize – compliance constraint	Matsize – stability constraint	#linear constraints
3x2x2	261	55	54	262
5x3x2	420	73	72	421
5x3x3	975	118	117	976
6x3x6	5763	307	306	5764
7x3x7	10716	424	423	10717

Tables 11.5 and 11.6 show the problem sizes and the computational results for this scenario presented in the same style as above.

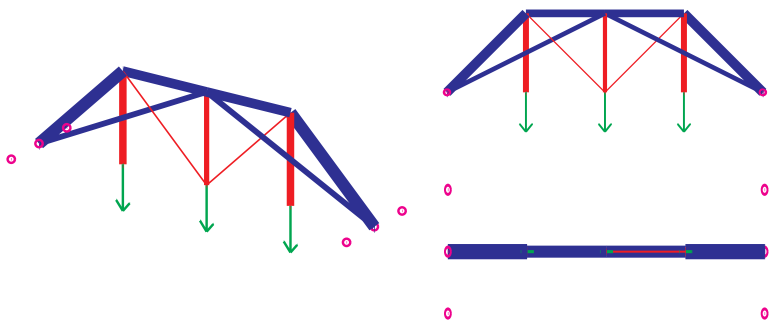


Figure 11.10: Scenario *Bridge*; single-load result

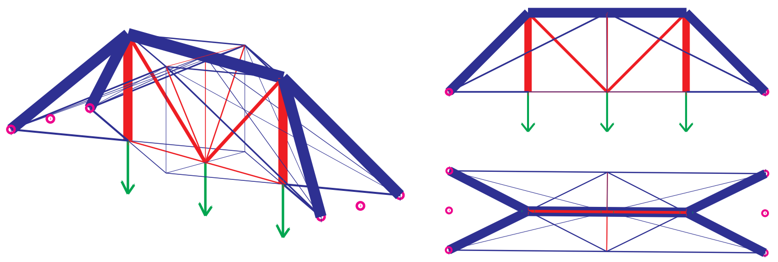


Figure 11.11: Scenario *Bridge*; vibration constraint added

Table 11.6: Scenario *Bridge*; numerical results

#Nodes	Sl-Iter	Sl-time	Vib-Iter	Vib-time	Buck-iter	Buck-time
3x2x2	9//82	1	10/101	3	10//92	71
5x3x2	9/80	4	10//87	8	10//98	334
5x3x3	10/96	31	11/104	50	11/103	10170
6x3x6	12/135	5954	13/194	10281	–	–
7x3x7	13/161	41496	13/222	65670	–	–

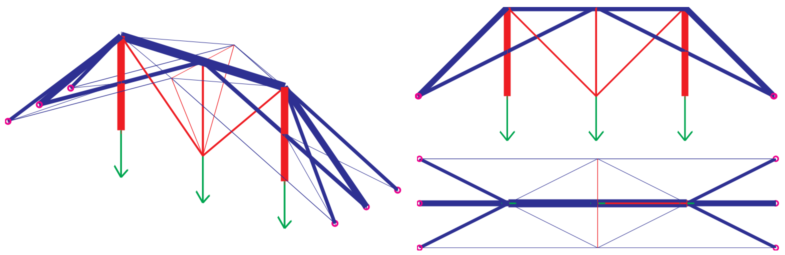


Figure 11.12: Scenario *Bridge*; stability constraint added

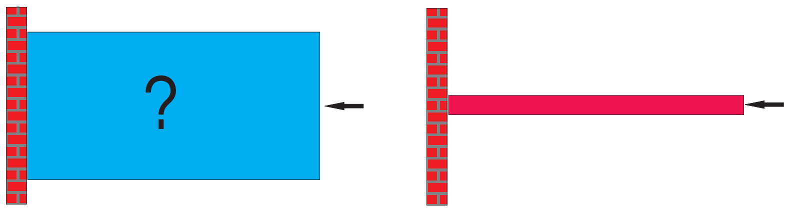


Figure 11.13: Scenario *Plate* – Geometry, boundary conditions & forces (left) and single-load result (right)

Scenario *Plate* The next scenario deals with a two-dimensional FMO problem. Consider a plate (depicted in Figure 11.13, left side) fixed on the left-hand side and subjected to a horizontal load concentrated on a small part of the right-hand side. The right part of Figure 11.13 shows a result of the minimum weight problem (10.12) (with no stability/vibration constraint) for a zero-Poisson-ration material. The optimal structure only consists of horizontal fibers and is, as such, extremely unstable to other than the given load. Figure 11.14 presents the results of problems (10.18) and (10.17) for the same material; the structures are obviously much more stable.

Table 11.7: Scenario *Plate*; problem dimensions

#Elements (=#Vars)	Matsize – compliance constraint	Matsize – stability constraint	#linear constraints
10 x 20 = 200	440	441	401
14 x 30 = 420	900	901	841
20 x 40 = 800	1680	1681	1601
30 x 60 = 1800	3720	3721	3601

Table 11.7 shows the problem dimensions for four discretizations with increasing number of elements. In Table 11.7 we present the corresponding computational results, which are again: number of outer/inner iterations and run time in seconds.

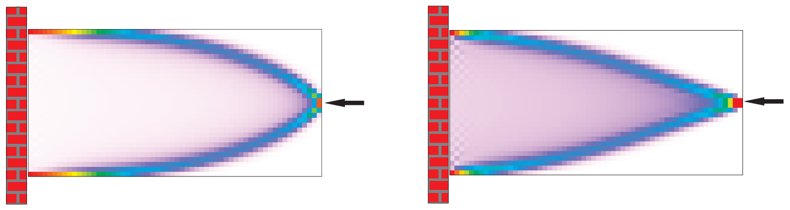


Figure 11.14: Scenario *Plate* – vibration result (left) and stability result (right)

Table 11.8: Scenario *Plate*; numerical results

#Elements	Sl-Iter	Sl-time	Vib-Iter	Vib-time	Buck-iter	Buck-time
10 x 20 = 200	14/96	52	14/99	119	14/188	5709
14 x 30 = 420	19/119	515	15/100	923	18/208	57108
20 x 40 = 800	22/141	2968	16/108	4721	20/180	$27732^{(*)}$
30 x 60 = 1800	23/142	22334	18/127	30705	13/137	$115810^{(*)}$

Scenario *Plate-Narrow* Scenario *Plate-Narrow* is very similar to scenario *Plate*. The only difference is in the geometry of the design space (compare Figure 11.15, left side). Again the pure single-load result, depicted in Figure 11.15 is unstable. Just as before we get much more stable results, when solving problems (10.18) and (10.17) for the same material (see Figure 11.16).

Table 11.9: Scenario *Plate-Narrow*; problem dimensions

#Elements (=#Vars)	Matsize – compliance constraint	Matsize – stability constraint	#linear constraints
4 x 32 = 128	320	321	257
8 x 64 = 512	1152	1153	1025
12 x 96 = 1152	2496	2497	2305
16 x 128 = 2048	4352	4353	4097

Table 11.9 shows the problem dimensions for four discretizations with increasing number of elements, whereas the corresponding computational results are given in Table 11.10.

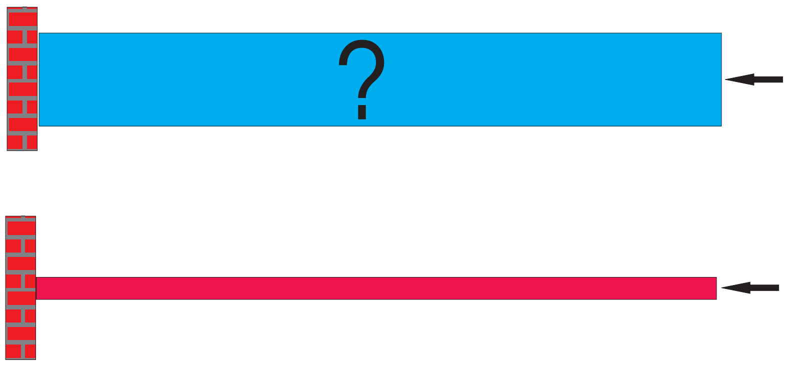


Figure 11.15: Scenario *Plate-Narrow* – Geometry, boundary conditions & forces (top), single-load result (bottom)

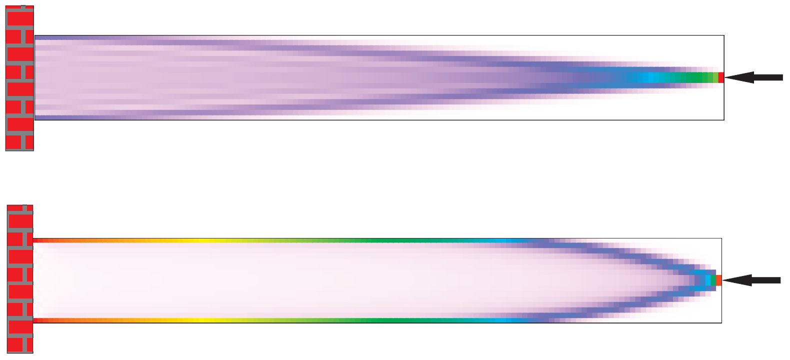


Figure 11.16: Scenario *Plate-Narrow* – vibration result (top) and stability result (bottom)

Table 11.10: Scenario *Plate-Narrow*; numerical results

#Elements	Sl-Iter	Sl-time	Vib-Iter	Vib-time	Buck-iter	Buck-time
4 x 32 = 128	13/76	16	13/84	50	16/96	936
8 x 64 = 512	20/135	976	16/104	1587	17/101	51743
12 x 96 = 1152	35/190	8640	18/137	13265	12/98	$47450^{(*)}$
16 x 128 = 2048	35/199	30616	19/171	57125	8/72	$75600^{(*)}$

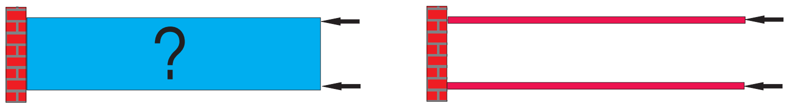


Figure 11.17: Scenario *Two-Forces* – Geometry, boundary conditions & forces (left), single-load result (right)

Scenario *Two-Forces* As in the preceding FMO scenarios, we consider a plate. Again the plate is fixed on the left-hand side, but this time there are two horizontal loads concentrated on two small areas of the right-hand side (see Figure 11.17, left side). The right part of Figure 11.17 shows a result of the minimum weight problem (10.12) (with no stability/vibration constraint). We can clearly see that the structure consists of two horizontal fibers, one at the bottom and another one at the top of the design space. Both fibres are not connected and thus the structure is again unstable. In Figure 11.18 we present results of problems (10.18) and (10.17) for the same material. In both cases we can see that most of the material is distributed again at the bottom and the top of the design space, but this time the fibers at the bottom and the top are connected. Therefore the results are more stable with respect to loading scenarios other than the given one.

Table 11.11: Scenario *Two-Forces*; problem dimensions

#Elements (=#Vars)	Matsize – compliance constraint	Matsize – stability constraint	#linear constraints
7 x 28 = 196	406	407	393
10 x 40 = 400	880	881	841
14 x 56 = 784	1680	1681	1601
20 x 80 = 1600	3360	3361	3601

Table 11.11 and Table 11.12 show problem dimensions and numerical results for this test scenario.

Table 11.12: Scenario *Two-Forces*; numerical results

#Elements	Sl-Iter	Sl-time	Vib-Iter	Vib-time	Buck-iter	Buck-time
7 x 28 = 196	13/81	43	13/76	111	13/92	2723
10 x 40 = 400	14/93	367	14/89	717	14/104	25930
14 x 56 = 784	15/110	2198	17/110	4363	15/110	$21934^{(*)}$
20 x 80 = 1600	16/120	11874	16/125	20696	14/105	$53187^{(*)}$

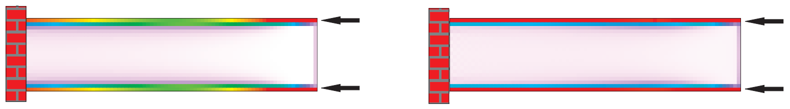


Figure 11.18: Scenario *Two-Forces* – vibration result (left) and stability result (right)

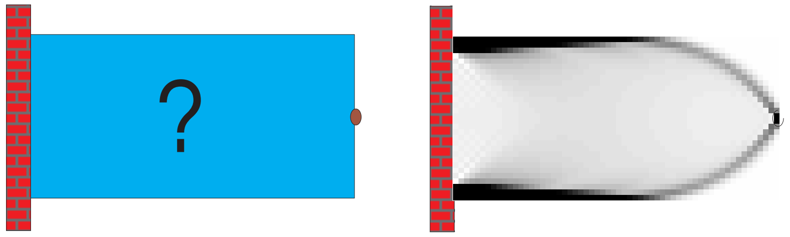


Figure 11.19: Maximizing Vibration Mode – Geometry and boundary conditions (left), Optimal Material (Density Plot) (right)

11.1.2 Maximizing the Minimal Eigenfrequency

We want to conclude this section by presenting a numerical study on material problems of type (10.19). In our model example we consider a plate, which is fixed on the left hand side. Furthermore there is a prescribed mass on the right hand side of the design space (compare Figure 11.19, left side). Now we are interested in finding a material, which supports the prescribed mass and the minimal eigenfrequency of which is as large as possible. Note that the resulting semidefinite problem is bi-linear and quasi-convex. Moreover the problem inherent structure is more favorable as in the nonlinear semidefinite programming problems investigated in the preceding section. Table 11.13 shows results for five different levels of discretizations. Next to computational times and numbers of iterations we list problem dimensions and the density of the constraint matrix. At the right hand side of Figure 11.19 we can see a density plot of the optimal material in the above sense for the finest level of discretization.

Table 11.13: Maximizing Vibration Mode; numerical results

#Elements	#Vars	#Constr	Matsize	#Iterations	#Nwtstps	Time	Density
16	17	33	44	15	81	1	1.0
200	201	401	440	20	94	28	.100
420	421	841	900	21	102	152	.0622
800	801	1601	1680	23	109	654	.0415
1800	1801	3601	3720	24	115	8922	.0213

Remark . If we look at the tables presented in this section, we observe that for almost all test scenarios the number of Newton steps required by PENNON in order to achieve a prescribed precision is growing with increasing size of the test problems. Let us just mention that this is not a general behavior of PENNON (compare, for example, Table 11.13 or Section 11.4), but rather due to the fact that the problems presented throughout this section become more and more ill-conditioned with increasing dimension.

11.2 Numerical Experiments with *COMPl$_e$ib* -SOF Problems

Below we present results of a numerical benchmark with PENBMI , a specialized BMI-version of PENNON , for the static output feedback problems in *COMPl$_e$ib* (*CO*nstrained *M*atrix–optimization *P*roblem *lib*rary), a benchmark collection for a very wide variety of algorithms solving matrix optimization problems. The heart of *COMPl$_e$ib* is the MATLAB function file *COMPleib.m* which is available over the internet. This function returns the data matrices A, B_1, B, C_1, C, D_{11}, D_{12} and D_{21} of the system (10.20) of each individual *COMPl$_e$ib* example. For a detailed description of the library we refer to [56].

All tests were performed on a 2.5 GHz Pentium 4 processor machine with 1 GB RDRAM under the operating system Linux. YALMIP 3.0 was used to build the $\mathcal{H}_2$- and $\mathcal{H}_\infty$-models and to transform them to PENBMI standard input. The tests where performed in the following way: We started to run all cases using default options. For many cases the algorithm was restarted few times and the initial multipliers where adapted automatically. In a second run the test cases where re-run using the automatically ”optimized” initial multipliers. The results of these reruns are presented in the tables below. Of course apart from CPU-time the results of the first and second runs are identical.

Tables 11.14 and 11.15 show the results of PENBMI on $\mathcal{H}_2$-BMI and $\mathcal{H}_\infty$-BMI problems. The results can be divided into seven groups:

- The first group consists of all test cases solved without any difficulties.
- The second and third group contain all cases, for which we had to relax our stopping criterion. These examples are marked by ”a” in the tables below, if the achieved precision is still close to our predefined stopping criterion, and by ”A”, if the deviation is significant.
- Then there are examples, for which we could calculate almost feasible solutions, but which failed to satisfy the Hurwitz-criterion, namely AC5 and NN10.
- The fourth group consists of medium and small scale cases for which the code failed due to ill conditioning. In the $\mathcal{H}_2$-setting these cases are AC7, AC9, AC13, AC18, JE1, JE2, JE3, REA4, DIS5, WEC1, WEC2, WEC3, UWV, PAS, NN1, NN3, NN5, NN6, NN7, NN9, NN12 and NN17, in the $\mathcal{H}_\infty$-setting JE1, JE2, JE3, REA4, DIS5, UWV, PAS, TF3, NN1, NN3, NN5, NN6, NN7 and NN13.
- The cases in the sixth group are large scale, ill conditioned problems, where PENBMI ran out of time (AC10, AC14, CSE2, EB5).
- Finally, for very large test cases our method ran of memory (HS1, BDT2, EB6, TL, CDP, NN18).

Only the cases of the first three groups are listed in the tables below.

Table 11.14: Results of PENBMI on $\mathcal{H}_2$-BMI problems

Ex.	CPU (sec)	n	m	nx / ny / nu / nw / nz	$\lambda^r_{\max} A(F)$	$\mathcal{H}_2$-perf	prec
AC1	0.79	27	17	5 / 3 / 3 / 3 / 2	-2.65e-01	1.007e-03	
AC2	1.60	39	20	5 / 3 / 3 / 3 / 5	-2.01e-07	5.041e-02	
AC3	0.72	38	20	5 / 4 / 2 / 5 / 5	-4.53e-01	4.570e+00	
AC4	0.79	15	14	4 / 2 / 1 / 2 / 2	-5.00e-02	1.103e+01	
AC6	3.22	64	28	7 / 4 / 2 / 7 / 7	-8.72e-01	3.798e+00	
AC8	8.14	53	29	9 / 5 / 1 / 10 / 2	-3.14e-01	1.194e+00	
AC11	3.56	38	20	5 / 4 / 2 / 5 / 5	-3.23e-01	3.942e+00	
AC12	1.28	23	13	4 / 4 / 3 / 3 / 1	-4.41e-01	2.424e-04	a
AC15	0.53	37	18	4 / 3 / 2 / 4 / 6	-3.46e-01	1.261e+01	
AC16	2.66	39	18	4 / 4 / 2 / 4 / 6	-3.19e-01	1.230e+01	
AC17	0.35	22	16	4 / 2 / 1 / 4 / 4	-7.19e-01	4.111e+00	
HE1	0.22	15	14	4 / 1 / 2 / 2 / 2	-1.21e-01	9.546e-02	
HE2	0.76	24	16	4 / 2 / 2 / 4 / 4	-3.17e-01	3.434e+00	
HE3	1.96	115	34	8 / 6 / 4 / 1 / 10	-1.43e-01	8.118e-01	
HE4	21.72	138	36	8 / 6 / 4 / 8 / 12	-9.28e-02	2.082e+01	
HE5	8.66	54	28	8 / 2 / 4 / 3 / 4	-9.83e-03	5.438e+00	
HE6	1101	370	76	20 / 6 / 4 / 6 / 16	-5.00e-03	6.317e+01	a
HE7	5135	370	76	20 / 6 / 4 / 9 / 16	-5.00e-03	6.372e+01	A
REA1	0.95	26	16	4 / 3 / 2 / 4 / 4	-1.68e+00	1.820e+00	
REA2	0.55	24	16	4 / 2 / 2 / 4 / 4	-1.22e+00	1.862e+00	
REA3	26.29	159	48	12 / 3 / 1 / 12 / 12	-2.06e-02	1.209e+01	
DIS1	6.45	88	32	8 / 4 / 4 / 1 / 8	-4.33e-01	2.660e+00	a
DIS2	0.23	16	12	3 / 2 / 2 / 3 / 3	-7.58e-01	1.416e+00	
DIS3	2.47	58	24	6 / 4 / 4 / 6 / 6	-1.40e+00	1.839e+00	
DIS4	3.73	66	24	6 / 6 / 4 / 6 / 6	-1.01e+00	1.692e+00	
TG1	107	114	40	10 / 2 / 2 / 10 / 10	-3.39e-01	2.231e+01	a
AGS	167	160	48	12 / 2 / 2 / 12 / 12	-2.03e-01	6.995e+00	
BDT1	3.56	96	39	11 / 3 / 3 / 1 / 6	-1.88e-03	1.745e-02	
MFP	0.59	26	16	4 / 2 / 3 / 4 / 4	-3.20e-02	9.162e+00	
IH	376	407	74	21 / 10 / 11 / 21 / 11	-4.79e-01	8.260e-04	
CSE1	50.96	308	72	20 / 10 / 2 / 1 / 12	-5.29e-02	1.208e-02	
EB1	20.72	59	32	10 / 1 / 1 / 2 / 2	-5.52e-02	1.700e+00	
EB2	26.91	59	32	10 / 1 / 1 / 2 / 2	-8.68e-02	7.736e-01	
EB3	8.92	59	32	10 / 1 / 1 / 2 / 2	-4.70e-02	8.345e-01	
EB4	499	214	62	20 / 1 / 1 / 2 / 2	-1.71e-05	5.043e+02	A
TF1	1.65	46	25	7 / 4 / 2 / 1 / 4	-6.88e-02	1.826e-01	
TF2	37.69	44	25	7 / 3 / 2 / 1 / 4	-1.00e-05	1.449e-01	A
TF3	2.61	44	25	7 / 3 / 2 / 1 / 4	-3.20e-03	2.781e-01	
PSM	2.57	49	26	7 / 3 / 2 / 2 / 5	-7.84e-01	1.504e+00	
NN2	0.28	7	8	2 / 1 / 1 / 2 / 2	-4.08e-01	1.565e+00	

Results of PENBMI on $\mathcal{H}_2$-BMI problems *(cont.)*

Ex.	CPU (sec)	n	m	nx / ny / nu / nw / nz	$\lambda^r_{\max} A(F)$	$\mathcal{H}_2$-perf	prec
NN4	0.38	26	16	4 / 3 / 2 / 4 / 4	-5.87e-01	1.875e+00	
NN8	0.34	16	12	3 / 2 / 2 / 3 / 3	-4.64e-01	2.280e+00	
NN11	285	157	51	16 / 5 / 3 / 3 / 3	-3.45e-01	1.198e+02	A
NN13	3.00	31	21	6 / 2 / 2 / 3 / 3	-2.37e+00	2.622e+01	
NN14	3.48	31	21	6 / 2 / 2 / 3 / 3	-1.83e+00	3.536e+01	
NN15	0.35	20	13	3 / 2 / 2 / 1 / 4	-1.17e-01	4.903e-02	
NN16	46.57	62	28	8 / 4 / 4 / 8 / 4	-8.06e-06	3.073e-01	A

Table 11.15: Results of PENBMI on $\mathcal{H}_\infty$-BMI problems

Ex.	CPU (sec)	n	m	nx / ny / nu / nw / nz	$\lambda^r_{\max} A(F)$	$\mathcal{H}_{\inf}$-perf	prec
AC1	0.37	25	16	5 / 3 / 3 / 3 / 2	-2.03e-01	2.505e-06	
AC2	0.57	25	19	5 / 3 / 3 / 3 / 5	-2.26e-07	1.115e-01	
AC3	4.96	24	21	5 / 4 / 2 / 5 / 5	-4.12e-01	3.402e+00	
AC4	0.82	13	13	4 / 2 / 1 / 2 / 2	-5.00e-02	9.355e-01	
AC6	1.28	37	29	7 / 4 / 2 / 7 / 7	-7.64e-01	4.114e+00	
AC7	6.08	48	24	9 / 2 / 1 / 4 / 1	-1.80e-02	2.097e+00	a
AC8	6.63	51	31	9 / 5 / 1 / 10 / 2	-3.55e-01	2.367e+00	a
AC9	42.42	76	33	10 / 5 / 4 / 10 / 2	-1.62e-01	1.039e+00	
AC11	8.00	24	21	5 / 4 / 2 / 5 / 5	-4.33e+00	2.820e+00	
AC12	1.59	23	13	4 / 4 / 3 / 3 / 1	-1.13e-01	3.978e-01	a
AC13	6270	419	113	28 / 4 / 3 / 28 /28	-2.13e-02	9.438e+02	A
AC15	0.19	17	19	4 / 3 / 2 / 4 / 6	-4.51e-01	1.517e+01	
AC16	1.08	19	19	4 / 4 / 2 / 4 / 6	-9.15e-01	1.486e+01	
AC17	0.13	13	17	4 / 2 / 1 / 4 / 4	-7.26e-01	6.612e+00	
AC18	56.77	60	29	10 / 2 / 2 / 3 / 5	1.33e+04	4.410e+02	A
HE1	0.63	13	13	4 / 1 / 2 / 2 / 2	-1.29e-01	1.538e-01	a
HE2	0.16	15	17	4 / 2 / 2 / 4 / 4	-4.03e-01	4.249e+00	
HE3	4.32	61	28	8 / 6 / 4 / 1 / 10	-2.22e-01	9.500e-01	
HE4	18.48	61	37	8 / 6 / 4 / 8 /12	-6.76e-02	2.284e+01	
HE5	4.91	45	24	8 / 2 / 4 / 3 / 4	-1.26e-01	8.895e+00	
HE6	1066	235	63	20 / 6 / 4 / 6 /16	-5.00e-03	9.712e+02	A
HE7	1096	235	66	20 / 6 / 4 / 9 /16	-5.00e-03	1.357e+03	A
REA1	0.98	17	17	4 / 3 / 2 / 4 / 4	-2.03e+00	8.657e-01	
REA2	3.00	15	17	4 / 2 / 2 / 4 / 4	-2.63e+00	1.149e+00	
REA3	2.85	82	49	12 / 3 / 1 / 12 /12	-2.07e-02	7.425e+01	a
DIS1	10.23	53	26	8 / 4 / 4 / 1 / 8	-7.15e-01	4.161e+00	

Results of PENBMI on $\mathcal{H}_\infty$-BMI problems *(cont.)*

Ex.	CPU (sec)	n	m	nx / ny / nu / nw / nz	$\lambda^r_{\max} A(F)$	$\mathcal{H}_{\inf}$-perf	prec
DIS2	0.30	11	13	3 / 2 / 2 / 3 / 3	-9.95e-01	1.055e+00	
DIS3	12.03	38	25	6 / 4 / 4 / 6 / 6	-1.31e+00	1.065e+00	
DIS4	2.83	46	25	6 / 6 / 4 / 6 / 6	-1.45e+00	7.318e-01	
TG1	3.84	60	41	10 / 2 / 2 / 10 /10	-3.28e-01	1.285e+01	a
AGS	4.18	83	49	12 / 2 / 2 / 12 /12	-2.07e-01	8.173e+00	
WEC1	10.30	68	41	10 / 4 / 3 / 10 /10	-8.08e-01	4.050e+00	a
WEC2	33.93	68	41	10 / 4 / 3 / 10 /10	-1.19e+00	4.245e+00	a
WEC3	10.91	68	41	10 / 4 / 3 / 10 /10	-1.14e+00	4.450e+00	a
BDT1	5.31	76	30	11 / 3 / 3 / 1 / 6	-3.31e-03	2.662e-01	
MFP	0.54	17	17	4 / 2 / 3 / 4 / 4	-3.64e-02	3.159e+01	a
IH	326	342	75	21 / 10 /11 / 21 /11	-2.15e-01	4.187e-02	
CSE1	36.05	231	54	20 / 10 / 2 / 1 /12	-9.22e-02	1.988e-02	
EB1	1.83	57	25	10 / 1 / 1 / 2 / 2	-5.61e-02	3.123e+00	
EB2	2.29	57	25	10 / 1 / 1 / 2 / 2	-7.83e-02	2.020e+00	
EB3	1.89	57	25	10 / 1 / 1 / 2 / 2	-3.95e-02	2.058e+00	
EB4	74.81	212	45	20 / 1 / 1 / 2 / 2	-2.01e-07	2.056e+00	
TF1	5.03	37	20	7 / 4 / 2 / 1 / 4	-6.57e-02	4.042e-01	A
TF2	7.77	35	20	7 / 3 / 2 / 1 / 4	-1.00e-05	2.556e-01	
PSM	0.74	35	22	7 / 3 / 2 / 2 / 5	-1.00e+00	9.202e-01	
NN2	0.08	5	9	2 / 1 / 1 / 2 / 2	-6.36e-01	2.222e+00	
NN4	0.90	17	17	4 / 3 / 2 / 4 / 4	-9.42e-01	1.359e+00	
NN8	1.85	11	13	3 / 2 / 2 / 3 / 3	-1.39e+00	2.885e+00	
NN9	8.56	22	17	5 / 2 / 3 / 2 / 4	-3.50e-01	2.500e+01	A
NN11	268	152	39	16 / 5 / 3 / 3 / 3	-5.43e-01	1.359e-01	A
NN12	5.75	26	25	6 / 2 / 2 / 6 / 6	-1.69e-01	1.629e+01	
NN14	1.41	26	19	6 / 2 / 2 / 3 / 3	-2.30e+00	1.748e+01	a
NN15	0.35	11	12	3 / 2 / 2 / 1 / 4	-9.20e-01	9.809e-02	
NN16	1.64	53	29	8 / 4 / 4 / 8 / 4	-7.80e-05	9.556e-01	
NN17	0.36	9	10	3 / 1 / 2 / 1 / 2	-4.36e-01	1.122e+01	

11.3 Numerical Experiments with Simultaneous Stabilization BMIs

We collected a suite of simultaneous stabilization problems selected from the recent literature. Table 11.16 gives basic characteristics of these problems.

Table 11.16: Collection of simultaneous stab. problems

problem	system order	contr. order	nb. of vertices	known feas. point	ref.
discrete	3	2	1	[200 100 50]	[83]
f4e	3	0	4	-0.8692	[1]
helicopter	3	2	4	[1.865 2.061 1.992 4.335 10.50]	[42]
interval1_0	3	0	16	0.0380	[13]
interval1_1	3	1	16	[66.16 52.01 38.18]	[13]
interval2_0	2	0	8	226.5	[13]
interval2_1	2	1	8	[397.4 214.4 -135.8]	[13]
mass_spring	1	4	2	[0.2887 1.6761 -2.1434 3.0755 2.7278]	[85]
oblique_wing	4	0	64	0.381	[3]
parametric	2	0	3	24.1489	[23]
servo_motor	2	1	4	[1.300, 26.88, 5.439, 0]	[14]
toy1	3	1	1	[0.5647 1.6138 1.5873]	[44]
toy2	3	1	1	[-2.9633 -2.2693 1.2783]	[44]

These problems were formulated as feasibility BMIs (10.28) and solved again by PENBMI on a 2.5 GHz Pentium 4 processor machine with 1 GB RDRAM under Linux. All test cases were solved for various initial points. The initial points were generated by a strategy described in [42]. Note that the solution of most problems is obtained in fractions of seconds by PENBMI . Thus many initial points cause not a real problem.

Another important issue was the choice of the weighting parameter w in (10.28). After performing many experiments, we obtained the best results with $w = 0.0001$.

Table 11.17 presents the results. For each problem, we show the number of initial points, the minimum and maximum number of internal PENBMI iterations (Hessian evaluations) and the number of successful and unsuccessful runs. By a successful run we mean the case when a feasible point was found (Recall that all problems are feasible). We also show the number of successful runs when the feasible point was at the upper or lower bound; in this case the point may not be a stabilizer and has to be checked a-posteriori. Unsuccessful run means that the optimal value in (10.28) was greater than or equal to zero. Note that in *all* cases PENBMI converged to a critical point.

Table 11.17: Results of PENBMI on simultaneous stab. problems

problem	no. of init. pts	min. iter	max. iter	succ. cases	bound reached	unsucc. cases
discrete	4	17	86	4	2	0
f4e	5	63	78	4	0	1
helicopter	5	85	107	3	0	2

Results of PENBMI on simultaneous stab. problems *(cont.)*

problem	no. of init. pts	min. iter	max. iter	succ. cases	bound reached	unsucc. cases
interval1_0	17	56	67	17	0	0
interval1_1	17	66	104	16	0	1
interval2_0	9	53	63	6	6	3
interval2_1	9	63	131	4	4	5
mass_spring	2	348	397	2	0	0
oblique_wing	65	65	90	64	0	1
parametric	4	62	73	3	3	0
servo_motor	5	84	179	5	5	0
toy1	4	16	17	4	0	0
toy2	4	66	243	4	3	0

A detailed analysis of these results is given again in [42].

11.4 A Benchmark with a Collection of Random BMI Problems

In the preceding sections we have investigated the behavior of PENNON for a special class of nonlinear SDP problems, namely BMI problems. In Section 10.2 we mainly presented numerical results for small to medium scale examples of this category. The reason for this choice was twofold: First, many of the large scale examples presented in literature (compare, for example, large scale cases in *COMPl$_e$ib*), were simply to huge and therefore PENNON was not applicable. Second, the existing large scale examples were often very ill conditioned and we were not able to achieve reasonable results. In order to demonstrate that PENNON is generally able to copy with large scale nonlinear semidefinite cases, we created a library of random BMI problems of the following form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n,\lambda\in\mathbb{R}} \lambda \quad \text{s.t.} \\ &-|c^\ell| \le x^\ell \le |c^\ell|, \qquad \ell = 1,\dots,n\,, \\ &\mathcal{B}(x) = A_0 + \sum_{k=1}^{n} x_k A_k + \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j K_{i,j} \preccurlyeq \lambda I_m\,, \end{aligned} \tag{11.1}$$

where $A_k \in \mathbb{S}^m$, $k = 0,\dots,n$, $K_{i,j} \in \mathbb{S}^m$, $i,j = 1,\dots,d$ and $0 < d \le n$. The bounds c_ℓ, $\ell = 1,\dots,n$ were generated by the MATLAB function `randn`, a random generator, which produces arrays of random numbers whose elements are normally distributed with mean 0, variance 1 and standard deviation 1. The matrices A_k, $k = 0,\dots,n$ and $K_{i,j}$, $i,j = 1,\dots,d$ were generated by the MATLAB function `sprandn`: `sprandn` generates a sparse random matrix with approximately rm^2 normally distributed non-zero entries, where r is a real value between 0 and 1. We symmetrized the matrices by copying the upper triangular part to the lower one after creation.

The test cases we created can be classified by the parameter set $(n; m; d; r)$. For each combination of parameters we created at least 20 instances of problem (11.1). There are two major groups of test cases, which we used to demonstrate the capabilities of the code:

1. Category I: In the first group we collected cases where n is significantly larger than m, d is chosen so that the inequality $d^2 > n$ holds and the density r of the data matrices is not too small. This is a quite typical setting for problems arising for instance in control theory. The main difficulty with these problems is that they are highly nonlinear; this is a consequence of the comparably large number of nonlinear terms in the bilinear constraint.

2. Category II: The second group of examples is characterized by the equation $n = m$, $d \approx 100$ and a parameter r, which guarantees that the matrices A_k, $k = 1, \ldots, n$ and $K_{i,j}$, $i, j = 1, \ldots, d$ have very few (typically 4) non-zero entries. Since d and the number of non-zero entries is independent of n, the matrix $\mathcal{B}(x)$ gets sparser and sparser with growing n. This group of test cases is used to demonstrate the ability of the code to solve large scale examples by exploiting sparsity in the data.

Table 11.18: Random Problems – Category I; numerical results

#Vars	Matsize	#Nwtsteps	#Iterations	Time
		min/avg/max	min/avg/max	min/avg/max
300	50	79/110/240	10/11/11	6/8/18
600	100	67/96/301	11/11/12	16/25/72
1200	200	104/113/126	18/20/23	420/463/539
2400	400	97/159/238	20/23/24	1157/1973/3011

Table 11.18 and Table 11.19 show results for problems of category I and category II, respectively. Along with problem dimensions we show number of iterations, number of Newton steps and CPU time. For each problem size we give worst, average and best values. Note that for all but one problems we were able to reach 6 digits of precision in all KKT criteria. When comparing the two tables, we can see that the problems of category I seem to be more difficult to solve.

Table 11.19: Random Problems – Category I; numerical results

#Vars (= Matsize)	avg. Density NLMI	#Nwtsteps min/avg/max	#Iterations min/avg/max	Time min/avg/max
250	0.26	61/82/265	13/15/19	4/7/15
500	0.15	68/90/134	14/17/19	33/40/57
1000	0.040	66/96/119	15/19/20	225/327/396
2000	0.027	72/100/285	16/18/21	249/372/1307
4000	0.024	76/93/103	17/21/23	1730/2150/2370
8000	0.021	83/94/105	18/20/22	17645/19638/22057

11.5 A Benchmark with Linear SDP Solvers

The linear SDP version of PENNON , called PENSDP is designed for the solution of the problem

$$\begin{gathered}\min_{x\in\mathbb{R}^n} b^T x\\ \text{subject to}\\ \mathcal{A}(x) \preccurlyeq 0\end{gathered} \tag{11.2}$$

where $\mathcal{A}(x) = A_0 + \sum_{i=1}^n x_i A_i$ and $A_i \in \mathbb{S}^m$ for all $i = 0, \dots, n$. The code was tested using two different sets of problems: the SDPLIB collection of linear SDPs by Borchers [19] and a set of various large-scale problems collected by Hans Mittelmann and called here HM-collection.

We describe the results of our testing of PENSDP and four other general purpose (linear) SDP codes available on the NEOS server, namely CSDP-4.9 by Borchers [18], SDPA-6.00 by Fujisawa *et all* [36], SDPT3-3.1 by Toh, Todd and Tütüncü [81], and DSDP-5.6 by Benson and Ye [10]. We have chosen these codes as they were, at the moment of writing this thesis, the fastest ones in the independent tests performed by Mittelmann [62]. All codes were used with standard setting; CSDP and PENSDP were linked with ATLAS-BLAS library, SDPT3 (with HKM direction) ran under MATLAB . All tests presented in this section where performed on a Pentium IV PC (3.2 GHz) with 2 GB RDRAM running Linux-2.4.19 and MATLAB 6.5 .

We will present the results of our numerical experiments in the following way: For each test set we will offer two tables. In the first table we will list the CPU time and the average DIMACS error: (compare Section 9.3.2) for each test case and each code. In a second table we will present accumulated CPU times and DIMACS errors (sum of CPU times, arithmetic mean of DIMACS errors) for certain subsets of each test case collection. We will further use the notation `-testcase1,-testcase2`, where the names `testcase*` stand for the names of particular problems from the test set, to express that results for all problems, but the ones listed with minus signs are accumulated in the corresponding line of the table. Furthermore we will use the abbreviations

- `fail`, if some code failed to converge for a certain test case,
- `m`, if some code ran out of memory for a certain test case and
- `n.a.`, if some accumulation operation is not applicable to the results produced by a certain code, since the code failed on at least one test case of the corresponding test set.

Note that the results presented in the following sections are partly taken from the web page `ftp://plato.la.asu.edu/pub/sdplib.txt` with a kind permission of the author.

11.5.1 SDPLIB

SDPLIB is a library of linear semidefinite test cases collected from various applications. The library is maintained and provided by Brian Borchers. The full library can be downloaded from the internet. Instead of presenting here the full SDPLIB results we selected just several representative problems. Table 11.20 lists these problems, along with their dimensions; the results for CSDP, DSDP, SDPA, SDPT3, and PENSDP are presented in Table 11.21

Table 11.20: Selected SDPLIB problems.

problem	no. of var.	size of matrix
arch8	174	335
control7	136	45
control10	1326	150
control11	1596	165
equalG11	801	801
equalG51	1001	1001
gpp250-4	250	250
gpp500-4	501	500
hinf15	91	37
maxG11	800	800
maxG32	2000	2000
maxG51	1001	1001
mcp250-1	250	250
mcp500-1	500	500
qap9	748	82
qap10	1021	101
qpG11	800	1600
qpG51	1000	2000
ss30	132	426
theta3	1106	150
theta4	1949	200

Selected SDPLIB problems. (*cont.*)

problem	no. of var.	size of matrix
theta5	3028	250
theta6	4375	300
thetaG11	2401	801
thetaG51	6910	1001
truss7	86	301
truss8	496	628
mcp500-1	500	500

We observe from Table 11.21 that PENSDP is for only very few of the SDPLIB problems the fastest code. This is, basically, due to the number of (inner) iterations used by the particular algorithms. Since the complexity of Hessian/Schur complement matrix assembling is approximately the same for most of the codes, and the data sparsity is handled in a similar way, the main time difference is given by the number of Newton steps. While, for instance, CSDP needs, on average, 15–30 steps, PENSDP needs often 2–3 times more steps (compare, for example, `control`-set or `thetaG51`). On the other hand, apart from the `control`-set the case `thetaG51` PENSDP is seems to be competitive in terms of CPU time. Moreover PENSDP is, behind CSDP, the most robust code concerning the quality of the solution in terms of the DIMACS error measures.

Table 11.21: Computational results for SDPLIB problems – CPU times and average DIMACS errors

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
arch8	3	2	3	5	13
	1,0E-09	1,7E-07	1,3E-07	2,0E-08	4,4E-08
control7	33	18	40	91	70
	9,6E-09	2,9E-06	1,7E-07	1,4E-07	3,9E-08
control10	151	84	199	104	680
	4,4e-08	6,2e-06	1,4e-06	1,8e-06	6,8e-06
control11	242	131	316	151	1041
	4,5e-08	6,4e-06	1,6e-06	2,2e-06	4,3e-08
equalG11	80	60	63	84	76
	9,7e-09	3,8e-08	7,5e-05	3,2e-07	4,5e-09
equalG51	207	112	127	157	208
	1,9e-09	1,9e-08	9,1e-06	2,4e-07	2,5e-08

Computational results for SDPLIB problems – CPU times and average DIMACS errors (*cont.*)

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
gpp250-4	6	2	2	7	3
	8,0e-09	8,0e-08	9,2e-06	3,0e-07	2,1e-08
gpp500-4	34	10	18	27	16
	6,8e-09	5,0e-07	2,1e-05	2,9e-09	1,8e-07
maxG11	42	10	88	33	19
	6,1e-10	1,6e-08	4,1e-09	1,7e-09	1,4e-08
maxG32	391	119	1378	428	195
	1,7e-10	2,4e-08	3,3e-09	1,2e-09	5,4e-09
maxG51	105	34	83	70	92
	7,7e-10	3,9e-08	7,0e-09	1,3e-10	2,7e-09
mcp250-1	3	1	2	3	1
	3,3e-10	3,1e-08	1,0e-08	3,4e-10	2,1e-07
mcp500-1	11	4	13	11	5
	3,3e-07	3,6e-08	2,6e-09	1,9e-10	3,8e-08
qap9	2	8	2	3	3
	3,3e-07	1,2e-07	1,3e-04	7,1e-05	2,2e-07
qap10	4	15	5	5	7
	6,9e-07	1,6e-06	5,3e-05	9,8e-05	6,1e-07
qpG11	314	50	332	214	73
	2,4e-09	6,9e-08	5,7e-09	1,1e-10	1,9e-08
qpG51	459	204	693	428	186
	9,4e-10	1,1e-07	6,2e-09	1,7e-09	3,1e-08
ss30	12	7	33	11	17
	4,2e-10	1,3e-07	9,7e-08	1,8e-07	1,6e-08
theta3	4	6	5	6	7
	1,3e-10	4,7e-09	2,6e-09	4,5e-10	6,2e-08
theta4	18	25	22	19	25
	7,1e-10	1,8e-09	3,7e-09	4,7e-10	8,3e-08
theta5	57	73	68	51	40
	1,6e-10	1,0e-08	4,0e-09	6,7e-10	3,5e-08
theta6	152	147	180	136	151

Computational results for SDPLIB problems – CPU times and average DIMACS errors (*cont.*)

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
	2,0e-09	1,5e-08	4,1e-09	1,0e-09	5,3e-08
thetaG11	100	232	129	95	217
	2,0e-09	1,5e-08	4,1e-09	1,0e-09	5,3e-08
thetaG51	1614	2553	1455	1681	6131
	6,4e-10	3,1e-07	1,6e-07	1,9e-07	5,6e-07
truss7	1	1	1	2	1
	8,1e-09	7,2e-08	2,4e-07	7,2e-08	8,9e-07
truss8	3	9	17	8	10
	3,1e-10	1,6e-08	1,3e-07	6,2e-10	5,2e-07

To see a kind of average behavior, in Tab. 11.22 we show the sum of CPU times and the average DIMACS errors for two subsets of problems, as well as for all problems from Tab. 11.20.

Table 11.22: Sum of CPU times and average DIMACS error for certain subsets of SDPLIB problems and for all problems from Tab. 11.21.

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
ALL	4048	3917	5274	3830	9287
	5,7E-08	7,3E-07	1,2E-05	6,7E-06	4,2E-07
-thetaG51	2434	1364	3819	2149	3156
	6,2E-08	7,7E-07	1,3E-05	7,3E-06	4,3E-07
-control*,	2041	1149	3304	1894	1435
-thetaG51	6,3E-08	1,4E-07	1,4E-05	7,7E-06	1,6E-07

11.5.2 HM Collection

Table 11.23 lists a selection of large-scale problems from the HM collection, together with their dimensions and number of non-zeros in the data matrices $A_i,\ i = 1,\ldots,n$.

Table 11.23: HM collection.

problem	no. of var.	size of matrix	no. of nzs.
buck-3	544	1,185	7,831
buck-4	1,200	2,545	17,509
buck-5	3,280	6,801	48,385
mater-3	1,439	3,587	45,189
mater-4	4,807	12,497	157,779
mater-5	10,143	26,819	338,757
mater-6	20,463	56,310	690,149
shmup3	420	2,641	21,600
shmup4	800	4,961	36,477
shmup5	1,800	11,041	82,317
trto-3	544	865	3,902
trto-4	1,200	1,873	8,734
trto-5	3,280	5,041	24,158
vibra-3	544	1,185	7,831
vibra-4	1,200	2,545	17,509
vibra-5	3,280	6,801	48,385
G40mb	2,000	2,000	2,003,000
G40mc	2,000	2,000	2,000
G48mc	3,000	3,000	3,000
G55mc	5,000	5,000	5,000
G59mc	5,000	5,000	5,000
butcher	22,843	6,434	206,992
cnhil8	1,716	120	7,260
cnhil10	5,005	220	24,310
cphil10	5,005	220	24,310
cphil12	12,376	363	66,429
neu1	3,003	254	31,880
neu1g	3,002	252	31,877
neu2	3,003	254	31,880
neu2g	3,002	252	31,877
neu2c	3,002	1,255	158,098
neu3	7,364	420	87,573
neu3g	8,007	462	106,952
rabmo	5,004	6,826	60,287
reimer5	6,187	102,606	719,806
rose13	2,379	105	5,564
rose15	3,860	137	9,182
taha1a	3,002	1,680	177,420

HM collection problems. (*cont.*)

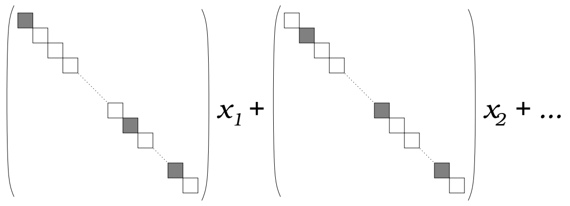
problem	no. of var.	size of matrix	no. of nzs.
taha1b	8,007	1,609	107,373
cancer	10,470	570	10,569
checker	3,971	3,970	3,970
foot	2,209	2,208	2,440,944
hand	1,297	1,296	841,752
ice_2.0	8,113	8,113	8,113
p_auss2	9.115	9,115	9,115
BH2.r14	1,743	2,166	142,919
CH2_1.r14	1,743	2,166	142,919
CH2_3.r14	1,743	2,166	142,919
H2O+.r14	1,743	2,166	142,919
H20_.r14	1,743	2,166	142,919
NH2.r14	1,743	2,166	142,919
H30_.r16	2,964	3,162	279,048
NH3.r16	2,964	3,162	279,048
inc_600	3,114	2,515	190,356
inc_1200	6,374	5,175	741,296
neosfbr20	362	7,201	309,624
r1_600_0	600	601	180,900
yalsdp	300	5,051	1,005,250

The HM collection consists of test cases arising from various areas of applications. As a consequence, the structures of the corresponding SDP problems are substantially different. In order to make the results of our experiments more transparent to the reader we decided to split the collection in certain subgroups, which are listed below:

1. STRUCTOPT
2. GRAPH
3. SOSTOOLS & GLOPTYPOLY
4. IMAGE
5. CHEMICAL
6. MIXED

In Table 11.23 these subgroups are separated by horizontal lines and show up in the same order as in the list above. In the sequel we will give brief descriptions for each of the test groups and present the results of our computational experiments.

The STRUCTOPT set The STRUCTOPT set itself consists of three classes of problems. The first class (`mater`) consists of multiple-load case free material optimization problems as formulated in (10.5). All examples solve the same problem (geometry, loads, boundary conditions) and differ only in the finite element discretization. The linear matrix operator $\mathcal{A}(x) = \sum A_i x_i$ has the following structure: A_i are block diagonal matrices with many ($\sim$5 000) small (11×11–20×20) blocks. Moreover, only few (6–12) of these blocks are non-zero in any A_i, as schematically shown in the figure below.



As a result, the Hessian of the augmented Lagrangian associated with this problem is a large and sparse matrix. The second class (`smhup`) consists of free material optimization problems subjected to vibration constraints (compare problem (10.18)). Again all examples differ only in the finite element discretization. The third class includes problems from truss topology design:

- `trto` are problems from single-load truss topology design. Normally formulated as LPs, here reformulated as SDPs for testing purposes.
- `vibra` are single load truss topology problems with a vibration constraint. The constraint guarantees that the minimal self-vibration frequency of the optimal structure is bigger than a given value.
- `buck` are single load truss topology problems with linearized global buckling constraint. Originally a nonlinear matrix inequality, the constraint should guarantee that the optimal structure is mechanically stable (does not buckle).

The problems `smhup`, `trto`, `vibra` and `buck` are characterized by sparsity of the linear matrix operator $\mathcal{A}$.

Table 11.24: Computational results for STRUCTOPT problems – CPU times and average DIMACS errors

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
buck-3	89	32	24	30	31
	3,1e-07	7,9e-06	5,0e-06	3,0e-05	2,9e-07

Computational results for STRUCTOPT problems – CPU times and average DIMACS errors (*cont.*)

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
buck-4	536	223	248	130	219
	1,0e-07	2,0e-06	4,6e-07	6,1e-06	4,4e-08
buck-5	5005	3964	4960	1350	2173
	1,8e-06	5,4e-06	3,0e-06	4,0e-05	1,9e-07
mater-3	12	16	1044	26	5
	1,3e-09	3,5e-08	7,9e-09	5,7e-10	7,9e-08
mater-4	290	102	138452	129	27
	3,0e-09	4,1e-07	4,3e-09	3,2e-10	1,5e-07
mater-5	3647	412	m	291	74
	8,2e-10	2,1e-07	m	4,0e-10	5,8e-07
mater-6	m	1765	m	919	205
	m	4,9e-07	m	9,7e-10	2,7e-07
shmup3	1498	232	419	247	233
	1,4e-09	8,8e-07	2,4e-06	6,1e-07	3,1e-07
shmup4	3774	1643	1988	1148	1283
	7,6e-08	1,0e-06	1,5e-06	5,3e-07	2,6e-07
shmup5	52295	16812	m	8740	9154
	1,5e-06	4,7e-06	m	2,1e-06	1,6e-06
trto-3	19	9	12	15	17
	2,3e-08	1,5e-05	9,2e-07	8,2e-05	1,0e-06
trto-4	238	86	125	64	81
	1,1e-06	2,7e-05	3,8e-06	6,2e-04	7,1e-06
trto-5	2671	1194	1963	694	914
	,6e-06	1,2e-04	3,9e-05	1,5e-04	3,5e-06
vibra-3	69	36	28	32	30
	3,0e-07	5,7e-06	9,5e-06	4,3e-05	1,7e-07
vibra-4	774	303	269	150	169
	2,5e-07	2,0e-05	1,5e-06	5,5e-05	5,6e-07
vibra-5	5984	4396	4740	2269	2050
	2,2e-06	9,5e-05	4,9e-06	5,7e-04	5,6e-07

As one can see from Table 11.24, PENSDP is particularly efficient for the group of `mater` problems. The following table with accumulated results shows that PENSDP is not only the fastest, but also the most robust code for the STRUCTOPT set.

Table 11.25: Sum of CPU times and average DIMACS error for certain subsets of STRUCTOPT problems and for all problems from Tab. 11.24.

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
ALL	n. a.	31225	n. a.	16234	16665
	n. a.	1,9E-05	5,8E-06	1,0E-04	1,0E-06
-mater6	76901	29460	n. a.	15315	16460
	8,2E-07	2,0E-05	n. a.	1,1E-04	1,1E-06
-shmup5,	20657	12118	14776	6129	7200
-mater*	9,8E-07	2,7E-05	6,5E-06	1,5E-04	1,3E-06

The GRAPH set The GRAPH set includes problems arising from SDP relaxations of max-cut problems collected in the graph library Gset, developed and maintained by Yinyu Ye. The interested reader is referred to

`http://www.stanford.edu/ yyye/yyye/Gset/`

for further information. The test cases in the GRAPH set are characterized by very sparse data matrices A_i, $i = 0, \ldots, n$, the equation $n = m + 1$, where n and m are the dimensions in problem (11.2), and in all but one case a sparse matrix operator $\mathcal{A}$.

Table 11.26: Computational results for GRAPH problems – CPU times and average DIMACS errors

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
G40mb	1896	451	1074	937	1070
	2,5E-09	1,4E-05	9,5E-05	2,1E-06	2,3E-08
G40mc	532	285	686	413	680
	1,1E-09	3,7E-08	2,0E-09	1,4E-10	3,9E-08
G48mc	1137	312	2077	580	479
	1,4E-08	1,3E-07	1,4E-08	1,3E-09	3,9E-08
G55mc	7199	2721	m	m	6862
	1,3E-09	1,7E-07	m	m	4,9E-08
G59mc	9211	3985	m	m	11902
	3,4E-09	1,1E-07	m	m	1,8E-08

Tables 11.26 and 11.27 show that only three codes where able to solve all cases. Among them DSDP is the fastest code. On the other hand the results calculated by PENSDP and CSDP are more precise. On the smaller set of problems, solved by all codes, PENSDP is one of the faster codes behind DSDP and SDPT3.

Table 11.27: Sum of CPU times and average DIMACS error for certain subsets of GRAPH problems and for all problems from Tab. 11.26.

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
ALL	19975	7754	n. a.	n. a.	20993
	4,5E-09	2,9E-06	n. a.	n. a.	3,4E-08
-G5*	3565	1048	3837	1930	2229
	5,9E-09	4,7E-06	3,2E-05	7,0E-07	3,4E-08

The SOSTOOLS & GLOPTYPOLY set All test cases in this set are created by one of the tools

- **GloptiPoly** – a Matlab add-on to build and solve convex linear matrix inequality (LMI) relaxations of the (generally non-convex) global optimization problem of minimizing a multivariable polynomial function subject to polynomial inequality, equality or integer constraints. More information, references, downloads etc. can by found at

  `http://www.laas.fr/ henrion/software/gloptipoly/`.

- **Sostools** – a sum of squares optimization toolbox for MATLAB. Detailed information is provided at

  `http://www.cds.caltech.edu/sostools/`.

Most problems are characterized by the inequality $n \gg m$, moderately sparse data matrices A_i, $i = 0, \ldots, n$ and a dense matrix operator $\mathcal{A}$. For an exact description of the particular test cases we refer to

`ftp://plato.asu.edu/pub/sdp/README`.

Table 11.28: Computational results for SOSTOOLS & GLOPTYPOLY problems – CPU times and average DIMACS errors

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
butcher	4032	3223	23864	1475	8600

Computational results for SOSTOOLS & GLOPTYPOLY problems – CPU times and average DIMACS errors (*cont.*)

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
	6,5E-09	4,0E-03	6,5E-02	1,6E-02	4,2E-07
cnhil8	22	25	31	7	91
	6,6E-09	3,8E-06	1,4E-04	2,0E-06	7,4E-08
cnhil10	398	387	505	54	1375
	8,3E-09	3,9E-06	1,1E-04	3,0E-05	7,8E-08
cphil10	93	405	337	31	22
	1,1E-10	3,6E-07	1,4E-08	2,0E-09	3,3E-07
cphil12	1078	4511	3825	m	130
	1,2E-10	4,9E-07	2,6E-08	m	2,4E-08
neu1	581	632	341	fail	1090
	2,1E-04	1,6E-02	2,9E-02	fail	1,2E-06
neu1g	835	684	279	290	708
	4,8E-08	6,5E-04	1,1E-04	2,2E-04	9,9E-07
neu2	852	fail	fail	fail	1004
	1,4E-02	fail	fail	fail	9,3E-08
neu2g	516	652	fail	601	1565
	3,4E-05	7,0E-04	fail	8,8E-04	1,8E-06
neu2c	2531	1335	1082	1532	3484
	1,8E-08	2,5E-04	5,1E-04	1,3E-05	4,7E-04
neu3	5807	15922	13938	m	8634
	1,8E-04	1,9E-03	4,8E-07	m	6,0E-04
neu3g	10366	20236	7070	fail	8147
	3,6E-09	1,5E-03	5,6E-07	fail	4,7E-04
rabmo	615	fail	7199	365	2305
	1,8E-09	fail	6,0E-06	2,0E-04	9,7E-08
reimer5	14436	fail	m	3373	17502
	3,8E-09	fail	m	2,7E-06	6,4E-06
rose13	104	192	72	85	283
	2,2E-08	4,9E-03	1,6E-06	5,1E-08	5,0E-08
rose15	634	fail	293	271	1028
	6,4E-04	fail	9,7E-05	4,4E-04	2,0E-06

Computational results for SOSTOOLS & GLOPTYPOLY problems – CPU times and average DIMACS errors (*cont.*)

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
taha1a	1440	fail	fail	832	2334
	1,8E-09	fail	fail	2,2E-02	1,2E-07
taha1b	3297	4782	10886	2849	9440
	1,4E-09	6,9E-04	1,1E-08	1,5E-09	6,2E-08

Table 11.28 shows that many of the test cases in this class are very hard to solve. This is mainly due to ill conditioning of the linear systems that have to be solved by the codes. In fact only two codes, namely CSDP and PENSDP where able to solve all instances of this class – but also for these two codes the average precision achieved is considerably low. In Table 11.29 show again average results for certain subsets of problems. In particular we summarized the computational results for the largest subset of test cases, which could be successfully solved by each of the codes CSDP, DSDP, SDPA and SDPT3. The names of the subsets in Table 11.29 below are constructed from the name of the corresponding codes. Moreover we separated the results for the group `neu*` from the other cases, since this group seems to be particularly difficult to solve. It is interesting to observe that the average precision of PENSDP on the resulting (reduced) test set is almost seven digits.

Table 11.29: Sum of CPU times and average DIMACS error for certain subsets of SOSTOOLS & GLOPTYPOLY problems and for all problems from Tab. 11.28.

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
ALL	47637	n. a.	n. a.	n. a.	67742
	8,4E-04	n. a.	n. a.	n. a.	8,6E-05
-neu*	26149	n. a.	n. a.	n. a.	43110
	5,8E-05	n. a.	n. a.	n. a.	8,8E-07
neu*	21488	n. a.	n. a.	n. a.	24632
	2,1E-03	n. a.	n. a.	n. a.	2,2E-04
DSDP-set	29660	52986	n. a.	n. a.	43569
	3,3E-05	2,4E-03	n. a.	n. a.	1,2E-04
SDPA-set	30393	n. a.	69722	n. a.	45337
	7,4E-05	n. a.	6,8E-03	n. a.	1,1E-04
SDPT3-set	28953	n. a.	n. a.	11765	48737
	5,2E-05	n. a.	n. a.	3,1E-03	3,7E-05

The IMAGE set The IMAGE set consists of test cases arising from binary image partitioning, perceptual grouping and restoration. The interested reader is referred to [50], [49] and the references at `ftp://plato.asu.edu/pub/sdp/README`. The data matrices A_i, $i = 0, \ldots, n$ of the cases in this class are extremely sparse. Moreover for all but one cases we have $n = m$ and a sparse matrix operator $\mathcal{A}$. In the exceptional case `cancer`, the operator $\mathcal{A}$ is dense.

Table 11.30: Computational results for IMAGE problems – CPU times and average DIMACS errors

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
cancer	2539	1568	1686	m	7261
	6,9E-08	8,8E-02	3,8E-05	m	2,5E-07
checker	9915	1940	m	2371	1408
	3,2E-09	1,1E-01	m	2,9E-10	2,3E-08
foot	4019	622	2156	1758	1870
	2,3E-05	4,9E-07	8,9E-06	2,4E-05	1,7E-08
hand	452	138	253	271	313
	1,5E-08	9,4E-07	9,6E-06	8,9E-06	2,5E-07
ice_2.0	m	14979	m	m	13122
	m	7,0E-08	m	m	1,6E-08
p_auss2	m	15758	m	m	14110
	m	1,4E-07	m	m	6,2E-09

Only DSDP and PENSDP where able to solve all cases of the IMAGE set. All other codes ran out of memory in at least two cases. Table 11.31 shows that (apart from `cancer`) PENSDP is one of the fastest codes for this group of test cases. One should mention that for the test case `cancer` a low precision result, which is still more precise as the DSDP result can be achieved by PENSDP in about 1000 seconds.

Table 11.31: Sum of CPU times and average DIMACS error for certain subsets of IMAGE problems and for all problems from Tab. 11.24.

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
ALL	n. a.	35005	n. a.	n. a.	38084
	n. a.	3,3E-02	n. a.	n. a.	9,4E-08
CSDP set	16925	4268	n. a.	n. a.	10852
	5,8E-06	5,0E-02	n. a.	n. a.	1,4E-07

Sum of CPU times and average DIMACS error for certain subsets of IMAGE problems and for all problems from Tab. 11.30 (*cont.*)

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
SDPA set	7010	2328	4095	n. a.	9444
	7,7E-06	2,9E-02	1,9E-05	n. a.	1,7E-07
SDPT3 set	14386	2700	n. a.	4400	3591
	7,7E-06	3,7E-02	n. a.	1,1E-05	9,7E-08

From Table 11.32 we see that all codes were able to solve all test cases successfully. Taking into consideration both, the CPU time and the precision of the solutions, CSDP seems to be the best code for this group of test cases. Faster results were produced by the codes SDPT3 and SDPA. PENSDP is comparably fast as CSDP, but less precise.

The CHEMICAL set The test cases in the set CHEMICAL are extracted from a larger set of test cases. The full library collects SDP benchmark problems from electronic structure calculations and can be downloaded from

`www://cims.nyu.edu/m̃ithuro/software.html`.

All test cases are characterized by the inequality $m \geq n$, moderately sparse data matrices $A_i, i = 1, \ldots, n$ and a dense matrix operator $\mathcal{A}$.

Table 11.32: Computational results for CHEMICAL problems – CPU times and average DIMACS errors

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
BH2.r14	1995	4770	1495	729	2445
	8,3E-10	4,5E-08	3,7E-04	3,7E-10	9,6E-08
C. _1.r14	1861	4684	1393	1633	2593
	1,0E-09	4,9E-08	2,2E-03	1,3E-02	5,6E-08
C._3.r14	1958	4362	1592	717	2206
	8,7E-10	6,1E-08	1,6E-03	5,9E-10	4,6E-08
H2O+.r14	1807	4445	1591	1690	2261
	1,1E-09	6,1E-08	6,1E-04	4,8E-03	6,4E-08
H20_.r14	1599	4378	1492	1583	1980
	1,0E-09	6,0E-08	5,9E-04	1,3E-09	7,8E-08
NH2.r14	1747	4593	1496	1514	2184
	1,4E-09	3,7E-08	9,8E-04	2,3E-08	7,1E-08

Computational results for CHEMICAL problems – CPU times and average DIMACS errors (*cont.*)

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
H30_.r16	16087	23123	15189	14154	16664
	9,2E-10	2,8E-08	4,3E-08	3,5E-10	4,7E-08
NH3.r16	16917	22120	15190	5268	15962
	3,9E-10	3,7E-08	4,0E-08	2,6E-03	1,8E-07

From Table 11.33 we see that SDPT3 was the fastest code for this group of test cases. On the other hand the average precision of the results produced by SDPT3 is much lower as for the codes CSDP, DSDP and PENSDP . Taking into account CPU time and precision, one could come to the conclusion that CSDP is the best code for this group of examples followed by PENSDP .

Table 11.33: Sum of CPU times and average DIMACS error for certain subsets of CHEMICAL problems and for all problems from Tab. 11.32.

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
ALL	43971	72475	39438	27288	46295
	9,4E-10	4,7E-08	7,9E-04	2,6E-03	8,0E-08

The MIXED set We collected all remaining test cases from the HM collection in a test set called MIXED. Again we refer to

`ftp://plato.asu.edu/pub/sdp/README`.

for detailed information on the particular test cases. The structure of the problems can be seen from Table 11.23.

Table 11.34: Computational results for MIXED problems – CPU times and average DIMACS errors

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
inc_600	549	fail	604	fail	1377
	4,4E-04	fail	6,7E-03	fail	2,5E-05
inc_1200	2937	fail	fail	fail	7550
	6,4E-04	fail	fail	fail	5,2E-04
neosfbr20	1650	1914	1663	1941	2045

Computational results for MIXED problems – CPU times and average DIMACS errors (*cont.*)

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
	1,9E-09	2,8E-09	7,1E-09	1,5E-10	6,4E-08
r1_6_0	63	18	30	49	36
	1,2E-08	1,3E-07	3,1E-05	9,9E-07	1,5E-08
yalsdp	1154	1311	1108	737	739
	3,8E-10	5,2E-07	7,3E-09	2,8E-08	3,9E-09

Tables 11.34 and 11.35 show the results of our experiments. While for the `inc*`-cases only CSDP and PENSDP were able to find approximate solutions, all codes behaved similar on the other cases contained in the MIXED set. This is the reason, why we give summarized results for all cases on the one hand and for all but the `inc*`-cases on the other hand in Table 11.35.

Table 11.35: Sum of CPU times and average DIMACS error for certain subsets of MIXED problems and for all problems from Tab. 11.34.

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
ALL	6353	n. a.	n. a.	n. a.	11747
	2,2E-04	n. a.	n. a.	n. a.	1,1E-04
-inc*	2867	3243	2801	2727	2820
	1,4E-08	6,5E-07	3,1E-05	1,0E-06	8,3E-08

At the end of this section, we want to determine an average behavior of all codes we have used in our experiments for the full HM collection. As we have seen throughout this section almost all codes failed for certain subsets of test cases. Consequently we decided to compare PENSDP to each code separately. The comparison is done on the maximal subset of problems, which could be successfully solved by CSDP, DSDP, SDPA and SDPT3, respectively. The corresponding subsets are named CSDP set, DSDP set, SDPA set and SDPT3 set. Moreover we present accumulated results for certain sub-groups of these sets. This is in order to avoid that the overall impression is completely dominated by very few disadvantageous results.

Table 11.36: Sum of CPU times and average DIMACS error for certain subsets of HM collection problems and for all problems from Tab. 11.23.

problem	CSDP	DSDP	SDPA	SDPT3	PENSDP
CSDP set	211762				174089
	2.9E-04				3.8E-05
-mater,	155518				164829
-smhup5	3.2E-04				4.1E-05
-neu*, -inc*,	186154				139502
-rose15*	7.9E-07				5.7E-07
DSDP set		202688			168426
		4.5E-03			3.1E-05
-SOS,-cancer,		146194			116188
-checker		9.5E-06			5.0E-07
SDPA set			135273		114702
			2.4E-03		3.5E-05
-SOS,-inc*,			130574		103881
			2.9E-04		1.6E-06
SDPT3 set				64344	120337
				6.7E-04	5.4E-07
-SOS,-inc*				47311	55638
				5.5E-05	5.8E-07

We want to conclude this section by a few remarks:

Remark .

- From Table 11.36 we can see that PENSDP is a reliable code for a wide class of linear SDP problems. Moreover PENSDP is on the average one of the best codes in terms of CPU time and the quality of the computed approximate solution with respect to the DIMACS criteria.
- It should be mentioned that the code CSDP achieved (sometimes by far) the best precision for many of the test cases of the HM collection.
- We considered a test run of a code to be failed, whenever the (average) DIMACS precision achieved was below 1.0E-01. Note that a different choice could significantly influence the avaraged information presented in the tables above.
- We did not use all test collected by H. Mittelmann for our experiments. A few test cases have been excluded for different reasons.

Chapter 12

Outlook

In the final chapter of this thesis we would like to give a short outlook. The following topics may be of interest in our future research:

- In Chapter 3 we have imposed the nondegeneracy constraint qualification assumption. Unfortunately this condition can be very strong, because it forces $n \geq r(r+1)/2$, where $r = \dim(\mathrm{Ker}\mathcal{A}(x^*))$. Of course, we can assume the nondegenaracy condition to hold for each matrix block separately, if the matrix constraint is block structured. Nevertheless from the inequality above follows that the nondegeneracy condition cannot hold in many of the large scale test cases we have presented in Section 11. On the other hand, the algorithm converged to an optimal solution in many of those cases. Therefore it would be interesting to find a convergence proof for the method proposed in this thesis, which does not require the nondegeneracy constraint qualification.

- The second point addresses a problem we have already mentioned briefly in Section 6. There is still a gap between the "theoretical" Algorithm 6.1.1, where we assume that each subproblem is solved exactly, and Algorithm 9.4.1, where we work with approximate solutions of the subproblems. R. Polyak has closed this gap in the nonlinear programming case; see [69] for further details. Of course it would be interesting to establish a similar result in the context of our method.

- A few years ago R. Polyak has invented the primal-dual nonlinear rescaling method for the solution of nonlinear programming problems of the form (NLP) (see [70]). The method is in certain respect based on the the modified barrier function method. Just recently, Polyak has shown asymptotic quadratic convergence for this method. A similar approach would be also interesting in the context of our method mainly for two reasons: the first reason is of course the faster convergence in the local neighborhood of the optimum; the second advantage could be a smaller number of inner iterations due to more frequent multiplier updates.

- Another possible extension concerns the effective handling of equality constraints. There are several interesting applications, where linear or nonlinear matrix constraints occur in combination with nonlinear equalities. So far, we handle such a situation by replacing the equality constraints by two inequalities. Sometimes the results produced by this approach are rather unsatisfactory. In order to improve the situation one could think of a “direct” handling of the equality constraints as it is usually done in primal-dual interior point approaches.

- A more practical issue is the implementation of an interface for optimization problems subjected to polynomial matrix inequalities of higher order. We have already started to realize such an approach in cooperation with J. Löfberg, the developer of YALMIP 3.0 (see [59]).

Bibliography

[1] J. Ackermann. *Robust control: systems with uncertain physical parameters*. Springer Verlag, 1993.

[2] P. Apkarian and H. D. Tuan. Concave programming in control theory. *Journal of Global Optimization*, 15:343–370, 1999.

[3] B. R. Barmish. *New tools for robustness of linear systems*. Macmillan, 1994.

[4] A. Ben-Tal, F. Jarre, M. Kočvara, A. Nemirovski, and J. Zowe. Optimal design of trusses under a nonconvex global buckling constraint. *Optimization and Engineering*, 1:189–213, 2000.

[5] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization*. MPS-SIAM Series on Optimization. SIAM Philadelphia, 2001.

[6] A. Ben-Tal, I. Yuzefovich, and M. Zibulevsky. Penalty/barrier multiplier method for minimax and smooth convex problems. Technical report, Optimization Laboratory, Faculty of Industrial Engineering and Management, Technion, 1992.

[7] A. Ben-Tal and M. Zibulevsky. Penalty/barrier multiplier methods for convex programming problems. *SIAM Journal on Optimization*, 7:347–366, 1997.

[8] M. Bendsøe and O. Sigmund. *Topology Optimization. Theory, Methods and Applications*. Springer-Verlag, Heidelberg, 2002.

[9] M.P. Bendsøe and O. Sigmund. *Topology Optimization. Theory, Methods and Applications*. Springer-Verlag, 2002.

[10] S. J. Benson and Y. Ye. DSDP4 users manual. Report ANL/MCS-TM-248, Argonne National Laboratory, Argonne, 2002. Available at `http://www-unix.mcs.anl.gov/~benson/`.

[11] E. Beran, L. Vandenberghe, and S. Boyd. A global BMI algorithm based on the generalized benders decomposition. In *Proceedings of the European Control Conference, Brussels, Belgium*, 1997.

[12] D. P. Bertsekas. *Constrained Optimization And Lagrange Multiplier Methods*. Athena Scientific, Belmont, Massachusetts, 1996.

[13] S. Bhattacharyya, L. H. Keel, and S. P. Bhattacharyya. Robust stabilizer synthesis for interval plants using H_∞ methods. In *Proceedings of the IEEE Conference on Decision and Control, San Antonio, Texas*, pages 3003–3008, 1993.

[14] S. P. Bhattacharyya, H. Chapellat, and L. H. Keel. *Robust control: the parametric approach*. Prentice Hall, 1995.

[15] V. D. Blondel. *Simultaneous stabilization of linear systems*. MacMillan, New York, 1994.

[16] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

[17] F. J. Bonnans and A. Shapiro. *Perturbation Analysis of Optimization Probelms*. Springer-Verlag New-York, 2000.

[18] B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 11:613–623, 1999. Available at `http://www.nmt.edu/~borchers/`.

[19] B. Borchers. SDPLIB 1.2, a library of semidefinite programming test problems. *Optimization Methods and Software*, 11 & 12:683–690, 1999. Available at `http://www.nmt.edu/~borchers/`.

[20] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, Philadelphia, PA, 1994.

[21] M. Breitfeld and D. Shanno. A globally convergent penalty-barrier algorithm for nonlinear programming and its computational performance. Technical report, Rutcor Research Report 12-94, Rutgers University, New Jersey., 1994.

[22] M. Breitfeld and D. Shanno. Experience with a modified log-barrier method for nonlinear programming. *Annals of operations Research*, 62:439–464, 1996.

[23] Y. Y. Cao and Y. X. Sun. Static output feedback simultaneous stabilization: ILMI approach. *International journal of control*, 70(5):803–814, 1998.

[24] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam, New York, Oxford, 1978.

[25] A. R. Conn, N. I. M. Gould, and P. L. Toint. A globally convergent augmented lagrangian algorithm for optimization with general inequality constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28:545–572, 1991.

[26] A. R. Conn, N. I. M. Gould, and P. L. Toint. A globally convergent lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Math. Comp.*, 66:261–288, 1997.

[27] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, 2000.

[28] R. Correa and C. H. Ramirez. A global algorithm for nonlinear semidefinite programming. *SIAM J. optim.*, 15:1791–1820, 2004.

[29] M. Doljansky and M. Teboulle. Ueber monotone matrixfunktionen. *Mathematische Zeitschrift*, 38:177–216, 1934.

[30] M. Doljansky and M. Teboulle. An interior proximal algorithm and the exponential multiplier method for semidefinite programming. *SIAM J. on Optimization*, 9:1–13, 1998.

[31] W. S. Dorn, R. E. Gomory, and H. J. Greenberg. Automatic design of optimal structures. *J. Mec.*, 3:25–52, 1964.

[32] L. El Ghaoui, F. Oustry, and M. Ait-Rami. A cone complementarity linearization algorithm for static output-feedback and related problems. *IEEE Transactions on Automatic Control*, 42:1171–1176, 1997.

[33] B. Fares, P. Apkarian, and D. Noll. An augmented lagrangian method for a class of lmi-constrained problems in robust control theory. *Internat. J. Control*, 74:348–360, 2001.

[34] B. Fares, D. Noll, and P. Apkarian. Robust control via sequential semidefinite programming. *SIAM Journal on Control and Optimization*, 40(6):1791–1820, 2002.

[35] K. Fujisawa, M. Kojima, and K. Nakata. Exploiting sparsity in primal-dual interior-point method for semidefinite programming. *Mathematical Programming*, 79:235–253, 1997.

[36] K. Fujisawa, M. Kojima, K. Nakata, and M. Yamashita. SDPA (SemiDefinite Programming Algorithm) User's Manual — Version 6.00. Available at `http://www.is.titech.ac.jp/˜yamashi9/sdpa/index.html`, Department of Mathematical and Computing Science, Tokyo University of Technology, 2002.

[37] A. George and J. W.-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, NJ, 1981.

[38] K. C. Goh, M. G. Safonov, and G. P. Papavassilopoulos. Global optimization for the biaffine matrix inequality problem. *Journal of Global Optimization*, 7:365–380, 1995.

[39] K. C. Goh, L. Turan, M. G. Safonov, G. P. Papavassilopoulos, and J. H. Ly. Biaffine matrix inequality properties and computational methods. In *Proceedings of the American Control Conference, Baltimore, MD*, 1994.

[40] L. Vanderberghe H. Wolkowicz, R. Saigal. *Handbook of Semidefinite Programming – Theory, Algorithms and Applications*. Kluwer Academic Publishers, 2000.

[41] J. W. Helton and O. Merino. Coordinate optimization for bi-convex matrix inequalities. In *Proceedings of the IEEE Conference on Decision and Control, San Diego, CA*, 1997.

[42] D. Henrion, M. Kočvara, and M. Stingl. Solving simultaneous stabilization bmi problems with pennon. LAAS-CNRS research report no. 04508, LAAS, Toulouse, 2003.

[43] D. Henrion, D. Peaucelle, D. Arzelier, and M. Šebek. Ellipsoidal approximation of the stability domain of a polynomial. In *Proceedings of the European Control Conference, Porto, Portugal*, 2001.

[44] D. Henrion, S. Tarbouriech, and M. Šebek. Rank-one LMI approach to simultaneous stabilization of linear systems. *Systems and control letters*, 38(2):79–89, 1999.

[45] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[46] F. Jarre. An interior method for nonconvex semidefinite programs. *Optimization and Engineering*, 1:347–372, 2000.

[47] C. Kanzow and C. Nagel. Some structural properties of a newton-type method for semidefinite programs. *J. of Opt. Theory and Applications*, 122:219–226, 2004.

[48] C. Kanzow and C. Nagel. Quadratic convergence of a newton-type method for semidefinite programs without strict complementarity. *SIAM Journal on Optimization*, to appear.

[49] J. Keuchel, C. Schellewald, D. Cremers, and C. Schnörr. Convex relaxations for binary image partitioning and perceptual grouping. *B. Radig, S. Florczyk (Eds.), Pattern Recognition, Lecture Notes in Computer Science*, 2191:353–360, 2001.

[50] J. Keuchel, C. Schnörr, C. Schellewald, and D. Cremers. Binary partitioning, perceptual grouping, and restoration with semidefinite programming. *IEEE Trasactions on Pattern Analysis and Machine Intelligence*, 25(11):1364–1379, 2003.

[51] M. Kočvara. On the modelling and solving of the truss design problem with global stability constraints. *Struct. Multidisc. Optimization*, 23(3):189–203, 2002.

[52] M. Kočvara and M. Stingl. Solving nonconvex sdp problems of structural optimization with stability control. *Optimization Methods and Software*, 19(5):595–609, 2004.

[53] M. Kočvara and M. Stingl. On the solution of large-scale sdp problems by the modified barrier method using iterative solvers. Preprint no. 304, Institute of Applied Mathematics, University of Erlangen-Nuremberg, 2005.

[54] V. Kučera. *Discrete linear control: the polynomial approach*. John Wiley and Sons, Chichester, UK, 1979.

[55] F. Leibfritz. *COMPl$_e$ib*: *CO*nstrained *M*atrix–optimization *P*roblem *lib*rary – a collection of test examples for nonlinear semidefinite programs, control system design and related problems. Technical report, University of Trier, Department of Mathematics, D–54286 Trier, Germany., 2003.

[56] F. Leibfritz and W. Lipinski. Description of the benchmark examples in *COMPl$_e$ib* 1.0. Technical report, University of Trier, Department of Mathematics, D–54286 Trier, Germany., 2003.

[57] F. Leibfritz and E. M. E. Mostafa. An interior point constrained trust region method for a special class of nonlinear semidefinite programming problems. *SIAM Journal on Optimization*, 12(4):1048–1074, 2002.

[58] J. W.-H. Liu. Modification of the minimum degree algorithm by multiple elimination. *ACM Transaction on Maathematical Software*, 11:141–153, 1985.

[59] J. Löfberg. YALMIP - yet another LMI parser, version 2.4. *Division of Automatic Control, Linköping University, Linköping, Sweden*, May 2003.

[60] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, Inc., Massachusetts, 1984.

[61] H. D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Math. Prog.*, 95:407–430, 2003.

[62] H. D. Mittelmann. Several SDP codes on sparse and other SDP problems. *Department of Mathematics and Statistics, Arizona State University, Tempe, AZ*, September 14, 2003.

[63] L. Mosheyev and M. Zibulevsky. Penalty/barrier multiplier algorithm for semidefinite programming. *Optimization Methods and Software*, 13:235–261, 2000.

[64] S. Nash, R. Polyak, and A. Sofer. A numerical comparison of barrier and modified method for large-scale bound-constrained optimization. *Large Scale Optimization , State of the Art, W.W. Hager, D. W. Hearn and P. M. Pardalos (Eds.), Kluwer Acad. Publ., Dordrecht*, pages 319–338, 1994.

[65] Yu. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, Philadelphia, PA, 1994.

[66] E. Ng and B. W. Peyton. Block sparse cholesky algorithms on advanced uniprocessor computers. *SIAM Journal on Scientific Computing*, 14:1034–1056, 1993.

[67] J. Petersson. On stiffness maximization of variable thickness sheet with unilateral contact. *Quart. of Applied Mathematics*, 54:541–550, 1996.

[68] R. Polyak. Modified barrier functions: Theory and methods. *Mathematical Programming*, 54:177–222, 1992.

[69] R. Polyak. Log-sigmoid multipliers methods in constrained optimization. *Annals of Operations Research*, 101:427–460, 2001.

[70] R. Polyak. Nonlinear rescaling vs. smoothing technique in convex optimization. *Mathematical Programming*, 92:197–235, 2002.

[71] R. T. Rockafellar. A dual approach to solving nonlinear programming problems by unconstrained optimization. *Mathematical Programming*, 5:354–373, 1973.

[72] A. Shapiro. First and second-order analysis of nonlinear semidefinite programs. *Mathematical Programming*, pages 301–320, 1997.

[73] A. Shapiro. On differentiability of symmetric matrix valued functions. Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA, 2002.

[74] R. Sommer. *Stabilitätsfragen beim optimalen Entwurf von Stabwerken*. PhD thesis, Friedrich-Alexander-Universität Erlangen—Institut für Angewandte Mathematik 2, 2000.

[75] M. Stingl. *Konvexe Semidefinite Programmierung*. Diplomarbeit, Institute Of Applied Mathematics, Friedrich-Alexander University of Erlangen-Nuremberg, 1999.

[76] D. Sun and J. Sun. Semismooth matrix-valued functions. *Math. Oper. Res.*, 27:150–169, 2002.

[77] C. M. Theobald. An inequality for the trace of the product of two symmetric matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, pages 77–265, 1975.

[78] S. P. Timoshenko and J. M. Gere. *Theory of Elastic Stability*. McGraw-Hill, New York, 1961.

[79] M. J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[80] P. Tseng and D. P. Bertsekas. On the convergence of the exponential multiplier method for convex programming. *Math. Programming*, 60:1–19, 1993.

[81] R.H. Tütüncü, K.C. Toh, and M.J. Todd. SDPT3 — A MATLAB software package for semidefinite-quadratic-linear programming, Version 3.0. Available at `http://www.orie.cornell.edu/~miketodd/todd.html`, School of Operations Research and Industrial Engineering, Cornell University, 2001.

[82] L. Vanderberghe and S. Boyd. Semidefinite programming. *SIAM Rev.*, 38:49–95, 1996.

[83] K. Wei. Simultaneous stabilization of single-input single-output discrete-time systems. *IEEE Transactions on Automatic Control*, 38(3):446–450, 1993.

[84] R. Werner. *Free Material Optimization*. PhD thesis, Friedrich-Alexander-Universität Erlangen—Institut für Angewandte Mathematik 2, 2000.

[85] B. Wie and D. S. Bernstein. Benchmark problems for robust control design. In *Proceedings of the American Control Conference, Chicago, Illinois*, pages 2047–2048, 1992.

[86] S. Wright. *Primal-Dual Interior-Point methods*. SIAM, 1997.

[87] M. Zibulevsky. *Penalty/barrier multiplier methods for large-scale nonlinear and semidefinite programming*. PhD thesis, Technion—Israel Institute of Technology, Haifa, 1996.

[88] J. Zowe, M. Kočvara, and M. Bendsøe. Free material optimization via mathematical programming. *Mathematical Programming, Series B*, 79:445–466, 1997.