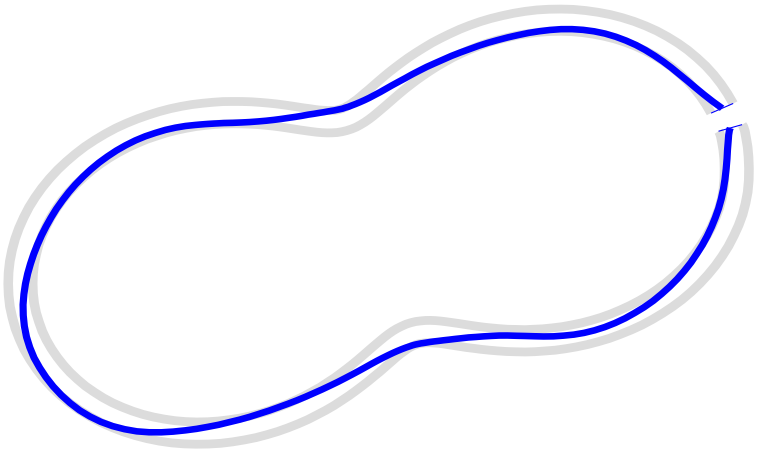
KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering

Sequential Convex Programming and Decomposition Approaches for Nonlinear Optimization

Quoc Tran Dinh



Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering Science

November 2012

Sequential Convex Programming and Decomposition Approaches for Nonlinear Optimization

Quoc Tran Dinh

Jury:
Prof. Dr. Jean Berlamont, chair
Prof. Dr. Moritz Diehl, promotor
Prof. Dr. Johan Suykens
Prof. Dr. Jan Swevers
Prof. Dr. Joos Vandewalle
Prof. Dr. Stefan Vandewalle
Prof. Dr. Ion Necoara
(Automation and Systems Engineering Department, University Politehnica of Bucharest, Romania)
Prof. Dr. Yurii Nesterov
(CORE, Université Catholique de Louvain)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering Science

November 2012

D/2012/7515/123
ISBN: 978-94-6018-589-2

Abstract

This thesis is devoted to studying numerical solution methods for some classes of nonlinear optimization problems. These methods are motivated from the fact that many optimization problems in practice possess certain structures such as convexity, separability and sparsity. Moreover, solving a convex optimization problem is more efficient and reliable than a nonconvex one by using the state-of-the-art of convex optimization algorithms. Exploiting such specific structures and convex optimization techniques can lead to more efficient and reliable solution methods than conventional approaches.

The content of the thesis is divided into two parts. Part I studies the *sequential convex programming* approach for solving nonconvex optimization problems, both parametric nonlinear programming and nonconvex semidefinite programming. A generic algorithmic framework which we call *adjoint-based predictor-corrector sequential convex programming* is proposed to treat *parametric nonconvex* optimization problems with *general convex* constraints. The algorithm is based on three ingredients, namely sequential convex programming, predictor-corrector path-following and adjoint-based optimization. The stability of the tracking errors between approximation solutions and the true ones is proved. Without parameters, the algorithm collapses to the one which we call the *sequential convex programming* (SCP) method for solving nonconvex optimization problems. As a special case of SCP, we develop a generalized *inner convex approximation* method and a generalized *convex-concave decomposition* algorithm for solving a class of nonconvex semidefinite programming problems. We also show applications of these algorithms in static state/output feedback controller design. Numerical results are benchmarked via several standard numerical examples.

Part II deals with *decomposition approaches* for separable optimization, both in the convex and nonconvex case. We develop several decomposition methods for solving separable convex optimization problems. The first class of algorithms is based on two main ingredients, namely *smoothing via prox-functions* and the *excessive gap technique*. The convergence of these algorithms is proved and the convergence rate is estimated. Extensions to the strongly convex case and inexact cases are also considered. The second class of algorithms makes use of *smoothing techniques via self-concordant barrier functions* and a *path-following method*. The algorithms developed in this part can be implemented in a parallel or distributed fashion. Several algorithmic variants are tested in numerical examples. We also show an application of these algorithms to the nonconvex case. This leads to a two-level decomposition algorithm for solving a class of separable nonconvex optimization problems.

Keywords: Sequential convex programming, decomposition method, path-following, generalized inner convex approximation, convex-concave decomposition, smoothing technique, parametric optimization, separable convex optimization, parallel and distributed algorithm.

Acknowledgements

This thesis has been completed during the years that I have spent as a PhD student at the OPTEC and SISTA research groups.

First and foremost, I wish to thank my promotor, Professor Moritz Diehl, who gave me the great opportunity to come to Leuven and guided my research. Many thanks to him for the fruitful discussions and for inspiring many of the ideas and results that are presented in this thesis. Above all, I thank him for his trust and his guidance during my PhD period in Leuven. I have also learned from him many great scientific skills which will be helpful for my future career.

I am also grateful to Dr. Michel Baes who, during his postdoc at OPTEC, kindly helped me at the early stages of my PhD. My thanks go to the committee, Professors Jan Swevers, Joos Vandewalle, Stefan Vandewalle, Johan Suykens, Ion Necoara, Jean Berlamont and Yurii Nesterov, who have contributed with helpful and constructive comments and suggestions to improve my thesis.

During my PhD period, I have been enjoying very much the scientific life at SISTA/OPTEC and had a great opportunity to work in an international environment. I had a lot of interaction and collaboration with many colleagues, especially my thanks go to Professor Wim Michiels, Dr. Suat Gumussoy, Dr. Carlo Savorgnan, Dr. Ion Necoara, Dr. Boris Houska, Dr. Paschalis Tsiaflakis, Dr. Marco Signoretto, Attila Kozma, Joel Andersson who have closely been working with me during my time in Leuven. I appreciated the whole administrative staff at ESAT, who helped me a lot at any time that I came. My special thanks to Mrs. Jacqueline De bruyn, Mrs. Ida Tassens, Mrs. Ilse Pardon, Mrs. Elsy Vermoesen and Mr. John Vos for all their help.

I would like to thank other colleagues and friends who gave me very much enjoyable moments both in private and scientific life. It is also an occasion to thank my Vietnamese friends and the Vietnamese community who have encouraged and helped me and my little family during our period in Belgium.

I also wish to thank my wife and my little son who have been going with me to the end of this journey. They have inspired me the motivation to go beyond several difficult moments in my life. I would like to thank my grandparents, my parents and the whole family for their patience and support in all things that I have been doing.

Finally, I want to stress that the thesis would not be completed if there were not supported by KU Leuven Research Council, Optimization in Engineering Center (OPTEC) and several PhD/postdoc and fellow grants.

Leuven, November 7, 2012
Quoc Tran Dinh

Contents

List of acronyms and notations

List of selected acronyms

AD	**A**utomatic **D**ifferentiation
ADMM	**A**lternating **D**irection **M**ethod of **M**ultipliers
APCSCP	**A**djoint-based **P**redictor-**C**orrector **S**equential **C**onvex **P**rogramming
BFGS	**B**royden - **F**letcher - **G**oldfarb - **S**hanno
BMI	**B**ilinear **M**atrix **I**nequality
DAE	**D**ifferential **A**lgebraic **E**quation
DC	**D**ifference of two **C**onvex functions
FASCP	**F**ull-step **A**djoint-based **S**equential **C**onvex **P**rogramming
IP	**I**nterior **P**oint
KKT	**K**arush-**K**uhn-**T**ucker
LICQ	**L**inear **I**ndependence **C**onstraint **Q**ualification
LMI	**L**inear **M**atrix **I**nequality
MSSCP	**M**ulti**s**tage **S**tochastic **C**onvex **P**rogramming
MPC	**M**odel **P**redictive **C**ontrol
NMPC	**N**onlinear **M**odel **P**redictive **C**ontrol
NSDP	**N**onconvex **S**emi**d**efinite **P**rogramming
ODE	**O**rdinary **D**ifferential **E**quation
PCSCP	**P**redictor-**C**orrector **S**equential **C**onvex **P**rogramming
QCQP	**Q**uadratically **C**onstrained **Q**uadratic **P**rogramming
QP	**Q**uadratic **P**rogramming
RTSCP	**R**eal-**T**ime **S**equential **C**onvex **P**rogramming
SepCOP	**Sep**arable **C**onvex **O**ptimization **P**roblem
SCP	**S**equential **C**onvex **P**rogramming
SepNCOP	**Sep**arable **N**on**c**onvex **O**ptimization **P**roblem

SDP	**S**emi**d**efinite **P**rogramming
SOC	**S**econd **O**rder **C**one
SOSC	**S**econd **O**rder **S**ufficient **C**ondition
SQP	**S**equential **Q**uadratic **P**rogramming
SSDP	**S**equential **S**emi**d**efinite **P**rogramming
SSOSC	**S**trong **S**econd **O**rder **S**ufficient **C**ondition

List of selected symbols

Basic

$x, y, f, g, \ldots$	variables, functions or vectors
$x_i, y_i, z_i, \ldots$	sub-vectors or vector components
$A, B, \Omega, \ldots$	matrices or sets
$x^T,\ A^T$	vector and matrix transpose
$A^{-1},\ \mathcal{L}^{-1}$	matrix inverse or inverse mapping
$>, \geq, <, \leq$	scalar or vector inequality
$\succ, \succeq, \prec, \preceq$	matrix inequality
$\lfloor a \rfloor$	maximum integer number which is less than or equal to a

Inner products and norms

$x^T y$	inner product of two vectors x and y
$\|\cdot\|$ and $\|\cdot\|_*$	generic norm and its dual norm
$\|\cdot\|_x$	local norm induced by a self-concordant function
$\|\cdot\|_2$	Euclidean norm
$\|\cdot\|_Q$	Euclidean norm induced by a positive definite matrix Q
$\|\cdot\|_F$	Frobenius norm of a matrix

Sets

$\mathbf{R}$	set of real numbers
$\mathbf{R}_+$	set of nonnegative numbers
$\mathbf{R}_{++}$	set of positive numbers
$\mathbf{R}^n$	set of n-real vectors
$\mathbb{C}$	set of complex numbers
$\mathcal{S}^n$	set of $n \times n$ symmetric matrices
$\mathcal{S}^n_+$	set of $n \times n$ symmetric positive semidefinite matrices
$\mathcal{S}^n_{++}$	set of $n \times n$ symmetric positive definite matrices
$\mathcal{B}(x, r)$	open ball of a radius r centered at x
$\mathrm{dom}(f)$	domain of a function f
$\overline{X}$	closure of a given set X
$\mathrm{ri}(X)$	relative interior of a given set X

$\text{int}(X)$	interior of a given set X
$a + X$	Minkowski sum of a vector a and a set X in $\mathbf{R}^n$

Derivatives

∇f or $\nabla_x f$	gradient vector of a function f (w.r.t. x)
$g'(x)$	Jacobian matrix of a vector function g at x
$\nabla^2 f$ or $\nabla_x^2 f$	Hessian matrix of a function f (w.r.t. x)
$\frac{\partial f}{\partial x}$	first order partial derivative of f w.r.t. x
$\frac{\partial^2 f}{\partial x^2}$ or $\frac{\partial^2 f}{\partial x \partial y}$	second order partial derivative of f w.r.t. x (and y)
$\mathcal{C}^k(X)$	set of k-times continuously differentiable functions ($k \geq 0$)

Chapter 1

Introduction

Many practical problems in science, engineering and business require the solution of an optimization problem [11, 25, 56]. This problem can be obtained directly from a mathematical formulation of a practical problem or results as an auxiliary problem from other solution methods. In some standard classes of convex optimization problems such as linear, quadratic and conic programming, theory and methods have been well-developed and numerous applications are carried out. In contrast to this, solving a nonconvex optimization problem is still a big challenge. It was shown theoretically that the problem of finding a global solution of a general optimization problem is unsolvable [142]. Several attempts in theory and methodology have been made for more than sixty years and resulted in many areas of engineering and sciences [66, 69, 148]. However, solving efficiently a nonconvex optimization problem is a crucial requirement in several applications nowadays [30, 142, 148].

1.1 Motivation and objectives

The motivation of the thesis is the observation that many optimization problems in practice possess certain structures such as convexity, sparsity and separability which can be exploited efficiently in numerical solution methods. These structures may originate naturally from the modeling stage of the problem or may appear later due to its simplification, reformulation or relaxation. In practice, the sparsity structure of problems is usually exploited at a lower-level of a solution method such as at the linear algebra level. In this research, we focus our consideration on two specific structures of the optimization problems,

namely *convexity* and *separability*. Several examples have shown that exploiting properly the structures of problems lead to an efficient solution method, see e.g. [47, 200, 209, 211].

Convexity. Convexity plays a central role in optimization problems and solving a convex optimization problem is more efficient and reliable than a nonconvex one. If the convexity appears explicitly in the optimization problems such as in linear programming and quadratic programming, then it can be exploited efficiently in numerical solution methods, e.g. by interior point methods [146, 213]. However, we often meet optimization problems which contain implicitly a convex structure. This convexity may arise as a natural source in the problem, by reformulating the original problem, by convexifying some components of the problem or by robustifying the original problem under uncertainties. Several methods based on exploiting implicitly convex structures have been developed recently and resulting in many applications, see e.g. [11, 12, 81, 102, 200, 209]. Unfortunately, many highly important practical problems are nonconvex. Nevertheless, they still possess convex substructures as we can see in the following examples.

Example 1.1.1.([51]) An approximate robust counterpart formulation of an optimization problem under uncertain parameters results in the following problem:

$$\begin{array}{ll} \min\limits_{\bar{w},u,D} & f_0(w,u) + \left\| D^T \frac{\partial f_0(\bar{w},u)}{\partial w}^T \right\|_2 \\ \text{s.t.} & f_i(\bar{w},u) + \left\| D^T \frac{\partial f_i(\bar{w},u)}{\partial w}^T \right\|_2 \leq 0,\ i = 1,\ldots,n_f, \\ & g(\bar{w},u,\bar{\xi}) = 0, \\ & \frac{\partial g(\bar{w},u,\bar{\xi})}{\partial w} D + \frac{\partial g(\bar{w},u,\bar{\xi})}{\partial \xi} = 0, \end{array} \tag{1.1.1}$$

where f_i and g are smooth and $\xi \in \Gamma := \left\{\eta \mid \left\|\eta - \bar{\xi}\right\| \leq 1\right\}$ is the set of uncertainties. In fact, problem (1.1.1) is obtained by linearizing the robust counterpart problem of an optimization problem. Here, D is a sensitivity matrix which is defined as $D := -\left(\frac{\partial g}{\partial w}(\bar{w},u,\bar{\xi})\right)^{-1} \frac{\partial g}{\partial \xi}(\bar{w},u,\bar{\xi})$. This matrix is also optimization variables of (1.1.1). Problem (1.1.1) is in general *nonconvex* but contains *second order cone* structures. ♢

Example 1.1.2. The second example that we are interested in is an optimization problem with a bilinear matrix inequality (BMI) constraint. This problem originates from the problem of finding a static output feedback control law $u = Fy$ to stabilize the linear time invariant system $\dot{x} = Ax + Bu$ and $y = Cx$. By employing Lyapunov's theory, this problem leads to the following

formulation:

$$\begin{array}{ll} \min\limits_{\gamma,F,P} & \gamma \\ \text{s.t.} & P \succ 0,\ (A+BFC)^T P + P(A+BFC) + 2\gamma P \prec 0, \end{array} \tag{1.1.2}$$

where A, B, C are given. This problem is indeed *nonconvex* and also known as an abscissa spectral problem [205]. However, if we either fix variables F and γ or variable P then the resulting problem becomes convex and is in fact a standard semidefinite program. More general, many other problems such as sparse linear static output feedback controller design, pseudo-spectral abscissa optimization, $\mathcal{H}_2$, $\mathcal{H}_\infty$ and mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control can be cast into optimization problems with BMI constraints [118, 191]. ♢

A natural idea to solve a nonconvex optimization problem is to approximate it by convex ones. Nevertheless, there are still several cornerstone questions that need to be tackled. For instance, how can we convexify the nonconvex parts? What is the relation between the convex approximation problem and the original one? In this research we focus on developing numerical solution methods for solving certain classes of nonconvex optimization problems.

In the framework of nonlinear model predictive control (NMPC) and moving horizon estimation (MHE) [21, 54, 68, 129, 158], the underlying optimal control problems usually depend on parameters such as initial states. Treating these problems by taking into account the parameter dependence are crucial for those applications. The theory related to parametric optimization has been intensively studied in several monographs such as [28, 66, 85, 112, 162]. A main feature of NMPC and MHE is the time limitation which any numerical optimization method must satisfy. One method which fits very well this requirement is the *real-time iteration* developed in [50, 53, 55]. The approach employs the fact that the solutions of the optimization problem at two successive parameter values are close to each other. This allows one to derive an approximate solution of the optimization problems by performing only *one iteration* of a Newton-type optimization algorithm. A similar approach based on continuation methods can be found in [149]. Motivated from these works, we study in this thesis a generalized framework of the real-time iteration scheme where we can handle both the inexactness of the derivative information and the generalized convexity of the constraints.

Without parameters, theory and methods for solving nonlinear optimization problems have been well developed and can be found in many numerical optimization monographs, e.g. [67, 69, 148]. Two conventional methods for solving nonparametric optimization problems are sequential quadratic programming (SQP) and interior point methods (IP). Roughly speaking, the SQP algorithms solve the nonlinear optimization problem by employing the

solutions of a sequence of quadratic programming subproblems, while, in one view, the nonlinear IP methods treat the problem via a sequence of linear equation systems obtained by linearizing the KKT system of the corresponding barrier problem. Both approaches require that all the nonlinear constraints of the original problem are linearized or approximately linearized. If the problem possesses a convex structure such as second order cone or semidefinite cone constraints then these approaches do not adequately capture the specific structures of the problem.

Separability. In addition to convexity, separable structures are also important for developing numerical solution methods in optimization. The separability of the problems arises naturally in several applications such as optimization in networks, machine learning, multistage stochastic optimization and distributed model predictive control. It also results from other solution methods such as direct multiple shooting methods in optimal control. Let us give some examples to illustrate our motivation.

Example 1.1.3. The following example covers two well-studied problems in networks, namely resource allocation and network utility maximization problems:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \sum_{i=1}^{M} f_i(x_i), \\ \text{s.t.} & \sum_{i=1}^{M} x_i \leq c, \\ & 0 \leq x_i \leq a_i, \quad i = 1, \ldots, M, \end{array} \tag{1.1.3}$$

where $f_i : \mathbf{R}^{n_i} \to \mathbf{R}$ represents a utility or a cost function, a_i is the vector of capacities of each agent for $i = 1, \ldots, M$ and c is the vector of total capacities. This problem is indeed separable and usually large-scale when the number M of agents or nodes in the network increases. ♢

Example 1.1.4. The second example is an optimization problem arising in machine learning. This problem is known as a sparse recovery formulation:

$$\begin{array}{ll} \min\limits_{x} & \|x\|_p \\ \text{s.t.} & Ax = y, \end{array} \tag{1.1.4}$$

where $A \in \mathbf{R}^{m\times n}$ and $y \in \mathbf{R}^m$ are given recover operator and measurements, respectively, and $p \in [0, 1]$, where $\|x\|_p := (\sum_{i=1}^{n} |x_i|^p)^{1/p}$ if $p > 0$ and $\|x\|_p := \#\{i \ : \ x_i \neq 0\}$ the cardinality of x if $p = 0$. This problem is usually nonconvex and turns out to be convex if $p = 1$. ♢

A common approach to tackle the separable structure in optimization is decomposition [17]. This approach decomposes the original optimization problem into several subproblems which can be solved in a parallel or distributed manner. If these problems possess a fixed sparsity structure then they may efficiently be solved by optimization solvers which make use of exploiting sparsity [19].

Here, we are more interested in the opposite case where the considered problem is not suitable for general-purpose solvers or possesses a dynamic structure where we can easily add or remove components without changing the whole implementation of the algorithm. We are particularly interested in the case when the subproblem formed by each component can be solved easily in a closed form. As the second major aim of the thesis, we develop several decomposition methods for solving separable optimization problems both in the convex and nonconvex cases which can be implemented in a parallel or distributed way. Note that separable convex optimization problems have extensively been studied in the literature, see e.g. [17, 45, 61, 136] both in general setting and particular applications. Nevertheless, the existing methods developed in this direction remains encountering several disadvantages such as slow convergence rate or limited in a specific class of problems. Our aim is to develop some numerical optimization methods for solving separable optimization problems which exploit the advantages of the existing methods and employing recent development in convex optimization [11, 30, 142].

1.2 Three main concepts

The methodology that we use in this thesis is based on several concepts and techniques. The three concepts which are particularly important are convexity, path-following and decomposition, and have been used throughout the thesis. Let us briefly present them here.

Convexity

Convexity is not only a central concept in optimization but also in many other areas of applied mathematics. Thanks to convex analysis [161] the understanding of convex optimization is more thorough than of nonconvex optimization. It is also a powerful tool for studying nonconvex optimization as well as related problems such as optimal control and variational inequalities. Moreover, all the solutions of a convex problem are global. Several subclasses of convex problems with explicit structures such as conic programming and geometric programming can be solved efficiently by means of polynomial time algorithms such as interior point methods [11, 146, 163]. These problem classes become standard in optimization. Besides, thanks to the development of disciplined convex programming [81, 124], several nonstandard convex optimization problems have been solved efficiently by a reformulation to the standard ones. The disciplined convex programming approach is a methodology to construct convex optimization models by enforcing a set of basic conventions on the models. This

set of conventions allows one to analyze, manipulate and transform automatically a general convex optimization model into simplified forms that can be tackled by standard optimization solvers. Alternatively to the disciplined convex optimization approach, a structural optimization approach has also been well-studied [142]. By properly exploiting the structures of the problem, one can design better methods for solving convex programming problems.

Recently, many convex optimization problems arising from statistics, image and signal processing and machine learning have attracted interest [10, 30] and demand new approaches to solve them. One active research direction is based on gradient and fast gradient methods originally developed by Nesterov [137]. Several variants of the fast gradient method have been developed rendering an exponential increase both in theory and applications of this direction, see e.g. [10, 49, 134, 141, 145].

In the nonconvex case, convex optimization approaches are also main tools to design solution methods for solving nonconvex optimization problems. Several techniques have been used to exploit convex structure such as nonlinear transformation, relaxation, dualization and convexification [200, 209]. Although many techniques have been proposed based on exploiting convexity of the problems, it is still worthwhile to develop more efficient and reliable methods for solving nonconvex problems by making use of a state-of-the-art convex optimization.

Path-following scheme

Path-following method is often also referred to as a continuation method or a homotopy method and is a technique to treat problems which depend on parameters, see e.g. [4] for a good review. A solution of those problems is usually represented as a mapping of the parameters and generates a trajectory or a path in the parameter space. Classical path-following methods for parametric optimization solve each optimization problem obtained at a given value of the parameter to generate an approximate solution at such a point and then update the parameters for the next step. Since solving the optimization problem until complete convergence is usually time consuming, these approaches are rather limited in real-time applications. Besides, in many applications, we do not require a highly accurate solution of the optimization problem at the intermediate values of the parameters. Therefore, we can solve this optimization problem up to a certain accuracy and then perform the next iteration. This idea was implemented in path-following interior-point methods which, at each value of the penalty parameter, only perform one Newton-type step of the whole

optimization procedure and then update the penalty parameter for the next iteration, see e.g. [79, 130, 142, 146, 213].

In the framework of nonlinear model predictive control, the idea of path-following methods has proved to be efficient. This method was proposed by Diehl *et al* [50, 53] and is called *real-time iteration*. The real-time iteration scheme solves the underlying optimization problems at a given sampling time by performing only *one iteration* of an SQP-type or a Gauss-Newton-type procedure. This means that only one QP problem is solved at each sampling time. By employing the tools of sensitivity analysis, it was proved under standard assumptions that the solution of the QP problem provides a good approximation to the solution of the underlying optimization problems to maintain the stability of the tracking errors between approximate solutions and the true ones provided that the sampling time is sufficiently small. A similar approach was proposed in [149]. Further extensions can be found in [217].

Decomposition

Optimization problems that have a fixed sparsity structure or are of small and medium size can be solved by standard centralized optimization algorithms [19, 78]. If the problems are large-scale and possess a dynamic structure due to the variation of the topology or have a distributed data location then decomposition approaches would be an appropriate choice. Roughly speaking, decomposition approaches divide the given optimization problem into several subproblems which can be solved more easily than the original one. However, this approach only works if the given problem possesses certain structures such as separability. Several decomposition methods have been proposed to solve large-scale convex optimization problems such as Dantzig-Wolfe decomposition, Benders' decomposition and Fenchel's dual decomposition [45, 75, 89, 219] to just name a few. If the problem is separable and convex then classical Lagrangian dual decomposition can be used. The main assumption in this approach is that strong duality holds. Unfortunately, this requirement is not fulfilled in the nonconvex case. Several attempts have been proposed to solve a nonconvex problem by applying augmented Lagrangian techniques, Jacobi and Gauss-Seidel iteration schemes [17, 164]. The approach in this thesis is to combine Lagrangian dual decomposition, smoothing techniques and a path-following method to design competitive algorithms for solving separable optimization problems.

1.3 Contributions of the thesis and overview

This thesis will focus on developing *numerical optimization methods* for some classes of *nonlinear optimization problems*, namely parametric nonconvex optimization, nonconvex semidefinite programming and separable convex and nonconvex optimization problems. These classes of problems cover many practical problems in different fields and subfields of engineering and science such as nonlinear model predictive control, robust optimization, robust control, optimization in networks, machine learning, multistage stochastic optimization and distributed model predictive control. The thesis is divided into two parts, namely *Sequential Convex Programming* (SCP) and *Decomposition in Separable Optimization*.

Sequential convex programming

The aim of Part I is to develop local optimization methods for solving some classes of nonconvex programming problems based on exploiting the convexity of the problems. The contribution of this part is presented in four chapters that cover the following two areas:

Adjoint-based predictor-corrector sequential convex programming. We develop a local optimization method called *adjoint-based predictor-corrector sequential convex programming* (APCSCP) for parametric nonconvex optimization in Chapter 2. The algorithm is a combination of three ingredients consisting of sequential convex programming (SCP), predictor-corrector path-following and adjoint based optimization techniques. In other words, it solves a sequence of nonconvex optimization problems by performing only one iteration of the whole optimization procedure, namely sequential convex programming (SCP) for each problem and makes use of inexact Jacobian information of the nonlinear equality constraints. The stability of the tracking errors between the approximate KKT points generated by the algorithm and the true KKT points of the problem is proved under standard assumptions including *strong regularity* [160]. This algorithm is suitable for nonlinear model predictive control applications. If the parameter is absent then the algorithm coincides with a local optimization method for solving nonlinear nonconvex programming problems which we call *sequential convex programming* (SCP) [188]. We show that the local convergence of this algorithm is linear. Both algorithms are tested through two numerical examples in Chapter 3. The first example is an NMPC problem of a hydro power plant with 259 states and 10 controls, while the second example is a time optimal trajectory planning problem of a car motion. The results presented

in Chapters 2 and 3 have been published in the journal paper [196] and the conference papers and book chapters [133, 187, 188, 198, 199].

Generalized inner convex approximation algorithms. We study a *generalized inner convex approximation method* for solving a class of nonconvex semidefinite programming problems in Chapter 4. As a variant of this method, a new algorithm called *generalized convex-concave decomposition* is obtained. The idea is to decompose a nonconvex matrix mapping in semidefinite constraints as a sum of a convex matrix-valued mapping and a concave matrix-valued mapping. Then the concave part is linearized to obtain a convex programming problem. An iterative method is derived to solve the given problem by exploiting standard semidefinite programming techniques. The convergence of these algorithms is proved. Applications of both algorithms to static state/output feedback controller design are shown in Chapter 5. We show that these algorithms can be applied to solve many optimization problems arising in static state/output feedback controller design including spectral abscissa, $\mathcal{H}_2$-control, $\mathcal{H}_\infty$-control and mixed $\mathcal{H}_2/\mathcal{H}_\infty$-control problems. Some heuristic procedures are also provided to compute the starting point for the algorithms. Numerical results and comparisons are made by using the data from the `COMPL`$_{\texttt{e}}$`ib` library [119]. The results presented in Chapters 4 and 5 have been published in the journal paper [191] and the conference paper [192].

Decomposition in separable optimization

The aim of Part II is to develop efficient numerical algorithms for solving separable optimization problems based on decomposition techniques. First, we focus our study on separable convex optimization problems and dual decomposition methods for solving them. Then we extend the results to separable nonconvex optimization problems as a special case of the SCP approach. The contribution of this part consists of five chapters that cover the following areas:

Smoothing via proximity functions and first order decomposition methods. After reviewing some related existing decomposition methods in convex and nonconvex optimization problems, we briefly recall the classical Lagrangian dual decomposition method and some concepts related to parallel and distributed algorithms and performance profiles in Chapter 6. In Chapter 7, we first present the smoothing technique based on proximity functions in [145] and the excessive gap technique introduced in [140] in the framework of dual decomposition. Then, we propose two new decomposition algorithms for solving separable convex

optimization problems. These algorithms can be classified as the first-order methods using the optimal scheme in the sense of Nesterov [142]. Two different variants of these algorithms are considered and their convergence second order sufficient conditionrate is established. As a special case, the second algorithm is specialized to the strongly convex case where we obtain a new variant which has the convergence rate $O(1/k^2)$, where k is the iteration counter. Extensions to the inexact case are also investigated, where we allow one to solve the primal subproblems formed from each component of the problem inexactly. Theoretical comparison is made and numerical tests are implemented to verify the performance of the algorithms and to compare them. The results obtained in Chapter 7 have contributed the main content to the two journal papers [193, 197].

Smoothing via barrier functions and path-following decomposition methods. In Chapter 8, we first present a smoothing technique based on self-concordant barrier functions. In addition to the related existing results in the literature, we provide some new bounds between the smoothed dual functions and the original dual function. Then, we propose a path-following gradient decomposition method and a fast gradient decomposition method for solving separable convex optimization problems. The convergence of these algorithms is analyzed and the local convergence rate is estimated. Numerical examples are also implemented to verify the theoretical development. The results presented in Chapter 8 have contributed the main content to the journal paper [194].

In Chapter 9, we present an inexact-perturbed path-following decomposition algorithm. In this algorithm, the primal convex subproblems formed from each component of the problem are assumed to be solved inexactly. This leads to an inexactness in the gradient vectors and the Hessian matrices of the smoothed dual function. An inexact-perturbed path-following method is proposed to solve the smoothed dual problem. Under appropriate choices of the accuracy levels, we prove that the worst-complexity bound of the method remains the same as in the exact case scaled by a constant. We also show that the exact path-following decomposition methods studied in [113, 131, 133, 171, 218] are special cases of this algorithm. The algorithms developed in this chapter are also verified via numerical examples. The results presented in Chapter 9 have been published in the journal paper [195].

Application to separable nonconvex optimization. In Chapter 10, we present an application of the previous approaches to solve separable nonconvex optimization problems. The idea is to combine the SCP approach in Part I and the decomposition in separable convex optimization problems to obtain a

two-level optimization algorithm for solving separable nonconvex optimization problems. More precisely, in the first level, we exploit the idea in [141, 143, 189] to build a new variant of the SCP algorithm proposed in Chapter 2 for solving the given nonconvex problem. Then, in the second level, the subproblems in this SCP variant are strongly convex and can be solved by applying the decomposition framework studied in the previous chapters. Numerical tests are also implemented to verify the proposed algorithm.

The interdependence of the chapters in this thesis is shown in Figure 1.1.

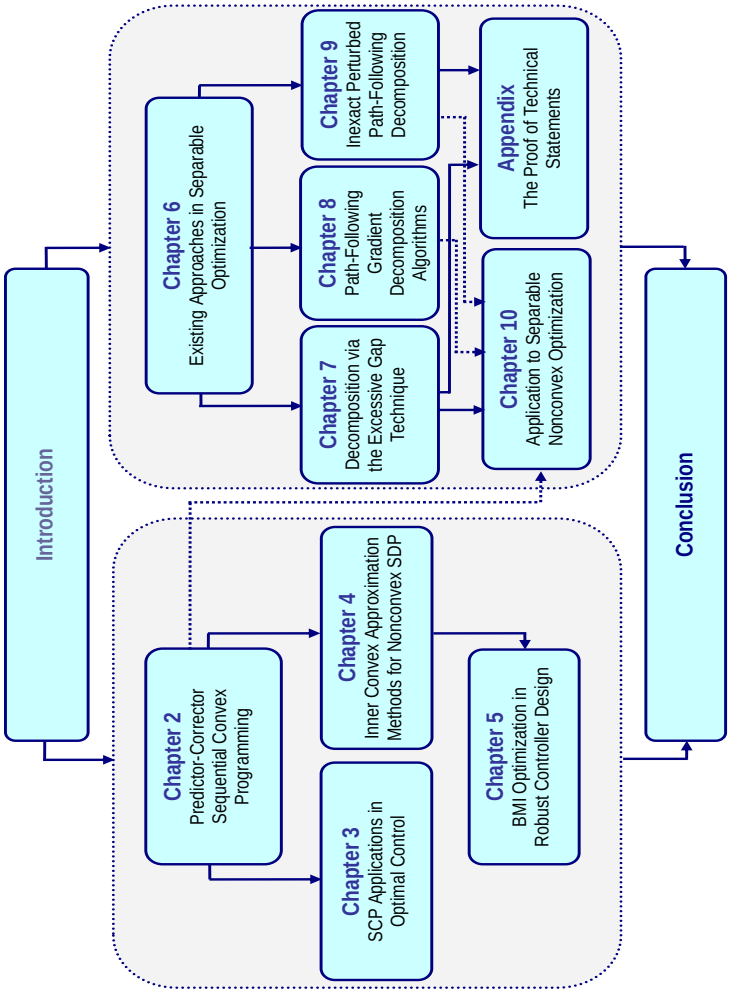


Figure 1.1: The structure of the thesis

Part I

Sequential Convex Programming

Chapter 2

Predictor-corrector sequential convex programming

2.1 Problem statement and contribution

We are interested in developing an optimization method for calculating the approximate solutions of a sequence of nonlinear optimization problem instances of the following parametric nonconvex optimization problem:

$$\left\{\begin{array}{ll} \min\limits_{x\in\mathbf{R}^n} & f(x) \\ \text{s.t.} & g(x)+M\xi=0, \\ & x\in\Omega, \end{array}\right. \qquad \mathrm{P}(\xi)$$

where $f:\mathbf{R}^n\to\mathbf{R}$ is convex, $g:\mathbf{R}^n\to\mathbf{R}^m$ is nonlinear, $\Omega\subseteq\mathbf{R}^n$ is a nonempty, closed convex set, and the parameter ξ belongs to a given subset $\mathcal{P}\subseteq\mathbf{R}^p$. Matrix $M\in\mathbf{R}^{m\times p}$ plays the role of embedding the parameter ξ into the equality constraints in a linear way. Throughout this chapter, f and g are assumed to be differentiable on their domain.

The problem formulation P(ξ) covers many (parametric) nonlinear programming problems in practice such as standard nonlinear programs, nonlinear second order cone programs, and nonlinear semidefinite programs [110, 148, 179]. Moreover, the theory and methods for parametric optimization have been extensively studied in many research papers and monographs, see, e.g. [29, 74, 160, 162].

This chapter deals with the efficient calculation of approximate solutions to a sequence of problems of the form P(ξ), where the parameter ξ is slowly varying.

In other words, for a sequence $\{\xi_k\}_{k\geq 0}$ such that $\|\xi_{k+1} - \xi_k\|$ is small, we want to solve the problems $\mathrm{P}(\xi_k)$ in an efficient way without requiring more accuracy than needed in the result.

In practice, sequences of problems of the form $\mathrm{P}(\xi)$ arise in the framework of real-time optimization, moving horizon estimation, online data assimilation as well as in nonlinear model predictive control (NMPC). A practical obstacle in these applications is the time limitation imposed on solving the underlying optimization problem for each value of the parameter. Instead of solving completely a nonlinear program at each sample time [20, 21, 26, 96], several online algorithms approximately solve the underlying nonlinear optimization problem by performing only one iteration of exact Newton, sequential quadratic programming (SQP), Gauss-Newton or interior point methods [53, 149, 217]. In [53, 149] the authors only considered the algorithms in the framework of SQP methods. This approach has been proved to be efficient in practice and is widely used in many applications [50]. Recently, Zavala and Anitescu [217] proposed an inexact Newton-type method for solving online optimization problems based on the framework of generalized equations [29, 160].

Other related work considers practical problems which possess general convexity structure such as second order cone and semidefinite cone constraints, nonsmooth convexity [65, 179]. In these applications, standard optimization methods may not perform satisfactorily. Many algorithms for nonlinear second order cone and nonlinear semidefinite programming have recently been proposed and found many applications in robust optimal control, experimental design, and topology optimization, see, e.g. [8, 65, 72, 111, 179]. These approaches can be considered as generalizations of the SQP method. Although solving semidefinite programming problems is in general time consuming due to matrix operations, in some practical applications, the problems may possess a few expensive constraints such as second order cone or semidefinite cone constraints. In this case handling these constraints directly in the algorithm may be more efficient than transforming them into scalar constraints.

Contribution of Chapter 2. The contribution of this chapter is as follows:

a) We start this chapter by proposing a generic framework called the *adjoint-based predictor-corrector sequential convex programming* (APCSCP) method for solving parametric optimization problems of the form $\mathrm{P}(\xi)$. The algorithm is specially suited for solving nonlinear model predictive control problems where the evaluations of the derivatives are time consuming. For example, it can show advantages with respect to standard techniques when applied to problems in which the number of state variables in the dynamic system is much larger than the number of control variables.

b) We prove the stability of the tracking error between the approximate solutions and the true ones for this algorithm (Theorem 2.4.2).

c) In the second part of this chapter the theory is specialized to the non-parametric case where a single optimization problem is solved. The local convergence of this variant is also obtained.

The APCSCP method develop in this chapter can be considered as a combination of three techniques, namely sequential convex programming, predictor-corrector path-following and adjoint-based optimization. SCP allows the algorithm to handle general convex constraints while the adjoint-based methods reduce significantly the computational time for evaluating the derivatives of the constraint functionals.

Outline of Chapter 2. The outline of this chapter is as follows. In Section 2.2 we briefly describe three ingredients that we use to develop the algorithms. Section 2.3 presents a generic framework of the *adjoint-based predictor-corrector SCP algorithm* (APCSCP). Section 2.4 proves the local contraction estimate for APCSCP and the stability of the approximation errors. Section 2.5 considers an *adjoint-based SCP algorithm* for solving nonlinear programming problems as a special case.

2.2 Three ingredients of the algorithm

APCSCP is based on three main ideas: sequential convex programming, predictor-corrector path-following and adjoint-based optimization. We briefly explain these methods in the following.

Sequential convex programming

The sequential convex programming (SCP) method is a local nonconvex optimization technique. SCP solves a sequence of convex approximations of the original problem by convexifying only the nonconvex parts and preserving the structures that can efficiently be exploited by convex optimization techniques [30, 128, 142]. Note that this method is different from SQP methods where quadratic programs are used as approximations of the problem. The SCP approach is useful when the problem possesses general convex structures such as conic constraints, a cost function depending on matrix variables or convex constraints resulting from a low level problem in multi-level settings [8, 51, 179]. Due to

the complexity of these structures, standard optimization techniques such as SQP and Gauss-Newton-type methods may not be convenient to apply. In the context of nonlinear conic programming, SCP approaches have been proposed under the names sequential semidefinite programming (SSDP) or SQP-type methods [44, 65, 72, 110, 111, 179]. It has been shown in [57] that the superlinear convergence is lost if the linear semidefinite programming subproblems in the SSDP algorithm are convexified. In [123] the authors considered a nonlinear program in the framework of a composite minimization problem, where the inner function is linearized to obtain a convex subproblem which is made strongly convex by adding a quadratic proximal term.

In this chapter, following the work in [65, 71, 133, 188, 197], we apply the SCP approach to solve problem P(ξ). The nonconvex constraint $g(x) + M\xi = 0$ is linearized at each iteration to obtain a convex approximation. The resulting subproblems can be solved by exploiting convex optimization techniques.

We would like to note that the term "sequential convex programming" was also often used in structural optimization but with a different view, see, e.g. [70, 221]. The cited papers are related to the method of moving asymptotes introduced by Svanberg [182].

Predictor-corrector path-following method

In order to illustrate the idea of the predictor-corrector path-following method [48, 217] and to distinguish it from the other "predictor-corrector" concepts, e.g. the well-known predictor-corrector interior point method proposed by Mehrotra in [130], we summarize the concept of "predictor-corrector path-following methods" in the case $\Omega \equiv \mathbb{R}^n$ as follows.

The KKT system of problem P(ξ) can be written as $F(z;\xi) = 0$, where $z = (x, y)$ is its primal-dual variable. The solution $z^*(\xi)$ that satisfies the KKT condition for a given ξ is in general a smooth map. By applying the implicit function theorem, the derivative of $z^*(\cdot)$ is expressed as:

$$\frac{\partial z^*}{\partial \xi}(\xi) = -\left[\frac{\partial F}{\partial z}(z^*(\xi);\xi)\right]^{-1} \frac{\partial F}{\partial \xi}(z^*(\xi);\xi).$$

In the parametric optimization context, we might have solved a problem at the parameter value $\bar{\xi}$ with the solution $\bar{z} = z^*(\bar{\xi})$ and want to solve the next problem for a new parameter value $\hat{\xi}$. The tangential predictor $\hat{z}$ for this new solution $z^*(\hat{\xi})$ is given by:

$$\hat{z} = z^*(\bar{\xi}) + \frac{\partial z^*}{\partial \xi}(\bar{\xi})(\hat{\xi} - \bar{\xi}) = z^*(\bar{\xi}) - \left[\frac{\partial F}{\partial z}(z^*(\bar{\xi});\bar{\xi})\right]^{-1} \frac{\partial F}{\partial \bar{\xi}}(z^*(\bar{\xi});\bar{\xi})(\hat{\xi} - \bar{\xi}).$$

Note the similarity with one step of a Newton method. In fact, a combination of the tangential predictor and the corrector due to a Newton method proves to be useful in the case that $\bar{z}$ was not the exact solution of $F(z;\bar{\xi}) = 0$, but only an approximation. In this case, linearization at $(\bar{z},\bar{\xi})$ yields a formula that one step of a *predictor-corrector path-following method* needs to satisfy:

$$F(\bar{z};\bar{\xi}) + \frac{\partial F}{\partial \xi}(\bar{z};\bar{\xi})(\hat{\xi}-\bar{\xi}) + \frac{\partial F}{\partial z}(\bar{z};\bar{\xi})(\hat{z}-\bar{z}) = 0. \tag{2.2.1}$$

Written explicitly, it delivers the solution guess $\hat{z}$ for the next value $\hat{\xi}$ as:

$$\hat{z} = \bar{z} \underbrace{- \left[\frac{\partial F}{\partial z}(\bar{z};\bar{\xi})\right]^{-1} \frac{\partial F}{\partial \xi}(\bar{z};\bar{\xi})(\hat{\xi}-\bar{\xi})}_{=\Delta z_{\text{predictor}}} \underbrace{- \left[\frac{\partial F}{\partial z}(\bar{z};\bar{\xi})\right]^{-1} F(\bar{z};\bar{\xi})}_{=\Delta z_{\text{corrector}}}. \tag{2.2.2}$$

Note that when the parameter enters linearly into F, we can write:

$$\frac{\partial F}{\partial \xi}(\bar{z};\bar{\xi})(\hat{\xi}-\bar{\xi}) = F(\bar{z};\hat{\xi}) - F(\bar{z};\bar{\xi}) \quad \text{and} \quad \frac{\partial F}{\partial z}(\bar{z};\bar{\xi}) = \frac{\partial F}{\partial z}(\bar{z}).$$

Thus, equation (2.2.1) reduces to:

$$F(\bar{z};\hat{\xi}) + \frac{\partial F}{\partial z}(\bar{z})(\hat{z}-\bar{z}) = 0. \tag{2.2.3}$$

It follows that the predictor-corrector step can be easily obtained by just applying one standard Newton step to the new problem $\mathrm{P}(\hat{\xi})$ initialized at the past solution guess $\bar{z}$, if we employ the parameter embedding in the problem formulation [50].

Based on the above analysis, the predictor-corrector path-following method only performs the first iteration of the exact Newton method for each new problem. In this chapter, by applying the generalized equation framework [160, 162], we generalize this idea to the case where more general convex constraints are considered. When the parameter does not enter linearly into the problem, we can always reformulate this problem as $\mathrm{P}(\xi)$ by using intermediate variables. In this case, the derivatives with respect to these intermediate variables contain the information of the predictor term. Finally, we notice that the real-time iteration scheme proposed in [53] can be considered as a variant of the above predictor-corrector method in the SQP context.

Adjoint-based optimization method

From a practical point of view, most of the time spent on solving optimization problems resulting from simulation-based methods is needed to evaluate

the functions and their derivatives [27]. Adjoint-based methods rely on the observation that it is not necessary to use exact Jacobian matrices of the constraints. Moreover, in some applications, the time needed to evaluate all the derivatives of the functions exceeds the time available to compute the solution of the lower level convex optimization problems. The adjoint-based Newton-type methods in [58, 84, 167] can work with an inexact Jacobian matrix and only require an exact evaluation of the Lagrange gradient using adjoint derivatives to form the approximate optimization subproblems in the algorithm. This technique still allows the algorithm to converge to the exact solutions but can save valuable time in the online implementation of the algorithm.

A tutorial example

The idea of the APCSCP method is illustrated in the following simple example.

Example 2.2.1.(*Tutorial example*) Let us consider a simple nonconvex parametric optimization problem:

$$\begin{cases} \min\limits_{x\in\mathbf{R}^2} & -x_1 \\ \text{s.t.} & x_1^2+2x_2+2-4\xi=0, \\ & x_1^2-x_2^2+1\le 0,\ x\ge 0, \end{cases} \tag{2.2.4}$$

where $\xi\in\mathcal{P}:=\{\xi\in\mathbb{R}\ :\ \xi\ge 1.2\}$ is a parameter. After a few calculations, we can show that $x_\xi^*=(2\sqrt{\xi-\sqrt{\xi}},2\sqrt{\xi}-1)^T$ is a stationary point of problem (2.2.4) which is also the unique global optimum. It is clear that problem (2.2.4) satisfies the *strong second order sufficient condition* (SSOSC) at x_ξ^*.

Note that the constraint $x_1^2-x_2^2+1\le 0$ can be rewritten as a second order cone constraint $\left\|(x_1,1)^T\right\|_2\le x_2$ under the condition $x_2\ge 0$. Let us define $g(x):=x_1^2+2x_2+2$, $M:=-4$ and $\Omega:=\{x\in\mathbb{R}^2\mid \left\|(x_1,1)^T\right\|_2\le x_2,\ x\ge 0\}$. Then, problem (2.2.4) can be cast into the form of P(ξ). The aim is to approximately solve problem (2.2.4) at each given value ξ_k of the parameter ξ. Instead of solving the nonlinear optimization problem at each ξ_k until complete convergence, APCSCP only performs the first step of the SCP algorithm to obtain an approximate solution x^k at ξ_k. Notice that the convex subproblem needed to be solved at each ξ_k in the APCSCP method is:

$$\begin{cases} \min\limits_{x} & -x_1 \\ \text{s.t.} & 2x_1^kx_1+2x_2-(x_1^k)^2+2-4\xi=0, \\ & \left\|(x_1,1)^T\right\|\le x_2, x\ge 0. \end{cases} \tag{2.2.5}$$

We compare this method with other known real-time iteration algorithms. The first one is the real-time iteration with an exact SQP method and the second

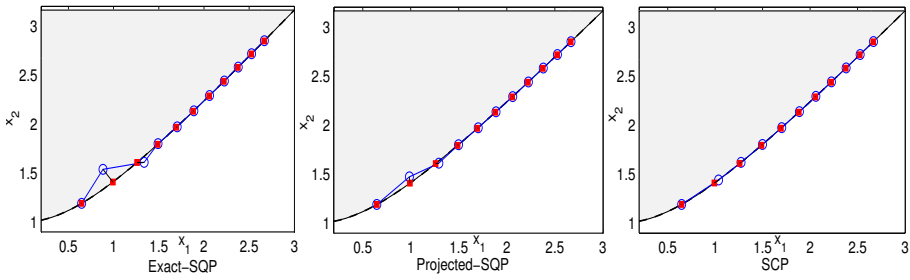


Figure 2.1: The trajectory of three methods ($k = 0, \cdots, 9$), ($\diamond$ is $x^*(\xi_k)$ and $\circ$ is x^k).

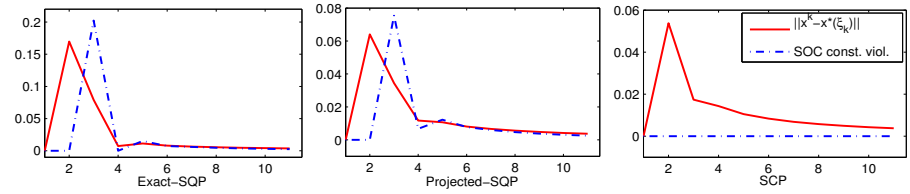


Figure 2.2: The tracking error and the cone constraint violation of three methods ($k = 0, \cdots, 9$).

algorithm is the real-time iteration with an SQP method using a projected Hessian [53, 106]. In the second algorithm, the Hessian matrix of the Lagrange function is projected onto the cone of symmetric positive semidefinite matrices to obtain a convex quadratic programming subproblem.

Figures 2.1 and 2.2 illustrate the performance of three methods when $\xi_k = 1.2 + k\Delta\xi_k$ for $k = 0, \ldots, 9$ and $\Delta\xi_k = 0.25$. Figure 2.1 presents the approximate solution trajectories given by three methods, while Figure 2.2 shows the tracking errors and the cone constraint violations of those methods. The initial point x^0 of three methods is chosen at the true solution of $\mathrm{P}(\xi_0)$. We can see that the performance of the exact SQP and the SQP using projected Hessian is quite similar. In particular, the second order cone constraint $\left\|(x_1, 1)^T\right\|_2 \leq x_2$ is violated in both methods. The SCP method preserves the feasibility and follows better the exact solution trajectory. Note that the subproblem in the exact SQP method is a nonconvex quadratic program, while it is a convex quadratic program in the projected SQP case and is a second order cone constrained program (2.2.5) in the SCP method. $\diamondsuit$

2.3 Adjoint-based predictor-corrector SCP algorithm

In this section, we present a generic algorithmic framework for solving the parametric optimization problem P(ξ). Traditionally, at each sample ξ_k of parameter ξ, a nonlinear program P(ξ_k) is solved to get a completely converged solution $\bar{z}(\xi_k)$. Exploiting the real-time iteration idea [50, 53], in our algorithm below, only one convex subproblem is solved to get an approximated solution z^k at ξ_k to $\bar{z}(\xi_k)$.

Let us assume that the convexity in the objective function and the constraint $x \in \Omega$ can be efficiently exploited by convex optimization techniques. We generate our convex subproblem by approximating the nonlinear constraint $g(x) + M\xi = 0$, adding a correction term to correct the difference between the approximate Jacobian of g and its true Jacobian g' and adding a regularization term to capture the curvature of the constraints. More precisely, the convex subproblem is generated as follows. Suppose that $z^k := (x^k, y^k) \in \Omega \times \mathbf{R}^m$ is a given KKT point of P(ξ_k) (more details can be found in the next section), A_k is a given $m \times n$ matrix and $H_k \in \mathcal{S}_+^n$. We consider the following parametric convex programming subproblem:

$$\left\{ \begin{array}{ll} \min\limits_{x \in \mathbf{R}^n} & \left\{ f(x) + (s^k)^T (x - x^k) + \frac{1}{2}(x - x^k)^T H_k (x - x^k) \right\} \\ \text{s.t.} & A_k(x - x^k) + g(x^k) + M\xi = 0, \\ & x \in \Omega, \end{array} \right. \qquad \mathrm{P}(z^k, A_k, H_k; \xi)$$

where $s^k := s(z^k, A_k) = \left(g'(x^k) - A_k\right)^T y^k$. Matrix A_k is an approximation to $g'(x^k)$ at x^k, H_k is a regularization or an approximation to $\nabla_x^2 \mathcal{L}(\bar{z}^k)$, where $\mathcal{L}$ is the Lagrange function of P(ξ) to be defined in Section 2.4. Vector s^k can be considered as a correction term of the inconsistency between A_k and $g'(x^k)$. Vector y^k is referred to as the Lagrange multiplier. Since f and Ω are convex and H_k is symmetric positive semidefinite, the subproblem P($z^k, A_k, H_k; \xi$) is convex. Here, z^k, A_k and H_k are considered as parameters.

Remark 2.3.1. *Note that computing the term $g'(x^k)^T y^k$ of the correction vector s^k does not require the whole Jacobian matrix $g'(x^k)$, which is usually time consuming to evaluate.*

When implementing the algorithm, the evaluation of the directional derivatives $\eta^k := g'(x^k)^T y^k$ can be done by the *reverse mode* (or adjoint mode) of automatic differentiation (AD). By using this technique, we can evaluate an adjoint directional derivative vector of the form $g'(x^k)^T y^k$ without evaluating the whole Jacobian matrix $g'(x^k)$ of the vector function g. More details of AD can be found in a monograph [82] or at `http://www.autodiff.org`. Particularly, in

the NMPC framework, the constraint function g is usually obtained from a dynamic system of the form:

$$\begin{cases} \dot{\eta}(t) = G(\eta(t), x, t), \quad t_0 \le t \le t_f, \\ \eta(t_0) = \eta_0(x), \end{cases} \tag{2.3.1}$$

by applying a direct transcription, where η is referred to as a state vector and x is a parameter vector. The adjoint directional derivative vector $g'(x)^T y$ is nothing else than the gradient vector $\frac{\partial V}{\partial x}$ of the function $V(x) := g(x)^T y$. In the dynamic system context, this function V is a special case of the general functional $V(x) := e(\eta(t_f)) + \int_{t_0}^{t_f} v(\eta, x, t)dt$. By simultaneously integrating the dynamic system and its adjoint sensitivity system $\dot{\lambda} = -G_\eta^T \lambda - v_\eta^T$ and $\lambda(t_f) = \nabla_\eta e(\eta(t_f))$, we can evaluate the gradient vector of V with respect to x as $\frac{dV}{dx} := \lambda^T(t_0)\frac{\partial \eta_0}{\partial x} + \int_{t_0}^{t_f}(v_x + \lambda^T G_x)dt$, where $\lambda(t_0)$ is the solution of the adjoint system at t_0. Note that the cost of integrating the adjoint system is of the same order as integrating the forward dynamics, and crucially, independent of the dimension of x. Adjoint differentiation of dynamic systems is performed, e.g. in an open source software package, `Sundials` [169]. For more details of adjoint sensitivity analysis of dynamic systems, we refer the reader to [38, 169].

The adjoint-based predictor-corrector SCP algorithmic framework (APCSCP) is described as follows.

Algorithm 2.3.1.(*Adjoint-based predictor-corrector SCP algorithm*).
Initialization. For a given parameter $\xi_0 \in \mathcal{P}$, solve approximately (off-line) $\mathrm{P}(\xi_0)$ to get an approximate KKT point $z^0 := (x^0, y^0)$. Compute $g(x^0)$, find a matrix A_0 which approximates $g'(x^0)$ and $H_0 \in \mathcal{S}_+^n$. Then, compute vector $s^0 := \big(g'(x^0) - A_0\big)^T y^0$.
Iteration k $(k = 0, 1, \dots)$. For given z^k, A_k and H_k, perform the following steps:

Step 1. Get a new parameter value $\xi_{k+1} \in \mathcal{P}$.

Step 2. Solve the convex subproblem $\mathrm{P}(z^k, A_k, H_k; \xi_{k+1})$ to obtain a solution x^{k+1} and the corresponding multiplier y^{k+1}.

Step 3. Evaluate $g(x^{k+1})$, update (or recompute) matrices A_{k+1} and $H_{k+1} \in \mathcal{S}_+^n$. Compute vector $s^{k+1} := g'(x^{k+1})^T y^{k+1} - A_{k+1}^T y^{k+1}$. Set $k := k + 1$ and go back to Step 1.

End.
The core step of Algorithm 2.3.1 is to solve the convex subproblem $\mathrm{P}(z^k, A_k, H_k; \xi)$ at each iteration. In Algorithm 2.3.1 we do not mention

explicitly the method to solve $\mathrm{P}(z^k, A_k, H_k; \xi)$. In practice, to reduce the computational time, we can either implement an optimization method which exploits the structure of the problem, e.g. block structure, separable structure [70, 198, 221] or rely on several efficient methods and software tools that are available for convex optimization [30, 142, 148, 180, 204]. In this chapter, we are most interested in the case where one evaluation of g' is very expensive. A possible simple choice of H_k is $H_k = 0$ for all $k \geq 0$.

The initial point z^0 is obtained by solving off-line $\mathrm{P}(\xi_0)$. However, as we will show later in Corollary 2.4.1 that if we choose z^0 close to the set of KKT points $Z^*(\xi_0)$ of $\mathrm{P}(\xi_0)$ (not necessarily an exact solution) then the new approximate KKT point z^1 of $\mathrm{P}(z^0, A_0, H_0; \xi^1)$ is still close to $Z^*(\xi_1)$ of $\mathrm{P}(\xi_1)$ provided that $\|\xi_1 - \xi_0\|$ is sufficiently small. Hence, in practice, we only need to approximately solve problem $\mathrm{P}(\xi_0)$ to get a starting point z^0.

In the NMPC framework, the parameter ξ usually coincides with the initial state of the dynamic system at the current time of the moving horizon. If matrix $A_k \equiv g'(x^k)$, the exact Jacobian matrix of g at x^k and $H_k \equiv 0$, then this algorithm collapses to the *real-time SCP method* (RTSCP) considered in [191].

2.4 Contraction estimate

In this section, we will show that under certain assumptions, the sequence $\{z^k\}_{k\geq 0}$ generated by Algorithm 2.3.1 remains close to the sequence of the true KKT points $\{\bar{z}_k\}_{k\geq 0}$ of problem $\mathrm{P}(\xi)$. Without loss of generality, we assume that the objective function f is linear, i.e. $f(x) = c^T x$, where $c \in \mathbb{R}^n$ is given. Indeed, since f is convex, by using a slack variable s, we can reformulate $\mathrm{P}(\xi)$ as a nonlinear program $\min_{(x,s)} \{s \mid g(x) + M\xi = 0,\ x \in \Omega,\ f(x) \leq s\}$.

KKT condition as a generalized equation

Let us first define the Lagrange function of problem $\mathrm{P}(\xi)$, where f is linear, as follows:

$$\mathcal{L}(x, y; \xi) := c^T x + (g(x) + M\xi)^T y,$$

where y is the Lagrange multiplier associated with the constraint $g(x) + M\xi = 0$. Since the constraint $x \in \Omega$ is convex and implicitly represented, we will consider it separately. The KKT condition for $\mathrm{P}(\xi)$ is now written as:

$$\begin{cases} 0 \in c + g'(x)^T y + \mathcal{N}_\Omega(x), \\ 0 = g(x) + M\xi, \end{cases} \tag{2.4.1}$$

where $\mathcal{N}_\Omega(x)$ is the normal cone of Ω at x defined as:

$$\mathcal{N}_\Omega(x) := \begin{cases} \left\{u \in \mathbf{R}^n \mid u^T(x-v) \geq 0,\ v \in \Omega\right\}, & \text{if } x \in \Omega \\ \emptyset, & \text{otherwise.} \end{cases} \tag{2.4.2}$$

Note that the first line of (2.4.1) implicitly includes the constraint $x \in \Omega$.

A pair $(\bar{x}(\xi), \bar{y}(\xi))$ satisfying (2.4.1) is called a KKT point of $\mathrm{P}(\xi)$ and $\bar{x}(\xi)$ is called a stationary point of $\mathrm{P}(\xi)$ with the corresponding multiplier $\bar{y}(\xi)$. Let us denote by $Z^*(\xi)$ and $X^*(\xi)$ the set of KKT points and the set of stationary points of $\mathrm{P}(\xi)$, respectively. In the sequel, we use the letter z for the pair of (x, y), i.e. $z := (x^T, y^T)^T$.

Throughout this chapter, we require the following assumptions which are standard in optimization.

Asumption A.2.4.1. *The function g is twice differentiable on its domain.*

Asumption A.2.4.2. *For a given $\xi_0 \in \mathcal{P}$, problem $\mathrm{P}(\xi_0)$ has at least one KKT point $\bar{z}^0$, i.e. $Z^*(\xi_0) \neq \emptyset$.*

Our aim here is to rewrite the KKT system (2.4.1) as a generalized equation and then using the theory of generalized equations to prove a main contraction result for Algorithm 2.3.1. Let us define:

$$F(z) := \begin{pmatrix} c + g'(x)^T y \\ g(x) \end{pmatrix}, \tag{2.4.3}$$

and $K := \Omega \times \mathbf{R}^m$. Then, the KKT condition (2.4.1) can be expressed in terms of a parametric generalized equation as follows:

$$0 \in F(z) + C\xi + \mathcal{N}_K(z), \tag{2.4.4}$$

where $C := \left[\begin{smallmatrix} 0 \\ M \end{smallmatrix}\right]$. Generalized equation is an essential tool to study many problems in nonlinear analysis, perturbation analysis, variational calculations as well as optimization [28, 112, 162].

Suppose that, for some $\xi_k \in \mathcal{P}$, the set of KKT points $Z^*(\xi_k)$ of $\mathrm{P}(\xi_k)$ is nonempty. For any fixed $\bar{z}^k \in Z^*(\xi_k)$, we define the following set-valued mapping:

$$L(z; \bar{z}^k, \xi_k) := F(\bar{z}^k) + F'(\bar{z}^k)(z - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z). \tag{2.4.5}$$

We also define the inverse mapping $L^{-1} : \mathbf{R}^{n+m} \to \mathbf{R}^{n+m}$ of $L(\cdot; \bar{z}^k, \xi_k)$ as follows:

$$L^{-1}(\delta; \bar{z}^k, \xi_k) := \left\{z \in \mathbf{R}^{n+m} \mid \delta \in L(z; \bar{z}^k, \xi_k)\right\}. \tag{2.4.6}$$

Now, we consider the KKT condition of the subproblem $\mathrm{P}(z^k, A_k, H_k; \xi)$. For given neighborhoods $\mathcal{B}(\bar{z}^k, r_z)$ of $\bar{z}^k$ and $\mathcal{B}(\xi_k, r_\xi)$ of ξ_k, and $z^k \in \mathcal{B}(\bar{z}^k, r_z)$, $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$ and given matrix A_k and matrix $H_k \in \mathcal{S}_+^n$, let us consider the convex subproblem $\mathrm{P}(z^k, A_k, H_k; \xi_{k+1})$ with respect to the parameter $(z^k, A_k, H_k, \xi_{k+1})$. The KKT condition of this problem is expressed as follows:

$$\begin{cases} 0 \in c + m(z^k, A_k) + H_k(x - x^k) + A_k^T y + \mathcal{N}_\Omega(x), \\ 0 = g(x^k) + A_k(x - x^k) + M\xi_{k+1}, \end{cases} \tag{2.4.7}$$

where $\mathcal{N}_\Omega(x)$ is defined by (2.4.2). Suppose that the Slater constraint qualification holds for the subproblem $\mathrm{P}(z^k, A_k, H_k; \xi_{k+1})$, i.e.:

$$\mathrm{ri}(\Omega) \cap \left\{ x \in \mathbf{R}^n \mid g(x^k) + A_k(x - x^k) + M\xi_{k+1} = 0 \right\} \neq \emptyset,$$

where $\mathrm{ri}(\Omega)$ is the relative interior of Ω. Then by convexity of Ω, a point $z^{k+1} := (x^{k+1}, y^{k+1})$ is a KKT point of $\mathrm{P}(z^k, A_k, H_k; \xi_{k+1})$ if and only if x^{k+1} is a solution to $\mathrm{P}(z^k, A_k, H_k; \xi_{k+1})$ associated with the multiplier y^{k+1}.

Since g is twice differentiable by Assumption **A.2.4.1** and f is linear, for a given $z = (x, y)$, we have:

$$\nabla_x^2 \mathcal{L}(z) = \sum_{i=1}^m y_i \nabla^2 g_i(x), \tag{2.4.8}$$

the Hessian matrix of the Lagrange function $\mathcal{L}$, where $\nabla^2 g_i(\cdot)$ is the Hessian matrix of g_i $(i = 1, \ldots, m)$. Let us define the following matrix:

$$\tilde{F}_k' := \begin{bmatrix} H_k & A_k^T \\ A_k & 0 \end{bmatrix}, \tag{2.4.9}$$

where $H_k \in \mathcal{S}_+^n$. The KKT condition (2.4.7) can be written as a parametric linear generalized equation:

$$0 \in F(z^k) + \tilde{F}_k'(z - z^k) + C\xi_{k+1} + \mathcal{N}_K(z), \tag{2.4.10}$$

where z^k, $\tilde{F}_k'$ and ξ_{k+1} are considered as parameters. Note that if $A_k = g'(x^k)$ and $H_k = \nabla_x^2 \mathcal{L}(z^k)$ then (2.4.10) is the linearization of the nonlinear generalized equation (2.4.4) at (z^k, ξ_{k+1}) with respect to z.

Remark 2.4.1. *Note that* (2.4.10) *is a generalization of* (2.2.3)*, where the approximate Jacobian* $\tilde{F}_k'$ *is used instead of the exact one. Therefore,* (2.4.10) *can be viewed as one iteration of the inexact predictor-corrector path-following method for solving* (2.4.4)*.*

Strong regularity concept

We recall the following definition of the *strong regularity* concept. This definition can be considered as the strong regularity of the generalized equation (2.4.4) in the context of nonlinear optimization, see [160].

Definition 2.4.1. *Let $\xi_k \in \mathcal{P}$ such that the set of KKT points $Z^*(\xi_k)$ of $\mathrm{P}(\xi_k)$ is nonempty. Let $\bar{z}^k \in Z^*(\xi_k)$ be a given KKT point of $\mathrm{P}(\xi_k)$. Problem $\mathrm{P}(\xi_k)$ is said to be strongly regular at $\bar{z}^k$ if there exist neighborhoods $\mathcal{B}(0, \bar{r}_\delta)$ of the origin and $\mathcal{B}(\bar{z}^k, \bar{r}_z)$ of $\bar{z}^k$ such that the mapping $z_k^*(\delta) := \mathcal{B}(\bar{z}^k, \bar{r}_z) \cap L^{-1}(\delta; \bar{z}^k, \xi_k)$ is single-valued (i.e. the set $z_k^*(\delta)$ only contains one element) and Lipschitz continuous in $\mathcal{B}(0, \bar{r}_\delta)$ with a Lipschitz constant $0 < \gamma < +\infty$, i.e.:*

$$\|z_k^*(\delta) - z_k^*(\delta')\| \leq \gamma \|\delta - \delta'\|, \quad \forall \delta, \delta' \in \mathcal{B}(0, \bar{r}_\delta). \tag{2.4.11}$$

Note that the constants γ, $\bar{r}_z$ and $\bar{r}_\delta$ in Definition 2.4.1 are global and do not depend on the index k.

From the definition of L^{-1} where strong regularity holds, there exists a unique $z_k^*(\delta)$ such that $\delta \in F(\bar{z}^k) + F'(\bar{z}^k)(z_k^*(\delta) - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z_k^*(\delta))$. Therefore,

$$\begin{aligned} z_k^*(\delta) &\in (F'(\bar{z}^k) + \mathcal{N}_K)^{-1} \left(F'(\bar{z}^k)\bar{z}^k - F(\bar{z}^k) - C\xi_k + \delta\right) \\ &= \bar{J}_k \left(F'(\bar{z}^k)\bar{z}^k - F(\bar{z}^k) - C\xi_k + \delta\right), \end{aligned}$$

where $\bar{J}_k := (F'(\bar{z}^k) + \mathcal{N}_K)^{-1}$. The strong regularity of $\mathrm{P}(\xi)$ at $\bar{z}^k$ is equivalent to the single-valuedness and the Lipschitz continuity of $\bar{J}_k$ around $v^k := F'(\bar{z}^k)\bar{z}^k - F(\bar{z}^k) - C\xi_k$.

The strong regularity concept is widely used in variational analysis, perturbation analysis as well as in optimization [28, 112, 153, 162]. In view of optimization, strong regularity implies the *strong second order sufficient optimality condition* (SSOSC) if the *linear independence constraint qualification* (LICQ) holds [160]. If the convex set Ω is *polyhedral* and the LICQ holds, then strong regularity is equivalent to SSOSC [60]. In order to interpret the strong regularity condition of $\mathrm{P}(\xi^k)$ at $\bar{z}^k \in Z^*(\xi_k)$ in terms of perturbed optimization, we consider the following optimization problem:

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} & (c - \delta_c)^T x + \frac{1}{2}(x - \bar{x}^k)^T \nabla_x^2 \mathcal{L}(\bar{x}^k, \bar{y}^k)(x - \bar{x}^k) \\ \text{s.t.} & g(\bar{x}^k) + g'(\bar{x}^k)(x - \bar{x}^k) + M\xi_k = \delta_g, \\ & x \in \Omega. \end{cases} \tag{2.4.12}$$

Here, $\delta = (\delta_c, \delta_g) \in \mathcal{B}(0, \bar{r}_\delta)$ is a perturbation. Problem $\mathrm{P}(\xi_k)$ is *strongly regular* at $\bar{z}^k$ if and only if (2.4.12) has a unique KKT point $z_k^*(\delta)$ in $\mathcal{B}(\bar{z}^k, \bar{r}_z)$ and $z_k^*(\cdot)$ is Lipschitz continuous in $\mathcal{B}(0, \bar{r}_\delta)$ with a Lipschitz constant γ.

Example 2.4.1. Let us recall example (2.2.1) in Subsection 2.2. The optimal multipliers associated with two constraints $x_1^2+2x_2+2-4\xi = 0$ and $x_1^2-x_2^2+1 \leq 0$ are $y_1^* = (2\sqrt{\xi}-1)[8\sqrt{\xi^2-\xi\sqrt{\xi}}]^{-1} > 0$ and $y_2^* = [8\sqrt{\xi^2-\xi\sqrt{\xi}}]^{-1} > 0$, respectively. Since the last inequality constraint is active while $x \geq 0$ is inactive, we can easily compute the critical cone as $\mathcal{C}(x_\xi^*, y^*) = \{(d_1, 0) \in \mathbb{R}^2 \mid x_{\xi 1}^* d_1 = 0\}$. The Hessian matrix $\nabla_x^2 \mathcal{L}(x_\xi^*, y^*) = \begin{bmatrix} 2(y_1^*+y_2^*) & 0 \\ 0 & -2y_2^* \end{bmatrix}$ of the Lagrange function $\mathcal{L}$ is positive definite in $\mathcal{C}(x_\xi^*, y^*)$. Hence, the second order sufficient optimality condition for (2.2.1) is satisfied. Moreover, $y_2^* > 0$ which says that the strict complementarity condition holds. Therefore, problem (2.2.1) satisfies the strong second order sufficient condition. On the other hand, it is easy to check that the LICQ condition holds for (2.2.1) at x_ξ^*. By applying [160, Theorem 4.1], we can conclude that (2.2.4) is strongly regular at (x_ξ^*, y^*). ♢

The following lemma shows the nonemptiness of $Z^*(\xi)$ in the neighborhood of the parameter value ξ_k.

Lemma 2.4.1. *Suppose that Assumption **A.2.4.1** is satisfied and $Z^*(\xi_k)$ is nonempty for a given $\xi_k \in \mathcal{P}$. Suppose further that problem $\mathrm{P}(\xi_k)$ is strongly regular at $\bar{z}^k$ for a given $\bar{z}^k \in Z^*(\xi_k)$. Then there exist neighborhoods $\mathcal{B}(\xi_k, r_\xi)$ of ξ_k and $\mathcal{B}(\bar{z}^k, r_z)$ of $\bar{z}^k$ such that $Z^*(\xi_{k+1})$ is nonempty for all $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$ and $Z^*(\xi_{k+1}) \cap \mathcal{B}(\bar{z}^k, r_z)$ contains only one point $\bar{z}^{k+1}$. Moreover, there exists a constant $0 \leq \bar{\sigma} < +\infty$ such that:*

$$\left\|\bar{z}^{k+1} - \bar{z}^k\right\| \leq \bar{\sigma} \left\|\xi_{k+1} - \xi_k\right\|. \tag{2.4.13}$$

Proof. Since the KKT condition of $\mathrm{P}(\xi_k)$ is equivalent to the generalized equation (2.4.4) with $\xi = \xi_k$, by applying [160, Theorem 2.1] we conclude that there exist neighborhoods $\mathcal{B}(\xi_k, r_\xi)$ of ξ_k and $\mathcal{B}(\bar{z}^k, r_z)$ of $\bar{z}^k$ such that $Z^*(\xi_{k+1})$ is nonempty for all $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$ and $Z^*(\xi_{k+1}) \cap \mathcal{B}(\bar{z}^k, r_z)$ contains only one point $\bar{z}^{k+1}$. On the other hand, since $\left\|F(\bar{z}^k) + C\xi_k - F(\bar{z}^k) - C\xi_{k+1}\right\| = \|M(\xi_k - \xi_{k+1})\| \leq \|M\| \|\xi_{k+1} - \xi_k\|$, by using the formula [160, p. 2.4], we obtain the estimate (2.4.13). □

Contraction estimate for APCSCP with inexact Jacobian

In order to prove a contraction estimate for APCSCP, throughout this section, we make the following assumptions.

Asumption A.2.4.3. *For a given $\bar{z}^k \in Z^*(\xi_k)$, $k \geq 0$, the following conditions are satisfied:*

a) *There exists a constant* $0 \leq \kappa < \frac{1}{2\gamma}$ *such that:*

$$\left\|F'(\bar{z}^k) - \tilde{F}'_k\right\| \leq \kappa, \tag{2.4.14}$$

where $\tilde{F}'_k$ *is defined by* (2.4.9) *and* γ *is the constant in Definition 2.4.1.*

b) *The Jacobian mapping* $F'(\cdot)$ *is Lipschitz continuous on* $\mathcal{B}(\bar{z}^k, r_z)$ *around* $\bar{z}^k$*, i.e. there exists a constant* $0 \leq \omega < +\infty$ *such that:*

$$\left\|F'(z) - F'(\bar{z}^k)\right\| \leq \omega \left\|z - \bar{z}^k\right\|, \ \forall z \in \mathcal{B}(\bar{z}^k, r_z). \tag{2.4.15}$$

Note that Assumption **A.2.4.3** is commonly used in the theory of Newton-type and Gauss-Newton methods [48, 52], where the residual term is required to be sufficiently small in a neighborhood of the local solution. From the definition of $\tilde{F}'_k$ we have:

$$F'(\bar{z}^k) - \tilde{F}'_k = \begin{bmatrix} \nabla^2_x \mathcal{L}(\bar{z}^k) - H_k & g'(\bar{x}^k)^T - A_k^T \\ g'(\bar{x}^k) - A_k & O \end{bmatrix}.$$

Hence, $\left\|F'(\bar{z}^k) - \tilde{F}'_k\right\|$ depends on the norms of $\nabla^2_x \mathcal{L}(\bar{z}^k) - H_k$ and $g'(\bar{x}^k) - A_k$. These quantities are the error of the approximations H_k and A_k to the Hessian matrix $\nabla^2_x \mathcal{L}(\bar{z}^k)$ and the Jacobian matrix $g'(\bar{x}^k)$, respectively. On the one hand, Assumption **A.2.4.3**a) requires the positive definiteness of H_k to be an approximation of $\nabla^2_x \mathcal{L}$ (which is not necessarily positive definite). On the other hand, it requires that matrix A_k is a sufficiently good approximation to the Jacobian matrix g' in the neighborhood of the stationary point $\bar{x}^k$. Note that the matrix H_k in the Newton-type method proposed in [29] is not necessarily positive definite.

Now, let us define the following mapping:

$$J_k := (\tilde{F}'_k + \mathcal{N}_K)^{-1}, \tag{2.4.16}$$

where $\tilde{F}'_k$ is defined by (2.4.9). The lemma below shows that J_k is single-valued and Lipschitz continuous in a neighbourhood of $\bar{v}^k := \tilde{F}'_k \bar{z}^k - F(\bar{z}^k) - C\xi_k$. Since $J_k(v)$ is a set for a given v, the single-valuedness of J_k means that the set $J_k(v)$ only contains one element for a given v. We note that $\mathcal{N}_K$ is a maximal monotone operator [162] and $\tilde{F}'_k$ is a matrix. If $\tilde{F}'_k$ is symmetric positive definite then J_k can be considered as a generalized *resolvent operator* in the sense of Moreau–Yosida regularization [162]. In our context, the operator J_k defined by (2.4.16) imitates a similar property of the resolvent operator but locally.

Lemma 2.4.2. *Suppose that Assumptions* ***A.2.4.1****,* ***A.2.4.2*** *and* ***A.2.4.3***a) *are satisfied. Then there exist neighborhoods* $\mathcal{B}(\xi_k, r_\xi)$ *and* $\mathcal{B}(\bar{z}^k, r_z)$ *such that if we take any* $z^k \in \mathcal{B}(\bar{z}^k, r_z)$ *and* $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$ *then the mapping* J_k *defined*

by (2.4.16) is single-valued in a neighbourhood $\mathcal{B}(\bar{v}^k, r_v)$, where $\bar{v}^k := \tilde{F}'_k \bar{z}^k - F(\bar{z}^k) - C\xi_k$. Moreover, the following inequality holds:

$$\|J_k(v) - J_k(v')\| \leq \beta \|v - v'\|, \ \forall v, v' \in \mathcal{B}(\bar{v}_k, r_v), \tag{2.4.17}$$

where $\beta := \frac{\gamma}{1-\gamma\kappa} > 0$ is a Lipschitz constant.

Proof. Let us fix a neighbourhood $\mathcal{B}(\bar{v}^k, r_v)$ of $\bar{v}^k$. Suppose for contradiction that J_k is not single-valued in $\mathcal{B}(\bar{v}^k, r_v)$, then for a given v the set $J_k(v)$ contains at least two points z and z' such that $\|z - z'\| \neq 0$. We have:

$$v \in \tilde{F}'_k z + \mathcal{N}_K(z) \text{ and } v \in \tilde{F}'_k z' + \mathcal{N}_K(z'). \tag{2.4.18}$$

Let

$$\begin{aligned} &\delta := v - [\tilde{F}'_k \bar{z}^k - F(\bar{z}^k) - C\xi_k] + [F'(\bar{z}^k) - \tilde{F}'_k](z - \bar{z}^k), \\ \text{and} \\ &\delta' := v - [\tilde{F}'_k \bar{z}^k - F(\bar{z}^k) - C\xi_k] + [F'(\bar{z}^k) - \tilde{F}'_k](z' - \bar{z}^k). \end{aligned} \tag{2.4.19}$$

Then (2.4.18) can be written as:

$$\begin{aligned} &\delta \in F(\bar{z}^k) + F'(\bar{z}^k)(z - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z), \\ \text{and} \\ &\delta' \in F(\bar{z}^k) + F'(\bar{z}^k)(z' - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z'). \end{aligned} \tag{2.4.20}$$

Since v in the neighbourhood $\mathcal{B}(\bar{v}^k, r_v)$ of $\bar{v}^k := \tilde{F}'_k \bar{z}^k - F(\bar{z}^k) - C\xi_k$, we have:

$$\begin{aligned} \|\delta\| &\leq \left\|v - \bar{v}^k\right\| + \left\|[F'(\bar{z}^k) - \tilde{F}'_k](z - \bar{z}^k)\right\| \\ &\leq r_v + \left\|F'(\bar{z}^k) - \tilde{F}'_k\right\| \left\|z - \bar{z}^k\right\| \\ &\overset{(2.4.14)}{\leq} r_v + \kappa \left\|z - \bar{z}^k\right\|. \end{aligned}$$

From this inequality, we see that we can shrink $\mathcal{B}(\bar{z}^k, r_z)$ and $\mathcal{B}(\bar{v}^k, r_v)$ sufficiently small (if necessary) such that $\|\delta\| \leq \bar{r}_\delta$. Hence, $\delta \in \mathcal{B}(0, \bar{r}_\delta)$. Similarly, $\delta' \in \mathcal{B}(0, \bar{r}_\delta)$.

Now, using the strong regularity assumption of $\mathrm{P}(\xi_k)$ at $\bar{z}^k$, it follows from (2.4.20) that:

$$\|z - z'\| \leq \gamma \|\delta - \delta'\|. \tag{2.4.21}$$

However, using (2.4.19), we have:

$$\|\delta - \delta'\| = \left\|[F'(\bar{z}^k) - \tilde{F}'_k](z - z')\right\| \leq \left\|F'(\bar{z}^k) - \tilde{F}'_k\right\| \|z - z'\|$$
$$\overset{(2.4.14)}{\leq} \kappa \|z - z'\|.$$

Plugging this inequality into (2.4.21) and then using the condition $\gamma\kappa < \frac{1}{2} < 1$, we get:

$$\|z - z'\| < \|z - z'\|,$$

which contradicts to $z \neq z'$. Hence, J_k is single-valued.

Finally, we prove the Lipschitz continuity of J_k. Let $z = J_k(v)$ and $z' = J_k(v')$, where $v, v' \in \mathcal{B}(\bar{v}^k, r_v)$. Similar to (2.4.20), these expressions can be written equivalently to:

$$\begin{aligned} &\delta \in F(\bar{z}^k) + F'(\bar{z}^k)(z - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z), \\ &\text{and} \\ &\delta' \in F(\bar{z}^k) + F'(\bar{z}^k)(z' - \bar{z}^k) + C\xi_k + \mathcal{N}_K(z'), \end{aligned} \tag{2.4.22}$$

where

$$\begin{aligned} &\delta := v - [\tilde{F}'_k\bar{z}^k - F(\bar{z}^k) - C\xi_k] + [F'(\bar{z}^k) - \tilde{F}'_k](z - \bar{z}^k), \\ &\text{and} \\ &\delta' := v' - [\tilde{F}'_k\bar{z}^k - F(\bar{z}^k) - C\xi_k] + [F'(\bar{z}^k) - \tilde{F}'_k](z' - \bar{z}^k). \end{aligned} \tag{2.4.23}$$

By using again the strong regularity assumption, it follows from (2.4.22) and (2.4.23) that:

$$\begin{aligned} \|z - z'\| &\leq \gamma \|\delta - \delta'\| \\ &\leq \gamma \|v - v'\| + \gamma \left\|[F'(\bar{z}^k) - \tilde{F}'_k](z - z')\right\| \\ &\overset{(2.4.14)}{\leq} \gamma \|v - v'\| + \gamma\kappa \|z - z'\|. \end{aligned}$$

Since $\gamma\kappa < \frac{1}{2} < 1$, rearranging the last inequality we get:

$$\|z - z'\| \leq \frac{\gamma}{1 - \gamma\kappa} \|v - v'\|,$$

which shows that J_k satisfies (2.4.17) with a constant $\beta := \frac{\gamma}{1-\gamma\kappa} > 0$. □

Let us recall that if z^{k+1} is a KKT point of the convex subproblem $\mathrm{P}(z^k, A_k, H_k; \xi_{k+1})$ then

$$0 \in \tilde{F}_k'(z^{k+1} - z^k) + F(z^k) + C\xi_{k+1} + \mathcal{N}_K(z^{k+1}).$$

According to Lemma 2.4.2, if $z^k \in \mathcal{B}(\bar{z}^k, r_z)$ then problem $\mathrm{P}(z^k, A_k, H_k; \xi)$ is uniquely solvable. We can write its KKT condition equivalently as:

$$z^{k+1} = J_k\left(\tilde{F}_k' z^k - F(z^k) - C\xi_{k+1}\right). \tag{2.4.24}$$

Since $\bar{z}^{k+1}$ is the solution of (2.4.4) at ξ_{k+1}, we have $0 = F(\bar{z}^{k+1}) + C\xi_{k+1} + \bar{u}^{k+1}$, where $\bar{u}^{k+1} \in \mathcal{N}_K(\bar{z}^{k+1})$. Moreover, since $\bar{z}^{k+1} = J_k(\tilde{F}_k'\bar{z}^{k+1} + \bar{u}^{k+1})$, we can write:

$$\bar{z}^{k+1} = J_k\left(\tilde{F}_k'\bar{z}^{k+1} - F(\bar{z}^{k+1}) - C\xi_{k+1}\right). \tag{2.4.25}$$

The main result of this section is stated in the following theorem.

Theorem 2.4.2. *Suppose that Assumptions **A.2.4.1**-**A.2.4.2** are satisfied for some $\xi_0 \in \mathcal{P}$. Then, for $k \geq 0$ and $\bar{z}^k \in Z^*(\xi_k)$, if $\mathrm{P}(\xi_k)$ is strongly regular at $\bar{z}^k$ then there exist neighborhoods $\mathcal{B}(\bar{z}^k, r_z)$ and $\mathcal{B}(\xi_k, r_\xi)$ such that:*

a) The set of KKT points $Z^*(\xi_{k+1})$ of $\mathrm{P}(\xi_{k+1})$ is nonempty for any $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$.

*b) If, in addition, Assumption **A.2.4.3**a) is satisfied then the subproblem $\mathrm{P}(z^k, A_k, H_k; \xi_{k+1})$ is uniquely solvable in the neighborhood $\mathcal{B}(\bar{z}^k, r_z)$.*

*c) Moreover, if, in addition, Assumption **A.2.4.3**b) is satisfied then the sequence $\{z^k\}_{k\geq 0}$ generated by Algorithm 2.3.1, where $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$, guarantees:*

$$\begin{aligned}\left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq \left(\alpha + c_1\left\|z^k - \bar{z}^k\right\|\right)\left\|z^k - \bar{z}^k\right\| \\ + \left(c_2 + c_3\left\|\xi_{k+1} - \xi_k\right\|\right)\left\|\xi_{k+1} - \xi_k\right\|,\end{aligned} \tag{2.4.26}$$

where $0 \leq \alpha < 1$, $0 \leq c_i < +\infty$, $i = 1, \ldots, 3$ and $c_2 > 0$ are given constants and $\bar{z}^{k+1} \in Z^*(\xi_{k+1})$.

Proof. We prove the theorem by induction. For $k = 0$, we have $Z^*(\xi_0)$ is nonempty by Assumption **A.2.4.2**. Now, we assume $Z^*(\xi_k)$ is nonempty for some $k \geq 0$. We will prove that $Z^*(\xi_{k+1})$ is nonempty for some $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$, a neighborhood of ξ_k.

Indeed, since $Z^*(\xi_k)$ is nonempty for some $\xi_k \in \mathcal{P}$, we take an arbitrary $\bar{z}^k \in Z^*(\xi_k)$ such that $\mathrm{P}(\xi_k)$ is strong regular at $\bar{z}^k$. Now, by applying Lemma

2.4.1 to problem $\mathrm{P}(\xi_k)$, then we conclude that there exist neighborhoods $\mathcal{B}(\bar{z}^k, r_z)$ of $\bar{z}^k$ and $\mathcal{B}(\xi_k, r_\xi)$ of ξ_k such that $Z^*(\xi_{k+1})$ is nonempty for any $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$.

Next, if, in addition, Assumption **A.2.4.3**a) holds then the conclusions of Lemma 2.4.2 hold. By induction, we conclude that the convex subproblem $\mathrm{P}(\bar{z}^k, A_k, \xi_k)$ is uniquely solvable in $\mathcal{B}(\bar{z}^k, r_z)$ for any $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$.

Finally, we prove inequality (2.4.26). From (2.4.24), (2.4.25) and the Lipschitz continuity of J_k in (2.4.17), we have:

$$\begin{aligned}
\left\|z^{k+1} - \bar{z}^{k+1}\right\| &\overset{(2.4.24)}{=} \left\|J_k\left((\tilde{F}'_k z^k - F(z^k) - C\xi_{k+1}\right) - \bar{z}^{k+1}\right\| \\
&\overset{(2.4.25)}{=} \left\|J_k\left(\tilde{F}'_k z^k - F(z^k) - C\xi_{k+1}\right) - J_k\left(\tilde{F}'_k \bar{z}^{k+1} - F(\bar{z}^{k+1}) - C\xi_{k+1}\right)\right\| \\
&\overset{(2.4.17)}{\leq} \beta\left\|\tilde{F}'_k(z^k - \bar{z}^{k+1}) - F(z^k) + F(\bar{z}^{k+1})\right\| \qquad (2.4.27) \\
&= \beta\Big\| \left[\tilde{F}'_k(z^k - \bar{z}^k) - F(z^k) + F(\bar{z}^k)\right] + \left[F(\bar{z}^{k+1}) - F(\bar{z}^k) - \tilde{F}'_k(\bar{z}^{k+1} - \bar{z}^k)\right] \Big\|.
\end{aligned}$$

By using the mean-value theorem and Assumption **A.2.4.3**b), we further estimate (2.4.27) as:

$$\begin{aligned}
\left\|z^{k+1} - \bar{z}^{k+1}\right\| &\leq \beta\Big\|[\tilde{F}'_k - F'(\bar{z}^k)](z^k - \bar{z}^k) \\
&- \int_0^1 [F'(\bar{z}^k + t(z^k - \bar{z}^k)) - F'(\bar{z}^k)](z^k - \bar{z}^k)dt\Big\| \\
&+ \beta\Big\|[\tilde{F}'_k - F'(\bar{z}^k)](\bar{z}^{k+1} - \bar{z}^k) - \int_0^1 [F'(\bar{z}^k + t(\bar{z}^{k+1} - \bar{z}^k)) - F'(\bar{z}^k)](z^{k+1} - \bar{z}^k)dt\Big\| \\
&\overset{(2.4.14)+(2.4.15)}{\leq} \beta\left(\kappa + \frac{\omega}{2}\left\|z^k - \bar{z}^k\right\|\right)\left\|z^k - \bar{z}^k\right\| \\
&+ \beta\left(\kappa + \frac{\omega}{2}\left\|\bar{z}^{k+1} - \bar{z}^k\right\|\right)\left\|\bar{z}^{k+1} - \bar{z}^k\right\|. \qquad (2.4.28)
\end{aligned}$$

By substituting (2.4.13) into (2.4.28) we obtain:

$$\begin{aligned}
\left\|z^{k+1} - \bar{z}^{k+1}\right\| &\leq \beta\left(\kappa + \frac{\omega}{2}\left\|z^k - \bar{z}^k\right\|\right)\left\|z^k - \bar{z}^k\right\| \\
&+ \beta\left(\kappa\bar{\sigma} + \frac{\omega\bar{\sigma}^2}{2}\left\|\xi_{k+1} - \xi_k\right\|\right)\left\|\xi_{k+1} - \xi_k\right\|.
\end{aligned}$$

If we define $\alpha := \beta\kappa = \frac{\gamma\kappa}{1-\gamma\kappa} < 1$ due to **A.2.4.3**a), $c_1 := \frac{\gamma\omega}{2(1-\gamma\kappa)} \geq 0$, $c_2 := \frac{\gamma\kappa\bar{\sigma}}{1-\gamma\kappa} > 0$ and $c_3 := \frac{\gamma\omega\bar{\sigma}^2}{2(1-\gamma\kappa)} \geq 0$ as four given constants then the last inequality is indeed (2.4.26). □

The following corollary shows the stability of the approximate sequence $\{z^k\}_{k\geq 0}$ generated by Algorithm 2.3.1.

Corollary 2.4.1. *Under the assumptions of Theorem 2.4.2, there exists a positive number $0 < r_z < \bar{r}_z := (1-\alpha)c_1^{-1}$ such that if the initial point z^0 in Algorithm 2.3.1 is chosen such that $\left\|z^0 - \bar{z}^0\right\| \leq r_z$, where $\bar{z}^0 \in Z^*(\xi_0)$ then, for any $k \geq 0$, we have:*

$$\left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq r_z, \tag{2.4.29}$$

provided that $\|\xi_{k+1} - \xi_k\| \leq r_\xi$, where $\bar{z}^{k+1} \in Z^*(\xi_{k+1})$ and $0 < r_\xi \leq \bar{r}_\xi$ with:

$$\bar{r}_\xi := \begin{cases} (2c_3)^{-1}\left[\sqrt{c_2^2 + 4c_3 r_z(1-\alpha-c_1 r_z)} - c_2\right] & \text{if } c_3 > 0, \\ c_2^{-1} r_z(1-\alpha-c_1 r_z) & \text{if } c_3 = 0. \end{cases}$$

Consequently, the error sequence $\{e_k\}_{k\geq 0}$, where $e_k := \left\|z^k - \bar{z}^k\right\|$, between the exact KKT point $\bar{z}^k$ and the approximate KKT point z^k of $\mathrm{P}(\xi_k)$ is nonincreasing and therefore bounded.

Proof. Since $0 \leq \alpha < 1$, we have $\bar{r}_z := (1-\alpha)c_1^{-1} > 0$. Let us choose r_z such that $0 < r_z < \bar{r}_z$. If $z^0 \in \mathcal{B}(\bar{z}^0, r_z)$, i.e. $\left\|z^0 - \bar{z}^0\right\| \leq r_z$, then it follows from (2.4.26) that:

$$\left\|z^1 - \bar{z}^1\right\| \leq (\alpha + c_1 r_z) r_z + (c_2 + c_3 \|\xi_1 - \xi_0\|) \|\xi_1 - \xi_0\|.$$

In order to ensure $\left\|z^1 - \bar{z}^1\right\| \leq r_z$, we need $(c_2 + c_3\|\xi_1 - \xi_0\|)\|\xi_1 - \xi_0\| \leq \rho := (1-\alpha-c_1 r_z) r_z$. Since $0 < r_z < \bar{r}_z$, $\rho > 0$. The last condition leads to $\|\xi_1 - \xi_0\| \leq (2c_3)^{-1}(\sqrt{c_2^2 + 4c_3\rho} - c_2)$ if $c_3 > 0$ and $\|\xi_1 - \xi_0\| \leq c_2^{-1} r_z(1-\alpha-c_1 r_z)$ if $c_3 = 0$. By induction, we conclude that inequality (2.4.29) holds for all $k \geq 0$. The nonincrease of $\{e_k\}$ follows directly from the estimate (2.4.29). □

The conclusion of Corollary 2.4.1 is illustrated in Figure 2.3, where the approximate sequence $\{z^k\}_{k\geq 0}$ computed by Algorithm 2.3.1 remains close to the sequence of the true KKT points $\{\bar{z}^k\}_{k\geq 0}$ if the starting point z^0 is sufficiently close to $\bar{z}_0$. Let us assume that the constant $\omega > 0$. Then we have $c_3 > 0$. If we choose $r_z := \frac{\bar{r}_z}{2} = \frac{1-2\gamma\kappa}{\gamma\omega}$ then the quantity $\bar{r}_\xi$ in Corollary 2.4.1 can be tightened to $\bar{r}_\xi = \left\{\gamma\omega\bar{\sigma}\left[\frac{\gamma\kappa}{(1-2\gamma\kappa)^2} + \frac{\bar{\sigma}}{1-\gamma\kappa}\right]\right\}^{-1}$.

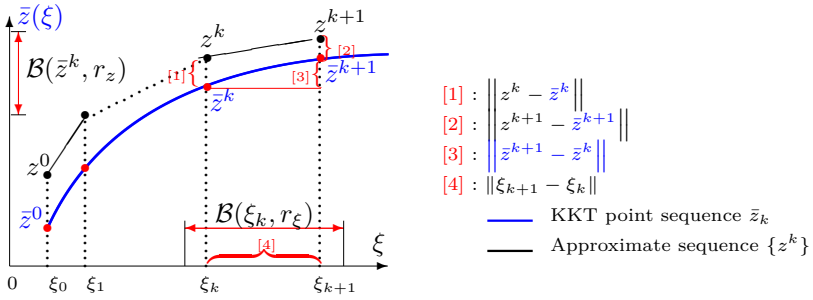


Figure 2.3: The approximate sequence $\{z^k\}_{k\geq 0}$ along the trajectory $\bar{z}(\cdot)$ of the KKT points.

We can also simplify the contraction estimate (2.4.26) as follows:

$$\left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq \nu \left\|z^k - \bar{z}^k\right\| + c \left\|\xi_{k+1} - \xi_k\right\|, \tag{2.4.30}$$

where $\nu := \alpha + c_1 r_z > 0$ and $c := c_2 + c_3 r_\xi > 0$. Since $\alpha \in (0,1)$, we can choose $r_z > 0$ sufficiently small such that $\nu \in (0,1)$.

Remark 2.4.3. *In Algorithm 2.3.1, instead of performing one SCP iteration for each value ξ_k of the parameter ξ, we can perform p SCP iterations ($p \geq 1$). In this case, we obtain the following contraction estimate:*

$$\left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq \nu^p \left\|z^k - \bar{z}^k\right\| + \nu^{p-1} c \left\|\xi_{k+1} - \xi_k\right\|. \tag{2.4.31}$$

Indeed, at the value ξ_k, we can apply (2.4.30) p times with conditions $\xi_k = \xi_{k+1} = \cdots = \xi_{k+p-1}$ and then we move to the next value ξ_{k+1} to obtain (2.4.30).

Contraction estimate for APCSCP with exact Jacobian

If $A_k \equiv g'(x^k)$ then the correction vector $s^k = 0$ and the convex subproblem $\mathrm{P}(z^k, A_k, H_k; \xi)$ collapses to the following one:

$$\left\{\begin{array}{ll} \min\limits_{x\in\mathbf{R}^n} & \left\{c^T x + \frac{1}{2}(x - x^k)^T H_k (x - x^k)\right\} \\ \text{s.t.} & g(x^k) + g'(x^k)(x - x^k) + M\xi = 0, \\ & x \in \Omega. \end{array}\right. \qquad \mathrm{P}(x^k, H_k; \xi)$$

Note that problem $\mathrm{P}(x^k, H_k; \xi)$ does not depend on the multiplier y^k if we choose H_k independently of y^k. We refer to a variant of Algorithm 2.3.1 where we use the convex subproblem $\mathrm{P}(x^k, H_k; \xi)$ instead of $\mathrm{P}(z^k, A_k, H_k; \xi)$ as a *predictor-corrector SCP algorithm* (PCSCP) for solving a sequence of the optimization problems $\{\mathrm{P}(\xi_k)\}_{k\geq 0}$.

Instead of Assumption **A.2.4.3**a) in the previous section, we make the following assumption.

A2.4.3'. *There exists a constant* $0 \leq \tilde{\kappa} < \frac{1}{2\gamma}$ *such that*

$$\left\|\nabla_x^2 \mathcal{L}(\bar{z}^k) - H_k\right\| \leq \tilde{\kappa}, \ \forall k \geq 0. \tag{2.4.32}$$

where $\nabla_x^2 \mathcal{L}(z)$ *defined by* (2.4.8).

Assumption **A.2.4.3'** requires that the approximation H_k to the Hessian matrix $\nabla_x^2 \mathcal{L}(\bar{z}^k)$ of the Lagrange function $\mathcal{L}$ at $\bar{z}^k$ is sufficiently close. Note that matrix H_k in the framework of the SSDP method in [44] is not necessarily positive definite.

Example 2.4.2. Let us continue analyzing example (2.2.4). The Hessian matrix of the Lagrange function $\mathcal{L}$ associated with the equality constraint $x_1^2 + 2x_2 + 2 - 4\xi = 0$ is $\nabla_x^2 \mathcal{L}(x_\xi^*, y_1^*) = \begin{bmatrix} 2y_1^* & 0 \\ 0 & 0 \end{bmatrix}$, where y_1^* is the multiplier associated with the equality constraint at x_ξ^*. Let us choose a positive semidefinite matrix $H_k := \begin{bmatrix} h_{11} & 0 \\ 0 & 0 \end{bmatrix}$, where $h_{11} \geq 0$, then $\left\|\nabla_x^2 \mathcal{L}(x_\xi^*, y_1^*) - H_k\right\| = |y_1^* - h_{11}|$. Since $y_1^* \geq 0$, for an arbitrary $\tilde{\kappa} > 0$, we can choose $h_{11} \geq 0$ such that $|h_{11} - y_1^*| \leq \tilde{\kappa}$. Consequently, the condition (2.4.32) is satisfied. In Example 2.2.4 of Subsection 2.2, we choose $h_{11} = 0$. ♢

The following theorem shows the same conclusions as in Theorem 2.4.2 and Corollary 2.4.1 for the *predictor-corrector SCP algorithm*.

Theorem 2.4.4. *Suppose that Assumptions* ***A.2.4.1*-*A.2.4.2*** *are satisfied for some* $\xi_0 \in \mathcal{P}$. *Then, for* $k \geq 0$ *and* $\bar{z}^k \in Z^*(\xi_k)$, *if* $\mathrm{P}(\xi_k)$ *is strongly regular at* $\bar{z}^k$ *then there exist neighborhoods* $\mathcal{B}(\bar{z}^k, r_z)$ *and* $\mathcal{B}(\xi_k, r_\xi)$ *such that:*

a) The set of KKT points $Z^*(\xi_{k+1})$ *of* $\mathrm{P}(\xi_{k+1})$ *is nonempty for any* $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$.

b) If, in addition, Assumption ***A.2.4.3'*** *is satisfied then the subproblem* $\mathrm{P}(x^k, H_k; \xi_{k+1})$ *is uniquely solvable in the neighborhood* $\mathcal{B}(\bar{z}^k, r_z)$.

c) Moreover, if, in addition, Assumption ***A.2.4.3***b) *holds then the sequence* $\{z^k\}_{k\geq 0}$ *generated by the PCSCP algorithm, where* $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$, *guarantees:*

$$\begin{aligned} \left\|z^{k+1} - \bar{z}^{k+1}\right\| \leq & \left(\tilde{\alpha} + \tilde{c}_1 \left\|z^k - \bar{z}^k\right\|\right) \left\|z^k - \bar{z}^k\right\| \\ & + \left(\tilde{c}_2 + \tilde{c}_3 \left\|\xi_{k+1} - \xi_k\right\|\right) \left\|\xi_{k+1} - \xi_k\right\|, \end{aligned} \tag{2.4.33}$$

where $0 \leq \tilde{\alpha} < 1$, $0 \leq \tilde{c}_i < +\infty$, $i = 1, \cdots, 3$ *and* $\tilde{c}_2 > 0$ *are given constants and* $\bar{z}^{k+1} \in Z^*(\xi_{k+1})$.

d) If the initial point z^0 in the PCSCP algorithm is chosen such that $\|z^0 - \bar{z}^0\| \le \tilde{r}_z$, where $\bar{z}^0 \in Z^*(\xi_0)$ and $0 < \tilde{r}_z < \bar{\tilde{r}}_z := \tilde{c}_1^{-1}(1-\tilde{\alpha})$, then:

$$\|z^{k+1} - \bar{z}^{k+1}\| \le \tilde{r}_z, \tag{2.4.34}$$

provided that $\|\xi_{k+1} - \xi_k\| \le \tilde{r}_\xi$ with $0 < \tilde{r}_\xi \le \bar{\tilde{r}}_\xi$,

$$\bar{\tilde{r}}_\xi := \begin{cases} (2\tilde{c}_3)^{-1}\left[\sqrt{\tilde{c}_2^2 + 4\tilde{c}_3\tilde{r}_z(1-\tilde{\alpha}-\tilde{c}_1\tilde{r}_z)} - \tilde{c}_2\right] & \text{if } \tilde{c}_3 > 0, \\ \tilde{c}_2^{-1}\tilde{r}_z(1-\tilde{\alpha}-\tilde{c}_1\tilde{r}_z) & \text{if } \tilde{c}_3 = 0. \end{cases}$$

Consequently, the error sequence $\{\|z^k - \bar{z}^k\|\}_{k\ge 0}$ between the exact KKT point $\bar{z}^k$ and the approximation KKT point z^k of $\mathrm{P}(\xi_k)$ is still nonincreasing and therefore bounded.

Proof. The statement a) of Theorem 2.4.4 follows from Theorem 2.4.2. We prove b). Since $A_k \equiv g'(x^k)$, the matrix $\hat{\tilde{F}}'_k$ defined in (2.4.9) becomes:

$$\hat{\tilde{F}}'_k := \begin{bmatrix} H_k & g'(x^k) \\ g'(x^k) & 0 \end{bmatrix},$$

Moreover, since g is twice differentiable due to Assumption **A.2.4.1**, g' is Lipschitz continuous with a Lipschitz constant $L_g \ge 0$ in $\mathcal{B}(\bar{x}^k, r_z)$. Therefore, by Assumption **A.2.4.3'**, we have:

$$\begin{aligned} \left\|F'(\bar{z}^k) - \hat{\tilde{F}}'_k\right\|^2 &= \left\| \begin{bmatrix} \nabla_x^2\mathcal{L}(\bar{z}^k) & g'(\bar{x}^k)^T - g'(x^k)^T \\ g'(\bar{x}^k) - g'(x^k) & 0 \end{bmatrix} \right\|^2 \\ &\le \left\|\nabla_x^2\mathcal{L}(\bar{z}^k) - H_k\right\|^2 + 2\left\|g'(x^k) - g'(\bar{x}^k)\right\|^2 \qquad (2.4.35) \\ &\le \tilde{\kappa}^2 + 2L_g^2\left\|x^k - \bar{x}^k\right\|^2. \end{aligned}$$

Since $\tilde{\kappa}\gamma < \frac{1}{2}$, we can shrink $\mathcal{B}(\bar{z}^k, r_z)$ sufficiently small such that:

$$\gamma(\tilde{\kappa}^2 + 2L_g^2 r_z^2)^{1/2} < \frac{1}{2}.$$

If we define $\tilde{\kappa}_1 := (\tilde{\kappa}^2 + 2L_g^2 r_z^2)^{1/2} \ge 0$ then the last inequality and (2.4.35) imply:

$$\left\|F'(\bar{z}^k) - \hat{\tilde{F}}'_k\right\| \le \tilde{\kappa}_1, \tag{2.4.36}$$

where $\tilde{\kappa}_1\gamma < \frac{1}{2}$. Similar to the proof of Lemma 2.4.2, we can show that the mapping $\hat{J}_k := (\hat{\tilde{F}}'_k + \mathcal{N}_K)^{-1}$ is single-valued and Lipschitz continuous with a Lipschitz constant $\tilde{\beta} := \gamma(1-\gamma\tilde{\kappa}_1)^{-1} > 0$ in $\mathcal{B}(\bar{z}^k, r_z)$. Consequently, the convex

problem $\mathrm{P}(x^k, H_k; \xi_{k+1})$ is uniquely solvable in $\mathcal{B}(\bar{z}^k, r_z)$ for all $\xi_{k+1} \in \mathcal{B}(\xi_k, r_\xi)$, which proves b).

With the same argument as the proof of Theorem 2.4.2, we can also prove the following estimate:

$$\left\|z^{k+1} - \bar{z}^k\right\| \leq \left(\tilde{\alpha}_k + \tilde{c}_1 \left\|z^k - \bar{z}^k\right\|\right) \left\|z^k - \bar{z}^k\right\| + \left(\tilde{c}_2 + \tilde{c}_3 \left\|\xi_{k+1} - \xi_k\right\|\right) \left\|\xi_{k+1} - \xi_k\right\|,$$

where $\tilde{\alpha} := \gamma\tilde{\kappa}_1(1 - \gamma\tilde{\kappa}_1)^{-1} \in [0, 1)$, $\tilde{c}_1 := \gamma\omega(2 - 2\gamma\tilde{\kappa}_1)^{-1} \geq 0$, $\tilde{c}_2 := \gamma\tilde{\kappa}_1\bar{\sigma}(1 - 1\gamma\tilde{\kappa}_1)^{-1} > 0$ and $\tilde{c}_3 := \gamma\omega\bar{\sigma}^2(2 - 2\gamma\tilde{\kappa}_1)^{-1} \geq 0$. The remaining statements of Theorem 2.4.4 are proved similarly to the proofs of Theorem 2.4.2 and Corollary 2.4.1. □

Similar to Corollary 2.4.1, the constant $\bar{r}_\xi$ in the statement d) of this theorem can be simplified to $\bar{\tilde{r}}_\xi = \left\{\gamma\omega\bar{\sigma}\left[\frac{\gamma\tilde{\kappa}_1}{(1-2\gamma\tilde{\kappa}_1)^2} + \frac{\bar{\sigma}}{1-\gamma\tilde{\kappa}_1}\right]\right\}^{-1}$.

Choices of matrix A_k

In the adjoint-based predictor-corrector SCP algorithm, an approximate matrix A_k of $g'(x^k)$ and a vector $s^k = (g'(x^k) - A_k)^T y^k$ are required at each iteration such that they maintain Assumption **A.2.4.3**. This matrix needs to be obtained in a cost efficient way, but shall also provide a sufficiently good approximation of $g'(x^k)$. These are conflicting objectives. Though not the subject of this chapter, let us discuss some ways to obtain A_k. First, it might be the exact Jacobian matrix, the most expensive option. Second, it might be computed by a user provided approximation algorithm, e.g. based on inaccurate differential equation solutions. Third, suppose that an initial approximation A_0 is known. For given z^k and A_k, $k \geq 0$, we need to compute A_{k+1} and s^{k+1} in an efficient way. If $\left\|A_k - g'(\bar{x}^{k+1})\right\|$ is still small then we can even use the same matrix A_k for the next iteration, i.e. $A_{k+1} = A_k$ due to Assumption **A.2.4.3**. Otherwise, matrix A_{k+1} can be constructed, e.g. by using low-rank updates. We can e.g. use the two sided rank-1 updates (TR1) [58, 84] or the Broyden formulas [167]. However, it is important to note that the use of the low-rank update for matrix A_k might destroy possible sparsity structure of matrix A_k. Then high-rank updates might be an option [27, 83].

In Algorithm 2.3.1 we can set matrix $H_k = 0$ for all $k \geq 0$. However, this matrix can alternatively be updated at each iteration by using BFGS-type formulas or the projection of $\nabla_x^2 \mathcal{L}(z^k)$ onto $\mathcal{S}_+^n$.

2.5 Applications in nonlinear programming

If the set of parameters Σ collapses to one point, i.e. $\Sigma := \{\xi\}$ then, without loss of generality, we assume that $\xi = 0$ and problem P(ξ) reduces to a nonlinear programming problem of the form:

$$\begin{cases} \min\limits_{x\in\mathbf{R}^n} & f(x) := c^T x \\ \text{s.t.} & g(x) = 0, \\ & x \in \Omega, \end{cases} \tag{P}$$

where c, g and Ω are as in P(ξ). In this section we describe a local optimization algorithm for solving (P) that is a special case of the APCSCP method.

The subproblem P($z^k, A_k, H_k; \xi$) in Algorithm 2.3.1 reduces to:

$$\begin{cases} \min\limits_{x\in\mathbf{R}^n} & c^T x + (s^j)^T(x - x^j) + \frac{1}{2}(x - x^j)^T H_j(x - x^j) \\ \text{s.t.} & g(x^j) + A_j(x - x^j) = 0, \\ & x \in \Omega. \end{cases} \qquad \text{P}(z^j, A_j, H_j)$$

Here, we use the index j in the algorithm for the nonparametric problem (see below) to distinguish it from the index k in the parametric case. Moreover, matrix H_j is chosen such that $H_j \in \mathcal{S}^n_+$ as in the APCSCP method.

In order to apply the theory in the previous sections, we only consider the full-step algorithm for solving (P) which we call *full-step adjoint-based sequential convex programming* (FASCP). It is described as follows:

Algorithm 2.5.1.(*Full-step adjoint-based SCP algorithm*).
Initialization. Find an initial guess $x^0 \in \Omega$ and $y^0 \in \mathbf{R}^m$, a matrix A_0 approximating $g'(x^0)$ and $H_0 \in \mathcal{S}^n_+$. Set $s^0 := (g'(x^0) - A_0)^T y^0$.
Iteration j **(**$j = 0, 1, 2, \cdots$**).** For given z^j, A_j and H_j, perform Steps 1-3:

Step 1. Solve the convex subproblem P(z^j, A_j, H_j) to obtain a solution x_t^{j+1} and the corresponding multiplier y^{j+1}.

Step 2. If $\left\| x_t^{j+1} - x^j \right\| \le \varepsilon$, for a given tolerance $\varepsilon > 0$, then terminate. Otherwise, compute the search direction $\Delta x^j := x_t^{j+1} - x^j$.

Step 3. Update $x^{j+1} := x^j + \Delta x^j$. Evaluate the function value $g(x^{j+1})$, update (or recompute) matrices A_{j+1} and $H_{j+1} \in \mathcal{S}^n_+$ (if necessary) and the correction vector s^{j+1}.

End.

The following corollary shows that the *full-step adjoint-based SCP algorithm* generates an iterative sequence that converges linearly to a KKT point of (P).

Corollary 2.5.1. *Let $\hat{Z}^* \neq \emptyset$ and $\hat{z}^* \in \hat{Z}^*$. Suppose that Assumption **A.2.4.1** holds and that problem (P) is strongly regular at $\hat{z}^*$ (in the sense of Definition 2.4.1). Suppose further that Assumption **A.2.4.3**a) is satisfied in $\mathcal{B}(\hat{z}^*, \hat{r}_z)$. Then there exists a neighborhood $\mathcal{B}(\hat{z}^*, r_z)$ of $\hat{z}^*$ such that, in this neighborhood, the convex subproblem $\mathrm{P}(x^j, A_j, H_j)$ has a unique KKT point z^{j+1} for any $z^j \in \mathcal{B}(\hat{z}^*, r_z)$. Moreover, if, in addition, Assumption **A.2.4.3**b) holds then the sequence $\{z^j\}_{j\geq 0}$ generated by Algorithm 2.5.1 starting from $z^0 \in \mathcal{B}(\hat{z}^*, r_z)$ satisfies:*

$$\left\|z^{j+1} - \hat{z}^*\right\| \leq (\hat{\alpha} + \hat{c}_1 \left\|z^j - \hat{z}^*\right\|) \left\|z^j - \hat{z}^*\right\|, \ \forall j \geq 0, \tag{2.5.1}$$

where $0 \leq \hat{\alpha} < 1$ and $0 \leq \hat{c}_1 < +\infty$ are given constants. Consequently, this sequence converges linearly to $\hat{z}^*$, the unique KKT point of (P) in $\mathcal{B}(\hat{z}^*, r_z)$.

Proof. The estimate (2.5.1) follows directly from Theorem 2.4.2 by taking $\xi_k = 0$ for all k. The remaining statement is a consequence of (2.5.1). □

If $A_j = g'(x^j)$ then Algorithm 2.5.1 collapses to the *full-step SCP algorithm* considered in [188]. The local convergence of this variant follows similarly from Theorem 2.4.4.

Remark 2.5.1. *The adjoint-based variant, Algorithm 2.5.1, is a generalization of the SSDP methods in [44, 65] or the SQP method presented in [106] when the subproblems of the form $\mathrm{P}(z^j, A_j, H_j)$ are convex.*

2.6 Conclusion

In this chapter, we have proposed a generic algorithmic framework which we call *adjoint-based predictor-corrector sequential convex programming* to treat parametric optimization problems. This method is a combination of three ingredients, namely sequential convex programming, predictor-corrector path-following and adjoint-based optimization. Under the strong regularity assumption, Assumption **A.2.4.3**a), and Assumption **A.2.4.3**b) we have proved that the tracking errors between the true KKT points of problem $\mathrm{P}(\xi)$ and the approximate ones provided by the algorithm are nonincreasing and therefore bounded. While the *strong regularity* concept is standard in optimization and nonlinear analysis, the two last assumptions are needed in any Newton-type algorithm. The main advantage of this algorithm is that it is suitable to treat nonlinear model predictive control applications which contain certain general

convex constraints and may have expensive sensitivity evaluations. When the exact Jacobian of the constraint function is used, we obtain a variant of APCSCP which we call PCSCP. The first algorithm has been specified to the nonparametric case to obtain a full-step adjoint-based SCP method for solving nonconvex programming problems. The local linear convergence of this algorithm is an immediate consequence of the general contraction theorem.

Chapter 3

SCP applications in optimal control

Optimal control is one main area where optimization algorithms can be of benefit. The problem obtained from any direct transcription of an optimal control problem is a finite dimensional optimization problem. In other words, the core procedure in the numerical solution of an optimal control problem is an optimization algorithm. In model predictive control, methods based on optimization techniques also require one to solve at each sampling time an optimization problem to calculate a feedback for the next sampling time. The aim of this chapter is to test the performance of the two algorithms, Algorithms 2.3.1 and 2.5.1, presented in Chapter 2 for solving a nonlinear model predictive control problem as well as an optimal control problem, respectively.

3.1 NMPC of a hydro power plant

In this section, a nonlinear model predictive control (NMPC) problem of a hydro power valley (HPV) is considered. We focus on tracking the steady state of the dynamic system under the effect of uncertainties in some input parameters. The full model of this problem was published in [166] as a benchmark problem. We first apply the multiple shooting method [27] to transform the optimal control at each time interval into a large-scale parametric optimization problem. Then Algorithm 2.3.1 in Chapter 2 is applied to solve this parametric optimization

problem. We note that this problem possesses a quadratic constraint which can be treated directly in Algorithm 2.3.1 compared to conventional approaches.

Dynamic model

We consider a hydro power plant composed of several subsystems connected together. The system includes six dams with turbines D_i $(i = 1, \ldots, 6)$ located along a river and three lakes L_1, L_2 and L_3 as visualized in Fig. 3.1. Here, U_1 is a duct connecting lakes L_1 and L_2; T_1 and T_2 are ducts equipped with turbines and C_1 and C_2 are ducts equipped with turbines and pumps. The flows through the turbines and pumps are the controlled variables. The complete model with all the parameters can be found in [166].

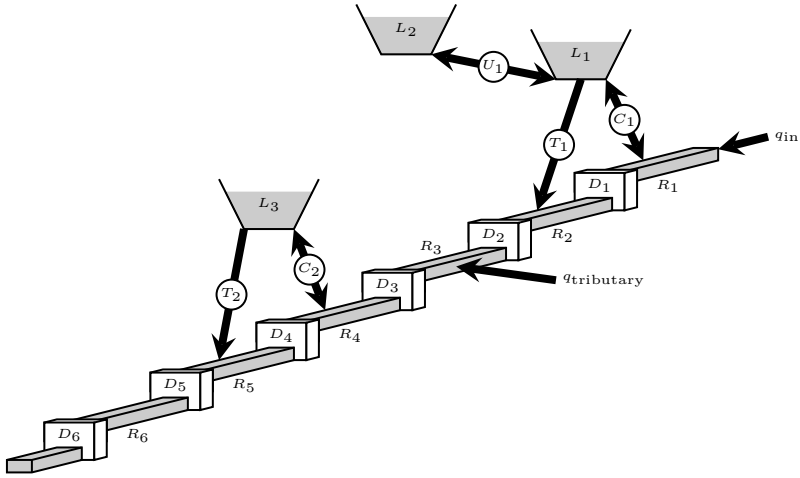


Figure 3.1: Overview of the hydro power plant.

The dynamics of the lakes is given by:

$$\frac{\partial h(t)}{\partial t} = \frac{q_{\text{in}}(t) - q_{\text{out}}(t)}{S}, \tag{3.1.1}$$

where $h(t)$ is the water level and S is the surface area of the lakes; q_{in} and q_{out} are the input and output flows, respectively. The dynamics of the reaches R_i $(i = 1, \ldots, 6)$ is described by the one-dimensional Saint-Venant partial differential equation:

$$\begin{cases} \frac{\partial q(t,y)}{\partial y} + \frac{\partial s(t,y)}{\partial t} = q_l(t), \\ \frac{1}{g}\frac{\partial}{\partial t}\left(\frac{q(t,y)}{s(t,y)}\right) + \frac{1}{2g}\frac{\partial}{\partial y}\left(\frac{q^2(t,y)}{s^2(t,y)}\right) + \frac{\partial h(t,y)}{\partial y} + I_f(t,y) - I_0(y) = 0. \end{cases} \tag{3.1.2}$$

Here, y is the spatial variable along the flow direction of the river, $q(\cdot,\cdot)$ is the river flow (or discharge), $s(\cdot,\cdot)$ is the wetted surface, $h(\cdot,\cdot)$ is the water level with respect to the river bed, g is the gravitation acceleration, I_f is the friction slope, I_0 is the river bed slope, and q_l is the lateral inflow per space unit. The relation between s and h is given by $h(t,y) = w_d(y)s(t,y)$, where $w_d(y)$ is the river width. Note that, by using the first equation of (3.1.2) with the condition $q_l(t) \equiv 0$, we can simplify the second equation of (3.1.2) as follows:

$$\frac{\partial q}{\partial t} = -\frac{q}{w_d h}\frac{\partial q}{\partial y} + g w_d h(I_0 - I_f) + \left(\frac{1}{w_d}\frac{q^2}{h^2} - g w_d h\right)\frac{\partial h}{\partial y}. \tag{3.1.3}$$

We first discretize the first equation of (3.1.2) and (3.1.3) and compute the steady states of these equations by fixing some parameters. The obtained steady states will be used as the functions for the initial conditions of q and h at $t = 0$. We refer to [166] for more details.

The partial differential equations (3.1.2)-(3.1.3) can be discretized by applying the method of lines in order to obtain a system of ordinary differential equations. Stacking all the equations together, we represent the dynamics of the system by:

$$\dot{w}(t) = f(w,u), \quad w(t_0) = w_0, \tag{3.1.4}$$

where the state vector $w \in R^{n_w}$ includes all the flows and the water levels, $u \in R^{n_u}$ represents the input vector and w_0 is a given initial state. The dynamic system consists of $n_w = 259$ states and $n_u = 10$ controls. The control inputs are the flows going in the turbines, the ducts and the reaches.

Nonlinear MPC formulation

We are interested in the following NMPC setting:

$$\begin{array}{ll} \min\limits_{w,u} & J(w(\cdot),u(\cdot)) \\ \text{s.t.} & \dot{w} = f(w,u),\ w(t) = w_0(t), \\ & u(\tau) \in U,\ w(\tau) \in W,\ \tau \in [t, t+T] \\ & w(t+T) \in \mathcal{R}_T, \end{array} \tag{3.1.5}$$

where the objective function $J(w(\cdot),u(\cdot))$ is given by:

$$J(w(\cdot),u(\cdot)) := \int_t^{t+T} \Big[(w(\tau)-w_s)^T P(w(\tau)-w_s) + (u(\tau)-u_s)^T Q(u(\tau)-u_s)\Big] d\tau$$
$$+ (w(t+T) - w_s)^T S(w(t+T) - w_s). \tag{3.1.6}$$

Here P, Q and S are given symmetric positive definite weighting matrices, and (w_s, u_s) is a steady state of the dynamics (3.1.4). The control variables are bounded by lower and upper bounds, while some state variables are also bounded and the others are unconstrained. Consequently, W and U are boxes in $\mathbb{R}^{n_w}$ and $\mathbb{R}^{n_u}$, respectively, but W is not necessarily bounded. The terminal region $\mathcal{R}_T$ is a control-invariant ellipsoidal set centered at w_s of radius $r > 0$ and scaling matrix S, i.e.:

$$\mathcal{R}_T := \left\{ w \in \mathbb{R}^{n_w} \mid (w - w_s)^T S (w - w_s) \leq r \right\}. \tag{3.1.7}$$

To compute matrix S and the radius r in (3.1.7) the procedure proposed in [40] can be used. In [104] it has been shown that the receding horizon control formulation (3.1.5) ensures the stability of the closed-loop system under mild assumptions. Therefore, the aim of this example is to track the steady state of the system and to ensure the stability of the system by satisfying the terminal constraint along the moving horizon. To have a more realistic simulation we added a disturbance to the input flow q_{in} at the beginning of the reach R_1 and the tributary flow $q_{\text{tributary}}$.

The matrices P and Q have been set to:

$$P := \operatorname{diag}\Big(\frac{0.01}{(w_s)_i^2 + 1} : 1 \leq i \leq n_w \Big), \quad Q := \operatorname{diag}\Big(\frac{4}{(u_l + u_b)_i^2 + 1} : 1 \leq i \leq n_u \Big),$$

where u_l and u_b are the lower and upper bound of the control input u, respectively.

A short description of the multiple shooting method

We briefly describe the multiple shooting formulation [27] which we use to discretize the continuous time problem (3.1.5). The time horizon $[t, t+T]$ of $T = 4$ hours is discretized into $H_p = 16$ shooting intervals with $\Delta\tau = 15$ minutes such that $\tau_0 = t$ and $\tau_{i+1} := \tau_i + \Delta\tau$ $(i = 0, \ldots, H_p - 1)$. The control $u(\cdot)$ is parametrized by using a piecewise constant function $u(\tau) = u_i$ for $\tau_i \leq \tau \leq \tau_i + \Delta\tau$ $(i = 0, \ldots, H_p - 1)$.

Let us introduce $H_p + 1$ shooting node variables s_i $(i = 0, \ldots, H_p)$. Then, by integrating the dynamic system $\dot{w} = f(w, u)$ in each interval $[\tau_i, \tau_i + \Delta\tau]$, the continuous dynamic (3.1.4) is transformed into the nonlinear equality constraints of the form:

$$g(x) + M\xi := \begin{bmatrix} s_0 - \xi \\ w(s_0, u_0) - s_1 \\ \cdots \\ w(s_{H_p-1}, u_{H_p-1}) - s_{H_p} \end{bmatrix} = 0. \tag{3.1.8}$$

Here, vector x combines all the controls and shooting node variables u_i and s_i as $x := (s_0^T, u_0^T, \ldots, s_{H_p-1}^T, u_{H_p-1}^T, s_{H_p}^T)^T$, ξ is the initial state $w_0(t)$ which is considered as a parameter, and $w(s_i, u_i)$ is the result of the integration of the dynamics from τ_i to $\tau_i + \Delta\tau$ where we set $u(\tau) = u_i$ and $w(\tau_i) = s_i$.

The objective function (3.1.6) is approximated by:

$$\begin{aligned} f(x) := & \sum_{i=0}^{H_p-1} \left[(s_i - w_s)^T P (s_i - w_s) + (u_i - u_s)^T Q (u_i - u_s)\right] \\ & + (s_{H_p} - w_s)^T S (s_{H_p} - w_s), \end{aligned} \tag{3.1.9}$$

while the constraints are imposed only at $\tau = \tau_i$, the beginning of the intervals, as:

$$s_i \in W,\ u_i \in U,\ s_{H_p} \in \mathcal{R}_T, (i = 0, \ldots, H_p - 1). \tag{3.1.10}$$

If we define $\Omega := U^{H_p} \times (W^{H_p} \times \mathcal{R}_T) \subset \mathbb{R}^{n_x}$ then Ω is convex. Moreover, the objective function (3.1.9) is convex quadratic. Therefore, the resulting optimization problem is indeed of the form P(ξ). Note that Ω is not a box but a curved convex set due to $\mathcal{R}_T$.

The nonlinear program to be solved at every sampling time has 4563 decision variables and 4403 equality constraints. We note that the equality constraint functions and their derivatives are expensive to evaluate due to the ODE integration.

Numerical simulation

Before presenting the simulation results, we give some details on the implementation. To evaluate the performance of the methods proposed in this section we implemented the following algorithms:

- Full-NMPC – the nonlinear program obtained by multiple shooting is solved at every sampling time to convergence by several SCP iterations.
- PCSCP – the implementation of Algorithm 2.3.1 using the exact Jacobian matrix of g.
- APCSCP – the implementation of Algorithm 2.3.1 with approximated Jacobian of g. Matrix A_k is fixed at $A_k = g'(x^0)$ for all $k \geq 0$, where x^0 is approximately computed off-line by performing the SCP algorithm (the exact variant of Algorithm 2.5.1) to solve the nonlinear programming P(ξ) with $\xi = \xi_0 = w_0(t)$.

- RTGN – the solution of the nonlinear program is approximated by solving a quadratic program obtained by linearizing the dynamics and the terminal constraint $s_{H_p} \in \mathcal{R}_T$. The exact Jacobian $g'(\cdot)$ of g is used. This method can be referred to as a classical real-time iteration [53] based on the constrained Gauss-Newton method [27, 48].

To compute the control-invariant set $\mathcal{R}_T$ a mixed Matlab and C++ code has been used. The computed value of r is 1.687836, while the matrix S is dense, symmetric and positive definite.

The quadratic programs (QPs) and the quadratically constrained quadratic programming problems (QCQPs) arising in the algorithms we implemented can be efficiently solved by means of interior point or other methods [30, 142]. In our implementation, we used the commercial solver `CPLEX` which can deal with both types of problems.

All the tests have been implemented in C++ running on a 16 cores 2.7GHz Intel®Xeon CPUs workstation with 12 GB of RAM. We used `CasADi`, an open source C++ package [5] which implements automatic differentiation to calculate the derivatives of the functions and offers an interface to `CVODES` from the `Sundials` package [169] to integrate the ordinary differential equations and compute the sensitivities. The integration has been parallelized by using `OpenMP`.

In the full-NMPC algorithm we performed at most 5 SCP iterations for each time interval. We stopped the SCP algorithm when the relative infinity-norm of the search direction as well as of the feasibility gap reached the tolerance $\varepsilon = 10^{-3}$. To have a fair comparison of the different methods, the starting point x^0 of the PCSCP, APCSCP and RTGN algorithms has been set to the solution of the first full-NMPC iteration.

The disturbances on the flows q_{in} and $q_{\text{tributary}}$ were generated randomly and varying from 0 to 30 and 0 to 10, respectively. All the simulations were perturbed with the same disturbance scenario.

We simulated the algorithms for $H_p = 30$ time intervals. The average computational time required by the four methods is summarized in the first part of Table 3.1. Here, `AvEvalTime` is the average time in seconds needed to evaluate the function g and its Jacobian; `AvSolTime` is the average time for solving the QP or QCQP problems; `AvAdjTime` is the average time for evaluating the adjoint direction $g'(x^k)^T y^k$ in Algorithm 2.3.1; `Total` corresponds to the sum of the previous terms and some preparation time. On average, the full-NMPC algorithm needed 3.32 iterations to converge to a solution.

The second part of Table 3.1 represents the minimum and maximum time corresponding to the evaluation of the function and its Jacobian, the solution of

Table 3.1: The average computational time of four methods

Methods	AvEvalTime[s]	AvSolTime[s]	AvAdjDirTime[s]	Total[s]
Full-NMPC	219.655 (82.43%)	46.804 (17.56%)	-	266.483
PCSCP	57.724 (89.23%)	7.627 (10.76%)	-	64.690
RTGN	58.095 (95.85%)	2.511 (4.14%)	-	60.608
APCSCP	0.443 (4.73%)	8.512 (78.90%)	1.527 (16.31%)	9.364
Methods	[min, max]	[min, max]	[min, max]	[min, max]
Full-NMPC	[164.884, 302.288]	[13.489, 114.899]	-	[179.664, 397.861]
PCSCP	[52.162, 70.776]	[4.427, 15.476]	-	[59.881, 86.258]
RTGN	[52.971, 68.021]	[2.265, 2.943]	-	[55.680, 70.333]
APCSCP	[0.402, 0.596]	[4.806, 13.110]	[1.331, 1.862]	[5.323, 14.153]

the subproblems, the calculation of the adjoint derivatives and the total time.

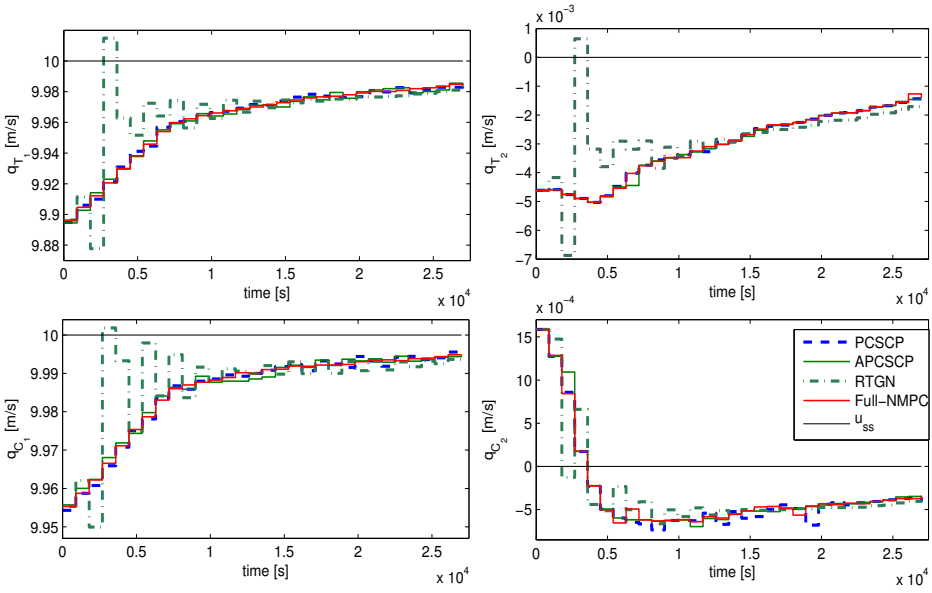


Figure 3.2: The controller profiles q_{T_1}, q_{C_1}, q_{T_2} and q_{C_1}.

It can be seen from Table 3.1 that evaluating the function and its Jacobian matrix costs approximately 82% to 96% of the total time. On the other hand, solving a QCQP problem is approximately $2-5$ times more expensive than solving a QP problem. The computationally expensive step at every iteration is the integration of the dynamics and its linearization. The average computational

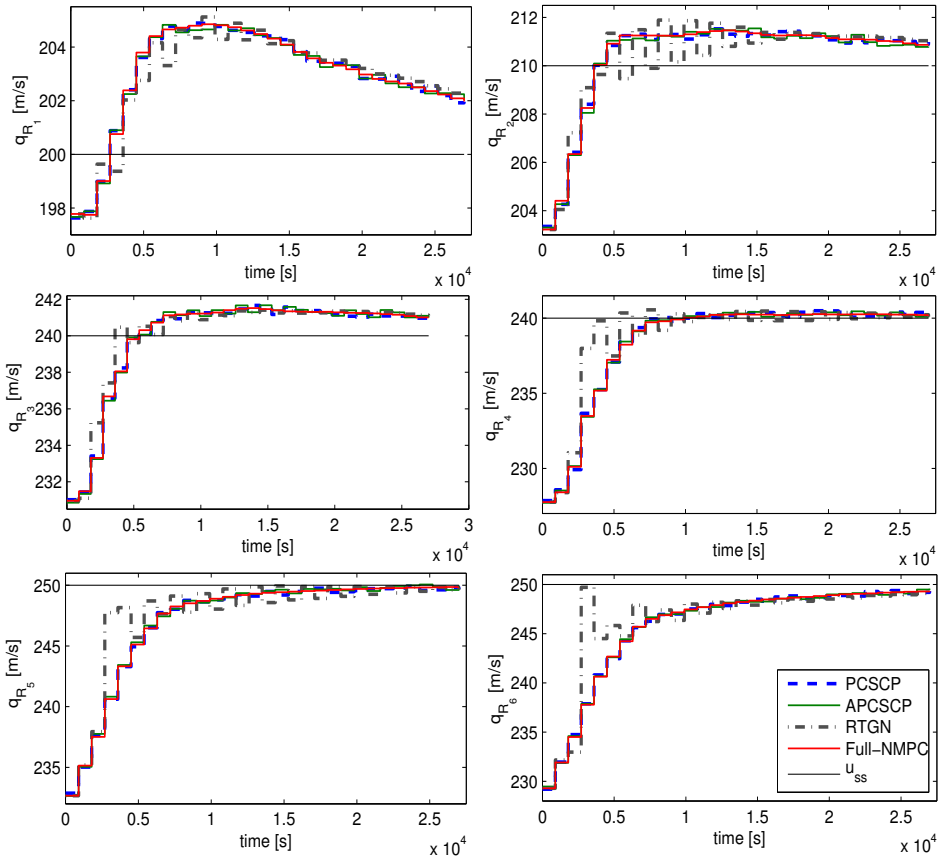


Figure 3.3: The controller profiles of $q_{R_1}, \ldots, q_{R_6}$.

time of PCSCP and RTGN is similar, while the time consumed in APCSCP is approximately six times less than PCSCP.

The closed-loop control profiles of the simulation are illustrated in Figures 3.2 and 3.3. Here, the first figure shows the flows in the turbines and the ducts of lakes L_1 and L_2, while the second one plots the flows to be controlled in the reaches R_i $(i = 1, \ldots, 6)$. We can observe that the control profiles achieved by PCSCP as well as APCSCP are close to the profiles obtained by Full-NMPC, while the results from RTGN oscillate in the first intervals due to the violation of the terminal constraint. The terminal constraint in the PCSCP was active in many iterations.

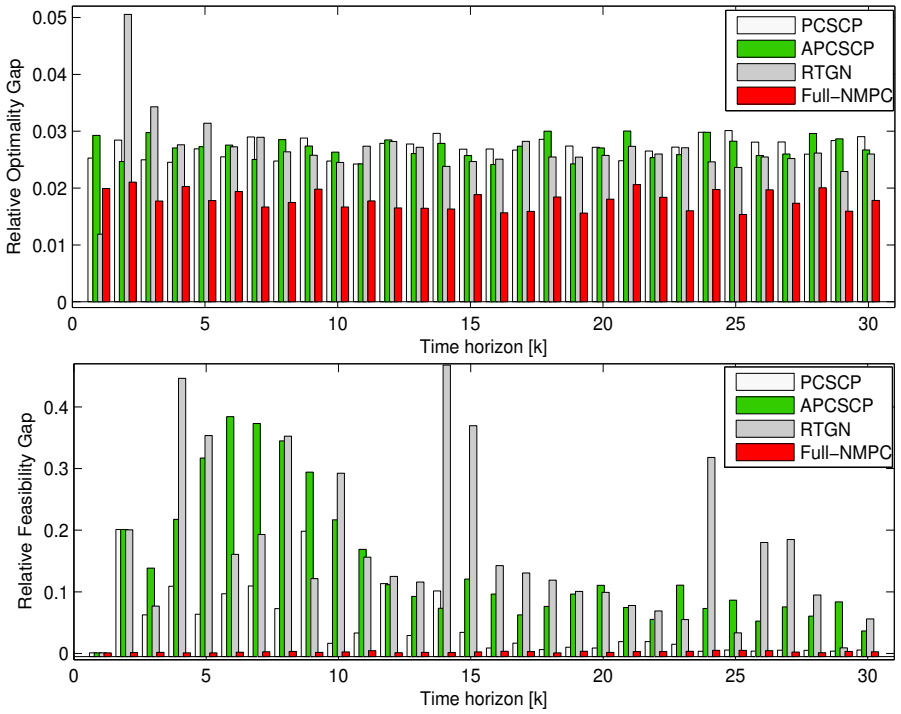


Figure 3.4: The relative feasibility and optimality gaps of PCSCP, APCSCP, RTGN and Full-NMPC.

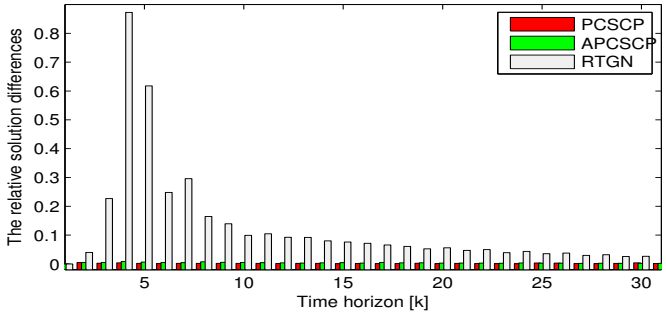


Figure 3.5: The relative differences between the approximate solution of Full-NMPC and PCSCP, APCSCP and RTGN.

Figure 3.4 shows the relative feasibility and optimality gaps of the four methods, where:

$$\texttt{Realative Feasibility Gap} := \left\|g(x^k)+M\xi_{k+1}\right\|_\infty / \max\{1.0, \left\|g(x^0)+M\xi_0\right\|_\infty\}$$

and

$$\texttt{Relative Optimality Gap} := \left\|\nabla_x \tilde{\mathcal{L}}(x^k,\lambda^k)\right\|_\infty / \max\{1.0, \left\|\nabla f(x^0)\right\|_\infty\},$$

with $\tilde{L}$ being the "full" Lagrange function of the optimization problem obtained from P(ξ) by fixing ξ at $\xi = \xi_{k+1}$. While the relative optimality gaps vary in the range $[0.01, 0.05]$ the feasibility gaps in PCSCP and Full-NMPC are smaller than in APCSCP and RTGN. The relative differences between the approximate solution of Full-NMPC and the approximate solutions of three other methods,

$$\texttt{The relative solution differences} := \left\| (x^k - x^k_{\text{full-nmpc}})./ \max\{\left|x^k_{\text{full-nmpc}}\right|, 1\} \right\|_\infty,$$

are plotted in Figure 3.5. These quantities in PCSCP and APCSCP are smaller than in RTGN. This happens because the linearization of the quadratic constraint can not adequately capture the shape of the terminal constraint $s_{H_p} \in \mathcal{R}_T$. The relative solution differences in APCSCP are as good as in PCSCP.

3.2 Time optimal trajectory planning problem

Time optimal control problems with geometric path appear frequently in mechanical engineering and industrial applications of robotic manipulators [2, 41, 155, 172, 173, 209]. In this section, we first propose a simple mathematical model for a time optimal trajectory planning problem of a car motion. Then, we show how to solve this optimal control problem by applying a direct transcription and Algorithm 2.5.1 proposed in the previous chapter.

Mathematically, based on a given reference path parameterization in a path coordinate system, the dynamic system of the car motion that we consider in this section is expressed as a differential-algebraic equation (DAE) with respect to pseudo time. The differential part of the dynamic system is linear while the algebraic part is nonlinear. The time optimal problem based on this dynamic system is described as an optimal control problem. Then, by a change of variables, the objective function of the later problem is transformed into a convex function [41, 155, 209] and the whole problem is again reformulated as an optimal control problem in the path coordinate system. To solve this problem, a direct transcription method is applied to transform it into a nonlinear optimization problem. Fortunately, this problem preserves the convexity of the objective function and the "near linearity" of the constraints. Then Algorithm 2.5.1 in Chapter 2 is applied to solve the resulting problem. Note that if SQP methods or IP methods are applied directly, they do not take into account the structure of the last problem. Therefore, we apply Algorithm 2.5.1 which exploits the specific structure of the problem and then uses freely available software [81, 124, 180, 204] for solving the convex subproblems. The numerical results show that Algorithm 2.5.1 requires few iterations to reach an optimal solution. In principle, our approach in this section can be extended to the time

optimal trajectory planning for robot control problems [155, 209] with a freedom to choose the geometric path.

Problem formulation

We consider a motion of a car along a given road shown in Figure 3.6. The car moves along the road based on a reference trajectory from A to B with the width fixed at $2b_{\max}$. Suppose that we allow the car to deviate from both sides of the reference trajectory but keep moving inside the road. The aim is to find

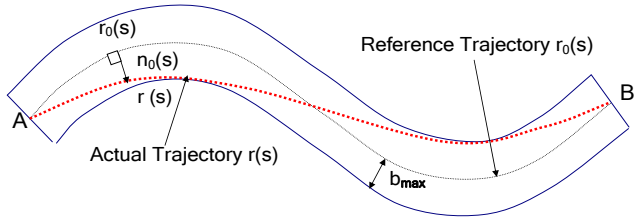


Figure 3.6: The motion of a car along a given path.

a control trajectory to steer the car from A to B in minimum time.

In order to formulate this problem, we consider a given path $r(s) := (x(s), y(s))$ of the car compounded by two components $x(s)$ and $y(s)$ in the Cartesian coordinate system, where s is a scalar path coordinate (i.e. the arc length s). Let $r_0(s) := (x_0(s), y_0(s))$ represent the reference trajectory of the road. The actual position of the car is expressed by:

$$r(s) = r_0(s) + b(s)n_0(s), \tag{3.2.1}$$

where $n_0(s)$ is the normal vector of $r_0(s)$ and $b(s)$ is the deviation from the reference trajectory. The path coordinate $b(s)$ determines the spatial geometry of the path, whereas the trajectory's time dependence follows from the relation $s(t)$. Without loss of generality, we assume that the trajectory starts at $t = 0$ and ends at $t = T$ such that $s(0) = 0 \leq s(t) \leq s(T) = 1$. By using the chain rule, the velocities $v(s)$ and the accelerations $a(s)$ of the motion can be expressed as:

$$v(s) = \dot{r}(s) = r'(s)\dot{s}, \quad a(s) = \ddot{r}(s) = r'(s)\ddot{s} + r''(s)\dot{s}^2, \tag{3.2.2}$$

where $\dot{s} = \frac{ds}{dt}$, $\ddot{s} = \frac{d^2s}{dt^2}$, $r'(s) = \frac{\partial r(s)}{\partial s}$ and $r''(s) = \frac{\partial^2 r(s)}{\partial s^2}$. Taking the first and the second order derivatives with respect to s in (3.2.1) and using the same

notation as above, we get:

$$\begin{aligned} r'(s) &= r_0'(s) + b(s)n_0'(s) + b'(s)n_0(s), \\ r''(s) &= r_0''(s) + b(s)n_0''(s) + 2b'(s)n_0'(s) + b''(s)n_0(s). \end{aligned} \tag{3.2.3}$$

Substituting (3.2.3) into the last term of (3.2.2), it implies:

$$\begin{aligned} a(s) = &[r_0'(s) + b(s)n_0'(s) + b'(s)n_0(s)]\ddot{s} \\ &+ [r_0''(s) + b(s)n_0''(s) + 2b'(s)n_0'(s) + b''(s)n_0(s)]\dot{s}^2. \end{aligned} \tag{3.2.4}$$

We assume that the acceleration is controlled by a driver and can vary in four directions (forward, backward, left and right) up to a given limit. More precisely, we have:

$$\underline{a} \leq D(s)a(s) \leq \bar{a}, \tag{3.2.5}$$

where $\underline{a} = (\underline{a}_t, \underline{a}_n)$ and $\bar{a} = (\bar{a}_t, \bar{a}_n)$ are the lower and the upper bounds of the acceleration a with respect to the two directions and $D(s) := \begin{bmatrix} r'(s)^T/\|r'(s)\| \\ n(s)^T \end{bmatrix}$ is the normalized matrix.

A time optimal trajectory planning problem minimizes the time T of the car motion from A to B. By using the same technique as in [155, 209] we can write:

$$T = \int_0^T dt = \int_{s(0)}^{s(T)} \frac{ds}{\dot{s}}. \tag{3.2.6}$$

If we denote $e(s) := \ddot{s}$ and $f(s) := \dot{s}^2$ then, by the chain rule, we have:

$$\dot{f}(s) = f'(s)\dot{s} = 2\dot{s}\ddot{s} = 2e(s)\dot{s}. \tag{3.2.7}$$

By assumption that $\dot{s} > 0$ almost everywhere, it follows from (3.2.7) that:

$$f'(s) = 2e(s), \tag{3.2.8}$$

For notational simplicity, we denote by $p_0(s) := r_0'(s)$, $p_1(s) := n_0'(s)$, $p_2(s) := n_0(s)$, $q_0(s) := r_0''(s)$ and $q_1(s) := n_0''(s)$. Then the acceleration $a(s)$ in (3.2.4) is expressed as follows:

$$\begin{aligned} a(s) \ = &[p_0(s) + b(s)p_1(s) + b'(s)p_2(s)]\ddot{s} \\ &+ [q_0(s) + b(s)q_1(s) + 2b'(s)p_1(s) + b''(s)p_2(s)]\dot{s}^2. \end{aligned} \tag{3.2.9}$$

By introducing new variables $\tilde{a}(s) := D(s)a(s)$, $c(s) := b'(s)$ and $d(s) := c'(s)$ and substituting them into (3.2.9) and (3.2.6), we obtain the following optimal

control problem:

$$
\begin{aligned}
&\min_{a(\cdot),b(\cdot),c(\cdot),d(\cdot),e(\cdot),f(\cdot)} \int_0^1 \frac{ds}{\sqrt{f(s)}} \\
\text{s.t. } &\tilde{a}(s) = D(s)[p_0(s) + b(s)p_1(s) + c(s)p_2(s)]e(s) \\
&\quad + D(s)[q_0(s) + b(s)q_1(s) + 2c(s)p_1(s) + d(s)p_2(s)]f(s) \\
&\quad b'(s) = c(s),\ c'(s) = d(s),\ f'(s) = 2e(s), \\
&\quad b(0) = b_0,\ b(1) = b_T,\ f(0) = \dot{s}_0^2,\ f(1) = \dot{s}_T^2, \\
&\quad f(s) \geq 0,\quad -b_{\max} \leq b(s) \leq b_{\max},\quad \underline{a} \leq \tilde{a}(s) \leq \bar{a},
\end{aligned}
\tag{3.2.10}
$$

for all $s \in [0, 1]$. The notation b_0 and b_T present the starting and finishing positions of the car, respectively. In most cases, $\dot{s}_0$ and $\dot{s}_T$ can be set to zero.

Note that (3.2.10) is an optimal control problem. Here, the dynamic system is a differential algebraic equation system (DAE) of pseudo time s, three differential states b, c and f, one two-dimensional algebraic state $\tilde{a}$ and two inputs e and d.

As a particular case, we show that if $b_{\max} = 0$ then problem (3.2.10) turns out to be convex [209].

Lemma 3.2.1. *If $b_{\max} = 0$ then problem (3.2.10) is convex.*

Proof. Note that the dynamic system part and the constraints from the fourth to the sixth lines of problem (3.2.10) are linear. If $b_{\max} = 0$ then it follows from the last line of (3.2.10) that $b(s) = 0$ for all $s \in [0, 1]$. Using the fourth and fifth lines we have $c(s) = 0$ which implies $d(s) = 0$ for $s \in [0, 1]$. From the definition of $D(s)$, it is easy to show that $D(s)$ is independent of $e(\cdot)$ and $f(\cdot)$. Substituting $b(s) = 0$, $c(s) = 0$, $d(s) = 0$ and $D(s)$ into the first constraint we obtain $\tilde{a}(s) = D(s)p_0(s)e(s) + D(s)q_0(s)f(s)$ which is linear. □

Numerical solution

We first transform the optimal control problem (3.2.10) into a nonlinear programming problem. Then we show how to apply Algorithm 2.5.1 in Chapter 2 to solve the resulting problem.

Direct transcription for optimal control and condensing

In order to transform the optimal control problem (3.2.10) into a finite dimensional optimization problem, we first discretize the pseudo-time interval $[0, 1]$ in the path coordinate s by $0 = s_0 < s_1 < \cdots < s_N = 1$, with $N + 1$ grid points s_k. Then, we parameterize the controls $d(\cdot)$ and $e(\cdot)$ by the piecewise constant functions $\hat{d}$ and $\hat{e}$ such that:

$$\hat{d}(s) = d^k := d(s^k),\ \hat{e}(s) = e^k := e(s^k),\quad \forall s \in [s_k, s_{k+1}),\ 0 \le k \le N-1.$$

By integrating the differential part of the dynamic systems of (3.2.10), we obtain a discretization of the state variables b, c and f as $c^{k+1/2} := \hat{c}(s_{k+1/2}) = (c^k + c^{k+1})/2$, $b^{k+1/2} := \hat{b}(s_{k+1/2}) = b^k + c^k \Delta s_k/2 + d^k \Delta s_k^2/8$ and $f^{k+1/2} := \hat{f}(s_{k+1/2}) = (f^k + f^{k+1})/2$. The function $\tilde{a}(\cdot)$ is evaluated at the middle points $s_{k+1/2} = (s_k + s_{k+1})/2$ of $[s_k, s_{k+1}]$ for all $k = 0, \ldots, N-1$ which means that $\tilde{a}^k := \tilde{a}(s_{k+1/2})$. All the coefficient functions p_0, p_1, p_2, q_0 and q_1 and the normalized matrix mapping D are evaluated at the middle points $s_{k+1/2}$ and their values denote by p_0^k, p_1^k, p_2^k, q_0^k, q_1^k and D^k for all $k = 0, \ldots, N-1$, respectively.

Next, we approximate the objective function $J(\cdot)$ by:

$$J(\tilde{a}, b, c, d, e, f) := \int_0^1 \frac{ds}{\sqrt{f(s)}} \approx \sum_{k=0}^{N-1} \frac{2\Delta s_k}{\sqrt{f^k} + \sqrt{f^{k+1}}}.$$

Put things together, we finally obtain the following nonlinear program:

$$\begin{aligned}
\min_{\tilde{a},b,c,d,e,f}\ & \hat{J}(\tilde{a}, b, c, d, e, f) := \sum_{k=0}^{N-1} \frac{2\Delta s_k}{\sqrt{f^k} + \sqrt{f^{k+1}}} \\
\text{s.t.}\quad & \tilde{a}^k = D^k[p_0^k + p_1^k b^{k+1/2} + p_2^k c^{k+1/2}]e^k \\
& \quad + D^k[q_0^k + q_1^k b^{k+1/2} + 2p_1^k c^{k+1/2} + p_2^k d^k]f^{k+1/2}, \\
& b^{k+1} - b^k = \Delta s_k c^k + \frac{1}{2}\Delta s_k^2 d^k, \\
& c^{k+1} - c^k = \Delta s_k d^k,\quad f^{k+1} - f^k = 2\Delta s_k e^k, \\
& f^0 = \dot{s}_0^2,\ f^N = \dot{s}_T^2,\quad b^0 = b_0,\ b^N = b_T, \\
& f^k \ge 0,\quad -b_{\max} \le b^k \le b_{\max},\quad a_{\min}^k \le \tilde{a}^k \le a_{\max}^k,
\end{aligned} \tag{3.2.11}$$

for all $k = 0, \ldots, N-1$.

Let us introduce a new vector $z := (\tilde{a}^0, \tilde{a}_2^0, \ldots, \tilde{a}_1^{N-1}, \tilde{a}_2^{N-1}, \ldots, f^0, \ldots, f^N)^T$ in $\mathbf{R}^{7(N+2)}$ and new functions:

$$F(z) := \sum_{k=0}^{N-1} \frac{2\Delta s_k}{\sqrt{f^k} + \sqrt{f^{k+1}}},$$

and $G(z) := (G^0(z)^T, G^1(z)^T, \ldots, G^{N-1}(z)^T)^T$ where:

$$\begin{aligned} G^k(z) &:= D^k[p_0^k + p_1^k b^{k+1/2} + p_2^k c^{k+1/2}]e^k \\ &+ D^k[q_0^k + q_1^k b^{k+1/2} + 2p_1^k c^{k+1/2} + p_2^k d^k]f^{k+1/2} - \tilde{a}^k. \end{aligned} \tag{3.2.12}$$

We also define Ω a subset in $\mathbf{R}^{7(N+2)}$ which consists of all the linear constraints of (3.2.11). Then, problem (3.2.11) can be rewritten in a short form:

$$\begin{cases} \min\limits_{z \in \mathbf{R}^{7(N+2)}} & F(z) \\ \text{s.t.} & G(z) = 0,\ z \in \Omega. \end{cases}$$

Note that the objective function $F(z)$ and the constraint set Ω of problem (3.2.11) is convex, while the equality constraint $G(z) = 0$ is nonlinear. In addition, according to Lemma 3.2.1, if b, c and d are fixed then $G(z)$ is linear.

Since the convexity is preserved under any linear transformation, we can eliminate the variables b^k, c^k and f^k in (3.2.11) to reduce the size of this problem. Such a technique is called *condensing* [50]. We introduce new variables $\tilde{a} := (\tilde{a}_1^0, \tilde{a}_2^0, \ldots, \tilde{a}_1^{N-1}, \tilde{a}_2^{N-1})^T$, $u := (c^0, d^0, \ldots, d^{N-1})^T$ and $e := (e^0, \ldots, e^{N-1})^T$. Then, using the notation $\bar{G}^k$ for the condensed form of the nonlinear constraints (3.2.12), after reduction calculations, we obtain the following optimization problem:

$$\min_{\tilde{a},u,e} J(e) := \sum_{k=0}^{N-1} \frac{2\Delta s_k}{\sqrt{P_k^T e + \dot{s}_0^2} + \sqrt{P_{k+1}^T e + \dot{s}_0^2}}$$

$$\text{s.t.} \begin{cases} \bar{G}^k(\tilde{a}^k, u, e) = 0,\ k = 0, \ldots, N-1, \\ r^T u = b_T - b_0, \\ q^T e = \dot{s}_T^2 - \dot{s}_0^2, \\ Pe + \dot{s}_0^2 \mathbf{1}_f \geq 0, \\ (-b_{\max} - b_0)\mathbf{1}_b \leq Nu \leq (b_{\max} - b_0)\mathbf{1}_b, \\ \underline{a} \leq \tilde{a}^k \leq \bar{a},\ k = 0, \ldots, N-1, \end{cases} \tag{3.2.13}$$

where $J(e)$ is convex, vectors r and q are given, $\mathbf{1}_b$ and $\mathbf{1}_f$ are two vectors whose components are 1, and P and N are two constant matrices.

If we define $w := (a^T, u^T, e^T)^T \in \mathbf{R}^{4N+1}$, $F(w) := J(e)$, $\bar{G}(w) := (\bar{G}^0(\tilde{a}^0, u, e)^T, \ldots, \bar{G}^{N-1}(\tilde{a}^0, u, e)^T)^T$ and the other linear constraints of (3.2.13) again by Ω then problem (3.2.13) can be rewritten as:

$$\begin{cases} \min\limits_{w} & F(w) \\ \text{s.t.} & G(w) = 0,\ w \in \Omega, \end{cases} \tag{3.2.14}$$

which collapses to the nonlinear programming problem (P) in Chapter 2. Finally, we note that using the *condensing technique* destroys the *sparsity* of the original problem.

Convex subproblem as a second order cone program

To apply the SCP method, Algorithm 2.5.1 in Chapter 2, the nonlinear equality constraint $G(w) = 0$ of (3.2.14) is linearized at the current iteration w^p as:

$$G'(w^p)\Delta w + G(w^p) = 0,$$

where $G'(w^p)$ is the Jacobian of G at w^p. We observe that the subproblem $\mathrm{P}(z^j, A_j, H_j)$ of (3.2.14) at w^p has a special structure that can be reformulated as a second order cone programming (SOCP) problem, see [209]. Therefore, we can exploit freely available software such as Sedumi [180] and SDPT3 [204] to solve the resulting SOCP problem.

The main step of the SOCP transformation is specified as follows. Let us introduce new slack variables $v := (v_0, \ldots, v_N)^T$ and $t := (t_0, \ldots, t_{N-1})^T$. Then the objective function (3.2.13) becomes linear:

$$J(e, v, t) := 2 \sum_{k=0}^{N-1} \Delta s_k t_k,$$

together with $2N$ additional second order convex cone constraints of the form:

$$\left\| \begin{bmatrix} 2v_k \\ P_k^T e + \dot{s}_0^2 - 1 \end{bmatrix} \right\|_2 \leq P_k^T e + \dot{s}_0^2 + 1,$$

$$\left\| \begin{bmatrix} 2 \\ v_k + v_{k+1} - t_k \end{bmatrix} \right\|_2 \leq v_k + v_{k+1} + t_k,$$

for all $k = 0, \ldots, N-1$. Replacing this objective function and adding these second order cone constraints into problem $\mathrm{P}(z^j, A_j, H_j)$ obtained by linearizing (3.2.14) at w^p we obtain an SOCP problem. In our numerical tests below, we will solve this problem by employing Sedumi [180].

Numerical results

Suppose that the reference trajectory $r_0(s) := (x_0(s), y_0(s)$ is given. By a change of variables, we compute the normal vector of this trajectory as:

$$n_0(s) = \frac{1}{\sqrt{x_0'(s)^2 + y_0'(s)^2}}\big(y_0'(s), -x_0'(s)\big) := \big(\cos\theta(s), \sin\theta(s)\big). \quad (3.2.15)$$

Now, by using the chain rule, it follows from the definition of p_0, p_1, p_2, q_0 and q_1 that $p_0(s) = (x_0'(s), y_0'(s))$, $p_1(s) = (\cos\theta(s), \sin\theta(s))\theta'(s)$, $p_2(s) = (\sin\theta(s), -\cos\theta(s))$, $q_0(s) = (x_0''(s), y_0''(s))$ and $q_1(s) = (-\sin\theta(s), \cos\theta(s))\theta'(s)^2 + (\cos\theta(s), \sin\theta(s))\theta''(s)$, where

$$\begin{cases} \theta'(s) = \frac{x_0'(s)y_0''(s) - x_0''(s)y_0'(s)}{x_0'(s)^2 + y_0'(s)^2}, \\ \theta''(s) = \frac{x_0'y_0''' - x_0'''y_0'}{x_0'(s)^2 + y_0'(s)^2} - \frac{2(x_0'y_0'' - x_0''y_0')(x_0'x_0'' + y_0'y_0'')}{(x_0'(s)^2 + y_0'(s)^2)^2}. \end{cases}$$

Algorithm 2.5.1 has been implemented in a Matlab package named `scp-cvx` and running on an Intel® Core TM2, Quad-Core Processor Q6600 (2.4GHz) PC Desktop with 3Gb RAM. We used `cvx` [81] as a modeling language and `Sedumi` [180] as a SOCP solver. In order to ensure the convergence, a back tracking line-search procedure has been implemented in Algorithm 2.5.1. We terminated Algorithm 2.5.1 if both the relative feasibility gap and the norm of the search direction reached 10^{-6}. We chose the number of grid points $N := 50$ and the parameters $\underline{a} := (-50, -10)^T$, $\bar{a} := (10, 10)^T$, $b_{\max} := 1$, $\dot{s}_0 = \dot{s}_T := 0$ and $b_0 = b_T := 0$. The sampling time was $\Delta s_k = \Delta = (s_T - s_0)/N := 1/N$ for all $k = 0, \ldots, N-1$. We test three cases as follows.

Case I: As in [155], we consider the following reference trajectory:

$$\begin{cases} x_0(s) = 50(s - 0.5), \\ y_0(s) = 200s^3 - 300s^2 + 100s. \end{cases} \quad (3.2.16)$$

With this choice, the Algorithm 2.5.1 required 10 iterations and 35 function evaluations. The results are shown in Figures 3.7 and 3.8. Here, Figure 3.7 presents the actual trajectory of the car which shows that the trajectory touches the highest bended parts of the trajectory. At the beginning motion, the trajectory is bended with a relatively big angle due to the low velocity. The vectors of velocities $v(\cdot)$ and accelerations $a(\cdot)$ of the car motion are shown in Figure 3.8, where the horizontal axis represents the s-coordinate and the vertical axis represents the velocities and accelerations in two directions x and y. The dashed path indicates the velocity or the acceleration along the x-axis while the dashed-dotted ones are the velocity and acceleration via the y-axis.

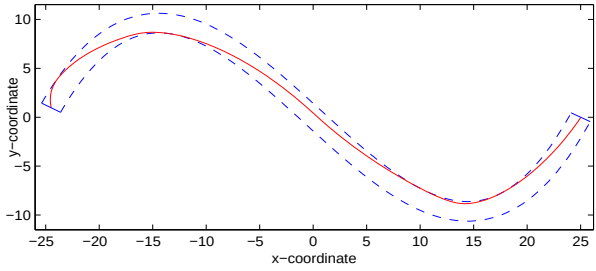


Figure 3.7: Actual trajectory of the motion for Case I (red).

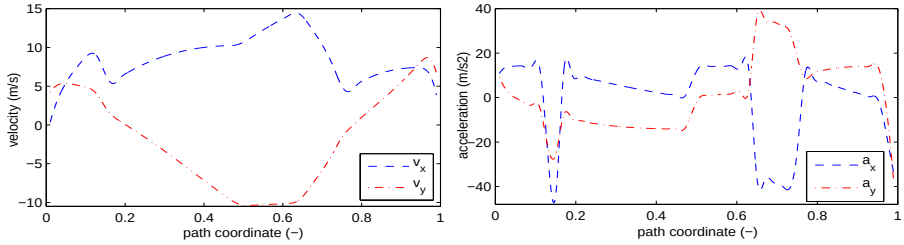


Figure 3.8: Velocities and accelerations of the motion in Case I.

Case II: In the second case, we parameterize the reference trajectory as follows:

$$\begin{cases} x_0(s) = 24\pi(s - 0.5), \\ y_0(s) = 6\sin(4\pi(s - 0.5)). \end{cases}$$

Then, the actual trajectory of the car is plotted in Figure 3.9. Similar to Case

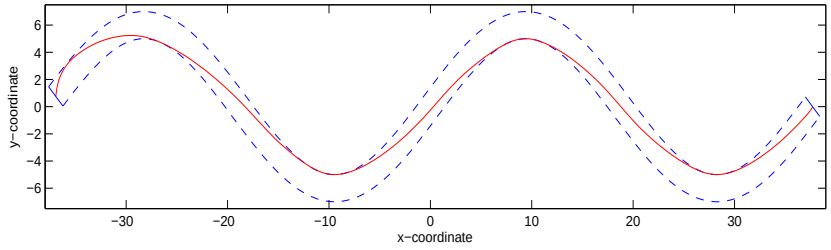


Figure 3.9: Actual trajectory based on a sin path for Case II (red).

I, the trajectory is bended at the beginning motion. Figure 3.10 shows the velocities $v(\cdot)$ and the accelerations $a(\cdot)$ of the car, respectively.

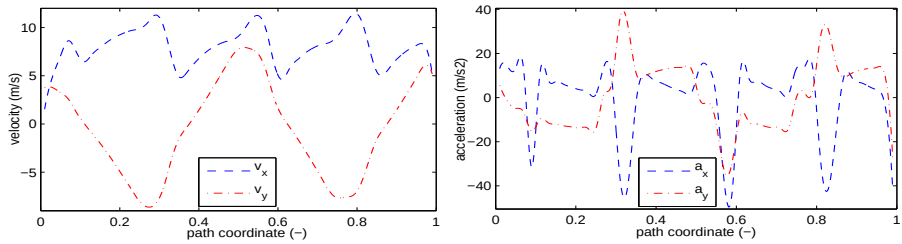


Figure 3.10: Velocities and accelerations of the motion in Case II.

Case III: Finally, we choose the following parameterization:

$$\begin{cases} x_0(s) = [20 + 10\cos(4\pi s)]\cos(2\pi s), \\ y_0(s) = [20 + 10\cos(4\pi s)]\sin(2\pi s). \end{cases}$$

Then the actual trajectory of the car is plotted in Figure 3.11. Figure 3.12 shows

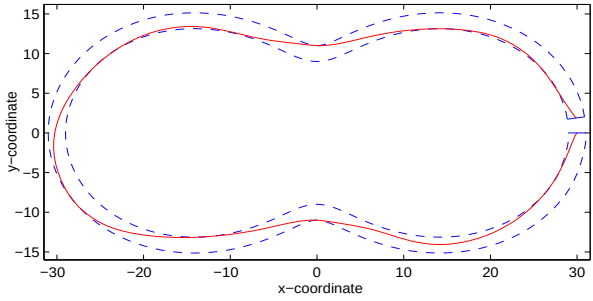


Figure 3.11: Actual trajectory of the car motion for Case III (red).

the velocities $v(\cdot)$ and the accelerations $a(\cdot)$ of the car, respectively. The results

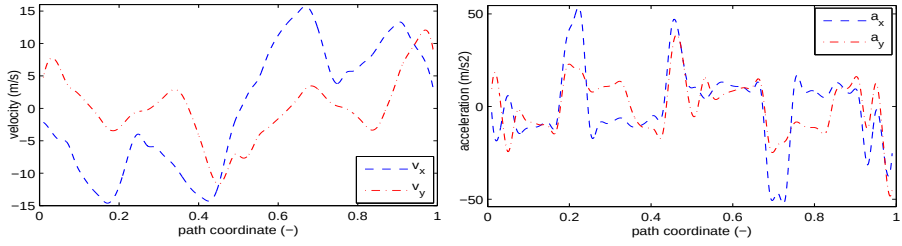


Figure 3.12: The velocities and accelerations of the motion for Case III.

and the performance of Algorithm 2.5.1 for three cases are summarized in Table

3.2. Here, `iter` is the number of iterations, `fun_eval` is the number of function

Table 3.2: The results and performance of Algorithm 2.5.1

Cases	I	II	III
iter	10	8	18
fun_eval.	21	17	36
cpu_time	90.16	83.36	160.81
constr_viol.	1.26×10^{-10}	5.83×10^{-12}	1.15×10^{-5}
obj_val	5.0907	7.3180	10.6223

evaluations, `cpu_time` is the computational time in seconds, `constr_viol` is the constraint violation and `obj_val` is the objective values. We can see from this table that the number of iterations required in Algorithm 2.5.1 for these three cases is relatively small.

Chapter 4

Inner convex approximation methods for a class of nonconvex SDP problems

4.1 A short literature review and contribution

Optimization involving matrix constraints has a broad interest and applications in static state/output feedback controller design, robust stability of systems, topology optimization and financial applications, see, e.g. [11, 31, 36, 116, 118, 120]. Many problems in these fields can be reformulated as an optimization problem with linear matrix inequality (LMI) constraints [31, 118]. Those problems can be solved efficiently and reliably by means of interior point methods for semidefinite programming (SDP) [11, 146] and efficient open-source software tools such as Sedumi [180], SDPT3 [204] and SDPA [215]. However, solving optimization problems involving nonlinear matrix inequality constraints is still a big challenge in practice. Methods and algorithms for nonlinear matrix constrained optimization problems are still limited [44, 72, 116].

In control theory, many problems related to the design of a reduced-order controller can conveniently be reformulated as a feasibility problem or an optimization problem with bilinear matrix inequality (BMI) constraints by means of, for instance, Lyapunov's theory. The BMI constraints make the problems much more difficult than the LMI ones due to their nonconvexity and possible nonsmoothness. It was shown in [24] that the optimization problems

involving BMI are NP-hard. Several approaches for solving optimization problems with BMI constraints have been proposed. For instance, Goh *et al* [76] considered problems in robust control by means of BMI optimization by using global optimization methods. Hol *et al* in [99] proposed to use a sum-of-squares approach to fixed order $\mathcal{H}$-infinity synthesis problems. Apkarian and Tuan [7] proposed local and global methods for solving BMIs which were also based on global optimization techniques. These authors further considered these problems by proposing parametric formulations and difference of two convex functions (DC) programming approaches. A similar approach can be found in [1]. However, finding a global optimum for an optimization problem with BMI constraints is in general impractical and global optimization methods are usually recommended only for low dimensional problems.

Alternatively, sequential semidefinite programming (SSDP) methods for nonlinear SDP were considered by Fares *et al* in [65]. These methods were applied to solve many problems in robust control. Thevenet *et al* [186] studied spectral SDP methods for solving problems involving BMIs arising in controller design. Another approach was based on the fact that the problems with BMI constraints can be reformulated as problems with LMI constraints coupled with additional rank constraints. In [150] Orsi *et al* developed a Newton-like method for solving problems of this type.

In this chapter, we propose two local optimization methods for solving a class of nonconvex semidefinite programming problems. In particular, these methods can be applied to solve optimization problems with BMI constraints.

Contribution of Chapter 4. The contribution of this chapter consists of the following three points:

a) We propose a local optimization algorithm for finding stationary points of a class of nonconvex semidefinite programming problems. This algorithm can be viewed as a generalization of classical *inner convex approximation* methods [9, 127]. A variant of this approach is derived which we call the *generalized convex-concave decomposition algorithm*. The later algorithm can be considered as a generalization of the DC algorithm (DCA) studied in [156, 177, 190] for scalar functionals.

b) We prove the convergence of both algorithms to a stationary point of the original problem under the standard assumptions which are usually required in nonconvex semidefinite programming.

c) As a particular case, we show that these algorithms can be applied to solve optimization problems with BMI constraints by providing

some formulations to build an *overestimate* as well as a *convex-concave decomposition* for given BMI constraints.

We note that both algorithms developed in this chapter are not only a technical extension of existing methods for scalar functions because many characterizations of standard nonlinear programming are no longer preserved in nonlinear semidefinite programming, see, e.g. [170, 181]. Moreover, converting a nonlinear semidefinite programming problem into a standard nonlinear programming one usually requires some spectral functions which are related to the eigenvalues of matrix-valued mappings. The resulting problem is in general nonconvex and nonsmooth, see, e.g. [34]. In addition, the algorithms are modified by using a *regularization technique* to ensure the strict descent of the objective function. The advantages of these algorithms are that they are *very simple to implement* by employing available semidefinite programming software tools [180, 204, 215] and *no globalization strategy* such as line-search or filtering procedures is needed. The second method still works in practice for nonsmooth optimization problems, where the objective function and the concave parts are only subdifferentiable, but not necessarily differentiable. Note that this algorithm is different from the standard DC method in [156, 177, 190] since we work directly with positive semidefinite matrix inequality constraints instead of transforming them into DC representations as in [1, 7].

Outline of Chapter 4. This chapter is organized as follows. In Section 4.2, after recalling some concepts in semidefinite programming, we present the problem formulation and its optimality condition. Section 4.3 considers a generalized inner convex approximation method and investigates its convergence. As a variant of the generalized inner convex approximation method, a generalized convex-concave decomposition algorithm is also studied in this section. The convergence of this algorithm is also proved under standard assumptions. Besides, we also provide some convex-concave decompositions for a given BMI mapping which will be used in the next chapter. We end this chapter with some conclusion.

4.2 Problem statement and optimality condition

Generalized convexity

Let us recall the following concepts which will be used in the sequel. For given matrices X and Y in $\mathcal{S}^p$, the relation $X \succeq Y$ (resp., $X \preceq Y$) means that

$X-Y \in \mathcal{S}^p_+$ (resp., $Y-X \in \mathcal{S}^p_+$) and $X \succ Y$ (resp., $X \prec Y$) means $X-Y \in \mathcal{S}^p_{++}$ (resp., $Y-X \in \mathcal{S}^p_{++}$). The notation $X \circ Y := \mathbf{trace}(X^T Y)$ denotes an inner product of two matrices X and Y defined on $\mathcal{S}^p$, where $\mathbf{trace}(Z)$ is the trace of matrix Z.

Definition 4.2.1 ([170])**.** *A matrix-valued mapping $G : \mathbf{R}^n \to \mathcal{S}^p$ is said to be positive semidefinite convex (psd-convex) on a convex subset $\Omega \subseteq \mathbf{R}^n$ if for all $t \in [0,1]$ and $x, y \in \Omega$, one has:*

$$G(tx + (1-t)y) \preceq tG(x) + (1-t)G(y). \tag{4.2.1}$$

If (4.2.1) holds true for $\prec$ instead of $\preceq$ for $t \in (0,1)$ then G is said to be strictly psd-convex on Ω. Alternatively, if we replace $\preceq$ in (4.2.1) by $\succeq$ then G is said to be psd-concave on Ω.

It is obvious that any convex function $f : \mathbf{R}^n \to \mathbf{R}$ is psd-convex with $p = 1$.

The derivative of a matrix-valued mapping G at x is a linear mapping $\mathbb{D}G$ from $\mathbf{R}^n$ to $\mathbf{R}^{p \times p}$ which is defined by:

$$\mathbb{D}G(x)h := \sum_{i=1}^{n} h_i \frac{\partial G}{\partial x_i}(x), \ \forall h \in \mathbf{R}^n.$$

For a given convex set $\Omega \in \mathbf{R}^n$, the matrix-valued mapping G is said to be differentiable on a subset Ω if its derivative $\mathbb{D}G(x)$ exists at every $x \in \Omega$. The definitions of the second order derivatives of matrix-valued mappings can be found, e.g., in [170]. Let $A : \mathbf{R}^n \to \mathcal{S}^p$ be a linear mapping defined as $Ax := \sum_{i=1}^n x_i A_i$, where $A_i \in \mathcal{S}^p$ for $i = 1, \ldots, n$. The adjoint operator of A, A^*, is defined as $A^* Z := (A_1 \circ Z, A_2 \circ Z, \cdots, A_n \circ Z)^T$ for any $Z \in \mathcal{S}^p$.

Lemma 4.2.1.

a) A matrix-valued mapping G is psd-convex on Ω if and only if for any $v \in \mathbf{R}^p$ the function $\varphi(x) := v^T G(x) v$ is convex on Ω.

b) A mapping G is psd-convex on Ω if and only if for all x and y in Ω, one has:

$$G(y) - G(x) \succeq \mathbb{D}G(x)(y-x). \tag{4.2.2}$$

Proof. The proof of the statement a) can be found in [170]. We prove b). Let $\varphi(x) = v^T G(x) v$ for any $v \in \mathbf{R}^p$. If G is psd-convex then φ is convex. We have $\varphi(y) - \varphi(x) \geq \nabla\varphi(x)^T (y-x)$. Now, $\nabla\varphi(x)^T (y-x) = \sum_{i=1}^n (y_i - x_i) v^T \frac{\partial G}{\partial x_i}(x) v = v^T [\mathbb{D}G(x)(y-x)] v$. Hence, $v^T \left[G(y) - G(x) - \mathbb{D}G(x)(y-x)\right] v \geq 0$ for all v. We conclude that (4.2.2) holds. Conversely, if (4.2.2) holds then, for any v, we have $v^T \left[G(y) - G(x) - \mathbb{D}G(x)(y-x)\right] v \geq 0$, which is equivalent to $\varphi(y) - \varphi(x) \geq$

$\nabla\varphi(x)^T(y-x)$. Thus φ is convex. By virtue of a), the mapping G is psd-convex. □

As a generalization of DC functions, we define a generalized convex-concave decomposition in the symmetric positive semidefinite cone as follows.

Definition 4.2.2. *A matrix-valued mapping $F : \mathbf{R}^n \to \mathcal{S}^p$ is said to be a psd-convex-concave mapping if F can be represented as a difference of two psd-convex mappings, i.e. $F(x) = G(x) - H(x)$, where G and H are psd-convex. The pair (H, G) is called a psd-DC (or psd-convex-concave) decomposition of F.*

Instead of using the vector x as a decision variable, we can use the matrix X as a matrix variable in $\mathbf{R}^{m\times n}$. Note that any matrix X can be considered as an $m \times n$-column vector by vectorizing with respect to its columns, i.e. $x = \mathbf{vec}(X) := (X_{11}, X_{21}, \ldots, X_{mn})^T$. The inverse mapping of **vec** is called **mat**. Since **vec** and **mat** are linear operators, the psd-convexity is still preserved under these operators.

Let us consider a bilinear matrix form:

$$F(X, Y) := X^T Y + Y^T X. \tag{4.2.3}$$

By using the Kronecker product, we can write F as $\mathbf{vec}(F(X,Y)) = (I_x \otimes X^T)\mathbf{vec}(Y) + (I_y \otimes Y^T)\mathbf{vec}(X) = (\sum_{i,j} x_i y_j)$, where I_x and I_y are two appropriate identity matrices, $\otimes$ denotes the Kronecker product. Hence, the vectorization of $F(X, Y)$ is indeed a bilinear form of two vectors $x := \mathbf{vec}(X)$ and $y := \mathbf{vec}(Y)$.

Example 4.2.1.(*Psd-convex-concave decompositions of BMIs*) We consider a bilinear matrix-valued mapping $b(X, Y) := X^T Y + Y^T X$. The following expression represents three different psd-convex-concave decompositions of $b(\cdot)$:

$$\begin{aligned} b(X,Y) &= (X+Y)^T(X+Y) - (X^TX + Y^TY) \\ &= X^TX + Y^TY - (X-Y)^T(X-Y) \\ &= \frac{1}{2}[(X+Y)^T(X+Y) - (X-Y)^T(X-Y)]. \end{aligned} \tag{4.2.4}$$

We will show in Lemma 5.2.1 in the next chapter that the matrix-valued mappings $(X+Y)^T(X+Y)$, X^TX, Y^TY and $(X-Y)^T(X-Y)$ are psd-convex. Intuitively, we can see that the first decomposition has a "strong curvature" on the second term, while the second and the third decompositions have "less curvature" on the second term due to the compensation between X and Y. We will later see in the next chapter that less curvature in the concave part is beneficial for the algorithms of this chapter. ♢

Note that each given psd-convex-concave mapping may possess many psd-convex-concave decompositions. Moreover, we can write $F(x) = G(x) - H(x) = [G(x) + K(x)] - [H(x) + K(x)]$ for any symmetric positive definite matrix-valued mapping K.

Optimization involving matrix inequality constraints

In this chapter we consider the following nonconvex semidefinite programming problem:

$$\begin{cases} \min\limits_{x \in \mathbf{R}^n} & f(x) \\ \text{s.t.} & F_i(x) \preceq 0, \quad i = 1, \ldots, m, \\ & x \in \Omega, \end{cases} \tag{NSDP}$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is convex, Ω is a nonempty, closed and convex set in $\mathbf{R}^n$ and $F_i : \mathbf{R}^n \to \mathcal{S}^{p_i}$ $(i = 1, \ldots, m)$ are nonconvex matrix-valued mappings and smooth. As a special case, if each matrix-valued mapping F_i is psd-convex-concave, i.e. $F_i(x) = G_i(x) - H_i(x)$ for $i = 1, \ldots, m$ then the problem (NSDP) collapses to a nonconvex semidefinite programming problem with psd-convex-concave constraints of the form:

$$\begin{cases} \min\limits_{x \in \mathbf{R}^n} & f(x) \\ \text{s.t.} & G_i(x) - H_i(x) \preceq 0, \ i = 1, \ldots, m, \\ & x \in \Omega, \end{cases} \tag{4.2.5}$$

where the function f and the set Ω are defined as in (NSDP). The problem (4.2.5) is referred to as a convex optimization problem with psd-convex-concave matrix inequality constraints.

Note that if H_i is affine for $i = 1, \ldots, m$ then (4.2.5) becomes a convex semidefinite program.

Throughout this chapter, we assume that all the functions and matrix-valued mappings are *twice differentiable* on their domain [170, 186] as stated in the following assumption. However, this assumption can be reduced to the *subdifferentiability* of the objective function and the concave parts of the convex-concave decompositions of matrix-valued mappings.

Asumption A. 4.2.4. *The function f and the matrix-valued mappings F_i (resp. G_i and H_i) are twice continuously differentiable on their domain for $i = 1, \ldots, m$.*

Optimality condition

Let us define $\mathcal{L}(x,\Lambda) := f(x) + \sum_{i=1}^{m} \Lambda_i \circ F_i(x)$ the Lagrange function of (NSDP), where $\Lambda_i \in \mathcal{S}^p$ is the Lagrange multiplier associated with the constraint $F_i(x) \preceq 0$ for $i = 1,\ldots,m$. The generalized KKT condition of (NSDP) is presented as:

$$\left\{\begin{array}{l} 0 \in \nabla f(x) + \sum_{i=1}^{m} \mathbb{D}F_i(x)^*\Lambda_i + \mathcal{N}_\Omega(x), \\ F_i(x) \preceq 0,\ \Lambda_i \succeq 0, \\ F_i(x) \circ \Lambda_i = 0,\ i = 1,\ldots,m. \end{array}\right. \tag{4.2.6}$$

Here, $\mathcal{N}_\Omega(x)$ is the normal cone of Ω at x. A pair (x^*,Λ^*) satisfying (4.2.6) is called a KKT point, x^* is called a stationary point and Λ^* is the corresponding multiplier of (NSDP). The generalized optimality condition for nonlinear semidefinite programming can be found in the literature, e.g., [170, 181].

Let us denote by:

$$\mathcal{D} := \{x \in \Omega \mid F_i(x) \preceq 0,\ i = 1,\ldots,m\}, \tag{4.2.7}$$

the feasible set of (NSDP) and by $\mathrm{ri}(\mathcal{D})$ the relative interior of $\mathcal{D}$ which is defined by:

$$\mathrm{ri}(\mathcal{D}) := \{x \in \mathrm{ri}(\Omega) \mid F_i(x) \prec 0,\ i = 1,\ldots,m\},$$

where $\mathrm{ri}(\Omega)$ is the set of classical relative interiors of Ω [30].

The following condition is a fundamental assumption in this chapter.

Asumption A.4.2.5. *The relative interior ri($\mathcal{D}$) of $\mathcal{D}$ is nonempty.*

Note that this assumption is crucial for our methods, because, as we shall see, the methods require a strictly feasible starting point $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$. Finding such a point is in principle not an easy task. However, in many problems, this assumption is always satisfied. In the next chapter we will propose different techniques to determine a starting point for some nonconvex sets formed by BMI constraints.

4.3 Generalized inner convex approximation algorithms

Let us first describe the idea of the inner convex approximation for the scalar case. Let $f : \mathbf{R}^n \to \mathbf{R}$ be a continuous nonconvex function. A convex function $g(\cdot; y)$ depending on a parameter y is called a convex overestimate of $f(\cdot)$ w.r.t.

the parameterization $y := \psi(x)$ if $g(x, \psi(x)) = f(x)$ and $f(z) \leq g(z; y)$ for all y and z. In this case, we have $\{z \mid g(z; y) \leq 0\} \subseteq \{z \mid f(z) \leq 0\}$. The idea of the algorithm is to approximate the nonconvex feasible set of the problem by a sequence of inner convex approximations and to solve the convex subproblems formed by these sets to obtain approximate solutions. In the sequel, we shall generalize this idea from scalar functions to matrix-valued mappings.

Psd-convex overestimate of a matrix-valued mapping

We start by considering a convex overestimate of a scalar function. Then we generalize the idea to nonconvex matrix-valued mappings.

Example 4.3.1. Let f be a continuously differentiable function such that its gradient ∇f is Lipschitz continuous with a Lipschitz constant $L_f > 0$, i.e. $\|\nabla f(y) - \nabla f(x)\| \leq L_f \|y - x\|$ for all x and y. Then, it is well-known that $|f(z) - f(x) - \nabla f(x)^T (z - x)| \leq \frac{L_f}{2} \|z - x\|^2$. Therefore, for any x and z we have $f(z) \leq g(z; x)$ with $g(z; x) := f(x) + \nabla f(x)^T (z - x) + \frac{L_f}{2} \|z - x\|^2$. Moreover, $f(x) = g(x; x)$ for any x. We conclude that $g(\cdot; x)$ is a convex overestimate of f w.r.t the parameterization $y = \psi(x) := x$. Now, since $f(z) \leq g(z; x)$, if we fix $x := \bar{x}$ and find a point v such that $g(v; \bar{x}) \leq 0$ then $f(v) \leq 0$. Consequently if the set $\{x \mid f(x) < 0\}$ is nonempty, we can find a point v such that $g(v; \bar{x}) \leq 0$. The convex set $\mathcal{C}(x) := \{z \mid g(z; x) \leq 0\}$ is called an inner convex approximation of $\{z \mid f(z) \leq 0\}$. ♢

Example 4.3.2.(see [9]) We consider the function $f(x) = x_1 x_2$ in $\mathbf{R}^2$. The function $g(x; y) = \frac{y}{2} x_1^2 + \frac{1}{2y} x_2^2$ is a convex overestimate of f w.r.t. the parameterization $y = \psi(x) := x_1/x_2$ provided that $y > 0$. This example shows that the parameterization ψ is not always the identity. ♢

Let us generalize the convex overestimate concept to matrix-valued mappings.

Definition 4.3.1. *Let us consider a psd-nonconvex matrix-valued mapping $F : \mathcal{X} \subseteq \mathbf{R}^n \to \mathcal{S}^p$. A psd-convex matrix-valued mapping $G(\cdot; y)$ is said to be a psd-convex overestimate of F w.r.t. the parameterization $y := \psi(x)$ if $G(x; \psi(x)) = F(x)$ and $F(z) \preceq G(z; y)$ for all x and z in $\mathcal{X}$.*

Let us provide two important examples that satisfy Definition 4.3.1.

Example 4.3.3. Let $\mathcal{B}_Q(X, Y) = X^T Q^{-1} Y + Y^T Q^{-1} X$ be a bilinear form with $Q = Q_1 + Q_2$, $Q_1 \succ 0$ and $Q_2 \succ 0$ arbitrary, where X and Y are two $n \times p$

matrices. We consider the parametric quadratic form:

$$\begin{aligned}\mathcal{Q}_Q(X,Y;\bar{X},\bar{Y}) :=&(X-\bar{X})^TQ_1^{-1}(X-\bar{X})+(Y-\bar{Y})^TQ_2^{-1}(Y-\bar{Y})\\ &+\bar{X}^TQ^{-1}Y+\bar{Y}^TQ^{-1}X+X^TQ^{-1}\bar{Y}\\ &+Y^TQ^{-1}\bar{X}-\bar{X}^TQ^{-1}\bar{Y}-\bar{Y}^TQ^{-1}\bar{X}.\end{aligned} \tag{4.3.1}$$

One can show that $\mathcal{Q}_Q(X,Y;\bar{X},\bar{Y})$ is a psd-convex overestimate of $\mathcal{B}_Q(X,Y)$ w.r.t. the parameterization $\psi(\bar{X},\bar{Y}) := (\bar{X},\bar{Y})$.

Indeed, it is obvious that $\mathcal{Q}_Q(\bar{X},\bar{Y};\bar{X},\bar{Y}) = \mathcal{B}_Q(\bar{X},\bar{Y})$. We only prove the second condition in Definition 4.3.1. We consider the expression $\mathcal{D}_Q := \bar{X}^TQ^{-1}Y+\bar{Y}^TQ^{-1}X+X^TQ^{-1}\bar{Y}+Y^TQ^{-1}\bar{X}-\bar{X}^TQ^{-1}\bar{Y}-\bar{Y}^TQ\bar{X}-X^TQ^{-1}Y-Y^TQ^{-1}X$. By rearranging this expression, we can easily show that $\mathcal{D}_Q = -(X-\bar{X})^TQ^{-1}(Y-\bar{Y})-(Y-\bar{Y})^TQ^{-1}(X-\bar{X})$. Now, since $Q = Q_1+Q_2$, by [13], we can write:

$$\begin{aligned}-\mathcal{D}_Q &= (X-\bar{X})^T(Q_1+Q_2)^{-1}(Y-\bar{Y})+(Y-\bar{Y})^T(Q_1+Q_2)^{-1}(X-\bar{X})\\ &\preceq (X-\bar{X})^TQ_1^{-1}(X-\bar{X})+(Y-\bar{Y})^TQ_2^{-1}(Y-\bar{Y}).\end{aligned} \tag{4.3.2}$$

Note that $\mathcal{D}_Q = \mathcal{Q}_Q-\mathcal{B}_Q-(X-\bar{X})^TQ_1^{-1}(X-\bar{X})+(Y-\bar{Y})^TQ_2^{-1}(Y-\bar{Y})$. Therefore, we have $\mathcal{Q}_Q(X,Y;\bar{X},\bar{Y}) \succeq \mathcal{B}_Q(X,Y)$ for all X,Y and $\bar{X},\bar{Y}$ due to (4.3.2). ♢

Example 4.3.4. Let us consider a psd-convex-concave mapping $F(x) := G(x)-H(x)$, where G and H are both psd-convex. Let H be differentiable and $\mathcal{L}_2(x;\bar{x}) := H(\bar{x})+\mathbb{D}H(\bar{x})(x-\bar{x})$ be the linearization of H at $\bar{x}$. We define $F(x;\bar{x}) := G(x)-\mathcal{L}_2(x;\bar{x})$. According to Lemma 4.2.1, we have:

$$-H(x) \preceq -H(\bar{x})-\mathbb{D}H(\bar{x})(x-\bar{x}), \quad \forall x,$$

which is equivalent to:

$$G(x)-H(x) \preceq G(x)-H(\bar{x})-\mathbb{D}H(\bar{x})(x-\bar{x}), \quad \forall x.$$

Hence, $F(\cdot;\bar{x})$ is a psd-convex overestimate of F w.r.t. the parametrization $\psi(\bar{x}) := \bar{x}$. ♢

Remark 4.3.1. *Example 4.3.3 shows that the "Lipschitz constant" of the approximating function (4.3.1) is (Q_1^{-1},Q_2^{-1}). Moreover, as indicated in Examples 4.3.3 and 4.3.4 that the psd-convex overestimate of a matrix-valued inequality constraint is not unique. In practice, it is important to find an appropriate psd-convex overestimate for this constraint to make the algorithm perform efficiently. Note that the psd-convex overestimate $\mathcal{Q}_Q$ of $\mathcal{B}_Q$ in Example 4.3.3 may be less conservative than the convex-concave decomposition since all the terms in $\mathcal{Q}_Q$ relate to the differences $X-\bar{X}$ and $Y-\bar{Y}$ rather than X and Y.*

The algorithms

In this subsection, we present two algorithms. The first algorithm is a generalized inner convex approximation method for solving (NSDP) and the second one is a generalized convex-concave decomposition method for solving (4.2.5).

Generalized inner convex approximation algorithm

For each $i = 1, \ldots, m$, we assume that $G_i(\cdot; y_i)$ is an overestimate of F_i with the parameterization $y_i = \psi_i(x)$. The main step of the algorithm is to solve a convex semidefinite programming problem formed at the iteration $\bar{x}^k \in \Omega$ by using inner psd-convex approximations. The convex subproblem is defined as follows:

$$\left\{\begin{array}{ll} \min\limits_{x} & \left\{f_k(x) := f(x) + \frac{1}{2}(x - \bar{x}^k)^T Q_k (x - \bar{x}^k)\right\} \\ \text{s.t.} & G_i(x; \bar{y}_i^k) \preceq 0,\ i = 1, \ldots, m, \\ & x \in \Omega. \end{array}\right. \tag{CSDP($\bar{x}^k$)}$$

Here, $Q_k \in \mathcal{S}_+^n$ is given and the second term in the objective function is referred to as a regularization term; $\bar{y}_i^k := \psi_i(\bar{x}^k)$ is the parameterization of the convex overestimate G_i of F_i.

Let us define by $\mathcal{S}(\bar{x}^k, Q_k)$ the solution mapping of CSDP($\bar{x}^k$) depending on the parameters $(\bar{x}^k, Q_k)$. Note that the problem CSDP($\bar{x}^k$) is convex, $\mathcal{S}(\bar{x}^k, Q_k)$ is multivalued and convex. The feasible set of CSDP($\bar{x}^k$) is written as:

$$\mathcal{D}(\bar{x}^k) := \left\{x \in \Omega \mid G_i(x; \psi_i(\bar{x}^k)) \preceq 0,\ i = 1, \ldots, m\right\}. \tag{4.3.3}$$

The algorithm for solving (NSDP) starts from an initial point $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$ and generates a sequence $\{\bar{x}^k\}_{k \geq 0}$ by solving a sequence of convex semidefinite programming subproblems CSDP($\bar{x}^k$) approximated at $\bar{x}^k$. More precisely, it is presented in detail as follows:

Algorithm 4.3.1.(*Generalized inner convex approximation algorithm*).
Initialization. Determine an initial point $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$. Compute $\bar{y}_i^0 := \psi_i(\bar{x}^0)$ for $i = 1, \ldots, m$. Choose a regularization matrix $Q_0 \in \mathcal{S}_+^n$.
Iteration. For $k = 0, 1, \ldots,$ perform the following steps:

Step 1. For given $\bar{x}^k$, if a given criterion is satisfied then terminate.

Step 2. Solve the convex semidefinite program CSDP($\bar{x}^k$) to obtain a solution $\bar{x}^{k+1}$ and the corresponding Lagrange multiplier $\bar{\Lambda}^{k+1}$.

Step 3. Update $\bar{y}_i^{k+1} := \psi_i(\bar{x}^{k+1})$, the regularization matrix $Q_{k+1} \in \mathcal{S}_+^n$ (if necessary) and go back to *Step 1.*

End.

We notice that the stopping criterion at Step 1 of Algorithm 4.3.1 will be specified in Chapter 5. The Lagrange multiplier is not directly used in Algorithm 4.3.1, but it may be used to update matrix Q_k if necessary.

Convex-concave decomposition algorithm

As a special case, if we use the convex overestimate given in Example 4.3.4, we obtain a variant of Algorithm 4.3.1. In this case, the convex subproblem CSDP($\bar{x}^k$) reduces to the following form:

$$\begin{cases} \min\limits_{x} & \left\{f_k(x) := f(x) + \frac{1}{2}(x-\bar{x}^k)^T Q_k (x-\bar{x}^k\right\} \\ \text{s.t.} & G_i(x) - H_i(\bar{x}^k) - \mathbb{D}H_i(\bar{x}^k)(x-\bar{x}^k) \preceq 0,\ i=1,\ldots,m, \\ & x \in \Omega. \end{cases} \tag{4.3.4}$$

Similar to (4.3.3), the feasible set of this problem becomes:

$$\mathcal{D}(\bar{x}^k) := \left\{x \in \Omega | G_i(x) - H_i(\bar{x}^k) - \mathbb{D}H_i(\bar{x}^k)(x-\bar{x}^k) \preceq 0, i=1,\ldots,m\right\}. \tag{4.3.5}$$

The *generalized convex-concave decomposition algorithm* for solving (4.2.5) is described as follows:

Algorithm 4.3.2.(*Generalized convex-concave decomposition algorithm*).
Initialization. Determine an initial point $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$. Choose a regularization matrix $Q_0 \in \mathcal{S}_+^n$.
Iteration. For $k = 0, 1, \ldots,$ perform the following steps:

Step 1: Solve the convex semidefinite program (4.3.4) to obtain a solution $\bar{x}^{k+1}$ and the corresponding Lagrange multiplier $\bar{\Lambda}^{k+1}$.

Step 2: If $\left\|\bar{x}^{k+1} - \bar{x}^k\right\| \leq \varepsilon$ for a given tolerance $\varepsilon > 0$ then terminate. Otherwise, update $Q_k \in \mathcal{S}_+^n$ (if necessary) and go back to *Step 1.*

End.

The following main property of Algorithms 4.3.1 and 4.3.2 makes an implementation very easy. If the initial point $\bar{x}^0$ belongs to the relative interior of the feasible set $\mathcal{D}$, i.e. $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$, then Algorithms 4.3.1 and 4.3.2 each generate a sequence $\left\{\bar{x}^k\right\}$ which still belongs to $\mathcal{D}$. This means that the sequence $\left\{\bar{x}^k\right\}$ is

feasible. Moreover, the corresponding sequence of the objective values $\{f(\bar{x}^k)\}$ is nonincreasing. In particular, no line-search procedure is needed to ensure global convergence. This properties following from the fact that $G_i(\cdot;\psi_i(\bar{x}^k))$ is an overestimate of F_i. Hence, if the subproblem CSDP($\bar{x}^k$) (resp. (4.3.4)) has a solution $\bar{x}^{k+1}$ then it is feasible to (NSDP) (resp. (4.2.5)). Geometrically, Algorithms 4.3.1 and 4.3.2 can be seen as inner approximation methods.

The main tasks of an implementation of Algorithms 4.3.1 and 4.3.2 consist of:

1. determining an initial point $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$, and
2. solving the convex semidefinite program CSDP($\bar{x}^k$) or (4.3.4) repeatedly.

As mentioned before, since $\mathcal{D}$ is nonconvex, finding an initial point $\bar{x}^0$ in $\mathrm{ri}(\mathcal{D})$ is, in principle, not an easy task. Nevertheless, in some practical problems, this can be done by exploiting the special structure of the problem (see Chapter 5 for more details).

To solve the convex subproblem (4.3.4) or CSDP($\bar{x}^k$), we can either implement an interior point method and exploit the structure of the problem or transform it into a standard SDP problem and then make use of available software tools for SDP [180, 204]. The regularization matrix Q_k can be fixed at an appropriate choice for all iterations, e.g. $Q_k = \rho I$, where $\rho > 0$ is sufficiently small and I is the identity matrix, or adaptively updated.

Lemma 4.3.1. *If $\bar{x}^k$ is a solution of* CSDP($\bar{x}^k$) *(resp.* (4.3.4)*) linearized at $\bar{x}^k$, i.e. $\bar{x}^{k+1} = \bar{x}^k$, then it is a stationary point of* (NSDP) *(resp.* (4.2.5)*).*

Proof. It is sufficient to prove the second case with (4.3.4). Suppose that $\bar{\Lambda}^{k+1}$ is a multiplier associated with $\bar{x}^k$, substituting $\bar{x}^k$ into the generalized KKT condition (4.3.7) of (4.3.4) we obtain (4.2.6). Thus $\bar{x}^k$ is a stationary point of (4.2.5). □

Convergence analysis

We denote by $\mathcal{L}_f(\alpha) := \{x \in \mathcal{D} \mid f(x) \leq \alpha\}$ the lower level (sublevel) set of the objective function. Let us assume that $G_i(\cdot; y)$ is continuously differentiable in $\mathcal{L}_f(f(\bar{x}^0))$ for any y. We say that the *Robinson qualification* condition [28] for CSDP($\bar{x}^k$) holds at $\bar{x}$ if:

$$0 \in \mathrm{int}(G_i(\bar{x}; \bar{y}_i^k) + \mathbb{D}G_i F(\bar{x}; \bar{y}_i^k)(\Omega - \bar{x}) + \mathcal{S}_+^p), \quad i = 1, \ldots, m.$$

Similarly, we say that the *Robinson qualification* condition for (4.3.4) holds at $\bar{x}$ if:

$$0 \in \operatorname{int}(G_i(\bar{x}) + \mathbb{D}G_i(\bar{x}) - H_i(\bar{x}^k)(\Omega - \bar{x}) + \mathcal{S}_+^p), \quad i = 1, \ldots, m.$$

In order to prove the convergence of Algorithm 4.3.1, we require the following assumption, which is standard in nonlinear optimization.

Asumption A.4.3.6. *The set of KKT points of* (NSDP) (*resp.* (4.2.5)) *is nonempty. For a given y, the matrix-valued mappings $G_i(\cdot; y)$ (resp. $G_i(\cdot)$) are continuously differentiable on $\mathcal{L}_f(f(\bar{x}^0))$ for $i = 1, \ldots, m$. The convex programming subproblem* CSDP($\bar{x}^k$) (*resp.* (4.3.4)) *is solvable and the Robinson qualification condition holds at its solutions.*

We first show that the sequence $\{\bar{x}^k\}_{k\geq 0}$ generated by either Algorithm 4.3.1 or Algorithm 4.3.2 is a strictly descent sequence, i.e. $f(\bar{x}^{k+1}) < f(\bar{x}^k)$ for all $k \geq 0$. For a given matrix $Q \in \mathcal{S}_+^n$ and a vector z, we denote by $\|z\|_Q := [z^T Q z]^{1/2}$. Note that $\|\cdot\|_Q$ is only a norm if $Q \in \mathcal{S}_{++}^n$.

Lemma 4.3.2. *Let $\{(\bar{x}^k, \bar{\Lambda}^k)\}_{k\geq 0}$ be a sequence generated by Algorithm 4.3.1. Then:*

a) *The feasible set $\mathcal{D}(\bar{x}^k) \subseteq \mathcal{D}$ for all $k \geq 0$.*

b) *It is a feasible sequence, i.e. $\{\bar{x}^k\}_{k\geq 0} \subset \mathcal{D}$.*

c) $\bar{x}^{k+1} \in \mathcal{D}(\bar{x}^k) \cap \mathcal{D}(\bar{x}^{k+1})$.

d) *For any $k \geq 0$, it holds that:*

$$f(\bar{x}^{k+1}) \leq f(\bar{x}^k) - \frac{1}{2}\left\|\bar{x}^{k+1} - \bar{x}^k\right\|_{Q_k}^2 - \frac{\rho_f}{2}\left\|\bar{x}^{k+1} - \bar{x}^k\right\|^2,$$

where $\rho_f \geq 0$ is the convexity parameter of f.

Proof. For a given $\bar{x}^k$, we have $\bar{y}_i^k = \psi_i(\bar{x}^k)$ and $F_i(x) \preceq G_i(x; \bar{y}_i^k) \preceq 0$ for $i = 1, \ldots, m$. Thus if $x \in \mathcal{D}(\bar{x}^k)$ then $x \in \mathcal{D}$, the statement a) holds. Consequently, the sequence $\{\bar{x}^k\}$ is feasible to (NSDP) which is indeed the statement b). Since $\bar{x}^{k+1}$ is a solution of CSDP($\bar{x}^k$), it shows that $\bar{x}^{k+1} \in \mathcal{D}(\bar{x}^k)$. Now, we have to show that it belongs to $\mathcal{D}(\bar{x}^{k+1})$. Indeed, since $G_i(\bar{x}^{k+1}, \bar{y}_i^{k+1}) = F_i(\bar{x}^{k+1}) \preceq 0$ by Definition 4.3.1 for all $i = 1, \ldots, m$, we conclude $\bar{x}^{k+1} \in \mathcal{D}(\bar{x}^{k+1})$. The statement c) is proved. Finally, we prove d). Since $\bar{x}^{k+1}$ is the optimal solution of CSDP($\bar{x}^k$), we have $f(\bar{x}^{k+1}) + \frac{1}{2}\left\|\bar{x}^{k+1} - \bar{x}^k\right\|_{Q_k}^2 \leq f(x) + \frac{1}{2}(x - \bar{x}^k)^T Q_k (x - \bar{x}^k) - \frac{\rho_f}{2}\left\|x - \bar{x}^{k+1}\right\|^2$ for all $x \in \mathcal{D}(\bar{x}^k)$. Moreover, we have $\bar{x}^k \in \mathcal{D}(\bar{x}^k)$ due to c). By substituting $x = \bar{x}^k$ in the previous inequality we obtain the estimate d) of the lemma. □

Similar to Lemma 4.3.2, we also obtain the following result.

Lemma 4.3.3. *Suppose that* $\{(\bar{x}^k, \bar{\Lambda}^k)\}_{k\geq 0}$ *is a sequence generated by Algorithm 4.3.2. Then:*

a) *The following inequality holds for* $k \geq 0$*:*

$$f(\bar{x}^{k+1}) \leq f(\bar{x}^k) - \left\|\bar{x}^{k+1} - \bar{x}^k\right\|_{Q_k}^2 - \frac{\rho_f}{2}\left\|\bar{x}^{k+1} - \bar{x}^k\right\|_2^2, \tag{4.3.6}$$

where ρ_f *is the convexity parameter of* f*.*

b) *If there exists at least one constraint* i_0*,* $i_0 \in \{1, 2, \ldots, m\}$*, to be strictly feasible at* $\bar{x}^k$*, i.e.* $G_{i_0}(\bar{x}^k) - H_{i_0}(\bar{x}^k) \prec 0$*, then* $f(\bar{x}^{k+1}) < f(\bar{x}^k)$ *provided that* $\bar{\Lambda}_{i_0}^{k+1} \succ 0$*.*

c) *If* $Q_k \in \mathcal{S}_{++}^n$ *then* $\Delta\bar{x}^k := \bar{x}^{k+1} - \bar{x}^k$ *is a sufficient descent direction of* (4.2.5)*, i.e.* $f(\bar{x}^{k+1}) - f(\bar{x}^k) \leq -\left\|\Delta\bar{x}^k\right\|_{Q_k}^2 < 0$ *for all* $k \geq 0$*.*

Proof. For any matrices $A, B \in \mathcal{S}_+^p$, we have $A \circ B \geq 0$. From *Step 1* of Algorithm 4.3.2, we have that $\bar{x}^{k+1}$ is a solution of the convex subproblem (4.3.4) and $\bar{\Lambda}^{k+1}$ is the corresponding multiplier, under Assumption **A.4.3.6**, they must satisfy the following generalized Kuhn-Tucker condition:

$$\left\{\begin{array}{l} 0 \in \nabla f(\bar{x}^{k+1}) + Q_k(\bar{x}^{k+1} - \bar{x}^k) + \left\{\sum_{i=1}^m \mathbb{D}[G_i(x) \right. \\ \left. -H_i(\bar{x}^k) - \mathbb{D}H_i(\bar{x}^k)(x - \bar{x}^k)]|_{\bar{x}^{k+1}}\right\}^* \bar{\Lambda}_i^{k+1} + \mathcal{N}_\Omega(\bar{x}^{k+1}), \\ G_i(\bar{x}^{k+1}) - H_i(\bar{x}^k) - \mathbb{D}H(\bar{x}^k)(\bar{x}^{k+1} - \bar{x}^k) \preceq 0,\ \bar{\Lambda}_i \succeq 0, \\ \left[G_i(\bar{x}^{k+1}) - H_i(\bar{x}^k) - \mathbb{D}H(\bar{x}^k)(\bar{x}^{k+1} - \bar{x}^k)\right] \circ \bar{\Lambda}_i^{k+1} = 0. \end{array}\right. \tag{4.3.7}$$

Noting that $\mathbb{D}\left[G_i(x) - H_i(\bar{x}^k) - \mathbb{D}H_i(\bar{x}^k)(x - \bar{x}^k)\right]|_{x=\bar{x}^{k+1}} = \mathbb{D}G_i(\bar{x}^{k+1}) - \mathbb{D}H_i(\bar{x}^k)$ for $i = 1, \ldots m$, it follows from the first line of (4.3.7) and the convexity of f that:

$$\begin{array}{l} f(y) - f(\bar{x}^{k+1}) + \left\{\sum_{i=1}^m [\mathbb{D}G_i(\bar{x}^{k+1}) - \mathbb{D}H_i(\bar{x}^k)]^* \bar{\Lambda}_i^{k+1}\right\}^T (y - \bar{x}^{k+1}) \\ \geq \left\{\nabla f(\bar{x}^{k+1}) + \sum_{i=1}^m [\mathbb{D}G_i(\bar{x}^{k+1}) - \mathbb{D}H_i(\bar{x}^k)]^* \bar{\Lambda}_i^{k+1}\right\}^T (y - \bar{x}^{k+1}) \\ + \frac{\rho_f}{2}\left\|y - \bar{x}^{k+1}\right\|_2^2 \\ \geq \frac{\rho_f}{2}\left\|y - \bar{x}^{k+1}\right\|_2^2 + (y - \bar{x}^{k+1})^T Q_k(\bar{x}^k - \bar{x}^{k+1}),\ \forall y \in \Omega. \end{array} \tag{4.3.8}$$

On the other hand, we have:

$$\begin{aligned}&\left\{[\mathbb{D}G_i(\bar{x}^{k+1}) - \mathbb{D}H_i(\bar{x}^k)]^*\bar{\Lambda}_i^{k+1}\right\}^T(y-\bar{x}^{k+1})\\&= \bar{\Lambda}_i^{k+1}\circ[\mathbb{D}G_i(\bar{x}^{k+1})(y-\bar{x}^{k+1}) - \mathbb{D}H_i(\bar{x}^k)(y-\bar{x}^{k+1})].\end{aligned} \tag{4.3.9}$$

Since G_i and H_i are psd-convex, by applying Lemma 4.2.1 we have:

$$\begin{aligned}&G_i(\bar{x}^k) - G_i(\bar{x}^{k+1}) \succeq \mathbb{D}G_i(\bar{x}^{k+1})(\bar{x}^k - \bar{x}^{k+1}),\\ \text{and}\quad &H_i(\bar{x}^{k+1}) - H_i(\bar{x}^k) \succeq \mathbb{D}H_i(\bar{x}^k)(\bar{x}^{k+1} - \bar{x}^k),\ i=1,\ldots,m.\end{aligned}$$

Summing up these inequalities we obtain:

$$\begin{aligned}&G_i(\bar{x}^k) - H_i(\bar{x}^k) - [G_i(\bar{x}^{k+1}) - H_i(\bar{x}^{k+1})]\\&\succeq [\mathbb{D}G_i(\bar{x}^{k+1})(\bar{x}^k - \bar{x}^{k+1}) - \mathbb{D}H_i(\bar{x}^k)(\bar{x}^k - \bar{x}^{k+1})].\end{aligned}$$

Using the fact that $\bar{\Lambda}_i^{k+1} \succeq 0$, the last inequality implies that:

$$\begin{aligned}&\bar{\Lambda}_i^{k+1}\circ\left\{G_i(\bar{x}^k) - H_i(\bar{x}^k) - [G_i(\bar{x}^{k+1}) - H_i(\bar{x}^{k+1})]\right\}\\&\succeq \bar{\Lambda}_i^{k+1}\circ[\mathbb{D}G_i(\bar{x}^{k+1})(\bar{x}^k - \bar{x}^{k+1}) - \mathbb{D}H_i(\bar{x}^k)(\bar{x}^k - \bar{x}^{k+1})].\end{aligned} \tag{4.3.10}$$

Substituting $y = \bar{x}^k$ into (4.3.8) and then combining the consequence, (4.3.9), (4.3.10) and the last line of (4.3.7) to obtain:

$$\begin{aligned}&f(\bar{x}^k) - f(\bar{x}^{k+1}) + \sum_{i=1}^m \bar{\Lambda}_i^{k+1}\circ[G_i(\bar{x}^k) - H_i(\bar{x}^k)]\\&\geq \frac{\rho_f}{2}\left\|\bar{x}^{k+1} - \bar{x}^k\right\|_2^2 + \left\|\bar{x}^{k+1} - \bar{x}^k\right\|_{Q_k}^2.\end{aligned} \tag{4.3.11}$$

Now, since $\bar{x}^k$ is the solution of the convex subproblem (4.3.4) linearized at $\bar{x}^{k-1}$. One has $G_i(\bar{x}^k) - H_i(\bar{x}^k) \preceq 0$. Moreover, since $\bar{\Lambda}_i^{k+1} \succeq 0$, we have $\bar{\Lambda}_i^{k+1}\circ\left[G_i(\bar{x}^k) - H_i(\bar{x}^k)\right] \leq 0$. Substituting this inequality into (4.3.11), we obtain:

$$f(\bar{x}^k) - f(\bar{x}^{k+1}) \geq \frac{\rho_f}{2}\left\|\bar{x}^k - \bar{x}^{k+1}\right\|_2^2 + \left\|\bar{x}^{k+1} - \bar{x}^k\right\|_{Q_k}^2.$$

This inequality is indeed (4.3.6) which proves the item a). If there exists at least one $i_0 \in \{1,\ldots,m\}$ such that $G_{i_0}(\bar{x}^k) - H_{i_0}(\bar{x}^k) \prec 0$ and $\bar{\Lambda}_{i_0}^{k+1} \succ 0$ then

$\Lambda_{i_0}^{k+1} \circ \left[G_{i_0}(\bar{x}^k) - H_{i_0}(\bar{x}^k)\right] < 0$. Substituting this inequality into (4.3.11) we conclude that $f(\bar{x}^{k+1}) < f(\bar{x}^k)$ which proves item b). The last statement c) follows directly from the inequality (4.3.6). □

The following theorem shows the convergence of Algorithms 4.3.1 (resp. 4.3.2) to a stationary point of (NSDP) (resp. (4.2.5)).

Theorem 4.3.2. *Suppose that Assumptions* ***A.4.2.5*** *and* ***A.4.3.6*** *are satisfied. Suppose further that the lower level set* $\mathcal{L}_f(f(\bar{x}^0))$ *is bounded. Let* $\{(\bar{x}^k, \bar{\Lambda}^k)\}_{k\geq 1}$ *be an infinite sequence generated by Algorithm 4.3.1 (resp. Algorithm 4.3.2) starting from* $\bar{x}^0 \in ri(\mathcal{D})$*. Assume that* $\lambda_{\max}(Q_k) \leq M < +\infty$*. Then if either* f *is strongly convex or* $\lambda_{\min}(Q_k) \geq \rho > 0$ *for* $k \geq 0$ *then every accumulation point* $(\bar{x}^*, \bar{\Lambda}^*)$ *of* $\{(\bar{x}^k, \bar{\Lambda}^k)\}$ *is a KKT point of* (NSDP) *(resp.* (4.2.5)*). Moreover, if the set of the KKT points of* (NSDP) *(resp.* (4.2.5)*) is finite then the whole sequence* $\left\{(\bar{x}^k, \bar{\Lambda}^k)\right\}$ *converges to a KKT point of* (NSDP) *(resp.* (4.2.5)*).*

Proof. It is sufficient to prove the first case for Algorithm 4.3.1. The second case can be proved similarly. Let $\mathcal{M}(\bar{x}^0) := \left\{\bar{x}^k \mid k \geq 0\right\}$ be the sequence of sample points generated by Algorithm 4.3.1 starting from $\bar{x}^0$. The idea of the proof is to apply Zangwill's convergence theorem [216, p. 91]. For a given $x \in \Omega$, let us define the following mapping:

$$\mathcal{S}(x, Q) := \underset{y\in\Omega}{\operatorname{argmin}} \left\{ f(y) + \frac{1}{2}(y-x)^T Q(y-x) \mid G_i(y; \psi_i(x)) \preceq 0,\ i = 1, \ldots, m \right\}.$$

Then, $\mathcal{S}(\cdot,\cdot)$ is a multivalued mapping and it can be considered as the solution mapping of the convex subproblem CSDP($\bar{x}^k$). Note that the sequence $\{\bar{x}^k\}_{k\geq 0}$ generated by Algorithm 4.3.1 satisfies $\bar{x}^{k+1} \in \mathcal{S}(\bar{x}^k, Q_k)$ for all $k \geq 0$. We first prove that $\mathcal{S}$ is a closed mapping. Indeed, by Assumption **A.4.3.6**, CSDP($\bar{x}^k$) is feasible. Moreover, CSDP($\bar{x}^k$) is strictly convex. Hence, $\mathcal{S}(\bar{x}^k, Q_k) = \left\{\bar{x}^{k+1}\right\}$, which is obviously closed. On the other hand, since f is either strongly convex or $\rho_k \equiv \rho > 0$ for all $k \geq 0$ and $Q_k \equiv Q$ is full-row-rank, it follows from Lemma 4.3.2 that the objective function f is strictly monotone on $\mathcal{M}(\bar{x}^0)$, i.e. $f(\bar{x}^{k+1}) < f(\bar{x}^k)$ for all $\bar{x}^k, \bar{x}^{k+1} \in \mathcal{M}(\bar{x}^0)$. Since $\mathcal{M}(\bar{x}^0) \subseteq \mathcal{L}_f(f(\bar{x}^0))$ and $\mathcal{L}_f(f(\bar{x}^0))$ is compact, $\mathcal{M}(\bar{x}^0)$ is also compact. By applying [132, Theorem 2] we conclude that every limit point of the sequence $\left\{\bar{x}^k\right\}_{k\geq 0}$ belongs to the set of stationary points S^*. Moreover, since f is bounded from below and either f is strongly convex or $\rho > 0$ and Q is full-row rank, it follows from (4.3.6) that $\lim_{k\to\infty} \left\|\bar{x}^{k+1} - \bar{x}^k\right\| = 0$. Therefore, S^* is connected and if S^* is finite then the whole sequence $\{\bar{x}^k\}_{k\geq 0}$ converges to $\bar{x}^*$ in S^*. □

We note that the assumptions of Theorem 4.3.2 are rather restrictive. One possibility is to relax these assumptions to obtain a more general results as done in [177] for the scalar case.

4.4 Conclusion

We have presented a generalized inner convex approximation method for solving a class of nonconvex SDP problems. We have also provided some explicit formulas to generate psd-convex overestimates of a given nonconvex matrix-valued mapping. Alternatively, we have developed a second algorithm which we call generalized convex-concave decomposition algorithm for solving convex SDP problems with generalized concave-concave constraints. This method can be considered on the one hand as a variant of the first algorithm and, on the other hand, as a generalization of classical convex-concave procedures for scalar DC programming. The convergence of both algorithms has been proved under standard assumptions used in nonlinear SDP and a fundamental assumption on the nonemptiness of the interior of the feasible set. In principle, these algorithms can be applied to solve any nonconvex SDP problem where we can be able to either find a psd-convex overestimate or a psd-convex-concave decomposition for the nonconvex matrix-valued mappings.

Chapter 5

BMI optimization in robust controller design

5.1 BMI optimization in static feedback control

In this chapter we focus on the optimization problems with bilinear matrix inequality (BMI) constraints derived from the following linear, time-invariant system:

$$\begin{cases} \dot{x} = Ax + B_1 w + Bu, \\ z = C_1 x + D_{11} w + D_{12} u, \\ y = Cx + D_{21} w, \end{cases} \tag{5.1.1}$$

where $x \in \mathbf{R}^{n_x}$ is the state vector, $w \in \mathbf{R}^{n_w}$ is the performance input, $u \in \mathbf{R}^{n_u}$ is the input vector, $z \in \mathbf{R}^{n_z}$ is the performance output, $y \in \mathbf{R}^{n_y}$ is the physical output vector, $A \in \mathbf{R}^{n_x \times n_x}$ is state matrix, $B \in \mathbf{R}^{n_x \times n_u}$ is input matrix and $C \in \mathbf{R}^{n_y \times n_x}$ is the output matrix. By using a static feedback controller of the form $u = Fy$ with $F \in \mathbf{R}^{n_u \times n_y}$, we can write the closed-loop system as follows:

$$\begin{cases} \dot{x}_F = A_F x_F + B_F w, \\ z = C_F x_F + D_F w, \end{cases} \tag{5.1.2}$$

where $A_F := A + BFC$, $B_F := B_1 + BFD_{21}$, $C_F := C_1 + D_{12}FC$ and $D_F := D_{11} + D_{12}FD_{21}$. Finding a fixed order controller that satisfies a given criterion such as stabilization, minimizing the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norm or mixed $\mathcal{H}_2/\mathcal{H}_\infty$ synthesis leads to an optimization problem with BMI constraints by means of, e.g. Lyapunov's theory. These problems have been studied in several research

papers both from theoretical aspects and numerical methods, see e.g. [7, 24, 33, 44, 76, 99, 116, 117, 121, 150, 186].

As contributions of this chapter, we first show that both algorithms, Algorithms 4.3.1 and 4.3.2, developed in Chapter 4 can be applied to solve the following optimization problems:

1. Sparse linear static output feedback controller design [92];
2. Spectral abscissa and pseudospectral abscissa optimization [33, 35, 119, 205];
3. $\mathcal{H}_2$ control;
4. $\mathcal{H}_\infty$ control;
5. and mixed $\mathcal{H}_2/\mathcal{H}_\infty$ synthesis control.

We implement both algorithms in Matlab and test their performance by using the data from [92, 151] and the COMPl$_e$ib library [119]. We propose several heuristic procedures to find a starting point for our algorithms.

Outline of Chapter 5. The outline of this chapter is as follows. In the next section, we present the implementation details of the algorithms. In Section 5.3, we study the optimization of sparse linear static output feedback controller design, spectral abscissa and pseudospectral abscissa optimization problems. Sections 5.4 and 5.5 deal with the BMI optimization problems of $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control, respectively. Section 5.6 presents the BMI optimization problem of mixed $\mathcal{H}_2/\mathcal{H}_\infty$ synthesis control. The last section gives some conclusion for this chapter.

5.2 Implementation details

We first present the following additional results which will be used in the sequel.

Definition 5.2.1. *A mapping* $F : \mathbb{R}^{p\times q} \times \mathcal{S}^p \to \mathcal{S}^p$ *given by* $F(X,Y) := XQ^{-1}X^T - Y$, *where* $Q \in \mathcal{S}^q_{++}$, *is called a Schur psd-convex*[1] *mapping.*

The following results will be used to transform a Schur psd-convex constraint to an LMI constraint.

[1] Due to Schur's complement form

Lemma 5.2.1.

a) The mappings $f(X) := X^T X$ *and* $g(X) := XX^T$ *are psd-convex on* $\mathbb{R}^{m\times n}$. *The mapping* $f(X) := X^{-1}$ *is psd-convex on* $\mathcal{S}^p_{++}$.

b) Suppose that $A \in \mathcal{S}^n$. *Then the matrix inequality* $BB^T - A \prec (\preceq)\ 0$ *is equivalent to:*

$$\begin{bmatrix} A & B \\ B^T & I \end{bmatrix} \succ (\succeq)\ 0. \tag{5.2.1}$$

c) Suppose that $A \in \mathcal{S}^n$, $D \succ 0$. *Then we have:*

$$\begin{bmatrix} A-BB^T & C \\ C^T & D \end{bmatrix} \succ (\succeq)\ 0 \Longleftrightarrow \begin{bmatrix} A & B & C \\ B^T & I & O \\ C^T & O & D \end{bmatrix} \succ (\succeq)\ 0. \tag{5.2.2}$$

The proof of this lemma is trivial by applying Schur's complement and Lemma 4.2.1 [30]. We omit the proof details here.

Since all the problems addressed in this chapter possess at least one BMI constraint, we propose two general schemes to treat these problems based on Algorithms 4.3.1 and 4.3.2, respectively.

Scheme S.5.2.1.(*For Algorithm 4.3.1*).

Step 1: Find a psd-convex overestimate $G_i(x; y_i)$ of $F_i(x)$ w.r.t. the parameterization $y_i = \psi_i(x)$ for $i = 1, \ldots, m$ (see, e.g. *Example 4.3.3*).

Step 2 (Phase 1): Find a starting point $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$.

Step 3: Whenever the iteration point $\bar{x}^k$ is available, we define a procedure to transform the convex semidefinite programming problem CSDP($\bar{x}^k$) into an optimization problem with LMI constraints.

Step 4 (Phase 2): Apply Algorithm 4.3.1 with the procedure at *Step 3* and an SDP solver to solve the given problem.

End.

Similar to **S.5.2.1**, the following scheme is using Algorithm 4.3.2.

Scheme S.5.2.2.(*For Algorithm 4.3.2*).

Step 1: Find a convex-concave decomposition of the BMI constraints as $G(x) - H(x) \preceq 0$.

Step 2 (*Phase 1*): Find a starting point $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$.

Step 3: Whenever the iteration point $\bar{x}^k$ is available, we linearize the concave part to obtain the convex constraint $G(x) - H_k(x) \preceq 0$, where H_k is the linearization of H at $\bar{x}^k$, to form the convex semidefinite programming problem of the form (4.3.4).

Step 4: Define a procedure to reformulate each convex constraint of problem (4.3.4) as an LMI constraint by means of Lemma 5.2.1.

Step 5 (*Phase 2*): Apply Algorithm 4.3.2 with the procedures at *Steps 3 and 4* and an SDP solver to solve the given problem.

End.

Next, we consider the stopping criterion to terminate Algorithms 4.3.1 and 4.3.2. We can terminate these algorithms if one of the following conditions is satisfied:

a) the subproblems CSDP($\bar{x}^k$) or (4.3.4) encounters a numerical problem;

b) the maximum number of iterations, $K_{\max}$, is reached;

c) $\left\|\bar{x}^{k+1} - \bar{x}^k\right\|_\infty / \max\left\{\left\|\bar{x}^k\right\|_\infty, 1\right\} \leq \varepsilon_d$ for a given tolerance $\varepsilon_d > 0$;

d) the objective values are not significantly improved after two successive iterations, i.e. $\left|f^{k+1} - f^k\right| \leq \varepsilon_f \max\left\{\left|f^k\right|, 1\right\}$ for some $k = \bar{k}$ and $k = \bar{k} + 1$, where $f^k := f(\bar{x}^k)$ and ε_f is a given tolerance.

In the following tests, we chose $\varepsilon_d = 10^{-3}$ and $\varepsilon_f = 10^{-4}$.

We implemented both schemes **S.5.2.1** and **S.5.2.2** in Matlab 7.11.0 (R2010b) running on an Intel® Core (TM)2 Quad CPU Q6600, 2.4GHz PC Desktop with 3Gb RAM. The algorithms have been tested by using the system data from [92, 151] and the `COMPl`$_{\text{e}}$`ib` library [119]. We used the YALMIP package [124] as a modeling language and SeDuMi 1.1 as an SDP solver [180] to solve the LMI optimization problems arising in the schemes **S.5.2.1** and **S.5.2.2** at the initial phase (Phase 1) and the subproblems CSDP($\bar{x}^k$) and (4.3.4). The Matlab codes can be found at: `http://www.kuleuven.be/optec/software/BMIsolver`.

We also benchmarked our methods with various examples and compared our results with HIFOO [86] and PENBMI [97] for all control problems. HIFOO is an open-source Matlab package for fixed-order controller design. It computes a fixed-order controller by using a hybrid algorithm for nonsmooth, nonconvex optimization based on quasi-Newton updating and gradient sampling techniques.

PENBMI [97] is a commercial software for solving optimization problems with quadratic objective function and BMI constraints, which is freely licensed for academic purposes. We initialized the initial controller for HIFOO and the BMI parameters for PENBMI at the initial values of our methods. As shown in [150], we can reformulate the spectral abscissa feasibility problem as a rank constrained LMI feasibility problem. Therefore, we also compared our results with LMIRank [150] (a MATLAB toolbox for solving rank constrained LMI feasibility problems) by implementing a simple procedure for solving the spectral abscissa optimization problem.

5.3 Linear output-feedback controller design

In this section, we consider two cases, namely sparse linear controller and abscissa optimization problems. Then, we show that the methods can be applied to solve an optimization problem of pseudospectral abscissa.

Sparse linear constant output feedback design

Let us consider a BMI optimization problem of sparse linear constant output-feedback design given as:

$$\begin{array}{ll} \min\limits_{\alpha,P,F} & \left\{ f(\alpha,P,F) := -\sigma\alpha + \sum_{i=1}^{n_u}\sum_{j=1}^{n_y} |F_{ij}| \right\} \\ \text{s.t} & (A+BFC)^T P + P(A+BFC) + 2\alpha P \prec 0, \\ & P = P^T,\ P \succ 0. \end{array} \tag{5.3.1}$$

Here, matrices A, B, C are given with appropriate dimensions, P and F are referred to as variables and $\sigma > 0$ is a weighting parameter. The objective function consists of two terms: the first term $\sigma\alpha$ is to stabilize the system (or to maximize the decay rate) and the second one is to ensure the sparsity of the gain matrix F. This problem is a modification of the first example in [92]. Let us illustrate the scheme **S.5.2.1** for solving this problem.

Step 1: Let $B_F := A + BFC + \alpha I$, where I is the identity matrix. Then, by Example 4.2.1 we can write:

$$(A+BFC)^T P + P(A+BFC) + 2\alpha P = B_F^T P + P B_F$$

$$= B_F^T B_F + P^T P - (B_F - P)^T (B_F - P), \tag{5.3.2}$$

$$= \frac{1}{2}\left[(B_F + P)^T (B_F + P) - (B_F - P)^T (B_F - P) \right]. \tag{5.3.3}$$

In our implementation, we used the decomposition (5.3.3). If we denote by:

$$\begin{aligned} & G(\alpha,P,F) := \frac{1}{2}(B_F+P)^T(B_F+P), \\ \text{and} \quad & \\ & H(\alpha,P,F) := \frac{1}{2}(B_F-P)^T(B_F-P), \end{aligned} \tag{5.3.4}$$

then the BMI constraint in (5.3.1) can be written equivalently as a psd-convex-concave matrix inequality constraint (of a variable x formed from (α,P,F) as $x := (\alpha, \mathbf{vec}(P)^T, \mathbf{vec}(F)^T)^T$) as follows:

$$G(\alpha,P,F) - H(\alpha,P,F) \prec 0. \tag{5.3.5}$$

Note that the objective function of (5.3.1) is convex but nonsmooth which is not directly suitable for the sequential SDP approach in [44], but, the nonconvex problem (5.3.1) can be reformulated in the form of (NSDP) by using slack variables.

Steps 2-5: The implementation is carried out as follows:

Phase 1. (*Determine a starting point* $\bar{x}^0 \in ri(\mathcal{D})$). Set $F^0 := 0$, $\alpha^0 := -\alpha_0(A^T+A)/2$ where $\alpha_0(A^T+A)$ is the maximum real part of the eigenvalues of the matrix A^T+A, and compute $P = P^0$ as the solution of the LMI feasibility problem:

$$(A+BF^0C)^TP + P(A+BF^0C) + 2\alpha^0 P \prec 0. \tag{5.3.6}$$

The above choice for (α^0, F^0) originates from the property that $P^0 = I$ renders the left hand size of (5.3.6) negative semidefinite (but not negative definite).

Phase 2. Perform Algorithms 4.3.1 or 4.3.2 with the starting point $\bar{x}^0$ found at Phase 1 by applying Schemes **S.5.2.1** or **S.5.2.2**, respectively.

As an example, let us now illustrate *Step 4* of Scheme **S.5.2.2**. After linearizing the concave part of the convex-concave reformulation of the last BMI constraint in (5.3.1) at (F^k, P^k, α^k) we obtain the linearization:

$$(A+BFC+\alpha I+P)^T(A+BFC+\alpha I+P) - H_k(F,P,\alpha) \prec 0, \tag{5.3.7}$$

where $H_k(F,P,\alpha)$ is a linear mapping of F, P and α. Now, by applying Lemma 5.2.1, (5.3.7) can be transformed into an LMI constraint of the form:

$$\begin{bmatrix} H_k(F,P,\alpha) & (A+BFC+\alpha I+P)^T \\ (A+BFC+\alpha I+P) & I \end{bmatrix} \succ 0.$$

With the above approach we solved problem (5.3.1) for the same system data as in [92]. Here, matrices A, B and C are given, respectively as:

$$A = \begin{bmatrix} -2.45 & -0.90 & 1.53 & -1.26 & 1.76 \\ -0.12 & -0.44 & -0.01 & 0.69 & 0.90 \\ 2.07 & -1.20 & -1.14 & 2.04 & -0.76 \\ -0.59 & 0.07 & 2.91 & -4.63 & -1.15 \\ -0.74 & -0.23 & -1.19 & -0.06 & -2.52 \end{bmatrix}, \quad B = \begin{bmatrix} 0.81 & -0.79 & 0.00 & 0.00 & -0.95 \\ -0.34 & -0.50 & 0.06 & 0.22 & 0.92 \\ -1.32 & 1.55 & -1.22 & -0.77 & -1.14 \\ -2.11 & 0.32 & 0.00 & -0.83 & 0.59 \\ 0.31 & -0.19 & -1.09 & 0.00 & 0.00 \end{bmatrix},$$

and

$$C = \begin{bmatrix} 0.00 & 0.00 & 0.16 & 0.00 & -1.78 \\ 1.23 & -0.38 & 0.75 & -0.38 & -0.00 \\ 0.46 & 0.00 & -0.05 & 0.00 & 0.00 \\ 0.00 & -0.12 & 0.23 & -0.12 & 1.14 \end{bmatrix}.$$

The weighting parameter σ was chosen by $\sigma = 3$. In this example, Algorithm 4.3.2 was terminated after 15 iterations, whereas the objective function was not significantly improved. However, after the 2^{nd} iteration, matrix F only has 3 nonzero elements, while the decay rate α is 1.17316. This value is much higher than the one reported in [92], $\alpha = 0.3543$ after 6 iterations. We obtained the gain matrix F as:

$$F = \begin{bmatrix} 0.6540 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -0.4872 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.1280 & 0.0000 & 0.0000 \end{bmatrix}.$$

With this matrix, the maximum real part of the eigenvalues of the closed-loop matrix in (5.1.2), $A_F := A + BFC$, is $\alpha_0(A_F) := -1.40706$. Simultaneously, $\alpha_0(A_F^T P + P A_F + 2\alpha P) = -0.327258 < 0$ and $\alpha_0(P) = 0.587574 > 0$. Note that $\alpha_0(A_F) \neq -\alpha$ due to the in-activeness of the BMI constraint in (5.3.1) at the 2^{nd} iteration.

Spectral abscissa and pseudo-spectral abscissa optimization

One popular problem in control theory is to optimize the spectral abscissa of the closed-loop system $\dot{x} = (A + BFC)x$. Briefly, this problem is presented as an unconstrained optimization problem of the form:

$$\min_{F \in \mathbb{R}^{n_u \times n_y}} \alpha_0(A + BFC), \tag{5.3.8}$$

where $\alpha_0(A+BFC) := \sup\{\text{Re}(\lambda) \mid \lambda \in \Lambda(A+BFC)\}$ is the spectral abscissa of $A + BFC$, $\text{Re}(\lambda)$ denotes the real part of $\lambda \in \mathbb{C}$ and $\Lambda(A + BFC)$ is the spectrum of $A + BFC$. Problem (5.3.8) has many drawbacks in terms of

numerical solution due to the nonsmoothness and non-Lipschitz continuity of the objective function α_0 [34].

In order to apply the method developed in Chapter 4, problem (5.3.8) is reformulated as an optimization problem with BMI constraints of the form, see, e.g. [34, 118]:

$$\begin{cases} \max\limits_{P,F,\beta} \ \beta \\ \text{s.t.} \ \ (A+BFC)^T P + P(A+BFC) + 2\beta P \prec 0, \\ \qquad P = P^T, \ P \succ 0. \end{cases} \tag{5.3.9}$$

Here, matrices $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$ and $C \in \mathbb{R}^{n_y \times n_x}$ are given. Matrices $P \in \mathbb{R}^{n_x \times n_x}$ and $F \in \mathbb{R}^{n_u \times n_y}$ and the scalar β are considered as variables. If the optimal value of (5.3.9) is strictly positive then the closed-loop feedback controller $u = Fy$ stabilizes the linear system $\dot{x} = (A + BFC)x$.

Problem (5.3.9) is very similar to (5.3.1). Therefore, by using the same trick as in (5.3.1), we can reformulate (5.3.9) in the form of (NSDP) or (4.2.5). More precisely, if we define $B_F := A + BFC + \beta I$ then the bilinear matrix mapping $A_F^T P + P A_F$ can be represented as a psd-convex-concave decomposition of the form (5.3.3) and problem (5.3.9) can be rewritten in the form of (4.2.5).

In this test, we implemented Algorithms 4.3.1 and 4.3.2 for solving this resulting problem by using the same parameters and the stopping criterion as in the previous subsection. In addition, we regularized the objective function by adding the term $\frac{\rho_F}{2}\left\|F - F^k\right\|_F^2 + \frac{\rho_P}{2}\left\|P - P^k\right\|_F^2$, with $\rho_F = \rho_P = 10^{-2}$. The maximum number of iterations $K_{\max}$ was set to 200.

We tested for several problems in COMPl$_e$ib and compared our results with the ones reported by HIFOO, PENBMI and LMIRank. For LMIRank, we implemented the algorithm proposed in [150]. We initialized the value of the decay rate α^0 at 10^{-4} and performed an iterative loop to increase α as $\alpha^{k+1} := \alpha^k + 0.1$. The algorithm was terminated if either the problems (12) or (21) in [150] with a corresponding value of α could not be solved or the maximum number of iterations $K_{\max} := 200$ was reached.

The numerical results of the four algorithms are reported in Table 5.1. Here, we initialized the algorithm in HIFOO at the same initial guess $F^0 = 0$. Since PENBMI and our methods solve the same BMI problems, they were initialized by the same initial values for P, F and β.

The notation in Table 5.1 consists of: `Name` is the name of problems, $\alpha_0(A)$, $\alpha_0(A_F)$ are the maximum real part of the eigenvalues of the open-loop and closed-loop matrices A, A_F, respectively; `iter` is the number of iterations, `time[s]` is the CPU time in seconds. The columns titled HIFOO, LMIRank

Table 5.1: Computational results for (5.3.9) in COMPl$_e$ib

Problem		Other Results, $\alpha_0(A_F)$			Algorithm 4.3.1			Algorithm 4.3.2		
Name	$\alpha_0(A)$	HIFOO	LMIRANK	PENBMI	$\alpha_0(A_F)$	Iter	time[s]	$\alpha_0(A_F)$	Iter	time[s]
AC1	0.000	**-0.2061**	**-8.4766**	**-7.0758**	**-0.7814**	55	19.510	**-0.8644**	62	23.580
AC4	2.579	**-0.0500**	**-0.0500**	**-0.0500**	**-0.0500**	14	4.380	**-0.0500**	14	6.060
AC5[a]	0.999	**-0.7746**	**-1.8001**	**-2.0438**	**-0.7389**	37	12.030	**-0.7389**	28	10.200
AC7	0.172	**-0.0322**	**-0.0204**	**0.0896**	**-0.0502**	90	80.710	**-0.0766**	200	95.830
AC8	0.012	**-0.1968**	**-0.4447**	**0.4447**	**-0.0640**	40	32.340	**-0.0755**	24	12.110
AC9	0.012	**-0.3389**	**-0.5230**	**-0.4450**	**-0.3926**	200	217.230	**-0.4053**	100	55.460
AC11	5.451	**-0.0003**	**-5.0577**	-	**-3.1573**	181	73.660	**-5.5960**	200	81.230
AC12	0.580	**-10.8645**	**-9.9658**	**-1.8757**	**-0.2948**	200	71.200	**-0.5890**	200	61.920
HE1	0.276	**-0.2457**	**-0.2071**	**-0.2468**	**-0.2134**	200	58.580	**-0.2241**	200	56.890
HE3	0.087	**-0.4621**	**-2.3009**	**-0.4063**	**-0.8380**	57	54.720	**-0.9936**	200	98.730
HE4	0.234	**-0.7446**	**-1.9221**	**-0.0909**	**-0.8375**	88	70.770	**-0.8647**	63	27.620
HE5	0.234	**-0.1823**	-	**-0.2932**	**-0.0609**	200	181.470	**-0.1115**	200	86.550
HE6	0.234	**-0.0050**	**-0.0050**	**-0.0050**	**-0.0050**	18	106.840	**-0.0050**	12	29.580
REA1	1.991	**-16.3918**	**-5.9736**	**-1.7984**	**-2.8932**	200	74.560	**-4.2792**	200	70.370
REA2	2.011	**-7.0152**	**-10.0292**	**-3.5928**	**-1.9514**	43	13.120	**-2.1778**	40	13.360
REA3	0.000	**-0.0207**	**-0.0207**	**-0.0207**	**-0.0207**	161	311.490	**-0.0207**	200	267.160
DIS2	1.675	**-6.8510**	**-10.1207**	**-8.3289**	**-8.3419**	44	12.600	**-8.4540**	28	9.430
DIS4	1.442	**-36.7203**	**-0.5420**	**-92.2842**	**-5.4467**	89	40.120	**-8.2729**	95	40.200
WEC1	0.008	**-8.9927**	**-8.7350**	**-0.9657**	**-0.8568**	68	76.000	**-0.8972**	200	121.300
IH	0.000	**-0.5000**	**-0.5000**	**-0.5000**	**-0.5000**	11	82.730	**-0.5000**	7	23.670
CSE1	0.000	**-0.4509**	**-0.4844**	**-0.4490**	**-0.2949**	200	1815.400	**-0.3093**	81	219.910
TF1	0.000	-	-	**-0.0618**	**-0.0704**	200	154.430	**-0.1598**	87	34.960
TF2	0.000	-	-	**-1.0e-5**	**-1.0e-5**	12	10.130	**-1.0e-5**	8	4.220
TF3	0.000	-	-	**-0.0032**	**-0.0032**	95	70.980	**-0.0031**	93	35.000
NN1	3.606	**-3.0458**	**-4.4021**	**-4.3358**	**0.1769**	200	59.230	**-1.5574**	200	57.370
NN5[a]	0.420	**-0.0942**	**-0.0057**	**-0.0942**	**-0.0490**	200	154.160	**-0.0722**	200	79.210
NN9	3.281	**-2.0789**	**-0.7048**	-	**0.0991**	44	13.860	**-0.0279**	33	11.880
NN13	1.945	**-3.2513**	**-4.5310**	**-9.0741**	**-0.2783**	32	12.430	**-3.4412**	181	64.500
NN15	0.000	**-6.9983**	**-11.0743**	**-0.0278**	**-1.0409**	200	60.930	**-1.0424**	200	58.440
NN17	1.170	**-0.6110**	**-0.5130**	-	**-0.5991**	132	34.820	**-0.6008**	99	27.190

and PENBMI give the maximum real part of the eigenvalues of the closed-loop system for a static output feedback controller computed by available software HIFOO [86], LMIRank [150] and PENBMI [97], respectively. Our results can be found in the sixth and ninth columns. The entries with a dashed sign indicate that there is no feasible solution found. Algorithms 4.3.1 and 4.3.2 fail or make only slow progress towards a local solution with 6 problems: AC18, DIS5, PAS, NN6, NN7, NN12 in COMPl$_e$ib. Problems AC5 and NN5 were initialized with a different matrix F^0 to avoid numerical problems.

Note that both Algorithms 4.3.1 and 4.3.2 as well as the algorithms implemented in HIFOO, LMIRank and PENBMI are local optimization methods, which only report a local minimizer and these solutions may not be the same. Because the LMIRank package can only handle feasibility problems, it cannot directly be used to solve problem (5.3.9). Therefore, we have used a direct search procedure for finding α. The computational time of the overall procedure is much higher than the other methods for the majority of the test problems.

To conclude this section, we show that our methods can also be applied to solve the problem of optimizing the pseudo-spectral abscissa in static feedback controller designs. This problem is described as follows (see [34, 118]):

$$\begin{aligned} &\max_{\beta,\mu,\omega,F,P} \beta \\ &\text{s.t.} \begin{bmatrix} 2\beta P + A_F^T P + P A_F + \mu I - \omega I_z & \varepsilon P \\ \varepsilon P & \omega I \end{bmatrix} \preceq 0, \\ &\quad P \succ 0,\ P = P^T,\ \mu > 0, \end{aligned} \tag{5.3.10}$$

where $A_F = A + BFC$ as before and $\omega \leq 0$.

We only illustrate Scheme **S.5.2.2**, but it can be done similarly for Scheme **S.5.2.1**. By using the same notation $B_F = A + BFC + \beta I$ as in (5.3.9) and applying the statement b) of Lemma 5.2.1, the BMI constraint in this problem can be transformed into a psd-convex-concave one:

$$\begin{bmatrix} B_F^T B_F + P^T P + (\mu - \omega) I & \epsilon P \\ \epsilon P & \omega I \end{bmatrix} - \begin{bmatrix} (B_F - P)^T (B_F - P) & 0 \\ 0 & 0 \end{bmatrix} \preceq 0.$$

If we denote the linearization of $(B_F - P)^T (B_F - P)$ at the iteration k by H_k, i.e. $H_k = (B_F - P)^T (B_F^k - P^k) + (B_F^k - P^k)^T (B_F - P) - (B_F^k - P^k)^T (B_F^k - P^k)$, then the linearized constraint in the subproblem (4.3.4) can be represented as an LMI thanks to Lemma 5.2.1:

$$\begin{bmatrix} H_k + (\omega - \mu) I & B_F^T & P & -\varepsilon P \\ B_F & I & 0 & 0 \\ P & 0 & I & 0 \\ -\varepsilon P & 0 & 0 & -\omega I \end{bmatrix} \succeq 0.$$

Hence, Algorithm 4.3.2 can be applied to solve problem (5.3.10).

Remark 5.3.1. *If we define $\bar{F} := BFC$ then the bilinear matrix mapping $A_F^T P + P A_F$ can be rewritten as:*

$$A_F^T P + P A_F = \frac{1}{2} \left[(P + \bar{F})^T (P + \bar{F}) - (P - \bar{F})^T (P - \bar{F}) \right] - A^T P - P A.$$

By using this decomposition, one can avoid the contribution of matrix A on the bilinear term. Consequently, Algorithm 4.3.2 may work better in some specific problems.

5.4 $\mathcal{H}_2$ control: BMI optimization approach

In this section, we consider an optimization problem arising in $\mathcal{H}_2$ synthesis of the linear system (5.1.1). Let us assume that $D_{12} = 0$ and $D_{21} = 0$, then

this problem is formulated as the following optimization problem with BMI constraints [119].

$$\begin{array}{ll}\min\limits_{F,Q,X} & \mathbf{trace}(X)\\ \text{s.t.} & (A+BFC)Q+Q(A+BFC)^T+B_1B_1^T \preceq 0,\\ & \begin{bmatrix} X & C_1Q\\ QC_1^T & Q\end{bmatrix} \succeq 0, \quad Q \succ 0.\end{array} \tag{5.4.1}$$

Here, we also assume that $B_1B_1^T$ is positive definite. Otherwise, we use $B_1B_1^T + \varepsilon I$ instead of $B_1B_1^T$ with $\varepsilon = 10^{-5}$ in (5.4.1).

In order to apply Algorithms 4.3.1 and 4.3.2 for solving problem (5.4.1), a starting point $\bar{x}^0 \in \mathrm{ri}(\mathcal{D})$ is required. This task can be done by performing some extra steps called *Phase 1*. This phase is described in detail as follows:

Algorithm 5.4.1.(**Phase 1**: *Determine a starting point* $\bar{x}^0 \in ri(\mathcal{D})$).
Step 1. If $\alpha_0(A + A^T) < 0$ then we set $F^0 := 0$. Otherwise, go to Step 3.
Step 2. Solve the following optimization problem with LMI constraints:

$$\begin{array}{ll}\min\limits_{Q,X} & \mathbf{trace}(X)\\ \text{s.t.} & A_{F^0}Q + QA_{F^0}^T + B_1B_1^T \prec 0,\\ & \begin{bmatrix} X & C_1Q\\ QC_1^T & Q\end{bmatrix} \succ 0, \quad Q \succ 0,\end{array} \tag{5.4.2}$$

where $A_{F^0} := A + BF^0C$. If this problem has a solution Q^0 and X^0 then terminate Phase 1 and using F^0 together with Q^0, X^0 as a starting point $\bar{x}^0$ for Phase 2. Otherwise, go to Step 3.
Step 3. Solve the following feasibility problem with LMI constraints:

$$\text{Find } P \succ 0 \text{ and } K \text{ such that:} \quad \begin{bmatrix} PA+A^TP+KC+C^TK^T & PB_1\\ B_1^TP & -\sigma_0^2 I_w\end{bmatrix} \preceq 0, \quad \begin{bmatrix} X & C_1\\ C_1^T & P\end{bmatrix} \succeq 0,$$

to obtain K^* and P^*, where σ_0 is a given regularization factor. Compute $F^* := B^+(P^*)^{-1}K^*$, where B^+ is a pseudo-inverse of B, and resolve problem (5.4.2) with $F^0 := F^*$. If problem (5.4.2) has a solution Q^0 and X^0 then set $\bar{x}^0 := (F^0, Q^0, X^0)$ and terminate Phase 1. Otherwise, perform Step 4.
Step 4. Apply the method in Section 5.3 to solve the following BMI feasibility problem:

$$\text{Find } F \text{ and } Q \succ 0 \text{ such that : } (A+BFC)Q+Q(A+BFC)^T+B_1B_1^T \prec 0.$$

If this problem has a solution F^0 then go back to Step 2. Otherwise, declare that no strictly feasible point is found.
End.

Note that Step 3 of Algorithm 5.4.1 corresponds to determining a full state feedback controller and approximating it subsequently with an output feedback controller. Step 4 of Algorithm 5.4.1 is usually time consuming. Therefore, in our numerical implementation, we terminate Step 4 after finding a point such that $\alpha_0((A+BFC)Q+Q(A+BFC)^T+B_1B_1^T)\leq -0.1$.

Remark 5.4.1. *Algorithm 5.4.1 is finite. It is terminated either at Step 4 if no feasible point is found or at Step 2 if a feasible point is found. Indeed, if a feasible matrix F^0 is found at Step 4 then the first BMI constraint of* (5.4.2) *is feasible with some $Q\succ 0$. Thus we can find an appropriate matrix X such that $X-CQC^T\prec 0$, which implies that the second LMI constraint of* (5.4.2) *is satisfied. Consequently, problem* (5.4.2) *has a solution.*

Algorithm 5.4.1 is slightly heuristic. It can be improved when we apply it to a specific problem. However, as we can see in the numerical results, it performs quite acceptable for the majority of the test problems.

In the following numerical examples, we have implemented **Phase 1** and **Phase 2** of Scheme **S.5.2.2** by using the decomposition:

$$A_FQ+QA_F^T+B_1B_1^T=\frac{1}{2}(A_F+Q)(A_F+Q)^T+B_1B_1^T-\frac{1}{2}(A_F-Q)(A_F-Q)^T$$

for the BMI form at the left-top corner of the first constraint in (5.4.1). We also used the convex overestimate presented in Example 4.3.3 in Algorithm 4.3.1. The regularization parameters and the stopping criterion for Algorithms 4.3.1 and 4.3.2 were chosen as in Section 5.3 and $K_{\max}:=300$.

We tested Algorithms 4.3.1 and 4.3.2 for many problems in `COMPl`$_{\texttt{e}}$`ib` and the computational results are reported in Table 5.2. For the comparison purpose, we also carried out the test with HIFOO [86] and PENBMI [97], and the results are put in the columns marked by HIFOO and PENBMI in Table 5.2, respectively. The initial controller for HIFOO was set to F^0 and the BMI parameters for PENBMI were initialized with $(F,Q,X)=(F^0,Q^0,X^0)$. Here, n_x,n_y,n_z,n_w,n_u are the dimensions of problems, the columns titled HIFOO and PENBMI give the $\mathcal{H}_2$ norm of the closed-loop system for the static output feedback controller computed by HIFOO and PENBMI; `iter` and `time[s]` are the number of iterations and CPU time in seconds of Algorithms 4.3.1 and 4.3.2, respectively, included Phase 1 and Phase 2. Problems marked by "b" mean that Step 4 in Phase 1 is performed. In Table 5.2, we only report the problems that were solved by Algorithms 4.3.1 and 4.3.2. The numerical results allow us to conclude that Algorithm 4.3.1, Algorithm 4.3.2, PENBMI and HIFOO reported similar values for the majority of the test problems in `COMPl`$_{\texttt{e}}$`ib`. Algorithm 4.3.1 failed in solving 4 problems: AC7, AC8, REA1 and DIS2.

Table 5.2: $\mathcal{H}_2$ synthesis benchmarks on COMPl$_e$ib plants

Problem						Other Results, $\mathcal{H}_2$		Algorithm 4.3.1			Algorithm 4.3.2		
Name	n_x	n_y	n_u	n_z	n_w	HIFOO	PENBMI	$\mathcal{H}_2$	iter	time[s]	$\mathcal{H}_2$	iter	time[s]
AC1[b]	5	3	3	2	3	**0.0250**	**0.0061**	**0.0508**	45	23.460	**0.0540**	3	3.380
AC2[b]	5	3	3	5	3	**0.0257**	**0.0075**	**0.0508**	45	24.180	**0.0540**	3	3.390
AC3	5	4	2	5	5	**2.0964**	**2.0823**	**2.1211**	300	147.990	**2.1117**	210	73.380
AC4	4	2	1	2	2	**11.0269**	-	**11.0269**	2	1.620	**11.0269**	2	0.990
AC6	7	4	2	7	7	**2.8648**	**2.8648**	**2.8660**	56	52.070	**2.8664**	153	124.230
AC7	9	2	1	1	4	**0.0172**	**0.0162**	-	-	-	**0.0176**	1	0.670
AC8	9	5	1	2	10	**0.6330**	**0.7403**	-	-	-	**0.6395**	300	282.760
AC12[b]	4	4	3	1	3	**0.0022**	**0.0106**	**0.0988**	78	55.200	**0.0992**	36	28.540
AC15[b]	4	3	2	6	4	**1.5458**	**1.4811**	**1.5211**	300	200.200	**1.5181**	264	85.390
AC16[b]	4	4	2	6	4	**1.4769**	**1.4016**	**1.4509**	300	159.210	**1.4427**	300	99.790
AC17	4	2	1	4	4	**1.5364**	**1.5347**	**1.5516**	300	141.660	**1.5507**	171	49.350
HE2	4	2	2	4	4	**3.4362**	**3.4362**	**4.7447**	51	27.480	**4.7406**	263	97.310
HE3[b]	8	6	4	10	1	**0.0197**	**0.0071**	**0.1800**	300	225.780	**0.1596**	249	217.360
HE4[b]	8	6	4	12	8	**6.6436**	**6.5785**	**7.2453**	300	254.230	**7.1242**	300	412.830
REA1	4	3	2	4	4	**0.9442**	**0.9422**	-	-	-	**1.0622**	249	80.810
REA2[b]	4	2	2	4	4	**1.0339**	**1.0229**	**1.2359**	300	139.700	**1.1989**	300	101.730
DIS1	8	4	4	8	1	**0.6705**	**0.1174**	**0.76125**	300	312.350	**0.7427**	300	255.810
DIS2	3	2	2	3	3	**0.4013**	**0.3700**	-	-	-	**0.3819**	4	1.370
DIS3	6	4	4	6	6	**0.9527**	**0.9434**	**1.0544**	300	161.330	**1.0322**	300	210.470
DIS4	6	6	4	6	6	**1.0117**	**0.9696**	**1.0518**	300	260.520	**1.0276**	300	210.690
WEC1[b]	10	4	3	10	10	**7.3940**	**8.1032**	**14.6015**	300	326.640	**12.9093**	119	190.150
WEC2[b]	10	4	3	10	10	**6.7908**	**7.6502**	**14.6574**	230	238.090	**12.2102**	261	407.470
AGS	12	2	2	12	12	**6.9737**	**6.9737**	**6.9833**	45	124.010	**6.9838**	18	28.830
BDT1	11	3	3	6	1	**0.0024**	-	**0.0019**	63	155.900	**0.0017**	51	64.410
MFP	4	2	3	4	4	**6.9724**	**6.9724**	**6.9755**	2	4.890	**7.0354**	300	114.560
PSM	7	3	2	5	2	**0.0330**	**0.0007**	**0.2287**	300	173.190	**0.1753**	300	217.250
EB2[b]	10	1	1	2	2	**0.0640**	**0.0084**	**0.1587**	248	299.520	**0.1604**	114	131.380
EB3	10	1	1	2	2	**0.0732**	**0.0072**	**0.1794**	300	330.320	**0.0079**	7	6.240
TF1[b]	7	4	2	4	1	**0.0945**	-	**0.1713**	300	231.470	**0.1599**	192	166.810
TF2	7	3	2	4	1	**11.1803**	-	**11.1803**	10	7.170	**11.1803**	3	2.810
TF3[b]	7	3	2	4	1	**0.1943**	**0.1424**	**0.2763**	162	149.840	**0.2565**	138	128.250
NN2	2	1	1	2	2	**1.1892**	**1.1892**	**1.1892**	6	3.070	**1.1892**	4	1.090
NN4	4	3	2	4	4	**1.8341**	**1.8335**	**1.8681**	300	145.300	**1.8590**	222	67.260
NN8	3	2	2	3	3	**1.5152**	**1.5117**	**1.5738**	300	139.680	**1.5725**	183	50.170
NN11	16	5	3	3	3	**0.1178**	**0.0790**	**0.1149**	249	1234.390	**0.1263**	39	91.070
NN13[b]	6	2	2	3	3	**26.1012**	**26.1314**	**62.3888**	300	180.060	**62.3995**	138	112.750
NN14[b]	6	2	2	3	3	**26.1448**	**26.1314**	**62.3888**	300	180.440	**62.3995**	138	110.020
NN15	3	2	2	4	1	**0.0245**	-	**0.0384**	36	146.860	**0.0210**	6	2.060
NN16[b]	8	4	4	4	8	**0.1195**	**0.1195**	**0.1195**	5	21.730	**0.1195**	3	23.030
NN17	3	1	2	2	1	**3.2530**	**3.2404**	**3.3581**	300	138.870	**3.3329**	300	88.770

If $D_{12} \neq 0$ then the second LMI constraint of (5.4.1) becomes a BMI constraint:

$$\begin{bmatrix} X & (C_1 + D_{12}FC)Q \\ Q(C_1 + D_{12}FC)^T & Q \end{bmatrix} \succeq 0, \tag{5.4.3}$$

which is equivalent to $X - C_F Q C_F^T \succeq 0$, where $C_F := C_1 + D_{12}FC$. Since $f(Q) := Q^{-1}$ is convex on $\mathbf{S}^{n_x}_{++}$ (see Lemma 5.2.1 a)), this BMI constraint can

be reformulated as a convex-concave matrix inequality constraint of the form:

$$\begin{bmatrix} X & C_F \\ C_F^T & O \end{bmatrix} + \begin{bmatrix} O & O \\ O & Q^{-1} \end{bmatrix} \succeq 0. \tag{5.4.4}$$

By linearizing the concave term $-f(Q)$ at $Q = Q^k$ as $(Q^k)^{-1} - (Q^k)^{-1}(Q - Q^k)(Q^k)^{-1}$ (see [30]), the resulting constraint can be written as an LMI constraint. Therefore, Algorithm 4.3.2 can be applied to solve problem (5.4.3) in the case $D_{12} \neq 0$.

5.5 $\mathcal{H}_\infty$ control: BMI optimization approach

Alternatively, we can also apply Algorithms 4.3.1 and 4.3.2 to solve the optimization with BMI constraints arising in $\mathcal{H}_\infty$ control of the linear system (5.1.1). Let us assume that $D_{21} = 0$, then this problem is reformulated as the following optimization problem with BMI constraints [119]:

$$\begin{array}{ll} \min\limits_{F,X,\gamma} & \gamma \\ \text{s.t.} & \begin{bmatrix} A_F^T X + X A_F & X B_1 & C_F^T \\ B_1^T X & -\gamma I_w & D_{11}^T \\ C_F & D_{11} & -\gamma I_z \end{bmatrix} \prec 0, \\ & X \succ 0,\ \gamma > 0. \end{array} \tag{5.5.1}$$

Here, as before, $A_F = A + BFC$ and $C_F = C_1 + D_{12}FC$. The bilinear matrix term $A_F^T X + X A_F$ at the top-left corner of the first constraint can be decomposed as (5.3.2) or (5.3.3). Therefore, we can use these decompositions to transform problem (5.5.1) into (4.2.5). After linearization, the resulting subproblem is also rewritten as a standard SDP problem by applying Lemma 5.2.1. The scheme **S.5.2.2** is then applied. We omit this specification here. Similarly, we can use the same trick as in Example 4.3.3 to form the convex subproblems of the form CSDP($\bar{x}^k$) in Algorithm 4.3.1 for this example.

To determine a starting point, we perform **Phase 1** which is similar to the one carried out in the $\mathcal{H}_2$-control subsection.

Algorithm 5.5.1.(**Phase 1:** *Determine a starting point* $\bar{x}^0 \in ri(\mathcal{D})$).
Step 1. If $\alpha_0(A^T + A) < 0$ then set $F^0 = 0$. Otherwise, go to Step 3.
Step 2. Solve the following optimization problem with LMI constraints:

$$\begin{array}{ll} \min\limits_{\gamma,X} & \gamma \\ \text{s.t.} & \begin{bmatrix} A_{F^0}^T X + X A_{F^0} & X B_1 & C_{F^0}^T \\ B_1^T X & -\gamma I_w & D_{11}^T \\ C_{F^0} & D_{11} & -\gamma I_z \end{bmatrix} \prec 0, \\ & X \succ 0,\ \gamma > 0, \end{array} \tag{5.5.2}$$

where $A_{F^0} := A + BF^0C$ and $C_{F^0} := C_1 + D_{12}F^0C$. If this problem has a solution γ^0 and X^0 then terminate Phase 1 and using F^0 together with γ^0, X^0 as a starting point $\bar{x}^0$ for Phase 2. Otherwise, go to Step 3.
Step 3. Solve the following feasibility problem of LMI constraints:

$$\begin{aligned}\text{Find} \quad & P \succ 0,\ \gamma > 0 \text{ and } K \text{ such that:}\\ & \begin{bmatrix} PA^T+AP+K^TB^T+BK & B_1 & PC_1+K^TD_{12}^T \\ B_1^T & -\gamma I_w & D_{11}^T \\ C_1P+D_{12}K & D_{11} & -\gamma I_z \end{bmatrix} \prec 0,\end{aligned}$$

to obtain K^*, γ^* and P^*. Compute $F^* := K^*(P^*)^{-1}C^+$, where C^+ is a pseudo-inverse of C, and resolve problem (5.5.2) with $F^0 := F^*$. If problem (5.5.2) has a solution X^0 and γ^0 then set $\bar{x}^0 := (F^0, X^0, \gamma^0)$ and terminate Phase 1. Otherwise, perform Step 4.
Step 4. Apply the method in Section 5.3 to solve the following BMI feasibility problem:

$$\text{Find } F \text{ and } P \succ 0 \text{ such that : } (A+BFC)^TP + P(A+BFC) \prec 0.$$

If this problem has a solution F^0 then go back to Step 2. Otherwise, declare that no strictly feasible point for (5.5.1) is found.
End.

As in the $\mathcal{H}_2$ control problem, Algorithm 5.5.1 of the $\mathcal{H}_\infty$ control problem is also terminated after finite iterations. In this Section, we also tested Algorithms 4.3.1 and 4.3.2 by using Algorithm 5.5.1 at **Phase 1** for several problems in `COMPl`$_{\texttt{e}}$`ib` and by using the same parameters and the stopping criterion as in the previous sections. The computational results are shown in Table 5.3. The numerical results computed by HIFOO and PENBMI are also included in Table 5.3. Here, the notation is the same as in Table 5.2, except that $\mathcal{H}_\infty$ denotes the $\mathcal{H}_\infty$-norm of the closed-loop system for the static output feedback controller. We can see from Table 5.3 that the optimal values reported by Algorithms 4.3.1 and 4.3.2 and HIFOO are almost similar for many problems whereas in general PENBMI has difficulties in finding a feasible solution.

5.6 Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control: BMI optimization approach

Motivated from the $\mathcal{H}_2$ and $\mathcal{H}_\infty$-control problem, in this section we consider the mixed $\mathcal{H}_2/\mathcal{H}_\infty$ synthesis control problem. Let us assume that $D_{11} = 0$, $D_{21} = 0$ and the performance output z is divided in two components, z_1 and

Table 5.3: $\mathcal{H}_\infty$ synthesis benchmarks on `COMPl_eib` plants

Problem						Other Results, $\mathcal{H}_\infty$		Algorithm 4.3.1			Algorithm 4.3.2		
Name	n_x	n_y	n_u	n_z	n_w	HIFOO	PENBMI	$\mathcal{H}_\infty$	iter	time[s]	$\mathcal{H}_\infty$	iter	time[s]
AC1	5	3	3	2	3	**0.0000**	-	**0.0195**	117	39.620	**0.0177**	93	28.050
AC2	5	3	3	5	3	**0.1115**	-	**0.1174**	120	91.560	**0.1140**	99	32.540
AC3	5	4	2	5	5	**4.7021**	-	**3.5053**	267	193.940	**3.4859**	210	76.170
AC4	4	2	1	2	2	**0.9355**	-	**69.9900**	2	2.690	**69.9900**	2	2.620
AC6	7	4	2	7	7	**4.1140**	-	**4.1954**	167	138.570	**4.1954**	167	138.570
AC7	9	2	1	1	4	**0.0651**	**0.3810**	**0.0339**	300	276.310	**0.0548**	300	278.330
AC8	9	5	1	2	10	**2.0050**	-	**4.5463**	224	230.990	**3.0520**	247	298.070
AC9[b]	10	5	4	2	10	**1.0048**	-	**4.2254**	300	365.910	**0.9237**	300	470.910
AC11[b]	5	4	2	5	5	**3.5603**	-	**3.4924**	300	255.620	**3.0104**	68	60.260
AC12	4	4	3	1	3	**0.3160**	-	**2.5345**	300	155.840	**2.3025**	300	181.870
AC15	4	3	2	6	4	**15.2074**	**427.4106**	**15.2036**	153	130.660	**15.1995**	105	36.700
AC16	4	4	2	6	4	**15.4969**	-	**15.0433**	267	201.360	**14.9881**	186	68.820
AC17	4	2	1	4	4	**6.6124**	-	**6.6571**	192	64.880	**6.6373**	129	42.400
HE1[b]	4	1	2	2	2	**0.1540**	**1.5258**	**0.2188**	300	97.760	**0.1807**	300	143.520
HE2	4	2	2	4	4	**4.4931**	-	**6.8168**	300	114.140	**6.7846**	177	67.470
HE3	8	6	4	10	1	**0.8545**	**1.6843**	**0.8640**	15	16.320	**0.9243**	105	95.000
HE4[b]	8	6	4	12	8	**23.3448**	-	**252.9511**	13	22.240	**22.8713**	252	325.580
HE5[b]	8	2	4	4	3	**8.8952**	-	**36.3330**	154	208.680	**37.3906**	300	430.820
REA1	4	3	2	4	4	**0.8975**	-	**0.8815**	183	67.790	**0.8815**	96	34.430
REA2[b]	4	2	2	4	4	**1.1881**	-	**1.4444**	300	109.430	**1.4188**	300	118.320
REA3	12	3	1	12	12	**74.2513**	**74.4460**	**75.0634**	2	137.120	**74.5478**	2	234.800
DIS1	8	4	4	8	1	**4.1716**	-	**4.2041**	129	110.330	**4.1943**	93	66.130
DIS2	3	2	2	3	3	**1.0548**	**1.7423**	**1.1570**	78	28.330	**1.1546**	54	17.120
DIS3	6	4	4	6	6	**1.0816**	-	**1.1701**	219	160.680	**1.1382**	285	195.960
DIS4	6	6	4	6	6	**0.7465**	-	**0.7532**	171	126.940	**0.7498**	126	93.220
TG1[b]	10	2	2	10	10	**12.8462**	-	**12.9461**	64	264.050	**12.9336**	45	84.380
AGS	12	2	2	12	12	**8.1732**	**188.0315**	**8.1733**	41	160.880	**8.1732**	24	55.290
WEC2	10	4	3	10	10	**4.2726**	**32.9935**	**8.8809**	300	1341.760	**6.6082**	300	476.010
WEC3	10	4	3	10	10	**4.4497**	**200.1467**	**7.8215**	225	875.100	**6.8402**	300	425.330
BDT1	11	3	3	6	1	**0.2664**	-	**0.8544**	3	5.290	**0.8562**	29	40.910
MFP	4	2	3	4	4	**31.5899**	-	**31.6388**	300	100.660	**31.6079**	171	57.430
IH	21	10	11	11	21	**1.9797**	-	**1.1861**	210	2782.880	**1.1858**	114	1206.340
CSE1	20	10	2	12	1	**0.0201**	-	**0.0219**	3	39.330	**0.0220**	3	20.250
PSM	7	3	2	5	2	**0.9202**	-	**0.9266**	153	104.170	**0.9227**	87	67.470
EB1	10	1	1	2	2	**3.1225**	**39.9526**	**2.0532**	300	299.380	**2.0276**	300	295.420
EB2	10	1	1	2	2	**2.0201**	**39.9547**	**0.8150**	120	103.400	**0.8148**	84	73.770
EB3	10	1	1	2	2	**2.0575**	**3995311.0743**	**0.8157**	117	116.390	**0.8153**	84	75.820
NN1	3	2	1	3	3	**13.9782**	**14.6882**	**18.4813**	300	144.940	**79.2681**	300	127.130
NN2	2	1	1	2	2	**2.2216**	-	**2.2216**	15	7.070	**2.2216**	9	4.060
NN4	4	3	2	4	4	**1.3627**	-	**1.3884**	204	70.200	**1.3802**	156	51.480
NN8	3	2	2	3	3	**2.8871**	**78281181.1490**	**2.9522**	240	84.510	**2.9345**	180	51.830
NN9[b]	5	2	3	4	2	**28.9083**	-	**37.7461**	300	272.250	**32.1222**	300	129.920
NN11[b]	16	5	3	3	3	**0.1037**	-	**0.1596**	15	86.770	**0.1566**	9	366.890
NN15	3	2	2	4	1	**0.1039**	-	**0.1201**	6	4.000	**0.1194**	6	3.740
NN16	8	4	4	4	8	**0.9557**	-	**0.9699**	36	32.200	**0.9656**	48	37.950
NN17	3	1	2	2	1	**11.2182**	-	**11.2538**	270	81.480	**11.2381**	117	32.160

z_2. Then the linear system (5.1.1) becomes:

$$\begin{cases} \dot{x} = Ax + B_1 w + Bu, \\ z_1 = C_1^{z_1} x + D_{12}^{z_1} u, \\ z_2 = C_1^{z_2} x + D_{12}^{z_2} u, \\ y = Cx. \end{cases} \tag{5.6.1}$$

The mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control problem is to find a static output feedback gain F such that, for $u = Fy$, the $\mathcal{H}_2$-norm of the closed loop from w to z_2 is minimized, while the $\mathcal{H}_\infty$-norm from w to z_1 is less than some imposed level γ [31, 118, 151].

This problem leads to the following optimization problem with two BMI constraints [151]:

$$\begin{aligned} \min_{F,P_1,P_2,Z} \quad & \mathbf{trace}(Z) \\ \text{s.t.} \quad & \begin{bmatrix} A_F^T P_1 + P_1 A_F + (C_F^{z_1})^T C_F^{z_1} & P_1 B_1 \\ B_1^T P_1 & -\gamma^2 I \end{bmatrix} \prec 0, \\ & \begin{bmatrix} A_F^T P_2 + P_2 A_F & P_2 B_1 \\ B_1^T P_2 & -I \end{bmatrix} \prec 0, \\ & \begin{bmatrix} P_2 & (C_F^{z_2})^T \\ C_F^{z_2} & Z \end{bmatrix} \succ 0, \ P_1 \succ 0, \ P_2 \succ 0, \end{aligned} \tag{5.6.2}$$

where $A_F := A + BFC$, $C_F^{z_1} := C_1^{z_1} + D_{12}^{z_1} FC$ and $C_F^{z_2} := C_1^{z_2} + D_{12}^{z_2} FC$. Note that if $C = I_{n_x}$, the identity matrix, then this problem becomes an optimization problem of mixed $\mathcal{H}_2/\mathcal{H}_\infty$-control for static state feedback design considered in [151]. In this section, we tested Algorithm 4.3.2 for the static state feedback and output feedback cases. We only tested Algorithm 4.3.1 for the second case.

Case 1.(*The static state feedback case* ($C = I_{n_x}$)). First, we applied the method in [151] to find an initial point via solving two optimization problems with LMI constraints. Then, we used the same approach as in the previous sections to transform problem (5.6.2) into an optimization problem with psd-convex-concave matrix inequality constraints. Finally, Algorithm 4.3.2 was implemented to solve the resulting problem. For convenience of implementation, we introduced a slack variable η and then replaced the objective function in (5.5.1) by $f(x) = \eta^2$ with an additional constraint $\mathbf{trace}(Z) \leq \eta^2$.

In the first case, we tested Algorithm 4.3.2 with three problems. The first problem was also considered in [92] with:

$$A = \begin{bmatrix} -1.40 & -0.49 & -1.93 \\ -1.73 & -1.69 & -1.25 \\ 0.99 & 2.08 & -2.49 \end{bmatrix}, B_1 = \begin{bmatrix} -0.16 & -1.29 \\ 0.81 & 0.96 \\ 0.41 & 0.65 \end{bmatrix}, B = \begin{bmatrix} 0.25 \\ 0.41 \\ 0.65 \end{bmatrix},$$

$$C_1^{z_1} = \begin{bmatrix} -0.41 & 0.44 & 0.68 \end{bmatrix}, \ C_1^{z_2} = \begin{bmatrix} -1.77 & 0.50 & -0.40 \end{bmatrix},$$

$$D_{12}^{z_1} = D_{12}^{z_2} = 1, \text{ and } \gamma = 2.$$

If the tolerance $\varepsilon = 10^{-3}$ was chosen then Algorithm 4.3.2 converged after 17 iterations and reported the value $\eta = 0.7489$ with:

$$F = \begin{bmatrix} 1.9485 & 0.3990 & -0.2119 \end{bmatrix}.$$

This result is similar to the one shown in [151]. If we regularized the subproblem (4.3.4) with $\rho = 0.5 \times 10^{-3}$ and $Q = I_{PF}$ then the number of iterations reduced to 10 iterations.

The second problem is `DIS4` in `COMPl`$_{\texttt{e}}$`ib` [119]. In this problem, we set $C_1^{z_1} = C_1^{z_2}$ and $D_{12}^{z_1} = D_{12}^{z_2}$ as in [151]. Algorithm 4.3.2 converged after 24 iterations with the same tolerance $\varepsilon = 10^{-3}$. It reported $\eta = 1.6925$ and $\gamma = 1.1996$ with:

$$F = \begin{bmatrix} -0.8663 & -0.6504 & -1.1115 & -0.1951 & -0.6099 & 0.2065 \\ 0.1591 & -0.4941 & -0.6322 & -0.5409 & -1.2895 & 0.2774 \\ -0.7017 & -0.0785 & 0.6121 & -0.8919 & 0.2518 & -0.2354 \\ -0.0522 & -0.5556 & -0.5838 & 0.4497 & -1.4279 & -0.6677 \end{bmatrix}.$$

If we regularized the subproblem (4.3.4) with $\rho = 0.5 \times 10^{-3}$ and $Q = I_{PF}$ then the number of iterations was 18.

The third problem is `AC16` in `COMPl`$_{\texttt{e}}$`ib` [119]. In this example we also chose $C_1^{z_1} = C_1^{z_2}$ and $D_{12}^{z_1} = D_{12}^{z_2}$ as in the previous problem. As mentioned in [151], if we chose a starting value $\gamma_0 = 100$, then the LMI problem could not be solved by the SDP solvers (e.g., Sedumi, SDPT3) due to numerical problems. Thus, we rescaled the LMI constraints by using the same trick as in [151]. After doing this, Algorithm 4.3.2 converged after 298 iterations with the same tolerance $\varepsilon = 10^{-3}$. The value of η reported in this case was $\eta = 12.3131$ and $\gamma = 20.1433$ with:

$$F = \begin{bmatrix} -1.8533 & 0.1737 & 0.6980 & 6.4208 \\ 4.2672 & -0.9668 & -1.5952 & -2.9240 \end{bmatrix}.$$

The results obtained by Algorithm 4.3.2 for solving problems `DIS4` and `AC16` in this paper confirm the results reported in [151].

Case 2.(*The static output feedback case*). As before, we first propose a procedure called Phase 1 to determine a starting point for Algorithms 4.3.1 and 4.3.2. We described this phase algorithmically as follows.

Algorithm 5.6.1.(**Phase 1:** *Determine a starting point* $\bar{x}^0$).
Step 1. If $\alpha_0(A^T + A) < 0$ then set $F^0 = 0$. Otherwise, go to Step 3.
Step 2. Solve the following linear SDP problem:

$$\begin{array}{ll} \min\limits_{P_1,P_2,Z} & \mathbf{trace}(Z) \\ \text{s.t.} & \begin{bmatrix} A_{F^0}^T P_1 + P_1 A_{F^0} + (C_{F^0}^{z_1})^T C_{F^0}^{z_1} & P_1 B_1 \\ B_1^T P_1 & -\gamma^2 I \end{bmatrix} \prec 0, \\ & \begin{bmatrix} A_{F^0}^T P_2 + P_2 A_{F^0} & P_2 B_1 \\ B_1^T P_2 & -I \end{bmatrix} \prec 0, \\ & \begin{bmatrix} P_2 & (C_{F^0}^{z_2})^T \\ C_{F^0}^{z_2} & Z \end{bmatrix} \succ 0, \quad P_1 \succ 0, \ P_2 \succ 0, \end{array} \tag{5.6.3}$$

where $A_{F^0} = A + BF^0C$, $C_{F^0}^{z_1} = C_1^{z_1} + D_{12}^{z_1}F^0C$ and $C_{F^0}^{z_2} = C_1^{z_2} + D_{12}^{z_2}F^0C$. If this problem has an optimal solution P_1^0, P_2^0 and Z^0 then terminate Phase 1. Set $\bar{x}^0 := (F^0, P_1^0, P_2^0, Z^0)$ for a starting point of Algorithm 4.3.1 or Algorithm 4.3.2 in Phase 2. Otherwise go to Step 3.

Step 3. Solve the following LMI feasibility problem:

Find $Q \succ 0$, W and Z such that:

$$\begin{bmatrix} AQ+QA^T+BW+W^TB^T & B_1 & (C_1+D_{12}W)^T \\ B_1^T & -I_w & O \\ C_1+D_{12}W & O & -\gamma^2 I_z \end{bmatrix} \prec 0, \tag{5.6.4}$$

$$\begin{bmatrix} AQ+QA^T+BW+W^TB^T & B_1 \\ B_1^T & -I_w \end{bmatrix} \prec 0, \quad \begin{bmatrix} Q & (C_1Q + D_{12}W)^T \\ C_1Q + D_{12}W & Z \end{bmatrix} \succ 0,$$

to obtain a solution Q^*, W^* and Z^*. Set $F^* := W^*(Q^*)^{-1}C^+$, where C^+ is the pseudo-inverse of C. Solve again problem (5.6.3) with $F^0 := F^*$. If problem (5.6.3) has a solution then terminate Phase 1. Otherwise, perform Step 4.

Step 4. Solve the following optimization with BMI constraints:

$$\max_{\beta, F, P_1 \succ 0, P_2 \succ 0} \beta$$

$$\text{s.t. } A_F^TP_1 + P_1A_F^T + (C_F^{z_1})^TC_F^{z_1} + \gamma^{-2}P_1B_1B_1^TP_1 \preceq -2\beta P_1,$$

$$A_F^TP_2 + P_2A_F + P_2B_1B_1^TP_2 \preceq -2\beta P_2$$

to obtain an optimal solution F^* corresponding to the optimal value β^*. If $\beta^* > 0$ then set $F^0 := F^*$ and go back to Step 2 to determine P_1^0, P_2^0 and Z^0. Otherwise, declare that no strictly feasible point of problem (5.6.2) is found.

End.

Since at Step 4 of Algorithm 5.6.1 requires one to solve an optimization problem with two BMI constrains, this task is usually expensive. In our implementation, we only terminate this step after finding a strictly feasible point with a feasible gap 0.1 as before. If matrix C is invertible then matrix F^* at Step 3 is $F^* = W^*(Q^*)^{-1}C^{-1}$. Hence, we can ignore Step 4 of Phase 1.

To avoid a numerical problem in Step 3, we can reformulate problem (5.6.4) equivalently to the following one:

Find $Q \succ 0$, W and Z such that:

$$\begin{bmatrix} AQ+QA^T+BW+W^TB^T & B_1 & (C_1+D_{12}W)^T \\ B_1^T & -\gamma I_w & O \\ C_1+D_{12}W & O & -\gamma I_z \end{bmatrix} \prec 0,$$

$$\begin{bmatrix} Q & (C_1Q + D_{12}W)^T \\ C_1Q + D_{12}W & Z \end{bmatrix} \succ 0.$$

Table 5.4: Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ synthesis benchmarks on COMPl$_e$ib plants reported by Algorithm 4.3.1.

Problem	Results and Performances ($\gamma = 4$)			Results and Performances ($\gamma = 10$)		
Name	$\mathcal{H}_2/\mathcal{H}_\infty$	iter	time[s]	$\mathcal{H}_2/\mathcal{H}_\infty$	iter	time[s]
AC1	**0.0587 / 0.0993**	2	2.410	**0.0587 / 0.0994**	1	2.000
AC2	**0.1071 / 0.1730**	1	2.920	**0.1071 / 0.1730**	1	2.720
AC3	**- / -**	-	-	**4.5720 / 5.1337**	57	94.620
AC6	**- / -**	-	-	**3.9951 / 5.3789**	28	61.460
AC7	**0.0438 / 0.0610**	34	50.080	**0.0441 / 0.0611**	3	6.110
AC11	**4.0914 / 3.9983**	110	150.340	**- / -**	-	-
AC12	**0.0924 / 0.3486**	-	73.46	**- / -**	-	-
AC17	**- / -**	-	-	**4.2061 / 6.6126**	165	100.130
HE1	**0.0973 / 0.2046**	1	34.860	**0.0973 / 0.2075**	1	35.260
HE2	**- / -**	-	-	**4.7326 / 9.8059**	135	97.560
REA1	**1.8217 / 1.4795**	51	23.140	**1.8296 / 1.4495**	300	172.700
REA2	**3.5021 / 3.5122**	72	36.630	**3.5024 / 3.4913**	141	107.180
DIS1	**- / -**	-	-	**4.2341 / 4.6736**	44	275.280
DIS2	**1.5080 / 1.8410**	45	17.960	**1.5080 / 1.8400**	45	20.280
DIS3	**2.0580 / 1.7969**	60	68.530	**2.0579 / 1.7727**	66	136.280
DIS4	**1.6932 / 1.1899**	72	69.000	**1.6932 / 1.1899**	72	68.120
AGS	**- / -**	-	-	**7.0356 / 8.2053**	9	82.160
PSM	**1.5157 / 0.9268**	237	241.210	**1.5158 / 0.9269**	264	281.580
EB2	**0.9023 / 0.8142**	1	124.200	**0.9012 / 0.8142**	1	122.170
EB3	**0.9144 / 0.8143**	1	123.470	**0.9137 / 0.8143**	1	126.810
NN2	**1.5652 / 2.4771**	18	20.540	**1.5651 / 2.4811**	24	37.010
NN4	**1.8778 / 2.0501**	202	154.49	**1.8928 / 2.2496**	257	139.900
NN8	**2.3609 / 3.9999**	21	15.71	**2.3383 / 4.5520**	99	68.700
NN15	**0.0490 / 0.1366**	24	52.410	**0.0488 / 0.1392**	27	49.940
NN16	**0.3544 / 0.9569**	108	126.160	**0.3910 / 0.9573**	300	405.340

We tested Algorithm 4.3.1 described above for several problems in COMPl$_e$ib with the level values $\gamma = 4$ and $\gamma = 10$. In these examples, we assume that the output signals $z_1 \equiv z_2$. Thus we have $C_1^{z_1} = C_1^{z_2} = C_1$ and $D_{12}^{z_1} = D_{12}^{z_2} = D_{12}$. The parameters and the stopping criterion of the algorithm were chosen as in Section 5.3. The computational results are reported in Table 5.4 with $\gamma = 4$ and $\gamma = 10$. Here, $\mathcal{H}_2/\mathcal{H}_\infty$ are the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norms of the closed-loop systems for the static output feedback controller, respectively. With $\gamma = 10$, the computational results show that Algorithm 4.3.1 satisfies the condition $\|P_\infty(s)\|_\infty \leq \gamma = 10$ for the test problems except problems AC11 and AC12. While, with $\gamma = 4$, there are 6 problems reported infeasible, which are denoted by "-". The $\mathcal{H}_\infty$-constraint of two problems AC11 and NN8 is active with respect to $\gamma = 4$.

We also tested Algorithm 4.3.2 by using the same parameters as in Algorithm 4.3.1 above. The results are reported in Table 5.5. With $\gamma = 10$ Algorithm 4.3.2 solved all the problems, while, with $\gamma = 4$, there are 5 problems reported

Table 5.5: Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ synthesis benchmarks on COMPl$_e$ib plants reported by Algorithm 4.3.2.

Problem	Results and Performances ($\gamma = 4$)			Results and Performances ($\gamma = 10$)		
Name	$\mathcal{H}_2/\mathcal{H}_\infty$	iter	time[s]	$\mathcal{H}_2/\mathcal{H}_\infty$	iter	time[s]
AC1	**0.0585 / 0.0990**	3	4.22	**0.0585 / 0.0990**	3	4.27
AC2	**0.1067 / 0.1723**	6	7.31	**0.1070 / 0.1727**	3	7.15
AC3	**5.2770 / 3.9999**	51	281.53	**4.5713 / 5.1298**	18	19.18
AC6	**- / -**	-	-	**4.0297 / 4.8753**	283	330.64
AC7	**0.0415 / 0.0961**	1	3.39	**0.0420 / 0.1286**	2	3.91
AC8	**1.2784 / 2.2288**	43	60.78	**1.3020 / 2.5719**	23	31.59
AC11	**4.0704 / 4.0000**	76	175.75	**4.0021 / 5.1949**	117	122.86
AC12	**0.0924 / 0.3486**	18	73.46	**1.4454 / 1.6444**	300	234.13
AC17	**- / -**	-	-	**4.1228 / 6.6472**	2	11.620
HE1	**0.1123 / 0.2257**	2	131.18	**0.0973 / 0.2080**	1	30.97
HE2	**- / -**	-	-	**4.7302 / 9.8931**	75	55.48
REA1	**1.8214 / 1.4740**	30	25.64	**1.8213 / 1.4730**	30	26.65
REA2	**3.5014 / 3.5180**	42	22.09	**3.5015 / 3.5209**	45	23.26
DIS1	**- / -**	-	-	**2.8505 / 4.7904**	15	30.51
DIS2	**1.5079 / 1.8500**	18	7.92	**1.5079 / 1.8520**	21	7.92
DIS3	**2.0577 / 1.7934**	27	25.03	**2.0577 / 1.7766**	30	24.54
DIS4	**1.6926 / 1.1952**	21	18.62	**1.6926 / 1.2009**	24	21.55
AGS	**- / -**	-	-	**7.0332 / 8.2035**	8	196.73
PSM	**1.5115 / 0.9248**	177	160.41	**1.5115 / 0.9248**	180	167.31
EB2	**0.7765 / 1.0828**	7	9.70	**0.7768 / 1.0867**	10	13.16
EB3	**0.8406 / 0.9249**	1	3.21	**0.8383 / 0.9418**	1	2.93
EB4	**1.0147 / 1.0707**	6	59.55	**0.9981 / 1.2146**	12	111.26
NN2	**1.5651 / 2.4834**	12	5.37	**1.5651 / 2.4876**	12	5.49
NN4	**1.8778 / 2.0501**	202	154.49	**1.8779 / 2.0519**	213	161.00
NN8	**2.3609 / 3.9999**	21	15.71	**2.3376 / 4.6514**	15	6.57
NN15	**0.0820 / 0.1010**	42	18.75	**0.0771/0.1012**	24	10.47
NN16	**0.3187 / 0.9574**	90	96.44	**0.3319 / 0.9572**	258	303.87

infeasible. The $\mathcal{H}_\infty$-constraint of three problems: AC8, AC11 and NN8 is active in this algorithm.

As we can see from Tables 5.4 and 5.5 that the results given by Algorithm 4.3.1 are similar to Algorithm 4.3.2 in the majority of the tested problems. However, Algorithm 4.3.1 encounters a difficulty for solving this example compared to Algorithm 4.3.2.

5.7 Conclusion

In this chapter we have shown the applications of two algorithms developed in Chapter 4 to solve optimization problems with BMI constraints arising

from static state/output feedback controller design. One main task of these algorithms is to find a starting point in the interior of the feasible set of the given problem. We have proposed several new procedures to find such a point which can be carried out in finite steps. Both algorithms have been tested via several numerical examples in static feedback controller design using the data from the COMPL$_{\text{ei}}$b library. We have also compared our codes with other software tools such as PENBMI, HIFOO and LMIRank. The numerical tests have shown that our codes provided competitive results to those solvers for the majority of the tested problems.

Part II

Decomposition in Separable Optimization

Chapter 6

Existing approaches in separable optimization

Many optimization problems fall into the class of large-scale and separable optimization and need to be solved in a parallel and distributed manner. Such problems appear in many fields of science and engineering such as graph theory, networks, transportation, distributed model predictive control, distributed estimation, multistage stochastic optimization, compressive sensing and machine learning, see e.g. [14, 42, 46, 77, 100, 113, 114, 157, 165, 174, 203, 206, 207, 212, 218] and the references quoted therein. Solving large-scale optimization problems is still a challenge in many applications [43] due to the limitations of computational devices and computer systems. Recently, thanks to the development of parallel and distributed computing systems, many large-scale problems have been solved by using the framework of decomposition. However, methods and algorithms for solving this type of problems, which can be run in a parallel or distributed manner, are still limited [17, 43]. Part II of the thesis focuses on developing numerical solution methods for solving separable optimization problems based on decomposition approaches. This part will be divided into five chapters. In Chapter 6, we mainly review some related existing methods for solving separable optimization problems, describe the Lagrangian dual decomposition framework and recall some concepts related to parallel and distributed algorithms and performance profiles. Chapters 7, 8 and 9 present alternatively two smoothing techniques in the dual decomposition framework and propose different decomposition methods for solving separable convex optimization problems. An extension to the nonconvex case is considered in Chapter 10.

Outline of Chapter 6. This chapter is organized as follows. First, we state our problem formulations both in the convex and nonconvex case and the optimality condition of these problems in Section 6.1. Next, we briefly review some existing but related methods for solving such problems in Section 6.2. Then, we recall the Lagrangian dual decomposition approach for separable convex optimization in Section 6.3. Section 6.4 recalls some concepts related to parallel and distributed algorithms. The last section briefly describes performance profile concepts.

6.1 Problem statements

From now on, we focus on separable optimization problems both in the convex and nonconvex case. For later references, we state these problems separately in the following two subsections.

Separable convex optimization

We are interested in the following *separable convex optimization problem*:

$$\phi^* := \begin{cases} \min\limits_{x\in\mathbf{R}^n} & \phi(x) := \sum\limits_{i=1}^{M} \phi_i(x_i) \\ \text{s.t.} & \sum\limits_{i=1}^{M} (A_i x_i - b_i) = 0, \\ & x_i \in X_i,\ i = 1, \cdots, M, \end{cases} \tag{SepCOP}$$

where $x := (x_1, x_2, \ldots, x_M) \in \mathbf{R}^n$ is a vector of decision variables, $\phi_i : \mathbf{R}^{n_i} \to \mathbf{R}$ is convex, $X_i \in \mathbf{R}^{n_i}$ is a nonempty, closed convex set, $A_i \in \mathbf{R}^{m\times n_i}$, $b_i \in \mathbf{R}^m$ for all $i = 1, \ldots, M$, and $n_1 + n_2 + \cdots + n_M = n$. As usual, we refer to the first constraint as a *coupling linear constraint* and the last constraints as local convex constraints.

In principle, all convex programming problems can be brought into this separable form by doubling the variables, i.e. by introducing new variables x_i and then imposing the constraint $x_i = x$. Despite the increasing number of variables, treating convex programming problems in such a way may be useful in some situations, see, e.g. [73, 77]. Let $X := X_1 \times X_2 \times \cdots \times X_M$ be the Cartesian product of X_i, $A := [A_1, A_2, \cdots, A_M]$ be a matrix formed from M blocks A_i, $i = 1, \ldots, M$ and $b := \sum_{i=1}^{M} b_i$. Then the first constraint of (SepCOP) can shortly be written as $Ax - b = 0$.

Alternatively to (SepCOP), we can also consider the maximization formulation:

$$\phi^* := \begin{cases} \max\limits_{x\in\mathbf{R}^n} & \phi(x) := \sum\limits_{i=1}^{M} \phi_i(x_i) \\ \text{s.t.} & \sum\limits_{i=1}^{M}(A_i x_i - b_i) = 0, \\ & x_i \in X_i,\ i = 1, \cdots, M. \end{cases} \tag{SepCOP$_{\max}$}$$

Here, the objective functions ϕ_i is assumed to be concave for $i = 1, \ldots, M$. Since $\max \phi(x) = -\min\{-\phi(x)\}$, both formulations (SepCOP) and (SepCOP$_{\max}$) are equivalent.

Fundamental assumption and optimality condition. Problem (SepCOP) is said to satisfy the *Slater constraint qualification condition* if:

$$\mathrm{ri}(X) \cap \{x \in \mathbf{R}^n \mid Ax = b\} \neq \emptyset, \tag{6.1.1}$$

where $\mathrm{ri}(X)$ is the relative interior of the convex set X. Let us denote by X^* the solution set of (SepCOP). Throughout Part II, we assume that the following assumption is satisfied [164].

Asumption A.6.1.7. *The solution set X^* is nonempty and either the Slater qualification condition for problem* (SepCOP) *holds or X_i is polyhedral. The function ϕ_i is proper, lower semicontinuous and convex in $\mathbf{R}^{n_i}$ for $i = 1, \ldots, M$.*

Note that the objective function ϕ is not necessarily smooth. For example, $\phi(x) = \|x\|_1 = \sum_{i=1}^{n} |x_{(i)}|$, which is nonsmooth and separable, see Example 1.1.4. In the maximization case (SepCOP$_{\max}$), the objective function ϕ is assumed to be proper, upper semicontinuous and concave in $\mathbf{R}^n$, $i = 1, \ldots, M$ in Assumption **A.6.1.7**. We use the same Assumption **A.6.1.7** for this case without repeating in the next chapters.

The optimality condition for (SepCOP) is expressed as:

$$\begin{cases} 0 & \in \partial\phi(x) + A^T y + \mathcal{N}_X(x), \\ 0 & = Ax - b. \end{cases} \tag{6.1.2}$$

Here, $\partial\phi(x)$ is the subdifferential of the convex function ϕ at x, $\mathcal{N}_X(x)$ is the normal cone of the convex set X at x and y is the Lagrange multiplier associated with the coupling constraint $Ax - b = 0$. Note that the first line of (6.1.2) explicitly ensures the condition $x \in X$. A point $(x, y) \in \mathbf{R}^n \times \mathbf{R}^m$ that satisfies (6.1.2) is called a Karush-Kuhn-Tucker (KKT) point of (SepCOP).

Alternatively, the optimality for the maximization problem ($\text{SepCOP}_{\max}$) is expressed as:

$$\begin{cases} 0 & \in \partial\phi(x) + A^T y - \mathcal{N}_X(x), \\ 0 & = Ax - b. \end{cases} \tag{6.1.3}$$

Here, $\partial\phi(x)$ denotes the superdifferential of the concave function ϕ at x. Under Assumption **A.6.1.7**, the optimality condition (6.1.2) (resp. (6.1.3)) is necessary and sufficient for the solution of (SepCOP) (resp. ($\text{SepCOP}_{\max}$)).

Separable nonconvex optimization

In the nonconvex case, we are interested in the following formulation:

$$\phi^* := \begin{cases} \min\limits_{x \in \mathbf{R}^n} & \phi(x) := \sum\limits_{i=1}^{M} [g_i(x_i) + h_i(F_i(x_i))] \\ \text{s.t.} & \sum\limits_{i=1}^{M} (A_i x_i - b_i) = 0, \\ & x_i \in X_i, \ i = 1, \cdots, M, \end{cases} \tag{SepNCOP}$$

where x_i, A_i and b_i are defined as in (SepCOP) for $i = 1, \ldots, M$. The function $g_i : \mathbf{R}^{n_i} \to \mathbf{R}$ is assumed to be proper, lower semicontinuous and convex (and possibly smooth), while $h_i : \mathbf{R}^{m_i} \to \mathbf{R}$ is proper, lower semicontinuous and convex but not necessarily smooth. The inner function $F_i : \mathbf{R}^{n_i} \to \mathbf{R}^{m_i}$ is continuously differentiable on its domain for $i = 1, \ldots, M$. If F_i is affine or h_i vanishes for $i = 1, \ldots, M$ then problem (SepNCOP) coincides with the separable convex programming problem (SepCOP).

Note that if either nonconvex coupling constraints or local nonconvex constraints are present then one can use slack variables and penalty functions to transform the given problem into (SepNCOP). We do not discuss these transformations in detail in this section.

6.2 Related existing approaches

This section briefly reviews some related existing approaches for solving separable optimization problems both in the convex and nonconvex case.

Methods for separable convex optimization

Sparse large-scale convex optimization problems can be solved efficiently by centralized optimization methods such as interior point, SQP and gradient-type methods thanks to the development of the underlying computational sparse linear algebra routines. In this thesis, we are interested in convex programming problems which possess the separability and dynamical structure (in the sense of dynamic topology and distributed location of problem data and devices). The first property leads to the decomposability of the problem and the second one may cause some difficulty for centralized optimization solvers.

In the literature, several approaches have been proposed for solving problem (SepCOP). For example, (augmented) Lagrangian relaxation and subgradient-type methods of multipliers [17, 61, 87, 136, 164, 206], Fenchel's dual decomposition [89], alternating direction methods (ADM) [32, 62, 77, 93, 94, 115], proximal point-type methods [16, 39, 201], splitting methods [63, 64], interior point methods [113, 131, 133, 171, 218], mean value cross decomposition [101], partial inverse method [176] and bundle methods [175] have been proposed among many others.

From the application side, problems of the form (SepCOP) cover very well resource allocation and network utility maximization problems [211], wireless and DSL spectrum management problems [107, 125, 126, 202, 203, 214], distributed model predictive control [37, 208], multistage stochastic optimization problems [23, 131, 218, 219] and machine learning [32, 174, 175] among many others. A good tutorial on decomposition methods for network utility maximization problems can be found in [154]. Particular approaches have also been proposed to those applications which aim at exploiting specific structure of the problems.

One of the classical approaches for solving (SepCOP) is Lagrangian dual decomposition [14, 17]. The main idea of this approach is to solve the dual problem by means of subgradient-type methods. It has been recognized in practice that these methods are usually slow and numerically sensitive to the choice of step sizes, see e.g. [142, 144]. In the special case of strongly convex primal problem, the dual function is differentiable. Consequently, classical gradient methods can be applied to solve the dual problem.

Recently, Nesterov [142] developed the smoothing techniques for solving nonsmooth convex optimization problems based on a fast gradient scheme which was introduced in his pioneering work [137]. The fast gradient schemes have been applied in numerous applications including image processing, compressed sensing, machine learning, networks and system identification, see, e.g. [3, 22, 77, 90, 91, 147, 212].

Exploiting Nesterov's smoothing idea in [140], Necoara and Suykens [134] applied those smoothing techniques to the dual problem in the framework of Lagrangian dual decomposition and then used the fast gradient scheme to maximize the smoothed dual function. This resulted in a new variant of dual decomposition algorithms for solving separable convex optimization. The authors proved that the rate of convergence of their algorithm is $O\left(\frac{1}{k}\right)$ which is much better than $O\left(\frac{1}{\sqrt{k}}\right)$ in the subgradient-type methods of multipliers considered recently in [61, 109, 136], where k is the iteration counter. A main disadvantage of this scheme is that the smoothness parameter needs to be given *a priori*. Moreover, this parameter crucially depends on a given desired accuracy. Since the Lipschitz constant of the gradient of the objective function in the smoothed dual problem is inversely proportional to the smoothness parameter, the algorithm usually generates short steps towards a solution of the dual problem although the rate of convergence is $O\left(\frac{1}{k}\right)$. In [105] the authors studied a distributed method for unconstrained optimizations where they proved that the rate of convergence of the method is $O\left(\frac{\log k}{k}\right)$.

Methods for separable nonconvex optimization

Unlike in convex optimization, strong duality is no longer preserved in the nonconvex case in general. Consequently, Lagrangian dual decomposition techniques are not directly applicable to nonconvex problems. Moreover, standard optimization techniques such as sequential quadratic programming require a globalization strategy such as line-search or trust-region procedures. These procedures are hard to implement in a parallel or distributed manner due to their global information requirement.

In order to avoid global information, penalty function or augmented Lagrangian methods can be applied. However, using penalty functions usually leads to nonsmoothness of the primal subproblems while the augmented Lagrangian functions encounter a crossproduct term which is not separable. Many attempts have been proposed to overcome the second difficulty. For instance, Bertsekas [15] proposed a convexification procedure by means of proximal point methods instead of augmented Lagrangian functions. Stephanopoulos and Westerberg in [178] approximated the crossproduct term by linear functions which led to a heuristic, complex and poorly performing method. Tanikawa and Mukai in [183] proposed a new approach based on the well-known Fletcher smooth augmented Lagrangian function. The authors used the trick that by introducing additional variables, the augmented Lagrangian function was decomposable. This method was further extended to the inequality constraint case by Tatjewski in [185]. However, only local convergence of the methods was considered in both

papers. Tatjewski in [184] proposed a three-level optimization method to solve (SepNCOP) by separating the quadratic term of the augmented Lagrangian function into M components via additional slack variables. Recently, Hamdi [88] combined the augmented Lagrangian function, proximal point method and alternating direction method of multipliers to obtain a new method for solving (SepNCOP). Another approach which is based on *sequential convex programming* was proposed in [133] for solving distributed nonconvex optimal control problems. From practical side, a good review of alternating direction methods of multipliers for distributed convex and nonconvex optimization can be found in [32].

6.3 Lagrangian decomposition in separable convex programming

In this section, we briefly recall the Lagrangian dual decomposition in separable convex optimization. For more details, we refer the reader to [14, 17].

Let us define the Lagrange function of the problem (SepCOP) with respect to the coupling constraint $Ax - b = 0$ as:

$$\mathcal{L}(x, y) := \phi(x) + y^T(Ax - b) = \sum_{i=1}^{M} \left[\phi_i(x_i) + y^T(A_i x_i - b_i)\right],$$

where $y \in \mathbf{R}^m$ is the multiplier associated with the coupling constraint $Ax - b = 0$. A point $(x^*, y^*) \in X \times \mathbf{R}^m$ is called a *saddle point* of $\mathcal{L}$ if:

$$\mathcal{L}(x^*, y) \leq \mathcal{L}(x^*, y^*) \leq \mathcal{L}(x, y^*), \ \forall x \in X, \ \forall y \in \mathbf{R}^m.$$

Next, we define the Lagrange dual function g of the problem (SepCOP) as:

$$g(y) := \min_{x \in X} \left\{\mathcal{L}(x, y) := \phi(x) + y^T(Ax - b)\right\}. \tag{6.3.1}$$

and then write down the dual problem of (SepCOP) as:

$$g^* := \max_{y \in \mathbf{R}^m} g(y). \tag{6.3.2}$$

By Assumption **A.6.1.7** *strong duality* holds and we have:

$$g^* = \max_{y \in \mathbf{R}^m} g(y) \overset{\text{strong duality}}{=} \min_{x \in X} \{\phi(x) \mid Ax = b\} = \phi^*.$$

Let us denote by Y^* the solution set of the dual problem (6.3.2). It is well-known that Y^* is bounded due to Assumption **A.6.1.7**.

It is important to note that the dual function $g(y)$ defined by (6.3.1) can be computed separately as:

$$g(y) = \sum_{i=1}^{M} g_i(y), \tag{6.3.3}$$

where

$$g_i(y) := \min_{x_i \in X_i} \left\{ \phi_i(x_i) + y^T (A_i x_i - b_i) \right\}, \ i = 1, \ldots, M. \tag{6.3.4}$$

We call the minimization problems in (6.3.4) the *primal subproblems*. We denote by $x_i^*(y)$ a solution of the primal subproblem i for $i = 1, \ldots, M$ and $x^*(y) := (x_1^*(y), \ldots, x_M^*(y))$. Note that if $x_i^*(y)$ is not unique for a given y then g_i is not differentiable at y $(i = 1, \cdots, M)$. Consequently, g is not differentiable at y. Numerical solution methods for maximizing the function g are a challenge. The representation (6.3.3)-(6.3.4) is usually called a *dual decomposition* of the dual function g.

6.4 Parallel algorithms vs distributed algorithms

In this section, we first briefly review some concepts in parallel and distributed mechanism including parallel and distributed computing systems and parallel and distributed optimization algorithms. Then we discuss some implementation issues of distributed optimization algorithms. For more details of these fields, we refer the reader to [6, 17, 18, 80, 168].

Parallel and distributed computing systems. Work on parallel and distributed computation spans several broad areas, such as the design of parallel machines, parallel programming languages, parallel algorithm development and analysis, and applications related issues [6, 17, 80, 168]. Roughly speaking, parallel computing systems consist of several processors or computing units that are located within a small distance of each other. Their main purpose is to divide a given computational task into smaller ones that can be carried out simultaneously in order to achieve a common goal. The communication between processors or computing units is reliable and predictable. Distributed computing systems are different in a number of ways. Processors or computing units may be far apart, interprocessor communication is more problematic, communication delays may be unpredictable and communication links themselves may be unreliable [18]. In one view, a distributed computing system consists of multiple autonomous computers or computing units that communicate through a network. These computers or computing units interact with each other to

achieve a common goal. There are several ingredients related to a parallel and distributed computing system such as global control mechanisms, synchronous and asynchronous operations and processor interconnections [17]. On top of a parallel and distributed computing system are computational algorithms. Generally, a parallel algorithm, as opposed to a traditional sequential/serial algorithm, is an algorithm which can specify and execute multiple operations at each step and put the results back together again at the end to get the correct result. Alternatively, a distributed algorithm is an algorithm designed to run on multiple processors or computing units, without tight centralized control. In principle, parallel and distributed algorithms usually depend on the architecture of computing systems. However, they still possess some common characterizations that can be studied independently without taking into account the architecture of computing systems [6, 17].

Parallel and distributed optimization algorithms. When we refer to the term *parallel or distributed optimization algorithm* we mean that this is an optimization algorithm that can be implemented in a parallel or distributed manner, respectively. In other words, it is a parallel or distributed algorithm for solving optimization problems. In general, these optimization algorithms are iterative methods. Their main step is to form a new iteration point by employing the oracle at the current iteration [142]. In parallel or distributed optimization methods, this step is usually divided in several tasks corresponding to solving the subproblems concurrently. It is quite often in parallel optimization algorithms that there are some tasks which require a global computation mechanism working on global data and global computations. For instance, to evaluate the objective function f at a given point x in an optimization algorithm we need to know such a point x and then evaluate the objective value $f(x)$. In this case the point x is in fact global data and the evaluation of $f(x)$ is a global computation. An obvious way to collect global data and carry out global computations is to offer a global control mechanism with a shared memory unit. However, in parallel computing systems, there are several way to form and store global data as well as to execute global computations [17]. In contrast to parallel algorithms where global data or global operations are required, the data and the operations in a distributed optimization algorithm are local. Each agent or node in a distributed computing system only communicates and exchanges data internally and possibly with its neighbours via communication links and data channels. The computations only take place internally in each agent. In the following chapters, we note that parallel and distributed optimization algorithms are designed based on exploiting the specific structure of optimization problems instead of exploiting the architecture of parallel and distributed computing systems. More precisely, we concentrate on exploiting separability structure of the problems by applying the dual decomposition framework. This approach

will be combined with other techniques in order to design different classes of parallel and distributed algorithms for solving separable convex and nonconvex optimization problems.

Implementation issues in distributed optimization algorithms. There are several aspects concerning the implementation of a distributed optimization algorithm. We end this section by presenting some points that we find more related to our algorithmic development and implementation. First, implementation of a globalization strategy such as line-search and filtering procedures is a difficult operation in any distributed algorithm. Indeed, in order to carry out such a globalization procedure, we need to evaluate the objective values at certain points. In this case, global data and global computations are required. Second, similarly to globalization strategies, checking stopping criterion via optimality conditions in a distributed implementation is also problematic due to global computation. Finally, synchronization and task location also needs to be taken into account. In our algorithms in the next chapters, each primal subproblems will be solved locally at each agent of a distributed computing system, the workload of solving each subproblem may be different. Therefore, it is important to allocate these workloads properly to trade-off the computational time between each agent and to synchronize the entire system.

6.5 Benchmarking optimization algorithms with performance profiles

In order to compare different optimization algorithms, we can use a concept call *performance profile* in [59]. We briefly present this concept here.

Recall that a performance profile is built based on a set $\mathcal{S}$ of n_s algorithms (solvers) and a collection $\mathcal{P}$ of n_p test problems. Suppose that we build a profile based on computational time. However, the concept presented here can be used for other measurements. Let us denote by

$$T_{p,s} := \textit{computational time required to solve problem } p \textit{ by solver } s.$$

We wish to compare the performance of algorithm s on problem p with the best performance of any algorithm on this problem. First, we compute the performance ratio:

$$r_{p,s} := \frac{T_{p,s}}{\min\{T_{p,\hat{s}} \mid \hat{s} \in \mathcal{S}\}}.$$

We assume that r_M is a given parameter such that $r_M \geq r_{p,s}$ for all p and s, and $r_{p,s} = r_M$ if and only if solver s does not solve problem p. It was shown in [59] that the choice of r_M does not affect the performance evaluation. Then, we consider the function $\tilde{\rho}$ defined by:

$$\tilde{\rho}_s(\tilde{\tau}) := \frac{1}{n_p} \text{size} \{p \in \mathcal{P} \mid r_{p,s} \leq \tilde{\tau}\}, \quad \tilde{\tau} \in \mathbb{R}_+.$$

The function $\tilde{\rho}_s : \mathbb{R} \to [0, 1]$ is the probability for solver s that a performance ratio is within a factor $\tilde{\tau}$ of the best possible ratio. We use the term "performance profile" for the distribution function $\tilde{\rho}_s$ of a performance metric. This function is nondecreasing, piecewise constant and continuous from the right at each breakpoint.

It was claimed in [59] that a plot of the performance profile reveals all of the major performance characteristics. In particular, if the set of problems $\mathcal{P}$ is sufficiently large and representative of problems that are likely to occur in applications, then solvers with large probability $\tilde{\rho}_s(\tilde{\tau})$ are to be preferred. We can also plot the performance profiles in log-scale, i.e.:

$$\rho_s(\tau) := \frac{1}{n_p} \text{size} \{p \in \mathcal{P} \mid \log_2(r_{p,s}) \leq \tau := \log_2 \tilde{\tau}\}.$$

In this case, the number of wins is revealed via the value $\rho_s(0)$. We will use the function ρ_s to benchmark our algorithms in the next chapters.

Chapter 7

Dual decomposition algorithms via the excessive gap technique

In this chapter, we propose two decomposition algorithms for solving separable convex optimization problems of the form (SepCOP). The basic idea is to combine three techniques, namely Lagrangian dual decomposition, excessive gap and primal-dual smoothing to build the algorithms. The main advantage of these algorithms is that they automatically and simultaneously update the algorithmic parameters and do not use any heuristic strategy to tune them. This significantly improves the performance of the algorithms in practice. The convergence of these algorithms is proved under weak conditions imposed on the original problem. The rate of convergence is $O(\frac{1}{k})$, where k is the iteration counter. All the algorithms developed in this chapter can be implemented in a parallel or distributed manner.

Contribution of Chapter 7. Let us state the contribution of this chapter more explicitly as follows:

a) By applying the smoothing technique via prox-functions to the primal problem and the excessive gap condition, we prove an estimate for the duality gap and the feasibility gap of the primal-dual problem. We also show some properties of the smoothed dual function which will be used to design the algorithms.

b) We propose two new decomposition algorithms for solving the separable convex programming problem (SepCOP) which we call the *decomposition algorithm with two primal steps* and the *decomposition algorithm with two dual steps*, respectively. These algorithms are then modified to obtain two different variants. Since all the algorithms are primal-dual, they allow us to obtain simultaneously the primal and dual solutions of the primal and dual problems.

c) The convergence of both algorithms and their variants is proved and the convergence rate is established. We show that the convergence rate of the algorithms is $O(\frac{1}{k})$ which is much higher than $O(\frac{1}{\sqrt{k}})$ in the subgradient methods studied recently in [14, 61, 136], where k is the iteration counter.

d) As a special case, we specialize the second algorithm to the strongly convex case, where we obtain the convergence rate $O(\frac{1}{k^2})$.

e) We also extend the proposed algorithms to the inexact case, where we allow one to solve the primal subproblem of each component inexactly, which is always the case in practice.

Outline of Chapter 7. The content of this chapter is organized as follows. In the next section, we present a smoothing technique via proximity-functions and show the relations between the original functions and the smoothed functions. In Section 7.2, we first discuss the solution of the primal subproblems. Then, we recall the excessive gap concept in [140] and extend it to an inexact case. Sections 7.3 and 7.4 present two new algorithms which we call the *decomposition algorithm with two primal steps* and the *decomposition algorithm with two dual steps*, respectively. The convergence of these algorithms is proved and the convergence rate is established. Two different variants of the proposed algorithms are investigated in Section 7.5. Section 7.6 shows an application of the algorithm in Section 7.4 to the strongly convex case and Section 7.7 is an extension to the inexact case. A theoretical comparison and implementation aspects are presented in Section 7.8. Section 7.9 is devoted to numerical tests. We end this chapter by some conclusion.

7.1 Smoothing via proximity functions

In this section, we present a smoothing technique by using proximity functions as proposed in [145]. This technique was further extended to Lagrangian dual decomposition in [134]. We prove some estimates between the smoothed dual functions and the original dual function g of (SepCOP).

For convenience, we recall the separable convex optimization problem defined by (SepCOP) and its dual problem defined by (6.3.2) as follows:

$$\phi^* := \begin{cases} \min\limits_{x\in\mathbf{R}^n} & \phi(x) := \sum_{i=1}^M \phi_i(x_i) \\ \text{s.t.} & \sum_{i=1}^M (A_i x_i - b_i) = 0, \\ & x_i \in X_i,\ i = 1, \cdots, M, \end{cases} \tag{SepCOP}$$

and

$$g^* := \max_{y\in\mathbf{R}^m} g(y), \tag{7.1.1}$$

where X_i, ϕ_i, A_i and b are defined as before for $i = 1, \cdots, M$ and the dual function g is defined by (6.3.1).

Proximity functions and Bregman distance

Let C be a given nonempty, closed and convex set in $\mathbf{R}^n$. The proximity function of C is defined as follows [139].

Definition 7.1.1. *A function p_C is called a proximity function (σ_C-prox-function) of a convex set C if p_C is continuous, strongly convex with a convexity parameter $\sigma_C > 0$ and $C \subseteq \mathrm{dom}(p_C)$.*

Let us give some examples. The simplest prox-function of C is the quadratic form $p_C(x) := \frac{1}{2}\|x - x^c\|_2^2$, where $x^c \in C$ is an arbitrary point. If C is the standard simplex of the form $C := \{x \in \mathbf{R}^n \mid x \geq 0, \sum_{i=1}^n x_i = 1\}$ then $p_C(x) := \sum_{i=1}^n x_i \ln(x_i) + \ln(n)$ is an 1-prox-function of C, which is known as the entropy function [145].

Associated with the prox function p_C, we can define its conjugate function with respect to C as:

$$p_C^*(s) := \max_x \left\{ s^T x - p_C(x) \mid x \in C \right\}.$$

Since p_C is strongly convex, p_C^* is well-defined and differentiable at any point $s \in \mathbf{R}^n$. We denote by $\widetilde{C} := \{x \mid x = \nabla p_C^*(s),\ s \in \mathbf{R}^n\}$. Clearly, the set $\widetilde{C}$ is convex and $\widetilde{C} \subseteq C$. Suppose that p_C is differentiable on C. Then the following mapping:

$$d_C(x, y) := p_C(y) - p_C(x) - \nabla p_C(x)^T (y - x), \quad \forall x \in \widetilde{C}, y \in C, \tag{7.1.2}$$

is called the *Bregman distance* function. It is obvious that, for $x \neq y$, $d_C(x, y) > 0$ and $d_C(x, x) = 0$. Moreover, for fixed $x \in \widetilde{C}$, $d_C(x, \cdot)$ is strongly convex in y and thus $d_C(x, y) \geq \frac{\sigma_C}{2}\|y - x\|_2^2$.

By strong convexity of p_C, the point x^c and the value p_C^* defined as:

$$x^c := \operatorname*{argmin}_{x \in C} p_C(x) \quad \text{and} \quad p_C^* = p_C(x^c), \tag{7.1.3}$$

are well-defined. Without loss of generality, we can assume that $p_C^* \geq 0$. Otherwise, we shift $\tilde{p}_C(x) := p_C(x) + r_0$, where the constant r_0 is chosen such that $r_0 + p_C^* \geq 0$. As usual, we refer to x^c as the *prox-center point* of C w.r.t. p_C. Let:

$$D_C := \begin{cases} \sup\limits_{x \in C} p_C(x) & \text{if } C \text{ is bounded,} \\ +\infty & \text{otherwise.} \end{cases} \tag{7.1.4}$$

It is clear that:

$$0 \leq p_C^* \leq p_C(x) \leq D_C \leq +\infty, \quad \forall x \in C.$$

If C is bounded then D_C is finite and $D_C := \max_{x \in C} p_C(x)$.

Smoothing via prox-functions

Throughout this chapter, we assume that the following assumption is satisfied.

Asumption A.7.1.8. *Each feasible set X_i of problem* (SepCOP) *is endowed with a σ_{X_i}-prox-function p_{X_i} such that $0 \leq p_{X_i}^* \leq D_{X_i} < +\infty$ $(i = 1, \cdots, M)$.*

Note that Assumption **A.7.1.8** is not very restrictive. Particularly, if X_i is bounded for $i = 1, \cdots, M$ then this assumption is satisfied. Let:

$$p_X(x) := \sum_{i=1}^{M} p_{X_i}(x_i), \; p_X^* := \sum_{i=1}^{M} p_{X_i}^* \geq 0, \text{ and } D_X := \sum_{i=1}^{M} D_{X_i} < +\infty. \tag{7.1.5}$$

We consider the following function:

$$g(y; \beta_1) := \sum_{i=1}^{M} g_i(y; \beta_1), \tag{7.1.6}$$

where

$$g_i(y; \beta_1) := \min_{x_i \in X_i} \left\{ \phi_i(x_i) + y^T(A_i x_i - b_i) + \beta_1 p_{X_i}(x_i) \right\}, \; i = 1, \cdots, M. \tag{7.1.7}$$

Here, $\beta_1 > 0$ is a given parameter called *smoothness parameter*. Note that the function $g(\cdot; \beta_1)$ is well-defined due to the strong convexity of p_{X_i}. For our

convenient future reference, we also call the minimization problem in (7.1.7) as the *primal subproblem*. We denote by $x_i^*(y;\beta_1)$ the solution of (7.1.7), i.e.:

$$x_i^*(y;\beta_1) := \arg\min_{x_i \in X_i} \left\{ \phi_i(x_i) + y^T(A_i x_i - b_i) + \beta_1 p_{X_i}(x_i) \right\}. \tag{7.1.8}$$

In principle, we can use different parameters β_1^i for $i = 1, \cdots, M$ in (7.1.7).

Let $x_{X_i}^c$ be the prox-center of X_i and D_{X_i} be the quantity defined in (7.1.4) for $i = 1, \cdots, M$. Under Assumption **A.7.1.8**, D_{X_i} is finite for $i = 1, \cdots, M$. The following lemma shows the main properties of $g(\cdot;\beta_1)$ whose proof can be found, e.g., in [134, 140].

Lemma 7.1.1. *Suppose that Assumptions **A.6.1.7** and **A.7.1.8** are satisfied. Then, for any $\beta_1 > 0$, the function $g_i(\cdot;\beta_1)$ defined by (7.1.7) is well-defined, concave and continuously differentiable on $\mathbf{R}^m$. Moreover, its gradient w.r.t. y is given by:*

$$\nabla_y g_i(y;\beta_1) = A_i x_i^*(y;\beta_1) - b_i,$$

and it is Lipschitz continuous with a Lipschitz constant $L_i^g(\beta_1) := \frac{\|A_i\|^2}{\beta_1 \sigma_{X_i}}$, $i = 1, \cdots, M$. In addition, the following estimates hold:

$$g_i(y;\beta_1) - \beta_1 D_{X_i} \le g_i(y) \le g_i(y;\beta_1), \quad i = 1, \cdots, M.$$

Consequently, the function $g(\cdot;\beta_1)$ defined by (7.1.6) is concave and continuously differentiable. Its gradient is given by $\nabla g_y(y;\beta_1) := Ax^*(y;\beta_1) - b$ and is Lipschitz continuous with a Lipschitz constant $L^g(\beta_1) := \frac{1}{\beta_1}\sum_{i=1}^2 \frac{\|A_i\|^2}{\sigma_{X_i}}$. Moreover, it holds that:

$$g(y;\beta_1) - \beta_1 D_X \le g(y) \le g(y;\beta_1), \tag{7.1.9}$$

and

$$g(\tilde{y};\beta_1) + \nabla_y g(\tilde{y};\beta_1)^T(y - \tilde{y}) - \frac{L^g(\beta_1)}{2}\|y - \tilde{y}\|^2 \le g(y;\beta_1), \tag{7.1.10}$$

for all y and $\tilde{y}$ in $\mathbf{R}^m$.

The inequalities (7.1.9) show that $g(\cdot;\beta_1)$ is an approximation of g. Moreover, $g(y;\beta_1)$ converges to $g(y)$ as β_1 tends to zero for a fixed $y \in \mathbf{R}^m$.

Remark 7.1.1. *Even without the boundedness of X, if the solution set X^* of (SepCOP) is bounded then, in principle, we can bound the feasible set X by a large compact set which contains all the sampling points generated by the algorithms (see Section 7.3). However, in the first algorithm we do not use D_X in any computational step. It only appears in the theoretical complexity estimates.*

Now, we show the variation of the function $g(y;\cdot)$ w.r.t. the parameter β_1 in the following lemma.

Lemma 7.1.2. *Let $y \in \mathbf{R}^m$. The function $g(y;\cdot)$ defined by (7.1.6) is well-defined, nondecreasing, concave and differentiable in $\mathbf{R}_{++}$. Moreover, it satisfies the following inequality:*

$$g(y;\beta_1) \leq g(y;\tilde{\beta}_1) + (\beta_1 - \tilde{\beta}_1)p_X(x^*(y;\tilde{\beta}_1)),\ \forall \beta_1, \tilde{\beta}_1 \in \mathbf{R}_{++}, \tag{7.1.11}$$

where $x^*(y;\tilde{\beta}_1)$ is defined by (7.1.8).

Proof. Since $g = \sum_{i=1}^M g_i$ and $p_X = \sum_{i=1}^M p_{X_i}$, it is sufficient to prove the inequality (7.1.11) for $g_i(y;\cdot)$, $i = 1, \cdots, M$. Let us fix $y \in \mathbf{R}^m$ and define $\phi_i(x_i;\beta_1) := \phi_i(x_i) + y^T(A_i x_i - b_i) + \beta_1 p_{X_i}(x_i)$, a function of two joint variables x_i and β_1. Since $\phi_i(\cdot;\cdot)$ is strongly convex w.r.t. x_i and linear w.r.t. β_1, $g_i(y;\beta_1) := \min_{x_i \in X_i} \phi_i(x_i;\beta_1)$ is well-defined and concave w.r.t. β_1. Moreover, it is differentiable w.r.t. β_1 and $\nabla_{\beta_1} g(y;\beta_1) = p_i(x_i^*(y;\beta_1)) \geq 0$, where $x_i^*(y;\beta_1)$ is defined by (7.1.8). Thus $g_i(y;\cdot)$ is nonincreasing. By using the concavity of $g_i(y;\cdot)$ we have:

$$g_i(y;\beta_1) \leq g_i(y;\tilde{\beta}_1) + (\beta_1 - \tilde{\beta}_1)\nabla_{\beta_1} g_i(y;\tilde{\beta}_1) = g_i(y;\tilde{\beta}_1) + (\beta_1 - \tilde{\beta}_1)p_i(x_i^*(y;\tilde{\beta}_1)).$$

By summing up these inequalities from $i = 1$ to M and using (7.1.5) we obtain (7.1.11). □

Approximation of the primal objective function

For a given $\beta_2 > 0$, we define a mapping $\psi(\cdot;\beta_2)$ from X to $\mathbf{R}$ by:

$$\psi(x;\beta_2) := \max_{y \in \mathbf{R}^m} \left\{ (Ax - b)^T y - \frac{\beta_2}{2} \|y\|_2^2 \right\}. \tag{7.1.12}$$

This function can be considered as a smoothed version of $\psi(x) := \max_{y \in \mathbf{R}^m} \left\{ (Ax - b)^T y \right\}$ via the 1-prox-function $p(y) := \frac{1}{2}\|y\|^2$. It is easy to show that the unique solution of the maximization problem (7.1.12) is given explicitly as $y^*(x;\beta_2) = \frac{1}{\beta_2}(Ax - b)$ and $\psi(x;\beta_2) = \frac{1}{2\beta_2}\|Ax - b\|^2$. Therefore, $\psi(\cdot;\beta_2)$ is well-defined and differentiable on X. Let

$$f(x;\beta_2) := \phi(x) + \psi(x;\beta_2) = \phi(x) + \frac{1}{2\beta_2}\|Ax - b\|^2. \tag{7.1.13}$$

Then f can be viewed as an approximation of the primal objective function ϕ of problem (SepCOP). The next lemma summarizes the properties of $\psi(\cdot;\beta_2)$.

Lemma 7.1.3. *For any $\beta_2 > 0$, the function $\psi(\cdot;\beta_2)$ defined by (7.1.12) is a quadratic function of the form $\psi(x;\beta_2) = \frac{1}{2\beta_2}\|Ax-b\|^2$ on X. Its gradient vector is given by:*

$$\nabla_x \psi(x;\beta_2) = \frac{1}{\beta_2} A^T(Ax-b), \tag{7.1.14}$$

which is Lipschitz continuous with a Lipschitz constant $L^{\psi}(\beta_2) := \frac{1}{\beta_2}\|A\|_2^2$. Moreover, the following estimate holds for all $x, \hat{x} \in X$:

$$\psi(x;\beta_2) \leq \psi(\hat{x};\beta_2) + \sum_{i=1}^{M}\left[\nabla_{x_i}\psi(\hat{x};\beta_2)^T(x_i-\hat{x}_i) + \frac{L_i^{\psi}(\beta_2)}{2}\|x_i-\hat{x}_i\|_2^2\right]_{[1]}, \tag{7.1.15}$$

where $L_i^{\psi}(\beta_2) := \frac{M}{\beta_2}\|A_i\|_2^2$ for $i = 1, \cdots, M$.

In addition, the following estimates hold:

$$f(x;\beta_2) - \frac{1}{2\beta_2}\|Ax-b\|_2^2 = \phi(x) \leq f(x;\beta_2). \tag{7.1.16}$$

Proof. Since $\psi(x;\beta_2) = \frac{1}{2\beta_2}\|Ax-b\|^2$, it is sufficient to only prove (7.1.15). Indeed, we have:

$$\psi(x;\beta_2) - \psi(\hat{x};\beta_2) - \nabla\psi(\hat{x};\beta_2)^T(x-\hat{x}) = \frac{1}{2\beta_2}\|A(x-\hat{x})\|_2^2. \tag{7.1.17}$$

Then the inequality (7.1.15) follows from (7.1.17) by applying an elementary inequality. □

It is clear from (7.1.16) that $f(\cdot;\beta_2)$ is an approximation of the objective function ϕ of (SepCOP).

Remark 7.1.2. *As we will see later in the algorithms in the next sections, the separability of the term $[\cdot]_{[1]}$ on the right-hand side of (7.1.15) will be used to generate and to solve the second primal subproblems in the algorithms in parallel. Moreover, the Lipschitz constant $L_i^{\psi}(\beta_2) := \frac{M}{\beta_2}\|A_i\|_2^2$ can be computed distributively.*

To conclude this section, let us define the *smoothed dual problem* of (6.3.2) for further reference:

$$g^*(\beta_1) := \max_{y \in \mathbf{R}^m} g(y;\beta_1). \tag{7.1.18}$$

Problem (7.1.18) is convex. Moreover, since the function $g(\cdot;\beta_1)$ is continuously differentiable and its gradient is Lipschitz continuous for any $\beta_1 > 0$, one can apply the fast gradient method in [142] to solve this problem, see, e.g. [134].

7.2 Solution of primal subproblems and excessive gap condition

In this section, we first show how we can solve the primal subproblems inexactly and then we recall the excessive gap condition introduced by Nesterov in [140] in the framework of decomposition.

Inexact solution of primal subproblems

In practice, solving the primal subproblem (7.1.7) exactly is only *conceptual*. In this section, we assume that we only solve this problem up to a given accuracy. In other words, the solution $x_i^*(y;\beta_1)$ of (7.1.7) is approximated by:

$$\tilde{x}_i^*(y;\beta_1) :\approx \arg\min_{x_i\in X_i} \left\{\phi_i(x_i) + y^T(A_ix_i - b_i) + \beta_1 p_{X_i}(x_i)\right\}, \tag{7.2.1}$$

for $i = 1,\cdots,M$, in the sense of the following definition.

Definition 7.2.1. *We say that the point $\tilde{x}_i^*(y;\beta_1)$ approximates $x_i^*(y;\beta_1)$ defined by (7.1.8) up to a given accuracy $\varepsilon_i \geq 0$ if:*

a) *it is feasible to X_i, i.e. $\tilde{x}_i^*(y;\beta_1) \in X_i$;*

b) *and the following condition is satisfied:*

$$0 \leq h_i(\tilde{x}_i^*(y;\beta_1);y,\beta_1) - h_i(x_i^*(y;\beta_1);y,\beta_1) \leq \frac{\beta_1\sigma_{X_i}}{2}\varepsilon_i^2, \tag{7.2.2}$$

where $h_i(x_i;y,\beta_1) := \phi_i(x_i) + y^T(A_ix_i - b_i) + \beta_1 p_{X_i}(x_i)$ for $i = 1,\cdots,M$.

Note that the condition (7.2.2) is computable. In practice, we often meet the case where X_i is simple such that the projection on X_i can be computed efficiently. Hence, one can apply classical convex optimization algorithms to solve (7.2.1) up to a given accuracy such that a) and b) are satisfied.

Since $h_i(\cdot;y,\beta_1)$ is strongly convex with a convexity parameter $\beta_1\sigma_{X_i} > 0$, one can estimate:

$$\frac{\beta_1\sigma_{X_i}}{2}\|\tilde{x}_i^*(y;\beta_1) - x_i^*(y;\beta_1)\|^2 \leq h_i(\tilde{x}_i^*(y;\beta_1);y,\beta_1) - h_i(x_i^*(y;\beta_1);y,\beta_1), \tag{7.2.3}$$

where $h_i(\cdot;y,\beta_1)$ is defined as in Definition 7.2.1. Consequently, we have:

$$\|\tilde{x}_i^*(y;\beta_1) - x_i^*(y;\beta_1)\| \leq \varepsilon_i, \quad i = 1,\cdots,M.$$

Let $\tilde{x}^*(y;\beta_1) := (\tilde{x}_1^*(y;\beta_1),\ldots,\tilde{x}_M^*(y;\beta_1))$ and

$$\widetilde{\nabla}_y g(y;\beta_1) := A\tilde{x}^*(y;\beta_1) - b. \tag{7.2.4}$$

The quantity $\widetilde{\nabla}_y g(\cdot;\beta_1)$ can be referred to as an approximation of the gradient $\nabla_y g(\cdot;\beta_1)$ defined in Lemma 7.1.1. If we denote by $\varepsilon := (\varepsilon_1,\ldots,\varepsilon_M)^T$ the vector of accuracies then we can easily estimate:

$$\left\|\widetilde{\nabla}_y g(y;\beta_1) - \nabla_y g(y;\beta_1)\right\| = \|A(\tilde{x}^*(y;\beta_1) - x^*(y;\beta_1))\| \le \|A\|\,\|\varepsilon\|. \tag{7.2.5}$$

Excessive gap condition

Since the primal-dual gap of the primal and dual problems (SepCOP)-(7.1.1) is measured by $e(x,y) := \phi(x) - g(y)$, if the gap e is equal to zero for a given feasible point (x,y) then this point is an optimal solution of (SepCOP)-(7.1.1). In this section, we apply the technique called *excessive gap* introduced by Nesterov in [140] to the Lagrangian dual decomposition framework. We also define an inexact excessive gap condition by modifying the exact one. More precisely, we give the following definition.

Definition 7.2.2. *We say that a point $(\bar{x},\bar{y}) \in X \times \mathbf{R}^m$ satisfies the excessive gap condition w.r.t. two smoothness parameters $\beta_1 > 0$ and $\beta_2 > 0$ if:*

$$f(\bar{x};\beta_2) \le g(\bar{y};\beta_1), \tag{7.2.6}$$

where $f(\cdot;\beta_2)$ and $g(\cdot;\beta_1)$ are defined by (7.1.13) and (7.1.6), respectively. For a given tolerance $\delta \ge 0$, we say that $(\bar{x},\bar{y})$ satisfies an inexact excessive gap condition (δ-excessive gap condition) w.r.t. two smoothness parameters β_1 and β_2 if:

$$f(\bar{x};\beta_2) \le g(\bar{y};\beta_1) + \delta, \tag{7.2.7}$$

The following lemma provides an upper bound estimate for the duality gap and the feasibility gap of the problems (SepCOP)-(7.1.1).

Lemma 7.2.1. *Suppose that $(\bar{x},\bar{y}) \in X \times \mathbf{R}^m$ and satisfies the δ-excessive gap condition (7.2.7) w.r.t. two positive smoothness parameters β_1 and β_2 for $\delta \ge 0$. Then for any $y^* \in Y^*$, we have:*

$$-\|y^*\|\,\|A\bar{x}-b\| \le \phi(\bar{x}) - g(\bar{y}) \le \beta_1 D_X + \delta - \frac{1}{2\beta_2}\|A\bar{x}-b\|^2 \le \beta_1 D_X + \delta, \tag{7.2.8}$$

and

$$\|A\bar{x}-b\| \le \beta_2\Big\{\|y^*\| + \Big[\|y^*\|^2 + 2\beta_1\beta_2^{-1}D_X + 2\delta\beta_2^{-1}\Big]^{1/2}\Big\}. \tag{7.2.9}$$

Proof. Suppose that $\bar{x}$ and $\bar{y}$ satisfy the condition (7.2.7). For a given $y^* \in Y^*$, one has:

$$g(\bar{y}) \leq g(y^*) = \min_{x \in X} \left\{ \phi(x) + (Ax - b)^T y^* \right\} \leq \phi(\bar{x}) + (A\bar{x} - b)^T y^*$$

$$\leq \phi(\bar{x}) + \|A\bar{x} - b\| \, \|y^*\| \, ,$$

which implies the first inequality of (7.2.8). By using Lemmas 7.1.1 and 7.1.3 we have:

$$\phi(\bar{x}) - g(\bar{y}) \overset{(7.1.9)+(7.1.16)}{\leq} f(\bar{x}; \beta_2) - g(\bar{y}; \beta_1) + \beta_1 D_X - \frac{1}{2\beta_2} \|A\bar{x} - b\|^2 \, .$$

Now, by substituting the condition (7.2.7) into this inequality, we obtain the second inequality of (7.2.8). Let $\xi := \|Ax - b\|$. It follows from (7.2.8) that $\xi^2 - 2\beta_2 \|y^*\| \, \xi - 2\beta_1 \beta_2 D_X - 2\beta_2 \delta \leq 0$. The estimate (7.2.9) follows from this inequality after a few simple calculations. □

We note that the conclusion of Lemma 7.2.1 hold for any $y \in Y^*$. Under Assumption **A.6.1.7**, the dual solution set Y^* is bounded. One can define the feasibility gap $\mathcal{F}(x) := \|Ax - b\|$ and:

$$R_{Y^*} := \min_{y^* \in Y^*} \|y^*\| \, , \quad D_{Y^*} := \min_{y^* \in Y^*} \left[\|y^*\| + \left(\|y^*\|^2 + 2D_X \right)^{1/2} \right]. \tag{7.2.10}$$

Then R_{Y^*} and D_{Y^*} are finite. The estimates (7.2.8) and (7.2.9) can be simplified as follows:

$$\begin{aligned} &-R_{Y^*} \mathcal{F}(\bar{x}) \leq \phi(\bar{x}) - g(\bar{y}) \leq \beta_1 D_X + \delta, \\ &\mathcal{F}(\bar{x}) = \|A\bar{x} - b\| \leq 2\beta_2 R_{Y^*} + \sqrt{2\beta_1 \beta_2 D_X + 2\beta_2 \delta}. \end{aligned} \tag{7.2.11}$$

If $\delta = 0$ and $\beta_1 = \beta_2$ then the feasibility gap is estimated by $\mathcal{F}(\bar{x}) \leq \beta_2 D_{Y^*}$.

Approximate proximal-gradient mapping

Let us consider the approximate function $f(\cdot; \beta_2) := \phi(\cdot) + \psi(\cdot; \beta_2)$ of ϕ defined by (7.1.13). For $i = 1, \cdots, M$, since ϕ_i is only assumed to be convex and not necessarily smooth, while $\psi(\cdot; \beta_2)$ is quadratic, if we define:

$$\begin{aligned} &q_i^{\psi}(x_i; \hat{x}, \beta_2) := M^{-1} \psi(\hat{x}; \beta_2) + \nabla_{x_i} \psi(\hat{x}; \beta_2)^T (x_i - \hat{x}_i) + \frac{L_i^{\psi}(\beta_2)}{2} \|x_i - \hat{x}_i\|^2, \\ &\varphi_i(x; \hat{x}, \beta_2) := \phi_i(x_i) + q_i^{\psi}(x_i; \hat{x}, \beta_2), \end{aligned} \tag{7.2.12}$$

then the mapping:

$$\mathcal{P}_i(\hat{x},\beta_2) := \arg\min_{x_i\in X_i}\varphi_i(x_i;\hat{x},\beta_2) = \arg\min_{x_i\in X_i}\left\{\phi_i(x_i)+q_i^{\psi}(x_i;\hat{x},\beta_2)\right\}, \quad (7.2.13)$$

is well-defined due to the strong convexity of $q_i^{\psi}(\cdot;\hat{x},\beta_2)$, where $L_i^{\psi}(\beta_2) := \frac{M\|A_i\|^2}{\beta_2}$ is the Lipschitz constant of $\nabla_{x_i}\psi(\cdot;\beta_2)$ defined in Lemma 7.1.3.

We also assume that, for given y and β_2, we can only solve the minimization problem (7.2.13) up to a given accuracy $\varepsilon_i \geq 0$ to obtain an approximate solution $\widetilde{\mathcal{P}}_i(\cdot;\beta_2)$ in the sense of Definition 7.2.1, i.e. $\widetilde{P}_i(\hat{x},\beta_2) \in X_i$ and:

$$\varphi_i(\widetilde{\mathcal{P}}_i(\hat{x},\beta_2);\hat{x},\beta_2) - \varphi_i(\mathcal{P}_i(\hat{x},\beta_2);\hat{x},\beta_2) \leq \frac{L_i^{\psi}(\beta_2)}{2}\varepsilon_i^2, \quad i = 1,\cdots,M. \quad (7.2.14)$$

We denote by $\mathcal{P} := (\mathcal{P}_1,\cdots,\mathcal{P}_M)$ and $\widetilde{\mathcal{P}} := (\widetilde{\mathcal{P}}_1,\cdots,\widetilde{\mathcal{P}}_M)$ the proximal-gradient and approximate proximal-gradient mappings of the function $\varphi := (\varphi_1,\cdots,\varphi_M)$.

Remark 7.2.1. *In particular, if ϕ_i is differentiable and its gradient is Lipschitz continuous with a Lipschitz constant $L^{\phi_i} > 0$ then one can replace the proximal-gradient mapping $\mathcal{P}_i$ by the following one:*

$$\mathcal{G}_i(\hat{x},\beta_2) := \arg\min_{x_i\in X_i}\left\{\nabla_{x_i}(\phi_i(\hat{x}_i)+\psi(\hat{x};\beta_2))^T(x_i-\hat{x}_i)+\frac{\tilde{L}_i(\beta_2)}{2}\|x_i-\hat{x}_i\|_2^2\right\},$$

where $\tilde{L}_i(\beta_2) := L^{\phi_i} + L_i^{\psi}(\beta_2)$. Note that the minimization problem defined $\mathcal{G}_i$ is a quadratic program with convex constraints. If X_i is polytopic then this problem becomes a convex quadratic programming problem.

Similar to $\tilde{P}_i$, we can define an approximate operator $\widetilde{\mathcal{G}}_i$ of $\mathcal{G}_i$ when the minimization problem (7.2.1) is solved approximately in the sense of Definition 7.2.1.

In contrast to $f(\cdot;\beta_2)$, the smoothed dual function $g(\cdot;\beta_1)$ is differentiable and its gradient is Lipschitz continuous on $\mathbf{R}^m$ with a Lipschitz constant $L^g(\beta_1)$ as stated in Lemma 7.1.1, we can define the following mapping:

$$\mathcal{G}^*(\hat{y};\beta_1) := \operatorname*{argmax}_{y\in\mathbf{R}^m}\left\{\nabla_y g(\hat{y};\beta_1)^T(y-\hat{y}) - \frac{L^g(\beta_1)}{2}\|y-\hat{y}\|_2^2\right\}, \quad (7.2.15)$$

where $\nabla_y g(\hat{y};\beta_1) = Ax^*(\hat{y};\beta_1) - b$. However, since $x^*(\hat{y};\beta_1)$ can not be computed exactly, we use its approximation $\tilde{x}^*(\hat{y};\beta_1)$ to form the problem:

$$\widetilde{\mathcal{G}}^*(\hat{y};\beta_1) := \operatorname*{argmax}_{y\in\mathbf{R}^m}\left\{\widetilde{\nabla}_y g(\hat{y};\beta_1)^T(y-\hat{y}) - \frac{L^g(\beta_1)}{2}\|y-\hat{y}\|_2^2\right\}. \quad (7.2.16)$$

This problem can explicitly be solved to get the unique solution:

$$\widetilde{\mathcal{G}}^*(\hat{y};\beta_1) = \hat{y} + L^g(\beta_1)^{-1}(A\tilde{x}^*(\hat{y};\beta_1) - b). \tag{7.2.17}$$

The mapping $\mathcal{G}^*(\cdot;\beta_1)$ is called gradient mapping of the function $g(\cdot;\beta_1)$ (see [142]). We also refer to $\widetilde{\mathcal{G}}^*(\cdot;\beta_1)$ as an approximate gradient mapping of $g(\cdot;\beta_1)$.

Remark 7.2.2. *We notice that, similar to [140], we can define the gradient mappings defined above by using the Bregman distance (7.1.2) and adapt the algorithms developed in the next sections by using these new mappings.*

7.3 Decomposition algorithm with two primal steps

In this section, we derive an iterative decomposition algorithm for solving (SepCOP) based on the excessive gap technique. We call this method the *decomposition algorithm with two primal steps*. The aim is to generate a point $(\bar{x}^k, \bar{y}^k) \in X \times \mathbf{R}^m$ at each iteration k such that this point maintains the excessive gap condition (7.2.6) while driving the parameters β_1^k and β_2^k to zero.

Finding a starting point

First, we state that the excessive gap condition (7.2.6) is well-defined by showing that there exists a point $(\bar{x}, \bar{y})$ that satisfies (7.2.6). We assume that the Lipschitz constant $L_i^{\psi}(\beta_2)$ defined in Lemma 7.1.3 is chosen by $L_i^{\psi}(\beta_2) = \frac{M}{\beta_2}\|A_i\|_2^2$ for all $i = 1, \cdots, M$. Let

$$\bar{L} := \Big[M \max_{1\le i\le M} \frac{\|A_i\|_2^2}{\sigma_{X_i}}\Big]^{1/2}. \tag{7.3.1}$$

We have the following lemma whose proof is postponed to Appendix A.1.

Lemma 7.3.1. *Suppose that $x^c \in X$ is the prox-center of the convex set X and the constant $\bar{L}$ is defined by (7.3.1). For a given $\beta_2 > 0$, let us define:*

$$\bar{y} := \beta_2^{-1}(Ax^c - b) \quad \text{and} \quad \bar{x} := \mathcal{P}(x^c;\beta_2). \tag{7.3.2}$$

If the parameter β_1 is chosen such that $\beta_1\beta_2 \ge \bar{L}^2$ then $(\bar{x}, \bar{y})$ satisfies the excessive gap condition (7.2.6).

Alternatively, for a given $\beta_1 > 0$, we define:

$$\bar{x} := x^*(0^m;\beta_1) \quad \text{and} \quad \bar{y} := L^g(\beta_1)^{-1}(A\bar{x} - b). \tag{7.3.3}$$

If the parameter β_2 is chosen such that $\beta_1\beta_2 \ge \bar{L}^2$ then $(\bar{x}, \bar{y})$ satisfies the excessive gap condition (7.2.6).

Main iteration scheme

Suppose that $(\bar{x}, \bar{y}) \in X \times \mathbf{R}^m$ and satisfies the excessive gap condition (7.2.6). We generate a new point $(\bar{x}^+, \bar{y}^+) \in X \times \mathbf{R}^m$ by applying the following scheme:

$$(\bar{x}^+, \bar{y}^+) := \mathcal{S}_{2\text{ps}}(\bar{x}, \bar{y}; \beta_1, \beta_2^+, \tau) \Leftrightarrow \begin{cases} \hat{x} := (1-\tau)\bar{x} + \tau x^*(\bar{y}; \beta_1), \\ \bar{y}^+ := (1-\tau)\bar{y} + \tau y^*(\hat{x}; \beta_2^+), \\ \bar{x}^+ := \mathcal{P}(\hat{x}; \beta_2^+), \end{cases} \tag{7.3.4}$$

and

$$\beta_1^+ := (1-\tau)\beta_1 \text{ and } \beta_2^+ = (1-\tau)\beta_2, \tag{7.3.5}$$

where $\mathcal{P}(\cdot; \beta_2^+)$ is defined in (7.2.13) and $\tau \in (0,1)$ will be chosen appropriately.

Remark 7.3.1. *In the scheme (7.3.4), the points $x^*(\bar{y}; \beta_1)$, $\hat{x}$ and $\bar{x}^+$ can be computed in parallel. To compute these points we need to perform two primal steps corresponding to solving M convex programming subproblems (7.1.8) and M convex primal subproblems (7.2.14) as indicated by the name of the algorithm.*

The following theorem shows that the scheme (7.3.4)-(7.3.5) maintains the excessive gap condition (7.2.6). The proof of this theorem is postponed to Appendix A.1.

Theorem 7.3.2. *Suppose that $(\bar{x}, \bar{y}) \in X \times \mathbf{R}^m$ satisfies (7.2.6) w.r.t. two values $\beta_1 > 0$ and $\beta_2 > 0$. Then if the parameter τ is chosen such that $\tau \in (0,1)$ and:*

$$\beta_1 \beta_2 \geq \frac{\tau^2}{(1-\tau)^2} \bar{L}^2. \tag{7.3.6}$$

then the new point $(\bar{x}^+, \bar{y}^+)$ generated by scheme (7.3.4)-(7.3.5) is in $X \times \mathbf{R}^m$ and maintains the excessive gap condition (7.2.6) w.r.t. two new values $\beta_1^+ < \beta_1$ and $\beta_2^+ < \beta_2$.

If ϕ_i is convex and differentiable such that its gradient is Lipschitz continuous with a Lipschitz constant $L_i^{\phi_i} \geq 0$ for some $i = 1, \cdots, M$, then instead of using the proximal-gradient mapping $\mathcal{P}_i(\cdot; \beta_2)$ in (7.3.4) we can use the mapping $\mathcal{G}$ defined in Remark 7.2.1 as stated in the following corollary. The proof of the corollary is moved to Appendix A.1.

Corollary 7.3.1. *If the function ϕ_i is differentiable and its gradient is Lipschitz continuous with a Lipschitz constant L^{ϕ_i} for some $i \in I_{\mathcal{G}} \subseteq \{1, \cdots, M\}$. Then if the parameter is chosen such that $\tau \in (0,1)$ and:*

$$\begin{cases} \frac{(1-\tau)}{\tau^2}\beta_1 \sigma_{X_i} \geq L^{\phi_i} + \frac{M\|A_i\|^2}{(1-\tau)\beta_2} & \text{if } i \in I_{\mathcal{G}}, \\ \beta_1 \beta_2 \geq \frac{\tau^2}{(1-\tau)^2}\bar{L}^2 & \text{otherwise}, \end{cases} \tag{7.3.7}$$

then the point $\hat{\bar{x}}^+ := (\hat{\bar{x}}_1^+, \ldots, \hat{\bar{x}}_M^+)$ *computed by:*

$$\hat{\bar{x}}_i^+ := \begin{cases} \mathcal{G}_i(\hat{x}; \beta_2^+) & \text{if } i \in I_{\mathcal{G}}, \\ \mathcal{P}_i(\hat{x}; \beta_2^+) & \text{otherwise}, \end{cases} \tag{7.3.8}$$

for $i = 1, \cdots, M$*, instead of* $\bar{x}^+$ *in the scheme* $\mathcal{S}_{2ps}$*, still maintains the excessive gap condition* (7.2.6).

Note that if the feasible set X_i ($i \in I_{\mathcal{G}}$) is simple, e.g. polytopic, then computing $\hat{\bar{x}}_i^+$ by (7.3.8) requires a lower computational cost than computing $\bar{x}_i^+$.

Step size update

In the next step we show how to update the parameter τ such that the condition (7.3.6) holds. From the update rule (7.3.5) we have $\beta_1^+\beta_2^+ = (1-\tau)^2\beta_1\beta_2$. Suppose that β_1 and β_2 satisfy the condition (7.3.6), i.e. $\beta_1\beta_2 \geq \frac{\tau^2}{(1-\tau)^2}\bar{L}^2$. The condition $\beta_1^+\beta_2^+ \geq \frac{\tau_+^2}{(1-\tau_+)^2}\bar{L}^2$ is satisfied if $\frac{\tau^2}{(1-\tau)^2} \geq \frac{\tau_+^2}{(1-\tau)^2(1-\tau_+)^2}$. This condition leads to $\tau \geq \frac{\tau_+}{1-\tau_+}$. Hence, (7.3.4)-(7.3.5) are well-defined. At the first iteration $k = 0$, both condition (7.3.6) and $\beta_1\beta_2 \geq \bar{L}^2$ in Lemma 7.3.1 need to be satisfied. This leads to $0 < \tau_0 \leq 0.5$.

Now, we define a rule to update the step size parameter τ.

Lemma 7.3.2. *Suppose that* τ_0 *is arbitrarily chosen in* $(0, \frac{1}{2}]$*. Then the sequence* $\{\tau_k\}_{k\geq 0}$ *generated by:*

$$\tau_{k+1} := \frac{\tau_k}{\tau_k + 1} \tag{7.3.9}$$

satisfies the following formula:

$$\tau_k = \frac{\tau_0}{1 + \tau_0 k}, \quad \forall k \geq 0. \tag{7.3.10}$$

Moreover, the sequence $\{\beta_k\}_{k\geq 0}$ *generated by* $\beta_{k+1} = (1-\tau_k)\beta_k$ *for a fixed* $\beta_0 > 0$ *satisfies:*

$$\beta_k = \frac{\beta_0}{\tau_0 k + 1}, \quad \forall k \geq 0. \tag{7.3.11}$$

Proof. Since $\tau_{k+1}^{-1} = \tau_k^{-1} + 1$ for $k \geq 0$ by (7.3.9), the update formula (7.3.10) holds. Moreover, since $\beta_{k+1} = \beta_0 \prod_{i=0}^{k}(1-\tau_i)$, by substituting (7.3.10) into the last expression, after some simple calculations we obtain (7.3.11). □

Remark 7.3.3. *Since* $\tau_0 \in (0, \frac{1}{2}]$, *we see from Lemma 7.3.2 that with* $\tau_0 := 0.5$ *the right-hand side of* (7.3.11) *is minimized. In this case, the update rule of* τ_k *is simplified to* $\tau_k := \frac{1}{k+2}$ *for* $k \geq 0$.

The algorithm and its worst-case complexity

Now, we combine the results of Lemma 7.3.1, Theorem 7.3.2 and Lemma 7.3.2 in order to build the following algorithm.

Algorithm 7.3.1.(*Decomposition algorithm with two primal steps*).
Initialization: Perform the following steps:

1. Set $\tau_0 := 0.5$. Choose $\beta_1^0 > 0$ and $\beta_2^0 > 0$ such that $\beta_1^0 = \beta_2^0 := \bar{L}$.
2. Compute $\bar{x}^0$ and $\bar{y}^0$ from (7.3.2) or (7.3.3).

Iteration: For $k = 0, 1, \cdots$, perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. Update the smoothness parameter $\beta_2^{k+1} := (1 - \tau_k)\beta_2^k$.
3. Compute $\bar{x}^{k+1}$ *in parallel* and $\bar{y}^{k+1}$ by using scheme (7.3.4):
$$(\bar{x}^{k+1}, \bar{y}^{k+1}) := \mathcal{S}_{2\text{ps}}(\bar{x}^k, \bar{y}^k; \beta_1^k, \beta_2^{k+1}, \tau_k).$$
4. Update the smoothness parameter: $\beta_1^{k+1} := (1 - \tau_k)\beta_1^k$.
5. Update the step size τ_k by: $\tau_{k+1} := \frac{1}{k+3}$.

End.

As mentioned in Remark 7.3.1, there are two steps in the scheme $\mathcal{S}_{2\text{ps}}$ of Algorithm 7.3.1 that can be parallelized. The first step is finding $x^*(\bar{y}^k; \beta_1)$ and the second is computing $\bar{x}^{k+1}$. In general, both steps require one to solve M convex primal subproblems in parallel. The stopping criterion of Algorithm 7.3.1 will be discussed in Section 7.8.

The next theorem proves the convergence of Algorithm 7.3.1.

Theorem 7.3.4. *Let* $\{(\bar{x}^k, \bar{y}^k)\}$ *be a sequence generated by Algorithm 7.3.1. Then the following duality gap and feasibility gap hold:*

$$-R_{Y^*} \left\| A\bar{x}^k - b \right\| \leq \phi(\bar{x}^k) - g(\bar{y}^k) \leq \frac{2\bar{L}D_X}{k+2}, \tag{7.3.12}$$

and

$$\mathcal{F}(\bar{x}^k) = \left\|A\bar{x}^k - b\right\| \leq \frac{2\bar{L}D_{Y^*}}{k+2}, \tag{7.3.13}$$

where $\bar{L}$, D_X, R_{Y^*} *and* D_{Y^*} *are defined by* (7.3.1), (7.1.5) *and* (7.2.10), *respectively.*

Proof. By the choice of $\beta_1^0 = \beta_2^0 = \bar{L}$ and Step 1 in the initialization phase of Algorithm 7.3.1 we see that $\beta_1^k = \beta_2^k$ for all $k \geq 0$. Moreover, since $\tau_0 = 0.5$, by Lemma 7.3.2, we have $\beta_1^k = \beta_2^k = \frac{\beta_0}{\tau_0 k+1} = \frac{\bar{L}}{0.5k+1}$. By applying Lemma 7.2.1 with β_1 and β_2 equal to β_1^k and β_2^k respectively, we obtain the bounds (7.4.9) and (7.4.10). □

Remark 7.3.5. *The worst-case complexity of Algorithm 7.3.1 is* $O(\frac{2\bar{L}R_0}{\varepsilon})$, *where* $R_0 := \max\{D_X, D_{Y^*}\}$. *Moreover, the constants in the bounds* (7.4.9) *and* (7.4.10) *also depend on the choice of* β_1^0 *and* β_2^0, *which satisfy the condition* $\beta_1^0\beta_2^0 \geq \bar{L}$ *of Lemma 7.3.1. The values of* β_1^0 *and* β_2^0 *will affect the magnitudes of the duality and feasibility gaps. By substituting* $\bar{L}$ *into the worst-case complexity formula, we obtain an alternative:*

$$O\left(\left[M \max_{1\leq i\leq M}\left\{\sigma_{X_i}^{-1}\left\|A_i\right\|^2\right\}\right]^{1/2} R_0\varepsilon^{-1}\right).$$

This formula shows that the worst-case complexity of Algorithm 7.3.1 also depends on the size of the problem. In particular, it depends on M, *the number of components of the problem.*

7.4 Decomposition algorithm with two dual steps

In Algorithm 7.3.1 two primal steps w.r.t. computing $x^*(\bar{y};\beta_1)$ and $\bar{x}^+$ are required. Since computing a primal step is equivalent to solving M convex primal subproblems in parallel, the cost per iteration may be relatively high. To overcome this disadvantage, we show in this section that we can only perform one primal step and two dual steps to maintain the excessive gap condition (7.2.6). Note that computing a dual step only needs matrix-vector multiplication.

Main iteration scheme

Let us assume that $(\bar{x}, \bar{y})$ is a given point in $X \times \mathbf{R}^m$ and satisfies the excessive gap condition (7.2.6) w.r.t. β_1 and β_2. The aim is to compute a new point

$(\bar{x}^+, \bar{y}^+)$ such that the condition (7.2.6) holds for new values β_1^+ and β_2^+ with $\beta_1^+ < \beta_1$ and $\beta_2^+ < \beta_2$. This can be done by performing the following scheme:

$$(\bar{x}^+, \bar{y}^+) := \mathcal{S}_{2\text{ds}}(\bar{x}, \bar{y}, \beta_1, \beta_2, \tau) \Longleftrightarrow \begin{cases} \hat{y} := (1-\tau)\bar{y} + \tau y^*(\bar{x}; \beta_2) \\ \bar{x}^+ := (1-\tau)\bar{x} + \tau x^*(\hat{y}; \beta_1) \\ \bar{y}^+ := \mathcal{G}^*(\hat{y}; \beta_1) \end{cases} \tag{7.4.1}$$

$$\beta_1^+ := (1 - \alpha\tau)\beta_1 \quad \text{and} \quad \beta_2^+ := (1-\tau)\beta_2, \tag{7.4.2}$$

where $0 < \alpha \leq 1$ is a damping factor and $\tau \in (0, 1)$ is a step size which will be appropriately updated, respectively. Note that computing $\hat{y}$ and $\bar{y}^+$ is just matrix-vector multiplication, i.e.:

$$\hat{y} := (1-\tau)\bar{y} + \tau\beta_2^{-1}(A\bar{x} - b) \quad \text{and} \quad \bar{y}^+ = \hat{y} + L^g(\beta_1)^{-1}(Ax^*(\hat{y}; \beta_1) - b). \tag{7.4.3}$$

The only step $\bar{x}^+$ requires one to solve M convex primal subproblems in *parallel*.

Now, in order to obtain a positive value α, we impose the following assumption.

Asumption A.7.4.9. *The lower bound p_X^* defined in (7.1.5) is positive.*

This assumption is only technical. Since in implementation, we can add any positive constant into p_X to obtain $p_X^* > 0$ without changing any step in the algorithm. However, this leads to a corresponding increase of the quantity D_X. Now, we define:

$$\alpha := \frac{p_X(x^*(\hat{y}; \beta_1))}{D_X} \in [\alpha^*, 1), \quad \text{where } \alpha^* := \frac{p_X^*}{D_X} > 0. \tag{7.4.4}$$

The next theorem provides a condition such that $(\bar{x}^+, \bar{y}^+)$ generated by (7.4.1) satisfies the excessive gap condition (7.2.6). The proof of this theorem can be found in Appendix A.1.

Theorem 7.4.1. *Suppose that Assumptions **A.6.1.7**, **A.7.1.8** and **A.7.4.9** are satisfied. Let $(\bar{x}, \bar{y}) \in X \times \mathbf{R}^m$ be a point satisfying the excessive gap condition (7.2.6) w.r.t. two values β_1 and β_2. Then if α is defined by (7.4.4) and the parameter τ is chosen such that $\tau \in (0, 1)$ and:*

$$\beta_1\beta_2 \geq \frac{\tau^2}{1-\tau}\bar{L}^2, \tag{7.4.5}$$

where $\bar{L}$ is defined by (7.3.1), then the new point $(\bar{x}^+, \bar{y}^+)$ generated by (7.4.1) and (7.4.2) also satisfy the excessive gap condition (7.2.6) w.r.t two new values $\beta_1^+ < \beta_1$ and $\beta_2^+ < \beta_2$

Step size update

Next, we show how to update the step size $\tau \in (0,1)$. Indeed, from the condition (7.4.5) of Theorem 7.4.1 we have $\beta_1\beta_2 \geq \frac{\tau^2}{1-\tau}\bar{L}^2$. By combining this inequality and (7.4.2) we have $\beta_1^+\beta_2^+ = (1-\tau)(1-\alpha\tau)\beta_1\beta_2 \geq (1-\alpha\tau)\tau^2\bar{L}^2$. In order to ensure $\beta_2^+\beta_2^+ \geq \frac{\tau_+^2}{1-\tau_+}\bar{L}^2$ for the next iteration, we need $(1-\alpha\tau)\tau^2 \geq \frac{\tau_+^2}{1-\tau_+}$. Since $\tau, \tau_+ \in (0,1)$ and $\alpha \in (0,1]$, we have:

$$0 < \tau_+ \leq 0.5\tau\left\{\left[(1-\alpha\tau)^2\tau^2 + 4(1-\alpha\tau)\right]^{1/2} - (1-\alpha\tau)\tau\right\} < \tau.$$

Hence, if we choose $\tau_+ := 0.5\tau\left[\left[(1-\alpha\tau)^2\tau^2 + 4(1-\alpha\tau)\right]^{1/2} - (1-\alpha\tau)\tau\right]$ then we obtain the tightest rule for updating τ.

Based on the above analysis, we eventually define a sequence $\{\tau_k\}_{k\geq 0}$ as follows:

$$\tau_{k+1} := \frac{\tau_k}{2}\left\{\left[(1-\alpha_k\tau_k)^2\tau_k^2 + 4(1-\alpha_k\tau_k)\right]^{1/2} - (1-\alpha_k\tau_k)\tau_k\right\}, \tag{7.4.6}$$

where $\tau_0 \in (0,1)$ is given and $\alpha_k := p_X(\tilde{x}^*(\hat{y}^k;\beta_1^k))/D_X \in [\alpha^*, 1)$.

Assumption **A.7.4.9** implies that the factor α in (7.4.2) is positive and bounded below away from zero. The following lemma shows an explicit formula to calculate τ_k whose proof can be found in Appendix A.1.

Lemma 7.4.1. *Suppose that Assumption **A.7.4.9** is satisfied. Let $\{\tau_k\}_{k\geq 0}$ be a sequence generated by (7.4.6) for a given τ_0 such that $0 < \tau_0 < [\max\{1, \alpha^*(1-\alpha^*)^{-1}\}]^{-1}$. Then:*

$$(k+\tau_0^{-1})^{-1} \leq \tau_k \leq \left[0.5(1+\alpha^*)k + \tau_0^{-1}\right]^{-1}. \tag{7.4.7}$$

Moreover, the sequences $\{\beta_1^k\}_{k\geq 0}$ and $\{\beta_2^k\}_{k\geq 0}$ generated by (7.4.2) satisfy:

$$\frac{\gamma}{(\tau_0 k+1)^{2/(1+\alpha^*)}} \leq \beta_1^{k+1} \leq \frac{\beta_1^0}{(\tau_0 k+1)^{\alpha^*}}, \quad \beta_2^{k+1} \leq \frac{\beta_2^0(1-\tau_0)}{\tau_0 k+1}, \tag{7.4.8}$$

$$\text{and} \qquad \beta_1^k\beta_2^{k+1} = \beta_1^0\beta_2^0\frac{(1-\tau_0)}{\tau_0^2}\tau_k^2,$$

for some positive constant γ.

Remark 7.4.2. *The estimates (7.4.7) show that the sequence $\{\tau_k\}$ converges to zero with the convergence rate $O(\frac{1}{k})$. Consequently, by (7.4.8), we see that the sequence $\{\beta_1^k\beta_2^k\}$ also converges to zero with the convergence rate $O(\frac{1}{k^2})$. From the condition of Lemma 7.3.1 and (7.4.5), we can derive the initial value $\tau_0 := \frac{\sqrt{5}-1}{2}$.*

The algorithm and its convergence

Finally, by combining the conclusions of Lemma 7.3.1, Theorem 7.4.1 and Lemma 7.4.1, we present the algorithm in detail as follows.

Algorithm 7.4.1.(*Decomposition algorithm with two dual steps*).
Initialization: Perform the following steps:

1. Choose $\tau_0 := 0.5(\sqrt{5}-1)$ and $\beta_1^0 > 0$. Set $\beta_2^0 = \frac{\bar{L}^2}{\beta_1^0}$.
2. Compute $\bar{x}^0$ and $\bar{y}^0$ from (7.3.2) or (7.3.3).

Iteration: For $k = 0, 1, \cdots$, perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. Compute $\bar{x}^{k+1}$ *in parallel* and $\bar{y}^{k+1}$ by using scheme (7.4.1):
$$(\bar{x}^{k+1}, \bar{y}^{k+1}) := \mathcal{S}_{2\text{ds}}(\bar{x}^k, \bar{y}^k; \beta_1^k, \beta_2, \tau_k).$$
3. Compute $\alpha_k := \frac{p_X(x^*(\hat{y};\beta_1))}{D_X}$.
4. Update $\beta_1^{k+1} := (1-\alpha_k\tau_k)\beta_1^k$ and $\beta_2^{k+1} := (1-\tau_k)\beta_2^k$.
5. Update the step size τ_k as:
$$\tau_{k+1} := 0.5\tau_k \left\{ \left[(1-\alpha_k\tau_k)^2\tau_k^2 + 4(1-\alpha_k\tau_k)\right]^{1/2} - (1-\alpha_k\tau_k)\tau_k \right\}.$$

End.

Note that the second step of $S_{2\text{ds}}$ at *Step 3* of Algorithm 7.4.1 can be parallelized. This step computes $x^*(\bar{y}^k;\beta_1)$ by solving M convex primal subproblems in *parallel*. The stopping criterion of Algorithm 7.4.1 at *Step 1* will be discussed in Section 7.8.

The following theorem shows the convergence of Algorithm 7.4.1.

Theorem 7.4.3. *Suppose that Assumptions* ***A.6.1.7***, ***A.7.1.8*** *and* ***A.7.4.9*** *are satisfied. Let* $\{(\bar{x}^k, \bar{y}^k)\}$ *be a sequence generated by Algorithm 7.4.1 after* k *iterations. Then the following duality gap holds:*

$$-R_{Y^*}\mathcal{F}(\bar{x}^{k+1}) \le \phi(\bar{x}^{k+1}) - g(\bar{y}^{k+1}) \le \frac{\beta_1^0 D_X}{[0.5(\sqrt{5}-1)k+1]^{\alpha^*}}, \tag{7.4.9}$$

and the feasibility gap satisfies:

$$\mathcal{F}(\bar{x}^{k+1}) = \left\|A\bar{x}^{k+1} - b\right\| \leq \frac{C_f}{0.25(\sqrt{5}-1)(1+\alpha^*)k+1}, \tag{7.4.10}$$

where $C_f := (3-\sqrt{5})\frac{\bar{L}^2}{\beta_1^0}R_{Y^*} + 0.5(\sqrt{5}-1)\bar{L}\sqrt{2D_X}$ *and* R_{Y^*} *is defined by* (7.2.10). *Consequently, the sequence* $\{(\bar{x}^k, \bar{y}^k)\}_{k\geq 0}$ *generated by Algorithm 7.4.1 converges to a solution* (x^*, y^*) *of the primal and dual problems* (SepCOP)-(7.1.1) *as* $k \to \infty$.

Proof. From Lemma 7.2.1, we can obtain the following estimates: $\mathcal{F}(\bar{x}^{k+1}) \leq 2\beta_2^{k+1}R_{Y^*} + \sqrt{2\beta_1^{k+1}\beta_2^{k+1}D_X}$ and $\phi(\bar{x}^{k+1}) - g(\bar{y}^{k+1}) \leq \beta_1^{k+1}D_X$. By combining these inequalities and (7.4.8) and then using the definition of C_f we obtain (7.4.9) and (7.4.10). □

Now, we consider a particular case, where we can get $O(1/\varepsilon_f)$ of the worst-case complexity, where ε_f is a desired accuracy.

Corollary 7.4.1. *Suppose that the smoothness parameter* β_1^k *in Algorithm 7.4.1 is fixed at* $\beta_1^k = \beta_1^0 = \bar{L}\varepsilon_f$ *for all* $k \geq 0$. *Suppose further that the sequence* $\{\tau_k\}$ *is updated by* $\tau_{k+1} := 0.5\tau_k\left(\sqrt{\tau^2+4} - \tau\right)$ *starting from* $\tau_0 := 0.5(\sqrt{5}-1)$. *Then after* $\bar{k} := \lfloor 2/\varepsilon_f \rfloor + 1$ *iterations, one has:*

$$\mathcal{F}(\bar{x}^{\bar{k}}) \leq C_f^0\varepsilon_f \quad \text{and} \quad \left|\phi(\bar{x}^{\bar{k}}) - g(\bar{y}^{\bar{k}})\right| \leq C_d^0\varepsilon_f, \tag{7.4.11}$$

where $C_f^0 := \bar{L}(2R_{Y^*} + \sqrt{2D_X})$ *and* $C_d^0 := \bar{L}\max\left\{2R_{Y^*} + \sqrt{2D_X}, D_X\right\}$.

Proof. If we assume that β_1^k is fixed in Algorithm 7.4.1 then, by the new update rule of $\{\tau_k\}$ we have $\beta_2^{k+1}\beta_1^0 = \bar{L}^2\tau_k^2 \leq \frac{4\bar{L}^2\tau_0^2}{(\tau_0 k+2)^2}$ due to (7.4.7) and (7.4.8) with $\alpha^* = 0$. Since $\beta_1^0 = \bar{L}\varepsilon_f$, if we choose $\bar{k} := \lfloor 2/\varepsilon_f \rfloor + 1$ then $\frac{2\tau_0}{\tau_0(\bar{k}-1)+2} \leq \varepsilon_f$. Furthermore, by Lemma 7.2.1 we have $\mathcal{F}(\bar{x}^{\bar{k}}) \leq 2\beta_2^{\bar{k}}R_{Y^*} + \sqrt{2\beta_1^0\beta_2^{\bar{k}}D_X} \leq \bar{L}(2R_{Y^*} + \sqrt{2D_X})\varepsilon_f$ and $-R_{Y^*}\mathcal{F}(\bar{x}^{\bar{k}}) \leq \phi(\bar{x}^{\bar{k}}) - g(\bar{y}^{\bar{k}}) \leq \beta_1^0 D_X = \bar{L}D_X\varepsilon_f$. By combining these estimates, we obtain the conclusion (7.4.11). □

Remark 7.4.4. *From Corollary 7.4.1 we also obtain:*

$$O\left(\left[M\max_{1\leq i\leq M}\left\{\sigma_{X_i}^{-1}\left\|A_i\right\|^2\right\}\right]^{1/2}\hat{R}_0\varepsilon_f^{-1}\right),$$

where $\hat{R}_0 := \max\left\{2R_{Y^*} + \sqrt{2D_X}, D_X\right\}$. *This formula again shows that the worst-case complexity of Algorithm 7.3.1 also depends on the size of the problem. In particular, it depends on* M, *the number of components.*

Remark 7.4.5. *The constant $\bar{L}$ in Algorithm 7.4.1 can be replaced by $\hat{\bar{L}} := \left[\sum_{i=1}^{M} \frac{\|A_i\|^2}{\sigma_{X_i}}\right]^{1/2}$ as suggested by the condition (A.1.15) in Appendix A.*

7.5 Decomposition algorithms with switching steps

In this section, we apply a switching strategy in [140] to derive two variants of Algorithms 7.3.1 and 7.4.1. These algorithms alternately switch between the primal step scheme $\mathcal{S}_{2\text{ps}}$ and the dual step scheme $\mathcal{S}_{2\text{ds}}$ depending on the iteration counter k being either even or odd. In the first variant, we simultaneously update the smoothness parameters β_1 and β_2, while, in the second variant, these parameters are alternatively updated.

The first variant

In the first variant, we simply switch between two schemes $\mathcal{S}_{2\text{ps}}$ and $\mathcal{S}_{2\text{ds}}$ to obtain a switching variant. In principle, we can either start with $\mathcal{S}_{2\text{ps}}$ then switch to $\mathcal{S}_{2\text{ds}}$ and repeat in the next iterations. We notice that this variant fills in the disadvantages of Algorithms 7.3.1 and 7.4.1 as we will see later. For simplicity of presentation, we make the following rule.

Rule R.7.5.1. *If the iteration counter k is even then we apply $\mathcal{S}_{2ps}$. Otherwise, $\mathcal{S}_{2ds}$ is used.*

By combining the conclusions of Theorems 7.3.2 and 7.4.1 we can see that the sequence $\left\{(\bar{x}^k, \bar{y}^k)\right\}_{k\geq 0}$ generated either by the scheme $\mathcal{S}_{2\text{ps}}$ or $\mathcal{S}_{2\text{ds}}$ satisfies the excessive condition (7.2.6).

Now, we can present the algorithm in detail as follows.

Algorithm 7.5.1.(*Decomposition algorithm I with switching primal-dual steps*).
Initialization: Perform as in Algorithm 7.4.1 with $\tau_0 := 0.5$.
Iteration: For $k = 0, 1, \cdots$ perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. If k is even then perform the scheme $\mathcal{S}_{2\text{ps}}$:
   2.1. Update $\beta_2^{k+1} := (1 - \tau_k)\beta_2^k$.
   2.2. Compute $(\bar{x}^{k+1}, \bar{y}^{k+1}) := S_{2\text{ps}}(\bar{x}^k, \bar{y}^k, \beta_1^k, \beta_2^{k+1}, \tau_k)$.
   2.3. Update $\beta_1^{k+1} := (1 - \tau_k)\beta_1^k$.

2.4. Update the step size τ_k as $\tau_{k+1} := \frac{\tau_k}{\tau_k+1}$.

3. Otherwise, (i.e. k is odd) perform the scheme $\mathcal{S}_{2\text{ds}}$:

3.1. Compute $(\bar{x}^{k+1}, \bar{y}^{k+1}) := S_{2\text{ds}}(\bar{x}^k, \bar{y}^k, \beta_1^k, \beta_2^k, \tau_k)$.

3.2. Compute the factor $\alpha_k := p_X(x^*(\hat{y}^k; \beta_1^k))/D_X$.

3.3. Update $\beta_1^{k+1} := (1 - \alpha_k\tau_k)\beta_1^k$ and $\beta_2^{k+1} := (1 - \tau_k)\beta_2^k$.

3.4. Update the step size τ_k as:

$$\tau_{k+1} := \frac{\tau_k}{2}\left\{\left[(1 - \alpha_k\tau_k)^2\tau_k^2 + 4(1 - \alpha_k\tau_k)\right]^{1/2} - (1 - \alpha_k\tau_k)\tau_k\right\}.$$

End.

As it has been shown in Algorithms 7.3.1 and 7.4.1 that the second line of the dual scheme $S_{2\text{ds}}$ can be parallelized by solving M convex subproblems simultaneously. In the primal scheme $S_{2\text{ps}}$, there are two steps that can be parallelized: the first and the third lines of $S_{2\text{ps}}$. Each line requires one to solve M convex subproblems in parallel.

Similar to the proof of Lemma 7.3.2 we can show that the sequence $\{\tau_k\}_{k\geq 0}$ generated by Step 2.4 or Step 3.4 of Algorithm 7.5.1 remains satisfying the estimates (7.4.7). Consequently, the estimate of β_2^k in (7.4.8) is still valid, while the parameter β_1^k satisfies $\beta_1^{k+1} \leq \frac{\beta_1^0}{(\tau_0 k+1)^{(1+\alpha^*)/2}}$.

Finally, we summarize the convergence results of Algorithm 7.5.1 in the following theorem.

Theorem 7.5.1. *Suppose that Assumptions* ***A.6.1.7****,* ***A.7.1.8*** *and* ***A.7.4.9*** *are satisfied. Let* $\{(\bar{x}^k, \bar{y}^k)\}$ *be a sequence generated by Algorithm 7.5.1 after* k *iterations. Then the following duality gap holds:*

$$-R_{Y^*}F(\bar{x}^{k+1}) \leq \phi(\bar{x}^{k+1}) - g(\bar{y}^{k+1}) \leq \frac{\beta_1^0 D_X}{(0.5k+1)^{(1+\alpha^*)/2}}, \tag{7.5.1}$$

and the feasibility gap satisfies:

$$\mathcal{F}(\bar{x}^{k+1}) = \left\|A\bar{x}^{k+1} - b\right\| \leq \frac{C_f}{0.25(1+\alpha^*)k+1}, \tag{7.5.2}$$

where $C_f := \frac{\bar{L}^2}{\beta_1^0}R_{Y^*} + 0.5\bar{L}\sqrt{2D_X}$ *and* R_{Y^*} *is defined by* (7.2.10)*. Consequently, the sequence* $\{(\bar{x}^k, \bar{y}^k)\}_{k\geq 0}$ *generated by Algorithm 7.5.1 converges to a solution* (x^*, y^*) *of the primal and dual problems* (SepCOP)-(7.1.1) *as* $k \to \infty$.

The proof of this theorem is similar to the proof of Theorem 7.4.3, we omit the details here. We can see from the right hand side of (7.5.1) in Theorem 7.5.1

that this term is better than the one in Theorem 7.4.3. Consequently, Algorithm 7.5.1 overcomes the difficulty of Algorithm 7.4.1 in Assumption **A.7.4.9** when α^* is small and ensures that the sequence $\{\beta_1^k\}$ decreases to zero. Nevertheless, as a compensation, at each even iteration, the scheme $S_{2\text{ps}}$ is performed, it requires additional cost to compute $\bar{x}^+$ at the third line of $S_{2\text{ps}}$.

The second variant

In this variant, we only update one smoothness parameter β_1 or β_2 at each iteration as done in [140]. By applying Rule 7.5.1 we first update β_1 if the iteration counter k is even. Otherwise, β_2 is updated. As we will see in Theorem 7.5.3 bellow, this variant has a better convergence rate than the one of the first variant.

The main iteration is now presented as follows:

$$(\bar{x}^+, \bar{y}^+) := \begin{cases} \mathcal{S}_{2\text{ps}}(\bar{x}, \bar{y}, \beta_1, \beta_2, \tau) \text{ and } \beta_1^+ := (1-\tau)\beta_1 & \text{if } k \text{ is even,} \\ \mathcal{S}_{2\text{ds}}(\bar{x}, \bar{y}, \beta_1, \beta_2, \tau) \text{ and } \beta_2^+ := (1-\tau)\beta_2 & \text{otherwise.} \end{cases} \tag{7.5.3}$$

Note that in this scheme, only one parameter is updated at each iteration which is different from (7.3.4).

The following lemma shows that $(\bar{x}^+, \bar{y}^+)$ generated by the scheme (7.5.3) maintains the excessive gap condition (7.2.6).

Lemma 7.5.1. *Suppose that $(\bar{x}, \bar{y}) \in X \times \mathbf{R}^m$ and satisfies (7.2.6) w.r.t. two values β_1 and β_2. Then if we choose the parameter $\tau \in (0,1)$ such that:*

$$\beta_1\beta_2 \geq \frac{\tau^2}{1-\tau}\bar{L}^2, \tag{7.5.4}$$

then the new point $(\bar{x}^+, \bar{y}^+)$ generated by the scheme (7.5.3) is in $X \times \mathbf{R}^m$ and maintains the excessive gap condition (7.2.6) w.r.t. either two new values $\beta_1^+ < \beta_1$ and β_2 or β_1 and $\beta_2^+ < \beta_2$.

The proof of this lemma is quite similar to [140, Theorem 4.2.] that we omit here.

Now, we show how to update the step size τ in order to maintain the excessive gap condition (7.2.6). Similar to (7.4.6), we update τ_+ such that:

$$0 < \tau_+ \leq \frac{\tau}{2}\left(\sqrt{\tau^2 + 4} - \tau\right) < \tau.$$

The tightest rule for updating the step size sequence $\{\tau_k\}_{k\geq 0}$ is:

$$\tau_{k+1} := \frac{\tau_k}{2}\left(\sqrt{\tau_k^2 + 4} - \tau_k\right), \tag{7.5.5}$$

for all $k \geq 0$ and $\tau_0 \in (0, \frac{\sqrt{5}-1}{2}]$. Associated with $\{\tau_k\}$, we generate two sequences $\{\beta_1^k\}$ and $\{\beta_2^k\}$ as:

$$\beta_1^{k+1} := \begin{cases} (1-\tau_k)\beta_1^k & \text{if } k \text{ is even} \\ \beta_1^k & \text{otherwise,} \end{cases} \quad \text{and} \quad \beta_2^{k+1} := \begin{cases} \beta_2^k & \text{if } k \text{ is even} \\ (1-\tau_k)\beta_2^k & \text{otherwise,} \end{cases} \tag{7.5.6}$$

where $\beta_1^0 = \beta_2^0 > 0$ are fixed.

Similar to Lemma 7.3.2, we have the following lemma.

Lemma 7.5.2. *Let $\{\tau_k\}$, $\{\beta_1^k\}$ and $\{\beta_2^k\}$ be three sequences generated by (7.5.5) and (7.5.6), respectively. Then:*

$$\frac{(1-\tau_0)\beta_1^0}{\tau_0 k + 1} < \beta_1^k < \frac{2\beta_1^0\sqrt{1-\tau_0}}{\tau_0 k}, \quad \text{and} \quad \frac{\beta_2^0\sqrt{1-\tau_0}}{\tau_0 k + 1} < \beta_2^k < \frac{2\beta_2^0}{\tau_0 k}, \tag{7.5.7}$$

for all $k \geq 1$, provided that $\beta_1^0 = \beta_2^0 > 0$.

The proof of this lemma can be found in Appendix A.1.

Remark 7.5.2. *We can see that the right-hand side $\eta_k^1(\tau_0) := \frac{2\beta_1^0\sqrt{1-\tau_0}}{\tau_0 k}$ and $\eta_k^2(\tau_0) := \frac{2\beta_2^0}{\tau_0 k}$ of (7.5.7) are decreasing in $(0,1)$ for $k \geq 1$. Therefore, we can choose τ_0 as large as possible to minimize $\eta_k(\cdot)$ in $(0,1)$. In this case, we choose $\tau_0 := \frac{\sqrt{5}-1}{2} \approx 0.618$.*

Now, we can present the algorithm in detail as follows:

Algorithm 7.5.2.(*Decomposition algorithm II with switching primal-dual steps*).
Initialization: Perform the following steps:

1. Choose $\tau_0 := 0.5(\sqrt{5}-1)$ and set $\beta_1^0 = \beta_2^0 := \bar{L}$.
2. Compute $\bar{x}^0$ and $\bar{y}^0$ as in Algorithm 7.3.1.

Iteration: For $k = 0, 1, \cdots$, perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. If k is even then:
   2a) Compute $(\bar{x}^{k+1}, \bar{y}^{k+1}) := \mathcal{S}_{2\text{ps}}(\bar{x}^k, \bar{y}^k; \beta_1^k, \beta_2^k, \tau_k)$.
   2b) Update the smoothness parameter β_1^k as $\beta_1^{k+1} := (1-\tau_k)\beta_1^k$.
3. Otherwise, i.e. if k is odd then:

3a) Compute $(\bar{x}^{k+1}, \bar{y}^{k+1}) := \mathcal{S}_{2\text{ds}}(\bar{x}^k, \bar{y}^k; \beta_1^k, \beta_2^k, \tau_k)$.

3b) Update the smoothness parameter β_2^k as $\beta_2^{k+1} := (1-\tau_k)\beta_2^k$.

4. Update the step size τ_k as: $\tau_{k+1} := \frac{\tau_k}{2}\left[(\tau_k^2+4)^{1/2} - \tau_k\right]$.

End.

The main steps of Algorithm 7.5.2 are Steps 2a and 2b, which requires us to compute either the primal step scheme $\mathcal{S}_{2\text{ps}}$ or the dual step scheme $\mathcal{S}_{2\text{ds}}$. The following theorem shows the convergence of this algorithm.

Theorem 7.5.3. *Let $\left\{(\bar{x}^k, \bar{y}^k)\right\}_{k\geq 0}$ be a sequence generated by Algorithm 7.5.2. Then the duality gap is satisfied:*

$$-R_{Y^*}\left\|A\bar{x}^{k+1} - b\right\| \leq \phi(\bar{x}^{k+1}) - g(\bar{y}^{k+1}) \leq \frac{2\bar{L}D_X}{k+1}, \tag{7.5.8}$$

and the feasibility gap holds:

$$\left\|A\bar{x}^{k+1} - b\right\| \leq \frac{(\sqrt{5}+1)\bar{L}D_{Y^*}}{4(k+1)}, \tag{7.5.9}$$

where $\bar{L}$, D_X, R_{Y^*} and D_{Y^*} are defined in (7.3.1), (7.1.5) and (7.2.10).

Proof. The conclusion of this theorem follows directly from Lemmas 7.2.1 and 7.3.2, the conditions $\tau_0 = \frac{\sqrt{5}-1}{2}$, $\beta_1^0 = \beta_2^0 = \bar{L}$ and the fact that $\beta_1^k \leq \beta_2^k$. □

Remark 7.5.4. *Note that the worst-case complexity of Algorithm 7.5.2 is still $O(\frac{\bar{L}R_0}{\varepsilon})$, where $R_0 := \max\{D_X, D_Y^*\}$. The constants in the complexity bounds (7.4.9) and (7.4.10) are similar to the ones in (7.5.1) and (7.5.2), respectively. As we discuss in Section 7.8, the rate of decrease of τ_k in Algorithm 7.5.2 is smaller than two times of τ_k in Algorithm 7.5.2. Consequently, the sequences $\{\beta_1^k\}$ and $\{\beta_2^k\}$ generated by Algorithm 7.3.1 approach zero faster than the ones generated by Algorithm 7.5.2. Hence, Algorithm 7.3.1 converges faster than Algorithm 7.5.2. By substituting $\bar{L}$ into the worst-case complexity formula, we obtain:*

$$O\left(\left[M \max_{1\leq i\leq M}\left\{\sigma_{X_i}^{-1}\left\|A_i\right\|^2\right\}\right]^{1/2} R_0\varepsilon^{-1}\right),$$

which is the same as the one in Remark 7.3.5.

Note that we can switch the role of the schemes $\mathcal{S}_{2\text{ps}}$ and $\mathcal{S}_{2\text{ds}}$ in Algorithm 7.5.2. We can also combine the primal and dual step schemes $\mathcal{S}_{2\text{ps}}$ and $\mathcal{S}_{2\text{ds}}$ in different ways to obtain other variants. For instance, we can apply twice dual scheme $\mathcal{S}_{2\text{ds}}$ and one primal scheme $\mathcal{S}_{2\text{ps}}$ and then switch them.

7.6 Application to strongly convex case

If ϕ_i in (SepCOP) is strongly convex for $i = 1, \cdots, M$ then the convergence rate of the dual scheme (7.4.1) can be accelerated up to $O(\frac{1}{k^2})$.

Suppose that ϕ_i is strongly convex with a convexity parameter $\sigma_{\phi_i} > 0$ for $i = 1, \cdots, M$. Then the original dual function g defined by (6.3.1) is well-defined, concave and differentiable. Moreover, its gradient is given by:

$$\nabla g(y) = Ax^*(y) - b, \tag{7.6.1}$$

which is Lipschitz continuous with a Lipschitz constant $L^g := \sum_{i=1}^{M} \frac{\|A_i\|_2^2}{\sigma_{\phi_i}}$, see [134, 145]. The excessive gap condition (7.2.6) in this case reduces to:

$$f(\bar{x}; \beta_2) \leq g(\bar{y}), \tag{7.6.2}$$

for given $\bar{x} \in X$, $\bar{y} \in \mathbf{R}^m$ and $\beta_2 > 0$. From Lemma 7.2.1 we conclude that if the point $(\bar{x}, \bar{y}) \in X \times \mathbf{R}^m$ and satisfies (7.6.2) then, for a given $y^* \in Y^*$, the following estimates hold:

$$-2\beta_2 \|y^*\|^2 \leq -\|y^*\| \|A\bar{x} - b\| \leq \phi(\bar{x}) - g(\bar{y}) \leq 0, \tag{7.6.3}$$

and

$$\mathcal{F}(\bar{x}) := \|A\bar{x} - b\| \leq 2\beta_2 \|y^*\|. \tag{7.6.4}$$

We now adapt the scheme (7.4.1) to this special case. Suppose $(\bar{x}, \bar{y}) \in X \times \mathbf{R}^m$ and satisfies (7.6.2), we generate a new pair $(\bar{x}^+, \bar{y}^+)$ as:

$$(\bar{x}^+, \bar{y}^+) := \mathcal{S}^s_{2\text{ds}}(\bar{x}, \bar{y}; \beta_2, \tau) \Longleftrightarrow \left\{ \begin{array}{ll} \hat{y} := (1-\tau)\bar{y} + \tau y^*(\bar{x}; \beta_2), \\ \bar{x}^+ := (1-\tau)\bar{x} + \tau x^*(\hat{y}), \\ \bar{y}^+ = \hat{y} + (L^g)^{-1}(Ax^*(\hat{y}) - b), \end{array} \right. \tag{7.6.5}$$

where $y^*(\bar{x}; \beta_2) = \frac{1}{\beta_2}(A\bar{x} - b)$, and $x^*(y) := (x_1^*(y), \ldots, x_M^*(y))$ is the solution of the minimization problems in (6.3.1). The parameter β_2 is updated by $\beta_2^+ := (1-\tau)\beta_2$ and $\tau \in (0, 1)$ will appropriately be chosen.

The following lemma shows that $(\bar{x}^+, \bar{y}^+)$ generated by (7.6.5) satisfies (7.6.2) whose proof can be found in [140].

Lemma 7.6.1. *Suppose that the point $(\bar{x}, \bar{y}) \in X \times \mathbf{R}^m$ and satisfies the excessive gap condition (7.6.2) with the value β_2. Then if the parameter τ is chosen such that $\tau \in (0, 1)$ and:*

$$\beta_2 \geq \frac{\tau^2 L^g}{1-\tau}. \tag{7.6.6}$$

then the new point $(\bar{x}^+, \bar{y}^+)$ computed by (7.6.5) is in $X \times \mathbf{R}^m$ and also satisfies (7.6.2) with a new parameter value $\beta_2^+ < \beta_2$.

Now, let us derive the rule to update the parameter τ. Suppose that β_2 satisfies (7.6.6). Since $\beta_2^+ = (1-\tau)\beta_2$, the condition (7.6.6) holds for β_2^+ if $\tau^2 \geq \frac{\tau_+^2}{1-\tau_+}$. Therefore, similar to Algorithm 7.5.2, we update the parameter τ by using the rule (7.5.5).

Before presenting the algorithm, it is necessary to find a starting point $(\bar{x}^0, \bar{y}^0)$ satisfying (7.6.2). Let $\beta_2 = L^g$. We compute $(\bar{x}^0, \bar{y}^0)$ as:

$$\bar{x}^0 := x^*(0^m) \quad \text{and} \quad \bar{y}^0 := (L^g)^{-1}(A\bar{x}^0 - b). \tag{7.6.7}$$

It follows from [140, Lemma 7.4.] that $(\bar{x}^0, \bar{y}^0)$ satisfies the excessive gap condition (7.6.2).

Finally, the decomposition algorithm for solving the strongly convex programming problem of the form (SepCOP) is described in detail as follows.

Algorithm 7.6.1.(*Decomposition algorithm for strongly convex case*).
Initialization: Perform the following steps:

1. Choose $\tau_0 := 0.5(\sqrt{5}-1)$. Set $\beta_2^0 := L^g$.
2. Compute $\bar{x}^0$ and $\bar{y}^0$ as:

$$\bar{x}^0 := x^*(0^m) \text{ and } \bar{y}^0 := (L^g)^{-1}(A\bar{x}^0 - b).$$

Iteration: For $k = 0, 1, \cdots$, perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. Compute $(\bar{x}^{k+1}, \bar{y}^{k+1}) := \mathcal{S}^s_{2\text{ds}}(\bar{x}^k, \bar{y}^k; \beta_2^k, \tau_k)$.
3. Update the smoothness parameter as: $\beta_2^{k+1} := (1-\tau_k)\beta_2^k$.
4. Update the step size τ_k as: $\tau_{k+1} := \frac{\tau_k}{2}\left[(\tau_k^2+4)^{1/2} - \tau_k\right]$.

End.

The convergence of Algorithm 7.6.1 is stated as in Theorem 7.6.1 below.

Theorem 7.6.1. *Let $\{(\bar{x}^k, \bar{y}^k)\}_{k\geq 0}$ be a sequence generated by Algorithm 7.6.1. Then the following duality and feasibility gaps are satisfied:*

$$-\frac{4L^g R_{Y^*}^2}{(k+2)^2} \leq \phi(\bar{x}^k) - g(\bar{y}^k) \leq 0 \quad \textit{and} \quad \left\|A\bar{x}^k - b\right\| \leq \frac{4L^g R_{Y^*}}{(k+2)^2}, \tag{7.6.8}$$

where $L^g := \sum_{i=1}^M \frac{\|A_i\|_2^2}{\sigma_{\phi_i}}$ and R_{Y^*} is defined in (7.2.10).

Proof. From the update rule of τ^k, we have $(1-\tau_{k+1}) = \frac{\tau_{k+1}^2}{\tau_k^2}$. Moreover, since $\beta_2^{k+1} = (1-\tau_k)\beta_2^k$, it implies that $\beta_2^{k+1} = \beta_2^0 \prod_{i=0}^k (1-\tau_i) = \frac{\beta_2^0(1-\tau_0)}{\tau_0^2}\tau_k^2$. By using the inequalities (7.4.7) wit $\alpha^* = 1$ and $\beta_2^0 = L^g$, we have $\beta_2^{k+1} < \frac{4L^g(1-\tau_0)}{(\tau_0 k+2)^2}$. With $\tau_0 = 0.5(\sqrt{5}-1)$, one has $\beta_2^k < \frac{4L^g}{(k+2)^2}$. By substituting this inequality into (7.6.3) and (7.6.4), we obtain (7.6.8). □

Theorem 7.6.1 shows that the worst-case complexity of Algorithm 7.6.1 is $O(\frac{2\sqrt{L^g}R_{Y^*}}{\sqrt{\varepsilon}})$. Moreover, at each iteration of this algorithm, only one primal step is performed *in parallel*. Note that the constant L^g defined in Theorem 7.6.1 is similar to the constant $\hat{\hat{L}}$ defined in Remark 7.4.5. We can write the worst-case complexity of Algorithm 7.6.1 as:

$$O\left(\left[\sum_{i=1}^{M} \sigma_{\phi_i}^{-1} \|A_i\|^2\right]^{1/2} R_{Y^*} \varepsilon^{-1/2}\right),$$

which depends on the dimension of the problem.

7.7 Extensions to inexact case

As we mentioned earlier, solving the convex primal subproblems (7.1.8) exactly is only *conceptual*. In practice, we can only solve these problems up to a certain accuracy as in Definition 7.2.1. In this section, we only extend Algorithms 7.3.1 and 7.4.1 to the inexact case. Extensions of the remaining algorithms to the inexact case can be done similarly.

For a given accuracy vector $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_M)$ in Definition 7.2.1. Let us first define the following quantities:

$$\begin{cases} \varepsilon_{[\sigma]} & := \left[\sum_{i=1}^M \sigma_{X_i} \varepsilon_i^2\right]^{1/2}, \\ D_\sigma & := \left[2\sum_{i=1}^M \frac{D_{X_i}}{\sigma_{X_i}}\right]^{1/2}, \\ C_d & := \|A\|_2^2 D_\sigma + \left\|A^T(Ax^c - b)\right\|_2, \end{cases} \tag{7.7.1}$$

From (7.7.1) we see that the quantity C_d depends on the data of the problem, i.e. matrix A, the quantities D_{X_i}, σ_{X_i} for $i = 1, \cdots, M$ and vectors b and x^c. Moreover, $\varepsilon_{[1]} = \|\varepsilon\|_2$. If we choose the accuracy level $\varepsilon_i = \hat{\varepsilon} \geq 0$ for all $i = 1, \cdots, M$ then the quantities $\varepsilon_{[1]} = M\hat{\varepsilon}$ and $\varepsilon_{[\sigma]} = [\sum_{i=1}^M \sigma_{X_i}]^{1/2}\hat{\varepsilon}$.

Next, if we assume that the convex primal subproblems (7.1.8) and (7.2.13) are solved inexactly in the sense of Definition 7.2.1 to obtain approximate solutions

$\tilde{x}^*(\cdot;\beta_1)$ and $\widetilde{\mathcal{P}}(\cdot;\beta_2)$, respectively then the algorithmic schemes $\mathcal{S}_{2\text{ps}}$ and $\mathcal{S}_{2\text{ds}}$ can be modified as follows:

$$(\bar{x}^+,\bar{y}^+):=\widetilde{\mathcal{S}}_{2\text{ps}}(\bar{x},\bar{y},\beta_1,\beta_2^+,\tau)\Longleftrightarrow\begin{cases}\hat{x}:=(1-\tau)\bar{x}+\tau\tilde{x}^*(\bar{y};\beta_1),\\ \bar{y}^+:=(1-\tau)\bar{y}+\tau y^*(\hat{x};\beta_2^+),\\ \bar{x}^+:=\widetilde{\mathcal{P}}(\hat{x};\beta_2^+),\end{cases}\tag{7.7.2}$$

and

$$(\bar{x}^+,\bar{y}^+):=\widetilde{\mathcal{S}}_{2\text{ds}}(\bar{x},\bar{y},\beta_1,\beta_2,\tau)\Longleftrightarrow\begin{cases}\hat{y}:=(1-\tau)\bar{y}+\tau y^*(\bar{x};\beta_2)\\ \bar{x}^+:=(1-\tau)\bar{x}+\tau\tilde{x}^*(\hat{y};\beta_1)\\ \bar{y}^+:=\widetilde{\mathcal{G}}^*(\hat{y};\beta_1).\end{cases}\tag{7.7.3}$$

The smoothness parameter β_1 and β_2 in both schemes are updated as in $\mathcal{S}_{2\text{ps}}$ and $\mathcal{S}_{2\text{ds}}$, respectively, while the damping factor α in (7.4.2) is updated by $\alpha:=\frac{p_X(\tilde{x}^*(\hat{y};\beta_1))}{D_X}$.

We also need to find an initial point in the inexact case. This can be done by performing one of the following schemes:

$$\text{a)}\begin{cases}\bar{y}^0&:=\beta_2^{-1}(A\bar{x}^c-b),\\ \bar{x}^0&:=\widetilde{\mathcal{P}}(x^c;\beta_2),\end{cases}\quad\text{or}\quad\text{b)}\begin{cases}\bar{x}^0&:=\tilde{x}^*(0^m;\beta_1),\\ \bar{y}^0&:=L^g(\beta_1)^{-1}(A\bar{x}^0-b),\end{cases}\tag{7.7.4}$$

where $\tilde{x}^*(y^c;\beta_1)$ defined by (7.2.1). Similar to the conclusion of Lemma 7.3.1 the point $(\bar{x}^0,\bar{y}^0)$ defined by (7.7.4) satisfies the δ_0 - excessive gap condition (7.2.7) under an appropriate choice of β_1, where:

$$\delta_0:=\begin{cases}0.5\beta_2^{-1}M\sum_{i=1}^M\|A_i\|^2\varepsilon_i^2 & \text{if } (\bar{x}^0,\bar{y}^0)\text{ is defined by a),}\\ \beta_1\left(\bar{L}^{-1}C_d\varepsilon_{[1]}+\frac{1}{2}\varepsilon_{[\sigma]}^2\right)\geq 0 & \text{if } (\bar{x}^0,\bar{y}^0)\text{ is defined by b).}\end{cases}\tag{7.7.5}$$

Lemma 7.7.1. *The point $(\bar{x}^0,\bar{y}^0)\in X\times\mathbb{R}^m$ generated by either scheme* a) *or scheme* b) *in* (7.7.4) *satisfies the δ_0-excessive gap condition* (7.2.7) *w.r.t. β_1 and β_2 provided that:*

$$\beta_1\beta_2\geq\bar{L}^2,\tag{7.7.6}$$

where δ_0 is defined by (7.7.5).

The proof of this lemma can be found in Appendix A.1.

Now, we consider the following functions:

$$\begin{cases}\eta_1(\tau,\beta_1,\beta_2,\bar{y},\varepsilon)&:=2\beta_1(1-\tau)D_\sigma\varepsilon_{[\sigma]}+0.5\sum_{i=1}^M L_i^\psi((1-\tau)\beta_2)\varepsilon_i^2,\\ \eta_2(\tau,\beta_1,\beta_2,\bar{y},\varepsilon)&:=\left[\bar{L}^{-1}\beta_1C_d+(1-\tau)\tau\left(\beta_2^{-1}C_d+\|A\|\,\|\bar{y}\|\right)\right]\varepsilon_{[1]}\\ &\quad+0.5\tau\beta_1\varepsilon_{[\sigma]}^2.\end{cases}\tag{7.7.7}$$

The following theorem shows that the new point $(\bar{x}^+, \bar{y}^+)$ generated by the schemes (7.7.2) and (7.7.3) still maintains the δ_+-excessive gap condition (7.2.7).

Theorem 7.7.1. *Suppose that Assumptions **A.6.1.7**, **A.7.1.8** and **A.7.4.9** are satisfied. Let $(\bar{x}, \bar{y}) \in X \times \mathbf{R}^m$ be a point satisfying (7.2.7) w.r.t. two values $\beta_1 > 0$ and $\beta_2 > 0$ and the accuracy $\delta \geq 0$. Then if the parameter τ is chosen such that $\tau \in (0, 1)$ and:*

$$\beta_1\beta_2 \geq \frac{\tau^2}{(1-\tau)^2}\bar{L}^2 \quad (\textit{resp. } \beta_1\beta_2 \geq \frac{\tau^2}{1-\tau}\bar{L}^2), \tag{7.7.8}$$

then the new point $(\bar{x}^+, \bar{y}^+)$ generated by scheme (7.7.2)-(7.3.5) (resp. (7.7.3)-(7.4.2)) is in $X \times \mathbf{R}^m$ and maintains the δ_+-excessive gap condition (7.2.7) w.r.t. two new values β_1^+ and β_2^+ and $\delta_+ := (1-\tau)\delta + \eta_1(\tau, \beta_1, \beta_2, \bar{y}, \varepsilon)$ (resp. $\delta_+ := (1-\tau)\delta + \eta_2(\tau, \beta_1, \beta_2, \bar{y}, \varepsilon)$).

The proof of this theorem is postponed to Appendix A.1.

Now, let $\{\eta_k\}_{k\geq 0}$ be a sequence generated by either $\eta_k := \eta_1(\tau^k, \beta_1^k, \beta_2^k, \bar{y}^k, \hat{\varepsilon}_k)$ or $\eta_k := \eta_2(\tau^k, \beta_1^k, \beta_2^k, \bar{y}^k, \hat{\varepsilon}_k)$. We update the sequence of accuracies $\{\delta_k\}_{k\geq 0}$ as:

$$\delta_{k+1} := (1-\tau_k)\delta_k + \eta_k = \delta_k + (\eta_k - \tau_k\delta_k), \quad \forall k \geq 0, \tag{7.7.9}$$

where $\delta_0 \geq 0$ is chosen a priori. We need to find a condition on $\hat{\epsilon}_k$ such that $\{\delta_k\}_{k\geq 0}$ is nonincreasing. Indeed, we define:

$$\begin{cases} R_k := 2(1-\tau_k)\beta_1^k D_\sigma \left(\sum_{i=0}^M \sigma_i\right)^{1/2} + 0.5M[(1-\tau_k)\beta_2^k]^{-1}\sum_{i=1}^M \|A_i\|^2, \\ Q_k := M\left[\bar{L}^{-1}\beta_1^k C_d + (1-\tau_k)\tau_k\left((\beta_2^k)^{-1}C_d + \|A\|\,\|\bar{y}^k\|\right)\right] \\ \quad + 0.5\tau_k\beta_1^k\sum_{i=1}^M \sigma_i. \end{cases} \tag{7.7.10}$$

The following lemma provides a condition to update the vector of accuracies $\hat{\varepsilon}_k$.

Lemma 7.7.2. *If the accuracy $\hat{\varepsilon}_{ik}$ at the iteration k is chosen such that $0 \leq \hat{\varepsilon}_{ik} \leq \bar{\varepsilon}_k := \frac{\tau_k\delta_k}{R_k}$ in the scheme (7.7.2) and $0 \leq \hat{\varepsilon}_{ik} \leq \bar{\varepsilon}_k := \frac{\tau_k\delta_k}{Q_k}$ in the scheme (7.7.3) for $i = 1, \cdots, M$ then the sequence $\{\delta_k\}_{k\geq 0}$ generated by (7.7.9) is nonincreasing.*

Proof. We only prove the first case. Since $0 \leq \hat{\varepsilon}_{ik} \leq \bar{\varepsilon}_k$ for all $i = 1, \cdots, M$, we have $(\varepsilon_{[1]})_k \leq M\bar{\varepsilon}_k$ and $[(\varepsilon_{[\sigma]})_k]^2 \leq \left(\sum_{i=1}^M \sigma_{X_i}\right)\bar{\varepsilon}_k^2 \leq \left(\sum_{i=1}^M \sigma_{X_i}\right)\bar{\varepsilon}_k$. By substituting these inequalities into the definition (7.7.7) of η and then using (7.7.10) and the notation $\eta_k = \eta_1(\tau_k, \beta_1^k, \beta_2^k, \bar{y}^k, \hat{\varepsilon}_k)$, we have:

$$\eta_k \leq R_k\bar{\varepsilon}_k.$$

On the other hand, from (7.7.9) we have $\delta_{k+1} = \delta_k + (\eta_k - \tau_k\delta_k)$ for all $k \geq 0$. Thus, $\{\delta_k\}_{k\geq 0}$ is nonincreasing if $\eta_k - \tau_k\delta_k \leq 0$ for all $k \geq 0$. If we choose $\bar{\varepsilon}_k$ such that $R_k\bar{\varepsilon}_k \leq \tau_k\delta_k$, i.e. $\bar{\varepsilon}_k \leq \frac{\tau_k\delta_k}{R_k}$, then $\eta_k \leq \tau_k\delta_k$. □

If we choose $\bar{\varepsilon}_0$ in Lemma 7.7.2 such that $\bar{\varepsilon}_0 := \frac{\tilde{\varepsilon}}{C_0}$, where

$$C_0 := \begin{cases} 0.5\beta_2^{-1}M\sum_{i=1}^{M}\|A_i\|^2 & \text{if } (\bar{x}^0, \bar{y}^0) \text{ is defined by a)} \\ \beta_1\left(\bar{L}^{-1}C_d + 0.5\sum_{i=1}^{M}\sigma_i\right) & \text{if } (\bar{x}^0, \bar{y}^0) \text{ is defined by b),} \end{cases} \tag{7.7.11}$$

and $\tilde{\varepsilon} \geq 0$ is a given accuracy, then the condition (7.2.7) holds with $\delta = \tilde{\varepsilon}$.

Now we present a variant of Algorithm 7.3.1 in the inexact case.

Algorithm 7.7.1.(*Inexact decomposition algorithm with two primal steps*).
Initialization: Perform the following steps:

1. Provide a desired accuracy $\tilde{\varepsilon} \geq 0$ for solving the primal subproblems (7.1.8). Set $\tau_0 := 0.5$, $\beta_1^0 = \beta^0 > 0$ and set $\beta_2^0 := \frac{\bar{L}^2}{\beta_0}$.
2. Compute C_0 by (7.7.11). Set $\bar{\varepsilon}_0 := \tilde{\varepsilon}/C_0$ and $\delta_0 := \tilde{\varepsilon}$.
3. Compute $\bar{x}^0$ and $\bar{y}^0$ from (7.7.4) up to the given accuracy $\bar{\varepsilon}_0$.

Iteration: For $k = 0, 1, \cdots$ perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. Compute R_k by (7.7.10). Set $\bar{\varepsilon}_k := \tau_k\delta_k/R_k$ and update $\delta_{k+1} := (1 - \tau_k)\delta_k + R_k\bar{\varepsilon}_k$.
3. Update $\beta_2^{k+1} := (1 - \tau_k)\beta_2^k$.
4. Compute $(\bar{x}^{k+1}, \bar{y}^{k+1}) := \widetilde{\mathcal{S}}_{2\text{ps}}(\bar{x}^k, \bar{y}^k, \beta_1^k, \beta_2^{k+1}, \tau_k)$ up to the accuracy $\bar{\varepsilon}^k$.
5. Update $\beta_1^{k+1} := (1 - \tau_k)\beta_1^k$.
6. Update the step size τ_k as $\tau_{k+1} := \frac{\tau_k}{\tau_k+1}$.

End.

Symmetrically, we also obtain an inexact variant of Algorithm 7.4.1 as follows:
Algorithm 7.7.2.(*Inexact decomposition algorithm with two dual steps*).
Initialization: Perform as in Algorithm 7.7.1 with $\tau_0 := 0.5(\sqrt{5} - 1)$.
Iteration: For $k = 0, 1, \cdots$, perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. Compute Q_k by (7.7.10). Set $\bar{\varepsilon}_k := \tau_k\delta_k/Q_k$ and update $\delta_{k+1} := (1-\tau_k)\delta_k + Q_k\bar{\varepsilon}_k$.
3. Compute $(\bar{x}^{k+1}, \bar{y}^{k+1}) := \widetilde{\mathcal{S}}_{2\text{ds}}(\bar{x}^k, \bar{y}^k, \beta_1^k, \beta_2^k, \tau_k)$ up to the accuracy $\bar{\varepsilon}_k$.
4. Compute the factor $\alpha_k := p_X(\tilde{x}^*(\hat{y}^k;\beta_1^k))/D_X$.
5. Update $\beta_1^{k+1} := (1-\alpha_k\tau_k)\beta_1^k$ and $\beta_2^{k+1} := (1-\tau_k)\beta_2^k$.
6. Update the step size τ_k as:

$$\tau_{k+1} := 0.5\tau_k\left\{\left[(1-\alpha_k\tau_k)^2\tau_k^2 + 4(1-\alpha_k\tau_k)\right]^{1/2} - (1-\alpha_k\tau_k)\tau_k\right\}.$$

End.

Finally, we summarize the convergence results of Algorithms 7.7.1 and 7.7.2 in the following theorem. The proofs of this theorem can be done similarly as the proof of Theorems 7.3.4 and 7.4.3, which we omit the details here.

Theorem 7.7.2. *Suppose that Assumptions* ***A.6.1.7****,* ***A.7.1.8*** *and* ***A.7.4.9*** *are satisfied. Let* $\{(\bar{x}^k, \bar{y}^k)\}$ *be a sequence generated by Algorithm 7.7.1 after* $\bar{k}$ *iterations. If the accuracy* $\tilde{\varepsilon}$ *in this algorithm is chosen such that* $0 \le \tilde{\varepsilon} \le \frac{2c_0}{\bar{k}+2}$ *for some positive constant* c_0 *then the following duality gap holds:*

$$-R_{Y^*}\mathcal{F}(\bar{x}^{\bar{k}}) \le \phi(\bar{x}^{\bar{k}}) - g(\bar{y}^{\bar{k}}) \le \frac{2(\beta^0 D_X + c_0)}{\bar{k}+2}, \tag{7.7.12}$$

and the feasibility gap satisfies:

$$\mathcal{F}(\bar{x}^{\bar{k}}) = \left\|A\bar{x}^{\bar{k}} - b\right\| \le \frac{2(\bar{L}^2\beta_0^{-1}R_{Y^*} + \bar{L}\sqrt{2D_X + 2c_0\beta_0^{-1}})}{\bar{k}+2}. \tag{7.7.13}$$

Alternatively, let $\{(\bar{x}^k, \bar{y}^k)\}$ *be a sequence generated by Algorithm 7.7.2 after* $\bar{k}$ *iterations. If the accuracy* $\tilde{\varepsilon}$ *in this algorithm is chosen such that* $0 \le \tilde{\varepsilon} \le \frac{c_0}{0.5(\sqrt{5}-1)\bar{k}+1}$ *for some positive constant* c_0 *then the following duality gap holds:*

$$-R_{Y^*}\mathcal{F}(\bar{x}^{\bar{k}+1}) \le \phi(\bar{x}^{\bar{k}+1}) - g(\bar{y}^{\bar{k}+1}) \le \frac{(\beta^0 D_X + c_0)}{[0.5(\sqrt{5}-1)\bar{k}+1]^{\alpha^*}}, \tag{7.7.14}$$

and the feasibility gap satisfies:

$$\mathcal{F}(\bar{x}^{\bar{k}+1}) = \left\|A\bar{x}^{\bar{k}+1} - b\right\| \le \frac{C_f}{0.25(\sqrt{5}-1)(1+\alpha^*)\bar{k}+1}, \tag{7.7.15}$$

where $C_f := (3-\sqrt{5})\frac{\bar{L}^2}{\beta_0}R_{Y^*} + 0.5\bar{L}(\sqrt{5}-1)(D_X + c_0\beta_0^{-1})^{1/2}$.

Consequently, the sequence $\{(\bar{x}^k, \bar{y}^k)\}_{k\geq 0}$ *generated by either Algorithm 7.7.1 or Algorithm 7.7.2 converges to a solution* (x^*, y^*) *of the primal and dual problems* (SepCOP)-(7.1.1) *as* $k \to \infty$ *and* $\tilde{\varepsilon} \to 0^+$.

The conclusion of Theorem 7.7.2 shows that the initial accuracy level $\tilde{\varepsilon}$ of solving the primal subproblems (7.1.8) needs to be chosen as $O(1/k)$. Then we have $\left|\phi(\bar{x}^k) - g(\bar{y}^k)\right| = O(1/k^{\alpha^*})$ and $\mathcal{F}(\bar{x}^k) = O(1/k)$ in Algorithm 7.7.2. We note that the accuracy level of solving the primal subproblems (7.1.8) has to update at each iteration k of Algorithms 7.7.1 and 7.7.2. The new value is computed by $\tilde{\varepsilon}^k = \tau_k\delta_k/Q_k$ at Step 2 which has the same order as $1/k^2$.

7.8 Comparison and implementation aspects

In this section, we first make a theoretical comparison of the algorithms proposed in the previous sections. Then we discuss the stopping criterion of these algorithms. Finally, we present some remarks on distributed implementation of the algorithms.

Theoretical comparison

As we can see from the conclusions of Theorems 7.3.4, 7.4.3, 7.5.1 and 7.5.3 that the convergence rate of the algorithms depends on the convergence rate of the sequences $\left\{\beta_1^k\right\}$ and $\left\{\beta_2^k\right\}$. We notice that Algorithms 7.3.1 and 7.5.2 have the same convergence rate. In this subsection, we make a theoretical comparison for these algorithms. Indeed, at each iteration, Algorithm 7.3.1 updates simultaneously β_1^k and β_2^k by using the same value of τ_k, while Algorithm 7.5.2 updates only one parameter. Therefore, to update both parameters β_1^k and β_2^k, Algorithm 7.5.2 needs two iterations. We analyze the update rule of τ_k in Algorithms 7.3.1 and 7.5.2 to compare the rate of convergence of both algorithms.

Let us define

$$\xi_1(\tau) := \frac{\tau}{\tau+1} \text{ and } \xi_2(\tau) := \frac{\tau}{2}\left[\sqrt{\tau^2+4} - \tau\right].$$

The function ξ_2 can be rewritten as $\xi_2(\tau) = \frac{\tau}{\sqrt{(\tau/2)^2+1}+\tau/2}$. Therefore, we can easily show that:

$$\xi_1(\tau) < \xi_2(\tau) < 2\xi_1(\tau).$$

If we denote by $\{\tau_k^{\mathrm{A}_1}\}_{k\geq 0}$ and $\{\tau_k^{\mathrm{A}_2}\}_{k\geq 0}$ the two sequences generated by Algorithms 7.3.1 and 7.5.2, respectively then we have $\tau_k^{\mathrm{A}_1} < \tau_k^{\mathrm{A}_2} < 2\tau_k^{\mathrm{A}_1}$ for all k provided that $2\tau_0^{\mathrm{A}_1} \geq \tau_0^{\mathrm{A}_2}$. Since Algorithm 7.3.1 updates β_1^k and β_2^k simultaneously while Algorithm 7.5.2 updates each of them at each iteration. If we choose $\tau_0^{\mathrm{A}_1} = 0.5$ and $\tau_0^{\mathrm{A}_2} = 0.5(\sqrt{5}-1)$ in Algorithms 7.3.1 and 7.5.2, respectively, then, by directly computing the values of $\tau_k^{\mathrm{A}_1}$ and $\tau_k^{\mathrm{A}_2}$, we can see that $2\tau_k^{\mathrm{A}_1} > \tau_k^{\mathrm{A}_2}$ for all $k \geq 1$. Consequently, the sequences $\{\beta_1^k\}$ and $\{\beta_2^k\}$ in Algorithm 7.3.1 converge to zero faster than in Algorithm 7.5.2. In other words, Algorithm 7.3.1 is faster than Algorithm 7.5.2.

Now, we compare Algorithms 7.3.1-7.5.2 and Algorithm 3.2. in [134] (see also [203]). Note that the smoothness parameter β_1 is fixed in Algorithm 3.2 of [134]. Moreover, this parameter is proportional to the given desired accuracy ε, i.e. $\beta_1 := \frac{\varepsilon}{D_X}$, which is often very small. Thus, the Lipschitz constant $L^d(\beta_1)$ is very large. Consequently, [134, Algorithm 3.2] makes slow progress at the very early iterations. In Algorithms 7.3.1-7.5.2, the parameters β_1 and β_2 are dynamically updated starting from given values. Besides, the cost per iteration of [134, Algorithm 3.2] is higher than Algorithms 7.3.1-7.5.2 since it requires one to perform two primal steps and two dual steps at each iteration.

Stopping criterion

In practice, we do not often encounter a problem which reaches the worst-case complexity bound. Therefore, it is necessary to provide a stopping criterion for the implementation of Algorithms 7.3.1-7.7.2 to terminate earlier than using the worst-case bound. In principle, we can use the KKT condition to terminate the algorithms. However, evaluating the global KKT tolerance in a distributed manner is impractical.

In the following implementation, we used the smoothed dual function $g(\cdot;\beta_1)$ to measure the stopping criterion. It is clear that if β_1 is small then $g(\cdot;\beta_1)$ is an approximation of the dual function g due to Lemma 7.1.1. Therefore, we can approximate the duality gap $\phi(x) - g(y)$ by $\phi(x) - g(y;\beta_1)$ and use this quantity in the stopping criterion. More precisely, we terminate the algorithms if:

$$\mathcal{F}(\bar{x}^k) := \left\|A\bar{x}^k - b\right\| / \max\{\left\|A\bar{x}^0 - b\right\|, 1.0\} \leq \varepsilon_{\mathrm{feas}}, \tag{7.8.1}$$

and either the approximate duality gap satisfies:

$$\left|\phi(\bar{x}^k) - g(\bar{y}^k;\beta_1^k)\right| \leq \varepsilon_{\mathrm{fun}} \max\left\{1.0, \left|g(\bar{y}^k;\beta_1^k)\right|, \left|\phi(\bar{x}^k)\right|\right\}, \tag{7.8.2}$$

or the value $\phi(\bar{x}^k)$ does not significantly change in $j_{\max}$ successive iterations, i.e.:

$$\left|\phi(\bar{x}^k) - \phi(\bar{x}^{k-j})\right| / \max\{1.0, \left|\phi(\bar{x}^k)\right|\} \leq \varepsilon_{\text{obj}} \text{ for } j = 1, \ldots, j_{\max}, \qquad (7.8.3)$$

where $\varepsilon_{\text{feas}}$, ε_{fun} and ε_{obj} are given tolerances.

According to Lemmas 7.1.1 and 7.1.3, the condition (7.8.2) can be used to ensure the decrease of the duality gap $\phi(\bar{x}^k) - g(\bar{y}^k)$ as β_1^k and β_2^k are sufficiently small. The condition (7.8.3) is heuristic and may not guarantee the approximate optimality of $(\bar{x}^k, \bar{y}^k)$.

Remarks on distributed implementation

We show that Algorithms 7.3.1-7.6.1 proposed in this chapter can be implemented in a distributed manner. First, we note that, in a distributed setting, each component of problem (SepCOP) is formed from a subsystem of the overall distributed system. Each subsystem is connected to its neighbours via communication links. It only communicates to its neighbours and can exchange data with them. This operation is characterized in the coupling matrix A of (SepCOP). We can see from the above algorithms that the *primal step* corresponding to solving either M primal subproblems (7.1.8) or M convex problems (7.2.13) can be performed *in parallel*, while the *dual step* (7.4.3) can be implemented distributively based on the structure of matrix A.

Next, we discuss the computation of the following global parameters of the algorithms:

- The parameters τ_k, β_1^k and β_2^k do not depend on the data of the problems. They can be parallelized by using the same formula and starting from the same value in all subsystems.
- The constant $\bar{L}$ is evaluated once at the initial phase and then it is sent to all subsystems.
- The Lipschitz constant $L_i^{\psi}(\beta_2)$ can be chosen as $L_i^{\psi}(\beta_2) := \frac{M\|A_i\|^2}{\beta_2}$ which can be computed in parallel.
- We can use the lower bound α^* instead of the factor α_k in Algorithms 7.4.1 and 7.5.1. This constant is given a priori.

Finally, we note that, for each $i \in \{1, \ldots, M\}$, the cost of solving the primal subproblem i depends on the complexity of the objective function ϕ_i and

the local constraint set X_i. We need to analyze the complexity of solving this problem in order to trade-off the computational time in the nodes of the distributed computing system as mentioned in Section 6.4 of Chapter 6.

7.9 Numerical tests

In this section, we test the algorithms presented in the previous sections for solving four numerical examples. The first example is a nonsmooth separable convex optimization problem which appears in resource allocation [108]. The second and the third ones are academic examples for separable quadratic programming and nonlinear convex programming, respectively. The last example is an application to DSL dynamic spectrum management [202, 203].

Implementation details

The algorithms developed in the previous sections have been implemented in C++ running on a 16 cores Intel ®Xeon 2.7GHz workstation with 12 GB of RAM. The algorithms were parallelized by using `OpenMP`. In order to solve general convex primal subproblems in these algorithms, we have used two different solvers:

- `Cplex` – a commercial software for (integer) linear/quadratic programing with academic license free [103];
- `IpOpt` – an open source software package based on interior point methods for nonlinear optimization [210];

The accuracy level of these solvers is fixed at 10^{-8}. Moreover, we warm-started the `Cplex` and `IpOpt` solvers at the iteration k at the point given by the previous iteration $k-1$ for $k \geq 1$. We compared our algorithms with the following algorithms in certain numerical examples:

- The proximal center based decomposition algorithm proposed in [134], which we abbreviated by `PCBDM`.
- An exact variant of the proximal based decomposition algorithm (`EPBDM`) proposed in [39].
- A parallel variant of the alternating direction method of multipliers considered in [122] which we named `ADMM`. In this algorithm, we used

three different strategies to update the penalty parameter. In the first strategy, we fixed the penalty parameter at 10^3, while in the second and the third versions, we updated the penalty parameter ρ_k by using a strategy proposed in [95] which started from $\rho_0 = 1$ and $\rho_0 = 10^3$, respectively. We denoted these variants by `ADMM-v3`, `ADMM-v1` and `ADMM-v2`, respectively.

We chose the quadratic prox-function $p_{X_i}(x_i) := \frac{1}{2}\|x_i - x_i^c\|^2 + r_i$ in the first four algorithms, i.e. Algorithms 7.3.1-7.5.2, where $x_i^c \in \mathbb{R}^{n_i}$ and $r_i = 0.75D_{X_i}$ are given, for $i = 1, \cdots, M$.

The parameter β_1 in the primal subproblems of `PCBDM` was fixed at $\beta_1 := \frac{\varepsilon_{\text{fun}} \max\{1.0, |\phi(\bar{x}^0)|\}}{D_X}$. For `EPBDM`, we used an exact variant of [39, Algorithm 1], where we chose the proximity parameter as follows. First, we chose $\varepsilon_c := 0.5 \min\left\{\frac{1}{3}, \frac{1}{2\|A\|+1}\right\}$ and then set $\underline{\beta}_1 := \left[\min\left\{\frac{1-\varepsilon_c}{2}, \frac{1-\varepsilon_c}{2\|A\|}\right\}\right]^{-1}$ and $\bar{\beta}_1 := \varepsilon_c^{-1}$. Finally, we selected $\beta_1 = 0.5(\underline{\beta}_1 + \bar{\beta}_1)$.

We terminated Algorithms 7.3.1-7.5.2 by using the conditions in Section 7.8, where $\varepsilon_{\text{feas}} = \varepsilon_{\text{fun}} = \varepsilon_{\text{obj}} = 10^{-3}$ and $j_{\max} = 5$. We terminated all the remaining algorithms if both conditions (7.8.1) and (7.8.3) were satisfied. The maximum number of iterations `maxiter` was set to 5000 in all algorithms. We declared that a problem could not be solved if `Cplex` or `IpOpt` failed or the maximum number of iterations `maxiter` is reached. We named Algorithms 7.3.1-7.5.2 by `A`.7.3.1-A.7.5.2, respectively, for short.

Nonsmooth separable convex optimization

Let us consider the following simple nonsmooth convex optimization problem:

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & \phi(x) := \sum_{i=1}^n i\,|x_i - x_i^a|\,, \\ \text{s.t.} & \sum_{i=1}^n x_i = b,\ x_i \in X_i,\ i = 1, \ldots, n, \end{cases} \tag{7.9.1}$$

where $b, x_i^a \in \mathbb{R}$ are given $(i = 1, \ldots, n)$. Let us assume that $x_i \in X_i := [l_i, u_i]$ a given interval in $\mathbb{R}$. Then, this problem can be formulated in the form of (SepCOP) with $M = n$. Since the Lagrange function $\mathcal{L}(x, y) = \sum_{i=1}^n [i\,|x_i - x_i^a| + y(x_i - b/n)]$ is nonsmooth, where $y \in \mathbb{R}$ is a Lagrange multiplier, we choose $p_{X_i}(x_i) := \frac{1}{2}\|x_i - x_i^c\|^2 + 0.75D_{X_i}$ such that the primal subproblem (7.1.6) can be written as:

$$g_i(y; \beta_1) := \min_{x_i \in [l_i, u_i]} \Big\{ i\,|x_i - x_i^a| + y\big(x_i - \frac{b}{n}\big) + \frac{\beta_1}{2}\,|x_i - x_i^c|^2 + 0.75D_{X_i} \Big\}, \tag{7.9.2}$$

where $\beta_1 > 0$. Now, we assume that we can choose the interval $[l_i, u_i]$ sufficiently large such that the constraint $x_i \in [l_i, u_i]$ is inactive. Then the solution of

problem (7.9.2) can be computed explicitly as: $x_i^*(y;\beta_1) := V_i(x_i^a, x_i^c, y, \beta_1, i)$, where the *shrinkage operator* V_i is defined as follows:

$$V_i(x_i^a, x_i^c, y, \beta_1, \gamma) := \begin{cases} x_i^c - \beta_1^{-1}(\gamma + y) & \text{if } x_i^c - \beta_1^{-1}(\gamma + y) > x_i^a, \\ x_i^c + \beta_1^{-1}(\gamma - y) & \text{if } x_i^c + \beta_1^{-1}(\gamma - y) < x_i^a, \\ x_i^a & \text{if } y + \beta_1(x_i^a - x_i^c) \in [-\gamma, \gamma]. \end{cases} \tag{7.9.3}$$

In this example, we tested five algorithms: Algorithm 7.3.1, Algorithm 7.4.1, Algorithm 7.5.1, Algorithm 7.5.2 and `PCBDM` for 10 problems with the size varying from $n = 5$ to $n = 100,000$. Note that if we reformulate (7.9.1) as a linear programming problem (LP) by introducing slack variables then the resulting LP problem has $2n$ variables and $2n + 1$ inequality constraints.

The data of these tests were created as follows. The value c was set to $b = 2n$, $x^a := (x_1^a, \ldots, x_n^a)^T$, where $x_a^i := i - n/2$. The maximum number of iterations `maxiter` was increased to $10,000$ instead of $5,000$. The performance of the five algorithms is reported in Table 7.1. Here, `iters` is the number of iterations

Table 7.1: Performance comparison of five algorithms for solving (7.9.1)

		Algorithm performance and results									
Size $[n]$		5	10	50	100	500	1,000	5,000	10,000	50,000	100,000
iters	A.7.3.1	226	184	704	843	1211	1277	**1371**	**1387**	1408	1409
	A.7.4.1	1216	925	**377**	**552**	**1092**	**1209**	1385	1422	1374	**1352**
	A.7.5.1	452	334	544	794	1142	1228	1415	1433	**1358**	1368
	A.7.5.2	612	458	830	887	1253	1341	1451	1428	1487	1446
	PCBDM	**62**	**123**	507	1036	3767	3693	6119	5816	3099	3285
time	A.7.3.1	0.0143	0.0105	0.0339	0.0495	0.0809	0.1078	0.2969	0.5943	2.5055	4.9713
	A.7.4.1	0.0592	0.0418	**0.0170**	**0.0266**	**0.0596**	**0.0827**	**0.2477**	0.4544	2.1970	4.3869
	A.7.5.1	0.0244	0.0166	0.0222	0.0406	0.0737	0.0909	0.3522	0.4646	**2.0875**	**4.2659**
	A.7.5.2	0.0316	0.0199	0.0351	0.0450	0.0716	0.0979	0.3013	**0.4416**	2.2879	4.3119
	PCBDM	**0.0027**	**0.0036**	0.0218	12.1021	0.2116	0.2232	1.1448	1.3084	3.0277	6.3322

and `time` is the CPU time in seconds.

As we can see from Table 7.1, Algorithm 7.4.1 is the best in terms of number of iterations and computational time. Algorithm 7.5.1 works better than Algorithm 7.5.2. The first four algorithms have consistently outperformed `PCBDM` in terms of number of iterations as well as computational time in this example.

Separable convex quadratic programming

We consider the following separable convex quadratic programming problem:

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} & \phi(x) := \sum_{i=1}^M \frac{1}{2} x_i^T Q_i x_i + q_i^T x_i, \\ \text{s.t.} & \sum_{i=1}^M A_i x_i = b, \\ & x_i \geq 0, \ i = 0, \cdots, M. \end{cases} \tag{7.9.4}$$

Here $Q_i \in \mathbb{R}^{n_i \times n_i}$ is a symmetric positive semidefinite matrix, $q_i \in \mathbb{R}^{n_i}$, $A_i \in \mathbb{R}^{m \times n_x}$ for $i = 1, \ldots, M$ and $b \in \mathbb{R}^m$. In this example, we compared the above algorithms by building their *performance profiles* in terms of number of iterations and the total of computational time, see Section 6.5 of Chapter 6.

Problem generation. The input data of the tested collection was generated as follows:

- Matrix $Q_i := R_i R_i^T$, where R_i is an $n_i \times r_i$ random matrix in $[l_Q, u_Q]$ with $r_i := \lfloor n_i/2 \rfloor$.
- Matrix A_i was generated randomly in $[l_A, u_A]$.
- Vector $q_i := -Q_i x_i^0$, where x_i^0 is a given feasible point in $(0, r_{x_0})$ and vector $b := \sum_{i=1}^{M} A_i x_i^0$.
- The density of both matrices A_i and R_i is γ_A.

Note that the problems generated as above are always feasible. Moreover, they are not strongly convex. The tested collection consisted of $n_p = 50$ problems with different sizes and the sizes were generated randomly as follows:

- *Class 1:* 20 problems with $20 < M < 100$, $50 < m < 500$, $5 < n_i < 100$ and $\gamma_A = 0.5$.
- *Class 2:* 20 problems with $100 < M < 1000$, $100 < m < 600$, $10 < n_i < 50$ and $\gamma_A = 0.1$.
- *Class 3:* 10 problems with $1000 < M < 2000$, $500 < m < 1000$, $100 < n_i < 200$ and $\gamma_A = 0.05$.

Scenarios. We considered two different scenarios:
Scenario I: In this scenario, we aimed at comparing Algorithms 7.3.1-7.5.2, `ADMM-v1` and `EPBDM`, where we generated the values of Q relatively small. More precisely, we chose $[l_Q, u_Q] = [-0.1, 0.1]$, $[l_A, u_A] = [-1, 1]$ and $r_{x_0} = 2$.

Scenario II: The second scenario aimed at testing the affect of matrix A and the update rule of the penalty parameter to the performance of `ADMM`. We chose $[l_Q, u_Q] = [-1, 1]$, $[l_A, u_A] = [-5, 5]$ and $r_{x_0} = 5$.

Results. In the first scenario, the size of the problems satisfied $23 \leq M \leq 1992$, $95 \leq m \leq 991$ and $1111 \leq n \leq 297818$. The performance profiles of the six algorithms are plotted in Figure 7.1 with respect to the number of iterations and computational time.

From these performance profiles, we can observe that Algorithms 7.3.1-7.5.2 converged for all problems. `ADMM-v1` was successful in solving 36/50 (72.00%) problems while `EPBDM` could only solve 9/50 (18.00%) problems. It shows that Algorithm 7.4.1 is the best in terms of number of iterations. It could solve up to 38/50 (76.00%) problems with the best performance. `ADMM-v1` solved 10/50 (20.00%) problems with the best performance, while this ratio was only 2/50 (4.00%) and 1/50 (2.00%) in Algorithm 7.5.2 and Algorithm 7.5.1, respectively. If we compare the computational time then Algorithm 7.4.1 is the best. It could solve up to 43/50 (86.00%) problems with the best performance. `ADMM-v1` solved 7/50 (14.00%) problems with the best performance.

Since the performance of Algorithms 7.3.1-7.5.2 and `ADMM` are relatively comparable, we tested Algorithms 7.3.1-7.5.2, `ADMM-v1`, `ADMM-v2` and `ADMM-v3` on a collection of $n_p = 50$ problems in the second scenario. The performance profiles of these algorithms are shown in Figure 7.2.

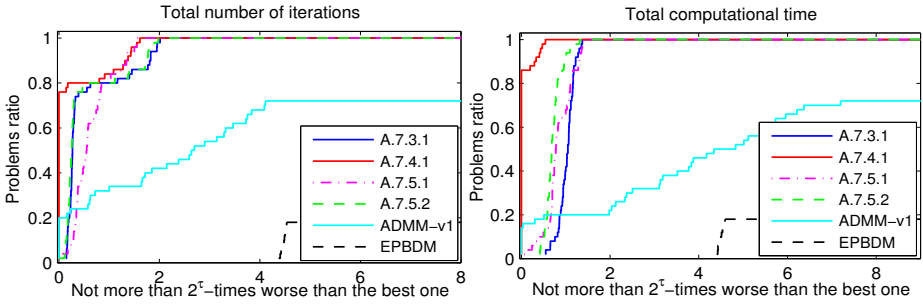


Figure 7.1: Performance profiles in $\log_2$ scale for Scenario I by using `IpOpt`: Left-Number of iterations, Right-Computational time.

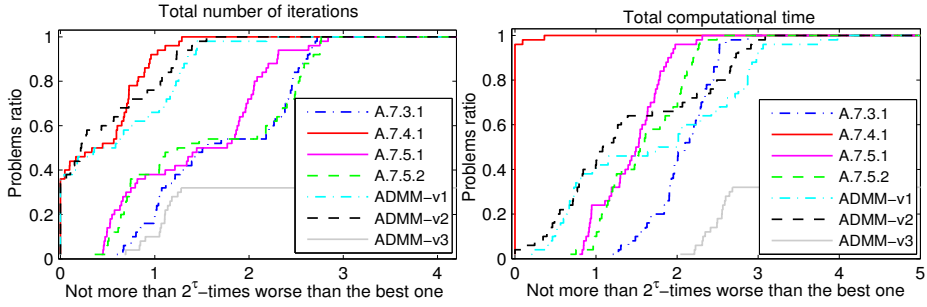


Figure 7.2: Performance profiles in $\log_2$ scale for Scenario II by using `Cplex` with Simplex method: Left-Number of iterations, Right-Computational time.

From these performance profiles we can observe the following. First, the six first algorithms were successful in solving all problems, while `ADMM-v3` could only solve 16/50 (32%) problems. Second, Algorithm 7.4.1 and `ADMM-v1` is the best in terms of number of iterations. It both solved 18/50 (36%) problems with the best performance. This ratio is 17/50 (34%) in `ADMM-v2`. Third, Algorithm 7.4.1 is the best in terms of computational time. It could solve 48/50 (96%) the problems with the best performance, while this number is 2/50 (4%) in `ADMM-v2`.

Nonlinear smooth separable convex programming

We consider the following nonlinear, smooth and separable convex programming problem:

$$\begin{cases} \min\limits_{x_i \in \mathbb{R}^{n_i}} & \Big\{\phi(x) := \sum\limits_{i=1}^{M} \frac{1}{2}(x_i - x_i^0) Q_i (x_i - x_i^0) - w_i \ln(1 + b_i^T x_i)\Big\}, \\ \text{s.t.} & \sum\limits_{i=1}^{M} A_i x_i = b,\ x_i \succeq 0, \quad i = 1, \cdots, M. \end{cases} \tag{7.9.5}$$

Here, Q_i is a positive semidefinite and x_i^0 is given vector, $i = 1, \cdots, M$.

Problem generation. In this example, we generated a collection of $n_p = 50$ test problems based on the following steps:

- Matrix Q_i is diagonal and was generated randomly in $[l_Q, u_Q]$.
- Matrix A_i was generated randomly in $[l_A, u_A]$ with the density γ_A.
- Vectors b_i and w_i were generated randomly in $[l_b, u_b]$ and $[0, 1]$, respectively, such that $w_i \geq 0$ and $\sum_{i=1}^{M} w_i = 1$.
- Vector $b := \sum_{i=1}^{M} A_i x_i^0$ for a given x_i^0 in $[0, r_{x_0}]$.

The size of the problems was generated randomly based on the following rules:

- *Class 1:* 10 problems with $20 < M < 50$, $50 < m < 100$, $10 < n_i < 50$ and $\gamma_A = 1.0$.
- *Class 2:* 10 problems with $50 < M < 250$, $100 < m < 200$, $20 < n_i < 50$ and $\gamma_A = 0.5$.
- *Class 3:* 10 problems with $250 < M < 1000$, $100 < m < 500$, $50 < n_i < 100$ and $\gamma_A = 0.1$.

- *Class 4:* 10 problems with $1000 < M < 5000$, $500 < m < 1000$, $50 < n_i < 100$ and $\gamma_A = 0.05$.
- *Class 5:* 10 problems with $5000 < M < 10000$, $500 < m < 1000$, $50 < n_i < 100$ and $\gamma_A = 0.01$.

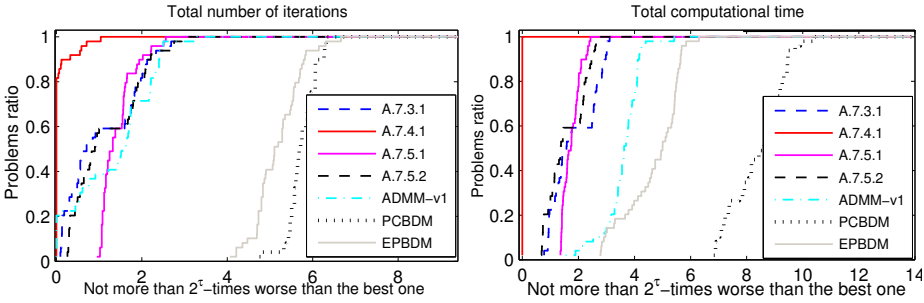


Figure 7.3: Performance profiles on `Scenario II` in $\log_2$ scale by using `IpOpt`: Left-Number of iterations, Right-Computational time.

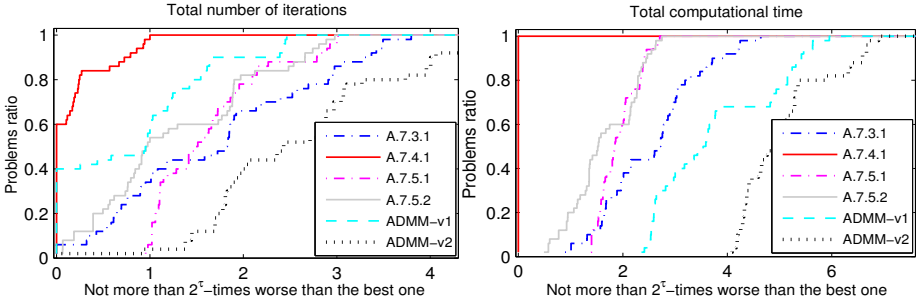


Figure 7.4: Performance profiles in $\log_2$ scale for `Scenario I` by using `IpOpt`: Left-Number of iterations, Right-Computational time.

Scenarios. We also considered two different scenarios as in the previous example:
Scenario I: Similar to the previous example, with this scenario, we aimed at comparing Algorithms 7.3.1-7.5.2, `ADMM-v1`, `PCBDM` and `EPBDM`. In this scenario, we chose: $[l_Q, u_Q] \equiv [-0.01, 0.01]$, $[l_b, u_b] \equiv [0, 100]$, $[l_A, u_A] \equiv [-1, 1]$ and $r_{x_0} = 1$.
Scenario II: In this scenario, we only tested first two variants of `ADMM` and compared them with the first four algorithms. Here, we chose $[l_Q, u_Q] \equiv [0.0, 0.0]$ (i.e. without quadratic term), $[l_b, u_b] \equiv [0, 100]$, $[l_A, u_A] \equiv [-1, 1]$ and $r_{x_0} = 10$.

Results. For *Scenario I*, we see that the size of the problems is in $20 \leq M \leq 9938$, $50 \leq m \leq 999$ and $695 \leq n \leq 741646$. The performance profiles of the algorithms are plotted in Figure 7.3. The results on this collection show that Algorithm 7.4.1 is the best in terms of number of iterations. It could solve up to 41/50 (82%) problems with the best performance, while `ADMM-v1` solved 10/50 (20%) problems with the best performance. Algorithm 7.4.1 is also the best in terms of computational time. It could solve 50/50 (100%) problems with the best performance. `PCBDM` was very slow compared to the rest in this scenario.

For *Scenario II*, the size of the problems was varying in $20 \leq M \leq 9200$, $50 \leq m \leq 946$ and $695 \leq n \leq 684468$. The performance profiles of the tested algorithms are plotted in Figure 7.4. We can see from these performance profiles that Algorithm 7.4.1 is the best in terms of number of iterations. It could solve up to 30/50 (60%) problems with the best performance, while this number were 3/50 (6%) and 20/50 (40%) problems in Algorithm 7.3.1 and `ADMM-v1`, respectively. Algorithm 7.4.1 was also the best in terms of computational time. It solved all problems with the best performance. `ADMM-v2` was slow compared to the rest in this scenario.

DSL dynamic spectrum management optimization

Finally, we applied Algorithm 7.6.1 to solve a separable convex programming problem arising in DSL dynamic spectrum management. This problem is a convex relaxation of the original DSL dynamic spectrum management formulation considered in [202]. The objective function of this problem is given by:

$$\phi(x) := \sum_{i=1}^{M} \phi_i(x_i), \quad \text{where} \quad \phi_i(x_i) := a_i^T x_i - \sum_{j=1}^{n_i} c_i^j \ln\Big(\sum_{l=1}^{n_i} p_i^{jl} x_i^l + q_i^l\Big). \quad (7.9.6)$$

Here, $a_i \in \mathbb{R}^{n_i}$, $c_i,\ q_i \in \mathbb{R}^{n_i}_+$ and $P_i := (p_i^{jk}) \in \mathbb{R}^{n_i \times n_i}_+$, $(i = 1, \cdots, M)$. As described in [203] the variable x_i refers to as a transmit power spectral density, $n_i = N$ for all $i = 1, \cdots, M$ is the number of users, M is the number of frequency tones which is usually large and ϕ_i is a convex approximation of a desired BER function[1], the coding gain and noise margin. A detailed model and parameter descriptions of this problem can be found in [202, 203].

Since the function ϕ is convex (but not strongly convex), we added a regularization term $\frac{\beta_1}{2}\|x - x^c\|^2$ to the objective of the original problem, where $\beta_1 > 0$ is relatively small and x^c is the prox-center of X. The objective function

[1] **B**it **E**rror **R**ate function

$\tilde{\phi}(x) := \phi(x) + \frac{\beta_1}{2}\|x - x^c\|^2$ of the resulting problem is strongly convex with a convexity parameter β_1. Moreover, we have $\left|\tilde{\phi}(x) - \phi(x)\right| \leq \beta_1 D_X$ for all $x \in X$, where D_X is defined by (7.1.5). Therefore, if we apply Algorithm 7.6.1 to find a vector $\bar{x}^k$ as an ε approximate solution of the resulting problem then $\bar{x}^k$ is also an $\varepsilon + \beta_1 D_X$ approximate solution of the original problem. In our problem, D_X is proportional to 10^{-6} and the magnitudes of the objective function are proportional to 10^3. In order to get the relative accuracy $O(10^{-3})$ we chose β_1 between $[10^5, 10^6]$. The resulting problem is indeed in the form of (SepCOP) with a strongly convex objective function.

We tested Algorithm 7.6.1 for solving the above resulting problem with different 9 scenarios and compared the results with `ADMM-v3`, `PCBDM` and `EPBDM`. The parameters of the problems were selected as in [202, 203]. In this example, we observed that `ADMM-v3` was the most suitable of the three ADMM variants. We also note that the problem possesses coupling inequality constraints. In `ADMM` we added a slack variable x_{M+1} to transform it into a problem with equality coupling constraints.

The numerical results of the four algorithms are reported in Table 7.2 for the 9 different scenarios.

Table 7.2: The performance information and results of Example 7.2.3.

Algorithm	scen	size	n_vars	iter	cpu_time[s]	obj_value	rel_fgap
Algorithm 7.6.1	P13	[477, 12]	5724	**36**	0.294	3565.508	6.150×10^{-4}
ADMM-v3				284	1.968	3560.602	9.430×10^{-4}
PCBDM				309	1.957	3567.030	9.540×10^{-4}
EPBDM				326	2.084	3566.794	9.610×10^{-4}
Algorithm 7.6.1	P23	[477, 12]	5724	**31**	0.216	3787.826	9.600×10^{-4}
ADMM-v3				239	1.608	3784.410	8.650×10^{-4}
PCBDM				335	2.186	3787.604	9.620×10^{-4}
EPBDM				409	2.705	3787.303	9.680×10^{-4}
Algorithm 7.6.1	P33	[477, 12]	5724	**35**	0.253	3340.921	7.520×10^{-4}
ADMM-v3				299	1.968	3329.525	7.760×10^{-4}
PCBDM				337	2.211	3341.863	9.640×10^{-4}
EPBDM				370	2.734	3341.517	9.640×10^{-4}
Algorithm 7.6.1	P43	[477, 12]	5724	**30**	0.248	3884.034	7.120×10^{-4}
ADMM-v3				173	1.243	3883.697	9.840×10^{-4}
PCBDM				276	1.896	3884.884	9.740×10^{-4}
EPBDM				347	2.226	3884.103	9.670×10^{-4}
Algorithm 7.6.1	P53	[1147, 6]	6882	**179**	1.559	3397.298	9.420×10^{-4}
ADMM-v3				806	6.769	3353.029	6.530×10^{-4}
PCBDM				3103	24.479	3399.864	9.960×10^{-4}
EPBDM				1242	11.893	3399.040	9.960×10^{-4}
Algorithm 7.6.1	P63	[224, 7]	1568	**33**	0.089	873.976	0.160×10^{-4}
ADMM-v3				47	0.098	875.427	5.060×10^{-4}
PCBDM				186	0.363	876.572	8.990×10^{-4}
EPBDM				166	0.324	876.631	9.190×10^{-4}
Algorithm 7.6.1	P73	[224, 7]	1568	**20**	0.044	1039.686	5.110×10^{-4}
ADMM-v3				122	0.241	1037.106	4.840×10^{-4}

PCBDM				135	0.267	1040.370	8.800×10^{-4}
EPBDM				106	0.241	1040.364	8.790×10^{-4}
Algorithm 7.6.1	P83	[224, 2]	448	**25**	0.025	445.135	5.140×10^{-4}
ADMM-v3				131	0.113	446.438	8.040×10^{-4}
PCBDM				420	0.350	447.245	9.740×10^{-4}
EPBDM				438	0.352	447.227	9.720×10^{-4}
Algorithm 7.6.1	P93	[1147, 12]	13764	**20**	0.481	1565.422	0.150×10^{-4}
ADMM-v3				1377	25.120	1566.989	9.750×10^{-4}
PCBDM				529	8.847	1567.579	9.840×10^{-4}
EPBDM				114	2.047	1567.543	9.840×10^{-4}

Here, `scen` indicates the scenarios; `size` and `n_vars` are the size of the problem (number of tones and number of users) and the number of variables, respectively; `iter` and `cpu_time` are the number of iterations and the CPU time in seconds, respectively; `obj_value` and `rel_fgap` are the objective value and relative feasibility gaps, respectively. As we can see from Table 7.2, Algorithm 7.6.1 shows the best performance both in terms of number of iterations and computational time in all the scenarios of this example.

7.10 Conclusion

In this chapter we have developed two new decomposition algorithms for solving separable convex optimization problems based on three techniques, namely dual decomposition, excessive gap and smoothing via prox-functions. We have analyzed the convergence of these algorithms and established their convergence rate. Both algorithms have been modified to obtain different variants including methods for the strongly convex case and the inexact case. The algorithms have been verified through several numerical examples and also compared with many other methods in the literature. The convergence rate of the proposed algorithms is $O(1/k)$, where k is the iteration counter, and thus they can be classified as the optimal first-order methods in the framework of dual decomposition (in the sense of Nesterov [142]). One main advantage of these algorithms is that they can update automatically the algorithmic parameters without any tuning strategy. Moreover, they can be implemented in a parallel or distributed manner.

Chapter 8

Path-following gradient decomposition algorithms

In the previous chapter, we have studied a class of first order methods by making use of a smoothing technique via prox-functions. However, prox-functions depend on the geometry of the feasible set of the problem. Besides, the primal subproblems in the proposed algorithms are still general convex programs. In this chapter, we study a smoothing technique via self-concordant barrier functions [131, 133, 138, 171, 218] and propose gradient-type decomposition algorithms for solving separable convex optimization problems. This approach has two advantages. First, the worst-case complexity bound only depends on the parameter of the barrier functions rather than the geometry of the feasible set. Second, the primal subproblems can be solved via a system of generalized equations instead of general convex programs as in the previous chapter.

Contribution of Chapter 8. The contribution of this chapter is as follows:

a) By applying smoothing techniques via self-concordant barrier functions to the primal problem, we prove some new estimates between the original dual function and the smoothed dual function. We also show other properties of the smoothed dual function which will be used to design the algorithms.

b) We propose a new path-following gradient-based decomposition algorithm, Algorithm 8.2.1, for solving the dual problem and prove its convergence. We also estimate the local convergence rate of this algorithm.

c) Then we adapt Algorithm 8.2.1 by using the framework of fast gradient methods to obtain a new variant, Algorithm 8.3.1, which possesses a better worst-case complexity bound, which is $O(\frac{2\hat{c}_A r_0^2}{\varepsilon})$, where ε is a given accuracy, $\hat{c}_A$ is a constant related to a local norm of matrix A and r_0 is the distance from the initial point to a solution. Both algorithms can be implemented in a *parallel manner*.

Let us emphasize the following points that relate to the contribution of this chapter. First, the estimates between the original dual function and the smoothed dual function have not been studied yet in [131, 133, 138, 171, 218]. Second, our methods also work with nonsmooth convex optimization problems and general local convex constraints which are not necessarily endowed with a self-concordant barrier, see Section 8.4 and Remark 8.1.1. Third, the worst-case complexity in most cases depends polynomially (of maximum order 2) on the dimension of the problems and, particularly, on the number of components, see e.g. Remark 8.2.3. Finally, we can solve the primal subproblem by transforming it into a generalized equation. The last problem can be solved, e.g. by means of Newton-type methods [29].

Outline of Chapter 8. This chapter consists of the following sections. In the next section, we present a smoothing technique via self-concordant barrier functions and provide a local estimate and global estimates between the original dual function and the smoothed dual function. Section 8.2 presents a path-following gradient-based algorithm for solving the smoothed dual problem. The convergence of this algorithm is investigated and the local convergence rate is estimated. Sections 8.3 deals with a fast gradient scheme of the dual decomposition and proves its worst-case complexity bound. Section 8.4 presents numerical examples. We end this chapter with some conclusion.

8.1 Smoothing via self-concordant barrier functions

For our presentation convenience, we prefer to consider in this section the maximization problem ($\text{SepCOP}_{\max}$) instead of the minimization problem (SepCOP). Let us recall this problem here for further reference:

$$\phi^* := \left\{ \begin{array}{ll} \max\limits_{x\in\mathbf{R}^n} & \phi(x) := \sum_{i=1}^M \phi_i(x_i) \\ \text{s.t.} & \sum_{i=1}^M (A_i x_i - b_i) = 0, \\ & x_i \in X_i,\ i = 1, \cdots, M. \end{array} \right. \tag{$\text{SepCOP}_{\max}$}$$

Here, ϕ_i, A_i, b_i and X_i are defined as before for $i = 1, \cdots, M$. The corresponding dual problem is defined as:

$$g^* := \min_{y \in \mathbf{R}^m} g(y), \tag{8.1.1}$$

where the dual function g is defined by:

$$g(y) := \max_{x \in \mathbf{R}^n} \Big\{ \sum_{i=1}^{M} [\phi_i(x_i) + (A_i x_i - b_i)^T y] \mid x_i \in X_i,\ i = 1, \cdots, M \Big\}. \tag{8.1.2}$$

Self-concordant functions and self-concordant barriers

In this subsection, we recall some definitions and properties of self-concordant functions and self-concordant barriers along the lines of [142, 146].

Let us consider a closed convex function $f \in \mathcal{C}^3(\mathrm{dom}(f))$ with open domain. Let us fix some $x \in \mathrm{dom}(f)$ and a direction $u \in \mathbf{R}^n$ and consider the function $\varphi(x;t) := f(x + tu)$ as a function of variable $t \in \mathrm{dom}(\varphi(x;\cdot)) \subseteq \mathbf{R}$. We denote by:

$$\mathbb{D}f(x)[u] := \varphi'(x;t) = \nabla f(x)^T u,$$

$$\mathbb{D}^2 f(x)[u,u] := \varphi''(x;t) = u^T \nabla f^2(x) u = \|u\|^2_{\nabla f^2(x)},$$

$$\mathbb{D}^3 f(x)[u,u,u] := \varphi'''(x;t) = (\mathbb{D}^3 f(x)[u]u)^T u.$$

Definition 8.1.1. *We call the function f self-concordant if there exists a constant $\kappa_f \geq 0$ such that:*

$$\mathbb{D}^3 f(x)[u,u,u] \leq \kappa_f \|u\|^{3/2}_{\nabla^2 f(x)}, \quad \forall x \in \mathrm{dom}(f), \quad u \in \mathbf{R}^n.$$

We call f a standard self-concordant function if f is self-concordant with $\kappa_f = 2$. Let F be a standard self-concordant function. We call it ν-self-concordant barrier for the set $\mathrm{Dom}(F)$ *if:*

$$\sup_{u \in \mathbf{R}^n} [2\nabla F(x)^T u - u^T \nabla^2 F(x) u] \leq \nu,$$

for all $x \in \mathrm{dom}(F)$, where $\mathrm{Dom}(F) := \overline{\mathrm{dom}(F)}$. *The value ν is called the parameter of the barrier F.*

For a given standard self-concordant function f and for a fixed $x \in \mathrm{dom}(f)$, we define the local norm $\|\cdot\|_x$ and its dual norm as:

$$\|u\|_x := \left[u^T \nabla^2 f(x) u\right]^{1/2}, \quad \|u\|^*_x := \sup_{\|v\|_x \leq 1} u^T v = \left[u^T \nabla^2 f(x)^{-1} u\right]^{1/2}, \tag{8.1.3}$$

for any vector $u \in \mathbf{R}^n$. We also recall two functions $\omega : \mathbf{R}_+ \to \mathbf{R}_+$ as $\omega(t) := t - \ln(1+t)$ and $\omega^* : [0,1) \to \mathbf{R}_+$ as $\omega^*(t) := -t - \ln(1-t)$. These functions are convex, nonnegative and nondecreasing as illustrated in Figure 8.1. Further properties and estimates of a self-concordant function and a self-

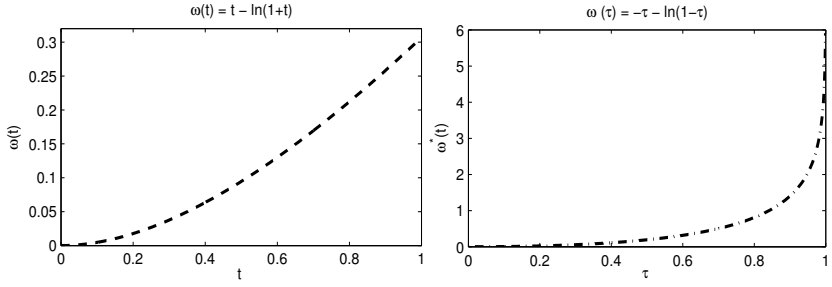


Figure 8.1: The appearance of the functions ω and ω^*

concordant barrier can be found in [98, 142, 146, 159]. Throughout Part II we will use the local norms defined in (8.1.3) and these two functions without recalling them.

Let us assume that the feasible set X_i possesses a ν_i-self-concordant barrier F_i for $i = 1, \cdots, M$. More precisely, we make the following assumption:

Asumption A. 8.1.10. *For each $i \in \{1, \cdots, M\}$, the feasible set X_i of* (SepCOP$_{\max}$) *is bounded in $\mathbf{R}^{n_i}$ with int$(X_i) \neq \emptyset$ and possesses a self-concordant barrier F_i with a parameter $\nu_i > 0$.*

Note that if X_i is polyhedral, ellipsoidal or formed by a set of linear matrix inequalities (LMIs) and does not contain any straight line (the infinite line) then Assumption **A.8.1.10** is satisfied. Moreover, the assumption on the boundedness of X_i is not restrictive as explained in Remark 7.1.1.

Remark 8.1.1. *The theory developed in this paper can be easily extended to the case X_i given as follows:*

$$X_i := X_i^c \cap X_i^a, \quad X_i^a := \{x_i \in \mathbf{R}^{n_i} \mid D_i x_i = d_i\}, \tag{8.1.4}$$

by applying standard linear algebra routines, where the set X_i^c has nonempty interior and is endowed with a ν_i-self-concordant barrier F_i for $i \in \{1, \cdots, M\}$. If, for some $i \in \{1, \cdots, M\}$, $X_i := X_i^c \cap X_i^g$, where X_i^g is a general convex set, then we can remove X_i^g from the set of constraints by adding the indicator function $\delta_{X_i^g}(\cdot)$ of this set to the objective function component ϕ_i, i.e. $\hat{\phi}_i := \phi_i + \delta_{X_i^g}$.

Let us denote by x_i^c the *analytic center* of X_i, which is defined as:

$$x_i^c := \operatorname{argmin} \{F_i(x_i) \mid x_i \in \operatorname{int}(X_i)\}, \ i = 1, \cdots, M.$$

Under Assumption **A.8.1.10**, $x^c := (x_1^c, \ldots, x_M^c)$ is well-defined due to [159, Corollary 2.3.6]. To compute x^c, one can apply the algorithms proposed in [142, pp. 204–205]. Moreover, the following estimates hold:

$$F_i(x_i) - F_i(x_i^c) \geq \omega(\|x_i - x_i^c\|_{x_i^c}) \text{ and } \|x_i - x_i^c\|_{x_i^c} \leq \nu_i + 2\sqrt{\nu_i}, \tag{8.1.5}$$

for all $x_i \in \overline{\operatorname{dom}}(F_i)$ and $i = 1, \cdots, M$, see [142, Theorems 4.1.13 and 4.2.6]. Without loss of generality, we can assume that $F_i(x_i^c) = 0$. Otherwise, we can shift F_i by $\tilde{F}_i(\cdot) := F_i(\cdot) - F_i(x_i^c)$ for $i = 1, \cdots, M$.

Smooth approximation of the dual function

Let us define the following function:

$$g(y; t) := \sum_{i=1}^{M} g_i(y; t), \tag{8.1.6}$$

where

$$g_i(y; t) := \max_{x_i \in \operatorname{int}(X_i)} \left\{\phi_i(x_i) + y^T(A_i x_i - b_i) - tF_i(x_i)\right\}, \ i = 1, \cdots, M, \tag{8.1.7}$$

where $t > 0$ is referred to as a *smoothness parameter* or *penalty parameter*. Similar to [113, 135, 171, 218], we can show that $g(\cdot; t)$ is well-defined and smooth due to strict convexity of F_i. We denote by $x_i^*(y; t)$ the unique solution of the maximization problem in (8.1.7) and $x^*(y; t) = (x_1^*(y; t), \ldots, x_M^*(y; t))$. We refer to $g(\cdot; t)$ as a *smoothed dual function* of the original dual function $g(\cdot)$ defined by (8.1.2) and to the maximization problem in (8.1.7) as *primal subproblem*. The optimality condition for the primal subproblem (8.1.7) is:

$$0 \in \partial\phi_i(x_i^*(y; t)) + A_i^T y - t\nabla F_i(x_i^*(y; t)), \ i = 1, \cdots, M \tag{8.1.8}$$

where $\partial\phi_i(x_i^*(y; t))$ is the super-differential of ϕ_i at $x_i^*(y; t)$. Since problem (8.1.7) is unconstrained and convex, this condition is necessary and sufficient. Moreover, the condition (8.1.8) is in fact a system of *generalized equations*. When ϕ_i is differentiable for $i = 1, \cdots, M$, it reduces to a system of nonlinear equations.

Associated with $g(\cdot; t)$, we consider the following *smoothed dual problem* (or *master problem*):

$$g^*(t) := \min_{y \in Y} g(y; t). \tag{8.1.9}$$

We denote by $y^*(t)$ a solution of (8.1.9) if it exists and by $x^*(t) := x^*(y^*(t);t)$. Before presenting the algorithms for solving the smoothed dual problem (8.1.9), we provide some estimates to see the connection between the dual function $g(\cdot)$ and the smoothed dual function $g(\cdot;t)$.

Local and global estimates of the dual function

This subsection provides a local estimate and global estimates between $g(\cdot)$ and $g(\cdot;t)$. Moreover, we will show that $g(y;t) \to g(y)$ as $t \to 0^+$ for any $y \in \mathbf{R}^m$.

Let $F(x) := \sum_{i=1}^M F_i(x_i)$. Then the function F is also a self-concordant barrier of X with a parameter $\nu := \sum_{i=1}^M \nu_i$. Let:

$$\lambda_{F_i}(x_i^*(y;t)) := \|\nabla F_i(x_i^*(y;t))\|^*_{x_i^*(y;t)}.$$

For a given $\beta \in (0,1)$, we define a neighbourhood in $\mathbf{R}^m$ w.r.t. F and $t > 0$ as:

$$\mathcal{N}_t^F(\beta) := \{y \in \mathbf{R}^m \mid \lambda_{F_i}(x_i^*(y;t)) \le \beta,\ i = 1, \cdots, M\}.$$

Since $x^c \in \mathcal{N}_t^F(\beta)$, we see that if $\partial\phi(x^c) \cap \mathrm{range}A^T \neq \emptyset$ then $\mathcal{N}_t^F(\beta)$ is nonempty. Let $\underline{\omega}(x^*(y;t)) := \sum_{i=1}^M \omega\left(\|x_i^*(y;t) - x_i^c\|_{x_i^c}\right)$ and $\bar{\omega}(x^*(y;t)) := \sum_{i=1}^M \nu_i\omega^{-1}(\nu_i^{-1}\omega_*(\lambda_{F_i}(x_i^*(y;t))))$. The following lemma provides a local estimate for the original dual function g.

Lemma 8.1.1. *Suppose that Assumptions **A.6.1.7** and **A.8.1.10** are satisfied. Suppose further that $\partial\phi(x^c) \bigcap \mathrm{range}A^T \neq \emptyset$. Then for any $\beta \in (0,1)$, the function $g(\cdot;t)$ defined by (8.1.6) satisfies:*

$$0 \le t\underline{\omega}(x^*(y;t)) \le g(y) - g(y;t) \le t\left[\bar{\omega}(x^*(y;t)) + \nu\right], \tag{8.1.10}$$

for all $y \in \mathcal{N}_t^F(\beta)$. Consequently, one has:

$$0 \le g(y) - g(y;t) \le t\left[\bar{\omega}_\beta + \nu\right], \quad \forall y \in \mathcal{N}_t^F(\beta),$$

where $\bar{\omega}_\beta := \sum_{i=1}^M \nu_i\omega^{-1}(\nu_i^{-1}\omega_*(\beta))$ and ω^{-1} is the inverse function of ω.

Proof. For notational simplicity, we denote by $x_i^* := x_i^*(y;t)$. The left-hand side of (8.1.10) follows from $F_i(x_i) - F_i(x_i^c) \ge \omega(\|x_i - x_i^c\|_{x_i^c}) \ge 0$ due to (8.1.5). We prove the right-hand side of (8.1.10). Since F_i is standard self-concordant, $x_i^c = \arg\min_{x_i \in \mathrm{int}(X_i)} F_i(x_i)$ and $F_i(x_i^c) = 0$, according to [142, Theorem 4.1.13], on one hand, we have:

$$F_i(x_i^*) = F_i(x_i^*) - F_i(x_i^c) \le \omega_*(\lambda_{F_i}(x_i^*)). \tag{8.1.11}$$

provided that $\lambda_{F_i}(x_i^*) < 1$. On the other hand, let $x_i(\alpha) := x_i^* + \alpha(x_{0i}^*(y) - x_i^*)$ for $\alpha \in [0,1)$. Since $x_i^* \in \operatorname{int}(X_i)$ and $\alpha < 1$, $x_i(\alpha) \in \operatorname{int}(X_i)$. By applying [146, inequality 2.3.3], we have $F_i(x_i(\alpha)) \le F_i(x_i^*) - \nu_i \ln(1-\alpha)$ which is equivalent to:

$$F_i(x_i(\alpha)) \le F_i(x_i^*) - \nu_i \ln(1-\alpha). \tag{8.1.12}$$

From the definition of $g_i(\cdot;t)$ and $g_i(\cdot)$, the concavity of ϕ_i, (8.1.11) and (8.1.12) we have:

$$\begin{aligned} g_i(y;t) &\ge \max_{\alpha\in[0,1)} \left\{\phi_i(x_i(\alpha)) + y^T(A_i x_i(\alpha) - b_i) - tF_i(x_i(\alpha))\right\} \\ &\ge \max_{\alpha\in[0,1)} \Big\{\alpha[\phi_i(x_{0i}^*(y)) + y^T(A_i x_{0i}^*(y) - b_i)] + (1-\alpha)[\phi_i(x_i^*) \\ &\quad + y^T(A_i x_i^* - b_i)] - tF_i(x_i^*) + \nu_i t\ln(1-\alpha)\Big\} \qquad (8.1.13) \\ &\overset{(8.1.11)}{\ge} \max_{\alpha\in[0,1)} \Big\{\alpha g_i(y) + (1-\alpha)g_i(y;t) + t\nu_i\ln(1-\alpha) - t\omega_*(\lambda_{F_i}(x_i^*))\Big\}. \end{aligned}$$

By solving the last maximization problem in (8.1.13) we obtain the solution:

$$\alpha^* := \begin{cases} 0 & \text{if } \ g_i(y) - g_i(y;t) \le t\nu_i, \\ 1 - [g_i(y) - g_i(y;t)]^{-1}\nu_i t & \text{otherwise.} \end{cases}$$

Substituting this solution into (8.1.13) we get:

$$g_i(y) - g_i(y;t) \le t\nu_i\Big\{1 + \ln\left[(g_i(y) - g_i(y;t))/(t\nu_i)\right] + \omega_*(\lambda_{F_i}(x_i^*))/\nu_i\Big\}, \tag{8.1.14}$$

provided that $g_i(y) - g_i(y;t) > t\nu_i$. By rearranging (8.1.14) we obtain $g_i(y) - g_i(y;t) \le t\nu_i\big(1 + \omega^{-1}(\omega_*(\lambda_{F_i}(x_i^*))/\nu_i))\big)$. Summing up the last inequalities from $i = 1$ to M we obtain the right-hand side of (8.1.10). □

Remark 8.1.2. *Lemma 8.1.1 implies that, for a given $\varepsilon_g > 0$, if we choose $t_f := (\bar{\omega}_\beta + \nu)^{-1}\varepsilon_g$, then $g(y;t_f) \le g(y) \le g(y;t_f) + \varepsilon_g$ for all $y \in \mathcal{N}_t^F(\beta)$. The last inequalities show that $g(\cdot;t)$ is a local approximation of g in $\mathcal{N}_t^F(\beta)$.*

Next, we provide a global approximation for g. Under Assumption **A.6.1.7**, the solution set Y^* of the dual problem (8.1.1) is bounded. Let Y be a compact set in $\mathbf{R}^m$ such that $Y^* \subseteq Y$ and $x_i^c \in \operatorname{ri}(\operatorname{dom}(\phi_i))$ for $i = 1, \cdots, M$. We define:

$$K_i := \max_{y\in Y} \max_{\xi_i \in \partial\phi_i(x_i^c)} \left\{ \left\|\xi_i + A_i^T y\right\|_{x_i^c}^* \right\} \in [0, +\infty), \ i = 1, \cdots, M. \tag{8.1.15}$$

The following lemma provides a global estimate of the dual function g.

Lemma 8.1.2. *Suppose that Assumptions* ***A.6.1.7*** *and* ***A.8.1.10*** *are satisfied and the constants* K_i, $i = 1, \cdots, M$, *are defined by* (8.1.15). *Then, for any* $t > 0$, *we have:*

$$t\underline{\omega}(x^*(y;t)) \le g(y) - g(y;t) \le tD_X(t), \quad \forall y \in Y, \tag{8.1.16}$$

where $D_X(t) := \sum_{i=1}^M \bar{\zeta}(K_i; \nu_i, t)$ *and* $\bar{\zeta}(\tau; a, b) := a\left(1 + \max\left\{0, \ln\left(\frac{\tau}{ab}\right)\right\}\right)$.

Consequently, for a given tolerance $\varepsilon_g > 0$ *and a constant* $\kappa \in (0,1)$ (*e.g.* $\kappa = 0.5$), *if:*

$$0 < t \le \bar{t} := \min_{1 \le i \le M} \left\{ \frac{K_i}{\nu_i}\kappa^{1/\kappa}, \left(\frac{\varepsilon_g}{\sum_{i=1}^M [\nu_i + \nu_i^{1-\kappa} K_i^\kappa]}\right)^{1/(1-\kappa)} \right\}, \tag{8.1.17}$$

then $g(y;t) \le g(y) \le g(y;t) + \varepsilon_g$ *for all* $y \in Y$.

Proof. The first inequality in (8.1.16) was proved in Lemma 8.1.1. We now prove the second one. Let us denote by $x_i^\tau(y) := x_i^c + \tau(x_{0i}^*(y) - x_i^c)$, where $\tau \in [0,1]$ and $g_i^c(y) := \phi_i(x_i^c) + y^T(A_i x_i^c - b_i)$. Since F_i is ν_i-self-concordant and $F_i(x_i^c) = 0$, it follows from [146, inequality (2.3.3)] that:

$$F_i(x_i^\tau(y)) \le F_i(x_i^c) - \nu_i \ln(1-\tau) = -\nu_i \ln(1-\tau), \ \tau \in [0,1).$$

Combining this inequality and the concavity of ϕ_i and then using the definitions of g_i^c and $g_i(\cdot)$ we have:

$$\begin{aligned} g_i(y;t) &\ge \max_{\tau \in [0,1)} \left\{\phi_i(x_i^\tau(y)) + y^T A_i(x_i^\tau(y) - b_i) - tF_i(x_i^\tau(y))\right\} \\ &\ge \max\Big\{(1-\tau)[\phi_i(x_i^c) + y^T(A_i x_i^c - b_i)] + \tau[\phi_i(x_{0i}^*(y)) \\ &\quad + y^T(A_i x_{0i}^*(y) - b_i)] + t\nu_i \ln(1-\tau) \mid \tau \in [0,1)\Big\} \\ &= \max_{\tau \in [0,1)} \left\{(1-\tau)g_i^c(y) + \tau g_i(y) + t\nu_i \ln(1-\tau)\right\}. \end{aligned} \tag{8.1.18}$$

Now, we maximize the function $\xi(\tau) := (1-\tau)g_i^c(y) + \tau g_i(y) + t\nu_i \ln(1-\tau)$ in last line of (8.1.18) w.r.t. $\tau \in [0,1)$ to obtain $\tau^* = \left[1 - \frac{t\nu_i}{g_i(y) - g_i^c(y)}\right]_+$, where $[a]_+ := \max\{0, a\}$. Therefore, if $g_i(y) - g_i^c(y) \le t\nu_i$ then $\tau^* = 0$. Otherwise, by substituting τ^* into the last line of (8.1.18), we obtain:

$$g_i(y) \le g_i(y;t) + t\nu_i\Big(1 + \left[\ln((t\nu_i)^{-1}(g_i(y) - g_i^c(y)))\right]_+\Big). \tag{8.1.19}$$

Furthermore, we note that $g_i(y)-g_i^c(y) = \max_{x_i\in X_i}\{\phi_i(x_i)+y^T(A_ix_i-b_i)\}-\big[\phi_i(x_i^c)+y^T(A_ix_i^c-b_i)\big] \geq 0$ for all $y\in Y$ and

$$\begin{aligned} g_i(y)-g_i^c(y) &\overset{\phi \text{ is concave}}{\leq} \max_{x_i\in X_i}\left\{\max_{\xi_i\in\partial\phi_i(x_i^c)}\left\{\left[\xi_i+A_i^Ty\right]^T(x_i-x_i^c)\right\}\right\} \\ &\leq \max_{x_i\in X_i}\left\{\max_{\xi_i\in\partial\phi_i(x_i^c)}\left\{\left\|\xi_i+A_i^Ty\right\|_{x_i^c}^*\|x_i-x_i^c\|_{x_i^c}\right\}\right\} \\ &\overset{(8.1.5)}{\leq} (\nu_i+2\sqrt{\nu_i})\max_{\xi_i\in\partial\phi_i(x_i^c)}\left\{\left\|\xi_i+A_i^Ty\right\|_{x_i^c}^*\right\} \\ &\leq K_i<+\infty,\ \forall y\in Y. \end{aligned} \tag{8.1.20}$$

Summing up the inequalities (8.1.19) for $i=1,\cdots,M$ and then using (8.1.20) we get (8.1.16).

Finally, for fixed $\kappa\in(0,1)$, since $\ln(x^{-1})\leq x^{-\kappa}$ for $0<x\leq\kappa^{1/\kappa}$, we have:

$$\nu_it\Big(1+\Big[\ln\frac{K_i}{\nu_it}\Big]_+\Big)\leq\nu_it\Big[1+\Big(\frac{K_i}{\nu_it}\Big)^\kappa\Big]\leq\big[\nu_i+K_i^\kappa\nu_i^{1-\kappa}\big]t^{1-\kappa},\quad\forall t\leq\frac{K_i}{\nu_i}\kappa^{1/\kappa}.$$

Consequently, if $t\leq\min\Big\{\frac{K_i}{\nu_i}\kappa^{1/\kappa},\Big(\frac{\varepsilon}{\sum_{i=1}^M[\nu_i+\nu_i^{1-\kappa}K_i^\kappa]}\Big)^{1/(1-\kappa)},\ i=1,\cdots,M\Big\}$ then $D_X(t)\leq\varepsilon$, where $D_X(t)$ is defined in Lemma 8.1.2. Combining this condition and (8.1.16) we get the last conclusion of Lemma 8.1.2. □

If we choose $\kappa=0.5$ in Lemma 8.1.2 then we have:

$$0<t\leq\bar{t}=\min_{1\leq i\leq M}\Big\{\frac{K_i}{4\nu_i},\Big(\frac{\varepsilon_g}{\sum_{i=1}^M[\nu_i+\sqrt{\nu_iK_i}]}\Big)^2\Big\}.$$

sets Lemma 8.1.2 shows that if we fix $t_f\in(0,\bar{t}]$ and minimize $g(\cdot,t_f)$ over Y, then the obtained solution $y^*(t_f)$ is an ε_g-solution of (8.1.1). Since $g(\cdot;t_f)$ is continuously differentiable, smooth optimization techniques such as gradient-based methods can be applied to minimize $g(\cdot;t_f)$ over Y, see Sections 8.2 and 8.3 below.

If a strictly feasible point $\bar{x}$ is available then we can also prove the following estimate which is global and independent of the boundedness of Y.

Lemma 8.1.3. *Suppose that Assumptions* ***A.6.1.7*** *and* ***A.8.1.10*** *are satisfied. Let $\bar{x}$ be a strictly feasible point to* (SepCOP$_{\max}$)*, i.e. $\bar{x}\in\operatorname{int}(X)\cap\{x\mid Ax=b\}$. Then, for any $t>0$, we have:*

$$g(y)-\phi(\bar{x})\geq 0\quad\textit{and}\quad g(y;t)-\phi(\bar{x})+tF(\bar{x})\geq 0. \tag{8.1.21}$$

Moreover, it holds that:

$$g(y;t) \leq g(y) \leq g(y;t) + t[\nu + F(\bar{x})] + 2\sqrt{t\nu}\,[g(y;t) + tF(\bar{x}) - \phi(\bar{x})]^{1/2}. \quad (8.1.22)$$

Proof. The two inequalities in (8.1.21) are trivial due to the definitions of $g(\cdot)$, $g(\cdot;t)$ and the feasibility of $\bar{x}$. We only prove (8.1.22). Indeed, since $\bar{x} \in \mathrm{int}(X)$ and $x^*(y) \in X$, if we define $x^*_\tau(y) = \bar{x} + \tau(x^*(y) - \bar{x})$ then $x_\tau(y) \in \mathrm{int}(X)$ if $\tau \in [0,1)$. By applying the inequality [146, p. 2.3.3] we have: $F(x_\tau(y)) \leq F(\bar{x}) - \nu \ln(1-\tau)$. Using this inequality together with the definition of $g(\cdot;t)$, the concavity of ϕ and $A\bar{x} = b$, we deduce:

$$\begin{aligned} g(y;t) &= \max_{x \in \mathrm{int}(X)} \left\{\phi(x) + y^T(Ax - b) - tF(x)\right\} \\ &\geq \max_{\tau \in [0,1)} \left\{\phi(x_\tau(y)) + y^T(Ax_\tau(y) - b) - tF(x_\tau(y))\right\} \quad (8.1.23) \\ &\geq \max_{\tau \in [0,1)} \left\{(1-\tau)\phi(\bar{x}) + \tau g(y) + t\nu \ln(1-\tau)\right\} - tF(\bar{x}). \end{aligned}$$

By solving the maximization problem on the right hand side of (8.1.23) and then rearranging the results, we obtain:

$$g(y) \leq g(y;t) + t\nu \left(1 + \left[\ln\left(\frac{g(y) - \phi(\bar{x})}{t\nu}\right)\right]_+\right) + tF(\bar{x}), \quad (8.1.24)$$

where $[\cdot]_+ = \max\{\cdot, 0\}$.

Moreover, it follows from (8.1.23) that:

$$\begin{aligned} g(y) - \phi(\bar{x}) &\leq \frac{1}{\tau}\left[g(y;t) - \phi(\bar{x}) + tF(\bar{x}) + t\nu \ln\left(1 + \frac{\tau}{1-\tau}\right)\right] \\ &\leq \frac{1}{\tau}\left[g(y;t) - \phi(\bar{x}) + tF(\bar{x})\right] + \frac{t\nu}{1-\tau}. \end{aligned}$$

If we minimize the right hand side of this inequality in $[0,1)$ then we get $g(y) - \phi(\bar{x}) \leq [(g(y;t) - \phi(\bar{x}) + tF(\bar{x}))^{1/2} + \sqrt{t\nu}]^2$. Finally, we plug this inequality into (8.1.24) to obtain:

$$\begin{aligned} g(y) &\leq g(y;t) + t\nu \left[1 + 2\ln\left(1 + ([g(y;t) - \phi(\bar{x}) + tF(\bar{x}]/(t\nu))^{1/2}\right)\right] + tF(\bar{x}) \\ &\leq g(y;t) + t\nu + tF(\bar{x}) + 2\sqrt{t\nu}\,[g(y;t) - \phi(\bar{x}) + tF(\bar{x})]^{1/2} \end{aligned}$$

which is indeed (8.1.22). □

It follows from (8.1.22) that $g(y) \leq (1+2\sqrt{t\nu})g(y;t)+t(\nu+F(\bar{x}))+2\sqrt{t\nu}(tF(\bar{x})-\phi(\bar{x}))$. Hence, $g(y;t) \to g(y)$ as $t \to 0^+$.

Finally, we prove some properties of $g(y;\cdot)$ and $g^*(\cdot)$ defined by (8.1.9) which will be used in the sequel. We notice that in Lemmas 8.1.1, 8.1.2 and 8.1.3 we do not impose any additional assumption on the objective function except the concavity of the objective functions in Assumption **A.6.1.7**.

Lemma 8.1.4. *Suppose that Assumptions **A.6.1.7** and **A.8.1.10** are satisfied. Then:*

(a) *The function $g(y;\cdot)$ is convex and non-increasing in $\mathbf{R}_{++}$ for a given $y \in \mathbf{R}^m$. Moreover, we have:*

$$g(y;\hat{t}) \geq g(y;t) - (\hat{t}-t)F(x^*(y;t)). \tag{8.1.25}$$

(b) *The function $g^*(\cdot)$ defined by (8.1.9) is differentiable and non-increasing in $\mathbf{R}_{++}$. Moreover, $g^*(t) \leq g^*$ and $\lim_{t\downarrow 0^+} g^*(t) = g^* = \phi^*$.*

(c) *The point $x^*(y^*(t);t)$ is feasible to the original problem (SepCOP$_{\max}$) and $\lim_{t\downarrow 0^+} x^*(y^*(t);t) = x^* \in X^*$.*

Proof. Since the function $\xi(x,y,t) := \phi(x) + y^T(Ax-b) - tF(x)$ is strictly concave in x and linear in t, it is well-known that $g(y;t) = \max\{\xi(x,y,t) \mid x \in \mathrm{int}(X)\}$ is differentiable w.r.t. t and its derivative is given by $\frac{\partial g(y;t)}{\partial t} = -F(x^*(y,t)) \leq -\omega(\|x^*(y,t)-x^c\|_{x^c}) \leq 0$ due to (8.1.5). Thus $g(y;\cdot)$ is nonincreasing in t as stated in a). Since $g(y;\cdot)$ is convex and differentiable, and $\frac{dg(y;t)}{dt} = -F(x^*(y;t)) \leq 0$, we have $g(y;\hat{t}) \geq g(y;t) + (\hat{t}-t)\frac{dg(y;t)}{dt} = g(y;t) - (\hat{t}-t)F(x^*(y;t))$.

From the definitions of $g^*(\cdot)$, $g(y,\cdot)$ and $y^*(\cdot)$ in (8.1.9), and strong duality we have:

$$\begin{aligned} g^*(t) &= \min_{y\in Y} g(y;t) \overset{\text{strong duality}}{=} \max_{x\in \mathrm{int}(X)} \min_{y\in Y} \left\{\phi(x)+y^T(Ax-b)-tF(x)\right\} \\ &= \max_{x\in \mathrm{int}(X)} \{\phi(x) - tF(x) \mid Ax = b\} \\ &= \phi(x^*(t)) - tF(x^*(t)). \end{aligned} \tag{8.1.26}$$

It follows from the second line of (8.1.26) that $g^*(\cdot)$ is differentiable and nonincreasing in $\mathbb{R}_{++}$. From the second line of (8.1.26), we also deduce that $x^*(t)$ is feasible to (SepCOP$_{\max}$). The limit in c) was proved in [218, Proposition 2]. Since $x^*(t)$ is feasible to (SepCOP$_{\max}$) and $F(x^*(t))-F(x^c) \geq 0$,

the last line of (8.1.26) implies that $g^*(t) \leq g^*$. We also obtain the limit $\lim_{t\downarrow 0^+} g^*(t) = g^* = \phi^*$. □

8.2 Path-following gradient-based decomposition algorithm

This section aims at designing a path-following gradient-based algorithm to solve the dual problem (8.1.1), analyzing the convergence of the algorithm and estimating the local convergence rate.

Since F is a barrier function, $g(\cdot;t)$ is strictly convex and smooth, so that we can write the optimality condition of (8.1.9) as:

$$\nabla g(y;t) = 0. \tag{8.2.1}$$

This equation has a unique solution $y^*(t)$. Moreover, the gradient of $g(\cdot;t)$ is given as:

$$\nabla g(y;t) := Ax^*(y;t) - b. \tag{8.2.2}$$

where $x^*(y;t)$ is the solution of the primal subproblem (8.1.7).

Local Lipschitz-type continuity of the gradient mapping

Since F is a self-concordant barrier, $\nabla^2 F(x) \succ 0$ for $x \in \operatorname{int}(X)$, we can define a matrix norm $\|A\|_x^* := \left\|A\nabla^2 F(x)^{-1}A^T\right\|_2$ for any matrix $A \in \mathbf{R}^{m\times n}$. We prove the following property for the function $g(\cdot;t)$.

Lemma 8.2.1. *Suppose that Assumptions* ***A.6.1.7*** *and* ***A.8.1.10*** *are satisfied. Then, for all $t > 0$ and $y, \hat{y} \in \mathbf{R}^m$, one has:*

$$[\nabla g(y;t) - \nabla g(\hat{y};t)]^T(y - \hat{y}) \geq \frac{t\,\|\nabla g(y;t) - \nabla g(\hat{y};t)\|_2^2}{c_A\,[c_A + \|\nabla g(y,t) - \nabla g(\hat{y};t)\|_2]}, \tag{8.2.3}$$

where $c_A := \|A\|^*_{x^*(y;t)}$. Consequently, it holds that:

$$g(\hat{y};t) \leq g(y;t) + \nabla g(y;t)^T(\hat{y} - y) + t\omega^*\Big(\frac{c_A\,\|\hat{y} - y\|_2}{t}\Big), \tag{8.2.4}$$

provided that $\|\hat{y} - y\|_2 < \frac{t}{c_A}$.

Proof. For notational simplicity, we denote by $x^* := x^*(y;t)$ and $\hat{x}^* := x^*(\hat{y};t)$. From the definition (8.2.2) of $\nabla g(\cdot;t)$ and the Cauchy-Schwarz inequality we have:

$$[\nabla g(y;t) - \nabla g(\hat{y};t)]^T(y-\hat{y}) = (y-\hat{y})^T A(x^* - \hat{x}^*). \tag{8.2.5}$$

$$\|\nabla g(\hat{y};t) - \nabla g(y;t)\|_2 \leq \|A\|_{x^*}^* \, \|\hat{x}^* - x^*\|_{x^*} . \tag{8.2.6}$$

It follows from the optimality condition (8.1.8) that:

$$A^T(y-\hat{y}) = t[\nabla F(x^*) - \nabla F(\hat{x}^*] - [\xi(x^*) - \xi(\hat{x}^*)],$$

where $\xi(\cdot) \in \partial\phi(\cdot)$. By multiplying this relation by $x^* - \hat{x}^*$ and then using [142, Theorem 4.1.7] and the concavity of ϕ we obtain:

$$\begin{aligned}
(y-\hat{y})^T A(x^* - \hat{x}^*) &= t[\nabla F(x^*) - \nabla F(\hat{x}^*)]^T(x^* - \hat{x}^*) - [\xi(x^*) - \xi(\hat{x}^*)]^T(x^* - \hat{x}^*) \\
&\overset{\phi-\text{concave}}{\geq} t[\nabla F(x^*) - \nabla F(\hat{x}^*)]^T(x^* - \hat{x}^*) \\
&\geq \frac{t\,\|x^* - \hat{x}^*\|_{x^*}^2}{1 + \|x^* - \hat{x}^*\|_{x^*}} \\
&\overset{(8.2.6)}{\geq} \frac{t\,[\|\nabla g(y;t) - \nabla g(\hat{y};t)\|_2]^2}{\|A\|_{x^*}^* \, [\|A\|_{x^*}^* + \|\nabla g(y;t) - \nabla g(\hat{y};t)\|_2]}.
\end{aligned}$$

Substituting this inequality into (8.2.5) we obtain (8.2.3).

By the Cauchy-Schwarz inequality, it follows from (8.2.3) that:

$$\|\nabla g(\hat{y};t) - \nabla g(y;t)\|_2 \leq \frac{c_A^2 \, \|\hat{y} - y\|_2}{t - c_A \, \|\hat{y} - y\|_2}, \tag{8.2.7}$$

provided that $\|\hat{y} - y\|_2 \leq t/c_A$. Finally, by using the mean-value theorem, we have:

$$\begin{aligned}
g(\hat{y};t) &= g(y;t) + \nabla g(y;t)^T(\hat{y} - y) \\
&\quad + \int_0^1 (\nabla g(y + s(\hat{y} - y);t) - \nabla g(y;t))^T(\hat{y} - y)ds \\
&\overset{(8.2.7)}{\leq} g(y;t) + \nabla g(y;t)^T(\hat{y} - y) + c_A \, \|\hat{y} - y\|_2 \int_0^1 \frac{c_A s \, \|\hat{y} - y\|_2}{t - c_A s \, \|\hat{y} - y\|_2} ds \\
&= g(y;t) + \nabla g(y;t)^T(\hat{y} - y) + t\omega^*(c_A \, \|\hat{y} - y\|_2 / t),
\end{aligned}$$

which is indeed (8.2.4) provided that $c_A \, \|\hat{y} - y\|_2 < t$. □

Path-following gradient step

In this subsection, we analyze one step of the path-following gradient scheme in order to derive an algorithm for solving (8.1.9). Indeed, let $y \in \mathbf{R}^m$ and $t > 0$ be the values at the current iteration. We compute the new values y_+ and t_+ for the next iteration as:

$$\begin{cases} t_+ := t - \Delta t, \\ y_+ := y - \alpha \nabla g(y, t_+), \end{cases} \tag{8.2.8}$$

where $\alpha := \alpha(y; t) > 0$ is the current step size and Δt is the decrement of the parameter t. Let us define the following notation:

$$x_1^* := x^*(y; t_+), \quad c_{A1} = \|A\|^*_{x^*(y;t_+)} \text{ and } \lambda_1 := \|\nabla g(y; t_+)\|_2 . \tag{8.2.9}$$

First, we prove an important property of the scheme (8.2.8).

Lemma 8.2.2. *Under Assumptions **A.6.1.7** and **A.8.1.10**, the following inequality holds:*

$$g(y_+; t_+) \leq g(y; t) - \left[\alpha \lambda_1^2 - t_+ \omega^*(c_{A1} \alpha \lambda_1 / t_+) - \Delta t F(x_1^*)\right]. \tag{8.2.10}$$

Proof. Since $t_+ = t - \Delta t$, by using (8.1.25) in Lemma 8.1.4 with t and t_+, we have:

$$g(y; t_+) \leq g(y; t) + \Delta t F(x^*(y; t_+)). \tag{8.2.11}$$

Next, by (8.2.4) we have $y_+ - y = -\alpha \nabla g(y; t_+)$ and $\lambda_1 := \|\nabla g(y; t_+)\|_2$. Hence, we can derive:

$$g(y_+; t_+) \leq g(y; t_+) - \alpha \lambda_1^2 + t_+ \omega^*(c_{A1} \alpha \lambda_1 / t_+). \tag{8.2.12}$$

By plugging (8.2.11) into (8.2.12), we obtain (8.2.10). □

Lemma 8.2.3. *For any $y \in \mathbf{R}^m$ and $t > 0$, the constant $c_A := \|A\|^*_{x^*(y;t_+)}$ is bounded. More precisely, $c_A \leq \bar{c}_A := \kappa \|A\|^*_{x^c} < +\infty$. Furthermore, $\lambda := \|\nabla g(y; t)\|_2$ is also bounded, i.e.: $\lambda \leq \bar{\lambda} := \kappa \|A\|^*_{x^c} + \|Ax^c - b\|_2$, where $\kappa := \sum_{i=1}^M [\nu_i + 2\sqrt{\nu_i}]$.*

Proof. For any $x \in \mathrm{int}(X)$, from the definition of $\|\cdot\|^*_x$, we can write:

$$\begin{aligned} \|A\|^*_x &= \sup \left\{ [v^T A \nabla^2 F(x)^{-1} A^T v]^{1/2} \mid \|v\|_2 = 1 \right\} \\ &= \sup \left\{ \|u\|^*_x \mid u = A^T v, \ \|v\|_2 = 1 \right\}. \end{aligned}$$

By using [142, Corollary 4.2.1], we can estimate $\|\|A\|\|_x^*$ as:

$$\begin{aligned}\|\|A\|\|_x^* &\leq \sup\left\{\kappa \|u\|_{x^c}^* \mid u = A^T v,\ \|v\|_2 = 1\right\}\\ &= \kappa \sup\left\{\left[v^T A \nabla^2 F(x^c)^{-1} A^T v\right]^{1/2} \mid \|v\|_2 = 1\right\}\\ &= \kappa \|\|A\|\|_{x^c}^*.\end{aligned}$$

By substituting $x = x^*(y;t)$ into the above inequality, we obtain the first conclusion. In order to prove the second bound, we note that $\nabla g(y;t) = Ax^*(y;t) - b$. Therefore, by using (8.1.5), we can estimate:

$$\begin{aligned}\|\nabla g(y;t)\|_2 &= \|Ax^*(y;t) - b\|_2 \leq \|A(x^*(y;t) - x^c)\|_2 + \|Ax^c - b\|_2\\ &\leq \|\|A\|\|_{x^c}^* \|x^*(y;t) - x^c\|_{x^c} + \|Ax^c - b\|_2\\ &\overset{(8.1.5)}{\leq} \kappa \|\|A\|\|_{x^c}^* + \|Ax^c - b\|_2,\end{aligned}$$

which is the second conclusion. □

Next, we show how to choose the step size α and the decrement Δt such that $g(y_+;t_+) < g(y;t)$ in Lemma 8.2.2. We note that $x^*(y;t_+)$ is obtained by solving the primal subproblem (8.1.7) and the quantity $c_F := F(x^*(y;t_+))$ is nonnegative (since $F(x^*(y;t_+)) \geq F(x^c) = 0$) and computable. By Lemma 8.2.3, we see that:

$$\alpha(t) := \frac{t}{c_{A1}(c_{A1} + \lambda_1)} \geq \underline{\alpha}_0(t) := \frac{t}{\bar{c}_A(\bar{c}_A + \bar{\lambda})}, \tag{8.2.13}$$

which shows that $\alpha(t)$ is bounded away from zero. We have the following estimate.

Lemma 8.2.4. *The step size $\alpha(t)$ defined by (8.2.13) satisfies:*

$$g(y_+;t_+) \leq g(y;t) - t_+\omega\left(\frac{\lambda_1}{c_{A1}}\right) + \Delta t F(x_1^*). \tag{8.2.14}$$

Proof. Let $\varphi(\alpha) := \alpha\lambda_1^2 - t_+\omega^*(c_{A1}t_+^{-1}\alpha\lambda_1) - t_+\omega(\lambda_1 c_{A1}^{-1})$. We can simplify this function as $\varphi(\alpha) = t_+[u + \ln(1-u)]$, where $u := t_+^{-1}\lambda_1^2\alpha + t_+^{-1}c_{A1}\lambda_1\alpha - c_{A1}^{-1}\lambda_1$. The function $\varphi(\alpha) \leq 0$ for all u and $\varphi(\alpha) = 0$ at $u = 0$ which leads to $\alpha := \frac{t}{c_{A1}(c_{A1}+\lambda_1)}$. □

Since $t_+ = t - \Delta t$, if we choose $\Delta t := \frac{t\omega(c_{A1}^{-1}\lambda_1)}{2[\omega(c_{A1}^{-1}\lambda_1) + F(x_1^*)]}$ then:

$$g(y_+;t_+) \leq g(y;t) - \frac{t}{2}\omega(c_{A1}^{-1}\lambda_1). \tag{8.2.15}$$

Therefore, the update rule for t can be written as:

$$t_+ := (1-\sigma)t, \text{ where } \sigma := \frac{\omega(c_{A1}^{-1}\lambda_1)}{2[\omega(c_{A1}^{-1}\lambda_1) + F(x_1^*)]} \in (0,1). \tag{8.2.16}$$

The algorithm

By combining the above analysis, we can describe in detail the path-following gradient-based decomposition algorithm as follows.

Algorithm 8.2.1.(*Path-following gradient decomposition algorithm*).
Initialization: Perform the following steps:

1. Choose an initial value $t_0 > 0$ and fix the tolerances ε_t and ε.
2. Take an initial point $y^0 \in \mathbf{R}^m$ and solve (8.1.7) *in parallel* to obtain $x_0^* := x^*(y^0; t_0)$.
3. Compute $c_A^0 := \|A\|_{x_0^*}^*$, $\lambda_0 := \left\|\nabla g(y^0; t_0)\right\|_2$, $\omega_0 := \omega(\lambda_0/c_A^0)$ and $c_F^0 := F(x_0^*)$.

Iteration: For $k = 0, 1, \cdots$, perform the following steps:

Step 1: If $t_k \le \varepsilon_t$ and $\lambda_k \le \varepsilon$ then terminate.

Step 2: Update the penalty parameter: $t_{k+1} := t_k(1-\sigma_k)$, where $\sigma_k := \frac{\omega_k}{2(\omega_k + c_F^k)}$.

Step 3: Solve (8.1.7) *in parallel* to obtain $x_k^* := x^*(y^k, t_{k+1})$. Then form a gradient vector $\nabla g(y^k; t_{k+1}) := Ax_k^* - b$.

Step 4: Compute $\lambda_{k+1} := \left\|\nabla g(y^k; t_{k+1})\right\|_2$, $c_A^{k+1} := \|A\|_{x_k^*}^*$, $\omega_{k+1} := \omega(\lambda_{k+1}/c_A^{k+1})$ and $c_F^{k+1} := F(x_k^*)$.

Step 5: Compute the step size $\alpha_{k+1} := \frac{t_{k+1}}{c_A^{k+1}(c_A^{k+1} + \lambda_{k+1})}$.

Step 6: Update y^{k+1} as $y^{k+1} := y^k - \alpha_{k+1}\nabla g(y^k, t_{k+1})$.

End.

The main step of Algorithm 8.2.1 is Step 3, where we need to solve M primal subproblems *in parallel*. From the update rule (8.2.16) of t_k we can see that $\sigma_k \to 0^+$ as $F(x_k^*) \to \infty$. This happens when the barrier function is approaching

the boundary of the feasible set X. Hence, the parameter t is not decreased. Let $\bar{c}_F$ be a sufficiently large positive constant. We can modify the update rule of t as:

$$t_{k+1} := \begin{cases} t_k(1 - \frac{\omega_k}{2(\omega_k + c_F^k)}) & \text{if } c_F^k \leq \bar{c}_F, \\ t_k & \text{otherwise,} \end{cases} \tag{8.2.17}$$

In this case, the sequence $\{t_k\}$ generated by Algorithm 8.2.1 might not converge to zero. Moreover, the step size α_k computed at Step 5 depends on the parameter t_k. If t_k is small then Algorithm 8.2.1 makes short steps towards a solution of (8.1.9).

Convergence analysis

Let us assume that $\underline{t} = \inf_{k \geq 0} t_k > 0$. Then, the following theorem shows the convergence of Algorithm 8.2.1.

Theorem 8.2.1. *Suppose that Assumptions* ***A.6.1.7*** *and* ***A.8.1.10*** *are satisfied. Suppose further that the sequence* $\{(y^k, t_k, \lambda_k)\}_{k \geq 0}$ *generated by Algorithm 8.2.1 satisfies* $\underline{t} := \inf_{k \geq 0}\{t_k\} > 0$. *Then:*

$$\lim_{k \to \infty} \left\| \nabla g(y^k, t_{k+1}) \right\|_2 = 0. \tag{8.2.18}$$

Consequently, there exists a limit point y^* *of* $\{y^k\}$ *such that* y^* *is a solution of* (8.1.9) *at* $t = \underline{t}$.

Proof. It is sufficient to prove (8.2.18). Indeed, from (8.2.15) we have:

$$\sum_{i=0}^{k} \frac{t_k}{2} \omega(\lambda_{k+1}/c_A^{k+1}) \leq g(y^0; t_0) - g(y^{k+1}; t_{k+1}) \leq g(y^0; t_0) - g^*.$$

Since $t_k \geq \underline{t} > 0$ due to assumption and $c_A^{k+1} \leq \bar{c}_A := \|\|A\|\|_{x^c}^*$ due to Lemma 8.2.3, the above inequality leads to:

$$\frac{\underline{t}}{2} \sum_{i=0}^{\infty} \omega(\lambda_{k+1}/\bar{c}_A) \leq g(y^0; t_0) - g^* < +\infty.$$

This inequality implies $\lim_{k \to \infty} \omega(\lambda_{k+1}/\bar{c}_A) = 0$ which leads to $\lim_{k \to \infty} \lambda_{k+1} = 0$. By the definition of λ_k we have $\lim_{k \to \infty} \left\| \nabla g(y^k; t_{k+1}) \right\|_2 = 0$. □

Remark 8.2.2. *From the proof of Theorem* (8.2.1)*, we can fix* $c_A^k \equiv \bar{c}_A := \kappa \|\|A\|\|_{x^c}^*$ *in Algorithm 8.2.1. This value can be computed a priori.*

Local convergence rate

Let us analyze the local convergence rate of Algorithm 8.2.1. Let y^0 be an initial point of Algorithm 8.2.1 and $y^*(t)$ be the unique solution of (8.1.9). We denote by:

$$r_0(t) := \left\|y^0 - y^*(t)\right\|_2 \text{ and } \bar{c}_A := \kappa \|\!|A|\!\|^*_{x^c}. \tag{8.2.19}$$

For simplicity of discussion, we assume that the smoothness parameter t_k is fixed at $\underline{t} > 0$ sufficiently small for all $k \geq 0$ (see Lemma 8.1.3). The convergence rate of Algorithm 8.2.1 in the case $t_k = \underline{t}$ is stated in the following lemma.

Lemma 8.2.5 (*Local convergence rate*)**.** *Suppose that the initial point y^0 is chosen such that $g(y^0;\underline{t}) - g^*(\underline{t}) \leq \bar{c}_A r_0(\underline{t})$. Then:*

$$g(y^k;\underline{t}) - g^*(\underline{t}) \leq \frac{4\bar{c}_A^2 r_0(\underline{t})^2}{4\bar{c}_A r_0(\underline{t}) + \underline{t}k}. \tag{8.2.20}$$

Consequently, the local convergence rate of Algorithm 8.2.1 is at least $O(\frac{4\bar{c}_A^2 r_0(\underline{t})^2}{\underline{t}k})$.

Proof. Let $r_k := \left\|y^k - y^*\right\|_2$, $\Delta_k := g(y^k;\underline{t}) - g^*(\underline{t}) \geq 0$, $\underline{y}^* := y^*(\underline{t})$, $\underline{\lambda}_k := \left\|\nabla g(y^k;\underline{t})\right\|_2$ and $\underline{c}_k := \|\!|A|\!\|^*_{x^*(y^k;\underline{t})}$. By using the fact that $\nabla g(\underline{y}^*;\underline{t}) = 0$ and (8.2.3) we have:

$$\begin{aligned} r_{k+1}^2 &= \left\|y^{k+1} - y^*\right\|^2 = \left\|y^k - \alpha_k \nabla g(y^k;\underline{t}) - y^*\right\|^2 \\ &= r_k^2 - 2\alpha_k \nabla g(y^k;\underline{t})^T (y^k - y^*) + \alpha_k^2 \left\|\nabla g(y^k;\underline{t})\right\|^2 \\ &\overset{(8.2.3)}{\leq} r_k^2 - 2\alpha_k \frac{\underline{t}\underline{\lambda}_k^2}{c_A^k(c_A^k + \underline{\lambda}_k)} + \alpha_k^2 \underline{\lambda}_k^2 \\ &\overset{(8.2.13)}{=} r_k^2 - \alpha_k^2 \underline{\lambda}_k^2. \end{aligned}$$

This inequality implies that $r_k \leq r_0$ for all $k \geq 0$. First, by the convexity of $g(\cdot;\underline{t})$ and the relation $r_k \leq r_0$ we have:

$$\Delta_k = g(y^k;\underline{t}) - g^*(\underline{t}) \leq \left\|\nabla g(y^k;\underline{t})\right\|_2 \left\|y^k - \underline{y}^*\right\|_2 \leq \underline{\lambda}_k \left\|y^0 - \underline{y}^*\right\|_2 \leq \underline{\lambda}_k r_0(\underline{t}).$$

This inequality implies:

$$\underline{\lambda}_k \geq \Delta_k / r_0(\underline{t}). \tag{8.2.21}$$

Since $t_k = \underline{t} > 0$ is fixed for all $k \geq 0$, it follows from (8.2.10) that:

$$g(y^{k+1};\underline{t}) \leq g(y^k;\underline{t}) - \underline{t}\omega(\underline{\lambda}_k / \underline{c}_k).$$

By using the definition of Δ_k, the last inequality is equivalent to:

$$\Delta_{k+1} \leq \Delta_k - \underline{t}\omega(\underline{\lambda}_k/\underline{c}_k). \tag{8.2.22}$$

Next, since $\omega(\tau) \geq \frac{\tau^2}{4}$ for all $0 \leq \tau \leq 1$ and $\underline{c}_k \leq \bar{c}_A$ due to Lemma 8.2.3, it follows from (8.2.21) and (8.2.22) that:

$$\Delta_{k+1} \leq \Delta_k - \frac{\underline{t}\Delta_k^2}{4r_0(\underline{t})^2\bar{c}_A^2}, \tag{8.2.23}$$

for all $\Delta_k \leq \bar{c}_A r_0(\underline{t})$. This inequality also implies $\Delta_k \leq \Delta_0$ for all $k \geq 0$.

Let $\eta := \underline{t}/(4r_0(\underline{t})^2\bar{c}_A^2)$. Since $\Delta_k \geq 0$, (8.2.23) implies:

$$\frac{1}{\Delta_{k+1}} \geq \frac{1}{\Delta_k(1-\eta\Delta_k)} = \frac{1}{\Delta_k} + \frac{\eta}{(1-\eta\Delta_k)} \geq \frac{1}{\Delta_k} + \eta.$$

By induction, this inequality leads to $\frac{1}{\Delta_k} \geq \frac{1}{\Delta_0} + \eta k$ which is equivalent to $\Delta_k \leq \frac{\Delta_0}{1+\eta\Delta_0 k}$ provided that $\Delta_0 \leq \bar{c}_A r_0(\underline{t})$. Since $\eta := \underline{t}/(4r_0(\underline{t})^2\bar{c}_A^2)$, this inequality is indeed (8.2.20). The last conclusion follows from (8.2.20). □

Remark 8.2.3. *Let us fix $\underline{t} := \varepsilon$. It follows from (8.2.20) that the worst-case complexity of Algorithm 8.2.1 to obtain an ε-solution y^k in the sense $g(y^k;\varepsilon) - g^*(\varepsilon) \leq \varepsilon$ is $O\left(\frac{\bar{c}_A^2 r_0^2}{\varepsilon^2}\right)$. We note that $\bar{c}_A = \kappa\|A\|_{x^c}^* = \sum_{i=1}^M(\nu_i + 2\sqrt{\nu_i})\|A_i\|_{x_i^c}^*$. However, in most cases, the parameter ν_i depends linearly on the dimension of the problem. Therefore, we can conclude that the worst-case complexity of Algorithm 8.2.1 is $O\left(\frac{(n\|A\|_{x^c}^* r_0)^2}{\varepsilon^2}\right)$.*

8.3 Accelerating gradient decomposition algorithm

Let us fix $t = \underline{t} > 0$ and define $\underline{g}(\cdot) := g(\cdot;\underline{t})$. The function $\underline{g}(\cdot)$ is convex and differentiable but its gradient is not Lipschitz continuous, we can not apply Nesterov's fast gradient algorithm [142] to solve (8.1.9). In this section, we modify Nesterov's fast gradient method in order to obtain an accelerating gradient method for solving (8.1.9).

One step of the *modified fast gradient method* is described as follows. Let y and v be given points in $\in \mathbf{R}^m$, we compute new points y_+ and v_+ as follows:

$$\begin{cases} y_+ := v - \alpha\nabla\underline{g}(v), \\ v_+ = \beta_1 y_+ + \beta_2 y + \beta_3 v, \end{cases} \tag{8.3.1}$$

where $\alpha > 0$ is the step size, β_1, β_2 and β_3 are three parameters which will be chosen appropriately. First, we prove the following estimate.

Lemma 8.3.1. *Let $\theta \in (0,1)$ be a given parameter, $\alpha := \frac{\underline{t}}{\hat{c}_A(\hat{c}_A+\lambda)}$ and $\rho := t/(2\theta\hat{c}_A^2)$ for some parameter $\hat{c}_A \geq c_A$, where $\lambda := \|\nabla \underline{g}(v)\|_2$ and $c_A := \|A\|^*_{x^*(v;\underline{t})}$. We define two vectors:*

$$r := \theta^{-1}[v-(1-\theta)y] \quad \text{and} \quad r_+ = r - \rho\nabla \underline{g}(v). \tag{8.3.2}$$

Then the new point y_+ generated by (8.3.1) satisfies:

$$\frac{1}{\theta^2}\left[\underline{g}(y_+)-\underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}}\left\|r_+ - \underline{y}^*\right\|_2^2 \leq \frac{(1-\theta)}{\theta^2}\left[\underline{g}(y)-\underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}}\left\|r-\underline{y}^*\right\|_2^2, \tag{8.3.3}$$

provided that $\lambda \leq \hat{c}_A$, where $\underline{y}^* := y^*(\underline{t})$ and $\underline{g}^* := \underline{g}(\underline{y}^*)$.

Proof. Since $y_+ = v - \alpha\nabla \underline{g}(v)$ with $\alpha = \frac{\underline{t}}{\hat{c}_A(\hat{c}_A+\lambda)}$, it follows from (8.2.4) that:

$$\underline{g}(y_+) \leq \underline{g}(v) - \underline{t}\,\omega\left(\frac{\|\nabla \underline{g}(v)\|_2}{\hat{c}_A}\right). \tag{8.3.4}$$

Now, since $\omega(\tau) \geq \tau^2/4$ for all $0 \leq \tau \leq 1$, the inequality (8.3.4) implies:

$$\underline{g}(y_+) \leq \underline{g}(v) - \frac{\underline{t}}{4\hat{c}_A^2}\|\nabla \underline{g}(v)\|_2^2, \tag{8.3.5}$$

provided that $\|\nabla \underline{g}(v)\|_2 \leq \hat{c}_A$. For any $u := (1-\theta)y + \theta\underline{y}^*$ and $\theta \in (0,1)$ we have:

$$\begin{aligned}\underline{g}(v) &\leq \underline{g}(u) + \nabla \underline{g}(v)^T(v-u) \leq (1-\theta)\underline{g}(y) + \theta \underline{g}(\underline{y}^*)\\ &\quad + \nabla \underline{g}(v)^T(v-(1-\theta)y-\theta\underline{y}^*).\end{aligned} \tag{8.3.6}$$

By substituting (8.3.6) and the relation $v-(1-\theta)y = \theta r$ into (8.3.5) we obtain:

$$\begin{aligned}\underline{g}(y_+) &\leq (1-\theta)\underline{g}(y) + \theta\underline{g}^* + \theta\nabla \underline{g}(v)^T(r-\underline{y}^*) - \frac{\underline{t}}{4\hat{c}_A^2}\|\nabla \underline{g}(v)\|_2^2\\ &= (1-\theta)\underline{g}(y) + \theta\underline{g}^* + \frac{\theta^2\hat{c}_A^2}{\underline{t}}\left[\|r-\underline{y}^*\|_2^2 - \left\|r - \frac{\underline{t}}{2\theta\hat{c}_A^2}\nabla \underline{g}(v) - \underline{y}^*\right\|_2^2\right]\\ &= (1-\theta)\underline{g}(y) + \theta\underline{g}^* + \frac{\theta^2\hat{c}_A^2}{\underline{t}}\left[\|r-\underline{y}^*\|_2^2 - \|r_+-\underline{y}^*\|_2^2\right].\end{aligned} \tag{8.3.7}$$

Since $1/\theta^2 = (1-\theta)/\theta^2 + 1/\theta$, by rearranging (8.3.7) we obtain (8.3.3). □

Next, we consider the update rule of θ. We can see from (8.3.3) that if θ_+ is updated such that $(1-\theta_+)/\theta_+^2 = 1/\theta^2$ then $\underline{g}(y_+) < \underline{g}(y)$. The last condition leads to:

$$\theta_+ = 0.5\theta(\sqrt{\theta^2+4} - \theta).$$

The following lemma was proved in Chapter 7.

Lemma 8.3.2. *The sequence* $\{\theta_k\}$ *generated by* $\theta_{k+1} := 0.5\theta_k[(\theta_k^2+4)^{1/2} - \theta_k]$ *and* $\theta_0 = 1$ *satisfies:*

$$\frac{1}{2k+1} \leq \theta_k \leq \frac{2}{k+2}, \quad \forall k \geq 0.$$

By Lemma 8.3.1, we have $r_+ = r - \rho\nabla\underline{g}(v)$ and $r_+ = \frac{1}{\theta_+}(v_+ - (1-\theta_+)y_+)$. From these relations, we deduce that:

$$v_+ = (1-\theta_+)y_+ + \theta_+(r - \rho\nabla\underline{g}(v)). \tag{8.3.8}$$

Note that if we combine (8.3.8) and (8.3.1) then:

$$v_+ = (1-\theta_+ - \frac{\rho\theta_+}{\alpha})y_+ - \frac{(1-\theta)\theta_+}{\theta}y + \left(\frac{1}{\theta} + \frac{\rho}{\alpha}\right)\theta_+ v.$$

This is in fact the second line of (8.3.1), where $\beta_1 := 1 - \theta_+ - \rho\theta_+\alpha^{-1}$, $\beta_2 := -(1-\theta)\theta_+\theta^{-1}$ and $\beta_3 := (\theta^{-1} + \rho\alpha^{-1})\theta_+$.

Before presenting the algorithm, we show how to choose $\hat{c}_A$ to ensure the condition $\lambda \leq \hat{c}_A$. Indeed, from Lemma 8.2.3 we see that if we choose $\hat{c}_A := \bar{c}_A + \|Ax^c - b\|_2$ then $\lambda \leq \hat{c}_A$. Now, by combining all the above analysis, we can describe the modified fast gradient algorithm in detail as follows:

Algorithm 8.3.1.(*Modified fast gradient decomposition algorithm*).
Initialization: Perform the following steps:

1. Given a tolerance $\varepsilon > 0$. Fix the parameter t at a certain value $\underline{t} > 0$ and compute $\hat{c}_A := \kappa|||A|||_{x^c}^* + \|Ax^c - b\|_2$.
2. Take an initial point $y^0 \in \mathbf{R}^m$.
3. Set $\theta_0 := 1$ and $v^0 := y^0$.

Iteration: For $k = 0, 1, \cdots$, perform the following steps:

Step 1: If $\lambda_k \leq \varepsilon$ then terminate.

Step 2: Compute $r^k := \theta_k^{-1}[v^k - (1-\theta_k)y^k]$.

Step 3: Update y^{k+1} as $y^{k+1} := v^k - \alpha_k \nabla g(v^k; \underline{t})$, where $\alpha_k = \frac{\underline{t}}{\hat{c}_A(\hat{c}_A + \lambda_k)}$.

Step 4: Update $\theta_{k+1} := \frac{1}{2}\theta_k[(\theta_k^2 + 4)^{1/2} - \theta_k]$.

Step 5: Update $v^{k+1} := (1 - \theta_{k+1})y^{k+1} + \theta_{k+1}(r^k - \rho_k \nabla g(v^k; \underline{t}))$, where $\rho_k := \frac{\underline{t}}{2\hat{c}_A^2 \theta_k}$.

Step 6: Solve (8.1.7) *in parallel* to obtain $x_{k+1}^* := x^*(v^{k+1}, \underline{t})$. Then, form a gradient vector $\nabla g(v^{k+1}; \underline{t}) := Ax_{k+1}^* - b$ and compute $\lambda_{k+1} := \left\|\nabla g(v^{k+1}; \underline{t})\right\|_2$.

End.

The core step of Algorithm 8.3.1 is Step 6, where we need to solve M primal subproblems of the form (8.1.7) in parallel.

The following theorem shows the convergence of Algorithm 8.3.1.

Theorem 8.3.1. *Let $y^0 \in \mathbf{R}^m$ be an initial point of Algorithm 8.3.1. Then the sequence $\{(y^k, v^k)\}_{k \geq 0}$ generated by Algorithm 8.3.1 satisfies:*

$$g(y^k; \underline{t}) - g^*(\underline{t}) \leq \frac{4\hat{c}_A^2}{\underline{t}(k+1)^2} \left\|y^0 - y^*(\underline{t})\right\|^2. \tag{8.3.9}$$

Proof. By the choice of $\hat{c}_A$ the condition $\lambda_k \leq \hat{c}_A$ is always satisfied. From (8.3.3) and the update rule of θ_k, we have:

$$\frac{1}{\theta_k^2}\left[\underline{g}(y^{k+1}) - \underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}}\left\|r^{k+1} - \underline{y}^*\right\|_2^2 \leq \frac{1}{\theta_{k-1}^2}\left[\underline{g}(y^k) - \underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}}\left\|r^k - \underline{y}^*\right\|_2^2.$$

By induction, we obtain from this inequality that:

$$\begin{aligned}\frac{1}{\theta_{k-1}^2}\left[\underline{g}(y^k) - \underline{g}^*\right] &\leq \frac{1}{\theta_0^2}\left[\underline{g}(y^1) - \underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}}\left\|r^1 - \underline{y}^*\right\|_2^2 \\ &\leq \frac{1 - \theta_0}{\theta_0^2}\left[\underline{g}(y^0) - \underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}}\left\|r^0 - \underline{y}^*\right\|_2^2,\end{aligned}$$

for $k \geq 1$. Since $\theta_0 = 1$ and $y^0 = v^0$, we have $r^0 = y^0$ and the last inequality implies $\underline{g}(y^k) - \underline{g}^* \leq \frac{\hat{c}_A^2 \theta_{k-1}^2}{\underline{t}}\left\|y^0 - \bar{y}\right\|_2^2$. Since $\theta_{k-1} \leq \frac{2}{k+1}$ due to Lemma 8.3.2, we obtain (8.3.9). □

Remark 8.3.2. *Let $\varepsilon > 0$ be a given accuracy. If we fix the penalty parameter $\underline{t} := \varepsilon$ then the worst-case complexity of Algorithm 8.3.1 is $O(\frac{2\hat{c}_A \underline{r}_0}{\varepsilon})$, where $\underline{r}_0 := r_0(\underline{t})$ is defined as above.*

Note that the constant $\hat{c}_A$ in Algorithm 8.3.1 looks rather large. Using this upper bound would lead to a slow convergence. In order to tune a better practical upper bound, let us take a constant $\hat{c}_A > 0$ and define:

$$\mathcal{R}(\hat{c}_A; \underline{t}) := \{y \in \mathbf{R}^m \mid \|\nabla g(y; \underline{t})\|_2 \leq \hat{c}_A\}. \tag{8.3.10}$$

It is obvious that $y^*(\underline{t}) \in \mathcal{R}(\hat{c}_A; \underline{t})$. This set is a neighbourhood of the solution $y^*(\underline{t})$ of problem (8.1.9). Moreover, by and observation that the sequence $\{v^k\}$ converges to the solution $y^*(\underline{t})$, we can assume that for k sufficiently large, $\{v^l\}_{l \geq k} \subseteq \mathcal{R}(\hat{c}_A; \underline{t})$. In this case, we can apply the following switching strategy.

Remark 8.3.3. (*Switching strategy*) *We can combine Algorithms 8.2.1 and 8.3.1 to obtain a switching variant:*

- *First, we apply Algorithm 8.2.1 to find a point $\hat{y}^0 \in \mathbf{R}^m$ and $\underline{t} > 0$ such that $\left\|\nabla g(\hat{y}^0; \underline{t})\right\|_2 \leq \hat{c}_A$.*
- *Then, we switch to use Algorithm 8.3.1.*

We notice that the sequence $\{\lambda_k\}_{k \geq 0}$ may not be monotone, the switching strategy does not ensure global convergence. However, as we will see in the following numerical tests, this variant still works well if we appropriately tune the parameter $\hat{c}_A$.

8.4 Numerical tests

In this section, we test the switching variant of Algorithms 8.2.1 and 8.3.1 proposed in Remark 8.3.3 which we name by `PFGDA` for solving the following convex programming problem:

$$\begin{array}{ll} \min\limits_{x \in \mathbf{R}^n} & \gamma \|x\|_1 + f(x) \\ \text{s.t.} & Ax = b,\ l \leq x \leq u, \end{array} \tag{8.4.1}$$

where $f(x) := \sum_{i=1}^n f_i(x_i)$, and $f_i : \mathbf{R} \to \mathbf{R}$ is a convex function, $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$ and $l, u \in \mathbf{R}^n$ such that $l \leq 0 < u$.

We note that the feasible set $X := [l, u]$ can be decomposed into n intervals $X_i := [l_i, u_i]$ and each interval is endowed with a 2-self concordant barrier $F_i(x_i) := -\ln(x_i - l_i) - \ln(u_i - x_i) + 2\ln((u_i - l_i)/2)$ for $i = 1, \ldots, n$. Moreover, if we define $\phi(x) := -\sum_{i=1}^n [f_i(x_i) + \gamma |x_i|]$ then ϕ is concave and separable. Problem (8.4.1) can be reformulated equivalently to (SepCOP$_{\max}$).

The smoothed dual function components $g_i(y;t)$ of (8.4.1) can be written as:

$$g_i(y;t) = \max_{l_i<x_i<u_i} \left\{-f_i(x_i) - \gamma\,|x_i| + (A_i^T y)x_i - tF_i(x_i)\right\} - b^T y/n,$$

for $i = 1,\ldots,n$. This one-variable minimization problem is nonsmooth but it can be solved easily. In particular, if f_i is affine then this problem can be solved in a *closed form*. In case f_i is smooth, we can reformulate (8.4.1) into a smooth convex program by adding n slack variables and $2n$ additional inequality constraints to handle the $\|x\|_1$ part.

We have implemented `PFGDA` in C++ running on a 16 cores Intel ®Xeon 2.7GHz workstation with 12 GB of RAM. The algorithm was parallelized by using `OpenMP`. We terminated `PFGDA` if:

$$\texttt{optim} := \left\|\nabla g(y^k;t_k)\right\|_2 / \max\left\{1, \left\|\nabla g(y^0;t_0)\right\|_2\right\} \le 10^{-3} \text{ and } t_k \le 10^{-2}.$$

We have also implemented two algorithms, Algorithm 7.3.1 and Algorithm 7.4.1 in Chapter 7 which we named `2pDecompAlg` and `2dDecompAlg`, respectively, for solving problem (8.4.1) and compared them with `PFGDA`. We terminated `2pDecompAlg` and `2dDecompAlg` by using the same conditions as in Chapter7 with the tolerances $\varepsilon_{\text{feas}} = \varepsilon_{\text{fun}} = \varepsilon_{\text{obj}} = 10^{-3}$ and $j_{\max} = 3$. We also terminated all three algorithms if the maximum number of iterations $\texttt{maxiter} := 10,000$ was reached. In the last case we declare that the algorithm is failed.

Basis pursuit problems

If the function $f(x) \equiv 0$ for all x then problem (8.4.1) becomes a bound constrained basis pursuit problem to recover the sparse coefficient vector x of given signals based on a transform operator A and a vector of observations b. We assume that $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$, where $m < n$ and x has k nonzero elements ($k \ll n$).

In this case, we only illustrate `PFGDA` by applying it to solve some small size test problems. In order to generate a test problem, we generate a random orthogonal matrix A and a random vector x_0 which has k nonzero elements. Then we define vector b as $b := Ax_0$.

We test `PFGDA` on the four problem instances such that $[m,n,k]$ are $[50,128,14]$, $[100,256,20]$, $[200,512,30]$ and $[500,1024,50]$. The results reported by `PFGDA` are plotted in Figure 8.2.

As we can see from these figures that the vector of recovered coefficients x matches very well the vector of original coefficients x_0 in these four problems. `PFGDA` requires $376, 334, 297$ and 332 iterations, respectively in the four problem instances.

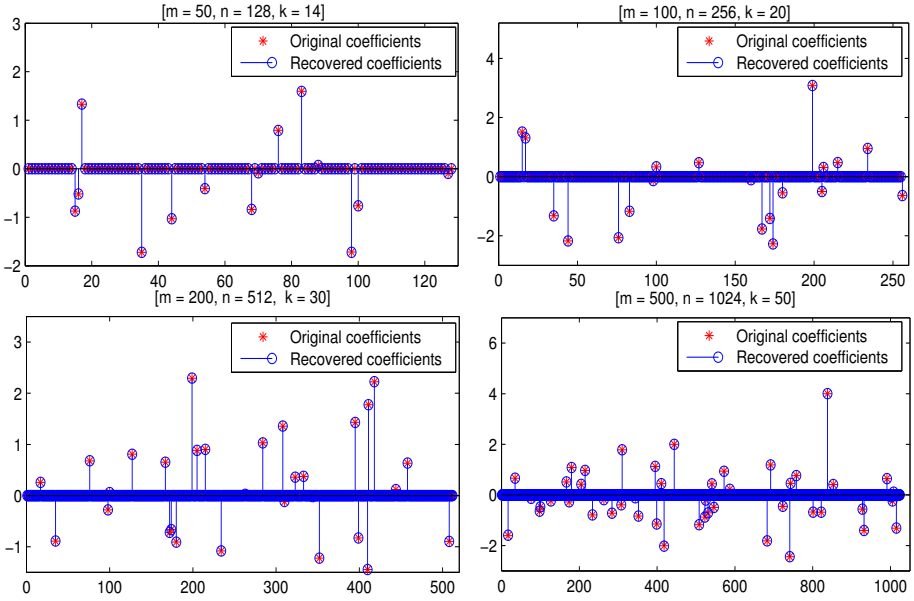


Figure 8.2: Illustration of PFGDA via the basis pursuit problem

Nonlinear separable convex programming problems

In order to test the performance of PFGDA, we generate in this case a large test-set of problems and compare the performance of PFGDA with 2pDecompAlg and 2dDecompAlg. The performance profiles were built as in Chapter 7.

The test problems were generated as follows. We chose the objective function $f_i(x_i) := e^{-\gamma_i x_i} - 1$, where $\gamma_i > 0$ is a given parameter for $i = 1, \ldots, n$. Matrix A was generated randomly in $[-1, 1]$ and then was normalized by $A/\|A\|_\infty$. We generated a sparse vector x_0 randomly in $[-2, 2]$ with the density 2% and defined a vector $b := A\bar{x}$. Vector $\gamma := (\gamma_1, \cdots, \gamma_n)^T$ was sparse and generated randomly in $[0, 0.5]$. The lower bound l_i and the upper bounds u_i were set to -3 and 3, respectively for all $i = 1, \ldots, n$.

We tested three algorithms on a collection of 50 random problem instances with m from 200 to 1,500 and n from 1,000 to 15,000. The performance profiles are plotted in Figure 8.3.

Based on this test, we can make the following observations. Both algorithms, PDGDA and 2dDecompAlg, can solve all the test problems, while 2pDecompAlg can only solve 46/50 (92%) problems. PFGDA requires a significantly fewer iterations

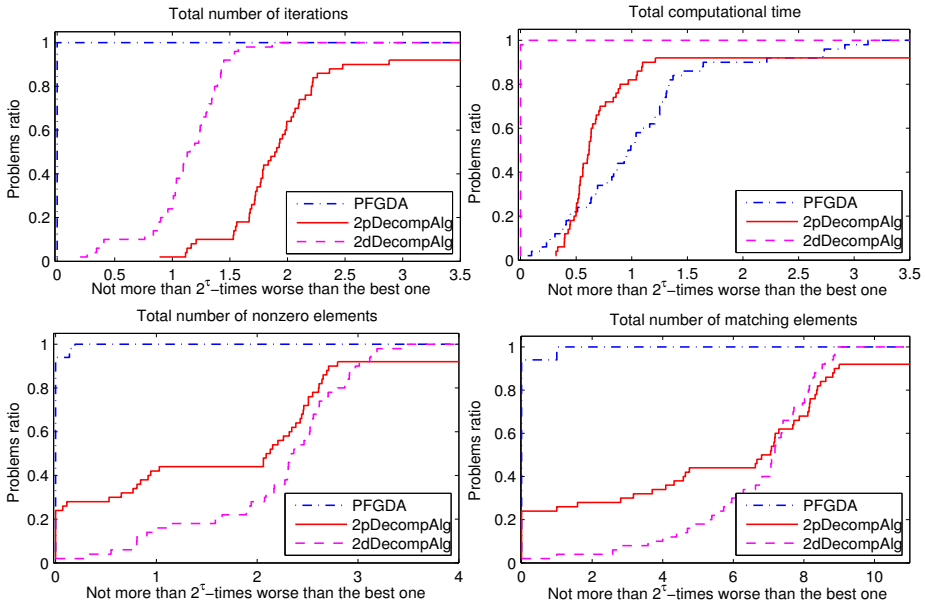


Figure 8.3: Performance profiles in $\log_2$ scale of three algorithms.

than `2pDecompAlg` and `2dDecompAlg`, and it has the best performance on 100% problems in terms of number of iterations. `2dDecompAlg` is the best in terms of computational time where it reaches 100% the test problem with the best performance. However, the number of nonzero elements of the obtained solution in `PFGDA` matches very well the vector of original coefficients x_0, while it is rather bad in `2pDecompAlg` and `2dDecompAlg` as we can see from the last figure. In other words, `2dDecompAlg` is not good at finding a sparse solution in this example.

8.5 Conclusion

This chapter has devoted to studying self-concordant barrier smoothing technique and gradient-type decomposition methods for separable convex optimization. We have proved some local and global estimates between the original dual function and the smoothed dual function. Then, two gradient-type decomposition algorithms have been proposed. The first algorithm is a path-following gradient decomposition method and the second is a modified fast gradient decomposition method. The convergence of both algorithms has been

investigated and their convergence rate has been established. These algorithms possess two advantages as indicated earlier compared to the methods proposed in the previous chapter. Finally, numerical tests have been implemented to verify the performance of these algorithms.

Chapter 9

An inexact perturbed path-following decomposition algorithm

In Chapter 8 we have studied a path-following gradient decomposition method for solving ($\text{SepCOP}_{\max}$). This method does not require any assumption imposed on the objective function of the problem except concavity. In this chapter, we take a closer look at the structure of this function where we assume that the objective function of problem ($\text{SepCOP}_{\max}$) is self-concordant or is *compatible* with the barrier of the feasible set [146]. Such problems also arise in many cases such as linear and quadratic programming. This allows us to apply interior-point decomposition methods to solve the smoothed dual problem instead of gradient-type methods.

Contribution of Chapter 9. The contribution of this chapter is as follows:

a) We propose a new *two-phase inexact perturbed path-following decomposition algorithm* for solving ($\text{SepCOP}_{\max}$). Both phases allow one to solve the primal subproblems approximately. The whole algorithm is highly *parallelizable*.

b) The convergence and the worst-case complexity of the algorithm are investigated under standard assumptions used in any interior point method.

c) As a special case, an exact path-following decomposition algorithm studied in [131, 135, 171, 218] is obtained. However, for this variant we obtain better values for the radius of the neighborhood of the central path compared to those from related existing methods.

Let us emphasize some differences between the proposed method and related existing methods. First, the new algorithm allows us to solve the primal subproblems inexactly, where the inexactness in the early iterations of the path-following algorithm can be high, resulting in significant time savings when the solution of the primal subproblems requires a high computational cost. Note that the proposed algorithm is different from the one considered in [220] for linear programming, where the inexactness of the primal subproblems was defined in a different way. Then, by analyzing directly the convergence of the algorithm based on the monograph [142], the theory presented in this chapter is self-contained. Moreover, it also allows us to optimally choose the parameters and to trade-off between the convergence rate of the master problem and the accuracy of the primal subproblems. Finally, in the exact case, the radius of the neighborhood of the central path is $(3-\sqrt{5})/2 \approx 0.38197$ which is larger than $2-\sqrt{3} \approx 0.26795$ of the previous methods [131, 135, 171, 218]. Moreover, since the performance of an interior point algorithm crucially depends on the parameters of the algorithm, we analyze directly the path-following iterations to select these parameters in an appropriate way.

Outline of Chapter 9. This chapter is organized as follows. Section 9.1 considers the self-concordance of the smoothed dual function and shows how to recover the optimality and the feasibility gaps of the original problem. Section 9.2 presents an inexact perturbed path-following decomposition algorithm and investigates the convergence and the worst-case complexity of the algorithm. Section 9.3 deals with an exact variant of the algorithm presented in Section 9.2. Section 9.4 discusses implementation details of the method. Section 9.5 provides a numerical example to test the performance of the proposed algorithms. We end this chapter with some conclusion.

9.1 Self-concordance of smoothed dual function

If the function $-\phi_i$ of problem ($\text{SepCOP}_{\max}$) is self-concordant on $\operatorname{dom}(\phi_i)$ with a parameter κ_{ϕ_i}, then the family of the functions $tF(\cdot) - \phi_i(\cdot)$ is also self-concordant on $\operatorname{dom}(\phi_i) \cap \operatorname{dom}(F_i)$. Consequently, the smoothed dual function $g(\cdot; t)$ is self-concordant due to Legendre's transformation as stated in the following lemma, see e.g. [131, 135, 171, 218].

Lemma 9.1.1. *Suppose that Assumptions* ***A.6.1.7*** *and* ***A.8.1.10*** *are satisfied. Suppose further that* $-\phi_i$ *is* κ_{ϕ_i}*-self-concordant. Then, for* $t > 0$*, the function* $g_i(\cdot;t)$ *defined by* (8.1.7) *is self-concordant with the parameter* $\kappa_{g_i} := \max\{\kappa_{\phi_i}, 2/\sqrt{t}\}$*,* $i = 1, \cdots, M$*. Consequently,* $g(\cdot;t)$ *is self-concordant with the parameter* $\kappa_g := \max_{1\leq i\leq M} \kappa_{g_i}$*.*

Similar to the standard path-following methods studied in [142, 146], in the following discussion, we assume that ϕ_i is linear as stated in the following assumption.

Asumption A. 9.1.11. *The function* ϕ_i *is linear, i.e.* $\phi_i(x_i) := c_i^T x_i$ *for* $i = 1, \cdots, M$*.*

Let $c := (c_1, \cdots, c_M)$ be a column vector formed from sub-vectors c_i for $i = 1, \cdots, M$. Assumption **A.9.1.11** and Lemma 9.1.1 imply that $g(\cdot,t)$ is $\frac{2}{\sqrt{t}}$-self-concordant. Since ϕ_i is linear, the optimality condition (8.1.8) is rewritten as:

$$c + A^T y - t\nabla F(x^*(y;t)) = 0. \tag{9.1.1}$$

Let $Y \subseteq \mathbf{R}^m$ be the restricted domain of the dual function g as considered in (8.1.15). The following lemma provides an explicit formula to compute the derivatives of $g(\cdot;t)$. The proof can be found in [135, 218].

Lemma 9.1.2. *Suppose that Assumptions* ***A.6.1.7****,* ***A.8.1.10*** *and* ***A.9.1.11*** *are satisfied. Then the gradient vector and the Hessian matrix of* $g(\cdot,t)$ *on* Y *are respectively given as:*

$$\nabla g(y;t) = Ax^*(y;t) - b \quad \text{and} \quad \nabla^2 g(y;t) = t^{-1} A \nabla^2 F(x^*(y;t))^{-1} A^T, \tag{9.1.2}$$

where $x^*(y;t)$ *is the solution of the primal subproblem* (8.1.7)*.*

Note that since A is full-row rank and $\nabla^2 F(x^*(y;t)) \succ 0$, we can see that $\nabla^2 g(y;t) \succ 0$ for any $y \in Y$. Now, since $g(\cdot;t)$ is $\frac{2}{\sqrt{t}}$ self-concordant, if we define:

$$\tilde{g}(y;t) := t^{-1} g(y;t), \tag{9.1.3}$$

then $\tilde{g}(\cdot;t)$ is standard self-concordant, i.e. $\kappa_{\tilde{g}} = 2$, due to [142, Corollary 4.1.2]. For a given vector $v \in \mathbb{R}^m$, we define the local norm $\|v\|_y$ of v w.r.t. $\tilde{g}(\cdot,t)$ as $\|v\|_y := [v^T \nabla^2 \tilde{g}(y;t) v]^{1/2}$ and the corresponding dual norm $\|u\|_y^*$ of u as $\|u\|_y^* := [u^T \nabla^2 \tilde{g}(y;t)^{-1} u]^{1/2}$.

Optimality and feasibility recovery

In this subsection, we show the relations between the master problem (8.1.9), the original dual problem (8.1.1) and the original primal problem ($\text{SepCOP}_{\max}$).

Let us define the Newton decrement of $\tilde{g}(\cdot, t)$ as follows:

$$\lambda = \lambda_{\tilde{g}(\cdot;t)}(y) := \|\nabla\tilde{g}(y;t)\|_y^* = \left[\nabla\tilde{g}(y;t)\nabla^2\tilde{g}(y;t)^{-1}\nabla\tilde{g}(y;t)\right]^{1/2}. \quad (9.1.4)$$

The following lemma shows the gap between $g(y;t)$ and $g^*(t)$.

Lemma 9.1.3. *Suppose that Assumptions* ***A.6.1.7****,* ***A.8.1.10*** *and* ***A.9.1.11*** *are satisfied. Then, for any $y \in Y$ and $t > 0$ such that $\lambda_{\tilde{g}(\cdot,t)}(y) \leq \beta < 1$, we have:*

$$0 \leq t\omega(\lambda_{\tilde{g}(\cdot,t)}(y)) \leq g(y;t) - g^*(t) \leq t\omega_*(\lambda_{\tilde{g}(\cdot,t)}(y)). \quad (9.1.5)$$

Moreover, it holds that:

$$(c + A^T y)^T(u - x^*(y;t)) \leq t\nu \text{ and } \|Ax^*(y;t) - b\|_y^* \leq t\beta, \quad (9.1.6)$$

for all $u \in X$.

Proof. Since $\tilde{g}(\cdot;t)$ is standard self-concordant and $y^*(t) = \operatorname{argmin}\{\tilde{g}(y;t) \mid y \in Y\}$, for any $y \in Y$ such that $\lambda \leq \beta < 1$, by applying [142, Theorem 4.1.13, inequality 4.1.17], we have:

$$0 \leq \omega(\lambda) \leq \tilde{g}(y;t) - \tilde{g}(y^*(t);t) \leq \omega_*(\lambda).$$

By (9.1.3), these inequalities are equivalent to (9.1.5). It follows from the optimality condition (9.1.1) that $c + A^T y = t\nabla F(x^*(y;t))$. Hence, by [142, Theorem 4.2.4], we have:

$$(c + A^T y)^T(u - x^*(y;t)) = t\nabla F(x^*(y;t))^T(u - x^*(y;t)) \leq t\nu,$$

for any $u \in \mathrm{dom}F$. Since $X \subseteq \mathrm{dom}F$, the last inequality implies the first condition in (9.1.6). Furthermore, from (9.1.2) we have $\nabla g(y;t) = Ax^*(y;t) - b$. Therefore,

$$\|Ax^*(y;t) - b\|_y^* = t\,\|\nabla\tilde{g}(y^*(t);t)\|_y^* = t\lambda_{\tilde{g}(\cdot,t)}(y) \leq t\beta,$$

which proves the second inequality in (9.1.6). □

Let us recall the optimality condition for the primal-dual problems ($\mathrm{SepCOP}_{\max}$)-(8.1.1) as:

$$\begin{cases} 0 & \in c + A^T y_0^* - \mathcal{N}_X(x_0^*), \\ 0 & = Ax_0^* - b, \end{cases} \quad \forall (x_0^*, y_0^*) \in \mathbb{R}^n \times \mathbb{R}^m, \quad (9.1.7)$$

where $\mathcal{N}_X(x)$ is the normal cone of X at x. Since X^* is nonempty, the first inclusion also indicates implicitly that $x_0^* \in X$. Moreover, if $x_0^* \in X$ then (9.1.7) can be expressed equivalently to:

$$(c + A^T y_0^*)^T (u - x_0^*) \leq 0, \quad \forall u \in X.$$

Now, we define an approximate solution of ($\text{SepCOP}_{\max}$)-(8.1.1) as follows.

Definition 9.1.1. *For a given tolerance $\varepsilon_p \geq 0$, a point $(\tilde{x}^*, \tilde{y}^*) \in X \times \mathbf{R}^m$ is said to be an ε_p-solution of ($\text{SepCOP}_{\max}$)-(8.1.1) if $(c + A^T \tilde{y}^*)^T (u - \tilde{x}^*) \leq \varepsilon_p$ for all $u \in X$ and $\|A\tilde{x}^* - b\|_{\tilde{y}^*}^* \leq \varepsilon_p$.*

It is clear that for any point $x \in \text{int}(X)$, $\mathcal{N}_X(x) = \{0\}$. Furthermore, according to (9.1.7), the conditions in Definition (9.1.1) are well-defined.

Finally, we note that $\nu \geq 1$, $\beta < 1$ and $x^*(y;t) \in \text{int}(X)$. By (9.1.6), if we choose the tolerance $\varepsilon_p := \nu t$ then $(x^*(y;t), y)$ is an ε_p-solution of ($\text{SepCOP}_{\max}$)-(8.1.1) in the sense of Definition 9.1.1. We denote the feasibility gap by $\mathcal{F}(y;t) := \|Ax^*(y;t) - b\|_y^*$ for further references.

9.2 Inexact perturbed path-following decomposition method

This section presents an inexact perturbed path-following decomposition algorithm for solving (8.1.1).

Inexact solution of the primal subproblems

First, we define an inexact solution of (8.1.7) by using local norms. For a given $y \in Y$ and $t > 0$, suppose that we solve approximately (8.1.7) up to a given accuracy $\bar{\delta} \geq 0$. More precisely, we define this approximation as follows.

Definition 9.2.1. *For given $\bar{\delta} \in [0, 1)$, a vector $\bar{x}_{\bar{\delta}}(y;t)$ is said to be a $\bar{\delta}$-approximate solution of $x^*(y;t)$ if:*

$$\|\bar{x}_{\bar{\delta}}(y;t) - x^*(y;t)\|_{x^*(y;t)} \leq \bar{\delta}. \tag{9.2.1}$$

Associated with $\bar{x}_{\bar{\delta}}(\cdot)$, we define the following function:

$$g_{\bar{\delta}}(y;t) := c^T \bar{x}_{\bar{\delta}}(y;t) + y^T (A\bar{x}_{\bar{\delta}}(y;t) - b) - tF(\bar{x}_{\bar{\delta}}(y;t)). \tag{9.2.2}$$

This function can be considered as an *inexact version* of g. Next, we introduce two quantities:

$$\nabla g_{\bar{\delta}}(y;t) := A\bar{x}_{\bar{\delta}}(y;t) - b, \text{ and } \nabla^2 g_{\bar{\delta}}(y;t) := t^{-1}A\nabla^2 F(\bar{x}_{\bar{\delta}}(y;t))^{-1}A^T. \quad (9.2.3)$$

Since $x^*(y;t) \in \mathrm{dom}(F)$, we can choose an appropriate $\bar{\delta} \geq 0$ such that $\bar{x}_{\bar{\delta}}(y;t) \in \mathrm{dom}(F)$. Hence, $\nabla^2 F(\bar{x}_{\bar{\delta}}(y;t))$ is positive definite which means that $\nabla^2 g_{\bar{\delta}}$ is well-defined. Note that $\nabla g_{\bar{\delta}}$ and $\nabla^2 g_{\bar{\delta}}$ are not the gradient vector and Hessian matrix of $g_{\bar{\delta}}(\cdot;t)$, respectively. Nevertheless, due to Lemma 9.1.2 and (9.2.1), we can consider these quantities as an approximate gradient vector and Hessian matrix of $g(\cdot;t)$, respectively.

Let

$$\tilde{g}_{\bar{\delta}}(y;t) := t^{-1}g_{\bar{\delta}}(y;t), \quad (9.2.4)$$

and $\bar{\lambda}$ be the inexact Newton decrement of $\tilde{g}_\delta$ which is defined by:

$$\bar{\lambda} = \bar{\lambda}_{\tilde{g}_{\bar{\delta}}(\cdot;t)}(y) := \|\nabla\tilde{g}_{\bar{\delta}}(y;t)\|_y^* = \left[\nabla\tilde{g}_{\bar{\delta}}(y;t)\nabla^2\tilde{g}_{\bar{\delta}}(y;t)^{-1}\nabla\tilde{g}_{\bar{\delta}}(y;t)\right]^{1/2}. \quad (9.2.5)$$

Here, we use the norm $\|\cdot\|_y$ to distinguish it from $\|\cdot\|_y$.

The algorithmic framework

From Lemma 9.1.3 we see that if we can generate a sequence $\{(y^k, t_k)\}_{k\geq 0}$ such that $\lambda_k := \lambda_{\tilde{g}(\cdot,t_k)}(y^k) \leq \beta < 1$, then:

$$g(y^k;t_k) \uparrow g^* = \phi^* \text{ and } \mathcal{F}(y^k;t_k) \to 0, \text{ as } t_k \downarrow 0^+.$$

Therefore, the aim of the algorithm is to generate a sequence $\{(y^k, t_k)\}_{k\geq 0}$ such that $\lambda_k \leq \beta < 1$ and $t_k \downarrow 0^+$. First, we fix $t = t_0 > 0$ and find a point $y^0 \in Y$ such that $\lambda_{\tilde{g}(\cdot;t_0)}(y^0) \leq \beta$. Then we simultaneously update y^k and t_k in a path-following manner such that $t_k \downarrow 0^+$. The algorithmic framework is presented as follows.

Algorithm 9.2.1.(*Inexact-perturbed path-following decomposition framework*).
Initialization. Choose an appropriate $\beta \in (0,1)$ and a tolerance $\varepsilon_g > 0$. Fix $t = t_0 > 0$ a priori.
Phase 1. (*Determine a starting point* $y^0 \in Y$ *such that* $\lambda_{\tilde{g}(\cdot;t_0)}(y^0) \leq \beta$).
Choose an initial vector $y^{0,0} \in Y$.
For $j = 0, 1, \ldots, j_{\max}$, perform the following steps:

1. If $\lambda_j := \lambda_{\tilde{g}(\cdot;t_0)}(y^{0,j}) \leq \beta$ then set $y^0 := y^{0,j}$ and terminate.
2. Solve (8.1.7) *in parallel* to obtain an approximation of $x^*(y^{0,j};t_0)$.

3. Evaluate $\nabla g_{\bar{\delta}}(y^{0,j};t_0)$ and $\nabla^2 g_{\bar{\delta}}(y^{0,j};t_0)$ by (9.2.3).
4. Perform the inexact-perturbed damped Newton step: $y^{0,j+1} := y^{0,j} - \alpha_j \nabla^2 g_{\bar{\delta}}(y^{0,j};t_0)^{-1} \nabla g_{\bar{\delta}}(y^{0,j};t_0)$, where $\alpha_j \in (0,1]$ is a given step size.

Phase 2. (*Path-following iterations*).
Compute an appropriate value $\sigma \in (0,1)$.
For $k = 0, 1, \ldots, k_{\max}$, perform the following steps:

1. If $t_k \leq \varepsilon_g / \omega_*(\beta)$ then terminate.
2. Update $t_{k+1} := (1-\sigma)t_k$.
3. Solve (8.1.7) *in parallel* to obtain an approximation of $x^*(y^k;t_{k+1})$.
4. Evaluate the quantities $\nabla g_{\bar{\delta}}(y^k;t_{k+1})$ and $\nabla^2 g_{\bar{\delta}}(y^k;t_{k+1})$ as in (9.1.2).
5. Perform the inexact-perturbed full-step Newton step as $y^{k+1} := y^k - \nabla^2 g_{\bar{\delta}}(y^k;t_{k+1})^{-1} \nabla g_{\bar{\delta}}(y^k;t_{k+1})$.

Output. An ε_g-approximate solution y^k of problem (8.1.9), i.e. $0 \leq g(y^k;t_k) - g^*(t_k) \leq \varepsilon_g$.
End.

This algorithm is still conceptual. In the following subsections, we shall discuss each step of this algorithmic framework in detail. We note that the proposed algorithm provides an ε_g-approximate solution y^k such that $t_k \leq \varepsilon_t := \omega_*(\beta)^{-1}\varepsilon_g$. Now, by solving the primal subproblem (8.1.7), we obtain $x^*(y^k;t_k)$ as an ε_p-solution of ($\text{SepCOP}_{\max}$) in the sense of Definition 9.1.1, where $\varepsilon_p := \nu\varepsilon_t$. The maximum numbers of iterations $j_{\max}$ and $k_{\max}$ will be defined in the sequel.

Computing inexact solutions

The condition (9.2.1) can not be used in practice to compute $\bar{x}_{\bar{\delta}}$ since $x^*(y;t)$ is unknown. We need to show how to compute $\bar{x}_{\bar{\delta}}$ practically such that (9.2.1) holds.

For the sake of notational simplicity, we abbreviate by $\bar{x}_{\bar{\delta}} := \bar{x}_{\bar{\delta}}(y;t)$ and $x^* := x^*(y;t)$. The error of the approximate solution $\bar{x}_{\bar{\delta}}$ to x^* is defined as:

$$\delta(\bar{x}_{\bar{\delta}}, x^*) := \|\bar{x}_{\bar{\delta}}(y;t) - x^*(y;t)\|_{x^*(y;t)}. \tag{9.2.6}$$

The following lemma gives a criterion such that the condition (9.2.1) holds.

Lemma 9.2.1. *Let $\delta(\bar{x}_{\bar{\delta}}, x^*)$ be defined by (9.2.6) such that $\delta(\bar{x}_{\bar{\delta}}, x^*) < 1$. Then:*

$$0 \leq t\omega(\delta(\bar{x}_{\bar{\delta}}, x^*)) \leq g(y;t) - g_{\bar{\delta}}(y;t) \leq t\omega_*(\delta(\bar{x}_{\bar{\delta}}, x^*)). \tag{9.2.7}$$

Moreover, if:

$$E^c_{\bar{\delta}} := \left\|c + A^T y - t\nabla F(\bar{x}_{\bar{\delta}})\right\|^*_{x^c} \leq \varepsilon_g := \left[\kappa_\nu(1+\bar{\delta})\right]^{-1}\bar{\delta}t, \tag{9.2.8}$$

where $\kappa_\nu := \nu + 2\sqrt{\nu}$, then $\bar{x}_{\bar{\delta}}(y;t)$ satisfies (9.2.1). Consequently, if $t \leq \omega_*(\beta)^{-1}\varepsilon_g$ and $\bar{\delta} < 1$ then:

$$|g_{\bar{\delta}}(y;t) - g^*(t)| \leq \left[1 + \omega_*(\beta)^{-1}\omega_*(\bar{\delta})\right]\varepsilon_g. \tag{9.2.9}$$

Proof. It follows from the definitions of $g(\cdot,t)$ and $g_{\bar{\delta}}(\cdot,t)$, and (9.1.1) that:

$$\begin{aligned} g(y;t) - g_{\bar{\delta}}(y;t) &= [c + A^T y](x^* - \bar{x}_{\bar{\delta}}) - t[F(x^*) - F(\bar{x}_{\bar{\delta}})] \\ &= -t[F(x^*) + \nabla F(x^*)^T(\bar{x}_{\bar{\delta}} - x^*) - F(\bar{x}_{\bar{\delta}})]. \end{aligned}$$

Since F is self-concordant, by applying [142, Theorems 4.1.7 and 4.1.8], and the definition of $\delta(\bar{x}_{\bar{\delta}}, x^*)$, the above equality implies that:

$$0 \leq t\omega(\delta(\bar{x}_{\bar{\delta}}, x^*)) \leq g(y;t) - g_{\bar{\delta}}(y;t) \leq t\omega_*(\delta(\bar{x}_{\bar{\delta}}, x^*)),$$

which is indeed (9.2.7).

Next, by using again (9.1.1) and the definition of $E^c_{\bar{\delta}}$ we have:

$$E^c_{\bar{\delta}} \overset{(9.1.1)}{=} t\left\|\nabla F(\bar{x}_{\bar{\delta}}) - \nabla F(x^*)\right\|^*_{x^c} \geq \kappa_\nu^{-1} t\left\|\nabla F(\bar{x}_{\bar{\delta}}) - \nabla F(x^*)\right\|^*_{x^*},$$

where the last inequality follows from [142, Corollary 4.2.1]. Combining this inequality and [142, Theorem 4.1.7], we obtain:

$$\begin{aligned} \frac{\delta(\bar{x}_{\bar{\delta}}, x^*)^2}{1 + \delta(\bar{x}_{\bar{\delta}}, x^*)} &\leq \left[\nabla F(\bar{x}_{\bar{\delta}}) - \nabla F(x^*)\right]^T(\bar{x}_{\bar{\delta}} - x^*) \\ &\leq \left\|\nabla F(\bar{x}_{\bar{\delta}}) - \nabla F(x^*)\right\|^*_{x^*}\left\|\bar{x}_{\bar{\delta}} - x^*\right\|_{x^*} \\ &\leq t^{-1}\kappa_\nu E^c_{\bar{\delta}}\delta(\bar{x}_{\bar{\delta}}, x^*). \end{aligned}$$

Hence, we get:

$$\delta(\bar{x}_{\bar{\delta}}, x^*) \leq \left[t - \kappa_\nu E^c_{\bar{\delta}}\right]^{-1}\kappa_\nu E^c_{\bar{\delta}}, \tag{9.2.10}$$

provided that $t > \kappa_\nu E^c_{\bar{\delta}}$. Let us define an accuracy ε_p for the primal subproblem (8.1.7) as $\varepsilon_p := [\kappa_\nu(1+\bar{\delta})]^{-1}\bar{\delta}t \geq 0$. Then it follows from (9.2.10) that if $E^c_{\bar{\delta}} \leq [\kappa_\nu(1+\bar{\delta})]^{-1}\bar{\delta}t$ then $\bar{x}_{\bar{\delta}}(y;t)$ satisfies (9.2.1). It remains to consider the distance from g_δ to $g^*(t)$ when t is sufficiently small. Suppose that $t \leq \omega_*(\beta)^{-1}\varepsilon_g$. Then, by combining (9.1.5) and (9.2.7) we obtain (9.2.9). □

Remark 9.2.1. *Since*

$$E_{\bar{\delta}} := \left\|c + A^T y - t\nabla F(\bar{x}_{\bar{\delta}})\right\|^*_{\bar{x}_{\bar{\delta}}} \geq (1-\bar{\delta})\left\|c + A^T y - t\nabla F(\bar{x}_{\bar{\delta}})\right\|^*_{x^*},$$

by the same argument as in the proof of Lemma 9.2.1, we can show that if $E_{\bar{\delta}} \leq \hat{\varepsilon}_p$, *where* $\hat{\varepsilon}_p := \frac{\bar{\delta}(1-\bar{\delta})t}{1+\bar{\delta}}$ *then* (9.2.1) *holds. This condition can be used instead of* (9.2.8) *to terminate the algorithm presented in the next section.*

Phase 2: The path-following scheme with inexact-perturbed full-step Newton iterations

Now, we analyze Steps 2-5 in Phase 2 of the algorithmic framework. In the path-following fashion, we only perform one inexact-perturbed full-step Newton (IPFNT) iteration for each value of the parameter t. Thus one iteration of this scheme is specified as follows:

$$\begin{cases} t_+ := t - \Delta t, \\ y_+ := y - \nabla^2 g_{\bar{\delta}}(y; t_+)^{-1}\nabla g_{\bar{\delta}}(y; t_+). \end{cases} \tag{9.2.11}$$

Since the Newton method is invariant under linear transformations, by (9.2.2), the second line of (9.2.11) is equivalent to:

$$y_+ := y - \nabla^2 \tilde{g}_{\bar{\delta}}(y; t_+)^{-1}\nabla \tilde{g}_{\bar{\delta}}(y; t_+). \tag{9.2.12}$$

For the sake of notational simplicity, we denote all the functions at $(y_+; t_+)$ and $(y; t_+)$ by the sub-index "$_+$" and "$_1$", respectively, and at $(y; t)$ without index in the following analysis. More precisely, we denote by:

$$\begin{array}{lll} \bar{\lambda}_+ := \bar{\lambda}_{\tilde{g}_{\bar{\delta}}(\cdot;t_+)}(y_+), & \delta_+ := \|\bar{x}_{\bar{\delta}}(y_+; t_+) - x^*(y_+; t_+)\|_{x^*(y_+;t_+)}, \\ \bar{\lambda}_1 := \bar{\lambda}_{\tilde{g}_{\bar{\delta}}(\cdot;t_+)}(y), & \delta_1 := \|\bar{x}_{\bar{\delta}}(y; t_+) - x^*(y; t_+)\|_{x^*(y;t_+)}, \\ \bar{\lambda} := \bar{\lambda}_{\tilde{g}_{\bar{\delta}}(\cdot,t)}(y), & \delta := \|\bar{x}_{\bar{\delta}}(y; t) - x^*(y; t)\|_{x^*(y;t)}, \end{array} \tag{9.2.13}$$

and by

$$\Delta := \|\bar{x}_{\bar{\delta}}(y; t_+) - \bar{x}_{\bar{\delta}}(y; t)\|_{\bar{x}_{\bar{\delta}}(y;t)} \text{ and } \Delta^* := \|x^*(y; t_+) - x^*(y; t)\|_{x^*(y;t)}. \tag{9.2.14}$$

Note that the above notation does not cause any confusion since it can be recognized from the context.

The main estimate

Now, by using the notation in (9.2.13) and (9.2.14), we provide a main estimate which will be used to analyze the convergence of the algorithm presented in Subsection 9.2.

Lemma 9.2.2. *Let $y \in Y$ be given and $t > 0$. Let (y_+, t_+) be a pair generated by (9.2.11). Suppose that $\delta_1 + 2\Delta + \bar{\lambda} < 1$, $\delta_+ < 1$ and $\xi := \frac{\Delta+\bar{\lambda}}{1-\delta_1-2\Delta-\bar{\lambda}}$. Then:*

$$\bar{\lambda}_+ \le (1-\delta_+)^{-1}\Big\{\delta_+ + \delta_1 + \xi^2 + \delta_1\left[(1-\delta_1)^{-2} + 2(1-\delta_1)^{-1}\right]\xi\Big\}. \quad (9.2.15)$$

Moreover, the right-hand side of (9.2.15) is nondecreasing w.r.t. all variables δ_+, δ_1, Δ and $\bar{\lambda}$.

In particular, if we set $\delta_+ = 0$ and $\delta_1 = 0$, i.e. the primal subproblem (8.1.7) is assumed to be solved exactly, then $\bar{\lambda}_+ = \lambda_+$, $\bar{\lambda} = \lambda$ and (9.2.15) reduces to:

$$\lambda_+ \le (1 - 2\Delta^* - \lambda)^{-2}(\lambda + \Delta^*)^2, \quad (9.2.16)$$

provided that $\lambda + 2\Delta^* < 1$.

For clarity of the exposition we move the proof of this lemma to Appendix A.2.

Maximum neighborhood of the central path

The key point of the path-following algorithm is to determine the maximum neighborhood of the central path $(\beta_*, \beta^*) \subseteq (0,1)$ such that:

For any $\beta \in (\beta_*, \beta^*)$, if $\bar{\lambda} \le \beta$ then $\bar{\lambda}_+ \le \beta$.

Now, we analyze the estimate (9.2.15) to find the parameters $\bar{\delta}$ and Δ such that the last condition holds.

Suppose that $\bar{\delta} \ge 0$ as in Definition 9.2.1. First, we construct the following parametric cubic polynomial:

$$\mathcal{P}_{\bar{\delta}}(\beta) := c_0(\bar{\delta}) + c_1(\bar{\delta})\beta + c_2(\bar{\delta})\beta^2 + c_3(\bar{\delta})\beta^3, \quad (9.2.17)$$

where the coefficients are given by:

$$\begin{cases} c_0(\bar{\delta}) & := -2\bar{\delta}(1-\bar{\delta})^2 \le 0, \\ c_1(\bar{\delta}) & := (1-\bar{\delta})^{-1}[1 - 3\bar{\delta} + \bar{\delta}^4], \\ c_2(\bar{\delta}) & := \bar{\delta}[(1-\bar{\delta})^{-2} + 2(1-\bar{\delta})^{-1}] - 3 + 2\bar{\delta}(1-\bar{\delta}), \\ c_3(\bar{\delta}) & := 1 - \bar{\delta} > 0. \end{cases}$$

Then we define:

$$p := \bar{\delta}[(1-\bar{\delta})^{-2} + 2(1-\bar{\delta})^{-1}],\ q := (1-\bar{\delta})\beta - 2\bar{\delta} \text{ and } \theta := 0.5(\sqrt{p^2+4q} - p). \quad (9.2.18)$$

The following theorem provides the conditions such that if $\bar{\lambda} \le \beta$ then $\bar{\lambda}_+ \le \beta$.

Theorem 9.2.2. *Suppose that* $\bar{\delta} \in [0, \bar{\delta}_{\max}] := [0, 0.043286]$ *is fixed and* θ *is defined by* (9.2.18). *Then the polynomial* $\mathcal{P}_{\bar{\delta}}$ *defined by* (9.2.17) *has three nonnegative real roots* $0 \le \beta_* < \beta^* < \beta_3$. *Moreover, if we choose* $\beta \in (\beta_*, \beta^*)$ *and compute* $\bar{\Delta} := \frac{\theta(1-\bar{\delta}-\beta)-\beta}{1+2\theta}$ *then* $\bar{\Delta} > 0$ *and, for* $0 \le \delta_+ \le \bar{\delta}$, $0 \le \delta_1 \le \bar{\delta}$ *and* $0 \le \Delta \le \bar{\Delta}$, *the condition* $\bar{\lambda} \le \beta$ *implies* $\bar{\lambda}_+ \le \beta$.

Proof. Let us define $\bar{\xi} := \frac{\Delta+\beta}{1-\bar{\delta}-\beta-2\Delta}$ and:

$$\varphi(\beta, \bar{\delta}, \Delta) := (1-\bar{\delta})^{-1}\big\{2\bar{\delta} + \bar{\xi}^2 + \bar{\delta}[(1-\bar{\delta})^{-2} + 2(1-\bar{\delta})^{-1}]\bar{\xi}\big\}.$$

By assumption $\bar{\lambda} \le \beta$, it follows from Lemma 9.2.2 that if $\varphi(\beta, \bar{\delta}, \Delta) \le \beta$ then $\bar{\lambda}_+ \le \beta$. This condition holds if:

a) $0 \le \bar{\xi} \le (\sqrt{p^2 + 4q} - p)/2 \equiv \theta$ and

b) $0 \le \bar{\delta} \le \beta/(\beta + 2)$,

where p and q are defined by (9.2.18). The condition a) is equivalent to $(1 + 2\theta)\Delta \le \theta(1 - \bar{\delta} - \beta) - \beta$. Because $\Delta > 0$, we need $\theta > (1 - \bar{\delta} - \beta)^{-1}\beta$. This is guaranteed if $\mathcal{P}_{\bar{\delta}}(\beta) > 0$, where $\mathcal{P}_{\bar{\delta}}$ is defined in (9.2.17). By a well-known characteristic of a cubic polynomial, $\mathcal{P}_{\bar{\delta}}(\beta)$ has three real roots if:

$$18c_0c_1c_2c_3 - 4c_2^3c_0 + c_2^2c_1^2 - 4c_3c_1^3 - 27c_3^2c_0^2 \ge 0.$$

By numerically checking the last condition, we can show that if $0 \le \bar{\delta} \le \bar{\delta}_{\max} := 0.043286$ then the three roots satisfy $0 \le \beta_* < \beta^* < \beta_3$ and $\mathcal{P}_{\bar{\delta}}(\beta) > 0$ for all $\beta \in (\beta_*, \beta^*)$. With such values of $\bar{\delta}$ and β we have $\theta > (1 - \bar{\delta} - \beta)^{-1}\beta$ and the condition b) is also satisfied. Eventually, if we define $\bar{\Delta} := \frac{\theta(1-\bar{\delta}-\beta)-\beta}{1+2\theta} > 0$ and choose $\bar{\delta}$, β and Δ such that $0 \le \bar{\delta} \le \bar{\delta}_{\max}$, $\beta \in (\beta_*, \beta^*)$ and $0 \le \Delta \le \bar{\Delta}$ then $\bar{\lambda} \le \beta$ implies $\bar{\lambda}_+ \le \beta$. □

Now, we illustrate the variation of the values of β_*, β^* and $\bar{\Delta}$ w.r.t. $\bar{\delta}$ in Figure 9.1. The left figure shows the values of β_* (solid) and β^* (dash) and the right one plots the value of $\bar{\Delta}$ when β is chosen by $\beta := \frac{\beta_*+\beta^*}{2}$ (dash) and $\beta := \frac{\beta^*}{4}$ (solid), respectively.

Update rule of the penalty parameter

It remains to quantify the decrement Δt of the penalty parameter t in (9.2.11). The following lemma shows how to update t.

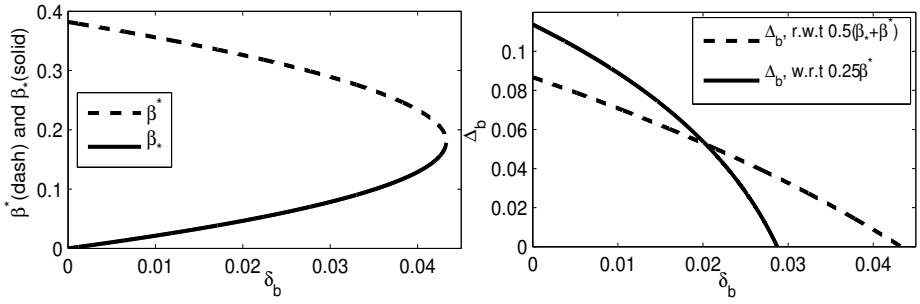


Figure 9.1: The values of β_*, β^* and $\bar{\Delta}$ varying w.r.t $\bar{\delta}$.

Lemma 9.2.3. *Let $\bar{\delta}$ and $\bar{\Delta}$ be defined as in Theorem 9.2.2 and let:*

$$\bar{\Delta}^* := \frac{1}{2}\Big[(1-\bar{\delta})\bar{\Delta} - \bar{\delta} + 1 - \sqrt{((1-\bar{\delta})\bar{\Delta} - \bar{\delta} - 1)^2 + 4\bar{\delta}}\Big]. \tag{9.2.19}$$

Then the penalty parameter t can be decreased linearly, i.e. $t_+ := (1-\sigma)t$, where $\sigma := [\sqrt{\nu} + \bar{\Delta}^*(\sqrt{\nu}+1)]^{-1}\bar{\Delta}^* \in (0,1)$.

Proof. It follows from (9.1.1) that $c + A^T y - t\nabla F(x^*) = 0$ and $c + A^T y - t_+\nabla F(x_1^*) = 0$, where $x^* := x^*(y;t)$ and $x_1^* := x^*(y;t_+)$. Subtracting these equalities and then using $t_+ = t - \Delta t$, we have $t_+[\nabla F(x_1^*) - \nabla F(x^*)] = \Delta t \nabla F(x^*)$. Using this relation together with [142, Theorem 4.1.7] and $\|\nabla F(x^*)\|_{x^*}^* \le \sqrt{\nu}$ (see [142, inequality 4.2.4]), we have:

$$\frac{t_+ \|x_1^* - x^*\|_{x^*}^2}{1 + \|x_1^* - x^*\|_{x^*}} \le t_+[\nabla F(x_1^*) - \nabla F(x^*)]^T(x_1^* - x^*) = \Delta t \nabla F(x^*)^T(x_1^* - x^*)$$

$$\le \Delta t \, \|\nabla F(x^*)\|_{x^*}^* \, \|x_1^* - x^*\|_{x^*} \le \Delta t\sqrt{\nu}\, \|x_1^* - x^*\|_{x^*}.$$

By the definition of Δ^* in (9.2.14), if $t > (\sqrt{\nu}+1)\Delta t$, then the above inequality leads to:

$$\Delta^* \le \bar{\Delta}^* := t\big[t - (\sqrt{\nu}+1)\Delta t\big]^{-1}\sqrt{\nu}\Delta. \tag{9.2.20}$$

Therefore,

$$\Delta t = t\big[\sqrt{\nu} + (\sqrt{\nu}+1)\bar{\Delta}^*\big]^{-1}\bar{\Delta}^*. \tag{9.2.21}$$

On the other hand, using the definitions of Δ and δ, we have:

$$\Delta := \|\bar{x}_{\bar{\delta}1} - \bar{x}_{\bar{\delta}}\|_{\bar{x}_{\bar{\delta}}} \overset{(A.2.3)}{\le} (1-\delta)^{-1}\Big[\|\bar{x}_{\bar{\delta}1} - x_1^*\|_{x^*} + \|x_1^* - x^*\|_{x^*} + \|x^* - \bar{x}_{\bar{\delta}}\|_{x^*}\Big]$$

$$\le (1-\delta)^{-1}\left[(1-\Delta^*)^{-1}\delta_1 + \Delta^* + \delta\right] \tag{9.2.22}$$

$$\overset{(9.2.20),\delta,\delta_1\le\bar{\delta}}{\le} (1-\bar{\delta})^{-1}\left[(1-\bar{\Delta}^*)^{-1}\bar{\delta} + \bar{\Delta}^* + \bar{\delta}\right].$$

Now, we need to find a condition such that $\Delta \leq \bar{\Delta}$, where $\bar{\Delta}$ is given in Theorem 9.2.2. It follows from (9.2.22) that $\Delta \leq \bar{\Delta}$ if $\frac{\bar{\delta}}{1-\Delta^*} + \Delta^* \leq (1-\bar{\delta})\bar{\Delta} - \bar{\delta}$. The last condition holds if:

$$0 \leq \bar{\Delta}^* \leq \frac{1}{2}\Big[(1-\bar{\delta})\bar{\Delta} - \bar{\delta} + 1 - \sqrt{((1-\bar{\delta})\bar{\Delta} - \bar{\delta} - 1)^2 + 4\bar{\delta}}\Big], \tag{9.2.23}$$

provided that $\bar{\delta} \leq \frac{\bar{\Delta}}{1+\bar{\Delta}}$, due to (9.2.20). Thus, we can fix $\bar{\Delta}^*$ at the upper bound as defined in (9.2.19). By (9.2.21), the update rule for the penalty parameter t becomes $t_+ := t - \sigma t = (1-\sigma)t$ where $\sigma := \frac{\bar{\Delta}^*}{\sqrt{\nu}+\bar{\Delta}^*(\sqrt{\nu}+1)} \in (0,1)$. □

Finally, we show that the conditions given in Theorem 9.2.2 and Lemma 9.2.3 are well-defined. Indeed, let us fix $\bar{\delta} := 0.01$. Then we can compute the values of β_* and β^* as $\beta_* \approx 0.021371 < \beta^* \approx 0.356037$. Therefore, if we choose $\beta := \frac{\beta^*}{4} \approx 0.089009 > \beta_*$ then $\bar{\Delta} \approx 0.089012$ and $\bar{\Delta}^* \approx 0.067399$.

The algorithm and its convergence

Before presenting the algorithm, we need to find a stopping criterion. By using Lemma A.2.1c. with Δ instead of δ, we have:

$$\lambda \leq (1-\delta)^{-1}(\bar{\lambda} + \delta), \tag{9.2.24}$$

provided that $\delta < 1$ and $\bar{\lambda} \leq \beta < 1$. Consequently, if $\bar{\lambda} \leq (1-\bar{\delta})\beta - \bar{\delta}$ then $\lambda \leq \beta$. Let us define $\vartheta := (1-\bar{\delta})\beta - \bar{\delta}$, where $0 < \bar{\delta} < \beta/(\beta+1)$. It follows from Lemma 9.1.3 that if $t\omega_*(\vartheta) \leq \varepsilon_g$ for a given tolerance $\varepsilon_g > 0$, then y is an ε_g-solution of (8.1.9).

The second phase of the algorithmic framework presented in Subsection 9.2 is now described in detail as follows.

Algorithm 9.2.2.(*Path-following algorithm with* IPFNT *iterations*).
Initialization: Choose $\bar{\delta} \in [0, \bar{\delta}_{\max}]$ and compute β_* and β^* as in Theorem 9.2.2.
Phase 1. Apply Algorithm 9.2.3 presented in Subsection 9.2 below to find $y^0 \in Y$ such that $\lambda_{\tilde{g}_{\bar{\delta}}(\cdot;t_0)}(y^0) \leq \beta$.

Phase 2.
Initialization of Phase 2: Perform the following steps:

1. Given a tolerance $\varepsilon_g > 0$.
2. Compute $\bar{\Delta}$ as in Theorem 9.2.2. Then, compute $\bar{\Delta}^*$ by (9.2.19).

3. Compute $\sigma := \frac{\bar{\Delta}^*}{\sqrt{\nu}+(\sqrt{\nu}+1)\bar{\Delta}^*}$ and the accuracy factor $\gamma := \frac{\bar{\delta}}{\kappa_\nu(1+\bar{\delta})}$.

Iteration: For $k = 0, 1, \cdots, k_{\max}$, perform the following steps:

1. If $t_k \le \frac{\varepsilon_g}{\omega_*(\vartheta)}$, where $\vartheta := (1-\bar{\delta})\beta - \bar{\delta}$, then terminate.
2. Compute an accuracy $\varepsilon_k := \gamma t_k$ for the primal subproblems.
3. Update $t_{k+1} := (1-\sigma)t_k$.
4. Solve approximately (8.1.7) *in parallel* up to the accuracy ε_k to obtain $\bar{x}_{\bar{\delta}}(y^k; t_{k+1})$.
5. Compute $\nabla g_{\bar{\delta}}(y^k; t_{k+1})$ and $\nabla^2 g_{\bar{\delta}}(y^k; t_{k+1})$ as in (9.2.3).
6. Update y^{k+1} as $y^{k+1} := y^k - \nabla^2 g_{\bar{\delta}}(y^k; t_{k+1})^{-1}\nabla g_{\bar{\delta}}(y^k; t_{k+1})$.

End.

The core steps of Phase 2 in Algorithm 9.2.2 are Steps 4 and 6, where we need to solve M convex primal subproblems *in parallel* and computing the IPFNT direction, respectively. Note that Step 6 requires one to solve a linear equation system. In addition, the quantity $\nabla^2 F(\bar{x}_{\bar{\delta}}(y^k; t_{k+1}))$ can also be computed *in parallel*.

The parameter t at Step 3 can be updated adaptively as $t_{k+1} := (1-\sigma_k)t_k$, where $\sigma_k := \frac{\bar{\Delta}^*}{R_{\bar{\delta}}+(R_{\bar{\delta}}+1)\bar{\Delta}^*}$ and $R_{\bar{\delta}} := (1-\bar{\delta})^{-1}\big[\bar{\delta}(1-\bar{\delta})^{-1} + \|\nabla F(\bar{x}_{\bar{\delta}})\|^*_{\bar{x}_{\bar{\delta}}}\big]$. The stopping criterion at Step 1 can be replaced by $\omega_*(\vartheta_k)t_k \le \varepsilon_g$, where $\vartheta_k := (1-\bar{\delta})^{-1}[\lambda_{\tilde{g}_{\bar{\delta}}(\cdot;t_k)}(y^k) + \bar{\delta}]$ due to Lemma 9.1.3 and (9.2.24).

Let us define $\lambda_{k+1} := \lambda_{\tilde{g}_{\bar{\delta}}(\cdot;t_{k+1})}(y^{k+1})$ and $\lambda_k := \lambda_{\tilde{g}_{\bar{\delta}}(\cdot;t_k)}(y^k)$. Then the local convergence of Algorithm 9.2.2 is stated in the following theorem.

Theorem 9.2.3. *Let $\{(y^k, t_k)\}$ be a sequence generated by Algorithm 9.2.2. Then the number of iterations to obtain an ε_g-solution of (8.1.9) does not exceed:*

$$k_{\max} := \Big\lfloor [\ln(1-\sigma)]^{-1} \ln\Big(\frac{\varepsilon_g}{t_0\omega_*(\vartheta)}\Big)\Big\rfloor + 1, \tag{9.2.25}$$

where $\sigma := \frac{\bar{\Delta}^*}{\sqrt{\nu}+(\sqrt{\nu}+1)\bar{\Delta}^*} \in (0,1)$ and $\vartheta := (1-\bar{\delta})\beta - \bar{\delta} \in (0,1)$.

Proof. Note that y^k is an ε_g-solution of (8.1.9) if $t_k \le \frac{\varepsilon_g}{\omega_*(\vartheta)}$ due to Lemma 9.1.3, where $\vartheta = (1-\bar{\delta})\beta - \bar{\delta}$. Since $t_k = (1-\sigma)^k t_0$ due to Step 3, we require $(1-\sigma)^k \le \frac{\varepsilon_g}{t_0\omega_*(\vartheta)}$. Consequently, we obtain (9.2.25). □

Remark 9.2.4 (**The worst-case complexity**). *Since* $(1-\sigma)^{-1} = 1 + \frac{\bar{\Delta}^*}{\sqrt{\nu}(\bar{\Delta}^*+1)}$, *we have* $-\ln(1-\sigma) \approx \sigma = \frac{\bar{\Delta}^*}{\sqrt{\nu}(\bar{\Delta}^*+1)}$. *It follows from Theorem 9.2.3 that the complexity of Algorithm 9.2.2 is* $O\left(\sqrt{\nu}\ln\frac{t_0}{\varepsilon_g}\right)$.

Remark 9.2.5 (**Linear convergence**). *The sequence* $\{t_k\}$ *linearly converges to zero with a contraction factor not greater than* $1-\sigma$. *When* $\lambda_{\tilde{g}_{\bar{\delta}}(\cdot;t)}(y) \leq \beta$, *it follows from* (9.1.3) *that* $\lambda_{g_{\bar{\delta}}(\cdot;t)}(y) \leq \beta\sqrt{t}$. *Thus the sequence of the Newton decrements* $\{\lambda_{g(\cdot;t_k)}(y^k)\}_k$ *of* g *also converges linearly to zero with a contraction factor not greater than* $\sqrt{1-\sigma}$.

Remark 9.2.6 (**The inexactness of the IPFNT direction**). *Note that we can also apply an inexact method to solve the linear system for computing an IPFNT direction in* (9.2.11). *For more details of this approach, one can refer to [211].*

Finally, as a consequence of Theorem 9.2.3, the following corollary shows how to recover the optimality and feasibility of the original primal-dual problems ($\text{SepCOP}_{\max}$)-(8.1.1).

Corollary 9.2.1. *Suppose that* $(y^k; t_k)$ *is the output of Algorithm 9.2.2 and* $x^*(y^k; t_k)$ *is the solution of the primal subproblem* (8.1.7). *Then* $(x^*(y^k; t_k), y^k)$ *is an* ε_p*-solution of* ($\text{SepCOP}_{\max}$)-(8.1.1), *where* $\varepsilon_p := \nu\omega_*(\beta)^{-1}\varepsilon_g$.

Phase 1: Finding a starting point

Phase 1 of the algorithmic framework aims at finding $y^0 \in Y$ such that $\lambda_{\tilde{g}_{\bar{\delta}}(\cdot;t)}(y^0) \leq \beta$. In this subsection, we consider an inexact perturbed damped Newton (IPDNT) method for finding such a point y^0.

Inexact perturbed damped Newton iteration

For a given value $t = t_0 > 0$ and a given accuracy $\bar{\delta} \geq 0$, let us assume that the current point $y \in Y$ is given, we compute the new point y_+ by applying the IPDNT iteration as follows:

$$y_+ := y - \alpha(y)\nabla^2 g_{\bar{\delta}}(y, t_0)^{-1}\nabla g_{\bar{\delta}}(y, t_0), \tag{9.2.26}$$

where $\alpha := \alpha(y) > 0$ is the step size which will be chosen appropriately. Note that since (9.2.26) is invariant under linear transformations, it is equivalent to:

$$y_+ := y - \alpha(y)\nabla^2 \tilde{g}_{\bar{\delta}}(y, t_0)^{-1}\nabla \tilde{g}_{\bar{\delta}}(y, t_0), \tag{9.2.27}$$

It follows from (9.1.3) that $\tilde{g}(\cdot;t_0)$ is standard self-concordant, and by [142, Theorem 4.1.8], we have:

$$\tilde{g}(y_+;t_0) \leq \tilde{g}(y,t_0) + \nabla\tilde{g}(y,t_0)^T(y_+ - y) + \omega_*(\|y_+ - y\|_y), \tag{9.2.28}$$

provided that $\|y_+ - y\|_y < 1$. On the other hand, (9.2.7) implies that:

$$0 \leq \omega(\delta(\bar{x}_{\bar{\delta}}, x^*)) \leq \tilde{g}(y,t_0) - \tilde{g}_{\bar{\delta}}(y,t_0) \leq \omega_*(\delta(\bar{x}_{\bar{\delta}}, x^*)), \tag{9.2.29}$$

which bounds the error between $\tilde{g}(\cdot;t_0)$ and $\tilde{g}_{\bar{\delta}}(\cdot;t_0)$. In order to analyze the convergence of the IPDNT iteration (9.2.26) we denote by:

$$\begin{aligned}
\hat{\delta}_+ &:= \|\bar{x}_{\bar{\delta}}(y_+;t_0) - x^*(y_+;t_0)\|_{x^*(y_+;t_0)}, \\
\hat{\delta} &:= \|\bar{x}_{\bar{\delta}}(y,t_0) - x^*(y,t_0)\|_{x^*(y,t_0)}, \\
\bar{\lambda}_0 &:= \lambda_{\tilde{g}_{\bar{\delta}}(\cdot;t_0)}(y) = \alpha(y)\|y_+ - y\|_y,
\end{aligned} \tag{9.2.30}$$

the solution differences of $g(\cdot;t_0)$ and $g_{\bar{\delta}}(\cdot;t_0)$ and the Newton decrement of $\tilde{g}_{\bar{\delta}}(\cdot;t_0)$, respectively.

Finding the step size

The following lemma provides a formula to update the step size $\alpha(y)$ in (9.2.26).

Lemma 9.2.4. *Let $0 < \bar{\hat{\delta}} < \hat{\delta}^* := \beta(2+\beta+2\sqrt{\beta+1})^{-1}$ and $\underline{\eta}$ be defined as:*

$$\underline{\eta} := \beta\Big[(1+\bar{\hat{\delta}})\beta + \sqrt{(1-\bar{\hat{\delta}})^2\beta^2 - 4\bar{\hat{\delta}}\beta}\Big]^{-1}\Big[(1-\bar{\hat{\delta}})\beta - 2\bar{\hat{\delta}} + \sqrt{(1-\bar{\hat{\delta}})^2\beta^2 - 4\bar{\hat{\delta}}\beta}\Big]. \tag{9.2.31}$$

Then $\underline{\eta} \in (0,1)$. Furthermore, if we choose the step size $\alpha(y)$ as:

$$\alpha(y) := \big[2\bar{\lambda}_0(1+\bar{\lambda}_0)\big]^{-1}\Big[(1-\bar{\hat{\delta}})\bar{\lambda}_0 - 2\bar{\hat{\delta}} + \sqrt{(1-\bar{\hat{\delta}})^2\bar{\lambda}_0^2 - 4\bar{\hat{\delta}}\bar{\lambda}_0}\Big], \tag{9.2.32}$$

then $\alpha(y) \in (0,1)$ and:

$$\tilde{g}_{\bar{\delta}}(y_+;t_0) \leq \tilde{g}_{\bar{\delta}}(y,t_0) - \omega(\underline{\eta}). \tag{9.2.33}$$

As a consequence, if $\bar{\hat{\delta}} = 0$ then $\underline{\eta} = \beta$ and $\alpha(y) := (1+\bar{\lambda}_0)^{-1}$.

The asymptotic behaviour of the functions $\eta(\cdot)$ and $\alpha(\cdot)$ w.r.t. $\bar{\hat{\delta}}$ is plotted in Figure 9.2 below. We can observe that α depends almost linearly on $\bar{\hat{\delta}}$.

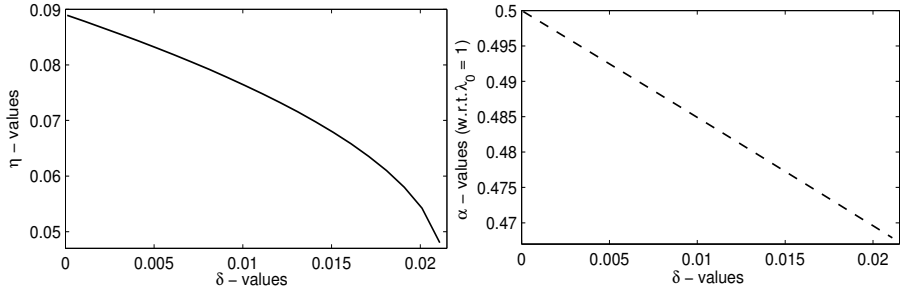


Figure 9.2: The asymptotic behaviour of η and α w.r.t. $\bar{\hat{\delta}}$ at $\lambda_0 = 1$ and $\beta = 0.089009$.

Proof. Let $p := y_+ - y$. From (9.2.28) and (9.2.29), we have:

$$\begin{aligned}
\tilde{g}_{\bar{\delta}}(y_+;t_0) &\overset{(9.2.29)}{\leq} \tilde{g}(y_+;t_0) \overset{(9.2.28)}{\leq} \tilde{g}(y,t_0) + \nabla\tilde{g}(y,t_0)^T(y_+ - y) + \omega_*(\|y_+ - y\|_y) \\
&\overset{(9.2.29)}{\leq} \tilde{g}_{\bar{\delta}}(y,t_0) + \nabla\tilde{g}(y,t_0)^T(y_+ - y) + \omega_*(\|y_+ - y\|_y) + \omega_*(\hat{\delta}) \\
&= \tilde{g}_{\bar{\delta}}(y,t_0) + \nabla\tilde{g}_{\bar{\delta}}(y,t_0)^T p + \left[\nabla\tilde{g}(y,t_0) - \nabla\tilde{g}_{\bar{\delta}}(y,t_0)\right]^T p + \omega_*(\|p\|_y) + \omega_*(\hat{\delta}) \\
&\overset{(9.2.26)}{\leq} \tilde{g}_{\bar{\delta}}(y,t_0) - \alpha\bar{\lambda}_0^2 + \|\nabla\tilde{g}(y,t_0) - \nabla\tilde{g}_{\bar{\delta}}(y,t_0)\|_y^* \|p\|_y + \omega_*(\|p\|_y) + \omega_*(\hat{\delta}) \\
&\overset{(A.2.2)}{\leq} \tilde{g}_{\bar{\delta}}(y,t_0) - \alpha\bar{\lambda}_0^2 + \hat{\delta}\|p\|_y + \omega_*(\|p\|_y) + \omega_*(\hat{\delta}). \qquad (9.2.34)
\end{aligned}$$

Furthermore, from (A.2.4) and the definition of $\nabla^2\tilde{g}$ and $\nabla^2\tilde{g}_{\bar{\delta}}$, we have:

$$(1-\hat{\delta})\nabla^2\tilde{g}_{\bar{\delta}}(y,t_0) \preceq \nabla^2\tilde{g}(y,t_0) \preceq (1-\hat{\delta})^{-2}\nabla^2\tilde{g}_{\bar{\delta}}(y,t_0).$$

These inequalities imply $(1-\hat{\delta})\|\|p\|\|_y \leq \|p\|_y \leq (1-\hat{\delta})^{-1}\|\|p\|\|_y$. Combining the previous inequalities, (9.2.27) and the definition of $\bar{\lambda}_0$ in (9.2.30) we get:

$$\alpha(1-\hat{\delta})\bar{\lambda}_0 \leq \|p\|_y \leq \alpha(1-\hat{\delta})^{-1}\bar{\lambda}_0.$$

Let us assume that $\alpha\bar{\lambda}_0 + \hat{\delta} < 1$. By substituting the second inequality into (9.2.34) and observing that the right hand side of (9.2.34) is nondecreasing w.r.t. $\|p\|_y$, we obtain:

$$\tilde{g}_{\bar{\delta}}(y_+;t_0) \leq \tilde{g}_{\bar{\delta}}(y,t_0) - \alpha\bar{\lambda}_0^2 + (1-\hat{\delta})^{-1}\alpha\bar{\lambda}\hat{\delta} + \omega_*\left((1-\hat{\delta})^{-1}\alpha\bar{\lambda}_0\right) + \omega_*(\hat{\delta}). \quad (9.2.35)$$

Now, let us simplify the last four terms of (9.2.35) which we denote by $[\cdot]_{[1]}$ as follows:

$$\begin{aligned}[\cdot]_{[1]} &:= -\alpha\bar{\lambda}_0^2 + (1-\hat{\delta})^{-1}\alpha\bar{\lambda}_0\hat{\delta} + \omega_*\left((1-\hat{\delta})^{-1}\alpha\bar{\lambda}_0\right) + \omega_*(\hat{\delta})\\ &= -\alpha\bar{\lambda}_0^2 - (\alpha\bar{\lambda}_0 + \hat{\delta}) - \ln\left[1 - (\alpha\bar{\lambda}_0 + \hat{\delta})\right] \qquad (9.2.36)\\ &= -\alpha\bar{\lambda}_0^2 + \omega_*(\alpha\bar{\lambda}_0 + \hat{\delta}).\end{aligned}$$

Suppose that we can choose $\eta > 0$ such that $\alpha\bar{\lambda}_0^2 - \omega_*(\alpha\bar{\lambda}_0 + \hat{\delta}) = \omega(\eta)$. This condition leads to $\alpha\bar{\lambda}_0^2 = (\alpha\bar{\lambda}_0 + \hat{\delta})\left[\alpha(\bar{\lambda}_0 + \bar{\lambda}_0) + \hat{\delta}\right]$ which is equivalent to:

$$\alpha = \left[2\bar{\lambda}_0(1+\bar{\lambda}_0)\right]^{-1}\left[(1-\hat{\delta})\bar{\lambda}_0 - 2\hat{\delta} + \sqrt{(1-\hat{\delta})^2\bar{\lambda}_0^2 - 4\hat{\delta}\bar{\lambda}_0}\right], \qquad (9.2.37)$$

provided that $0 \le \hat{\delta} < \bar{\hat{\delta}} := \frac{2+\bar{\lambda}_0-2\sqrt{1+\bar{\lambda}_0}}{\bar{\lambda}_0}$. Consequently, we deduce:

$$\eta = \bar{\lambda}_0\left[(1+\hat{\delta})\bar{\lambda}_0 + \sqrt{(1-\hat{\delta})^2\bar{\lambda}_0^2 - 4\hat{\delta}\bar{\lambda}_0}\right]^{-1}\left[(1-\hat{\delta})\bar{\lambda}_0 - 2\hat{\delta} + \sqrt{(1-\hat{\delta})^2\bar{\lambda}_0^2 - 4\hat{\delta}\bar{\lambda}_0}\right].$$

We assume that $\bar{\lambda}_0 \ge \beta$ for a given $\beta \in (0,1)$. Let us fix $\bar{\hat{\delta}}$ such that:

$$0 < \bar{\hat{\delta}} < \hat{\delta}^* := \beta^{-1}\left[2 + \beta - 2\sqrt{1+\beta}\right] = \left[2 + \beta + 2\sqrt{1+\beta}\right]^{-1}\beta.$$

If we choose the step size $\alpha(y)$ as in (9.2.32) for the IPDNT iteration (9.2.26) then we obtain (9.2.33) with $\underline{\eta}$ defined by (9.2.31). □

Finally, we estimate the constant $\underline{\eta}$ for the case $\beta \approx 0.089009$. We first obtain $\hat{\delta}^* \approx 0.021314$. Let $\bar{\hat{\delta}} := \frac{1}{2}\hat{\delta}^* \approx 0.010657$. Then we get $\underline{\eta} \approx 0.075496$ and $\omega(\underline{\eta}) \approx 0.003002$.

The algorithm and its worst-case complexity

In summary, the algorithm for finding $y^0 \in Y$ is presented in detail as follows.

Algorithm 9.2.3.(*Phase 1: Finding a starting point* $y^0 \in Y$).
Initialization: Perform the following steps:

1. Select $\beta \in (\beta_*, \beta^*)$ and $t_0 > 0$ as desired (e.g. $\beta = \frac{1}{4}\beta^* \approx 0.089009$).
2. Take an arbitrary point $y^{0,0} \in Y$.

3. Compute $\hat{\delta}^* := \beta[2+\beta+2\sqrt{1+\beta}]^{-1}$ and fix $\bar{\hat{\delta}} \in (0, \hat{\delta}^*)$ (e.g. $\bar{\hat{\delta}} = 0.5\hat{\delta}^*$).
4. Compute an accuracy $\varepsilon_0 := \frac{t_0\bar{\hat{\delta}}}{\kappa_\nu(1+\hat{\delta})}$.

Iteration: For $j = 0, 1, \cdots, j_{\max}$, perform the following steps:

1. Solve approximately (8.1.7) *in parallel* up to the accuracy ε_0 to obtain $\bar{x}_{\bar{\delta}}(y^{0,j}; t_0)$.
2. Compute $\bar{\lambda}_j := \bar{\lambda}_{\tilde{g}_{\bar{\delta}}(\cdot;t_0)}(y^{0,j})$.
3. If $\bar{\lambda}_j \leq \beta$ then set $y^0 := y^{0,j}$ and terminate.
4. Update $y^{0,j+1}$ as:

$$y^{0,j+1} := y^{0,j} - \alpha_j \nabla^2 g_{\bar{\delta}}(y^{0,j}, t_0)^{-1} \nabla g_{\bar{\delta}}(y^{0,j}, t_0),$$

where

$$\alpha_j := \left[2\bar{\lambda}_j(1+\bar{\lambda}_j)\right]^{-1}\left[(1-\bar{\hat{\delta}})\bar{\lambda}_j - 2\bar{\hat{\delta}} + \sqrt{(1-\bar{\hat{\delta}})^2\bar{\lambda}_j^2 - 4\bar{\hat{\delta}}\bar{\lambda}_j}\right] \in (0,1).$$

End.

The convergence of this algorithm is stated in the following theorem.

Theorem 9.2.7. *The number of iterations required in Algorithm 9.2.3 does not exceed:*

$$j_{\max} := \left\lfloor [t_0\omega(\underline{\eta})]^{-1}\left[g_{\bar{\delta}}(y^{0,0}, t_0) - g^*(t_0) + \omega_*(\bar{\hat{\delta}})\right]\right\rfloor + 1, \tag{9.2.38}$$

where $g^*(t_0) = \min_{y\in Y} g(y, t_0)$ *and* $\underline{\eta}$ *is given by* (9.2.31).

Proof. Summing up (9.2.33) from $j = 0$ to $j = k$ and then using (9.2.29) we have $0 \leq \tilde{g}(y^{0,k}, t_0) - \tilde{g}^*(t_0) \leq \tilde{g}_{\bar{\delta}}(y^{0,k}, t_0) + \omega_*(\bar{\hat{\delta}}) - \tilde{g}^*(t_0) \leq \tilde{g}_{\bar{\delta}}(y^{0,0}, t_0) + \omega_*(\bar{\hat{\delta}}) - \tilde{g}^*(t_0) - k\omega(\underline{\eta})$. This inequality together with (9.1.3) and (9.2.4) imply:

$$j \leq [t_0\omega(\underline{\eta})]^{-1}\left[g_{\bar{\delta}}(y^{0,0}, t_0) - g^*(t_0) + \omega_*(\bar{\hat{\delta}})\right].$$

Hence, the maximum iteration number in Algorithm 9.2.3 does not exceed $j_{\max}$ defined by (9.2.38). □

Since $g^*(t_0)$ is unknown, the constant $j_{\max}$ in (9.2.38) only gives an upper bound for Algorithm 9.2.3. However, in this algorithm, we do not use $j_{\max}$ as a stopping criterion.

9.3 Exact path-following decomposition algorithm

In Algorithm **6.1.7**, if we set $\bar{\delta} = 0$, then this algorithm reduces to the ones considered in [113, 135, 171, 218, 219] as a special case. Note that, in [113, 135, 171, 218, 219], the primal subproblem (8.1.7) is assumed to be solved exactly so that the family $\{g(\cdot;t)\}_{t>0}$ of the smoothed dual functions is strongly self-concordant due to the Legendre transformation. Consequently, the standard theory of interior point methods in [146] can be applied to minimize this function. In contrast to those methods, in this section we analyze directly the path-following iterations to select appropriate parameters for implementation. Moreover, the radius of the neighbourhood of the central path in Algorithm 9.3.1 below is $(3-\sqrt{5})/2 \approx 0.381966$ compared to the one, $2-\sqrt{3} \approx 0.267949$, in the mentioned papers.

The exact path-following iteration

Let us assume that the primal subproblem (8.1.7) is solved exactly, i.e. $\bar{\delta} = 0$ in Definition 9.2.1. Then, we have $\bar{x}_{\bar{\delta}} \equiv x^*$ and $\delta(\bar{x}_{\bar{\delta}}, x^*) = 0$ for all $y \in Y$ and $t > 0$. Moreover, it follows from (9.2.20) that $\Delta = \Delta^* = \|x^*(y;t_+) - x^*(y;t)\|_{x^*(y;t)}$. We consider one step of the path-following scheme with exact full-step Newton iterations:

$$\begin{cases} t_+ := t - \Delta t, \;\; \Delta t > 0, \\ y_+ := y - \nabla^2 g(y;t_+)^{-1}\nabla g(y;t_+) \equiv y - \nabla^2 \tilde{g}(y;t_+)^{-1}\nabla \tilde{g}(y;t_+). \end{cases} \tag{9.3.1}$$

For the sake of notational simplicity, we denote by $\tilde{\lambda} := \lambda_{\tilde{g}(\cdot;t)}(y)$, $\tilde{\lambda}_1 := \lambda_{\tilde{g}(\cdot,t_+)}(y)$ and $\tilde{\lambda}_+ := \lambda_{\tilde{g}(\cdot;t_+)}(y_+)$. It follows from (9.2.16) of Lemma 9.2.2 that:

$$\tilde{\lambda}_+ \leq \big(1 - 2\Delta^* - \tilde{\lambda}\big)^{-2}\big(\tilde{\lambda} + \Delta^*\big)^2. \tag{9.3.2}$$

Now, we fix $\beta \in (0,1)$ and assume that $\tilde{\lambda} \leq \beta$. We need to find a condition on Δ such that $\tilde{\lambda}_+ \leq \beta$. Indeed, since the right-hand side of (9.3.2) is nondecreasing w.r.t. $\tilde{\lambda}$, it implies that $\tilde{\lambda}_+ \leq (1-2\Delta^*-\beta)^{-2}(\Delta^*+\beta)^2$. Thus if $\frac{\Delta^*+\beta}{1-2\Delta^*-\beta} \leq \sqrt{\beta}$ then $\tilde{\lambda}_+ \leq \beta$. The last condition leads to:

$$0 \leq \Delta^* \leq \bar{\Delta}^* := (1+2\sqrt{\beta})^{-1}\sqrt{\beta}(1-\sqrt{\beta}-\beta), \tag{9.3.3}$$

provided that:

$$0 < \beta < \beta^* := (3-\sqrt{5})/2 \approx 0.381966. \tag{9.3.4}$$

In particular, if we choose $\beta = \frac{\beta^*}{4} \approx 0.095492$ then $\bar{\Delta}^* \approx 0.113729$. Since $\Delta \equiv \Delta^*$, according to (9.2.21) and (9.3.1), we can update t as:

$$t_+ := (1-\sigma)t, \quad \text{where } \sigma := \left[\sqrt{\nu} + (\sqrt{\nu}+1)\bar{\Delta}^*\right]^{-1}\bar{\Delta}^* \in (0,1). \tag{9.3.5}$$

The algorithm and its convergence

The exact variant of Algorithms 9.2.2 and 9.2.3 is presented in detail as follows.

Algorithm 9.3.1.(*Path-following algorithm with exact Newton iterations*).
Initialization: Given a tolerance $\varepsilon_g > 0$ and choose an initial value $t_0 > 0$. Fix a constant $\beta \in (0, \beta^*)$, where $\beta^* = \frac{3-\sqrt{5}}{2} \approx 0.381966$. Then, compute:

$$\bar{\Delta}^* := \frac{\sqrt{\beta}(1-\sqrt{\beta}-\beta)}{1+2\sqrt{\beta}} \text{ and } \sigma := \frac{\bar{\Delta}^*}{\sqrt{\nu}+(\sqrt{\nu}+1)\bar{\Delta}^*}.$$

Phase 1. (*Finding a starting point*).
Choose an arbitrary starting point $y^{0,0} \in Y$.
For $j = 0, 1, \cdots, \tilde{j}_{\max}$, perform the following steps:

1. Solve *exactly* the primal subproblem (8.1.7) *in parallel* to obtain $x^*(y^{0,j}; t_0)$.
2. Evaluate $\nabla g(y^{0,j}; t_0)$ and $\nabla^2 g(y^{0,j}, t_0)$ as in (9.1.2). Then compute the Newton decrement $\tilde{\lambda}_j = \lambda_{\tilde{g}(\cdot;t_0)}(y^{0,j})$.
3. If $\tilde{\lambda}_j \le \beta$ then set $y^0 := y^{0,j}$ and terminate.
4. Update $y^{0,j+1}$ as:

$$y^{0,j+1} := y^{0,j} - (1+\tilde{\lambda}_j)^{-1}\nabla^2 g(y^{0,j}, t_0)^{-1}\nabla g(y^{0,j}, t_0).$$

Phase 2. (*Path-following iterations*).
For $k = 0, 1, \cdots, \tilde{k}_{\max}$, perform the following steps:

1. If $t_k \le \frac{\varepsilon_g}{\omega_*(\beta)}$ then terminate.
2. Update t_k as $t_{k+1} := (1-\sigma)t_k$.
3. Solve *exactly* the primal subproblem (8.1.7) *in parallel* to obtain $x^*(y^k; t_{k+1})$.
4. Evaluate $\nabla g(y^k; t_{k+1})$ and $\nabla^2 g(y^k; t_{k+1})$ as in (9.1.2).

5. Update y^{k+1} as:

$$y^{k+1} := y^k + \Delta y^k = y^k - \nabla^2 g(y^k; t_{k+1})^{-1} \nabla g(y^k; t_{k+1}).$$

End.

Since $\tilde{g}(\cdot; t_0)$ is standard self-concordant due to Lemma 9.1.1. By [142, Theorem 4.1.12], the number of iterations required in Phase 1 does not exceed:

$$\tilde{j}_{\max} := \left\lfloor \left[\tilde{g}(y^{0,0}; t_0) - \tilde{g}^*(t_0)\right] \omega(\beta)^{-1} \right\rfloor + 1 = \left\lfloor \left[g(y^{0,0}; t_0) - g^*(t_0)\right] [t_0 \omega(\beta)]^{-1} \right\rfloor + 1.$$

The convergence of Phase 2 in Algorithm 9.3.1 is stated in the following theorem.

Theorem 9.3.1. *The maximum number of iterations needed in Phase 2 of Algorithm 9.3.1 to obtain an ε_g - solution y^k of (8.1.9) does not exceed:*

$$\tilde{k}_{\max} := \left\lfloor \ln\left(\frac{t_0 \omega_*(\beta)}{\varepsilon_g}\right) \left[\ln\left(1 + \frac{\bar{\Delta}^*}{\sqrt{\nu}(\bar{\Delta}^* + 1)}\right)\right]^{-1} \right\rfloor + 1, \tag{9.3.6}$$

where $\bar{\Delta}^*$ is defined by (9.3.3).

Proof. From Step 2 of Algorithm 9.3.1, we have $t_k = (1-\sigma)^k t_0$. Hence, if $t_k \leq \frac{\varepsilon_g}{\omega_*(\beta)}$ then $k \geq \ln\left(\frac{\omega_*(\beta) t_0}{\varepsilon_g}\right) [\ln(1-\sigma)^{-1}]^{-1}$. However, since $(1-\sigma)^{-1} = 1 + \frac{\bar{\Delta}^*}{\sqrt{\nu}(\bar{\Delta}^*+1)}$, it implies from the previous relation that:

$$k \geq \ln\left(\frac{\omega_*(\beta) t_0}{\varepsilon_g}\right) \ln\left(1 + \frac{\bar{\Delta}^*}{\sqrt{\nu}(\bar{\Delta}^* + 1)}\right)^{-1},$$

which leads to (9.3.6). □

Remark 9.3.2 (**The worst-case complexity**). *Since $\ln\left(1 + \frac{\bar{\Delta}^*}{\sqrt{\nu}(\bar{\Delta}^*+1)}\right) \approx \frac{\bar{\Delta}^*}{\sqrt{\nu}(\bar{\Delta}^*+1)}$, the worst-case complexity of Algorithm 9.3.1 is $O\left(\sqrt{\nu} \ln\left(\frac{t_0}{\varepsilon_g}\right)\right)$ which is similar to Algorithm 9.2.2.*

Remark 9.3.3 (**Damped Newton iteration**). *Note that, at Step 5 of Algorithm 9.3.1, we can use a damped Newton iteration:*

$$y^{k+1} := y^k - \alpha_k \nabla^2 g(y^k; t_{k+1})^{-1} \nabla g(y^k, t_{k+1}),$$

instead of the full-step Newton iteration, where $\alpha_k = (1 + \lambda_{\tilde{g}(\cdot; t_{k+1})}(y^k))^{-1}$. In this case, with the same argument as before, we can compute $\beta^* = 0.5$ and $\Delta^* = \frac{\sqrt{0.5\beta} - \beta}{1 + \sqrt{0.5\beta}}$.

9.4 Discussion on implementation

In this section, we further discuss the implementation issues of the proposed algorithms.

Handling nonlinear objective function and local equality constraints

If the objective function ϕ_i in ($\text{SepCOP}_{\max}$) is nonlinear, concave and its epigraph is endowed with a self-concordant log-barrier for some $i \in \{1, \cdots, M\}$ then we propose to use a slack variable to move the objective function into the constraints and reformulate it as an optimization problem with linear objective function. By elimination of variables, it is not difficult to show that the optimality condition of the resulting problem collapses to the optimality condition of the original problem, i.e.:

$$\nabla\phi_i(x_i) + A_i^T y - t\nabla F_i(x_i) = 0.$$

The algorithms developed in the previous sections can be applied to solve such a problem without moving the nonlinear objective function into the constraints.

We also note that, in Algorithms 9.2.2 and 9.2.3, we need to solve approximately the primal subproblems in (8.1.7) up to a desired accuracy. Instead of solving directly these primal subproblems, we can treat them from the optimality condition (9.1.1). Since the objective function associated with this optimality condition is self-concordant, Newton-type methods can be applied to solve such a problem, see, e.g. [30, 142].

If local equality constraints $E_i x_i = f_i$ are considered in ($\text{SepCOP}_{\max}$) for some $i \in \{1, \cdots, M\}$, then the KKT conditions of the primal subproblem i become:

$$\begin{cases} c_i + A_i^T y + E_i^T z_i - t\nabla F_i(x_i) = 0, \\ E_i x_i - f_i = 0. \end{cases} \tag{9.4.1}$$

Instead of the full KKT system (9.4.1), we consider a reduced KKT condition as follows:

$$Z_i^T(c_i + A_i^T y) - tZ_i^T \nabla F_i(Z_i x_i^z + R_i^{-T} f_i) = 0.$$

Here, (Q_i, R_i) is a QR-factorization of E_i^T and $Q_i = [Y_i, Z_i]$ is a basis of the range space and the null space of E_i^T, respectively. Due to the invariance of the norm $\|\cdot\|_{x^*}$, we can show that $\|\bar{x}_{\bar{\delta}} - x^*\|_{x^*} = \left\|\bar{x}_{\bar{\delta}}^z - x^{z*}\right\|_{x^{z*}}$. Therefore, the condition (9.2.1) coincides with $\left\|\bar{x}_{\bar{\delta}}^z - x^{z*}\right\|_{x^{z*}} \leq \bar{\delta}$. However, the last condition

is satisfied if:

$$\left\|Z_i^T(c_i + A_i^T y) - tZ_i^T \nabla F_i(Z_i x_i^z + R_i^{-T} f_i)\right\|_{x_i^{z_c}}^* \leq \varepsilon_i(t),\ i = 1, \cdots, M.$$

Note that the QR-factorization of E_i^T is only computed once, a priori.

Computing the inexact perturbed Newton direction

Regarding the Newton direction in Algorithms 9.2.2 and 9.2.3, one has to solve the linear system:

$$\nabla^2 g_{\bar{\delta}}(y^k; t)\Delta y^k = -\nabla g_{\bar{\delta}}(y^k; t). \tag{9.4.2}$$

Here, the gradient vector $\nabla g_{\bar{\delta}}$ is computed as:

$$\nabla g_{\bar{\delta}}(y^k; t) = A\bar{x}_{\bar{\delta}}(y^k; t) - b = \sum_{i=1}^{M}(A_i\bar{x}_i(y^k; t) - b_i) := h_k,$$

and the Hessian matrix $\nabla^2 g_{\bar{\delta}}(y^k; t)$ is obtained from:

$$\nabla^2 g_{\bar{\delta}}(y^k; t) = t^{-1}\sum_{i=1}^{M} A_i \nabla^2 F_i(\bar{x}_i(y^k; t))^{-1} A_i^T := \sum_{i=1}^{M} G_i^k.$$

Note that each block $G_i^k := t^{-1}A_i\nabla^2 F_i(\bar{x}_i(y^k; t))^{-1}A_i^T$ can be computed *in parallel*. Then, the linear system (9.4.2) can be written as:

$$\Big(\sum_{i=1}^{M} G_i^k\Big)\Delta y^k = -h_k. \tag{9.4.3}$$

Sine matrix $G^k := \sum_{i=1}^{M} G_i^k \succ 0$, one can apply either Cholesky-type factorizations or conjugate gradient (CG) methods to solve (9.4.3). Note that the CG method only requires matrix-vector operations. More details on parallel solution of (9.4.3) can be found, e.g., in [135, 218].

9.5 Numerical tests

In this section, we test the algorithms developed in the previous sections by solving a routing problem with congestion cost. This problem appears in many areas including telecommunications, network and transportation [101].

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a network of $n_{\mathcal{N}}$ nodes and $n_{\mathcal{A}}$ links, and $\mathcal{C}$ be a set of $n_{\mathcal{C}}$ commodities to be sent through the network $\mathcal{G}$, where each commodity $k \in \mathcal{C}$

has a source $s_k \in \mathcal{N}$, a destination $d_k \in \mathcal{N}$ and a certain amount of demand r_k. The optimization model of the routing problem with congestion (RPC) can be formulated as follows (see, e.g. [101] for more details):

$$\begin{aligned} \min_{u_{ijk}, v_{ij}} \quad & \textstyle\sum_{k\in\mathcal{C}} \sum_{(i,j)\in\mathcal{A}} c_{ij} u_{ijk} + \sum_{(i,j)\in\mathcal{A}} w_{ij} g_{ij}(v_{ij}) \\ \text{s.t.} \quad & \textstyle\sum_{j:(i,j)\in\mathcal{A}} u_{ijk} - \sum_{j:(j,i)\in\mathcal{A}} u_{jik} = \begin{cases} r_k & \text{if } i = s_k, \\ -r_k & \text{if } i = d_k, \\ 0 & \text{otherwise,} \end{cases} \\ & \textstyle\sum_{k\in C} u_{ijk} - v_{ij} = b_{ij}, \ (i,j) \in \mathcal{A}, \\ & u_{ijk} \geq 0, \ v_{ij} \geq 0, \ (i,j) \in \mathcal{A}, \end{aligned} \tag{9.5.1}$$

where $w_{ij} \geq 0$ is the weighting of the additional cost function g_{ij} for $(i,j) \in \mathcal{A}$.

In this example we assume that the additional cost function g_{ij} is given by either a) $g_{ij}(v_{ij}) = -\ln(v_{ij})$, the logarithmic function or b) $g_{ij}(v_{ij}) = v_{ij}\ln(v_{ij})$, the entropy function. It was shown in [142] that the epigraph of g_{ij} possesses a standard self-concordant barrier a) $F_{ij}(v_{ij}, s_{ij}) = -\ln v_{ij} - \ln(\ln v_{ij} + s_{ij})$ or b) $F_{ij}(v_{ij}, s_{ij}) = -\ln v_{ij} - \ln(s_{ij} - v_{ij}\ln v_{ij})$, respectively.

By using slack variables s_{ij}, we can move the nonlinear terms of the objective function to the constraints. The objective function of the resulting problem becomes:

$$f(u, v, s) := \sum_{k\in\mathcal{C}} \sum_{(i,j)\in\mathcal{A}} c_{ij} u_{ijk} + \sum_{(i,j)\in\mathcal{A}} w_{ij} s_{ij}, \tag{9.5.2}$$

with additional constraints $g_{ij}(v_{ij}) \leq s_{ij}$, $(i,j) \in \mathcal{A}$. It is clear that problem (9.5.1) is separably convex. Let:

$$X_{ij} := \Big\{ v_{ij} \geq 0, \sum_{k\in\mathcal{C}} u_{ijk} - v_{ij} = b_{ij}, \ g_{ij}(v_{ij}) \leq s_{ij}, (i,j) \in \mathcal{A}, k \in \mathcal{C} \Big\}, \ (i,j) \in \mathcal{A}. \tag{9.5.3}$$

Then problem (9.5.1) can be reformulated in the form of ($\text{SepCOP}_{\max}$) with linear objective function (9.5.2) and the local constraint set (9.5.3). Moreover, the resulting problem has $M := n_{\mathcal{A}}$ components, $n := n_{\mathcal{C}} n_{\mathcal{A}} + 2n_{\mathcal{A}}$ variables including u_{ijk}, v_{ij} and s_{ij}; and $m := n_{\mathcal{C}} n_{\mathcal{N}}$ coupling constraints. Each primal subproblem (8.1.7) has $n_i := n_{\mathcal{C}} + 2$ variables and one local linear equality constraint.

The aim is to compare the effect of the inexactness on the performance of the algorithms. We consider two variants of Algorithm 9.2.2, where we set $\bar{\delta} = 0.5\bar{\delta}^*$ and $\bar{\delta} = 0.25\bar{\delta}^*$ in Phase 1 and $\bar{\delta} = 0.01$ and $\bar{\delta} = 0.005$ in Phase 2, respectively. We denote these variants by `A1-v1` and `A1-v2`, respectively. For Algorithm 9.3.1, we also consider two cases. In the first case we set the tolerance of the primal subproblems to $\varepsilon_p = 10^{-6}$, and the second one is 10^{-10} to which we

will refer to as `A3-v1` and `A3-v2`, respectively. All variants are terminated with the same tolerance $\varepsilon_d = 10^{-4}$. The initial penalty parameter value is set to $t_0 := 0.25$.

We benchmarked four variants with performance profiles as in the previous chapters. All the algorithms were implemented in C++ running on an Intel® Core TM2, Quad-Core Processor Q6600 (2.4GHz) PC Desktop with 3Gb RAM and were paralellized by using `OpenMP`. The input data is generated randomly, where the nodes of the network are generated in a rectangle $[0, 100] \times [0, 300]$, the demand r_k is in $[50, 500]$, the weighting vector w is set to 10, the congestion b_{ij} is in $[10, 100]$ and the linear cost c_{ij} is the Euclidean length of the link $(i, j) \in \mathcal{A}$. The nonlinear cost function g_{ij} is chosen randomly between two functions in a) and b) defined above with the same probability.

We tested the algorithms on a collection of 108 random problems. The size of these problems varies from $M = 6$ to $14,280$ components, $n = 84$ to $77,142$ variables and $m = 15$ to 500 coupling constraints. The performance profiles of the four algorithms in terms of computational time are shown in Figure 9.3, where the x-axis is the factor τ (not more than 2^τ-times worse than the best one) and the y-axis is the probability function values $\rho_s(\tau)$ (problems ratio).

As we can see from Figure 9.3 that Algorithm 9.2.2 performs better than Algorithm 9.3.1 both in the total computational time and the time for solving the primal subproblems. This provides an evidence on the effect of the inexactness on the performance of the algorithm. We also observed that the numbers of iterations for solving the master problem in Phase 1 of all variants are similar, while they are different in Phase 2. However, since Phase 2 is performed when the approximate point is in the quadratic convergence region, it requires few iterations toward the desired approximate solution. Therefore, the computational time of Phase 1 dominates Phase 2. We notice that, in this particular example, the structure of the master problem is almost dense and we did not use any sparse linear algebra solver.

We also compared the total number of iterations for solving the primal subproblems in Figure 9.4. It shows that Algorithm 9.2.2 is superior to Algorithm 9.3.1 in terms of iteration number, although the accuracy of solving the primal subproblem in Algorithm 9.3.1 is only set to 10^{-6} which is not exact as theoretically required. This performance profile also reveals the effect of the inexactness on the number of iterations. In our numerical results, the inexact version `A1-v1` saves 22% (resp. 23%) of the total number of iterations to solve the primal subproblems compared to `A3-v1` (resp. `A3-v2`); while the variant `A1-v2` saves 20% (reps. 21%) compared to `A3-v1` (resp. `A3-v2`).

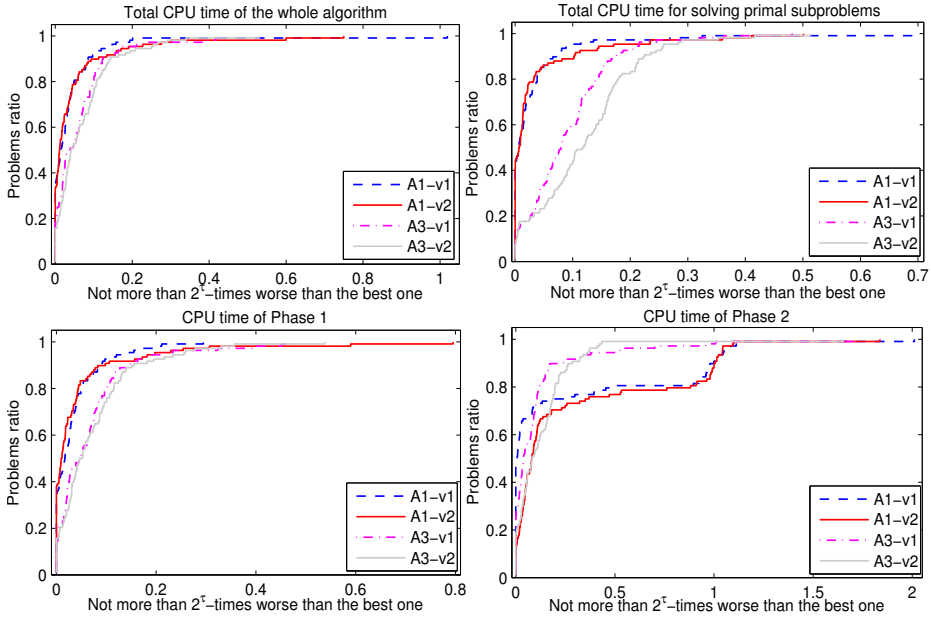


Figure 9.3: The performance profiles of the four variants in terms of computational time.

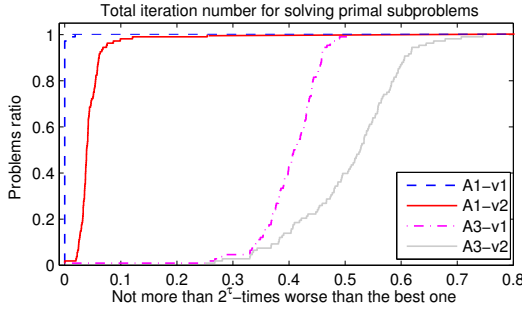


Figure 9.4: The performance profile of the four variants in terms of iteration number.

9.6 Conclusion

We have developed a two-phase interior point decomposition algorithm for solving some classes of separable convex optimization problems. A main

assumption imposed on the problem is that the objective function is self-concordant or compatible with the feasible set [146]. This requirement fits very well in some classes of convex programming problems such as conic programming, monotropic programming and network optimization problems. The proposed method possesses two key features. First, we have allowed the algorithm to solve the primal subproblems inexactly which leads to the inexactness in the gradient and Hessian of the smoothed dual function. Second, the parameters of the algorithm have appropriately been chosen via convergence analysis that allows us to control them in implementations. The worst-case complexity of the algorithm has been estimated and is similar to the one in standard interior-point decomposition methods. Without the inexactness, this algorithm collapses to an exact interior-point methods which is similar to several variants studied in the literature, see e.g. [113, 131, 135, 171, 218, 220] but still possesses some advantages compared to those. Numerical tests on a network routing problem with congestion have shown the efficiency of the methods compared with existing algorithms.

Chapter 10

Application to separable nonconvex optimization

The aim of this chapter is to design an optimization algorithm for solving *separable nonconvex* optimization problems of the form (SepNCOP) which can be implemented in *a parallel or distributed manner*. Unfortunately, due to *nonconvexity*, the *strong duality* condition in the Lagrangian duality framework is no longer preserved in this setting. The dual decomposition approach in the previous chapters can not be applied. Besides, conventional optimization methods such as SQP-type and interior-point algorithms usually require some global computation procedures such as evaluations of the objective function and constraints or globalization. These requirements are not suitable in a distributed implementation. In this chapter, we combine the *sequential convex programming* (SCP) framework developed in Part I and the algorithms proposed in the previous chapters to build a two-level decomposition algorithm for solving (SepNCOP). The SCP framework proposed in this chapter can be considered as a combination of the methods proposed in [141, 143], see also [189].

Contribution of Chapter 10. The contribution of this chapter is as follows:

a) We first propose a new sequential convex programming (SCP) scheme for solving separable nonconvex (possibly nonsmooth) optimization problems of the form (SepNCOP).

b) Then, we combine this SCP scheme and Algorithm 7.6.1 to obtain a two-level decomposition algorithm for solving (SepNCOP) and show the

convergence of this algorithm.

Outline of Chapter 10. This chapter consists of the following sections. In Section 10.1, we provide a sequential convex programming scheme to tackle problem (SepNCOP) and prove its global convergence. Section 10.2 presents a two-level decomposition algorithm for solving (SepNCOP). Section 10.3 deals with a numerical example to verify the proposed algorithm. We end this chapter by giving some conclusion.

10.1 Sequential convex programming approach for separable nonconvex optimization

Let us recall the separable nonconvex optimization problem defined by (SepNCOP) in this chapter for further references:

$$\phi^* := \begin{cases} \min\limits_{x\in\mathbf{R}^n} & \phi(x) := \sum\limits_{i=1}^{M}[g_i(x_i) + h_i(F_i(x_i))] \\ \text{s.t.} & \sum\limits_{i=1}^{M}(A_i x_i - b_i) = 0, \\ & x_i \in X_i,\ i = 1, \cdots, M, \end{cases} \tag{SepNCOP}$$

where x_i, A_i and b_i are defined as in (SepCOP) for $i = 1, \cdots, M$. The function $g_i : \mathbf{R}^{n_i} \to \mathbf{R}$ is assumed to be proper, lower semicontinuous, convex and possibly smooth, while $h_i : \mathbf{R}^{m_i} \to \mathbf{R}$ is proper, lower semicontinuous and convex but not necessarily smooth. The inner function $F_i : \mathbf{R}^{n_i} \to \mathbf{R}^{m_i}$ is continuously differentiable on its domain for $i = 1, \cdots, M$. We note that the functions g_i and h_i used in this chapter are different from the ones in the previous chapters although we use the same notation.

Optimality condition

Let us denote by $g := \sum_{i=1}^{m} g_i$, $h := \sum_{i=1}^{M} h_i$, $F := (F_1^T, \cdots, F_i^T)^T$, $X := X_1 \times \cdots \times X_M$ and $\Omega := \{x \in X \mid Ax - b = 0\}$. The optimality condition for problem (SepNCOP) can be expressed as:

$$0 \in \partial g(x^*) + F'(x^*)^T \partial h(F(x^*)) + \mathcal{N}_\Omega(x^*), \tag{10.1.1}$$

where $\mathcal{N}_\Omega(x)$ is the normal cone of the convex set Ω at x, $\partial g(\cdot)$ and $\partial h(\cdot)$ are the subdifferential of g and h at $(\cdot)$, respectively, and F' is the Jacobian mapping

of F. The condition (10.1.1) can be written equivalently as:

$$\partial h(F(x^*)) \cap \left\{v \mid -F'(x^*)^T v \in \partial g(x^*) + \mathcal{N}_\Omega(x^*)\right\} \neq \emptyset. \tag{10.1.2}$$

A point x^* satisfying the condition (10.1.1) or (10.1.2) is called a stationary point. We denote by Ω^* the set of stationary points of (SepNCOP).

Instead of (10.1.1), we consider an approximate optimality condition for (SepNCOP) as follows:

$$[\xi_g + F'(\tilde{x}^*)^T \xi_h]^T (u - \tilde{x}^*) \geq -\varepsilon, \quad \forall u \in \Omega, \tag{10.1.3}$$

where $\varepsilon \geq 0$ is a given tolerance, $\xi_g \in \partial g(\tilde{x}^*)$ and $\xi_h \in \partial h(F(\tilde{x}^*))$ are subgradients of g at $\tilde{x}^*$ and of $h(F(\cdot)$ at $F(\tilde{x}^*)$, respectively. In this case $\tilde{x}^*$ is called an *ε-approximate stationary point* of (SepNCOP).

Sequential convex programming scheme

Let $\mathcal{D}$ be a closed convex set in $\mathbf{R}^n$ with nonempty interior that contains Ω. We make the following assumptions:

Asumption A.10.1.12. *The function g is convex in $\mathcal{D}$. The function h is convex and L_h-Lipschitz continuous in $\mathbf{R}^m$, i.e.:*

$$|h(u) - h(v)| \leq L_h \|u - v\|, \quad \forall u, v \in \mathbf{R}^m.$$

The function F is differentiable in $\mathcal{D}$ and its Jacobian mapping is $L_{F'}$-Lipschitz continuous in $\mathcal{D}$, i.e.:

$$\|F'(x) - F'(\hat{x})\| \leq L_{F'} \|x - \hat{x}\|, \quad \forall x, \hat{x} \in \mathcal{D}.$$

As a simple example, the function $h(u) := \rho \|u\|$ is convex and Lipschitz continuous with a Lipschitz constant $L_h := \rho$ on $\mathbf{R}^m$ for any $\rho > 0$.

For a given $\bar{x}$ in $\mathcal{D}$, let us define the following partial linearization of the objective function of (SepNCOP):

$$\psi(x; \bar{x}) := g(x) + h(F(\bar{x}) + F'(\bar{x})(x - \bar{x})). \tag{10.1.4}$$

Since g and h are convex, $\psi(\cdot; \bar{x})$ is also convex. If, in addition, g is differentiable and its gradient is $L_{g'}$-Lipschitz continuous in $\mathcal{D}$ then we can consider:

$$\psi_L(x; \bar{x}) := g(\bar{x}) + \nabla g(\bar{x})^T (x - \bar{x}) + h(F(\bar{x}) + F'(\bar{x})(x - \bar{x})). \tag{10.1.5}$$

We have the following estimates.

Lemma 10.1.1. *Under Assumption **A.10.1.12**, the function $\psi(\cdot;\bar{x})$ defined by (10.1.4) satisfies:*

$$|\phi(x)-\psi(x;\bar{x})| \le \frac{M_\psi}{2}\|x-\bar{x}\|^2, \ \forall x\in\mathcal{D}, \tag{10.1.6}$$

where $M_\psi := L_h L_{F'} > 0$.

If, in addition, g is differentiable and its gradient is $L_{g'}$-Lipschitz continuous in $\mathcal{D}$ then the following estimate holds:

$$|\phi(x)-\psi_L(x;\bar{x})| \le \frac{M_{\psi_L}}{2}\|x-\bar{x}\|^2, \ \forall x\in\mathcal{D}, \tag{10.1.7}$$

where $M_{\psi_L} := L_{g'} + L_h L_{F'} > 0$.

Proof. Since F' is $L_{F'}$-Lipschitz continuous, for any $x,\ \bar{x}\in\mathcal{D}$, we have $\|F(x)-F(\bar{x})-F'(\bar{x})(x-\bar{x})\| \le \frac{1}{2}L_{F'}\|x-\bar{x}\|^2$. By using this estimate and the Lipschitz continuity of h, we have:

$$\begin{aligned}|\phi(x)-\psi(x;\bar{x})| &= |h(F(x)) - h(F(\bar{x})+F'(\bar{x})(x-\bar{x}))|\\ &\le L_h\|F(x)-F(\bar{x})-F'(\bar{x})(x-\bar{x})\|\\ &\le \frac{1}{2}L_h L_{F'}\|x-\bar{x}\|^2,\end{aligned}$$

which is indeed (10.1.6). Similarly, by the Lipschitz continuity of ∇g we have:

$$\begin{aligned}|\phi(x)-\psi_L(x;\bar{x})| &\le \left|g(x)-g(\bar{x})-\nabla g(\bar{x})^T(x-\bar{x})\right|\\ &\quad + |h(F(x)) - h(F(\bar{x})+F'(\bar{x})(x-\bar{x}))|\\ &\le \frac{L_{g'}}{2}\|x-\bar{x}\|^2 + L_h\|F(x)-F(\bar{x})-F'(\bar{x})(x-\bar{x})\|\\ &\le \frac{L_{g'}+L_h L_{F'}}{2}\|x-\bar{x}\|^2,\end{aligned}$$

which proves (10.1.7). □

For simplicity of presentation, we will use the formula ψ defined by (10.1.4) in the following SCP scheme. However, the obtained results remain true if we replace ψ by ψ_L defined by (10.1.5) with some modification.

Let $\bar{x}\in\Omega$ be a given point and $\beta > 0$. We define:

$$q(x;\bar{x},\beta) := \psi(x;\bar{x}) + \frac{\beta}{2}\|x-\bar{x}\|^2 \ \text{ and } \ v_0(\bar{x};\beta) := \operatorname*{argmin}_{x\in\Omega} q(x;\bar{x},\beta). \tag{10.1.8}$$

Since $q(\cdot;\bar{x},\beta)$ is strongly convex with a convexity parameter $\beta > 0$, v_0 is well-defined and single-valued. The optimality condition for problem (10.1.8) becomes:

$$\xi_q^T(u - v_0(\bar{x};\beta)) \geq 0, \quad \forall u \in \Omega,$$

for some $\xi_q \in \partial q(v_0(\bar{x};\beta);\bar{x},\beta)$, which is necessary and sufficient for optimality.

Now, we define the *proximal-gradient mapping* of $q(\cdot;\bar{x},\beta)$ as:

$$G_0(\bar{x};\beta) := \beta(\bar{x} - v_0(\bar{x};\beta)), \tag{10.1.9}$$

We also define the error norm and the optimal value of (10.1.8), respectively as:

$$e_0(\bar{x};\beta) := \|\bar{x} - v_0(\bar{x};\beta)\| \text{ and } \varphi_0(\bar{x};\beta) := \psi(v_0(\bar{x};\beta);\bar{x}) + \frac{\beta}{2}\|v_0(\bar{x};\beta) - \bar{x}\|^2.$$

In practice, we can not solve problem (10.1.8) *exactly*. We can only solve this problem up to a given accuracy $\varepsilon > 0$ to get:

$$v_\varepsilon(\bar{x};\beta) :\approx \operatorname*{argmin}_{x\in\Omega} q(x;\bar{x},\beta), \tag{10.1.10}$$

as in the sense of the following definition.

Definition 10.1.1. *For a given tolerance $\varepsilon \geq 0$, a point v_ε is said to be an ε-solution to (10.1.8) if:*

$$\xi_{q\varepsilon}^T(u - v_\varepsilon) \geq -\varepsilon, \quad \forall u \in \Omega, \tag{10.1.11}$$

for some $\xi_{q\varepsilon} \in \partial q(v_\varepsilon;\bar{x},\beta)$, where $\partial q(v_\varepsilon;\bar{x},\beta)$ is the subdifferential of $q(\cdot;\bar{x},\beta)$ at v_ε which can be computed as:

$$\partial q(v_\varepsilon;\bar{x},\beta) := \partial g(v_\varepsilon) + F'(\bar{x})^T \partial h(F(\bar{x}) + F'(\bar{x})(v_\varepsilon - \bar{x})) + \beta(v_\varepsilon - \bar{x}).$$

Alternatively to G_0 and e_0, we define the *approximate proximal-gradient mapping*, the approximate error norm and the approximate optimal value of (10.1.8), respectively as:

$$G_\varepsilon(\bar{x};\beta) := \beta(\bar{x} - v_\varepsilon(\bar{x};\beta)), \tag{10.1.12}$$

and

$$e_\varepsilon(\bar{x};\beta) := \|\bar{x} - v_\varepsilon(\bar{x};\beta)\| \text{ and } \varphi_\varepsilon(\bar{x};\beta) := \psi(v_\varepsilon(\bar{x};\beta);\bar{x}) + \frac{\beta}{2}\|v_\varepsilon(\bar{x};\beta) - \bar{x}\|^2.$$

The following lemma shows the properties of $\|G_0(\bar{x};\cdot)\|$ and $e_0(\bar{x};\cdot)$.

Lemma 10.1.2. *The function $\|G_0(\bar{x};\cdot)\|$ is nondecreasing in $\mathbf{R}_{++}$ and $e_0(\bar{x};\cdot)$ is nonincreasing in $\mathbf{R}_{++}$. Moreover, we have:*

$$\phi(\bar{x}) - \varphi_\varepsilon(\bar{x};\beta) \geq \frac{\beta}{2} e_\varepsilon(\bar{x};\beta)^2 - \sqrt{2\beta\varepsilon}\, e_\varepsilon(\bar{x};\beta). \tag{10.1.13}$$

Proof. Since the function $k(t,x) := \psi(x;\bar{x}) + \frac{1}{2t}\|x-\bar{x}\|^2$ is convex w.r.t. two variables x and t. We have $\eta(t) := \min_{x\in\Omega} k(t,x)$ is still convex. It is easy to show that $\eta'(t) = -\frac{1}{2t^2}e_0(\bar{x};1/t)^2 = -\frac{1}{2}\|G_0(\bar{x};1/t)\|^2$. Since $\eta(t)$ is convex, $\eta'(t)$ is nondecreasing in t. This implies that $\|G_0(\bar{x};1/t)\|$ is nonincreasing in t. Thus $\|G_0(\bar{x};\beta)\|$ is nondecreasing in β and $\|e_0(\bar{x};\beta)\|$ is nonincreasing in β.

From the convexity of η, we have $\phi(\bar{x}) = \eta(0) \geq \eta(t) + \eta'(t)(0-t) = \eta(t) + \frac{1}{2t}e_0^2(\bar{x};1/t)$. On the other hand, we have $\varphi_0(\bar{x};\beta) = \eta(1/\beta)$. Substituting this relation into the last inequality we obtain:

$$\phi(\bar{x}) - \varphi_0(\bar{x};\beta) \geq \frac{\beta}{2}e_0(\bar{x};\beta)^2. \tag{10.1.14}$$

For simplicity of notation, we denote by $v_0 := v_0(\bar{x};\beta)$ and $v_\varepsilon := v_\varepsilon(\bar{x};\beta)$. From the strong convexity of $q(\cdot;\bar{x},\beta)$ and (10.1.11), we can show that:

$$\frac{\beta}{2}\|v_\varepsilon - v_0\|^2 \leq \varphi_\varepsilon(\bar{x};\beta) - \varphi_0(\bar{x};\beta) \leq \varepsilon. \tag{10.1.15}$$

This inequalities imply $\|v_\varepsilon - v_0\| \leq \sqrt{2\varepsilon/\beta}$. Using the last inequality we can estimate $e_0(\bar{x};\beta)^2 = \|v_0-\bar{x}\|^2 \geq |e_\varepsilon(\bar{x};\beta) - \|v_\varepsilon - v_0\||^2 \geq e_\varepsilon(\bar{x};\beta)^2 - 2\sqrt{2\varepsilon/\beta}e_\varepsilon(\bar{x};\beta) + 2\varepsilon/\beta$. Substituting the last inequality into (10.1.14) and then using (10.1.15) we obtain (10.1.13). □

Lemma 10.1.2 shows that if we choose the regularization parameter too large then $\|G_0(\bar{x};\cdot)\|$ may increase while the error $e_0(\bar{x};\cdot)$ may decrease. This makes a slow progress towards a stationary point of problem (SepNCOP). For ε sufficiently small, the inequality (10.1.13) provides a locally approximate quadratic bound for the objective function ϕ of (SepNCOP).

The following statement is obvious and can be obtained directly from the approximate optimality conditions (10.1.11) and (10.1.3).

Lemma 10.1.3. *If $\bar{x}$ is a fixed point of the mapping $v_\varepsilon(\cdot;\beta)$, i.e. $\bar{x} = v_\varepsilon(\bar{x};\beta)$ then it is an ε- stationary point of* (SepNCOP).

Now, we are ready to prove a main estimate which will be used to show the global convergence of the SCP scheme described below.

Lemma 10.1.4. *Suppose that Assumption **A.10.1.12** is satisfied and $\bar{x} \in \Omega$ is a given point and $\beta > 0$. Then the point $v_\varepsilon(\bar{x};\beta)$ defined by* (10.1.8) *satisfies the estimate:*

$$\phi(\bar{x}) - \phi(v_\varepsilon(\bar{x};\beta)) \geq \frac{2\beta - M_\psi}{2}e_\varepsilon(\bar{x};\beta)^2 - \varepsilon = \frac{2\beta - M_\psi}{2\beta^2}\|G_\varepsilon(\bar{x};\beta)\|_*^2 - \varepsilon, \tag{10.1.16}$$

where $M_\psi := L_h L_{F'}$.

Proof. Let us denote by $v_\varepsilon := v_\varepsilon(\bar{x};\beta)$. By using (10.1.6), we have:

$$\phi(v_\varepsilon) \le g(v_\varepsilon) + h(F(\bar{x}) + F'(\bar{x})(v_\varepsilon - \bar{x})) + \frac{M_\psi}{2}\|v_\varepsilon - \bar{x}\|^2. \tag{10.1.17}$$

Now, since g and h are convex, for any v we have:

$$\begin{aligned} &g(\bar{x}) - g(v) \ge \xi_g(v)^T(\bar{x} - v), \\ &h(F(\bar{x})) - h(F(\bar{x}) + F'(\bar{x})(v - \bar{x})) \ge \xi_h(F)^T F'(\bar{x})(\bar{x} - v), \end{aligned} \tag{10.1.18}$$

where $\xi_h(F) \in \partial h(F(\bar{x}) + F'(\bar{x})(v - x))$. Next, we consider the approximate optimality condition (10.1.11) of (10.1.8). By letting $u = \bar{x}$ into (10.1.11), then combining the result and (10.1.18), we obtain:

$$\begin{aligned} g(\bar{x}) + h(F(\bar{x})) &\ge g(v_\varepsilon) + h(F(\bar{x}) + F'(\bar{x})(v_\varepsilon - \bar{x})) \\ &\quad + \left[\xi_g(v_\varepsilon) + F'(\bar{x})^T\xi_h(F)\right]^T(\bar{x} - v_\varepsilon) \\ &\overset{(10.1.11)}{\ge} g(v_\varepsilon) + h(F(\bar{x}) + F'(\bar{x})(v_\varepsilon - \bar{x})) + \beta\|v_\varepsilon - \bar{x}\|^2 - \varepsilon. \end{aligned} \tag{10.1.19}$$

Now, by using the Lipschitz continuity of h and F' in Assumption **A.10.1.12**, for any v we have:

$$\begin{aligned} h(F(v)) - h(F(\bar{x}) + F'(\bar{x})(v - \bar{x})) &\le L_h\|F(v) - F(\bar{x}) - F'(\bar{x})(v - \bar{x})\| \\ &\le \frac{M_\psi}{2}\|v - \bar{x}\|^2. \end{aligned}$$

Substituting this inequality with $v = v_\varepsilon$ into (10.1.19) we obtain:

$$g(\bar{x}) + h(F(\bar{x})) - g(v_\varepsilon) - h(F(v_\varepsilon)) \ge \frac{2\beta - M_\psi}{2}\|v_\varepsilon - \bar{x}\|^2 - \varepsilon. \tag{10.1.20}$$

Finally, by using the definitions of $e_\varepsilon(\bar{x};\beta)$, $G_\varepsilon(\bar{x};\beta)$ and $\phi(\cdot)$, it follows from the last inequality that (10.1.16) holds. □

Next, we define a sublevel set of $\phi(\cdot)$ restricted to Ω as:

$$\mathcal{L}_\phi(\alpha) := \{x \in \Omega \mid \phi(x) \le \alpha\}. \tag{10.1.21}$$

Then we have the following statement whose the proof can be done similarly as the proof of [143, Lemma 2.5].

Lemma 10.1.5. *Let $\varepsilon \ge 0$ be given. Suppose that $\mathcal{L}_\phi(\phi(\bar{x})) + \varepsilon\mathcal{B}(0,1) \subseteq \mathrm{int}(\mathcal{D})$. Then, if $\beta \ge M_\psi$ then $v_\varepsilon(\bar{x};\beta) \in \mathcal{L}_\phi(\phi(\bar{x}))$.*

Now, we can describe one step of the SCP algorithm as follows. The SCP algorithm generates a sequence $\{x^k\}_{k\geq 0}$ starting from $x^0 \in \Omega$ by applying the following scheme:

Scheme S.10.1.1.(SCP scheme**).**

- Fix a constant $\mu \in (0,1)$. Choose $x^0 \in \Omega$ and $\varepsilon_0 > 0$ sufficiently small.
- For $k = 0, 1, 2, \ldots,$ perform:

$$x^{k+1} := v_{\varepsilon_k}(x^k; \beta_k), \tag{10.1.22}$$

where $M_\psi \leq \beta_k \leq \bar{\beta}$ and $\varepsilon_{k+1} := \min\left\{\varepsilon_k, \frac{\mu M_\psi}{2}\left\|x^{k+1} - x^k\right\|^2\right\}$.

End.

The following theorem shows that the sequence $\left\{x^k\right\}_{k\geq 0}$ generated by Scheme **S.10.1.1** converges to a stationary point $x^* \in \Omega^*$.

Theorem 10.1.1. *Let $\{x^k\}_{k\geq 0}$ be a sequence generated by Scheme **S.10.1.1**. Then:*

$$\phi(x^k) - \phi^* \geq \frac{(1-\mu)M_\psi}{2}\sum_{j=k}^{\infty} e_{\varepsilon_j}(x^j;\beta_j)^2 - \varepsilon_k, \tag{10.1.23}$$

where $\varepsilon_k \geq 0$ and ϕ^* is the optimal value of (SepNCOP) in $\mathcal{L}_\phi(\phi(x^0))$. Consequently, one has $\lim_{k\to\infty}\left\|x^{k+1} - x^k\right\| = 0$ and the set of limit points Ω^* of $\{x^k\}$ is either empty or nonempty and connected. Suppose further that the sublevel set $\mathcal{L}_\phi(\phi(x^0))$ is bounded from below. Then every limit point of $\left\{x^k\right\}$ is a stationary point of (SepNCOP). Moreover, if Ω^* is finite then the whole sequence $\left\{x^k\right\}_{k\geq 0}$ converges to a point x^* in Ω^*.

Proof. From Lemma 10.1.4 and the condition $\beta_k \geq M_\psi$ we have:

$$\phi(x^k) - \phi(x^{k+1}) \geq \frac{2\beta_k - M_\psi}{2}e_{\varepsilon_k}(x^k;\beta)^2 - \varepsilon_k \geq \frac{M_\psi}{2}e_{\varepsilon_k}(x^k;\beta_k)^2 - \varepsilon_k,\ \forall k \geq 0.$$

Summing up this inequality from $j = k$ to $j = N \geq k$ with noting that $\varepsilon_k = \min\left\{\frac{\mu M_\psi}{2}\left\|x^k - x^{k-1}\right\|^2, \varepsilon_{k-1}\right\} \leq \frac{\mu M_\psi}{2}\left\|x^k - x^{k-1}\right\|^2$ for $k \geq 1$ we get:

$$\phi(x^k) - \phi(x^{N+1}) \geq \sum_{j=k}^{N-1}(1-\mu)\frac{M_\psi}{2}e_{\varepsilon_j}(x^j;\beta_j)^2 + \frac{M_\psi}{2}e_{\varepsilon_N}(x^N;\beta_N) + \frac{M_\psi}{2}e_{\varepsilon_k}(x^k;\beta_k) - \varepsilon_k$$

$$\geq \frac{(1-\mu)M_\psi}{2}\sum_{j=k}^{N} e_{\varepsilon_j}(x^j;\beta_j)^2 - \varepsilon_k + \varepsilon_{N+1}, \tag{10.1.24}$$

which can be rewritten as:

$$[\phi(x^k)+\varepsilon_k]-[\phi(x^{N+1})+\varepsilon_{N+1}]\geq\frac{(1-\mu)M_\psi}{2}\sum_{i=k}^{N}e_{\varepsilon_j}(x^j;\beta_j)^2. \tag{10.1.25}$$

Since the sequence $\{\phi(x^k)+\varepsilon_k\}$ is bounded from below and the sequence $\{\varepsilon_k\}$ does not increase, passing to the limit $N\to\infty$ in (10.1.24) we obtain (10.1.23). Next, we set $k=0$ in (10.1.24) and passing to the limit $N\to\infty$ we have $\sum_{j=0}^{\infty}e_{\varepsilon_j}(x^j;\beta_j)^2<+\infty$. Therefore, $\lim_{k\to\infty}\|x^k-x^{k+1}\|=0$. This limit implies that the set of limit points of $\{x^k\}_{k\geq0}$ is either empty or nonempty and connected.

Since the approximate level set $\mathcal{L}_\phi(\phi(x^0))$ is bounded, by Lemma 10.1.5, we conclude that the sequence $\{x^k\}_{k\geq0}$ is bounded. Thus the set of limit points Ω^* is nonempty. Next, by passing to the limit through a subsequence and then combining the result and Lemma 10.1.3 we can easily prove that every limit point is a stationary point of (SepNCOP). If the set of limit points Ω^* is finite, by applying the result [152, Chapt. 28], we obtain the proof of the remaining statement. □

10.2 Two-level decomposition algorithm

Recall that the primal subproblem (10.1.10) in the SCP scheme **S.10.1.1** is convex and separable. In principle, we can apply the algorithms developed in the previous chapters to obtain a two-level decomposition algorithm for solving (SepNCOP). In particular, since the objective function of (10.1.10) is strongly convex with a convexity parameter $\beta>0$, in the following algorithm, we apply Algorithm 7.6.1 in Chapter 7 to solve (10.1.10).

Problem (10.1.10) can be written explicitly as follows:

$$\begin{aligned}\min_x\ &\sum_{i=1}^{M}\left[g_i(x_i)+h_i(F_i(\bar{x}_i)+F_i'(\bar{x}_i)(x_i-\bar{x}_i))+\frac{\beta}{2}\|x_i-\bar{x}_i\|^2\right]\\ \text{s.t.}\ &\sum_{i=1}^{M}(A_ix_i-b_i)=0,\\ &x_i\in X_i,\ i=1,\cdots,M.\end{aligned} \tag{10.2.1}$$

This problem is in fact in the form of (SepCOP). Therefore, we combine the SCP scheme **S.10.1.1** and Algorithm 7.6.1 to obtain a two-level algorithm for solving (SepNCOP). In this case, we also assume that the Lipschitz constants L_h and $L_{F'}$ are known *a priori*. The algorithm is described in detail as follows:

Algorithm 10.2.1.(*Two-level SCP decomposition algorithm*).
Initialization: Perform the following steps:

1. Given a tolerance $\varepsilon_{\text{outer}} > 0$ for the outer loop.
2. Choose positive numbers $\bar{\beta} > \underline{\beta} \geq M_\psi := L_h L_{F'}$ and $\mu \in (0, 1)$.
3. For each component i, choose an initial point $x_i^0 \in X_i$ $(i = 1, \cdots, M)$.
4. Select a sufficiently small accuracy value $0 < \varepsilon_0 < \varepsilon_{\text{outer}}$.

Outer iteration: For $k = 0, 1, 2, \ldots,$ perform the following steps:

Step 1: Select an appropriate value $\beta_k \in [\underline{\beta}, \bar{\beta}]$.

Step 2: (inner iteration). For a given $x^k := (x_1^k, \cdots, x_M^k)$, apply Algorithm 7.6.1 to solve the *separable and strongly convex programming problem* (10.2.1) up to the given accuracy ε_k to obtain a solution x^{k+1}.

Step 3: If $\left\|x_i^{k+1} - x_i^k\right\| \leq \varepsilon_{\text{outer}}$ for $i = 1, \cdots, M$ then terminate the outer loop k.

Step 4: For $i = 1, \cdots, M$, evaluate the function F_i and its Jacobian at the new point x_i^{k+1} *in parallel*. Compute the new accuracy $\varepsilon_{k+1} := \min\left\{\frac{\mu M_\psi}{2}\left\|x^{k+1} - x^k\right\|^2, \varepsilon_k\right\}$.

End.

By strong convexity of (10.2.1), the inner-loop carried out by Algorithm 10.2.1 at Step 2 converges sublinearly with the convergence rate $O(1/j^2)$. Therefore, we can terminate the inner-loop after a certain number of iterations which can be defined *a priori*. The convergence of the inner-loop has been proved in Theorem 7.6.1.

The following theorem shows the convergence of Algorithm 10.2.1 which can be considered as a consequence of Theorem 10.1.1.

Theorem 10.2.1. *Under the assumptions of Theorem 10.1.1, the sequence* $\left\{x^k\right\}_{k\geq 0}$ *generated by Algorithm 10.2.1 still satisfies the conclusions of Theorem 10.1.1.*

Note that if g is differentiable and its gradient is $L_{g'}$-Lipschitz continuous in $\mathcal{D}$ then we can modify Algorithm 10.2.1 to process this case. The computation of the accuracy ε_k in Algorithm 10.2.1 still requires global information, the norm of the difference of the vectors x^k and x^{k+1}.

10.3 Numerical tests

In this section, we verify Algorithm 10.2.1 by applying it to solve the following separable nonconvex optimization problem:

$$\begin{cases} \min\limits_{x\in\mathbf{R}^n} & \phi(x) := \sum\limits_{i=1}^{n}\left[\frac{1}{2}p_i x_i^2 + q_i x_i + \rho\left|\frac{1}{2}c_i x_i^2 + d_i x_i + e_i\right|\right] \\ \text{s.t.} & \sum\limits_{i=1}^{n}(A_i x_i - b_i) = 0, \\ & 0 \le x_i \le 1,\ i = 1,\cdots,n. \end{cases} \tag{10.3.1}$$

Here p, q, c, d, e are given vectors in $\mathbf{R}^n$ with the components p_i, q_i, c_i, d_i, e_i, respectively, and $p_i \ge 0$ for $i = 1,\ldots,n$; $A := [A_1,\ldots,A_n] \in \mathbf{R}^{m\times n}$, $b_i \in \mathbf{R}^m$ for $i = 1,\ldots,n$ and $\rho > 0$ are given.

Let us define $g(x) := \sum_i^n(\frac{1}{2}p_i x_i^2 + q_i x_i)$, $h(u) := \rho\|u\|_1$, $F(x) := (\frac{1}{2}c_1 x_1^2 + d_1 x_1 + e_1,\cdots,\frac{1}{2}c_n x_n^2 + d_n x_n + e_n)^T$, $b := \sum_{i=1}^n b_i$ and $X := [0,1]^n$. Since $p_i \ge 0$ for all $i = 1,\ldots,n$, g is convex. It is also easy to check that h is convex and Lipschitz continuous with a Lipschitz constant $L_h := \rho$, and $F'(x) := \mathrm{diag}(c_1 x_1 + d_1,\cdots,c_n x_n + d_n)$ is also Lipschitz continuous with a Lipschitz constant $L_{F'} := \max_{1\le i\le n}|c_i|$. Hence, problem (10.3.1) can be written in the form (SepNCOP).

We have implemented Algorithm 10.2.1 in C++ running on a 16 cores Intel ®Xeon 2.7GHz workstation with 12 GB of RAM. The inner loop at Step 2 of Algorithm 10.2.1 was parallelized by using `OpenMP`. We terminated the outer loop of the algorithm if the relative feasibility gap $\texttt{rfgap} := \|Ax^k - b\| / \max\{\|Ax^0 - b\|, 1.0\} \le 10^{-3}$ and either $\texttt{error} := \|x^{k+1} - x^k\| / \max\{\|x^k\|, 1.0\} \le 10^{-3}$ or the quantity:

$$\mathrm{rfval}_{kj} := |\phi(x^k) - \phi(x^{k-j})| / \max\{|\phi(x^k)|, 1.0\}$$

does not change significantly after five successive iterations, i.e. $\texttt{rfval}_{kj} \le 10^{-3}$ for $j = 1,\ldots,5$. The initial tolerance ε_0 for the inner loop was set to $\varepsilon_0 := 0.5\times 10^{-3}$, and then was updated by $\varepsilon_{k+1} := \left\{\frac{0.5M_\psi}{2}\|x^{k+1} - x^k\|^2, \varepsilon_k\right\}$. The maximum number of iterations in the inner loop was set to $j_{\max} := 10,000$. The parameter β_k was fixed at $\beta_k := 1.005M_\psi$. We notice that the primal subproblems formed from each component of (10.2.1) in this example can be solved in a *closed form*.

The data of the problem instances was generated as follows. Vectors d, e, q and matrix A were generated randomly in $[-1, 1]$. Vectors c and p were also generated randomly in $[-0.5, 0.5]$ and $[0, 1]$, respectively. Vector b was computed

by $b := Ax_t$ for a given test point $x_t \in [0,1]^n$. The penalty parameter ρ was fixed at $\rho := 10$.

We have tested Algorithm 10.2.1 for 25 problem instances. The performance information and results reported by this algorithm are shown in Table 10.1.

Table 10.1: Performance information and results of Algorithm 10.2.1

P_{n^o}	Size		Performance			Results		
	m	n	`oiter`	`iiter`	`time[s]`	`error`	`rfgap`	`objval`
#1	50	1000	11	1868	5.31	9.012×10^{-3}	0.475×10^{3}	3539.450
#2	100	1000	11	2149	9.15	6.775×10^{-3}	0.586×10^{3}	3469.270
#3	150	1000	11	2749	16.91	4.566×10^{-3}	0.558×10^{3}	3685.160
#4	200	1000	11	3143	23.82	7.307×10^{-3}	0.606×10^{3}	3929.740
#5	250	1000	11	3464	30.48	8.298×10^{-3}	0.619×10^{3}	3940.790
#6	300	1000	11	3852	40.08	5.861×10^{-3}	0.587×10^{3}	3973.160
#7	350	1000	11	4004	47.76	5.299×10^{-3}	0.723×10^{3}	4259.940
#8	400	1000	11	4114	54.33	5.174×10^{-3}	0.731×10^{3}	4486.220
#9	450	1000	11	4711	67.86	5.831×10^{-3}	0.740×10^{3}	4730.130
#10	500	1000	11	4885	78.33	8.147×10^{-3}	0.691×10^{3}	4848.320
#11	550	2000	11	4804	226.33	8.743×10^{-3}	0.672×10^{3}	8324.100
#12	600	2000	11	5159	316.39	5.292×10^{-3}	0.623×10^{3}	8087.260
#13	650	2000	11	5255	366.47	7.940×10^{-3}	0.667×10^{3}	8373.030
#14	700	2000	11	5645	402.23	7.805×10^{-3}	0.707×10^{3}	8976.730
#15	750	2000	11	5898	493.21	5.860×10^{-3}	0.668×10^{3}	8530.690
#16	800	3000	11	6184	974.17	8.942×10^{-3}	0.648×10^{3}	12270.300
#17	850	3000	11	6103	788.78	6.926×10^{-3}	0.674×10^{3}	12361.500
#18	900	3000	11	6214	900.54	6.939×10^{-3}	0.661×10^{3}	12305.300
#19	950	3000	11	6444	1029.44	7.514×10^{-3}	0.662×10^{3}	12469.400
#20	1000	3000	11	6666	1076.14	7.629×10^{-3}	0.714×10^{3}	13007.200
#21	1000	3500	11	6404	1364.64	6.842×10^{-3}	0.703×10^{3}	14458.000
#22	1000	4000	11	6721	1482.57	10.381×10^{-3}	0.648×10^{3}	16178.600
#23	1000	4250	11	6847	1317.29	7.160×10^{-3}	0.615×10^{3}	16349.500
#24	1000	4500	11	6668	1552.98	7.750×10^{-3}	0.635×10^{3}	17722.600
#25	1000	5000	11	6777	1973.95	7.895×10^{-3}	0.611×10^{3}	19873.700

Here, the first column is the problem number; `oiter` is the number of outer iterations; `iiter` is the number of average inner iterations in Algorithm 7.6.1; `time[s]` is the total of computational time in seconds; two quantities `rfgap` and `error` are defined as above; and `objval` is the objective values.

As we can observe from Table 10.1 that the number of outer iterations in Algorithm 10.2.1 is rather small and does not change when the size of the problem increases. However, since the algorithm of the inner loop is just a first-order method, it requires many iterations and the number of iterations increases significantly when the problem size increases. Note that we can reduce the number of inner iterations by reducing the accuracy ε_0. In contrast, the number of outer iterations increases accordingly.

10.4 Conclusion

In this chapter, we have presented a two-level decomposition algorithm for solving a class of separable nonconvex optimization problems. This algorithm can be considered as a combination of an inexact SCP scheme in the context of Part I for nonconvex optimization and the decomposition algorithm for separable and strongly convex optimization, Algorithm 7.6.1, in Chapter 7. We have proved the global convergence of the SCP outer loop and, consequently, we have obtained the convergence of the whole algorithm. This algorithm has been tested via a numerical example. However, the theoretical and numerical results presented in this chapter are still preliminary due to several mathematical challenges of nonconvex optimization problems.

Chapter 11

Conclusion

11.1 Conclusion

In this thesis we have integrated two structure-exploiting approaches for solving nonlinear optimization problems both in the convex and nonconvex case, namely *sequential convex programming* and *decomposition methods*, and extended the existing state-of-the-art by new algorithms and convergence proofs.

Part I: Sequential Convex Programming (SCP)

In Chapter 2, we have proposed a general *adjoint-based predictor-corrector SCP algorithm* and two variants for solving parametric optimization problems as well as nonlinear optimization problems. We proved the stability of the tracking error for the online SCP algorithm and the local convergence of the SCP algorithm. These methods are suitable for nonconvex problems that possess convex substructures which can be efficiently handled by using convex optimization techniques. The performance of the algorithms has been validated by two numerical examples in nonlinear model predictive control as well as optimal control in Chapter 3. The basic assumptions used in our development are the strong regularity, the Lipschitz continuity assumption **A.2.4.3**b) and the approximation assumption **A.2.4.3**a) or **A.2.4.3'**. The *strong regularity* concept was introduced by Robinson in [160] and has been widely used in optimization and nonlinear analysis, while the two last assumptions are needed in any Newton-type algorithm. As in SQP methods, these assumptions involve some Lipschitz constants that may be difficult to determine in practice.

In Chapter 4, we have proposed a new algorithm for solving a class of nonconvex semidefinite programming problems, which can be viewed as a *generalization* of the *inner convex approximation method* [9, 127]. The key idea is to locally approximate the nonconvex feasible set of the problem by a sequence of inner positive semidefinite (psd)-convex sets. As a special case, we derived a new variant of this algorithm which we call a *generalized convex-concave decomposition* algorithm. This algorithm covers both the difference of two convex functions algorithm [156] and the convex-concave procedure [177] in the case of scalar functions as special cases. The convergence of both algorithms has been investigated under standard assumptions usually used in nonconvex semidefinite programming. Both algorithms are easy to implement by using available semidefinite programming solvers. We have shown that these algorithms are suitable to treat optimization problems with bilinear matrix inequality (BMI) constraints. However, the second algorithm depends crucially on the psd-convex-concave decomposition of the given BMI constraints. In practice, it is important to exploit the specific structure of the problems and find an appropriate psd-convex-concave decomposition for this algorithm. The methods developed in Chapter 4 can be extended to solve general nonlinear semidefinite programming problems, where either an inner convex approximation or a psd-convex-concave decomposition of the nonconvex mappings is available. Numerical tests in static feedback controller design presented in Chapter 5 confirmed the theoretical development.

Part II: Decomposition in separable optimization

In Chapter 6, we have briefly reviewed the related existing methods for solving separable optimization problems both in the convex and nonconvex case. Then we recalled the Lagrangian dual decomposition for separable convex optimization problems and some concepts in parallel and distributed computing mechanism and performance profiles which have been used throughout Chapters 7-10.

In Chapter 7, two new decomposition algorithms for large-scale separable convex optimization have been proposed. These algorithms were designed based on three techniques, namely dual decomposition, *smoothing via proximity-functions* and *excessive gap*. The convergence of both algorithms has been proved and the worst-case complexity bound has been established. The main advantage of these algorithms is that they automatically update the smoothness parameters without using any tuning strategy. This allows one to control the step-size of the algorithms in order to generate a larger step at the first iterations towards a solution. Although the global convergence rate is still sublinear, i.e. $O(1/k)$, where k is the iteration counter, the computational results are remarkable, especially when the number of variables as well as the number of components

increase. Two switching variants of the proposed algorithms were obtained. As a special case, the convergence of the second algorithm can be accelerated up to $O(1/k^2)$ when it is applied to solve strongly convex programming problems. This is similar to existing fast gradient methods in [142]. Extensions to inexact cases have also been investigated. In these algorithms we allowed that one solves the primal subproblems inexactly which is always the case in any practical implementation. From a theoretical point of view, the algorithms possess a good worst-case performance, due to the use of automatic strategies in updating the algorithm parameters.

In Chapter 8, we proposed a new path-following gradient-based decomposition algorithm for solving separable convex optimization problems. This algorithm is also based on three techniques, namely dual decomposition, *path-following* and *smoothing via self-concordant barriers*. The convergence of the algorithm has been proved and the rate of local convergence has been estimated. We have also adapted Nesterov's fast gradient scheme to this framework. We obtained a new variant of the fast gradient method for solving separable convex optimization problems which has the worst-case complexity $O(1/\varepsilon)$, where ε is a given accuracy. Both algorithms have the following advantages. First, the convergence rate of Algorithm 8.3.1 is $O(1/k)$ which is faster than the subgradient methods of multipliers recently considered in [61, 136]. Second, the worst-case complexity bound of the algorithms does not depend on the size of the feasible set as the one in [134]. It only depends on the size of the problem. Moreover, we can solve the primal subproblems of each component in both algorithms via a system of generalized equations instead of general convex programs as in the previous chapter.

In Chapter 9, we took a closer look at the structure of the objective function of the separable convex optimization problem where we assumed that this function is self-concordant. It allows us to both apply smoothing techniques via self-concordant barriers and the path-following method based on Newton-type iterations. We have proposed a path-following algorithm with inexact perturbed Newton iterations. The convergence of the algorithm has been analyzed and its complexity has been estimated. The theory presented in this chapter is significant in practice, since it allows us to solve the primal subproblems inexactly. Moreover, we can balance between the accuracy of solving the primal subproblem and the convergence rate of the path-following algorithm. As a special case, we have obtained again the path-following methods studied by Zhao [218], Mehrotra *et al* [131], Shida [171] and Necoara and Suykens [133]. However, by analyzing directly the path-following scheme, we can optimally choose the algorithm parameters compared to those methods in [131, 133, 171, 218]. Numerical tests and comparisons have been implemented to verify the theoretical development.

In Chapter 10, we have shown that we can apply the algorithms proposed in the previous chapters to solve separable nonconvex optimization problems. By combining an inexact sequential convex programming scheme in the context of Part I and the algorithms proposed in Part II, we obtained a two-level decomposition algorithm for solving separable nonconvex optimization problems. This algorithm has also been verified via numerical tests.

11.2 Future research directions

The *sequential convex programming* (SCP) approach remains a promising direction to pursue. In this thesis we have only proposed a generic adjoint based predictor-corrector sequential convex programming algorithmic framework for parametric optimization. As a consequence, we obtained an SCP algorithm for solving nonconvex optimization problems and its local convergence result. However, globalization strategies, global convergence as well as implementation aspects for this algorithm have not completely been studied yet. This offers more research to investigate further the theory and implementation of the SCP algorithm. Besides, two important applications of SCP, namely in robust optimization and experimental design have not been covered yet in the thesis. Developing new variants of SCP which allows one to exploit specific structures of these applications is interesting for future research.

Regarding the *generalized inner approximation* methods developed in Chapter 4, we have showed that these algorithms can be applied to solve several nonconvex semidefinite programs arising in static feedback controller design. However, these algorithms can be applied to other applications, e.g. topology optimization, where we can find a suitable inner convex approximation for such problems by taking into account their specific structure. Extensions of this approach to other problem classes will be an interesting research direction to discover.

In the second part of the thesis, we have focused on the *dual decomposition* methods for separable convex optimization. Nevertheless, many optimization applications in practice are nonconvex and hence dual decomposition approaches are no longer directly applicable to these problems. Although we have proposed a *two-level decomposition* method to solve separable nonconvex optimization problems, but this algorithm still has several disadvantages. Despite of mathematical challenges, it is necessary to develop new decomposition algorithms for solving nonconvex optimization problems. In particular, one can improve the two-level algorithm to obtain an one-loop algorithm. Besides, the research on global iteration-complexity, distributed implementations of the

second order decomposition methods as well as online decomposition algorithms remain open questions.

Alternatively to the theory development, we have only provided some representative applications and academic numerical examples for testing the algorithms. Developing new problem formulations for applications as well as looking for practical problems which can be solved by the methodologies developed in this thesis is also worthwhile for a future research direction.

Appendix A

The proof of technical statements

A.1 The proof of technical statements in Chapt. 7

In this appendix, we provide a full proof of Lemmas and Theorems presented in Chapter 7.

The proof of Theorem 7.7.1. We divide the proof of this theorem into two cases. First, we show that the point $(\bar{x}^+, \bar{y}^+)$ generated by the scheme $\widetilde{\mathcal{S}}_{2\text{ps}}$ maintains the δ_+-excessive gap condition (7.2.7). Then we prove for the scheme $\widetilde{\mathcal{S}}_{2\text{ds}}$.

Case 1 (*For scheme* $\widetilde{\mathcal{S}}_{2ps}$). Let us denote by $y^{2+} := y^*(\hat{x}; \beta_2^+)$, $x^1 := x^*(\bar{y}; \beta_1)$, $\tilde{x}^1 := \tilde{x}^*(\bar{y}; \beta_1)$, $\bar{x}^{*+} := \mathcal{P}(\hat{x}; \beta_2^+)$ and $\left\|x - x^1\right\|_\sigma^2 := \sum_{i=1}^M \sigma_{X_i} \left\|x_i - x_i^1\right\|^2$.

From the definition of $g(\cdot; \beta_1)$, (7.7.2)[line 2] and $\beta_1^+ = (1-\tau)\beta_1$ we have:

$$
\begin{aligned}
g(\bar{y}^+; \beta_1^+) &= \min_{x \in X} \left\{ \phi(x) + (\bar{y}^+)^T (Ax - b) + \beta_1^+ p_X(x) \right\} \\
&\overset{\text{line 2(7.7.2)}}{=} \min_{x \in X} \Big\{ (1-\tau) \left[\phi(x) + \bar{y}^T (Ax - b) + \beta_1 p_X(x) \right]_{[1]} \qquad \text{(A.1.1)} \\
&\quad + \tau \left[\phi(x) + (y^{2+})^T (Ax - b) \right]_{[2]} \Big\}.
\end{aligned}
$$

First, we estimate the term $[\cdot]_{[1]}$ in (A.1.1). Since the function $\phi(\cdot) + \bar{y}^T(A \cdot -b) + \beta_1 p_X(\cdot)$ is strongly convex with a convexity parameter $\beta_1\sigma_X > 0$, by using the optimality condition, one can show:

$$\begin{aligned}
[\cdot]_{[1]} &\geq \min_{x\in X} \left\{\phi(x) + \bar{y}^T(Ax-b) + \beta_1 p_X(x)\right\} + \frac{\beta_1}{2}\left\|x - x^1\right\|_\sigma^2 \\
&\overset{(7.1.7)}{=} g(\bar{y};\beta_1) + \frac{\beta_1}{2}\left\|x - x^1\right\|_\sigma^2 \qquad \text{(A.1.2)} \\
&\overset{(7.2.7)}{\geq} f(\bar{x};\beta_2) + \frac{\beta_1}{2}\left\|x - x^1\right\|_\sigma^2 - \delta.
\end{aligned}$$

Note that since $\psi(\bar{x};\beta_2) = \frac{1}{2\beta_2}\left\|A\bar{x}-b\right\|^2 = \frac{(1-\tau)}{2\beta_2^+}\left\|A\bar{x}-b\right\|^2 = (1-\tau)\psi(\bar{x};\beta_2^+)$, by substituting this relation into (A.1.2) we obtain:

$$\begin{aligned}
[\cdot]_{[1]} &\geq \phi(\bar{x}) + \psi(\bar{x};\beta_2) + \frac{\beta_1}{2}\left\|x - x^1\right\|_\sigma^2 - \delta \\
&= \phi(\bar{x}) + \psi(\bar{x};\beta_2^+) - \tau\psi(\bar{x};\beta_2^+) + \frac{\beta_1}{2}\left\|x - x^1\right\|_\sigma^2 - \delta \\
&\overset{\text{def. }\psi}{\geq} \phi(\bar{x}) + \psi(\hat{x};\beta_2^+) + \nabla_x\psi(\hat{x};\beta_2^+)^T(\bar{x}-\hat{x}) + \frac{\beta_1}{2}\left\|x-x^1\right\|_\sigma^2 - \delta \\
&\quad + \frac{1}{2\beta_2^+}\left\|A(\bar{x}-\hat{x})\right\|^2 - \tau\psi(\bar{x};\beta_2^+). \qquad \text{(A.1.3)}
\end{aligned}$$

Here, the last inequality follows from the fact that $\psi(\bar{x};\beta_2^+) = \frac{1}{2\beta_2^+}\left\|A\bar{x}-b\right\|^2$.

Next, we consider the term $[\cdot]_{[2]}$ of (A.1.1). We have:

$$\begin{aligned}
[\cdot]_{[2]} &= \phi(x) + (y^{2+})^T A(x-\hat{x}) + (A\hat{x}-b)^T y^{2+} \\
&\overset{\text{Lemma } 7.1.3}{=} \phi(x) + \nabla_x\psi(\hat{x};\beta_2^+)^T(x-\hat{x}) + \frac{1}{\beta_2^+}\left\|A\hat{x}-b\right\|^2 \qquad \text{(A.1.4)} \\
&= \phi(x) + \psi(\hat{x};\beta_2^+) + \nabla_x\psi(\hat{x};\beta_2^+)^T(x-\hat{x}) + \psi(\hat{x};\beta_2^+).
\end{aligned}$$

From the definitions of $\|\cdot\|_\sigma$, D_σ and $\varepsilon_{[\sigma]}$ we have $\|x-x^c\|_\sigma \leq D_\sigma$, $\left\|\tilde{x}^1 - x^c\right\|_\sigma \leq D_\sigma$ and $\left\|x^1 - \tilde{x}^1\right\|_\sigma \leq \varepsilon_{[\sigma]}$. Moreover, $\left\|x-x^1\right\|_\sigma \geq \left\|x-\tilde{x}^1\right\|_\sigma - \left\|x^1-\tilde{x}^1\right\|_\sigma$.

By using these estimates, we can derive:

$$\begin{aligned}\left\|x-x^1\right\|_\sigma^2 &\geq \left[\left\|x-\tilde{x}^1\right\|_\sigma - \left\|x^1-\tilde{x}^1\right\|_\sigma\right]^2\\ &= \left\|x-\tilde{x}^1\right\|_\sigma^2 - 2\left\|x-\tilde{x}^1\right\|_\sigma \left\|x^1-\tilde{x}^1\right\|_\sigma + \left\|x^1-\tilde{x}^1\right\|_\sigma^2\\ &\geq \left\|x-\tilde{x}^1\right\|_\sigma^2 - 2\left\|x^1-\tilde{x}^1\right\|_\sigma \left[\|x-x^c\|_\sigma + \left\|\tilde{x}^1-x^c\right\|_\sigma\right]\\ &\geq \left\|x-\tilde{x}^1\right\|_\sigma^2 - 4D_\sigma\varepsilon_{[\sigma]}. \end{aligned} \tag{A.1.5}$$

Furthermore, the condition (7.7.8) can be expressed as:

$$\frac{(1-\tau)}{\tau^2}\beta_1\sigma_i \geq \frac{\bar{L}^2}{(1-\tau)\beta_2} = L_i^\psi(\beta_2^+),\ i=1,\cdots,M. \tag{A.1.6}$$

By substituting (A.1.3) and (A.1.4) into (A.1.1) we get:

$$\begin{aligned} g(\bar{y}^+;\beta_1^+) &= \min_{x\in X}\left\{(1-\tau)[\cdot]_{[1]} + \tau[\cdot]_{[2]}\right\}\\ &\overset{(A.1.3)+(A.1.4)}{\geq} \min_{x\in X}\Big\{(1-\tau)\phi(\bar{x})+\tau\phi(x)+\psi(\hat{x};\beta_2^+)+\nabla\psi(\hat{x};\beta_2^+)^T\\ &\quad [(1-\tau)(\bar{x}-\hat{x})+\tau(x-\hat{x})] + \frac{(1-\tau)\beta_1}{2}\left\|x-x^1\right\|_\sigma^2\Big\} - (1-\tau)\delta\\ &\quad + \left[\tau\psi(\hat{x};\beta_2^+) - (1-\tau)\tau\psi(\bar{x};\beta_2^+) + \frac{(1-\tau)}{2\beta_2^+}\|A(\bar{x}-\hat{x})\|^2\right]_{[3]}. \end{aligned}$$

Since $\tau(x-\tilde{x}^1) = (1-\tau)\bar{x}+\tau x-\hat{x}$, by using (A.1.5) and (A.1.6) we can further estimate the last inequality as:

$$\begin{aligned} g(\bar{y}^+;\beta_1^+) &\overset{\phi-\text{convex}}{\geq} \min_{x\in X}\Big\{\phi((1-\tau)\bar{x}+\tau x)+\psi(\hat{x};\beta_2^+)+\nabla\psi(\hat{x};\beta_2^+)((1-\tau)\bar{x}+\tau x-\hat{x})\\ &\quad + \frac{(1-\tau)\beta_1}{2}\left\|x-\tilde{x}^1\right\|_\sigma^2\Big\} - (1-\tau)\delta - 2(1-\tau)\beta_1 D_\sigma\varepsilon_{[\sigma]} + [\cdot]_{[3]}\\ &= \min_{u:=(1-\tau)\bar{x}+\tau x\in X}\Big\{\phi(u)+\psi(\hat{x};\beta_2^+)+\nabla\psi(\hat{x};\beta_2^+)(u-\hat{x})+\frac{(1-\tau)\beta_1}{2\tau^2}\|u-\hat{x}\|_\sigma^2\Big\}\\ &\quad - 2(1-\tau)\beta_1 D_\sigma\varepsilon_{[\sigma]} - (1-\tau)\delta + [\cdot]_{[3]}\\ &\overset{(A.1.6)}{\geq} \min_{u\in X}\Big\{\phi(u)+\psi(\hat{x};\beta_2^+)+\nabla\psi(\hat{x};\beta_2^+)(u-\hat{x})+\frac{L^\psi(\beta_2^+)}{2}\|u-\hat{x}\|_\sigma^2\Big\}\\ &\quad - 2(1-\tau)\beta_1 D_\sigma\varepsilon_{[\sigma]} - (1-\tau)\delta + [\cdot]_{[3]}. \end{aligned} \tag{A.1.7}$$

Now, by using the definition of φ in (7.2.13) the above inequality implies:

$$g(\bar{y}^+;\beta_1^+) \overset{(7.2.13)}{\geq} \varphi(\bar{x}^{*+};\hat{x},\beta_2^+) - 2(1-\tau)\beta_1 D_\sigma \varepsilon_{[\sigma]} - (1-\tau)\delta + [\cdot]_{[3]}$$

$$\overset{(7.2.14)}{\geq} \varphi(\bar{x}^+;\hat{x},\beta_2^+) - 2(1-\tau)\beta_1 D_\sigma \varepsilon_{[\sigma]} - (1-\tau)\delta - 0.5\varepsilon_A^2 + [\cdot]_{[3]}$$

$$\overset{(7.1.15)}{\geq} f(\bar{x}^+;\beta_2^+) - 2(1-\tau)\beta_1 D_\sigma \varepsilon_{[\sigma]} - (1-\tau)\delta - 0.5\varepsilon_A^2 + [\cdot]_{[3]}, \quad \text{(A.1.8)}$$

where $\varepsilon_A := [\sum_{i=1}^M L_i^\psi(\beta_2^+)\varepsilon_i^2]^{1/2}$.

To complete the proof, we estimate $[\cdot]_{[3]}$ as follows:

$$[\cdot]_{[3]} = \tau\psi(\hat{x};\beta_2^+) - \tau(1-\tau)\psi(\bar{x};\beta_2^+) + \frac{(1-\tau)}{2\beta_2^+}\|A(\bar{x}-\hat{x})\|^2$$

$$= \frac{1}{2\beta_2^+}\left[\tau\|A\hat{x}-b\|^2 - \tau(1-\tau)\|A\bar{x}-b\|^2 + (1-\tau)\|A(\bar{x}-\hat{x})\|^2\right]$$

$$= \frac{1}{2\beta_2^+}\|(A\hat{x}-b)-(1-\tau)(A\bar{x}-b)\|^2 \geq 0. \quad \text{(A.1.9)}$$

By substituting (A.1.9) into (A.1.8) and then using the definition of δ_+ in (7.1.12) we obtain:

$$g(\bar{y}^+;\beta_1^+) \geq f(\bar{x}^+;\beta_2^+) - \delta_+,$$

where

$$\delta_+ := (1-\tau)\delta + 2\beta_1(1-\tau)D_\sigma\varepsilon_{[\sigma]} + 0.5\sum_{i=1}^M L_i^\psi(\beta_2^+)\varepsilon_i^2$$

$$= (1-\tau)\delta + \eta_1(\tau,\beta_1,\beta_2,\varepsilon).$$

This is indeed the δ_+-excessive gap condition (7.2.7).

Case 2 (*For scheme* $\widetilde{\mathcal{S}}_{2ds}$). Let us denote by $\bar{y}^2 := y^*(\bar{x};\beta_2)$, $\hat{x}^1 := x^*(\hat{y};\beta_1)$ and $\tilde{\hat{x}}^1 = \tilde{x}^*(\hat{y};\beta_1)$. From the definition of f, the second line of (7.7.3), we have:

$$f(\bar{x}^+;\beta_2^+) := \phi(\bar{x}^+) + \psi(\bar{x}^+;\beta_2^+)$$

$$\overset{\text{line 2}(7.7.3)}{=} \phi((1-\tau)\bar{x}+\tau\tilde{\hat{x}}^1) + \max_{y\in\mathbf{R}^m}\left\{\left[A((1-\tau)\bar{x}+\tau\tilde{\hat{x}}^1)-b\right]^T y - \beta_2^+ p_Y(y)\right\}$$

$$\overset{\phi-\text{convex}+(7.4.2)}{\leq} \max_{y\in\mathbf{R}^m}\Big\{(1-\tau)\left[\phi(\bar{x})+(A\bar{x}-b)^T y - \beta_2 p_Y(y)\right]_{[4]}$$

$$+ \tau\left[\phi(\tilde{\hat{x}}^1)+(A\tilde{\hat{x}}^1-b)^T y\right]_{[5]}\Big\}. \quad \text{(A.1.10)}$$

Now, we estimate two terms $[\cdot]_{[4]}$ and $[\cdot]_{[5]}$ in the last line of (A.1.10). First we note that $p_Y(y) = \frac{1}{2}\|y\|^2$ and $a^T y - \frac{\beta}{2}\|y\|^2 = \frac{1}{2\beta}\|a\|^2 - \frac{\beta}{2}\left\|y - \frac{1}{\beta}a\right\|^2$ for any vectors a and y and $\beta > 0$. Moreover, since $\bar{y}^2$ is the solution of the strongly concave maximization (7.1.12) with a concavity parameter β_2, we can estimate:

$$
\begin{aligned}
[\cdot]_{[4]} &= \phi(\bar{x}) + \frac{1}{2\beta_2}\|A\bar{x} - b\|^2 - \frac{\beta_2}{2}\left\|y - \bar{y}^2\right\|^2 \\
&= \phi(\bar{x}) + \psi(\bar{x};\beta_2) - \frac{\beta_2}{2}\left\|y - \bar{y}^2\right\|^2 \overset{(7.1.13)}{=} f(\bar{x};\beta_2) - \frac{\beta_2}{2}\left\|y - \bar{y}^2\right\|^2 \\
&\overset{(7.2.7)}{\leq} g(\bar{y};\beta_1) - \frac{\beta_2}{2}\left\|y - \bar{y}^2\right\|^2 + \delta \qquad \text{(A.1.11)} \\
&\overset{g(\cdot;\beta_1)-\text{concave}}{\leq} g(\hat{y};\beta_1) + \nabla_y g(\hat{y};\beta_1)^T(\bar{y} - \hat{y}) - \frac{\beta_2}{2}\left\|y - \bar{y}^2\right\|^2 + \delta \\
&\overset{(7.2.4)}{\leq} g(\hat{y};\beta_1) + \widetilde{\nabla}_y g(\hat{y};\beta_1)^T(\bar{y} - \hat{y}) - \frac{\beta_2}{2}\left\|y - \bar{y}^2\right\|_2^2 + (\bar{y} - \hat{y})^T A(\hat{x}^1 - \tilde{\hat{x}}^1) + \delta.
\end{aligned}
$$

Alternatively, by using (7.7.1), the second term $[\cdot]_{[5]}$ can be estimated as:

$$
\begin{aligned}
[\cdot]_{[5]} &= \phi(\tilde{\hat{x}}^1) + (A\tilde{\hat{x}}^1 - b)^T\hat{y} + \beta_1 p_X(\tilde{\hat{x}}^1) \\
&\quad + (A\tilde{\hat{x}}^1 - b)^T(y - \hat{y}) - \beta_1 p_X(\tilde{\hat{x}}^1) \\
&\overset{(7.2.2)}{\leq} \phi(\hat{x}^1) + (A\hat{x}^1 - b)^T\hat{y} + \beta_1 p_X(\hat{x}^1) \\
&\quad + (A\tilde{\hat{x}}^1 - b)^T(y - \hat{y}) - \beta_1 p_X(\tilde{\hat{x}}^1) + \frac{\beta_1}{2}\varepsilon_{[\sigma]}^2 \\
&\overset{(7.1.7)+(7.2.4)}{=} g(\hat{y};\beta_1) + \widetilde{\nabla}_y g(\hat{y};\beta_1)^T(y - \hat{y}) \\
&\quad - \beta_1 p_X(\tilde{\hat{x}}^1) + \frac{\beta_1}{2}\varepsilon_{[\sigma]}^2. \qquad \text{(A.1.12)}
\end{aligned}
$$

Next, we consider the point $u := \bar{y} + \tau(y - \bar{y})$ with $\tau \in (0,1)$. It is easy to see that if $y \in \mathbf{R}^m$ then $u \in \mathbf{R}^m$. Moreover, we have

$$
\begin{cases}
(1-\tau)(\bar{y} - \hat{y}) + \tau(y - \hat{y}) = \bar{y} + \tau(y - \bar{y}) - \hat{y} = u - \hat{y}, \\
u - \hat{y} = u - (1-\tau)\bar{y} - \tau\bar{y}^2 = \tau(y - \bar{y}^2).
\end{cases} \qquad \text{(A.1.13)}
$$

By substituting (A.1.11) and (A.1.12) into (A.1.10) and then using (A.1.13), we deduce:

$$f(\bar{x}^+;\beta_2^+) \le \max_{y\in\mathrm{R}^m}\left\{(1-\tau)[\cdot]_{[4]}+\tau[\cdot]_{[5]}\right\} \tag{A.1.14}$$

$$\begin{aligned}
&\overset{(A.1.11)+(A.1.12)}{\le} \max_{y\in\mathbf{R}^m}\Big\{(1-\tau)g(\hat{y};\beta_1)+\tau g(\hat{y};\beta_1)\\
&\quad+\widetilde{\nabla}_y g(\hat{y};\beta_1)^T\left[(1-\tau)(\bar{y}-\hat{y})+\tau(y-\hat{y})\right]-\frac{(1-\tau)\beta_2}{2}\left\|y-\bar{y}^2\right\|^2\Big\}\\
&\quad-\tau\beta_1 p_X(\tilde{\hat{x}}^1)+0.5\tau\beta_1\varepsilon_{[\sigma]}^2+(1-\tau)\delta+(1-\tau)(\bar{y}-\hat{y})^T A(\hat{x}^1-\tilde{\hat{x}}^1)\\
&\overset{(A.1.13)}{=}\left[\max_{u\in\mathbf{R}^m}\left\{g(\hat{y};\beta_1)+\widetilde{\nabla}_y g(\hat{y};\beta_1)^T(u-\hat{y})-\frac{(1-\tau)\beta_2}{2\tau^2}\|u-\hat{y}\|^2\right\}\right]_{[6]}\\
&\quad+\left[0.5\tau\beta_1\varepsilon_{[\sigma]}^2+(1-\tau)\delta+(1-\tau)(\bar{y}-\hat{y})^T A(\hat{x}^1-\tilde{\hat{x}}^1)-\tau\beta_1 p_X(\tilde{\hat{x}}^1)\right]_{[7]}.
\end{aligned}$$

From (7.7.8) we have:

$$\frac{(1-\tau)}{\tau^2}\beta_2 \ge \frac{\bar{L}^2}{\beta_1} \overset{\text{Lemma 7.1.1}}{\ge} L^g(\beta_1),\ i=1,\cdots,M. \tag{A.1.15}$$

Let us consider the first term $[\cdot]_{[6]}$ of (A.1.14). By using (A.1.15), we can estimate $[\cdot]_{[6]}$ as:

$$\begin{aligned}
[\cdot]_{[6]} &= \max_{u\in\mathbf{R}^m}\left\{g(\hat{y};\beta_1)+\widetilde{\nabla}_y g(\hat{y};\beta_1)^T(u-\hat{y})-\frac{(1-\tau)\beta_2}{2\tau^2}\|u-\hat{y}\|^2\right\}\\
&\overset{(A.1.15)}{\le} \max_{u\in\mathbf{R}^m}\left\{g(\hat{y};\beta_1)+\widetilde{\nabla}_y g(\hat{y};\beta_1)^T(u-\hat{y})-\frac{L^g(\beta_1)}{2}\|u-\hat{y}\|^2\right\}\\
&\overset{(7.7.3)(\text{line 3})}{=} g(\hat{y};\beta_1)+\widetilde{\nabla}_y g(\hat{y};\beta_1)^T(\bar{y}^+-\hat{y})-\frac{L^g(\beta_1)}{2}\left\|\bar{y}^+-\hat{y}\right\|^2\\
&\overset{(7.2.4)}{=} g(\hat{y};\beta_1)+\nabla_y g(\hat{y};\beta_1)^T(\bar{y}^+-\hat{y})-\frac{L^g(\beta_1)}{2}\left\|\bar{y}^+-\hat{y}\right\|^2+(\bar{y}^+-\hat{y})^T A(\tilde{\hat{x}}^1-\hat{x}^1)\\
&\overset{(7.1.10)}{\le} g(\bar{y}^+;\beta_1)+(\bar{y}^+-\hat{y})^T A(\hat{x}^1-\tilde{\hat{x}}^1)\\
&\overset{(7.1.11)}{\le} g(\bar{y}^+;\beta_1^+)+(\beta_1-\beta_1^+)p_X(x^*(\bar{y}^+;\beta_1^+))+(\bar{y}^+-\hat{y})^T A(\hat{x}^1-\tilde{\hat{x}}^1)\\
&\overset{(7.4.2)+(7.4.4)}{\le} g(\bar{y}^+;\beta_1^+)+\left[\alpha\tau\beta_1 D_X+(\bar{y}^+-\hat{y})^T A(\hat{x}^1-\tilde{\hat{x}}^1)\right]_{[8]}.
\end{aligned} \tag{A.1.16}$$

In order to estimate the term: $[\cdot]_{[7]} + [\cdot]_{[8]}$, we see that:

$$\begin{aligned}(\bar{y}^+ - \hat{y}) - (1-\tau)(\hat{y} - \bar{y}) &\overset{(7.7.3)\text{line 1}}{=} L^g(\beta_1)^{-1}(A\tilde{\hat{x}}^1 - b) + (1-\tau)\tau(\bar{y}^2 - \bar{y}) \\ &= L^g(\beta_1)^{-1}A(\tilde{\hat{x}}^1 - x^c) + L^g(\beta_1)^{-1}(Ax^c - b) - (1-\tau)\tau\bar{y} \\ &+ \beta_2^{-1}(1-\tau)\tau A(\bar{x} - x^c) + \beta_2^{-1}(1-\tau)\tau(Ax^c - b),\end{aligned}$$

which leads to

$$\begin{aligned}A^T\left[(\bar{y}^+ - \hat{y}) - (1-\tau)(\hat{y} - \bar{y})\right] &\leq \bar{L}^{-1}\beta_1 \|A\|^2 \left\|\tilde{\hat{x}}^1 - x^c\right\| + \bar{L}^{-1}\beta_1 \left\|A^T(Ax^c - b)\right\| \\ &+ \beta_2^{-1}(1-\tau)\tau \|A\|^2 \|\bar{x} - x^c\| + \beta_2^{-1}(1-\tau)\tau \left\|A^T(Ax^c - b)\right\| \\ &+ (1-\tau)\tau \|A\| \|\bar{y}\|. \qquad \text{(A.1.17)}\end{aligned}$$

From the definition of D_σ in (7.7.1), we have $\left\|\tilde{\hat{x}}^1 - x^c\right\| \leq D_\sigma$ and $\|\bar{x} - x^c\| \leq D_\sigma$. By substituting these estimates into (A.1.17) and using the definitions of $[\cdot]_{[7]}$ and $[\cdot]_{[8]}$ we have:

$$\begin{aligned}[\cdot]_{[7]} + [\cdot]_{[8]} &\leq (1-\tau)\delta + \frac{\tau\beta_1}{2}\varepsilon_{[\sigma]}^2 + \tau\beta_1(\alpha D_X - p_X(\tilde{\hat{x}}^1)) \\ &+ \left[\bar{L}^{-1}\beta_1 C_d + (1-\tau)\tau\left(\beta_2^{-1}C_d + \|A\| \|\bar{y}\|\right)\right]\varepsilon_{[1]}. \qquad \text{(A.1.18)}\end{aligned}$$

By combining (A.1.14), (A.1.16) and (A.1.18), and note that $\alpha D_X - p_X(\tilde{\hat{x}}^1) \leq 0$, we obtain:

$$\begin{aligned}f(\bar{x}^+; \beta_2^+) &\leq g(\bar{y}^+; \beta_1^+) + \tau\beta_1(\alpha D_X - p_X(\tilde{\hat{x}}^1)) + (1-\tau)\delta + \eta(\tau, \beta_1, \beta_2, \bar{y}, \varepsilon) \\ &\leq g(\bar{y}^+; \beta_1^+) + (1-\tau)\delta + \eta(\tau, \beta_1, \beta_2, \bar{y}, \varepsilon) \\ &= g(\bar{y}^+; \beta_1^+) + \delta_+,\end{aligned}$$

which is indeed the inequality (7.2.7) w.r.t. β_1^+, β_2^+ and δ_+. □

The proof of Theorem 7.3.2 and Theorem 7.4.1. The conclusions of Theorems 7.3.2 and 7.4.1 follow directly from Theorem 7.7.1 as a consequence by replacing the accuracy $\varepsilon = 0$ in this theorem. □

The proof of Lemma 7.7.1. Similar to the proof of Theorem 7.7.1, we divide the proof of this lemma into two cases corresponding to a) and b) in (7.7.4).

Case 1 (*The initial point* $(\bar{x}^0, \bar{y}^0)$ *is generated by scheme* a) *of* (7.7.4)). Let us denote by $\bar{x}^{*0} := \mathcal{P}(x^c; \beta_2)$. Since $\bar{y}^0 := \beta_2^{-1}(Ax^c - b)$, we have:

$$\frac{1}{\beta_2}\|Ax^c - b\|^2 + \frac{1}{\beta_2}(Ax^c - b)^T A(x - x^c) = (\bar{y}^0)^T(Ax - b) - \frac{1}{2\beta_2}\|Ax^c - b\|^2$$
$$\leq (\bar{y}^0)^T(Ax - b). \tag{A.1.19}$$

It follows from the definition of $\mathcal{P}(x^c; \beta_2)$ in (7.2.13), the definition of ψ and (A.1.19) that:

$$\varphi(\bar{x}^{*0}; x^c, \beta_2) = \min_{x \in X} \varphi(x; x^c, \beta_2)$$
$$= \min_{x \in X} \left\{ \phi(x) + \psi(x^c; \beta_2) + \nabla_x \psi(x^c; \beta_2)^T(x - x^c) + \sum_{i=1}^{M} \frac{L_i^{\psi}(\beta_2)}{2}\|x_i - x_i^c\|^2 \right\}$$
$$\overset{(7.1.12)+(A.1.19)}{\leq} \min_{x \in X} \left\{ \phi(x) + (Ax - b)^T\bar{y}^0 + \sum_{i=1}^{M} \frac{L_i^{\psi}(\beta_2)}{2}\|x_i - x_i^c\|^2 \right\}. \tag{A.1.20}$$

From the condition $\beta_1\beta_2 \geq \bar{L}^2$ we have:

$$\beta_1\sigma_{X_i} \geq \frac{M}{\beta_2}\|A_i\|^2 = L_i^{\psi}(\beta_2), \quad i = 1, \cdots, M.$$

Substituting these inequalities into (A.1.20) and using the definition of g in (7.1.8) we obtain:

$$\varphi(\bar{x}^{*0}; x^c, \beta_2) = \min_{x \in X} \varphi(x; x^c, \beta_2)$$
$$\leq \min_{x \in X} \left\{ \phi(x) + (Ax - b)^T\bar{y}^0 + \sum_{i=1}^{M} \frac{\beta_1\sigma_i}{2}\|x_i - x_i^c\|^2 \right\}$$
$$\leq \min_{x \in X} \left\{ \phi(x) + (Ax - b)^T\bar{y}^0 + \sum_{i=1}^{M} p_{X_i}(x_i) \right\}$$
$$= g(\bar{y}^0; \beta_1). \tag{A.1.21}$$

Now, by the condition (7.2.14), it follows from (A.1.21) that:

$$\varphi(\bar{x}^0; x^c, \beta_2) \leq g(\bar{y}^0; \beta_1) + 0.5\sum_{i=1}^{M} L_i^{\psi}(\beta_2)\varepsilon_i^2. \tag{A.1.22}$$

On the other hand, from (7.1.15) we have:

$$f(\bar{x}^0;\beta_2)=\phi(\bar{x}^0)+\psi(\bar{x}^0;\beta_2)\leq\varphi(\bar{x}^0;x^c,\beta_2). \tag{A.1.23}$$

Combining (A.1.23) and (A.1.22), we obtain:

$$f(\bar{x}^0;\beta_2)\leq g(\bar{y}^0;\beta_1)+\delta_0,$$

which is indeed the δ_0-excessive gap condition (7.2.7), where $\delta_0:=0.5\sum_{i=1}^M L_i^{\psi}(\beta_2)\varepsilon_i^2$.

Case 2 (*The initial point* $(\bar{x}^0,\bar{y}^0)$ *is generated by scheme* b) *of* (7.7.4)). For notational simplicity, we denote by $\hat{x}^*:=x^*(0^m;\beta_1)$, $\tilde{x}^*:=\tilde{x}^*(0^m;\beta_1)$, $h(\cdot;y,\beta_1):=\sum_{i=1}^M h_i(\cdot;y,\beta_1)$ and $g_1(y):=g(y;\beta_1)$, where h_i is defined in Definition 7.2.1. By using the inexactness in the inequality (7.2.2), we have $h(\tilde{x}^*;y,\beta_1)\leq h(\hat{x}^*;y,\beta_1)+\frac{1}{2}\beta_1\varepsilon_{[\sigma]}^2$ which is rewritten as:

$$\phi(\tilde{x}^*)+\beta_1 p_X(\tilde{x}^*)\leq\phi(\hat{x}^*)+\beta_1 p_X(\hat{x}^*)+\frac{\beta_1}{2}\varepsilon_{[\sigma]}^2\overset{(7.1.7)}{=}g_1(0^m)+\frac{\beta_1}{2}\varepsilon_{[\sigma]}^2. \tag{A.1.24}$$

Since g_1 is concave, by using (7.1.10) and $\nabla_y g_1(0^m)=A\hat{x}^*-b$ we have:

$$\begin{aligned}
g_1(\bar{y}^0)&\geq g_1(0^m)+\nabla_y g_1(0^m)^T\bar{y}^0-\frac{L^g(\beta_1)}{2}\left\|\bar{y}^0\right\|^2\\
&=g_1(0^m)+(A\hat{x}^*-b)^T\bar{y}^0-\frac{L^g(\beta_1)}{2}\left\|\bar{y}^0\right\|^2\\
&\overset{(A.1.24)}{\geq}\phi(\tilde{x}^*)+\beta_1 p_X(\tilde{x}^*)+(A\hat{x}^*-b)^T\bar{y}^0-\frac{L^g(\beta_1)}{2}\left\|\bar{y}^0\right\|^2-\frac{\beta_1}{2}\varepsilon_{[\sigma]}^2\\
&=\phi(\tilde{x}^*)+(A\tilde{x}^*-b)^T\bar{y}^0-\frac{L^g(\beta_1)}{2}\left\|\bar{y}^0\right\|^2\\
&\quad+(\bar{y}^0)^T A(\hat{x}^*-\tilde{x}^*)+\beta_1 p_X(\tilde{x}^*)-\frac{\beta_1}{2}\varepsilon_{[\sigma]}^2.
\end{aligned} \tag{A.1.25}$$

Since $\|\hat{x}^*-\tilde{x}^*\|\leq\varepsilon_{[1]}$, $p_X(\tilde{x}^*)\geq p_X^*>0$ and $\bar{y}^0$ is the solution of (7.1.12), we estimate the last inequality (A.1.25) as:

$$\begin{aligned}
g_1(\bar{y}^0)&\geq\phi(\tilde{x}^*)+\max_{y\in\mathbb{R}^m}\left\{(A\tilde{x}^*-b)^T y-\frac{L^g(\beta_1)}{2}\|y\|^2\right\}-\left\|A^T\bar{y}^0\right\|\|\hat{x}^*-\tilde{x}^*\|-\frac{\beta_1}{2}\varepsilon_{[\sigma]}^2\\
&\overset{(7.7.6)+(7.7.1)}{\geq}\phi(\tilde{x}^*)+\max_{y\in\mathbb{R}^m}\left\{(A\tilde{x}^*-b)^T y-\frac{\beta_2}{2}\|y\|^2\right\}-\left\|A^T\bar{y}^0\right\|\varepsilon_{[1]}-\frac{\beta_1}{2}\varepsilon_{[\sigma]}^2\\
&\overset{(7.1.13)}{\geq}f(\bar{x}^0;\beta_2)-\left[\left\|A^T\bar{y}^0\right\|\varepsilon_{[1]}+\frac{\beta_1}{2}\varepsilon_{[\sigma]}^2\right].
\end{aligned} \tag{A.1.26}$$

Now, we see that $p_X^* + \frac{\sigma_i}{2}\left\|\bar{x}_i^0 - x_i^c\right\|^2 \leq p_{X_i}(\bar{x}_i^0) \leq \sup_{x_i \in X_i} p_{X_i}(x_i) = D_{X_i}$. Thus, $\left\|\bar{x}_i^0 - x_i^c\right\|^2 \leq \frac{2}{\sigma_i}(D_{X_i} - p_{X_i}^*)$ for all $i = 1, \cdots, M$. By using the definition of D_σ in (7.7.1), the last inequalities imply:

$$\left\|\bar{x}^0 - x^c\right\| \leq D_\sigma. \tag{A.1.27}$$

Finally, we note that $A^T\bar{y}^0 = (L^g(\beta_1))^{-1}A^T(A\bar{x}^0 - b)$ due to (7.7.4). This relation leads to

$$\begin{aligned}
\left\|A^T\bar{y}^0\right\| &= L^g(\beta_1)^{-1}\left\|A^T(A\bar{x}^0 - b)\right\| = L^g(\beta_1)^{-1}\left\|A^T(A(\bar{x}^0 - x^c) + Ax^c - b)\right\| \\
&\leq L^g(\beta_1)^{-1}\left[\left\|A^TA\right\|\left\|\bar{x}^0 - x^c\right\| + \left\|A^T(Ax^c - b)\right\|\right] \\
&\overset{(A.1.27)}{\leq} \bar{L}^{-1}\beta_1\left[\|A\|^2 D_\sigma + \left\|A^T(Ax^c - b)\right\|\right] \\
&\overset{(7.7.1)}{=} \bar{L}^{-1}\beta_1 C_d.
\end{aligned} \tag{A.1.28}$$

By substituting (A.1.27) and (A.1.28) into (A.1.26) and then using the definition of δ_0 we obtain the conclusion of the lemma. □

The proof of Lemma 7.3.1. Lemma 7.3.1 is a special case of Lemma 7.7.1 without the inexactness. We obtain its conclusions directly from the proof of Lemma 7.7.1. □

The proof of Corollary 7.3.1. Indeed, let us prove the condition $g(\bar{y}^+; \beta_1^+) \geq f(\hat{\bar{x}}^+; \beta_2^+)$, where $\hat{\bar{x}}^+ := \mathcal{G}(\hat{x}; \beta_2^+)$. First, by using the convexity of ϕ_i and the Lipschitz continuity of its gradient, we have:

$$\begin{cases}
\phi_i(\hat{x}_i) + \nabla\phi_i(\hat{x}_i)^T(u_i - \hat{x}_i) \leq \phi_i(u_i), \\
\phi_i(u_i) \leq \phi_i(\hat{x}_i) + \nabla\phi_i(\hat{x}_i)^T(u_i - \hat{x}_i) + \frac{L^{\phi_i}}{2}\left\|u_i - \hat{x}_i\right\|^2.
\end{cases} \tag{A.1.29}$$

Next, by summing up the second inequality for $i = 1, \cdots, M$ and adding to (7.1.15) we have:

$$\begin{aligned}
\phi(u) + \psi(u; \beta_2^+) \leq \phi(\hat{x}) + \psi(\hat{x}; \beta_2^+) + \left[\nabla\phi(\hat{x}) + \nabla\psi(\hat{x}; \beta_2^+)\right]^T(u - \hat{x}) \\
+ \sum_{i=1}^M \frac{\hat{L}_i^\psi(\beta_2^+)}{2}\left\|u_i - \hat{x}_i\right\|^2.
\end{aligned} \tag{A.1.30}$$

Finally, by we substitute $\varepsilon = 0$ into the second inequality of (A.1.7), one can obtain:

$$g(\bar{y}^+;\beta_1^+)-[\cdot]_{[3]} \overset{(A.1)}{\geq} \min_{u\in X}\Big\{\phi(u)+\psi(\hat{x};\beta_2^+)+\nabla\psi(\hat{x};\beta_2^+)^T(u-\hat{x})$$

$$+\sum_{i=1}^{M}\frac{(1-\tau)\beta_1\sigma_1}{2\tau^2}\|u_i-\hat{x}_i\|^2\Big\}$$

$$\overset{\phi-\text{convex}+(A.1.30)}{\geq} \min_{u\in X}\Big\{\phi(\hat{x})+\nabla\phi(\hat{x})^T(u-\hat{x})+\psi(\hat{x};\beta_2^+)+\nabla\psi(\hat{x};\beta_2^+)^T(u-\hat{x})$$

$$+\sum_{i=1}^{M}\frac{\hat{L}_i^{\psi}(\beta_2^+)}{2}\|u_i-\hat{x}_i\|^2\Big\}$$

$$\overset{(7.1.1)}{=} \phi(\hat{x})+\psi(\hat{x};\beta_2^+)+\left[\nabla\phi(\hat{x})+\nabla\psi(\hat{x};\beta_2^+)\right]^T(\hat{\hat{x}}^+-\hat{x})+\sum_{i=1}^{M}\frac{\hat{L}_i^{\psi}(\beta_2^+)}{2}\left\|\hat{\hat{x}}_i^+-\hat{x}_i\right\|^2$$

$$\overset{(A.1.30)}{\geq} \phi(\hat{\hat{x}}^+)+\psi(\hat{\hat{x}}^+;\beta_2^+)=f(\hat{\hat{x}}^+;\beta_2^+).$$

In this case, the conclusion of Theorem 7.3.2 is still valid for the substitution $\hat{\hat{x}}^+ := \mathcal{G}(\hat{x};\beta_2^+)$ provided that:

$$\frac{(1-\tau)}{\tau^2}\beta_1\sigma_{X_i} \geq L^{\phi_i}+\frac{M\|A_i\|^2}{(1-\tau)\beta_2},\ i\in\{1,\cdots,M\}.$$

This completes the proof. □

The proof of Lemma 7.4.1. Let us consider $\xi(t;\alpha) := \frac{2}{\sqrt{t^3/(t-2\alpha)+1}+1}$, where $\alpha\in[0,1]$ and $t\geq 2$. After a few simple calculations, we can estimate $t+\alpha\leq\sqrt{t^3/(t-2\alpha)+1}\leq t+1$ for all $t>2\max\{1,\alpha(1-\alpha)^{-1}\}$. These estimates lead to

$$\frac{2}{t+2}\leq\xi(t;\alpha)\leq\frac{2}{t+1+\alpha},\ \forall t>2\max\{1,(1-\alpha)^{-1}\alpha\}.$$

From the update rule (7.4.6) we can show that the sequence $\{\tau_k\}_{k\geq 0}$ satisfies $\tau_{k+1}:=\xi(\frac{2}{\tau_k};\alpha_k)$. If we define $t_k:=\frac{2}{\tau_k}$ then $\frac{2}{t_{k+1}}=\xi(\tau_k;\alpha_k)$. Therefore, one can estimate $t_k+1+\alpha_k\leq t_{k+1}\leq t_k+2$ for $t_k>2\max\{1,(1-\alpha_k)^{-1}\alpha_k\}$. Note that $\alpha_k\geq\alpha^*$ by Assumption **7.4.9**, by induction, we can show that $t_0+(1+\alpha^*)k\leq t_k\leq t_0+2k$ for $k\geq 0$ and $t_0>2\max\{1,(1-\alpha^*)^{-1}\alpha^*\}$.

However, since $t_k = \frac{2}{\tau_k}$, the last inequalities lead to

$$\frac{1}{k+1/\tau_0} = \frac{1}{k+t_0/2} \leq \tau_k \leq \frac{1}{0.5(1+\alpha^*)k+t_0/2} = \frac{1}{0.5(1+\alpha^*)k+1/\tau_0},$$

which is indeed (7.4.7). Here, $0 < \tau_0 = 2/t_0$ and $\tau_0 < [\max\{1, (1-\alpha^*)^{-1}\alpha^*\}]^{-1}$.

In order to prove (7.4.8), we note that $(1-\alpha_k\tau_k)(1-\tau_{k+1}) = \frac{\tau_{k+1}^2}{\tau_k^2}$. By induction, we have:

$$\prod_{i=0}^{k-1}(1-\alpha_i\tau_i)\prod_{i=0}^{k-1}(1-\tau_{i+1}) = \frac{(1-\tau_0)\tau_k^2}{\tau_0^2}.$$

By combining this relation and the update rule (7.4.2), we deduce $\beta_1^k\beta_2^{k+1} = \beta_1^0\beta_2^0\frac{(1-\tau_0)\tau_k^2}{\tau_0^2}$ which is the third statement of (7.4.8).

Next, we prove the bound on β_1^k. Since $\beta_1^{k+1} = \beta_1^0\prod_{i=0}^k(1-\alpha_i\tau_i)$, we have:

$$\beta_1^0\prod_{i=0}^k(1-\tau_i) \leq \beta_1^{k+1} \leq \beta_1^0\prod_{i=0}^k(1-\alpha^*\tau_i).$$

By using the following elementary inequalities $-t-t^2 \leq \ln(1-t) \leq -t$ for all $t \in [0, 1/2]$, we obtain $\beta_1^0 e^{-S_1-S_2} \leq \beta_1^{k+1} \leq \beta_1^0 e^{-\alpha^* S_1}$, where $S_1 := \sum_{i=0}^k \tau_i$ and $S_2 := \sum_{i=0}^k \tau_i^2$. From (7.4.7), on the one hand, we have:

$$\sum_{i=0}^k \frac{1}{i+1/\tau_0} \leq S_1 \leq \sum_{i=0}^k \frac{1}{0.5(1+\alpha^*)i+1/\tau_0},$$

which lead to

$$\ln(k+1/\tau_0)+\ln\tau_0 \leq S_1 \leq \frac{1}{0.5(1+\alpha^*)}\ln(k+1/\tau_0)+\gamma_0,$$

for some constant γ_0. On the other hand, we can show that S_2 converges to some constant $\gamma_2 > 0$. Combining all the above estimates together we get $\frac{\gamma}{(\tau_0 k+1)^{2/(1+\alpha^*)}} \leq \beta_1^{k+1} \leq \frac{\beta_1^0}{(\tau_0 k+1)^{\alpha^*}}$ for some positive constant γ. Finally, we estimate the bound on β_2^k. Indeed, it follows from (7.4.7) that:

$$\beta_2^{k+1} = \beta_2^0\prod_{i=0}^k(1-\tau_k) \leq \beta_2^0\prod_{i=0}^k(1-\frac{1}{k+1/\tau_0}) = \beta_2^0\frac{1/\tau_0-1}{k+1/\tau_0} = \frac{\beta_2^0(1-\tau_0)}{\tau_0 k+1},$$

which is the second formula in (7.4.8). □

The proof of Lemma 7.5.2. Let us define $\xi(t) := \frac{2}{\sqrt{1+4/t^2}+1}$. It is easy to show that ξ is increasing in $(0,1)$. Moreover, $\tau_{k+1} = \xi(\tau_k)$ for all $k \geq 0$. Let us introduce $u := 2/t$. Then, we can show that $\frac{2}{u+2} < \xi(\frac{2}{u}) < \frac{2}{u+1}$. By using this inequalities and the increase of ξ in $(0,1)$, we have:

$$\frac{\tau_0}{1+\tau_0 k} \equiv \frac{2}{u_0+2k} < \tau_k < \frac{2}{u_0+k} \equiv \frac{2\tau_0}{2+\tau_0 k}. \tag{A.1.31}$$

Now, by the update rule (7.5.6), at each iteration k, we only either update β_1^k or β_2^k. Hence, this implies:

$$\begin{aligned} \beta_1^k &= (1-\tau_0)(1-\tau_2)\cdots(1-\tau_{2\lfloor k/2 \rfloor})\beta_1^0, \\ \beta_2^k &= (1-\tau_1)(1-\tau_3)\cdots(1-\tau_{2\lfloor k/2 \rfloor - 1})\beta_2^0, \end{aligned} \tag{A.1.32}$$

where $\lfloor x \rfloor$ is the largest integer number which is less than or equal to the positive real number x. On the other hand, since $\tau_{i+1} < \tau_i$ for $i \geq 0$, for any $l \geq 0$, we have:

$$(1-\tau_0)\prod_{i=0}^{2l}(1-\tau_i) < \left[(1-\tau_0)(1-\tau_2)\cdots(1-\tau_{2l})\right]^2 < \prod_{i=0}^{2l+1}(1-\tau_i), \tag{A.1.33}$$

$$\text{and } \prod_{i=0}^{2l-1}(1-\tau_i) < \left[(1-\tau_1)(1-\tau_3)\cdots(1-\tau_{2l-1})\right]^2 < (1-\tau_0)^{-1}\prod_{i=0}^{2l}(1-\tau_i).$$

Note that $\prod_{i=0}^{k}(1-\tau_i) = \frac{(1-\tau_0)}{\tau_0^2}\tau_k^2$, it follows from (A.1.32) and (A.1.33) for $k \geq 1$ that:

$$\frac{(1-\tau_0)\beta_1^0}{\tau_0}\tau_{k+1} < \beta_1^{k+1} < \frac{\beta_1^0\sqrt{1-\tau_0}}{\tau_0}\tau_{k-1}, \text{ and } \frac{\beta_2^0\sqrt{1-\tau_0}}{\tau_0}\tau_{k+1} < \beta_2^{k+1} < \frac{\beta_2^0}{\tau_0}\tau_{k-1}.$$

By combining these inequalities and (A.1.31), and noting that $\tau_0 \in (0,1)$, we obtain (7.5.7). □

A.2 The proof of technical statements in Chapt. 9

In order to prove Lemma A.2.1 in Chapter 9, we need the following auxiliary results.

Lemma A.2.1. *Suppose that Assumptions **A.6.1.7**, **A.8.1.10** and **A.9.1.11** are satisfied. Then:*

a) $\nabla^2\tilde{g}$ *and* $\nabla^2\tilde{g}_{\bar{\delta}}$ *defined by* (9.1.2) *and* (9.2.3)*, respectively, guarantee:*

$$(1-\delta_+)^2\nabla^2\tilde{g}(y_+;t_+) \preceq \nabla^2\tilde{g}_{\bar{\delta}}(y_+;t_+) \preceq (1-\delta_+)^{-2}\nabla^2\tilde{g}(y_+;t_+), \quad \text{(A.2.1)}$$

where $\delta_+ < 1$ *defined by* (9.2.6).

b) *Moreover, one has:*

$$\|\nabla\tilde{g}_{\bar{\delta}}(y;t) - \nabla\tilde{g}(y;t)\|_y^* \leq \|\bar{x}_{\bar{\delta}} - x^*\|_{x^*}. \quad \text{(A.2.2)}$$

c) *If* $\Delta < 1$ *then* $\bar{\lambda}_1 \leq \frac{\Delta+\bar{\lambda}}{1-\Delta}$.

Proof. Since F is standard self-concordant, for any $x \in \mathrm{dom}(F)$ and z such that $\|z-x\|_x < 1$, it follows from [142, Theorem 4.1.6] that:

$$(1-\|z-x\|_x)^2\nabla^2F(x) \preceq \nabla^2F(z) \preceq (1-\|z-x\|_x)^{-2}\nabla^2F(x). \quad \text{(A.2.3)}$$

Since $\nabla^2F(x)$ is symmetric positive definite, by applying [13, Proposition 8.6.6] to two matrices $(1-\|z-x\|_x)^{-2}\nabla^2F(x)$ and $\nabla^2F(z)$, and then to two matrices $(1-\|z-x\|_x)^2\nabla^2F(x)$ and $\nabla^2F(z)$ we obtain:

$$\begin{aligned}(1-\|z-x\|_x)^2A\nabla^2F(x)^{-1}A^T &\preceq A\nabla^2F(z)^{-1}A^T \\ &\preceq (1-\|z-x\|_x)^{-2}A\nabla^2F(x)^{-1}A^T. \quad \text{(A.2.4)}\end{aligned}$$

Using again [13, Proposition 8.6.6] for (A.2.4) we get:

$$\begin{aligned}(1-\|z-x\|_x)^2A^T[A\nabla^2F(x)^{-1}A^T]^{-1}A &\preceq A^T[A\nabla^2F(z)^{-1}A^T]^{-1}A \\ &\preceq (1-\|z-x\|_x)^{-2}A^T[A\nabla^2F(x)^{-1}A^T]^{-1}A. \quad \text{(A.2.5)}\end{aligned}$$

Now, by using (9.1.2) and (9.1.3), we have $\nabla^2\tilde{g}(y;t) = t^{-2}A\nabla^2F(x^*)^{-1}A^T$. Alternatively, by using (9.2.3) and (9.2.4), we get $\nabla^2\tilde{g}_{\bar{\delta}}(y;t) = t^{-2}A\nabla^2F(\bar{x}_{\bar{\delta}})^{-1}A^T$. Substituting these relations with $x = x_+^*$ and $z = \bar{x}_{\bar{\delta}+}$ into (A.2.4) and noting that $\delta_+ = \delta(\bar{x}_+, x_+^*)$ defined by (9.2.6), we obtain (A.2.1).

Next, we prove b). For any $x \in \mathrm{dom}(F)$, we have $\nabla^2F(x) \succ 0$. We show that the matrix $M(x) := \begin{bmatrix}\nabla^2F(x) & A^T \\ A & A\nabla^2F(x)^{-1}A^T\end{bmatrix}$ is symmetric positive semidefinite.

Indeed, for any $z = (u, v) \in \mathbf{R}^n \times \mathbf{R}^m$, we have:

$$\begin{aligned} z^T M(x) z &= u^T \nabla^2 F(x) u + u^T A^T v + v^T A u + v^T A \nabla^2 F(x)^{-1} A^T v \\ &= \left\| \nabla^2 F(x)^{1/2} u \right\|^2 + 2(\nabla^2 F(x)^{1/2} u)^T (\nabla^2 F(x)^{-1/2} A^T v) \\ &+ \left\| \nabla^2 F(x)^{-1/2} A^T v \right\|^2 \\ &= \left\| \nabla^2 F(x)^{1/2} u + \nabla^2 F(x)^{-1/2} A^T v \right\|^2 \geq 0, \end{aligned}$$

which shows that $M(x) \succeq 0$. Since A has full-row rank, $A\nabla^2 F(x)^{-1} A^T \succ 0$. By applying Schur's complement to $M(x)$, see [13], we obtain:

$$A^T [A \nabla^2 F(x)^{-1} A^T]^{-1} A \preceq \nabla^2 F(x). \tag{A.2.6}$$

To prove (A.2.2) we note that $\nabla g_{\bar{\delta}}(y;t) - \nabla g(y;t) = A(\bar{x}_{\bar{\delta}} - x^*)$. Thus $\nabla \tilde{g}_{\bar{\delta}}(y;t) - \nabla \tilde{g}(y;t) = \frac{1}{t} A(\bar{x}_{\bar{\delta}} - x^*)$. This implies:

$$\begin{aligned} \left[\|\nabla \tilde{g}_{\bar{\delta}}(y;t) - \nabla \tilde{g}(y;t)\|_y^* \right]^2 &= t^{-2} (\bar{x}_{\bar{\delta}} - x^*)^T A^T \nabla^2 \tilde{g}(y;t)^{-1} A (\bar{x}_{\bar{\delta}} - x^*) \\ &\overset{(9.1.2),(9.1.3)}{=} (\bar{x}_{\bar{\delta}} - x^*)^T A^T [A \nabla^2 F(x^*)^{-1} A^T]^{-1} A (\bar{x}_{\bar{\delta}} - x^*) \\ &\overset{(A.2.6)}{\leq} (\bar{x}_{\bar{\delta}} - x^*)^T \nabla^2 F(x^*) (\bar{x}_{\bar{\delta}} - x^*) \\ &= \|\bar{x}_{\bar{\delta}} - x^*\|_{x^*}^2, \end{aligned}$$

which is equivalent to (A.2.2).

Finally, we prove c). By using the definitions of $\nabla \tilde{g}_{\bar{\delta}}(\cdot; t_+)$ and $\nabla^2 \tilde{g}_{\bar{\delta}}(\cdot; t_+)$ in (9.2.3), of $\tilde{g}_{\bar{\delta}}(\cdot; t_+)$ in (9.2.4), for any feasible point $\hat{x}$ of (SepCOP$_{\max}$), it follows from the definition of $\bar{\lambda}_1$ in (9.2.5) and $A\hat{x} = b$ that:

$$\begin{aligned} \bar{\lambda}_1^2 = \left[\|\nabla \tilde{g}_{\bar{\delta}}(y;t_+)\|_y^* \right]^2 &\overset{(9.2.5)}{=} \nabla \tilde{g}_{\bar{\delta}}(y;t_+) \nabla^2 \tilde{g}_{\bar{\delta}}(y;t_+)^{-1} \nabla \tilde{g}_{\bar{\delta}}(y;t_+) \\ &\overset{(9.2.4)}{=} t_+^{-1} \nabla g_{\bar{\delta}}(y;t_+) \nabla^2 g_{\bar{\delta}}(y;t_+)^{-1} \nabla g_{\bar{\delta}}(y;t_+) \\ &\overset{(9.2.3)}{=} (\bar{x}_{\bar{\delta}1} - \hat{x})^T A^T \left[A \nabla^2 F(\bar{x}_{\bar{\delta}1})^{-1} A^T \right]^{-1} A (\bar{x}_{\bar{\delta}1} - \hat{x}). \end{aligned} \tag{A.2.7}$$

Since $\Delta = \|\bar{x}_{\bar{\delta}1} - \bar{x}_{\bar{\delta}}\|_{\bar{x}_{\bar{\delta}}} < 1$ by assumption, we can apply the right-hand side of (A.2.5) with $x = \bar{x}_{\bar{\delta}}$ and $z = \bar{x}_{\bar{\delta}1}$ to obtain:

$$\bar{\lambda}_1^2 \leq (1 - \Delta)^{-2} (\bar{x}_{\bar{\delta}1} - \hat{x})^T A^T \left[A \nabla^2 F(\bar{x}_{\bar{\delta}})^{-1} A^T \right]^{-1} A (\bar{x}_{\bar{\delta}1} - \hat{x}). \tag{A.2.8}$$

Now, for any symmetric positive semidefinite matrix Q in $\mathbb{R}^{n\times n}$ and $u, v \in \mathbb{R}^n$, one can easily show that:

$$(u+v)^T Q(u+v) \leq \left[(u^T Qu)^{1/2} + (v^T Qv)^{1/2}\right]^2. \tag{A.2.9}$$

Since $H_{\bar{\delta}} := A^T \left[A\nabla^2 F(\bar{x}_{\bar{\delta}})^{-1} A^T\right]^{-1} A \succeq 0$, by applying (A.2.9) with $Q = H_{\bar{\delta}}$, $u = \bar{x}_{\bar{\delta}1} - \bar{x}_{\bar{\delta}}$ and $v = \bar{x}_{\bar{\delta}} - \hat{x}$, we have:

$$\bar{\lambda}_1^2 \leq (1-\Delta)^{-2}\left\{\left[(\bar{x}_{\bar{\delta}1}-\bar{x}_{\bar{\delta}})^T H_{\bar{\delta}}(\bar{x}_{\bar{\delta}1}-\bar{x}_{\bar{\delta}})\right]_{[1]}^{1/2} + \left[(\bar{x}_{\bar{\delta}}-\hat{x})^T H_{\bar{\delta}}(\bar{x}_{\bar{\delta}}-\hat{x})\right]_{[2]}^{1/2}\right\}^2. \tag{A.2.10}$$

Note that $H_{\bar{\delta}} \preceq \nabla^2 F(\bar{x}_{\bar{\delta}})$ due to (A.2.6). The first term $[\cdot]_{[1]}$ in (A.2.10) satisfies:

$$[\cdot]_{[1]} \leq (\bar{x}_{\bar{\delta}+} - \bar{x}_{\bar{\delta}})^T \nabla^2 F(\bar{x}_{\bar{\delta}})(\bar{x}_{\bar{\delta}1} - \bar{x}_{\bar{\delta}}) = \Delta^2. \tag{A.2.11}$$

On the other hand, by substituting $\bar{x}_{\bar{\delta}1}$ by $\bar{x}_{\bar{\delta}}$ into (A.2.7), we get:

$$\bar{\lambda}^2 = (\bar{x}_{\bar{\delta}}-\hat{x})^T A^T \left[A\nabla^2 F(\bar{x}_{\bar{\delta}})^{-1} A^T\right]^{-1} A(\bar{x}_{\bar{\delta}}-\hat{x}) = (\bar{x}_{\bar{\delta}}-\hat{x})^T H_{\bar{\delta}}(\bar{x}_{\bar{\delta}}-\hat{x}). \tag{A.2.12}$$

Combining (A.2.10), (A.2.11) and (A.2.12), we obtain $\bar{\lambda}_1 \leq \frac{\Delta+\bar{\lambda}}{1-\Delta}$ which is indeed the statement c). □

The proof of Lemma A.2.1. Since $\delta_1 + 2\Delta + \bar{\lambda} < 1$, it implies that $\delta_1 < 1$, $\Delta < 1/2$ and $\bar{\lambda} < 1$. The proof of Lemma 9.2.2 is divided into several steps as follows.

Step 1. First, let $p := y_+ - y$, we prove the following inequality:

$$\bar{\lambda}_+ \leq (1-\delta_+)^{-1}\left\{\delta_+ + (1-\|p\|_y)^{-1}\left[\delta_1 + \frac{(2\delta_1 - \delta_1^2)}{(1-\delta_1)^2}\|p\|_y + \frac{\|p\|_y^2}{1-\|p\|_y}\right]\right\}. \tag{A.2.13}$$

Indeed, it follows from (A.2.1) that:

$$\begin{aligned}\bar{\lambda}_+ &= \|\nabla \tilde{g}_{\bar{\delta}}(y_+;t_+)\|_{y_+}^* = \left[\nabla \tilde{g}_{\bar{\delta}}(y_+;t_+)\nabla^2 \tilde{g}_{\bar{\delta}}(y_+;t_+)^{-1}\nabla \tilde{g}_{\bar{\delta}}(y_+;t_+)\right]^{1/2} \\ &\overset{(A.2.1)}{\leq} (1-\delta_+)^{-1}\left[\nabla \tilde{g}_{\bar{\delta}}(y_+;t_+)\nabla^2 \tilde{g}(y_+;t_+)^{-1}\nabla \tilde{g}_{\bar{\delta}}(y_+;t_+)\right]^{1/2} \\ &\leq (1-\delta_+)^{-1}\|\nabla \tilde{g}_{\bar{\delta}}(y_+;t_+)\|_{y_+}^*.\end{aligned} \tag{A.2.14}$$

Furthermore, by using (A.2.2) we have:

$$\begin{aligned}\|\nabla \tilde{g}_{\bar{\delta}}(y_+;t_+)\|_{y_+}^* &\leq \|\nabla \tilde{g}(y_+;t_+)\|_{y_+}^* + \|\nabla \tilde{g}_{\bar{\delta}}(y_+;t_+) - \nabla \tilde{g}(y_+;t_+)\|_{y_+}^* \\ &\overset{(A.2.2)}{\leq} \|\nabla \tilde{g}(y_+;t_+)\|_{y_+}^* + \delta_+.\end{aligned} \tag{A.2.15}$$

Since $\tilde{g}(\cdot;t_+)$ is standard self-concordant due to Lemma 9.1.1, one has:

$$\begin{aligned}\|\nabla\tilde{g}(y_+;t_+)\|^*_{y_+} &\leq (1-\|y_+-y\|_y)^{-1}\|\nabla\tilde{g}(y_+;t_+)\|^*_y\\ &= (1-\|p\|_y)^{-1}\|\nabla\tilde{g}(y_+;t_+)\|^*_y. \end{aligned}\tag{A.2.16}$$

Plugging (A.2.16) and (A.2.15) into (A.2.14) we obtain:

$$\bar{\lambda}_+ \leq (1-\delta_+)^{-1}\left[(1-\|p\|_y)^{-1}\|\nabla\tilde{g}(y_+;t_+)\|^*_y+\delta_+\right]. \tag{A.2.17}$$

On the other hand, from (9.2.12), we have:

$$\begin{aligned}\nabla\tilde{g}(y_+;t_+) &\overset{(9.2.12)}{=} \nabla\tilde{g}(y_+;t_+) - \left[\nabla\tilde{g}_{\bar{\delta}}(y,t_+)+\nabla^2\tilde{g}_{\bar{\delta}}(y;t_+)(y_+-y)\right]\\ &= \left[\nabla\tilde{g}(y;t_+)-\nabla\tilde{g}_{\bar{\delta}}(y;t_+)\right]_{[1]}\\ &+ \left\{[\nabla^2\tilde{g}(y;t_+)-\nabla^2\tilde{g}_{\bar{\delta}}(y;t_+)](y_+-y)\right\}_{[2]}\\ &+ \left[\nabla\tilde{g}(y_+;t_+)-\nabla\tilde{g}(y;t_+)-\nabla^2\tilde{g}(y;t_+)(y_+-y)\right]_{[3]}. \end{aligned}\tag{A.2.18}$$

By substituting t by t_+ in (A.2.2), we obtain an estimate for $[\cdot]_{[1]}$ of (A.2.18) as:

$$\|\nabla\tilde{g}(y;t_+)-\nabla\tilde{g}_{\bar{\delta}}(y;t_+)\|^*_y \leq \|\bar{x}_{\bar{\delta}1}-x^*_1\|_{x^*_1}=\delta_1. \tag{A.2.19}$$

Next, we consider the second term $[\cdot]_{[2]}$ of (A.2.18). It follows from (A.2.1) that:

$$\begin{aligned}\left[(1-\delta_1)^2-1\right]\nabla^2\tilde{g}(y;t_+) &\preceq \nabla^2\tilde{g}_{\bar{\delta}}(y;t_+)-\nabla^2\tilde{g}(y;t_+)\\ &\preceq \left[(1-\delta_1)^{-2}-1\right]\nabla^2\tilde{g}(y;t_+). \end{aligned}\tag{A.2.20}$$

If we define $G := \left[\nabla^2\tilde{g}_{\bar{\delta}}(y;t_+)-\nabla^2\tilde{g}(y;t_+)\right]$ and $H := \nabla^2\tilde{g}(y;t_+)^{-1/2}G\nabla^2\tilde{g}(y;t_+)^{-1/2}$ then:

$$\left\|[\nabla^2\tilde{g}(y;t)-\nabla^2\tilde{g}_{\bar{\delta}}(y;t_+)](y_+-y)\right\|^*_y = \|Gp\|^*_y \leq \|H\|\,\|p\|_y. \tag{A.2.21}$$

By virtue of (A.2.20) and the condition $\delta_1<1$, one has:

$$\|H\| \leq \max\left\{1-(1-\delta_1)^2,(1-\delta_1)^{-2}-1\right\} = (1-\delta_1)^{-2}(2\delta_1-\delta_1^2).$$

Hence, (A.2.21) leads to:

$$\left\|[\nabla^2\tilde{g}(y;t)-\nabla^2\tilde{g}_{\bar{\delta}}(y;t_+)](y_+-y)\right\|^*_y \leq (1-\delta_1)^{-2}(2\delta_1-\delta_1^2)\|p\|_y. \tag{A.2.22}$$

Furthermore, since $\tilde{g}(\cdot;t)$ is standard self-concordant, similar to the proof of [142, Theorem 4.1.14], the third term $[\cdot]_{[3]}$ of (A.2.18) is estimated as:

$$\left\|\nabla\tilde{g}(y_+;t_+)-\nabla\tilde{g}(y;t_+)-\nabla^2\tilde{g}(y;t_+)(y_+-y)\right\|_y^* \leq (1-\|p\|_y)^{-1}\|p\|_y^2. \quad \text{(A.2.23)}$$

Now, we apply the triangle inequality $\|a+b+c\|_y^* \leq \|a\|_y^* + \|b\|_y^* + \|c\|_y^*$ to (A.2.18) and then plugging (A.2.19), (A.2.22) and (A.2.23) into the resulting inequality to obtain:

$$\|\nabla\tilde{g}_{\bar{\delta}}(y_+;t_+)\|_y^* \leq \delta_1 + (1-\delta_1)^{-2}(2\delta_1-\delta_1^2)\|p\|_y + (1-\|p\|_y)^{-1}\|p\|_y^2.$$

Finally, by substituting the last inequality into (A.2.17) we get (A.2.13).

Step 2. Next, we estimate (A.2.13) in terms of $\bar{\lambda}_1$ to obtain:

$$\bar{\lambda}_+ \leq (1-\delta_+)^{-1}\left[\left(\frac{\bar{\lambda}_1}{1-\delta_1-\bar{\lambda}_1}\right)^2 + \frac{(2\delta_1-\delta^2)}{(1-\delta_1)^2}\left(\frac{\bar{\lambda}_1}{1-\delta_1-\bar{\lambda}_1}\right) + \frac{(1-\delta_1)\delta_1}{1-\delta_1-\bar{\lambda}_1} + \delta_+\right]. \quad \text{(A.2.24)}$$

Indeed, by using (A.2.4) with $x = \bar{x}_{\bar{\delta}1}$ and $z = x_1^*$ and then (9.1.2) we have:

$$(1-\delta_1)^2\nabla^2\tilde{g}_{\bar{\delta}}(y;t_+) \preceq \nabla^2\tilde{g}(y;t_+) \preceq (1-\delta_1)^{-2}\nabla^2\tilde{g}_{\bar{\delta}}(y,t_+).$$

These inequalities together with the definition of $\|\!|\cdot|\!\|_y$ imply:

$$(1-\delta_1)\|\!|p|\!\|_y \leq \|p\|_y = \left[p^T\nabla^2 g(y;t_+)p\right]^{1/2} \leq (1-\delta_1)^{-1}\|\!|p|\!\|_y.$$

Moreover, since $\|\!|p|\!\|_y = \|\!|\nabla\tilde{g}_{\bar{\delta}}(y;t_+)|\!\|_y^* = \bar{\lambda}_1$ due to (9.2.12), the last inequality is equivalent to:

$$\|p\|_y \leq (1-\delta_1)^{-1}\bar{\lambda}_1. \quad \text{(A.2.25)}$$

Note that the right-hand side of (A.2.13) is nondecreasing w.r.t. $\|p\|_y$ in $[0,1)$. Substituting (A.2.25) into (A.2.13) we finally obtain (A.2.24).

Step 3. We further estimate (A.2.24) in terms of Δ and $\bar{\lambda}$. First, we can easily check that the right-hand side of (A.2.24) is nondecreasing w.r.t. $\bar{\lambda}_1$, δ_1 and δ_+. Now, by using the definitions of Δ and $\bar{\lambda}$, it follows from Lemma A.2.1 c) that:

$$\bar{\lambda}_1 \leq (1-\Delta)^{-1}(\bar{\lambda}+\Delta).$$

Since $\delta_+ < 1$ and $\delta_1 + 2\Delta + \bar{\lambda} < 1$, substituting this inequality into (A.2.24), we obtain

$$\bar{\lambda}_+ \leq (1-\delta_+)^{-1}\left[\delta_+ + \left(\frac{\bar{\lambda}+\Delta}{1-\delta_1-2\Delta-\bar{\lambda}}\right)^2 + \frac{(2\delta_1-\delta_1^2)}{(1-\delta_1)^2}\left(\frac{\bar{\lambda}+\Delta}{1-\delta_1-2\Delta-\bar{\lambda}}\right)\right.$$
$$\left. + \frac{\delta_1(1-\delta_1)(1-\Delta)}{1-\delta_1-2\Delta-\bar{\lambda}}\right]. \quad \text{(A.2.26)}$$

The right-hand side of (A.2.26) is well-defined and nondecreasing w.r.t. all variables.

Step 4. Finally, we facilitate the right-hand side of (A.2.26) to obtain (9.2.15). Since $\bar{\lambda} \geq 0$, we have:

$$\begin{aligned}(1-\delta_1)(1-\Delta) &= [1-\delta_1-2\Delta-\bar{\lambda}] + (\bar{\lambda}+\Delta) + \delta_1\Delta \\ &\leq [1-\delta_1-2\Delta-\bar{\lambda}] + (1+\delta_1)(\bar{\lambda}+\Delta).\end{aligned}$$

The last inequality implies:

$$\frac{\delta_1(1-\delta_1)(1-\Delta)}{1-\delta_1-2\Delta-\bar{\lambda}} \leq \delta_1 + \delta_1(1+\delta_1)\Big(\frac{\Delta+\bar{\lambda}}{1-\delta_1-2\Delta-\bar{\lambda}}\Big). \tag{A.2.27}$$

Alternatively, since $0 \leq \delta_1 < 1$, we have $1+\delta_1 \leq \frac{1}{1-\delta_1}$. Thus:

$$\begin{aligned}(1-\delta_1)^{-2}(2\delta_1-\delta_1^2) + \delta_1(1+\delta_1) &= \delta_1\left[(1-\delta_1)^{-2} + (1-\delta_1)^{-1} + (1+\delta_1)\right] \\ &\leq \delta_1\left[(1-\delta_1)^{-2} + 2(1-\delta_1)^{-1}\right].\end{aligned}$$

Substituting inequality (A.2.27) into (A.2.26) and then using the last inequality and $\xi := \frac{\bar{\lambda}+\Delta}{1-\delta_1-2\Delta-\bar{\lambda}}$, we obtain (9.2.15).

Step 5. The nondecrease of the right-hand side of (9.2.15) is obvious. The inequality (9.2.16) follows directly from (9.2.15) by noting that $\bar{\lambda} \equiv \lambda$ and $\bar{x}_{\bar{\delta}} \equiv x^*$. □

Bibliography

[1] T. Alamo, J. Bravo, M. Redondo, and E. Camacho. "A set-membership state estimation algorithm based on DC programming". In: *Automatica* 44.1 (2008), pp. 216–224.

[2] J. Aldrich and R. Skelton. "Time-energy optimal control of hyper-actuated mechanical systems with geometric path constraints". In: *Decision and Control and 2005 European Control Conference.* 2005, pp. 8246–8253.

[3] d'A. Alexandre, B. Onureena, and E. Laurent. "First-order methods for sparse covariance selection". In: *SIAM J. Matrix Anal. Appl.* 30.1 (2008), pp. 56–66.

[4] E. Allgower and K. Georg. "Continuation and path-following". In: *Acta Numerica* 2 (1992), pp. 1–64.

[5] J. Andersson, B. Houska, and M. Diehl. "Towards a Computer Algebra System with Automatic Differentiation for use with Object-Oriented modelling languages". In: *3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, Oslo, Norway, October 3.* 2010.

[6] G. Andrews. *Foundations of Multithreaded, Parallel, and Distributed Programming.* University of Arizona: Addison-Wesley, 2000.

[7] P. Apkarian and H. Tuan. "Robust Control via Concave Minimization - Local and Global Algorithms". In: *IEEE Trans. Autom. Control* 45.2 (2000), pp. 299–305.

[8] I. Bauer, H. Bock, S. Körkel, and J. Schlöder. "Numerical methods for optimum experimental design in DAE systems". In: *J. Comput. Appl. Math.* 120.1-2 (2000), pp. 1–15.

[9] A. Beck, A. Ben-Tal, and L. Tetruashvili. "A sequential parametric convex approximation method with applications to nonconvex truss topology design problems". In: *J. Global Optim.* 47 (2010), pp. 29–51.

[10] A. Beck and M. Teboulle. “A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems”. In: *SIAM J. Imaging Sci.* 2.1 (2009), pp. 183–202.

[11] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. Ed. by Philadelphia. SIAM, 2001.

[12] A. Ben-Tal and M. Teboulle. “Hidden convexity in some nonconvex quadratically constrained quadratic programming”. In: *Math. Program.* 72 (1996), pp. 51–63.

[13] D. Bernstein. *Matrix mathematics*. Princeton University Press, 2005.

[14] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1996. ISBN: 1886529043.

[15] D. Bertsekas. “Convexification Procedures and Decomposition Methods for Nonconvex Optimization Problems”. In: *J. Optim. Theory and Appl.* 29.2 (1979), pp. 169–197.

[16] D. Bertsekas. “Incremental proximal methods for large scale convex optimization”. In: *Math. Program.* 129.2 (2011), pp. 163–195.

[17] D. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: Numerical methods*. Prentice Hall, 1989.

[18] D. Bertsekas and J. Tsitsiklis. “Some Aspects of Parallel and Distributed Iterative Algorithms - A Survey”. In: *Automatica* 27.1 (1991), pp. 3–21.

[19] L. T. Biegler. *Nonlinear Programming*. MOS-SIAM Series on Optimization. SIAM, 2010.

[20] L. Biegler. “Efficient solution of dynamic optimization and NMPC problems”. In: *Nonlinear Predictive Control*. Ed. by F. Allgöwer and A. Zheng. Vol. 26. Progress in Systems Theory. Basel Boston Berlin: Birkhäuser, 2000, pp. 219–244.

[21] L. Biegler and J. Rawlings. “Optimization approaches to nonlinear model predictive control”. In: *Proc. 4th International Conference on Chemical Process Control - CPC IV*. Ed. by W. Ray and Y. Arkun. AIChE, CACHE, 1991, pp. 543–571.

[22] D. Bienstock and G. Iyengar. “Approximating fractional packings and coverings in $O(1/\epsilon)$ iterations”. In: *SIAM J. Comput.* 35.4 (2006), pp. 825–854.

[23] J. Birge. “Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs”. In: *Operations Research* 33.5 (1985), pp. 989–1007.

[24] V. Blondel and J. Tsitsiklis. “NP-hardness of some linear control design problems”. In: *SIAM J. Control Optim.* 35.21 (1997), pp. 18–27.

[25] H. G. Bock, E. Kostina, H. X. Phu, and R. Rannacher, eds. *Modeling, Simulation and Optimization of Complex Processes*. Springer, 2003.

[26] H. Bock, M. Diehl, D. Leineweber, and J. Schlöder. "A direct multiple shooting method for real-time optimization of nonlinear DAE processes". In: *Nonlinear Predictive Control*. Ed. by F. Allgöwer and A. Zheng. Vol. 26. Progress in Systems Theory. Basel Boston Berlin: Birkhäuser, 2000, pp. 246–267.

[27] H. Bock and K. Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: *Proceedings 9th IFAC World Congress Budapest*. Pergamon Press, 1984, pp. 243–247.

[28] J. F. Bonnans and A. Shapiro. *Perturbation Analysis of Optimization Problems*. Springer, 2000.

[29] J. Bonnans. "Local Analysis of Newton-Type Methods for Variational Inequalities and Nonlinear Programming". In: *Appl. Math. Optim* 29 (1994), pp. 161–186.

[30] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[31] S. Boyd, L. Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. Ed. by S. studies in applied mathematics. Vol. 14. SIAM, 1994.

[32] S. Boyd, N. Parikh, E. Chu, and B. Peleato. "Distributed Optimization and Statistics via Alternating Direction Method of Multipliers". In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

[33] J. Burke, A. Lewis, and M. Overton. "Optimization and Pseudospectra, with Applications to Robust Stability". In: *SIAM J. Matrix Anal. Appl.* 25(1) (2003), pp. 80–104.

[34] J. Burke, A. Lewis, and M. Overton. "Two numerical methods for optimizing matrix stability". In: *Linear Algebra and Its Applications* 351-352 (2002), pp. 117–145.

[35] L. A. Burke J.V. and M. Overton. "Optimizing Matrix Stability". In: *Proceedings of the American Mathematical Society* 129 (2001), pp. 1635–1642.

[36] J. Camino, J. W. Helton, and R. E. Skelton. "Solving Matrix Inequalities whos Unknowns are Matrices". In: *SIAM J. Optim.* 17.1 (2006), pp. 1–36.

[37] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar. "Distributed model predictive control". In: *Control Systems Magazine* 22.1 (2002), pp. 44–52.

[38] Y. Cao, S. Li, L. Petzold, and R. Serban. "Adjoint Sensitivity Analysis for Differential-Algebraic Equations: The Adjoint DAE System and its Numerical Solution". In: *SIAM J. Sci. Comput.* 24.3 (2003), pp. 1076–1089.

[39] G. Chen and M. Teboulle. "A proximal-based decomposition method for convex minimization problems". In: *Math. Program.* 64 (1994), pp. 81–101.

[40] H. Chen and F. Allgöwer. "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability". In: *Automatica* 34.10 (1998), pp. 1205–1218.

[41] H. Choset, W. Burgard, S. Hutchinson, G. Kantor, L. Kavraki, K. Lynch, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms and Implementation*. MIT Press, 2005.

[42] G. Cohen. "Optimization by decomposition and coordination: A unified approach". In: *IEEE Trans. Automat. Control* AC-23.2 (1978), pp. 222–232.

[43] A. Connejo, R. Mínguez, E. Castillo, and R. García-Bertrand. *Decomposition Techniques in Mathematical Programming: Engineering and Science Applications*. Springer-Verlag, 2006.

[44] R. Correa and H. R. C. "A global algorithm for nonlinear semidefinite programming". In: *SIAM J. Optim.* 15.1 (2004), pp. 303–318.

[45] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[46] I. Daubechies, R. Devore, M. Fornasier, and C. Güntürk. "Iteratively Reweighted Least Squares Minimization for Sparse Recovery". In: *Communications on Pure and Applied Mathemathics* 63 (2010), pp. 1–38.

[47] F. Debrouwere, W. V. Loock, G. Pipeleers, Q. Tran-Dinh, M. Diehl, J. D. Schutter, and J. Swevers. "Time-optimal robot path following with Cartesian acceleration constraints: a convex optimization approach." In: *The 13th Mechatronics Forum International Conference*. Vol. 2. Mechatronics Forum International Conference. Linz, Austria, 2012, pp. 469–475.

[48] P. Deuflhard. *Newton Methods for Nonlinear Problems*. New York: Springer, 2004.

[49] O. Devolder, F. Glineur, and Y. Nesterov. "First-order methods of smooth convex optimization with inexact oracle". In: *CORE Discussion papers* (2010).

[50] M. Diehl. "Real-Time Optimization for Large Scale Nonlinear Processes". PhD thesis. Universität Heidelberg, 2001. URL: http://www.ub.uni-heidelberg.de/archiv/1659/.

[51] M. Diehl, H. Bock, and E. Kostina. "An approximation technique for robust nonlinear optimization". In: *Math. Program.* 107 (2006), pp. 213–230.

[52] M. Diehl, H. Bock, and J. Schlöder. "A real-time iteration scheme for nonlinear optimization in optimal feedback control". In: *SIAM J. Control Optim.* 43.5 (2005), pp. 1714–1736.

[53] M. Diehl, H. Bock, J. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. "Real-time optimization and Nonlinear Model Predictive Control of processes governed by differential-algebraic equations". In: *J. Proc. Contr.* 12.4 (2002), pp. 577–585.

[54] M. Diehl, H. J. Ferreau, and N. Haverbeke. "Nonlinear model predictive control". In: ed. by L. Magni, M. Raimondo, and F. Allgöwer. Vol. 384. Lecture Notes in Control and Information Sciences. Springer, 2009. Chap. Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, pp. 391–417.

[55] M. Diehl, R. Findeisen, F. Allgöwer, H. Bock, and J. Schlöder. "Nominal Stability of the Real-Time Iteration Scheme for Nonlinear Model Predictive Control". In: *IEE Proc.-Control Theory Appl.* 152.3 (2005), pp. 296–308.

[56] M. Diehl, F. Glineur, E. Jarlebring, and W. Michiels. *Recent Advances in Optimization and its Applications in Engineering*. Springer, 2010.

[57] M. Diehl, F. Jarre, and C. Vogelbusch. "Loss of superlinear convergence for an SQP-type method with conic constraints". In: *SIAM J. Optim.* 16.4 (2006), pp. 1201–1210.

[58] M. Diehl, A. Walther, H. Bock, and E. Kostina. "An adjoint-based SQP algorithm with quasi-Newton Jacobian updates for inequality constrained optimization". In: *Optim. Methods Softw.* 25.4 (2010), pp. 531–552.

[59] E. Dolan and J. Moré. "Benchmarking optimization software with performance profiles". In: *Math. Program.* 91 (2002), pp. 201–213.

[60] A. L. Dontchev and T. R. Rockafellar. "Characterizations of strong regularity for variational inequalities over polyhedral convex sets". In: *SIAM J. Optim.* 6.4 (1996), pp. 1087–1105.

[61] J. Duchi, A. Agarwal, and M. Wainwright. "Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling". In: *IEEE Trans. Autom. Control* 57.3 (2012), pp. 592–606.

[62] J. Eckstein. "Parallel alternating direction multiplier decomposition of convex programs". In: *J. Optim. Theory Appl.* 80.1 (1994), pp. 39–62.

[63] J. Eckstein and D. Bertsekas. "On the Douglas - Rachford splitting method and the proximal point algorithm for maximal monotone operators". In: *Math. Program.* 55 (1992), pp. 293–318.

[64] F. Facchinei and J.-S. Pang. *Finite-dimensional variational inequalities and complementarity problems*. Ed. by N. York. Vol. 1-2. Springer-Verlag, 2003.

[65] B. Fares, D. Noll, and P. Apkarian. "Robust Control via Sequential Semidefinite Programming". In: *SIAM J. Control Optim.* 40.6 (2002), pp. 1791–1820.

[66] A. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming*. New York: Academic Press, 1983.

[67] A. Fiacco and G. P. McCormick. "Nonlinear Programming: Sequential Unconstrained Minimization Techniques". In: *SIAM publications* (1990).

[68] R. Findeisen, F. Allgöwer, and L. Biegler, eds. *Assessment and Future Directions of Nonlinear Model Predictive Control*. Lecture Notes in Control and Information Sciences. Springer, 2006.

[69] R. Fletcher. *Practical Methods of Optimization*. 2nd. Chichester: Wiley, 1987.

[70] C. Fleury. "Sequential convex programming for structural optimization problems". In: *Optimization of large structural systems; Proceedings of the NATO/DFG Advanced Study Institute*. Vol. 1. Berchtesgaden, Germany, 1991, pp. 531–533.

[71] R. Freund and F. Jarre. *A sensitivity analysis and a convergence result for a sequential semidefinite programming method*. Tech. rep. Murray Hill: Bell Laboratories, 2003.

[72] R. Freund, F. Jarre, and C. Vogelbusch. "Nonlinear semidefinite programming: sensitivity, convergence, and an application in passive reduced-order modeling". In: *Math. Program. Ser. B.* 109 (2007), pp. 581–611.

[73] M. Fukushima, M. Haddou, N. V. Hien, J. Strodiot, T. Sugimoto, and E. Yamakawa. "A parallel descent algorithm for convex programming". In: *Comput. Optim. Appl.* 5.1 (1996), pp. 5–37.

[74] J. Gauvin and R. Janin. "Directional behaviour of optimal solutions in nonlinear mathematical programming". In: *Mathematics of Operations Research* 13.4 (1988), pp. 629–649.

[75] A. Geoffrion. "Generalized Benders Decomposition". In: *Journal of Optimization Theory and Applications* 10 (1972), pp. 237–260.

[76] K. Goh. "Robust control synthesis via bilinear matrix inequalities". PhD Thesis. Los Angeles, CA: University of Southern California, 1995.

[77] D. Goldfarb and S. Ma. "Fast Multiple Splitting Algorithms for Convex Optimization". In: *SIAM J. Optim.* 22.2 (2012), pp. 533–556.

[78] J. Gondzio and A. Grothey. "Parallel Processing and Applied Mathematics PPAM 2005". In: ed. by R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Wasniewski. Lecture Notes in Computer Science, 3911. Berlin: Springer-Verlag, 2006. Chap. Direct Solution of Linear Systems of Size 10^9 Arising in Optimization with Interior Point Methods, pp. 513–525.

[79] C. Gonzaga. “Path-following methods for linear programming”. In: *SIAM Review* 34.2 (1992), pp. 167–224.

[80] A. Grama, G. Karypis, V. Kumar, and A. Gupta. *Introduction to Parallel Computing*. 2nd ed. Addison–Wesley, 2003.

[81] M. Grant. “Disciplined Convex Programming”. PhD thesis. Stanford University, 2004.

[82] A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. Frontiers in Appl. Math. 19. Philadelphia: SIAM, 2000.

[83] A. Griewank and P. Toint. “Partitioned variable metric updates for large structured optimization problems”. In: *Numerische Mathematik* 39 (1982), pp. 119–137.

[84] A. Griewank and A. Walther. “On Constrained Optimization by Adjoint based quasi-Newton Methods”. In: *Optim. Method Softw.* 17 (2002), pp. 869 –889.

[85] J. Guddat, F. G. Vasquez, and H. Jongen. *Parametric Optimization: Singularities, Pathfollowing and Jumps*. Stuttgart: Teubner, 1990.

[86] S. Gumussoy, D. Henrion, M. Millstone, and M. Overton. “Multiobjective Robust Control with HIFOO 2.0.” In: *Proceedings of the 6th IFAC Symposium on Robust Control Design*. Haifa, Israel, 2009.

[87] A. Hamdi. “Decomposition for structured convex programs with smooth multiplier methods”. In: *Applied Mathematics and Computation* 169 (2005), pp. 218–241.

[88] A. Hamdi. “Two-level primal-dual proximal decomposition technique to solve large-scale optimization problems”. In: *Appl. Math. Comput.* 160 (2005), pp. 921–938.

[89] S. Han and G. Lou. “A Parallel Algorithm for a Class of Convex Programs”. In: *SIAM J. Control Optim.* 26 (1988), pp. 345–355.

[90] L. Hans-Jakob and D. Jörg. “Convex risk measures for portfolio optimization and concepts of flexibility”. In: *Math. Program.* 104.2-3 (2005), pp. 541–559.

[91] L. Hariharan and F. Pucci. “Decentralized resource allocation in dynamic networks of agents”. In: *SIAM J. Optim.* 19.2 (2008), pp. 911–940.

[92] A. Hassibi, J. How, and S. Boyd. “A path following method for solving BMI problems in control”. In: *Proceedings of American Control Conference*. Vol. 2. 1999, pp. 1385–1389.

[93] B. He, M. Tao, M. Xu, and X. Yuan. “Alternating directions based contraction method for generally separable linearly constrained convex programming problems”. In: *Optimization* (to appear) (2011).

[94] B. He, M. Xu, and X. Yuan. "Solving large-scale least squares covariance matrix problems by alternating direction methods". In: *SIAM J. Matrix Analy. Appli.* 32 (2011), pp. 136–152.

[95] B. He, H. Yang, and S. Wang. "Alternating directions method with self-adaptive penalty parameters for monotone variational inequalities". In: *J. Optim. Theory Appl.* 106 (2000), pp. 349–368.

[96] A. Helbig, O. Abel, and W. Marquardt. "Model Predictive Control for On-line Optimization of Semi-batch Reactors". In: *Proc. Amer. Contr. Conf.* Philadelphia, 1998, pp. 1695–1699.

[97] D. Henrion, J. Loefberg, M. Kocvara, and M. Stingl. "Solving polynomial static output feedback problems with PENBMI". In: *Proc. the IEEE Conf. Decision Control and Europ. Control Conf.* Sevilla, Spain, 2005.

[98] D. Hertog. "Interior point approach to linear, quadratic and convex programming: Algorithms and complexity". PhD Thesis. Netherland: Delf University, 1992.

[99] C. Hol and C. Scherer. "Positive Polynomials in Control". In: ed. by D. Henrion and A. Garulli. Springer-Verlag, 2005. Chap. A sum-of-squares approach to fixed-order H-infinity synthesis, pp. 45–71.

[100] K. Holmberg. "Experiments with primal-dual decomposition and subgradient methods for the uncapacitated facility location problem". In: *Optimization* 49.5–6 (2001), pp. 495–516.

[101] K. Holmberg and K. Kiwiel. "Mean value cross decomposition for nonlinear convex problem". In: *Optim. Methods and Softw.* 21.3 (2006), pp. 401–417.

[102] B. Houska. "Robust Optimization of Dynamic Systems". (ISBN: 978-94-6018-394-2). PhD thesis. Katholieke Universiteit Leuven, 2011.

[103] IBM Corp. *IBM ILOG CPLEX V12.1, User's Manual for CPLEX*. 2009.

[104] A. Jadbabaie and J. Hauser. "On the stability of receding horizon control with a general terminal cost". In: *IEEE Trans. Autom. Control* 5 (2005), pp. 674–678.

[105] X. J. Jakovetić D. and J. Moura. "Fast distributed gradient methods". In: (2011), pp. 1–32. URL: http://arxiv.org/abs/1112.2972.

[106] F. Jarre. *On an approximation of the Hessian of the Lagrangian*. 2003. URL: http://www.optimization-online.org/DB_HTML/2003/12/800.html.

[107] B. Johansson. "On Distributed Optimization in Networked Systems". PhD Thesis. Stockholm, Sweden: Automatic Control Laboratory, School of Electrical Engineering, Royal Institute of Technology (KTH), 2008.

[108] B. Johansson and M. Johansson. “Distributed non-smooth resource allocation over a network”. In: *Proc. IEEE conference on Decision and Control*. 2009, pp. 1678–1683.

[109] B. Johansson, M. Rabi, and M. Johansson. “A randomized incremental subgradient method for distributed optimization in networked systems”. In: *SIAM J. Optim.* 20.3 (2009), pp. 1157–1170.

[110] C. Kanzow, C. Nagel, H. Kato, and M. Fukushima. “Successive linearization methods for nonlinear semidefinite programs”. In: *Comput. Optim. Appl.* 31 (2005), pp. 251–273.

[111] H. Kato and M. Fukushima. “An SQP-type algorithm for nonlinear second-order cone programs”. In: *Optimization Letters* 1 (2007), pp. 129–144.

[112] D. Klatte and B. Kummer. *Nonsmooth Equations in Optimization: Regularity, Calculus, Methods and Applications*. Dordrecht: Kluwer Academic Publishers, 2002.

[113] M. Kojima, N. Megiddo, S. Mizuno, and et al. *Horizontal and vertical decomposition in interior point methods for linear programs*. Technical Report. Tokyo: Information Sciences, Tokyo Institute of Technology, 1993.

[114] N. Komodakis, N. Paragios, and G. Tziritas. “MRF Energy Minimization & Beyond via Dual Decomposition”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2010).

[115] S. Kontogiorgis, R. Leone, and R. Meyer. “Alternating direction splittings for block angular parallel optimization”. In: *J. Optim. Theory Appl.* 90.1 (1996), pp. 1–29.

[116] M. Kočvara, F. Leibfritz, M. Stingl, and D. Henrion. “A nonlinear SDP algorithm for static output feedback problems in COMPL$_{\text{e}}$ib”. In: *Proc. IFAC World Congress*. Prague, Czech Rep., 2005.

[117] F. Leibfritz. “A LMI-based algorithm for designing suboptimal static $\mathcal{H}_2/\mathcal{H}_\infty$ output feedback controllers”. In: *SIAM J. Control Optim.* 39.6 (2001), pp. 1711–1735.

[118] F. Leibfritz. *COMPleib: Constraint matrix optimization problem library - a collection of test examples for nonlinear semidefinite programs, control system design and related problems*. Tech. Rep. Trier, Germany: Dept. Math., Univ. Trier, 2004.

[119] F. Leibfritz and W. Lipinski. *Description of the benchmark examples in COMPleib 1.0*. Tech. Rep. Trier, Germany: Dept. Math., Univ. Trier, 2003.

[120] F. Leibfritz and J. Maruhn. “A successive SDP-NSDP approach to a robust optimization problem in finance”. In: *Comput. Optim. Appl.* 44.3 (2009), pp. 443–466.

[121] F. Leibfritz and E. Mostafa. “An Interior Point Constrained Trust Region Method for a Special Class of Nonlinear Semidefinite Programming Problems”. In: *SIAM J. Optim.* 12.4 (2002), pp. 1048–1074.

[122] A. Lenoir and P. Mahey. “Accelerating convergence of a separable augmented Lagrangian algorithm”. In: *Tech. Report., LIMOS/RR-07-14* (2007), pp. 1–34.

[123] A. S. Lewis and S. J. Wright. *A proximal method for composite minimization*. 2008. URL: http://arxiv.org/abs/0812.0423.

[124] J. Löfberg. “YALMIP: A Toolbox for Modeling and Optimization in MATLAB”. In: *Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004.

[125] C. M. Low S. H. and J. C. Doyle. “Cross-Layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks”. In: *INFOCOM 2006 - 25th IEEE International Conference on Computer Communications*. 2006, pp. 1–13.

[126] R. Madan. “Distributed algorithms for maximum lifetime routing in wireless sensor networks”. In: *IEEE Trans. Wirel. Commun.* 5.8 (2006), pp. 2185–2193.

[127] B. R. Marks and G. P. Wright. “A General Inner Approximation Algorithm for Nonconvex Mathematical Programs”. In: *Operations Research* 26.4 (1978), pp. 681–683.

[128] J. Mattingley, Y. Wang, and S. Boyd. “Code Generation for Receding Horizon Control”. In: *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design*. Yokohama, Japan, 2010.

[129] D. Q. Mayne. “Nonlinear model predictive control: Challenges and opportunities”. In: *Nonlinear Predictive Control*. Ed. by F. Allgöwer and A. Zheng. Vol. 26. Progress in Systems Theory. Basel Boston Berlin: Birkhäuser, 2000, pp. 23–44.

[130] S. Mehrotra. “On the Implementation of a Primal-Dual Interior Point Method”. In: *SIAM J. Optim.* 2.4 (1992), pp. 575–601.

[131] S. Mehrotra and M. Ozevin. “Decomposition Based Interior Point Methods for Two-Stage Stochastic Convex Quadratic Programs with Recourse”. In: *Operation Research* 57.4 (2009), pp. 964–974.

[132] R. R. Meyer. “Sufficient conditions for the convergence of monotonic mathematical programming algorithms”. In: *Journal of Computer and System Sciences* 12 (1976), pp. 108–121.

[133] I. Necoara, C. Savorgnan, Q. Tran-Dinh, J. A. K. Suykens, and M. Diehl. "Distributed Nonlinear Optimal Control Using Sequential Convex Programming and Smoothing Techniques". In: *Proceedings of the 48th IEEE Conference on Decision and Control*. Shanghai, China, 2009.

[134] I. Necoara and J. Suykens. "Applications of a smoothing technique to decomposition in convex optimization." In: *IEEE Trans. Autom. Control* 53.11 (2008), pp. 2674–2679.

[135] I. Necoara and J. Suykens. "Interior-point Lagrangian decomposition method for separable convex optimization". In: *J. Optim. Theory and Appl.* 143.3 (2009), pp. 567–588.

[136] A. Nedíc and A. Ozdaglar. "Distributed subgradient methods for multi-agent optimization". In: *IEEE Trans. Autom. Control* 54 (2009), pp. 48–61.

[137] Y. Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$". In: *Doklady AN SSSR* 269.translated as Soviet Math. Dokl. (1983), pp. 543–547.

[138] Y. Nesterov. "Barrier subgradient method". In: *Math. Program., Ser. B* 127 (2011), pp. 31–56.

[139] Y. Nesterov. "Dual extrapolation and its applications to solving variational inequalities and related problems". In: *Math. Program.* 109.2-3 (2007), pp. 319–344.

[140] Y. Nesterov. "Excessive gap technique in nonsmooth convex minimization". In: *SIAM J. Optim.* 16.1 (2005), pp. 235–249.

[141] Y. Nesterov. "Gradient methods for minimizing composite objective function". In: *CORE Discussion paper* 76 (2007).

[142] Y. Nesterov. *Introductory lectures on convex optimization: a basic course*. Vol. 87. Applied Optimization. Kluwer Academic Publishers, 2004.

[143] Y. Nesterov. "Modified Gauss-Newton scheme with worst case guarantees for global performance". In: *Optim. Method Softw.* 22.3 (2007), pp. 469–483.

[144] Y. Nesterov. "Primal-dual subgradient methods for convex problems". In: *Math. Program.* 120.1 (2009), pp. 221–259.

[145] Y. Nesterov. "Smooth minimization of non-smooth functions". In: *Math. Program.* 103.1 (2005), pp. 127–152.

[146] Y. Nesterov and A. Nemirovski. *Interior-point Polynomial Algorithms in Convex Programming*. Society for Industrial Mathematics, 1994.

[147] G. Neveen and K. Jochen. "Faster and simpler algorithms for multicommodity flow and other fractional packing problems". In: *SIAM J. Comput.* 37.2 (2007), pp. 630–652.

[148] J. Nocedal and S. Wright. *Numerical Optimization*. 2nd ed. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[149] T. Ohtsuka. "A Continuation/GMRES Method for Fast Computation of Nonlinear Receding Horizon Control". In: *Automatica* 40.4 (2004), pp. 563–574.

[150] R. Orsi, U. Helmke, and J. Moore. "A Newton-like method for solving rank constrained linear matrix inequalities". In: *Automatica* 42.11 (2006), pp. 1875–1882.

[151] E. Ostertag. "An improved path-following method for mixed H_2/H_∞ controller design". In: *IEEE Trans. Autom. Control* 53.8 (2008), pp. 1967–1971.

[152] A. Ostrowski. *Solutions of Equations and Systems of Equations*. New York: Academic Press, 1966.

[153] J. Outrata. "Optimality Conditions for a Class of Mathematical Programs with Equilibrium Constraints". In: *Math. Oper. Res.* 24.3 (1999), pp. 627–644.

[154] D. Palomar and M. Chiang. "A Tutorial on Decomposition Methods for Network Utility Maximization". In: *IEEE J. Selected Areas in Communications* 24.8 (2006), pp. 1439–1451.

[155] F. Pfeifer and R. Johanni. "A concept for manipulator trajectory planning". In: *IEEE Journal of Robotics and Automation* RA-3.2 (1987), pp. 115–123.

[156] D. Pham and H. L. Thi. "A DC optimization algorithms for solving the trust region subproblem". In: *SIAM J. Optim.* 8 (1998), pp. 476–507.

[157] P. Purkayastha and J. Baras. "An optimal distributed routing algorithm using dual decomposition techniques". In: *Commun. Inf. Syst.* 8.3 (2008), pp. 277–302.

[158] J. Rawlings, E. Meadows, and K. Muske. "Nonlinear model predictive control: A tutorial and survey". In: *Proc. Int. Symp. Adv. Control of Chemical Processes, ADCHEM*. Kyoto, Japan, 1994.

[159] J. Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*. Vol. 2. MPS/SIAM Series on Optimization. SIAM, 2001.

[160] S. M. Robinson. "Strongly Regular Generalized Equations". In: *Mathematics of Operations Research, Vol. 5, No. 1 (Feb., 1980), pp. 43-62* 5 (1980), pp. 43–62.

[161] R. T. Rockafellar. *Convex Analysis*. Vol. 28. Princeton Mathematics Series. Princeton University Press, 1970.

[162] R. Rockafellar and R. J.-B. Wets. *Variational Analysis*. Ed. by N. York. Springer-Verlag, 1997.

[163] C. Roos, T. Terlaky, and J.-P. Vial. *Interior Point Methods for Linear Optimization*. Heidelberg/Boston: Springer Science, 2006.

[164] A. Ruszczyński. "On convergence of an augmented Lagrangian decomposition method for sparse convex optimization". In: *Mathematics of Operations Research* 20 (1995), pp. 634–656.

[165] S. Samar, S. Boyd, and D. Gorinevsky. "Distributed Estimation via Dual Decomposition". In: *Proceedings European Control Conference (ECC)*. Kos, Greece, 2007, pp. 1511–1516.

[166] C. Savorgnan and M. Diehl. *Control benchmark of a hydro power plant*. Tech. Report. Optimization in Engineering Center, KU Leuven, 2010. URL: http:/homes.esat.kuleuven.be/~mdiehl.

[167] S. Schlenkrich, A. Griewank, and A. Walther. "On the local convergence of adjoint Broyden methods". In: *Math. Program.* 121.2 (2010), pp. 221–247.

[168] G. Schnitger. *Parallel and Distributed Algorithms*. Institut für Informatik. 2006.

[169] R. Serban and A. Hindmarsh. "CVODES: the Sensitivity-Enabled ODE Solver in SUNDIALS". In: *Proceedings of IDETC/CIE 2005*. 2005.

[170] A. Shapiro. "First and second order analysis of nonlinear semidefinite programs". In: *Math. Program.* 77.1 (1997), pp. 301–320.

[171] M. Shida. "An interior-point smoothing technique for Lagrangian relaxation in large-scale convex programming". In: *Optimization* 57.1 (2008), pp. 183–200.

[172] Z. Shiller and H.-H. Lu. "Computation of path constrained time optimal motions with dynamic singularities". In: *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control* 114 (1992), pp. 34–40.

[173] M. Shugen and W. Mitsuru. "Time-optimal control of kinematically redundant manipulators with limit heat characteristics of actuators". In: *Advanced Robotics* 16.8 (2002), pp. 735–749.

[174] M. Signoretto, Q. Tran-Dinh, L. De-Lathauwer, and J. Suykens. *Learning with Tensors: a framework based on convex optimization and spectral regularization*. Technical Report 11-129. (Submitted for publication). ESAT SCD/SISTA, K.U. Leuven, 2011.

[175] A. Smola, S. V. N. Vishwanathan, and Q. V. Le. "Bundle methods for machine learning". In: *Advances in Neural Information Processing Systems 20*. Ed. by D. Koller and Y. Singer. Cambridge MA, MIT Press, 2007.

[176] J. Spingarn. "Applications of the method of partial inverses to convex programming: Decomposition". In: *Math. Program. Ser. A* 32 (1985), pp. 199–223.

[177] B. Sriperumbudur and G. Lanckriet. “On the convergence of the concave-convex procedure”. In: *Neural Information Processing Systems, NIPS* (2009).

[178] G. Stephanopoulos and W. A.W. “The use of Hestenes’ method of multipliers to resolve dual gaps in engineering system optimization”. In: *J. Optim. Theory and Appl.* 15 (1975), pp. 285–309.

[179] M. Stingl, M. Kocvara, and G. Leugering. “A Sequential Convex Semidefinite Programming Algorithm for Multiple-Load Free Material Optimization”. In: *SIAM J. Optim.* 20.1 (2009), pp. 130–155.

[180] F. Sturm. “Using SeDuMi 1.02: A Matlab toolbox for optimization over symmetric cones”. In: *Optim. Methods Software* 11-12 (1999), pp. 625–653.

[181] D. Sun. “The strong second order sufficient condition and constraint non-degeneracy in nonlinear semidefinite programming and their implications”. In: *Mathematics of Operation Research* 31.4 (2006), pp. 761–776.

[182] K. Svanberg. “The method of moving asymptotes - a new method for structural optimization”. In: *Int. J. Numer. Meths. Engng.* 24 (1987), pp. 359–373.

[183] A. Tanikawa and H. Mukai. “A new technique for nonconvex primal-dual decomposition of a large-scale separable optimization problem”. In: *IEEE Trans. Automatic Control* 30.2 (1985), pp. 133–143.

[184] P. Tatjewski. “New Dual-Type Decomposition Algorithm for Nonconvex Separable Optimization Problems”. In: *Automatica* 25.2 (1989), pp. 233–242.

[185] P. Tatjewski and B. Engelmann. “Two-Level Primal-Dual Decomposition Technique for Large-Scale Nonconvex Optimization Problems with Constraints”. In: *J. Optim. Theory and Appl.* 64.1 (1990), pp. 183–205.

[186] J. Thevenet, D. Noll, and P. Apkarian. “Nonlinear spectral SDP method for BMI-constrained problems: applications to control design”. In: *Informatics in Control, Automation and Robotics* 1 (2006), pp. 61–72.

[187] Q. Tran-Dinh and M. Diehl. “An application of sequential convex programming to time optimal trajectory planning for a car motion”. In: *Proceedings of the 48th IEEE Conference on Decision and Control*. Shanghai, China, 2009, pp. 4366–4371. DOI: 10.1109/CDC.2009.5399823.

[188] Q. Tran-Dinh and M. Diehl. “Local Convergence of Sequential Convex Programming for Nonconvex Optimization”. In: *Recent advances in optimization and its application in engineering*. Ed. by G. F. J. E. Diehl M. and W. Michiels. Springer-Verlag, 2010, pp. 93–103.

[189] Q. Tran-Dinh and M. Diehl. *Proximal methods for minimizing the sum of a convex function and a composite function*. Tech. Report. Belgium: KU Leuven, OPTEC and ESAT/SCD, 2011.

[190] Q. Tran-Dinh and M. Diehl. *Sequential Convex Programming Methods for Solving Nonlinear Optimization Problems with DC constraints*. Tech. Report. Belgium: ESAT/SCD and OPTEC, KU Leuven, 2009. URL: http://arxiv.org/abs/1107.5841.

[191] Q. Tran-Dinh, S. Gumussoy, W. Michiels, and M. Diehl. "Combining convex-concave decompositions and linearization approaches for solving BMIs, with application to static output feedback". In: *IEEE Trans. Autom. Control* 57.6 (2012), pp. 1377–1390.

[192] Q. Tran-Dinh, W. Michiels, and M. Diehl. "An inner convex approximation algorithm for BMI optimization and applications in control". In: *Proc. of The 52th IEEE Conference on Decision and Control*. 2012, (accepted).

[193] Q. Tran-Dinh, I. Necoara, and M. Diehl. "Fast Inexact Decomposition Algorithms for Large-Scale Separable Convex Optimization". In: *Submitted to J. Optim. Theory Appl.* (2012), pp. 1–29.

[194] Q. Tran-Dinh, I. Necoara, and M. Diehl. "Path-Following Gradient-Based Decomposition Algorithms For Separable Convex Optimization". In: (2012). (Submitted for publication).

[195] Q. Tran-Dinh, I. Necoara, C. Savorgnan, and M. Diehl. "An Inexact Perturbed Path-Following Method for Lagrangian Decomposition in Large-Scale Separable Convex Optimization". In: *SIAM J. Optim.* under revision (2012).

[196] Q. Tran-Dinh, C. Savorgnan, and M. Diehl. "Adjoint-based Predictor-Corrector Sequential Convex Programming for Parametric Nonlinear Optimization". In: *SIAM J. Optim.* 22.4 (2012), pp. 1258–1284.

[197] Q. Tran-Dinh, C. Savorgnan, and M. Diehl. "Combining Lagrangian Decomposition and Excessive Gap Smoothing Technique for Solving Large-Scale Separable Convex Optimization Problems". In: *Comput. Optim. Appl.* under revision (2011), pp. 1–29.

[198] Q. Tran-Dinh, C. Savorgnan, and M. Diehl. "Real-Time Sequential Convex Programming for Nonlinear Model Predictive Control and Application to a Hydro-Power Plant". In: *Proc. of the 50th IEEE Conference on Decision and Control (CDC) and the European Control Conference (ECC)*. 2011, pp. 5905–5910.

[199] Q. Tran-Dinh, C. Savorgnan, and M. Diehl. "Real-Time Sequential Convex Programming for Optimal Control Applications". In: *Modeling, Simulation and Optimization of Complex Processes*. Ed. by P. H. R. R. Bock H. and J. Schlöder. Springer-Verlag, 2009, pp. 91–101.

[200] J. Tropp. “Just relax: convex programming methods for identifying sparse signals in noise”. In: *IEEE Trans. Inf. Theory* 52.3 (2006), pp. 1030–1051.

[201] P. Tseng. “Alternating projection-proximal methods for convex programming and variational inequalities”. In: *SIAM J. Optim.* 7.4 (1997), pp. 951–965.

[202] P. Tsiaflakis, M. Diehl, and M. Moonen. “Distributed spectrum management algorithms for multi-user DSL networks”. In: *IEEE Trans. Signal Processing* 56.10 (2008), pp. 4825–4843.

[203] P. Tsiaflakis, I. Necoara, J. Suykens, and M. Moonen. “Improved Dual Decomposition Based Optimization for DSL Dynamic Spectrum Management”. In: *IEEE Transactions on Signal Processing* 58.4 (2010), pp. 2230–2245.

[204] R. Tütünkü, K. Toh, and M. Todd. “Solving semidefinite-quadratic-linear programs using SDPT3”. In: *Math. Program.* 95 (2003), pp. 189–217.

[205] J. Vanbiervliet, B. Vandereycken, W. Michiels, S. Vandewalle, and M. Diehl. “The smoothed spectral abscissa for robust stability optimization”. In: *SIAM J. Optim.* 20.1 (2009), pp. 156–171.

[206] D. Vania. “Finding Approximate Solutions for Large Scale Linear Programs”. PhD Thesis, No 18188. ETH Zurich, 2009.

[207] A. Venkat, I. Hiskens, J. Rawlings, and S. Wright. “Distributed MPC strategies with application to power system automatic generation control”. In: *IEEE Trans. Control Syst. Technol.* 16.6 (2008), pp. 1192–126.

[208] A. Venkat. “Distributed Model Predictive Control: Theory and Applications”. PhD thesis. University of Wisconsin-Madison, 2006.

[209] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl. “Time-Energy Optimal Path Tracking for Robots: a Numerically Efficient Optimization Approach”. In: *Proceedings of the IEEE International Workshop on Advanced Motion Control, Trento, Italy.* 2008.

[210] A. Wächter and L. Biegler. “On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming”. In: *Math. Program.* 106.1 (2006), pp. 25–57.

[211] E. Wei, A. Ozdaglar, and A.Jadbabaie. “A Distributed Newton Method for Network Utility Maximization”. In: (2011). URL: http://web.mit.edu/asuman/www/publications.htm.

[212] N. R. Wright S. J. and M. Figueiredo. “Sparse Reconstruction by Separable Approximation”. In: *IEEE Trans. Signal Process.* 57 (2009), pp. 2479–2493.

[213] S. Wright. *Primal-Dual Interior-Point Methods.* Philadelphia: SIAM Publications, 1997.

[214] L. Xiao, M. Johansson, and S. Boyd. "Simultaneous routing and resource allocation via dual decomposition". In: *IEEE Trans. Commun.* 52.7 (2004), pp. 1136–1144.

[215] M. Yamashita, K. Fujisawa, and M. Kojima. "Implementation and evaluation of SDPA 6.0 (SemiDefinite Programming Algorithm 6.0)". In: *Optim. Method Softw.* 18 (2003), pp. 491–505.

[216] W. Zangwill. *Nonlinear Programming*. Ed. by N. J. E. Cliffs. Prentice Hall, 1969.

[217] V. Zavala and M. Anitescu. "Real-Time Nonlinear Optimization as a Generalized Equation". In: *SIAM J. Control Optim.* 48.8 (2010), pp. 5444–5467.

[218] G. Zhao. "A Lagrangian dual method with self-concordant barriers for multistage stochastic convex programming". In: *Math. Progam.* 102 (2005), pp. 1–24.

[219] G. Zhao. "A Log-barrier with Benders decomposition for solving two-stage stochastic programs". In: *Math. Program.* 90 (2001), pp. 507–536.

[220] G. Zhao. "Interior point methods with decomposition for solving large-scale linear programs". In: *J. Optim. Theory Appl.* 102 (1999), pp. 169–192.

[221] C. Zillober, K. Schittkowski, and K. Moritzen. "Very large scale optimization by sequential convex programming". In: *Optimization Methods and Software* 19 (2004), pp. 103–120.

Publications by the author contains in the thesis

Articles in international journals

1. Q. Tran Dinh, I. Necoara and M. Diehl: Path-Following Gradient-Based Decomposition Algorithms for Separable Convex Optimization. *Submitted to J. Global Optim.*, pp. 1–19, 2012.

2. Q. Tran Dinh, I. Necoara and M. Diehl: Fast Inexact Decomposition Algorithm for Large-Scale Separable Convex Optimization. *Submitted to J. Optim. Theory Appl.*, pp. 1–29, 2012.

3. Q. Tran Dinh, C. Savorgnan and M. Diehl: Combining Lagrangian Decomposition and Excessive Gap Smoothing Technique for Solving Large-Scale Separable Convex Optimization Problems. *Comput. Optim. Appl.*, 2011 (under revision).

4. Q. Tran Dinh, I. Necoara, C. Savorgnan and M. Diehl: An inexact perturbed path-following method for Lagrangian decomposition in large-scale separable convex optimization. *SIAM J. Optim.*, 2012 (accepted for publication).

5. Q. Tran Dinh, C. Savorgnan and M. Diehl: Adjoint-based predictor-corrector sequential convex programming for parametric nonlinear optimization. *SIAM J. Optim.*, Vol. 22, No. 4, pp. 1258–1284, 2012.

6. Q. Tran Dinh, S. Gumussoy, W. Michiels and M. Diehl: Combining convex-concave decompositions and linearization approaches for solving BMIs, with application to static output feedback. *IEEE Trans. Autom. Control*, Vol. 57, No. 6, pp. 1377–1390, 2012.

Book chapters

1. Q. Tran Dinh and M. Diehl: Local convergence of sequential convex programming for nonlinear programming. In: Diehl, M., Glineur, F., Jarlebring, E. and Michiels, W. (Eds.): *Recent advances in optimization and its application in engineering*. Springer-Verlag, pp. 93–102, 2010.

2. Q. Tran Dinh, C. Savorgnan and M. Diehl: Real-time sequential convex programming for optimal control applications. In: Bock, H., Phu, H.X., Rannacher, R. and Schlöder, J.P. (Eds.): *Modeling, Simulation and Optimization of Complex Processes*. Springer-Verlag, pp. 91–101, 2012.

Articles in international conference proceedings

1. Q. Tran Dinh, W. Michiels, S. Gros and M. Diehl: An inner convex approximation algorithm for BMI optimization and applications in control. Accepted for publication in *Proc. of 51th IEEE Conference on Decision and Control*, Hawaii, US, December, 6 pages, 2012.

2. Q. Tran Dinh, C. Savorgnan and M. Diehl: Real-Time Sequential Convex Programming for Nonlinear Model Predictive Control and Application to a Hydro-Power Plant. Proc. of *the 50th IEEE Conference on Decision and Control*, Orlando, Florida, USA, pp. 5905–5910, 2011.

3. Q. Tran Dinh and M. Diehl: An application of sequential convex programming methods to time optimal trajectory planning of a car motion. Proc. of *the 48th IEEE Conference on Decision and Control*, Shanghai, China, 4366–4371, 2009.

4. I. Necoara, C. Savorgnan, Q. Tran Dinh, J.A.K. Suykens and M. Diehl: Distributed Nonlinear Optimal Control Using Sequential Convex Programming and Smoothing Techniques. Proc. of *the 48th IEEE Conference on Decision and Control*, Shanghai, China, pp. 543–548, 2009.

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering
SISTA Research Group and Optimization in Engineering Center (OPTEC)
Kasteelpark Arenberg 10, Bus 2446
B-3001 Heverlee