Matrix Free Methods for Large Scale Optimization

Jiashan Wang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

James V. Burke, Chair

Dmitriy Drusvyatskiy

John Sylvester

Program Authorized to Offer Degree:
Mathematics

University of Washington

Abstract

Matrix Free Methods for Large Scale Optimization

Jiashan Wang

Chair of the Supervisory Committee:
Professor James V. Burke
Department of Mathematics

Sequential quadratic optimization (SQP) methods are widely used to solve large-scale nonlinear optimization problems. We build two matrix-free methods for approximately solving exact penalty subproblems that arise when using SQP methods to solve large-scale optimization problems. The first approach is a novel iterative re-weighting algorithm. The second approach is based on alternating direction augmented Lagrangian technology applied to our setting. We prove that both algorithms are globally convergent under loose assumptions.

SQP methods can be plagued by poor behavior of the global convergence mechanisms. Here we consider global convergence results that use an exact penalty function to compute stepsizes. To confront this issue, we propose a dynamic penalty parameter updating strategy to be employed *within* the subproblem solver in such a way that the resulting search direction predicts progress toward both feasibility and optimality. We prove that does not decrease the penalty parameter unnecessarily in the neighborhood of points satisfying certain common assumptions. We also discuss a coordinate descent subproblem solver in which our updating strategy can be readily incorporated.

In the final application of the thesis, we consider a block coordinate descent (BCD) method applied to graphical model learning with special structures, in particular, hub structure and latent variable selection. We tackle the issue of maintaining the positive definiteness of covariance matrices for general rank 2 updates. An active set strategy is employed to speed up BCD for hub structure problem. For latent variable selection problems,

we propose a method for maintaining a low rank factorization for the covariance matrix while preserving the convexity of the subproblems for SBCD. We show that our proposed method converges to a stationary point of a non-convex formulation. Extensive numerical experiments are discussed for both models.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Without the support from my family and my advisor, I would not be able to finish this thesis.

I would like to express my deepest thanks to my advisor, Prof. James Burke. Thanks for his support in many respects of my life at UW. He spent enormous time on teaching me optimization, discussing research ideas and exposing me to different optimization research topics. He also encouraged me to study statistics and pursue industry internships which helped me to secure an industry job afterwards.

This thesis is a collaborative effort, and contains many important contributions from mentors and colleagues. Chapter 2 to 4 have been published on SIAM Journal on Optimization as a joint paper with Jim, Frank Curtis and Hao Wang, whose perspective and ideas have been invaluable. Chapter 7 is a collaborative undertaking with Jim, Frank Curtis and Hao, while Chapter 8 is a collaborative working paper with Jim, Karthik Mohan and Maryam Fazel.

I would like also to thank the members of my committee: Dmitriy Drusvyatskiy, John Sylvester and Donald Percival.

DEDICATION

To my parents and my wife, Hua Gu

Chapter 1

INTRODUCTION

1.1 Matrix-free Method for Exact Penalty Subproblem

The prototypical convex-composite optimization problem is

$$\min_{x \in X} \; f(x) + \operatorname{dist}\left(F(x) \,|\, C\right), \tag{1.1.1}$$

where the sets $X \subset \mathbb{R}^n$ and $C \subset \mathbb{R}^m$ are non-empty, closed, and convex, the functions $f : \mathbb{R}^n \to \mathbb{R}$ and $F : \mathbb{R}^n \to \mathbb{R}^m$ are smooth, and the distance function is defined as

$$\operatorname{dist}\left(y \,|\, C\right) := \inf_{z \in C} \; \|y - z\|,$$

with $\|\cdot\|$ a given norm on $\mathbb{R}^m$ [7, 25, 56]. The objective in problem (1.1.1) is an exact penalty function for the optimization problem

$$\min_{x \in X} \; f(x) \text{ subject to } F(x) \in C,$$

where the penalty parameter has been absorbed into the distance function. Problem (1.1.1) is also useful in the study of feasibility problems where one takes $f \equiv 0$.

Problems of the form (1.1.1) and algorithms for solving them have received a great deal of study over the last 30 years [2, 27, 59]. The typical approach for solving such problems is to apply a Gauss-Newton strategy to either define a direction-finding subproblem paired with a line search, or a trust-region subproblem to define a step to a new point [7, 56]. The first part of this thesis concerns the design, analysis, and implementation of methods for approximately solving the subproblems in either type of approach in large-scale settings. These subproblems take the form

$$\min_{x \in X} \; g^T x + \frac{1}{2} x^T H x + \operatorname{dist}\left(Ax + b \,|\, C\right), \tag{1.1.2}$$

where $g \in \mathbb{R}^n$, $H \in \mathbb{R}^{n \times n}$ is symmetric, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $X \subset \mathbb{R}^n$ and $C \subset \mathbb{R}^m$ may be modified versions of the corresponding sets in (1.1.1). In particular, the set X

may include the addition of a trust-region constraint. In practice, the matrix H is an approximation to the Hessian of the Lagrangian for the problem (1.1.1) [8, 25, 56], and so may be indefinite depending on how it is formed. However, in this paper, we assume that it is positive semi-definite so that subproblem (1.1.2) is convex.

To solve large-scale instances of (1.1.2), we develop two solution methods based on linear least-squares subproblems for the set C which can be written as a product of simple sets. These solution methods are matrix-free in the sense that the least-squares subproblems can be solved in a matrix-free manner. The first approach is a novel iterative re-weighting strategy [3, 51, 54, 61, 69], while the second is based on ADAL technology [4, 22, 63] adapted to this setting. We prove that both algorithms are globally convergent under loose assumptions, and that each requires at most $O(1/\varepsilon^2)$ iterations to reach ε-optimality of the objective of (1.1.2). We conclude with numerical experiments that compare these two approaches.

Chapter 2 introduces the IRWA and establishes its global convergence property, while chapter 3 adapts ADAL to our setting to solve (1.1.2). Chapter 4 includes extensive numerical comparison of these two approaches. Chapter 5 introduces a possible extension of IRWA.

1.2 Dynamic Penalty Parameter Updating

We consider the use of sequential quadratic optimization (commonly known as SQP) methods for solving large-scale instances of nonlinear optimization problems (NLPs). While they have proved to be effective for solving small- to medium-scale problems, SQP methods have traditionally faltered in large-scale settings due to the expense of (accurately) solving large-scale quadratic subproblems (QPs) during each iteration. However, with the use of matrix-free methods for solving the subproblems, one may consider the acceptance of inexact subproblem solutions. Such a feature offers the possibility of terminating the subproblem solver early, perhaps well before an accurate solution has been computed. This characterizes the types of strategies that we propose in this paper.

Recently, some work has been done to provide global convergence guarantees for SQP methods that allow inexact subproblem solves [18]. However, the practical efficiency of such

an approach remains an open question. A critical aspect of any implementation of such an approach is the choice of subproblem solver. This is the case as the solver must be able to provide good inexact solutions quickly, as well as have the ability to compute highly accurate solutions—say, by exploiting well-chosen starting points—in the neighborhood of a solution of the NLP. In addition, while a global convergence mechanism such as a merit function or filter is necessary to guarantee convergence from remote starting points, an NLP algorithm can suffer when such a mechanism does not immediately guide the algorithm toward promising regions of the search space. To confront this issue when an exact penalty function is used as a merit function, we propose a dynamic penalty parameter updating strategy to be incorporated *within* the subproblem solver so that each computed search direction predicts progress toward both feasibility and optimality. This strategy represents a stark contrast to previously proposed techniques that only update the penalty parameter after a sequence of iterations [50, §18.3] or at the expense of multiple subproblem solves within a single iteration [9, 10, 12, 23, 30], [50, §18.5].

In §7.1, we introduce a basic penalty-SQP algorithm that will form the framework for which we will introduce our penalty parameter updating strategy (see §7.2) and a coordinate descent subproblem solver (see Chapter 6 and §7.3). We discuss the results of extensive numerical experiments in §7.4. Concluding remarks are provided in §7.5.

1.3 Block Coordinate Descent for Structured Graphical Model

Graphical models play a ubiquitous role in machine learning and statistical modeling. Existing literature [67] has focused extensively on addressing the problem of inference using graphical models. There has also been some literature [28, 19, 37] on learning the structure of undirected graphical models. In the literature, sparsity is often used as a prior for learning graphical models. Sparse graphical models that have a small maximum clique size enjoy efficient inference [67]. Also, sparse graphical models are also more interpretable with applications in computational biology [19], social networks [57], etc. However there is often more prior information in these applications than just sparsity. For example, consider the problem of learning two gene-regulatory networks, where one corresponds to a healthy condition and the other corresponds to a cancer condition. The prior information that can

be leveraged in this application is that the networks themselves are sparse (making them interpretable) and that there could possibly be a few mutant genes [19] that change their correlations drastically between the two networks. Leveraging this prior information into a model would make the model more accurate. As another example, in directed social networks, it is of interest to identify influential nodes [36] in the network. In the context of learning undirected graphical models, instead of learning just a sparse network, it would be more appropriate to model social networks through scale-free graphical models (to identify influential nodes) [44, 20]. Recently, convex optimization models have been proposed [48, 64] for learning Gaussian graphical models with hub structure. These models leverage the prior information that the graph contains a few hubs (although the location or the number of these nodes is unknown). This is done through a convex regularizer that promotes hub structure in graphical models. Another example of a model that leverages structured sparsity is when learning the structure of latent Gaussian variables in graphical models [17]. In this model, the precision matrix is expressed as the sum of a low-rank and a sparse matrix. While these models are convex, the problem includes either a log-determinant term (e.g. graphical lasso [28], hub graphical lasso [64], latent variable selection [17], etc) or an explicit positive definite constraint (e.g. hub covariance selection problem [64]). The positive definite constraint necessitates the use of computationally expensive SVD in standard algorithms such as ADMM [5] and first-order methods such as proximal gradient descent [62]. Thus these algorithms don't easily scale to larger problem sizes (i.e. when the problem dimension p is on the order of thousands of variables).

1.3.1 Optimization model

In chapter 8, we consider the following optimization problem:

$$\begin{aligned} \underset{V\in\mathbb{R}^{p\times p},\, Z\in\mathcal{S}^p}{\text{minimize}} \quad & f(V,Z)+g(V,Z) \\ \text{subject to} \quad & V+V^T \succeq 0 \quad (\text{or } \succ 0), \end{aligned} \tag{1.3.1}$$

where $f : \mathbb{R}^{p\times p} \times \mathcal{S}^p \to \mathbb{R}\cup\{+\infty\}$ is convex and continuously differentiable on its domain and $g : \mathbb{R}^{p\times p} \times \mathcal{S}^p \to \mathbb{R}\cup\{+\infty\}$ is convex and block separable in the columns of V and Z. Many problems in the context of learning Gaussian graphical models can be modeled

through (1.3.1) (see e.g. [28, 19, 64, 17]).

1.3.2 Applications

We mention a few applications of the optimization model in (1.3.1) below:

1. Learning a Gaussian precision matrix with hub structure. Hub graphical lasso [64, 48] was proposed as a model for this problem:

$$\begin{array}{ll}\underset{V,Z}{\text{minimize}} & -\log\det(V+V^T+Z)+\langle V+V^T+Z,S\rangle+ \\ & \lambda_1\|Z-\text{Diag}(Z)\|_1+\lambda_2\|V-\text{Diag}(V)\|_1+\lambda_3\|V-\text{Diag}(V)\|_{1,2}\end{array}$$

   Here,

$$\begin{aligned} f(V,Z) &= -\log\det(V+V^T+Z)+\langle V+V^T+Z,S\rangle \\ g(V,Z) &= \lambda_1\|Z-\text{Diag}(Z)\|_1+\lambda_2\|V-\text{Diag}(V)\|_1+\lambda_3\|V-\text{Diag}(V)\|_{1,2}\end{aligned}$$

2. Learning a Gaussian covariance matrix with hub structure. Hub covariance selection [64] was proposed as a model for this problem:

$$\underset{V:V+V^T\succ 0}{\text{minimize}} \quad \tfrac{1}{2}\|V+V^T-S\|_F^2+\lambda\|V-\text{Diag}(V)\|_{1,2} \tag{1.3.2}$$

   Here,

$$\begin{aligned} f(V) &= \tfrac{1}{2}\|V+V^T-S\|_F^2 \\ g(V) &= \lambda\|V-\text{Diag}(V)\|_{1,2}\end{aligned}$$

3. Learning latent variables in graphical models. A low-rank and sparse formulation [17] was proposed as a model for this problem:

$$\underset{L\succeq 0,Y-L\succ 0}{\text{minimize}} -\log\det(Y-L)+\langle Y-L,S\rangle+\lambda_1\|Y\|_1+\lambda_2\,\text{trace}(L). \tag{1.3.3}$$

   Let $L=V+V^T$ and $Y=Z$, then (1.3.3) is equivalent to

$$\underset{V+V^T\succeq 0}{\text{minimize}} -\log\det(Z-V-V^T)+\langle Z-V-V^T,S\rangle+\lambda_1\|Z\|_1+2\lambda_2\,\text{trace}(V).$$

   In this case, we have

$$\begin{aligned} f(V,Z) &= -\log\det(Z-(V+V^T))+\langle Z-(V+V^T),S\rangle \\ g(V,Z) &= \lambda_1\|Z\|_1+2\lambda_2\,\text{trace}(V)\end{aligned}$$

1.3.3 Literature review

While there is much literature on inference given a graphical model, learning the structure of graphical models plays an important role in biological applications [24], social networks [57], etc. In biological applications, it is often useful to have an interpretable graphical model. A sparse graphical model (e.g. a scale-free network) is much more interpretable than a graph that is densely connected. Thus, much of the literature has focused on learning sparse graphical models with an emphasis on the popular convex model of *graphical lasso* [28]. A whole range of algorithms have been proposed for graphical lasso including GLasso, QUIC, etc [47, 41, 53, 42, 21, 60]. As an extension to learning a single graphical model, there has been literature [19, 74, 65, 38] on learning two or more graphical models where there is a sharing of information between the graphical models. While most of these models promote sparse graphical models with sparse sharing of information, there has also been literature on learning graphical models with specific structures. One such structure is learning graphs with hubs. This has been explored through convex models such as the hub graphical lasso[64] and perturbed node joint graphical lasso [48]. In these models, instead of just learning a sparse graphical model, structured sparse graphical models are learned, thus encoding prior information better than a sparse graphical model. Other examples of learning structured graphical models include a low-rank and sparse model which arises in the context of learning latent variables in a graphical model [17]. There are a few efficient algorithms that have been proposed for graphical lasso that can scale to large problem sizes with thousands of variables (see e.g. [42, 21]). On the other hand, existing algorithms (e.g. ADMM) for learning structured graphical models for the applications mentioned above mostly use a SVD based approach which can be very expensive when scaling to larger problem sizes. In [21] an SVD-free proximal Newton method is proposed to solve (1.3.1) with only one variable V and without any explicit constraint. Although this method applies to hub graphical lasso [64], it doesn't appply to other problems such as hub covariance selection [64], latent variable selection [17] and problems that have an explicit positive definite constraint. Similarly the Glasso algorithm for graphical lasso [28] can be extended to solve the hub graphical lasso (see chapter 8), but it doesn't easily extend to solve other problems.

In §8.1, we introduce the framework of SBCD algorithm. In §8.2, we apply SBCD to learn graphical models with hub structures and extensive numerical results are presented. In §8.3, SBCD is applied to solve the latent variable selection problem. Our contribution is summarized as

1. We apply and specialize the SBCD algorithm to the following problems: hub graphical lasso, hub covariance selection problem [64] and latent variable selection problem [17]. We propose efficient ways to ensure positive definiteness of iterates in our algorithm. Specifically, we give precise step-size bounds for which a rank two update to a positive definite matrix preserves positive definiteness. This proves useful for designing block-coordinate descent algorithms for problems involve a positive definite constraint such as hub graphical lasso, hub covariance selection problem [64], and latent variable selection problem. The cost of this update is $O(p^2)$. For hub graphical lasso and hub covariance selection problem, we propose an active set strategy which speeds up SBCD significantly. For latent variable selection application [17], we propose an efficient method to compute and update a low-rank factorization for a low-rank positive semi-definite matrix. We show that our proposed SBCD method converges to a stationary point of a non convex formulation. A practical version of SBCD for latent variable selection problem is introduced to reduce the per iteration cost from $O(p^3)$ to $O(rp^2)$ where r is the rank for the latent part.

2. Our empirical results for the above problems demonstrate the efficiency of our algorithm. As an example, on synthetic data sets for hub graphical lasso we observe that SBCD with active set is faster than the SVD-based ADMM algorithm by a factor of 200 for $p = 1000$. Similarly, for a real data example for hub graphical lasso with $p = 500$, we are faster than ADMM by a factor of 1000. For latent variable selection problem, our proposed P-SBCD is at least 10 time faster on $p = 1000, 2000, 3000, 4000$.

Chapter 2

ITERATIVE RE-WEIGHTED ALGORITHM (IRWA)

2.1 Introduction

As a first refinement of (1.1.1), we suppose that C has the product space structure

$$C := C_1 \times \cdots \times C_l, \tag{2.1.1}$$

where, for each $i \in \mathcal{I} := \{1, 2, \ldots, l\}$, the set $C_i \subset \mathbb{R}^{m_i}$ is convex and $\sum_{i\in\mathcal{I}} m_i = m$. Conformally decomposing A and b, we write

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_l \end{bmatrix} \quad \text{and} \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_l \end{pmatrix},$$

where, for each $i \in \mathcal{I}$, we have $A_i \in \mathbb{R}^{m_i \times n}$ and $b_i \in \mathbb{R}^{m_i}$. On the product space $\mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_l}$, we define a norm adapted to this structure as

$$\left\|(y_1^T, y_2^T, \ldots, y_l^T)^T\right\| := \sum_{i\in\mathcal{I}} \|y_i\|_2 \,. \tag{2.1.2}$$

It is easily verified that the corresponding dual norm is

$$\|y\|_* = \sup_{i\in\mathcal{I}} \|y_i\|_2 \,.$$

With this notation, we may write

$$\operatorname{dist}(y \mid C) = \sum_{i\in\mathcal{I}} \operatorname{dist}_2(y_i \mid C_i)\,, \tag{2.1.3}$$

where, for any set S, we define the distance function $\operatorname{dist}_2(y \mid S) := \inf_{z\in S} \|y - z\|_2$. Hence, with $\varphi(x) := g^T x + \frac{1}{2} x^T H x$, subproblem (1.1.2) takes the form

$$\min_{x\in X} \; J_0(x), \quad \text{where} \quad J_0(x) := \varphi(x) + \sum_{i\in\mathcal{I}} \operatorname{dist}_2(A_i x + b_i \mid C_i)\,. \tag{2.1.4}$$

Throughout our algorithm development and analysis, it is important to keep in mind that $\|y\| \neq \|y\|_2$ since we make heavy use of *both* of these norms.

Example 2.1.1 (Intersections of Convex Sets)**.** In many applications, the affine constraint has the representation $\hat{A}x + \hat{b} \in \hat{C} := \bigcap_{i \in \mathcal{I}} C_i$, where $C_i \subset \mathbb{R}^{m_i}$ is non-empty, closed, and convex for each $i \in \mathcal{I}$. Problems of this type are easily modeled in our framework by setting $A_i := \hat{A}$ and $b_i := \hat{b}$ for each $i \in \mathcal{I}$, and $C := C_1 \times \cdots \times C_l$.

2.2 Notation

Much of the notation that we use is standard and based on that employed in [59]. For convenience, we review some of this notation here. The set $\mathbb{R}^n$ is the real n-dimensional Euclidean space with $\mathbb{R}^n_+$ being the positive orthant in $\mathbb{R}^n$ and $\mathbb{R}^n_{++}$ the interior of $\mathbb{R}^n_+$. The set of real $m \times n$ matrices will be denoted as $\mathbb{R}^{m \times n}$. The Euclidean norm on $\mathbb{R}^n$ is denoted $\|\cdot\|_2$, and its closed unit ball is $\mathbb{B}_2 := \{x \mid \|x\|_2 \leq 1\}$. The closed unit ball of the norm defined in (2.1.2) will be denoted by $\mathbb{B}$. Vectors in $\mathbb{R}^n$ will be considered as column vectors and so we can write the standard inner product on $\mathbb{R}^n$ as $\langle u, v \rangle := u^T v$ for all $\{u, v\} \subset \mathbb{R}^n$. The set $\mathbb{N}$ is the set of natural numbers $\{1, 2, \dots\}$. Given $\{u, v\} \subset \mathbb{R}^n$, the line segment connecting them is denoted by $[u, v]$. Given a set $X \subset \mathbb{R}^n$, we define the convex indicator for X by

$$\delta(x \mid X) := \begin{cases} 0 & \text{if } x \in X, \\ +\infty & \text{if } x \notin X, \end{cases}$$

and its support function by

$$\delta^*(y \mid X) := \sup_{x \in X} \langle y, x \rangle .$$

A function $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{+\infty\}$ is said to be convex if its epigraph,

$$\operatorname{epi}(f) := \{(x, \mu) \mid f(x) \leq \mu\},$$

is a convex set. The function f is said to be closed (or lower semi-continuous) if $\operatorname{epi}(f)$ is closed, and f is said to be proper if $f(x) > -\infty$ for all $x \in \mathbb{R}^n$ and $\operatorname{dom}(f) := \{x \mid f(x) < \infty\} \neq \emptyset$. If f is convex, then the subdifferential of f at $\bar{x}$ is given by

$$\partial f(\bar{x}) := \{z \mid f(\bar{x}) + \langle z, x - \bar{x} \rangle \leq f(x) \ \forall\, x \in \mathbb{R}^n\}.$$

Given a closed convex $X \subset \mathbb{R}^n$, the normal cone to X at a point $\bar{x} \in X$ is given by

$$N(\bar{x} \mid X) := \{z \mid \langle z, x - \bar{x} \rangle \leq 0 \ \forall\, x \in X\}.$$

It is well known that $N(\bar{x} \mid X) = \partial\delta(\bar{x} \mid X)$; e.g., see [59]. Given a set $S \subset \mathbb{R}^m$ and a matrix $M \in \mathbb{R}^{m\times n}$, the inverse image of S under M is given by

$$M^{-1}S := \{x \mid Mx \in S\}.$$

Since the set C in (2.1.1) is non-empty, closed, and convex, the distance function $\operatorname{dist}(y \mid C)$ is convex. Using the techniques of [59], it is easily shown that the subdifferential of the distance function (2.1.3) is

$$\partial\operatorname{dist}(p \mid C) = \partial\operatorname{dist}_2(p_1 \mid C_1) \times \cdots \times \partial\operatorname{dist}_2(p_l \mid C_l), \tag{2.2.1}$$

where, for each $i \in \mathcal{I}$, we have

$$\partial\operatorname{dist}_2(p_i \mid C_i) = \begin{cases} \frac{(I-P_{C_i})p_i}{\|(I-P_{C_i})p_i\|_2} & \text{if } i \notin \mathcal{A}(p), \\ \mathbb{B}_2 \cap N(p_i \mid C_i) & \text{if } i \in \mathcal{A}(p). \end{cases} \tag{2.2.2}$$

Here, we have defined

$$\mathcal{A}(p) := \{i \in \mathcal{I} \mid \operatorname{dist}_2(p_i \mid C_i) = 0\} \quad \forall\, p \in \mathbb{R}^m,$$

and let $P_C(p)$ denote the projection of p onto the set C (see Theorem 2.3.2).

Since we will be working on the product space $\mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_l}$, we will need notation for the components of the vectors in this space. Given a vector $w \in \mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_l}$, we denote the components in $\mathbb{R}^{m_i}$ by w_i and the jth component of w_i by w_{ij} for $j = 1, \ldots, m_i$ and $i \in \mathcal{I}$ so that $w = (w_1^T, \ldots, w_l^T)^T$. Correspondingly, given vectors $w_i \in \mathbb{R}^{m_i}$ for $i \in \mathcal{I}$, we denote by $w \in \mathbb{R}^m$ the vector $w = (w_1^T, \ldots, w_l^T)^T$.

2.3 IRWA

We now describe an iterative algorithm for minimizing the function J_0 in (2.1.4), where in each iteration one solves a subproblem whose objective is the sum of φ and a weighted linear least-squares term. An advantage of this approach is that the subproblems can be solved using matrix-free methods, e.g., the conjugate gradient (CG), projected gradient, and Lanczos [34] methods. The objectives of the subproblems are localized approximations to J_0 based on projections. In this manner, we will make use of the following theorem.

Theorem 2.3.1. *[73] Let $C \subset \mathbb{R}^m$ be non-empty, closed, and convex. Then, to every $y \in \mathbb{R}^m$, there is a unique $\bar{y} \in C$ such that*

$$\|y - \bar{y}\|_2 = \operatorname{dist}_2 (y \mid C) .$$

We call $\bar{y} = P_C(y)$ the projection of y onto C. Moreover, the following hold:

1. *$\bar{y} = P_C(y)$ if and only if $\bar{y} \in C$ and $(y - \bar{y}) \in N (\bar{y} \,|\, C)$ [58].*

2. *For all $\{y, z\} \subset \mathbb{R}^m$, the operator P_C yields*

$$\|P_C(y) - P_C(z)\|_2^2 + \|(I - P_C)y - (I - P_C)z\|_2^2 \leq \|y - z\|_2^2 .$$

Since H is symmetric and positive semi-definite, there exists $A_0 \in \mathbb{R}^{m_0 \times n}$, where $m_0 := \operatorname{rank}(H)$, such that $H = A_0^T A_0$. We use this representation for H in order to simplify our mathematical presentation; this factorization is not required in order to implement our methods. Define $b_0 := 0 \in \mathbb{R}^n$, $C_0 := \{0\} \subset \mathbb{R}^n$, and $\mathcal{I}_0 := \{0\} \cup \mathcal{I} = \{0, 1, \ldots, l\}$. Using this notation, we define our local approximation to J_0 at a given point $\tilde{x}$ and with a given *relaxation vector* $\epsilon \in \mathbb{R}^l_{++}$ by

$$\hat{G}_{(\tilde{x},\epsilon)}(x) := g^T x + \frac{1}{2} \sum_{i \in \mathcal{I}_0} w_i(\tilde{x}, \epsilon) \, \|A_i x + b_i - P_{C_i}(A_i \tilde{x} + b_i)\|_2^2 ,$$

where, for any $x \in \mathbb{R}^n$, we define

$$w_0(x, \epsilon) := 1, \quad w_i(x, \epsilon) := \left(\operatorname{dist}_2^2(A_i x + b_i \mid C_i) + \epsilon_i^2\right)^{-1/2} \quad \forall\, i \in \mathcal{I}, \tag{2.3.1}$$

$$\text{and} \quad W(x, \epsilon) := \operatorname{diag}(w_0(x, \epsilon) I_{m_0}, \ldots, w_l(x, \epsilon) I_{m_l}).$$

Define

$$\tilde{A} := \begin{bmatrix} A_0 \\ A \end{bmatrix} . \tag{2.3.2}$$

We now state the algorithm.

We now describe an iterative algorithm for minimizing the function J_0 in (2.1.4), where in each iteration one solves a subproblem whose objective is the sum of φ and a weighted linear least-squares term. An advantage of this approach is that the subproblems can be

solved using matrix-free methods, e.g., the conjugate gradient (CG), projected gradient, and Lanczos [34] methods. The objectives of the subproblems are localized approximations to J_0 based on projections. In this manner, we will make use of the following theorem.

Theorem 2.3.2. *[73] Let $C \subset \mathbb{R}^m$ be non-empty, closed, and convex. Then, to every $y \in \mathbb{R}^m$, there is a unique $\bar{y} \in C$ such that*

$$\|y - \bar{y}\|_2 = \operatorname{dist}_2(y \mid C).$$

We call $\bar{y} = P_C(y)$ the projection of y onto C. Moreover, the following hold:

(1) $\bar{y} = P_C(y)$ if and only if $\bar{y} \in C$ and $(y - \bar{y}) \in N(\bar{y} \mid C)$.

(2) For all $\{y, z\} \subset \mathbb{R}^m$, the operator P_C yields

$$\|P_C(y) - P_C(z)\|_2^2 + \|(I - P_C)y - (I - P_C)z\|_2^2 \leq \|y - z\|_2^2.$$

Since H is symmetric and positive semi-definite, there exists $A_0 \in \mathbb{R}^{m_0 \times n}$, where $m_0 := \operatorname{rank}(H)$, such that $H = A_0^T A_0$. We use this representation for H in order to simplify our mathematical presentation; this factorization is not required in order to implement our methods. Define $b_0 := 0 \in \mathbb{R}^n$, $C_0 := \{0\} \subset \mathbb{R}^n$, and $\mathcal{I}_0 := \{0\} \cup \mathcal{I} = \{0, 1, \ldots, l\}$. Using this notation, we define our local approximation to J_0 at a given point $\tilde{x}$ and with a given *relaxation vector* $\epsilon \in \mathbb{R}^l_{++}$ by

$$\hat{G}_{(\tilde{x},\epsilon)}(x) := g^T x + \frac{1}{2} \sum_{i \in \mathcal{I}_0} w_i(\tilde{x}, \epsilon) \|A_i x + b_i - P_{C_i}(A_i \tilde{x} + b_i)\|_2^2, \tag{2.3.3}$$

where, for any $x \in \mathbb{R}^n$, we define

$$w_0(x, \epsilon) := 1, \quad w_i(x, \epsilon) := \left(\operatorname{dist}_2^2(A_i x + b_i \mid C_i) + \epsilon_i^2\right)^{-1/2} \quad \forall\, i \in \mathcal{I}, \tag{2.3.4}$$
$$\text{and} \quad W(x, \epsilon) := \operatorname{diag}(w_0(x, \epsilon) I_{m_0}, \ldots, w_l(x, \epsilon) I_{m_l}).$$

Define

$$\tilde{A} := \begin{bmatrix} A_0 \\ A \end{bmatrix}. \tag{2.3.5}$$

We now state the algorithm.

Iterative Re-Weighting Algorithm (IRWA)

Step 0: (Initialization) Choose an initial point $x^0 \in X$, an initial relaxation vector $\epsilon^0 \in \mathbb{R}^l_{++}$, and scaling parameters $\eta \in (0,1)$, $\gamma > 0$, and $M > 0$. Let $\sigma \geq 0$ and $\sigma' \geq 0$ be two scalars which serve as termination tolerances for the stepsize and relaxation parameter, respectively. Set $k := 0$.

Step 1: (Solve the re-weighted subproblem for x^{k+1})

Compute a solution x^{k+1} to the problem

$$\mathcal{G}(x^k, \epsilon^k): \quad \min_{x \in X} \hat{G}_{(x^k, \epsilon^k)}(x). \tag{2.3.6}$$

Step 2: (Set the new relaxation vector ϵ^{k+1})

Set

$$q_i^k := A_i(x^{k+1} - x^k) \;\text{ and }\; r_i^k := (I - P_{C_i})(A_i x^k + b_i) \quad \forall\, i \in \mathcal{I}_0.$$

If

$$\left\| q_i^k \right\|_2 \leq M \left[\left\| r_i^k \right\|_2^2 + (\epsilon_i^k)^2 \right]^{\frac{1}{2}+\gamma} \quad \forall\, i \in \mathcal{I}, \tag{2.3.7}$$

then choose $\epsilon^{k+1} \in (0,\ \eta\epsilon^k]$; else, set $\epsilon^{k+1} := \epsilon^k$.

Step 3: (Check stopping criteria)

If $\left\| x^{k+1} - x^k \right\|_2 \leq \sigma$ and $\left\| \epsilon^k \right\|_2 \leq \sigma'$, then stop; else, set $k := k+1$ and go to Step 1.

Remark 2.3.3. In cases where $C_i = \{0\} \subset \mathbb{R}$ for all $i \in \mathcal{I}$ and $\phi \equiv 0$, this algorithm has a long history in the literature. Two early references are [3] and [61]. In such cases, the algorithm reduces to the classical algorithm for minimizing $\|Ax + b\|_1$ using iteratively re-weighted least-squares.

Remark 2.3.4. If there exists z_0 such that $A_0^T z_0 = g$, then, by setting $b_0 := z_0$, the linear term in the definition of $\hat{G}$, namely $g^T x$, can be eliminated.

Remark 2.3.5. It is often advantageous to employ a stopping criterion based on a percent reduction in the duality gap rather than the stopping criteria given in Step 3 above [7, 9].

In such cases, one keeps track of both the primal objective values $J_0^k := J_0(x^k)$ and the dual objective values

$$\hat{J}_0^k := \frac{1}{2}(g + A^T\tilde{u}^k)^T H^{-1}(g + A^T\tilde{u}^k) - b^T\tilde{u}^k + \sum_{i\in\mathcal{I}} \delta^*\left(\tilde{u}_i^k \mid C_i\right),$$

where the vectors $\tilde{u}^k := W_k r^k$ are dual feasible (see (4.2.2) for a discussion of the dual problem). Given $\sigma \in (0,1)$, Step 3 above can be replaced by

Step 3': (Check stopping criterion)

If $(J_0^1 + \hat{J}_0^k) \leq \sigma(J_0^1 - J_0^k)$, then stop; else, set $k := k+1$ and go to Step 1.

This is the stopping criterion employed in some of our numerical experiments. Nonetheless, for our analysis, we employ Step 3 as it is stated in the formal description of IRWA for those instances when dual values $\hat{J}_0^k$ are unavailable, such as when these computations are costly or subject to error.

2.3.1 Smooth approximation to J_0

Our analysis of IRWA is based on a smooth approximation to J_0. Given $\epsilon \in \mathbb{R}^l_+$, define the ϵ-smoothing of J_0 by

$$J(x,\epsilon) := \varphi(x) + \sum_{i\in\mathcal{I}} \sqrt{\operatorname{dist}_2^2(A_i x + b_i \mid C_i) + \epsilon_i^2}\,. \tag{2.3.8}$$

Note that $J_0(x) \equiv J(x,0)$ and that $J(x,\epsilon)$ is jointly convex in (x,ϵ) since

$$J(x,\epsilon) = \varphi(x) + \sum_{i\in\mathcal{I}} \operatorname{dist}_2\left(\begin{bmatrix} A_i & 0 \\ 0 & e_i^T \end{bmatrix}\begin{pmatrix} x \\ \epsilon \end{pmatrix} + \begin{pmatrix} b_i \\ 0 \end{pmatrix} \mid C_i \times \{0\}\right),$$

where e_i is the ith unit coordinate vector. By [59, Corollary 10.11], (2.2.1), and (2.2.2),

$$\partial J_0(x) = \partial_x J(x,0) = \nabla\varphi(x) + A^T \partial \operatorname{dist}(\cdot \mid C)(Ax+b) = \tag{2.3.9}$$
$$\nabla\varphi(x) + \sum_{i\notin\mathcal{A}(Ax+b)} A_i^T \frac{(I - P_{C_i})(A_i x + b_i)}{\|(I - P_{C_i})(A_i x + b_i)\|_2} + \sum_{i\in\mathcal{A}(Ax+b)} A_i^T(\mathbb{B}_2 \cap N(A_i x + b_i \,|\, C_i)).$$

Given $\tilde{x} \in \mathbb{R}^n$ and $\tilde{\epsilon} \in \mathbb{R}^l_{++}$, we define a weighted approximation to $J(\cdot,\tilde{\epsilon})$ at $\tilde{x}$ by

$$G_{(\tilde{x},\tilde{\epsilon})}(x) := g^T x + \frac{1}{2}\sum_{i\in\mathcal{I}_0} w_i(\tilde{x},\tilde{\epsilon}) \operatorname{dist}_2^2(A_i x + b_i \mid C_i)\,.$$

Observe that G is defined similarly to $\hat{G}$ from (2.3.3), except that in $\hat{G}$ the distance from $A_i x + b_i$ is taken to the projection of $A_i \tilde{x} + b_i$, not to that of $A_i x + b_i$ itself.

We have the following fundamental fact about solutions of $\mathcal{G}(\tilde{x}, \tilde{\epsilon})$ defined by (2.3.6).

Lemma 2.3.6. *Let $\tilde{x} \in X$, $\tilde{\epsilon} \in \mathbb{R}^l_{++}$, $\hat{\epsilon} \in (0, \tilde{\epsilon}]$, and $\hat{x} \in \operatorname{argmin}_{x \in X} \hat{G}_{(\tilde{x},\tilde{\epsilon})}(x)$. Set $\tilde{w}_i := w_i(\tilde{x}, \tilde{\epsilon})$ and $q_i := A_i(\hat{x} - \tilde{x})$ for $i \in \mathcal{I}_0$, $\tilde{W} := W(\tilde{x}, \tilde{\epsilon})$, and $q := (q_0^T, \ldots, q_l^T)^T$. Then,*

$$G_{(\tilde{x},\tilde{\epsilon})}(\hat{x}) - G_{(\tilde{x},\tilde{\epsilon})}(\tilde{x}) \leq -\frac{1}{2} q^T \tilde{W} q \tag{2.3.10}$$

and

$$J(\tilde{x}, \tilde{\epsilon}) - J(\hat{x}, \hat{\epsilon}) \geq \frac{1}{2} q^T \tilde{W} q. \tag{2.3.11}$$

Proof. We first prove (2.3.10). Define $\hat{r}_i := (I - P_{C_i})(A_i \hat{x} + b_i)$ and $\tilde{r}_i := (I - P_{C_i})(A_i \tilde{x} + b_i)$ for $i \in \mathcal{I}_0$, and set $\hat{r} := (\hat{r}_0^T, \ldots, \hat{r}_l^T)^T$ and $\tilde{r} := (\tilde{r}_0^T, \ldots, \tilde{r}_l^T)^T$. Since $\hat{x} \in \operatorname{argmin}_{x \in X} \hat{G}_{(\tilde{x},\tilde{\epsilon})}(x)$, there exists $\hat{v} \in N(\hat{x} \,|\, X)$ such that

$$0 = g + \tilde{A}^T \tilde{W}(\tilde{A}\hat{x} + b - P_C(\tilde{A}\tilde{x} + b)) + \hat{v} = g + \tilde{A}^T \tilde{W}(q + \tilde{r}) + \hat{v},$$

or, equivalently,

$$-\hat{v} = g + \tilde{A}^T \tilde{W}(q + \tilde{r}). \tag{2.3.12}$$

Moreover, by the definition of the projection operator P_{C_i}, we know that

$$\|\hat{r}_i\|_2 = \|(I - P_{C_i})(A_i \hat{x} + b_i)\|_2 \leq \|A_i \hat{x} + b_i - P_{C_i}(A_i \tilde{x} + b_i)\|_2 = \|q_i + \tilde{r}_i\|_2$$

so that

$$\|\hat{r}_i\|_2^2 - \|q_i + \tilde{r}_i\|_2^2 \leq 0 \quad \forall\, i \in \mathcal{I}_0. \tag{2.3.13}$$

Therefore,

$$
\begin{aligned}
& G_{(\tilde{x},\tilde{\epsilon})}(\hat{x}) - G_{(\tilde{x},\tilde{\epsilon})}(\tilde{x}) \\
&= g^T(\hat{x}-\tilde{x}) + \frac{1}{2}\textstyle\sum_{i\in\mathcal{I}_0} \tilde{w}_i[\|\hat{r}_i\|_2^2 - \|\tilde{r}_i\|_2^2] \\
&= g^T(\hat{x}-\tilde{x}) + \frac{1}{2}\textstyle\sum_{i\in\mathcal{I}_0} \tilde{w}_i[(\|\hat{r}_i\|_2^2 - \|q_i+\tilde{r}_i\|_2^2) + (\|q_i+\tilde{r}_i\|_2^2 - \|\tilde{r}_i\|_2^2)] \\
&\le g^T(\hat{x}-\tilde{x}) + \frac{1}{2}\textstyle\sum_{i\in\mathcal{I}_0} \tilde{w}_i[\|q_i+\tilde{r}_i\|_2^2 - \|\tilde{r}_i\|_2^2] \qquad \text{(by (2.3.13))} \\
&= g^T(\hat{x}-\tilde{x}) + \frac{1}{2}\textstyle\sum_{i\in\mathcal{I}_0} \tilde{w}_i[\|q_i\|_2^2 + 2\langle q_i,\, \tilde{r}_i\rangle] \\
&= g^T(\hat{x}-\tilde{x}) + \frac{1}{2}\textstyle\sum_{i\in\mathcal{I}_0} \tilde{w}_i[-\|q_i\|_2^2 + 2\langle q_i,\, q_i+\tilde{r}_i\rangle] \\
&= -\frac{1}{2}q^T\tilde{W}q + g^T(\hat{x}-\tilde{x}) + q^T\tilde{W}(q+\tilde{r}) \\
&= -\frac{1}{2}q^T\tilde{W}q + (\hat{x}-\tilde{x})^T(g + \tilde{A}^T\tilde{W}(q+\tilde{r})) \\
&= -\frac{1}{2}q^T\tilde{W}q + (\tilde{x}-\hat{x})^T\hat{v} \qquad \text{(by (2.3.12))} \\
&\le -\frac{1}{2}q^T\tilde{W}q,
\end{aligned}
$$

where the final inequality follows since $\tilde{x} \in X$ and $\hat{v} \in N(\hat{x} \,|\, X)$.

We now prove (2.3.11). Since $\sqrt{t}$ is a concave function of t on $\mathbb{R}_+$, we have

$$\sqrt{\hat{t}} \le \sqrt{\tilde{t}} + \frac{\hat{t}-\tilde{t}}{2\sqrt{\tilde{t}}} \quad \forall\, \{\hat{t},\tilde{t}\} \subset \mathbb{R}_{++},$$

and so, for $i \in \mathcal{I}$, we have

$$
\begin{aligned}
& \sqrt{\mathrm{dist}_2^2(A_i\hat{x}+b_i \mid C_i) + \tilde{\epsilon}_i^2} \\
&\le \sqrt{\mathrm{dist}_2^2(A_i\tilde{x}+b_i \mid C_i) + \tilde{\epsilon}_i^2} + \frac{\mathrm{dist}_2^2(A_i\hat{x}+b_i \mid C_i) - \mathrm{dist}_2^2(A_i\tilde{x}+b_i \mid C_i)}{2\sqrt{\mathrm{dist}_2^2(A_i\tilde{x}+b_i \mid C_i) + \tilde{\epsilon}_i^2}}. \qquad (2.3.14)
\end{aligned}
$$

Hence,

$$
\begin{aligned}
J(\hat{x},\hat{\epsilon}) \le J(\hat{x},\tilde{\epsilon}) &= \varphi(\hat{x}) + \sum_{i\in\mathcal{I}} \sqrt{\mathrm{dist}_2^2(A_i\hat{x}+b_i \mid C_i) + \tilde{\epsilon}_i^2} \\
&\le J(\tilde{x},\tilde{\epsilon}) + (\varphi(\hat{x}) - \varphi(\tilde{x})) + \frac{1}{2}\sum_{i\in\mathcal{I}} \frac{\mathrm{dist}_2^2(A_i\hat{x}+b_i \mid C_i) - \mathrm{dist}_2^2(A_i\tilde{x}+b_i \mid C_i)}{\sqrt{\mathrm{dist}_2^2(A_i\tilde{x}+b_i \mid C_i) + \tilde{\epsilon}_i^2}} \\
&= J(\tilde{x},\tilde{\epsilon}) + [G_{(\tilde{x},\tilde{\epsilon})}(\hat{x}) - G_{(\tilde{x},\tilde{\epsilon})}(\tilde{x})] \\
&\le J(\tilde{x},\tilde{\epsilon}) - \frac{1}{2}q^T\tilde{W}q,
\end{aligned}
$$

where the first inequality follows from $\hat{\epsilon} \in (0, \tilde{\epsilon}]$, the second inequality follows from (2.3.14), and the third inequality follows from (2.3.10). □

2.3.2 Coercivity of J

Lemma 2.3.6 tells us that IRWA is a descent method for the function J. Consequently, both the existence of solutions to (2.1.4) as well as the existence of cluster points to IRWA can be guaranteed by understanding conditions under which the function J is coercive, or equivalently, conditions that guarantee the boundedness of the lower level sets of J over X. For this, we need to consider the asymptotic geometry of J and X.

Definition 2.3.7. [59, Definition 3.3] Given $Y \subset \mathbb{R}^m$, the horizon cone of Y is

$$Y^\infty := \left\{ z \;\middle|\; \exists\, t^k \downarrow 0,\ \{y^k\} \subset Y \text{ such that } t^k y^k \to z \right\}.$$

We have the basic facts about horizon cones given in the following proposition.

Proposition 2.3.8. *The following hold:*

(1) [59, Theorem 3.5] The set $Y \subset \mathbb{R}^m$ is bounded if and only if $Y^\infty = \{0\}$.

(2) [59, Exercise 3.11] Given $Y_i \subset \mathbb{R}^{m_i}$ for $i \in \mathcal{I}$, we have $(Y_1 \times \cdots \times Y_l)^\infty = Y_1^\infty \times \cdots \times Y_l^\infty$.

(3) [59, Theorem 3.6] If $C \subset \mathbb{R}^m$ is non-empty, closed, and convex, then

$$C^\infty = \{ z \mid C + z \subset C \}.$$

We now prove the following result about the lower level sets of J.

Theorem 2.3.9. *Let $\alpha > 0$ and $\epsilon \in \mathbb{R}^l_+$ be such that the set*

$$L(\alpha, \epsilon) := \{ x \in X \mid J(x, \epsilon) \leq \alpha \}$$

is non-empty. Then,

$$L(\alpha, \epsilon)^\infty = \left\{ \bar{x} \in X^\infty \;\middle|\; g^T \bar{x} \leq 0,\ H\bar{x} = 0,\ A\bar{x} \in C^\infty \right\}. \tag{2.3.15}$$

Moreover, $L(\alpha, \epsilon)$ is compact for all $(\alpha, \epsilon) \in \mathbb{R}^{l+1}_+$ if and only if

$$\left[\bar{x} \in X^\infty \cap \ker(H) \cap A^{-1} C^\infty \text{ satisfies } g^T \bar{x} \leq 0 \right] \iff \bar{x} = 0. \tag{2.3.16}$$

Proof. Let $x \in L(\alpha, \epsilon)$ and let $\bar{x}$ be an element of the set on the right-hand side of (2.3.15). Then, by Proposition 2.3.8, for all $\lambda \geq 0$ we have $x + \lambda\bar{x} \in X$ and $\lambda A_i\bar{x} + C_i \subset C_i$ for all $i \in \mathcal{I}$, and so for each $i \in \mathcal{I}$ we have

$$\begin{aligned} \operatorname{dist}(A_i(x+\lambda\bar{x}) + b_i \mid C_i) &\leq \operatorname{dist}(A_i(x+\lambda\bar{x}) + b_i \mid \lambda A_i\bar{x} + C_i) \\ &= \operatorname{dist}((A_ix + b_i) + \lambda A_i\bar{x} \mid \lambda A_i\bar{x} + C_i) \\ &= \operatorname{dist}(A_ix + b_i \mid C_i). \end{aligned}$$

Therefore,

$$\begin{aligned} J(x+\lambda\bar{x}, \epsilon) &= \varphi(x) + \lambda g^T\bar{x} + \sum_{i\in\mathcal{I}} \sqrt{\operatorname{dist}_2^2(A_i(x+\lambda\bar{x}) + b_i \mid C_i) + \epsilon_i^2} \\ &\leq \varphi(x) + \sum_{i\in\mathcal{I}} \sqrt{\operatorname{dist}_2^2(A_ix + b_i \mid C_i) + \epsilon_i^2} = J(x, \epsilon) \leq \alpha. \end{aligned}$$

Consequently, $\bar{x} \in L(\alpha, \epsilon)^\infty$.

On the other hand, let $\bar{x} \in L(\alpha, \epsilon)^\infty$. We need to show that $\bar{x}$ is an element of the set on the right-hand side of (2.3.15). For this, we may as well assume that $\bar{x} \neq 0$. By the fact that $\bar{x} \in L(\alpha, \epsilon)^\infty$, there exists $t^k \downarrow 0$ and $\{x^k\} \subset X$ such that $J(x^k, \epsilon) \leq \alpha$ and $t^kx^k \to \bar{x}$. Consequently, $\bar{x} \in X^\infty$. Moreover,

$$g^T(t^kx^k) = t^k(g^Tx^k) \leq t^kJ(x^k, \epsilon) \leq t^k\alpha \to 0$$

and so

$$\begin{aligned} 0 \leq \left\|A_0(t^kx^k)\right\|^2 &= (t^kx^k)^TH(t^kx^k) = (t^k)^2(x^k)^THx^k \\ &\leq (t^k)^22(J(x^k, \epsilon) - g^Tx^k) \leq (t^k)^22\alpha - t^k2g^T(t^kx^k) \to 0. \end{aligned}$$

Therefore, $g^T\bar{x} \leq 0$ and $H\bar{x} = 0$. Now, define $z^k := P_C(Ax^k + b)$ for $k \in \mathbb{N}$. Then, by Theorem 2.3.2(2), we have

$$\left\|z^k\right\|_2 \leq \left\|(I - P_C)(Ax^k + b)\right\|_2 + \left\|Ax^k + b\right\|_2 \leq \alpha + \left\|Ax^k + b\right\|_2,$$

which, since $A(t^kx^k) + t^kb \to A\bar{x}$, implies that the sequence $\{t^kz^k\}$ is bounded. Hence, without loss of generality, we can assume that there is a vector $\bar{z}$ such that $t^kz^k \to \bar{z}$, where

by the definition of z^k we have $\bar{z} \in C^\infty$. But,

$$0 \leq \left\| A(t^k x^k) + t^k b - (t^k z^k) \right\|_2 = t^k \mathrm{dist}_2 \left(Ax^k + b \mid C \right)$$
$$\leq t^k J(x^k, \epsilon) - t^k g^T x^k \leq t^k \alpha - g^T (t^k x^k) \to 0,$$

while

$$\left\| A(t^k x^k) + b - (t^k z^k) \right\|_2 \to \| A\bar{x} - \bar{z} \|_2 .$$

Consequently, $\bar{x} \in X^\infty,\ g^T \bar{x} \leq 0,\ H\bar{x} = 0$, and $A\bar{x} \in C^\infty$, which together imply that $\bar{x}$ is in the set on the right-hand side of (2.3.15). □

Corollary 2.3.10. *Suppose that the sequence $\{(x^k, \epsilon^k)\}$ is generated by IRWA with initial point $x^0 \in X$ and relaxation vector $\epsilon^0 \in \mathbb{R}^l_{++}$. Then, $\{x^k\}$ is bounded if* (2.3.16) *is satisfied, which follows if at least one of the following conditions holds:*

(1) X is compact.

(2) H is positive definite.

(3) C is compact and $X^\infty \cap \ker(H) \cap \ker(A) = \{0\}$.

Remark 2.3.11. For future reference, observe that

$$\ker(H) \cap \ker(A) = \ker(\tilde{A}), \tag{2.3.17}$$

where $\tilde{A}$ is defined in (2.3.5).

2.4 Convergence of IRWA

We now return to our analysis of the convergence of IRWA by first proving the following lemma that discusses critical properties of the sequence of iterates computed in the algorithm.

Lemma 2.4.1. *Suppose that the sequence $\{(x^k, \epsilon^k)\}$ is generated by IRWA with initial point $x^0 \in X$ and relaxation vector $\epsilon^0 \in \mathbb{R}^l_{++}$, and, for $k \in \mathbb{N}$, let q_i^k and r_i^k for $i \in \mathcal{I}_0$ be as defined in Step 2 of the algorithm with*

$$q^k := ((q_0^k)^T, \ldots, (q_l^k)^T)^T \quad \text{and} \quad r^k := ((r_0^k)^T, \ldots, (r_l^k)^T)^T.$$

Moreover, for $k \in \mathbb{N}$, define

$$w_i^k := w_i(x^k, \epsilon^k) \text{ for } i \in \mathcal{I}_0 \quad \text{and} \quad W_k := W(x^k, \epsilon^k),$$

and set $S := \left\{k \,\middle|\, \epsilon^{k+1} \leq \eta\epsilon^k\right\}$. Then, the sequence $\{J(x^k, \epsilon^k)\}$ is monotonically decreasing. Moreover, either $\inf_{k\in\mathbb{N}} J(x^k, \epsilon^k) = -\infty$, in which case $\inf_{x\in X} J_0(x) = -\infty$, or the following hold:

(1) $\sum_{k=0}^{\infty} (q^k)^T W_k q^k < \infty$.

(2) $\epsilon^k \to 0$ *and* $H(x^{k+1} - x^k) \to 0$.

(3) $W_k q^k \overset{S}{\to} 0$.

(4) $w_i^k r_i^k = r_i^k / \sqrt{\left\|r_i^k\right\|_2^2 + \epsilon_i^k} \in \mathbb{B}_2 \cap N\left(P_{C_i}(A_i x^k + b_i) \,|\, C_i\right)$, $i \in \mathcal{I}$, $k \in \mathbb{N}$.

(5) $-\tilde{A}^T W_k q^k \in (\nabla\varphi(x^k) + \sum_{i\in\mathcal{I}} A_i^T w_i^k r_i^k) + N\left(x^{k+1} \,|\, X\right)$, $k \in \mathbb{N}$.

(6) If $\{\text{dist}\left(Ax^k + b \,|\, C\right)\}_{k\in S}$ *is bounded, then* $q^k \overset{S}{\to} 0$.

Proof. The fact that $\{J(x^k, \epsilon^k)\}$ is monotonically decreasing is an immediate consequence of the monotonicity of the sequence $\{\epsilon_k\}$, Lemma 2.3.6, and the fact that W_k is positive definite for all $k \in \mathbb{N}$. If $J(x^k, \epsilon^k) \to -\infty$, then $\inf_{x\in X} J_0(x) = -\infty$ since $J_0(x) = J(x, 0) \leq J(x, \epsilon)$ for all $x \in \mathbb{R}^n$ and $\epsilon \in \mathbb{R}^l_+$. All that remains is to show that Parts (1)–(6) hold when $\inf_{k\in\mathbb{N}} J(x^k, \epsilon^k) > -\infty$, in which case we may assume that the sequence $\{J(x^k, \epsilon^k)\}$ is bounded below. We define the lower bound $\tilde{J} := \inf_{k\in\mathbb{N}} J(x^k, \epsilon^k) = \lim_{k\in\mathbb{N}} J(x^k, \epsilon^k)$ for the remainder of the proof.

(1) By Lemma 2.3.6, for every positive integer $\bar{k}$ we have

$$\begin{aligned}
\frac{1}{2}\sum_{k=0}^{\bar{k}} (q^k)^T W_k q^k &\leq \sum_{k=0}^{\bar{k}} [J(x^k, \epsilon^k) - J(x^{k+1}, \epsilon^{k+1})] \\
&= J(x^0, \epsilon^0) - J(x^{\bar{k}+1}, \epsilon^{\bar{k}+1}) \\
&\leq J(x^0, \epsilon^0) - \tilde{J}.
\end{aligned}$$

Therefore, as desired, we have

$$\sum_{k=0}^{\infty}(q^k)^T W_k q^k \leq 2(J(x^0,\epsilon^0)-\tilde{J}) < \infty.$$

(2) Since $\eta \in (0,1)$, if $\epsilon^k \nrightarrow 0$, then there exists an integer $\bar{k} \geq 0$ and a scalar $\bar{\epsilon} > 0$ such that $\epsilon^k = \bar{\epsilon}$ for all $k \geq \bar{k}$. Part (1) implies that $(q^k)^T W_k q^k$ is summable so that $(w_i^k \left\|q_i^k\right\|_2)(\left\|q_i^k\right\|_2) = w_i^k \left\|q_i^k\right\|_2^2 \to 0$ for each $i \in \mathcal{I}_0$. In particular, since $w_0^k := 1$ for all $k \in \mathbb{N}$, this implies that $q_0^k \to 0$, or equivalently that $H(x^{k+1}-x^k) \to 0$. In addition, since for each $i \in \mathcal{I}$ both sequences $\{\left\|q_i^k\right\|_2\}$ and $\{w_i^k \left\|q_i^k\right\|_2\}$ cannot be bounded away from 0, there is a subsequence $\hat{S} \subset \mathbb{N}$ and a partition $\{\mathcal{I}_1, \mathcal{I}_2\}$ of $\mathcal{I}$ such that $\left\|q_i^k\right\|_2 \overset{\hat{S}}{\to} 0$ for all $i \in \mathcal{I}_1$ and $w_i^k \left\|q_i^k\right\|_2 \overset{\hat{S}}{\to} 0$ for all $i \in \mathcal{I}_2$. Hence, there exists $k_0 \in \hat{S}$ such that for all $k \geq k_0$ we have

$$\begin{aligned} \left\|q_i^k\right\|_2 &\leq M\Big[\left\|r_i^k\right\|_2^2 + \bar{\epsilon}_i^2\Big]^{\frac{1}{2}+\gamma} \quad \forall\, i \in \mathcal{I}_1 \\ \text{and} \quad w_i^k \left\|q_i^k\right\|_2 &\leq M\Big[\left\|r_i^k\right\|_2^2 + \bar{\epsilon}_i^2\Big]^{\gamma} \quad \forall\, i \in \mathcal{I}_2. \end{aligned}$$

Therefore, since $w_i^k = (\|r_i^k\|^2 + (\epsilon_i^k)^2)^{-1/2}$, we have for all $k_0 \leq k \in \hat{S}$ that

$$\left\|q_i^k\right\|_2 \leq M\Big[\left\|r_i^k\right\|_2^2 + \bar{\epsilon}_i^2\Big]^{\frac{1}{2}+\gamma} \quad \forall i \in \mathcal{I}.$$

However, for every such k, Step 2 of the algorithm chooses $\epsilon^{k+1} \in (0, \eta\epsilon^k]$. This contradicts the supposition that $\epsilon^k = \bar{\epsilon} > 0$ for all $k \geq \bar{k}$, so we conclude that $\epsilon^k \to 0$.

(3) It has just been shown in Part (2) that $w_0^k q_0^k = q_0^k \to 0$, so we need only show that $w_i^k \left\|q_i^k\right\|_2 \overset{S}{\to} 0$ for each $i \in \mathcal{I}$.

Our first step is to show that for every subsequence $\hat{S} \subset S$ and $i_0 \in \mathcal{I}$, there is a further subsequence $\tilde{S} \subset \hat{S}$ such that $w_{i_0}^k \left\|q_{i_0}^k\right\|_2 \overset{\tilde{S}}{\to} 0$. The proof uses a trick from the proof of Part (2). Let $\hat{S} \subset S$ be a subsequence and $i_0 \in \mathcal{I}$. Part (1) implies that $(w_i^k \left\|q_i^k\right\|_2)(\left\|q_i^k\right\|_2) = w_i^k \left\|q_i^k\right\|_2^2 \to 0$ for each $i \in \mathcal{I}_0$. As in the proof of Part (2), this implies that there is a further subsequence $\tilde{S} \subset \hat{S}$ and a partition $\{\mathcal{I}_1, \mathcal{I}_2\}$ of $\mathcal{I}$ such that $\left\|q_i^k\right\|_2 \overset{\tilde{S}}{\to} 0$ for all $i \in \mathcal{I}_1$ and $w_i^k \left\|q_i^k\right\|_2 \overset{\tilde{S}}{\to} 0$ for all $i \in \mathcal{I}_2$. If $i_0 \in \mathcal{I}_2$, then we would be done, so let us assume that $i_0 \in \mathcal{I}_1$. We can assume that $\tilde{S}$ contains no

subsequence on which $w_{i_0}^k \left\|q_{i_0}^k\right\|_2$ converges to 0 since, otherwise, again we would be done. Hence, we assume that $w_{i_0}^k \left\|q_{i_0}^k\right\|_2 \overset{\tilde{S}}{\nrightarrow} 0$. Since $\left\|q_{i_0}^k\right\|_2 \overset{\tilde{S}}{\to} 0$ as $i_0 \in \mathcal{I}_1$, this implies that there is a subsequence $\tilde{S}_0 \subset \tilde{S}$ such that $w_{i_0}^k \overset{\tilde{S}_0}{\to} \infty$, i.e., $(\left\|r_{i_0}^k\right\|_2^2 + (\epsilon_{i_0}^k)^2) \overset{\tilde{S}_0}{\to} 0$. But, by Step 2 of the algorithm, for all $k \in S$,

$$\left\|q_i^k\right\|_2 \leq M\Big[\left\|r_i^k\right\|_2^2 + (\epsilon_i^k)^2\Big]^{\frac{1}{2}+\gamma} \quad \forall\, i \in \mathcal{I},$$

or, equivalently,

$$w_i^k \left\|q_i^k\right\|_2 \leq M\Big[\left\|r_i^k\right\|_2^2 + (\epsilon_i^k)^2\Big]^{\gamma} \quad \forall\, i \in \mathcal{I},$$

giving the contradiction $w_{i_0}^k \left\|q_{i_0}^k\right\|_2 \overset{\tilde{S}_0}{\to} 0$. Hence, $w_{i_0}^k \left\|q_{i_0}^k\right\|_2 \overset{\tilde{S}}{\to} 0$, and we have shown that for every subsequence $\hat{S} \subset S$ and $i_0 \in \mathcal{I}$, there is $\tilde{S} \subset \hat{S}$ such that $w_i^k \left\|q_i^k\right\|_2 \overset{\tilde{S}}{\to} 0$.

Now, if $W_k q^k \overset{S}{\nrightarrow} 0$, then there would exist a subsequence $\hat{S} \subset S$ and an index $i \in \mathcal{I}$ such that $\{w_i^k \left\|q_i^k\right\|_2\}_{k\in\hat{S}}$ remains bounded away from 0. But, by what we have just shown in the previous paragraph, $\hat{S}$ contains a further subsequence $\tilde{S} \subset \hat{S}$ with $w_i^k \left\|q_i^k\right\|_2 \overset{\tilde{S}}{\to} 0$. This contradiction establishes the result.

(4) By Theorem 2.3.2, we have

$$r_i^k \in N\Big(P_{C_i}(A_i x^k + b_i) \,|\, C_i\Big) \quad \forall\, i \in \mathcal{I}_0,\ k \in \mathbb{N},$$

from which the result follows.

(5) By convexity, the condition $x^{k+1} \in \operatorname{argmin}_{x\in X} \hat{G}_{(x^k,\epsilon^k)}(x)$ is equivalent to

$$\begin{aligned} 0 &\in \nabla_x \hat{G}_{(x^k,\epsilon^k)}(x^{k+1}) + N\Big(x^{k+1} \,|\, X\Big) \\ &= g + \sum_{i\in\mathcal{I}_0} A_i^T w_i^k (q_i^k + r_i^k) + N\Big(x^{k+1} \,|\, X\Big) \\ &= \tilde{A}^T W_k q^k + \nabla\varphi(x^k) + \sum_{i\in\mathcal{I}} A_i^T w_i^k r_i^k + N\Big(x^{k+1} \,|\, X\Big). \end{aligned}$$

(6) Let $i \in \mathcal{I}$. We know from Part (3) that $w_i^k \left\|q_i^k\right\|_2 \overset{S}{\to} 0$. If $\left\|q_i^k\right\|_2 \overset{S}{\nrightarrow} 0$, then there exists a subsequence $\hat{S} \subset S$ such that $\{\left\|q_i^k\right\|_2\}_{k\in\hat{S}}$ is bounded away from 0, which would imply that $(\left\|r_i^k\right\|_2^2 + (\epsilon_i^k)^2)^{-1/2} = w_i^k \overset{\hat{S}}{\to} 0$. But then $\left\|r_i^k\right\|_2 \overset{\hat{S}}{\to} \infty$ since $0 \leq \epsilon^k \leq \epsilon^0$, which contradicts the boundedness of $\{\operatorname{dist}\big(Ax^k + b \,|\, C\big)\}_{k\in S}$.

□

In the next result, we give conditions under which every cluster point of the subsequence $\{x^k\}_{k\in S}$ is a solution to $\min_{x\in X} J_0(x)$, where S is defined in Lemma 2.4.1. Since J_0 is convex, this is equivalent to showing that $0 \in \partial J_0(\bar{x}) + N(\bar{x} \,|\, X)$.

Theorem 2.4.2. *Suppose that the sequence $\{(x^k, \epsilon^k)\}$ is generated by IRWA with initial point $x^0 \in X$ and relaxation vector $\epsilon^0 \in \mathbb{R}^l_{++}$, and that the sequence $\{J(x^k, \epsilon^k)\}$ is bounded below. Let S be defined as in Lemma 2.4.1. If either*

(a) $\ker(A) \cap \ker(H) = \{0\}$ and $\{\mathrm{dist}\left(Ax^k + b \,|\, C\right)\}_{k\in S}$ is bounded, or

(b) $X = \mathbb{R}^n$,

then any cluster point $\bar{x}$ of the subsequence $\{x^k\}_{k\in S}$ satisfies $0 \in \partial J_0(\bar{x}) + N(\bar{x} \,|\, X)$. Moreover, if (a) holds, then $(x^{k+1} - x^k) \overset{S}{\to} 0$.

Proof. Let the sequences $\{q^k\}$, $\{r^k\}$ and $\{W_k\}$ be defined as in Lemma 2.4.1, and let $\bar{x}$ be a cluster point of the subsequence $\{x^k\}_{k\in S}$. Let $\hat{S} \subset S$ be a subsequence such that $x^k \overset{\hat{S}}{\to} \bar{x}$. Without loss of generality, due to the upper semi-continuity of the normal cone operator [59, Proposition 6.6], the continuity of the projection operator and Lemma 2.4.1(4), we can assume that for each $i \in \mathcal{A}(A\bar{x} + b)$ there exists

$$\bar{u}_i \in \mathbb{B}_2 \cap N(A_i\bar{x} + b_i \,|\, C_i) \quad \text{such that} \quad w_i^k r_i^k \overset{\hat{S}}{\to} \bar{u}_i. \tag{2.4.1}$$

Also due to the continuity of the projection operator, for each $i \notin I(A\bar{x} + b)$ we have

$$w_i^k r_i^k \overset{\hat{S}}{\to} \frac{(I - P_{C_i})(A_i\bar{x} + b_i)}{\|(I - P_{C_i})(A_i\bar{x} + b_i)\|_2}. \tag{2.4.2}$$

Let us first suppose that (b) holds, i.e., that $X = \mathbb{R}^n$ so that $N(x \,|\, X) = \{0\}$ for all $x \in \mathbb{R}^n$. By (2.4.1)-(2.4.2), Lemma 2.4.1 Parts (3) and (5), and (2.3.9), we have

$$\begin{aligned} 0 \in \nabla\varphi(\bar{x}) &+ \sum_{i\notin\mathcal{A}(A\bar{x}+b)} A_i^T \frac{(I - P_{C_i})(A_i\bar{x} + b_i)}{\|(I - P_{C_i})(A_i\bar{x} + b_i)\|_2} + \sum_{i\in\mathcal{A}(A\bar{x}+b)} A_i^T(\mathbb{B}_2 \cap N(A_i\bar{x} + b_i \,|\, C_i)) \\ &= \partial J_0(\bar{x}). \end{aligned}$$

Next, suppose that (a) holds, i.e., that $\ker(A) \cap \ker(H) = \{0\}$ and the set $\{\text{dist}\left(Ax^k + b \,|\, C\right)\}_{k \in S}$ is bounded. This latter fact and Lemma 2.4.1(6) implies that $q^k \overset{S}{\to} 0$. We now show that $(x^{k+1} - x^k) \overset{S}{\to} 0$. Indeed, if this were not the case, then there would exist a subsequence $\hat{S} \subset S$ and a vector $\bar{w} \in \mathbb{R}^n$ with $\|\bar{w}\|_2 = 1$ such that $\{\left\|x^{k+1} - x^k\right\|_2\}_{\hat{S}}$ is bounded away from 0 while $\frac{x^{k+1}-x^k}{\left\|x^{k+1}-x^k\right\|_2} \overset{\hat{S}}{\to} \bar{w}$. But then $q^k / \left\|x^{k+1} - x^k\right\|_2 \overset{\hat{S}}{\to} 0$ while $q^k / \left\|x^{k+1} - x^k\right\|_2 = \tilde{A}\frac{x^{k+1}-x^k}{\left\|x^{k+1}-x^k\right\|_2} \overset{\hat{S}}{\to} \tilde{A}\bar{w}$, where $\tilde{A}$ is defined in (2.3.5). But then $0 \neq \bar{w} \in \ker(H) \cap \ker(A) = \ker(\tilde{A})$, a contradiction. Hence, $(x^{k+1} - x^k) \overset{S}{\to} 0$, and so $x^{k+1} = x^k + (x^{k+1} - x^k) \overset{S}{\to} \bar{x}$. In particular, this and the upper semi-continuity of the normal cone operator imply that $\limsup_{k \in S} N\left(x^{k+1} \,|\, X\right) \subset N\left(\bar{x} \,|\, X\right)$. Hence, by (2.4.1)–(2.4.2), Lemma 2.4.1 Parts (3) and (5), and (2.3.9), we have

$$\begin{aligned} 0 \in \nabla\varphi(\bar{x}) &+ \sum_{i \notin I(A\bar{x}+b)} A_i^T \frac{(I - P_{C_i})(A_i\bar{x} + b_i)}{\|(I - P_{C_i})(A_i\bar{x} + b_i)\|_2} + \sum_{i \in I(A\bar{x}+b)} A_i^T(\mathbb{B}_2 \cap N\left(A_i\bar{x} + b_i \,|\, C_i\right)) \\ &+ N\left(\bar{x} \,|\, X\right) \\ &= \partial J_0(\bar{x}) + N\left(\bar{x} \,|\, X\right), \end{aligned}$$

as desired. □

The previously stated Corollary 2.3.10 provides conditions under which the sequence $\{x^k\}$ has cluster points. One of these conditions is that H is positive definite. In such cases, the function J_0 is strongly convex and so the problem (2.1.4) has a unique global solution x^*, meaning that the entire sequence converges to x^*. We formalize this conclusion with the following theorem.

Theorem 2.4.3. *Suppose that H is positive definite and the sequence $\{(x^k, \epsilon^k)\}$ is generated by IRWA with initial point $x^0 \in X$ and relaxation vector $\epsilon^0 \in \mathbb{R}^l_{++}$. Then, the problem* (2.1.4) *has a unique global solution x^* and $x^k \to x^*$.*

Proof. Since H is positive definite, the function $J(x, \epsilon)$ is strongly convex in x for all $\epsilon \in \mathbb{R}^l_+$. In particular, J_0 is strongly convex and so (2.1.4) has a unique global solution x^*. By Corollary 2.3.10, the set $L(J(x^0, \epsilon^0), \epsilon^0)$ is compact, and, by Lemma 2.3.6, the sequence $J(x^k, \epsilon^k)$ is decreasing; hence, $\{x^k\} \subset L(J(x^0, \epsilon^0), \epsilon^0)$. Therefore, the set $\{\text{dist}\left(Ax^k + b \,|\, C\right)\}_{k \in S}$ is bounded and $\ker(H) \cap \ker(A) \subset \ker(H) = \{0\}$, and so, by Theorem 2.4.2, the subsequence

$\{x^k\}_{k\in S}$ has a cluster point $\bar{x}$ satisfying $0 \in \partial J_0(\bar{x}) + N(\bar{x} \,|\, X)$. But the only such point is $\bar{x} = x^*$, and hence $x^k \overset{S}{\to} x^*$.

Since the sequence $\{J(x^k, \epsilon^k)\}$ is monotonically decreasing and bounded below by Corollary 2.3.10, it has a limit $\tilde{J}$. Since $x^k \overset{S}{\to} x^*$, we have $\tilde{J} = \min_{x\in X} J_0(x)$. Let $\tilde{S}$ be any subsequence of $\mathbb{N}$. Since $\{x^k\}_{k\in\tilde{S}} \subset L(J(x^0, \epsilon^0), \epsilon^0)$ (which is compact by Corollary 2.3.10(2)), this subsequence has a further subsequence $\tilde{S}_0 \subset \tilde{S}$ such that $x^k \overset{\tilde{S}_0}{\to} \bar{x}$ for some $\bar{x} \in X$. For this subsequence, $J(x^k, \epsilon^0) \overset{\tilde{S}_0}{\to} \tilde{J}$, and, by continuity, $J(x^k, \epsilon^0) \overset{\tilde{S}_0}{\to} J(\bar{x}, 0) = J_0(\bar{x})$. Hence, $\bar{x} = x^*$ by uniqueness. Therefore, since every subsequence of $\{x^k\}$ has a further subsequence that converges to x^*, it must be the case that the entire sequence converges to x^*. □

2.5 Complexity of IRWA

A point $\tilde{x} \in X$ is an ε-optimal solution to (2.1.4) if

$$J_0(\tilde{x}) \leq \inf_{x\in X} J_0(x) + \varepsilon. \tag{2.5.1}$$

In this section, we prove the following result.

Theorem 2.5.1. *Consider the problem* (2.1.4) *with* $X = \mathbb{R}^n$ *and* H *positive definite. Let* $\varepsilon > 0$ *and* $\epsilon \in \mathbb{R}^l_{++}$ *be such that*

$$\|\epsilon\|_1 \leq \varepsilon/2 \quad \textit{and} \quad \varepsilon \leq 4l\tilde{\varepsilon}, \tag{2.5.2}$$

where $\tilde{\varepsilon} := \min_{i\in\mathcal{I}} \epsilon_i$. *Suppose that the sequence* $\{(x^k, \epsilon^k)\}$ *is generated by IRWA with initial point* $x^0 \in \mathbb{R}^n$ *and relaxation vector* $\epsilon^0 = \epsilon \in \mathbb{R}^l_{++}$, *and that the relaxation vector is kept fixed so that* $\epsilon^k = \epsilon$ *for all* $k \in \mathbb{N}$. *Then, in at most* $O(1/\varepsilon^2)$ *iterations,* x^k *is an* ε*-optimal solution to* (2.1.4), *i.e.,* (2.5.1) *holds with* $\tilde{x} = x^k$.

The proof of this result requires a few preliminary lemmas. For ease of presentation, we assume that the hypotheses of Theorem 2.5.1 hold throughout this section. Thus, in particular, Corollary 2.3.10 and the strict convexity and coercivity of J tells us that there exists $\tau > 0$ such that

$$\left\| x^k - x^\epsilon \right\|_2 \leq \tau \quad \text{for all } k \in \mathbb{N}, \tag{2.5.3}$$

where x^ϵ is the solution to $\min_{x\in\mathbb{R}^n} J(x,\epsilon)$. Let w_i for $i \in \mathcal{I}$ and $\tilde{A}$ be given as in (2.3.4) and (2.3.5), respectively. In addition, define

$$R_i(r_i) := \frac{r_i}{\sqrt{\|r_i\|_2^2 + \epsilon_i^2}}, \quad r_i(x) := (I - P_{C_i})(A_i x + b_i) \text{ for } i \in \mathcal{I}$$
$$\text{and} \quad u(x,\epsilon) := \nabla\varphi(x) + \sum_{i\in\mathcal{I}} w_i(x,\epsilon) A_i^T r_i(x).$$

Recall that

$$\begin{aligned}\partial_x J(x,\epsilon) = \nabla\varphi(x) &+ \sum_{i\notin\mathcal{A}(Ax+b)} w_i(x,\epsilon) A_i^T r_i(x) \\ &+ \sum_{i\in\mathcal{A}(Ax+b)} w_i(x,\epsilon) A_i^T (\mathbb{B}_2 \cap N(A_i x + b_i | C_i)),\end{aligned}$$

so that $u(x,\epsilon) \in \partial_x J(x,\epsilon)$. It is straightforward to show that, for each $i \in \mathcal{I}$, we have

$$\nabla_{r_i} R_i(r_i) = \frac{1}{\sqrt{\|r_i\|_2^2 + \epsilon_i^2}} \left(I - \frac{r_i r_i^T}{\|r_i\|_2^2 + \epsilon_i^2} \right)$$

so that

$$\|\nabla_{r_i} R_i(r_i)\|_2 \leq 1/\epsilon_i \quad \forall\, r_i. \tag{2.5.4}$$

Consequently, for each $i \in \mathcal{I}$, the function R_i is globally Lipschitz continuous with Lipschitz constant $1/\epsilon_i$. This allows us to establish a similar result for the mapping $u(x,\epsilon)$ as a function of x, which we prove as our next result. For convenience, we use

$$\bar{u} := u(\bar{x},\epsilon),\ \hat{u} := u(\hat{x},\epsilon),\ \text{and } u^k := u(x^k,\epsilon),$$

and similar shorthand for $w_i(x,\epsilon_i)$, $W(x,\epsilon)$, and $r_i(x)$.

Lemma 2.5.2. *Let the hypotheses of Theorem 2.5.1 hold. Moreover, let λ be the largest eigenvalue of H and σ_1 be an upper bound on all singular values of the matrices A_i for $i \in \mathcal{I}$. Then, as a function of x, the mapping $u(x,\epsilon)$ is globally Lipschitz continuous with Lipschitz constant $\beta := \lambda + l\sigma_1^2/\tilde{\varepsilon}$.*

Proof. By Theorem 2.3.2, for all $\{\bar{x},\hat{x}\} \subset \mathbb{R}^n$, we have

$$\|\bar{r}_i - \hat{r}_i\|_2 \leq \|A_i(\bar{x} - \hat{x})\|_2 \leq \sigma_1 \|\bar{x} - \hat{x}\|_2 . \tag{2.5.5}$$

Therefore,

$$\begin{aligned}
\|\bar{u} - \hat{u}\|_2 &= \left\| H(\bar{x} - \hat{x}) + \sum_{i \in \mathcal{I}} A_i^T (R_i(\bar{r}_i) - R_i(\hat{r}_i)) \right\|_2 \\
&\leq \|H\|_2 \|\bar{x} - \hat{x}\|_2 + \tfrac{1}{\tilde{\epsilon}} \sum_{i \in \mathcal{I}} \|A_i\|_2 \|\bar{r}_i - \hat{r}_i\|_2 \\
&\leq \|H\|_2 \|\bar{x} - \hat{x}\|_2 + \tfrac{1}{\tilde{\epsilon}} \sum_{i \in \mathcal{I}} \|A_i\|_2^2 \|\bar{x} - \hat{x}\|_2 \\
&\leq (\lambda + l\sigma_1^2/\tilde{\varepsilon}) \|\bar{x} - \hat{x}\|_2 ,
\end{aligned}$$

where the first inequality follows from (2.5.4), the second from (2.5.5), and the last from the fact that the 2-norm of a matrix equals its largest singular value. □

By Lemma 2.5.2 and the subgradient inequality, we obtain the bound

$$0 \leq J(\bar{x}, \epsilon) - J(\hat{x}, \epsilon) - \langle \hat{u},\, \bar{x} - \hat{x} \rangle \leq \langle \bar{u} - \hat{u},\, \bar{x} - \hat{x} \rangle \leq \beta \|\bar{x} - \hat{x}\|_2^2 . \tag{2.5.6}$$

Moreover, by Part (5) of Lemma 2.4.1, we have

$$-\tilde{A}^T W_k q^k = -\tilde{A}^T W_k \tilde{A}(x^{k+1} - x^k) = u^k \in \partial_x J(x^k, \epsilon).$$

If we now define $D_k := \tilde{A}^T W_k \tilde{A}$, then $x^k - x^{k+1} = D_k^{-1} u^k$ and

$$(q^k)^T W_k q^k = (x^k - x^{k+1})^T D_k (x^k - x^{k+1}) = (u^k)^T D_k^{-1} u^k. \tag{2.5.7}$$

This gives the following bound on the decrease in J when going from x^k to x^{k+1}.

Lemma 2.5.3. *Let the hypotheses of Lemma 2.5.2 hold. Then,*

$$J(x^{k+1}, \epsilon) - J(x^k, \epsilon) \leq -\alpha \|u^k\|_2^2,$$

where $\alpha := \tilde{\varepsilon}/(2\sigma_0^2)$ with σ_0 the largest singular value of $\tilde{A}$.

Proof. By Lemma 2.3.6 and (2.5.7), we have

$$J(x^{k+1}, \epsilon) - J(x^k, \epsilon) \leq -\frac{1}{2}(q^k)^T W_k q^k = -\frac{1}{2}(u^k)^T D_k^{-1} u^k.$$

Since the $\|D_k\|_2 \leq \|W_k^{1/2}\|_2^2 \|\tilde{A}\|_2^2$, we have that the largest eigenvalue of D_k is bounded above by $\sigma_0^2/\tilde{\varepsilon}$. This implies $\frac{1}{2}(u^k)^T D_k^{-1} u^k \geq \alpha \left\|u^k\right\|_2^2$, which gives the result. □

The following theorem is the main tool for proving Theorem 2.5.1.

Theorem 2.5.4. *Let the hypotheses of Lemma 2.5.3 hold, and, as in* (2.5.3)*, let* x^ϵ *be the solution to* $\min_{x\in\mathbb{R}^n} J(x,\epsilon)$*. Then,*

$$J(x^k,\epsilon) - J(x^\epsilon,\epsilon) \le \frac{32l^2\sigma_0^2\tau^2}{k\varepsilon}\left[\frac{\|u^\epsilon\|_2\,\varepsilon + \tau(\lambda\varepsilon + l\sigma_1^2)}{\|u^\epsilon\|_2\,\varepsilon + \tau(\lambda\varepsilon + 4l^2\sigma_1^2) + 8l\tau\sigma_0^2/k}\right]. \tag{2.5.8}$$

Therefore, IRWA requires $O(1/\varepsilon^2)$ *iterations to reach* ε*-optimality for* $J(x,\epsilon)$*, i.e.,*

$$J(x^k,\epsilon) - J(x^\epsilon,\epsilon) \le \varepsilon.$$

Proof. Set $\delta^j := J(x^j,\epsilon) - J(x^\epsilon,\epsilon)$ for all $j \in \mathbb{N}$. Then, by Lemma 2.5.3,

$$\begin{aligned} 0 \le \delta^{j+1} &= J(x^{j+1},\epsilon) - J(x^\epsilon,\epsilon) \\ &\le J(x^j,\epsilon) - J(x^\epsilon,\epsilon) - \alpha\|u^j\|_2^2 = \delta^j - \alpha\|u^j\|_2^2 \le \delta^j. \end{aligned} \tag{2.5.9}$$

If for some $j < k$ we have $\delta^j = 0$, then (2.5.9) implies that $\delta^k = 0$ and $u^k = 0$, which in turn implies that $x^{k+1} = x^\epsilon$ and the bound (2.5.8) holds trivially. In the remainder of the proof, we only consider the nontrivial case where $\delta^j > 0$ for $j = 0, ..., k-1$.

Consider $j \in \{0,\dots,k-1\}$. By the convexity of J and (2.5.3), we have

$$\delta^j = J(x^j,\epsilon) - J(x^\epsilon,\epsilon) \le (u^j)^T(x^j - x^\epsilon) \le \|u^j\|_2\|x^j - x^\epsilon\| \le \tau\|u^j\|_2.$$

Combining this with (2.5.9), gives

$$\delta^{j+1} \le \delta^j - \frac{\alpha}{\tau^2}(\delta^j)^2.$$

Dividing both sides by $\delta^{j+1}\delta^j$ and noting that $\frac{\delta^j}{\delta^{j+1}} \ge 1$ yields

$$\frac{1}{\delta^{j+1}} - \frac{1}{\delta^j} \ge \frac{\alpha}{\tau^2}\frac{\delta^j}{\delta^{j+1}} \ge \frac{\alpha}{\tau^2}. \tag{2.5.10}$$

Summing both sides of (2.5.10) from 0 to $k-1$, we obtain

$$\frac{1}{\delta^k} \ge \frac{\alpha k}{\tau^2} + \frac{1}{\delta^0} = \frac{\alpha\delta^0 k + \tau^2}{\delta^0\tau^2}, \tag{2.5.11}$$

or, equivalently,

$$\delta^k \le \frac{\delta^0\tau^2}{\alpha\delta^0 k + \tau^2}. \tag{2.5.12}$$

The inequality (2.5.6) implies that

$$\delta^0 = J(x^0, \epsilon) - J(x^\epsilon, \epsilon) \leq (u^\epsilon)^T (x^0 - x^\epsilon) + \beta \|x^0 - x^\epsilon\|_2^2 \leq \tau(\|u^\epsilon\|_2 + \beta\tau),$$

which, together with (2.5.11), implies that

$$\frac{\alpha\delta^0 k + \tau^2}{\delta^0 \tau^2} \geq \frac{\alpha k}{\tau^2} + \frac{1}{\tau(\|u^\epsilon\|_2 + \beta\tau)}.$$

Rearranging, one has

$$\frac{\tau^2 \delta^0}{\alpha k \delta^0 + \tau^2} \leq \frac{\tau^2(\|u^\epsilon\|_2 + \beta\tau)}{\alpha k(\|u^\epsilon\|_2 + \beta\tau) + \tau}.$$

Substituting in $\beta = \lambda + l\sigma_1^2/\tilde{\varepsilon}$ and $\alpha = \tilde{\varepsilon}/(2\sigma_0^2)$ defined in Lemmas 2.5.2 and 2.5.3, respectively, and then combining with (2.5.12) gives

$$\delta^k \leq \frac{\tau^2(\|u^\epsilon\|_2 + \tau(\lambda + l\sigma_1^2/\tilde{\varepsilon}))}{(\tilde{\varepsilon}/(2\sigma_0^2))k(\|u^\epsilon\|_2 + \tau(\lambda + l\sigma_1^2/\tilde{\varepsilon})) + \tau} = \frac{2\sigma_0^2\tau^2}{k\tilde{\varepsilon}} \left[\frac{\|u^\epsilon\|_2 \tilde{\varepsilon} + \tau(\lambda\tilde{\varepsilon} + l\sigma_1^2)}{\|u^\epsilon\|_2 \tilde{\varepsilon} + \tau(\lambda\tilde{\varepsilon} + l\sigma_1^2) + 2\tau\sigma_0^2/k}\right].$$

Finally, using the inequalities $\varepsilon \leq 4l\tilde{\varepsilon}$ and $\tilde{\varepsilon} \leq \varepsilon$ (recall (2.5.2)) gives

$$\delta^k \leq \frac{32l^2\sigma_0^2\tau^2}{k\varepsilon} \left[\frac{\|u^\epsilon\|_2 \varepsilon + \tau(\lambda\varepsilon + l\sigma_1^2)}{\|u^\epsilon\|_2 \varepsilon + \tau(\lambda\varepsilon + 4l^2\sigma_1^2) + 8l\tau\sigma_0^2/k}\right],$$

which is the desired inequality. □

We can now prove Theorem 2.5.1.

Theorem 2.5.1. Let $x^* = \arg\min_{x \in \mathbb{R}^n} J_0(x)$. Then, by convexity in ϵ,

$$\begin{aligned} J(x^\epsilon, \epsilon) - J(x^*, 0) &\leq [\partial_\epsilon J(x^\epsilon, \epsilon)]^T (\epsilon - 0) \\ &= \sum_{i \in \mathcal{I}} \frac{\epsilon_i^2}{\sqrt{\|r_i(x^\epsilon)\|_2^2 + \epsilon_i^2}} \leq \sum_{i \in \mathcal{I}} \epsilon_i = \|\epsilon\|_1 \leq \varepsilon/2. \end{aligned}$$

By Theorem 2.5.4, IRWA needs $O(1/\varepsilon^2)$ iterations to reach

$$J(x^k, \epsilon) - J(x^\epsilon, \epsilon) \leq \varepsilon/2.$$

Combining these two inequalities yields the result. □

Chapter 3

ALTERNATING DIRECTION AUGMENTED LAGRANGIAN ALGORITHM (ADAL)

3.1 Introduction

For comparison with IRWA, we now describe an alternating direction augmented Lagrangian method for solving problem (2.1.4). This approach, like IRWA, can be solved by matrix-free methods. Defining

$$\hat{J}(x,p) := \varphi(x) + \operatorname{dist}(p \mid C),$$

where $\operatorname{dist}(p \mid C)$ is defined as in (2.1.3), the problem (2.1.4) has the equivalent form

$$\min_{x\in X,p} \ \hat{J}(x,p) \text{ subject to } Ax + b = p, \tag{3.1.1}$$

where $p := (p_1^T, \ldots, p_l^T)^T$. In particular, note that $J_0(x) = \hat{J}(x, Ax + b)$. Defining dual variables $(u_1, \ldots, u_l)$, a partial Lagrangian for (3.1.1) is given by

$$L(x,p,u) := \hat{J}(x,p) + \langle u,\, Ax + b - p\rangle + \delta(x \mid X),$$

and the corresponding augmented Lagrangian, with penalty parameter $\mu > 0$, is

$$L(x,p,u,\mu) := \hat{J}(x,p) + \tfrac{1}{2\mu}\,\|Ax + b - p + \mu u\|_2^2 - \tfrac{\mu}{2}\|u\|_2^2 + \delta(x \mid X).$$

(Observe that due to their differing numbers of inputs, the Lagrangian value $L(x,p,u)$ and augmented Lagrangian value $L(x,p,u,\mu)$ should not be confused with each other, nor with the level set value $L(\alpha,\epsilon)$ defined in Theorem 2.3.9.)

We now state the algorithm.

Alternating Direction Augmented Lagrangian Algorithm (ADAL)

Step 0: (Initialization) Choose an initial point $x^0 \in X$, dual vectors $u_i^0 \in \mathbb{R}^{m_i}$ for $i \in \mathcal{I}$, and penalty parameter $\mu > 0$. Let $\sigma \geq 0$ and $\sigma'' \geq 0$ be two scalars which serve as termination tolerances for the stepsize and constraint residual, respectively. Set $k := 0$.

Step 1: (Solve the augmented Lagrangian subproblems for (x^{k+1}, p^{k+1}))

Compute a solution p^{k+1} to the problem

$$\mathcal{L}_p(x^k, p, u^k, \mu): \quad \min_p L(x^k, p, u^k, \mu),$$

and a solution x^{k+1} to the problem

$$\mathcal{L}_x(x, p^{k+1}, u^k, \mu): \quad \min_x L(x, p^{k+1}, u^k, \mu).$$

Step 2: (Set the new multipliers u^{k+1})

Set

$$u^{k+1} := u^k + \tfrac{1}{\mu}(Ax^{k+1} + b - p^{k+1}).$$

Step 3: (Check stopping criteria)

If $\left\|x^{k+1} - x^k\right\|_2 \leq \sigma$ and $\left\|Ax^{k+1} + b - p^{k+1}\right\|_* \leq \sigma''$, then stop; else, set $k := k + 1$ and go to Step 1.

Remark 3.1.1. As for IRWA, one can also replace the stopping criteria of Step 3 with a criterion based on a percent reduction in duality gap; recall Remark 2.3.5.

3.2 Properties of the ADAL Subproblems

Before addressing the convergence properties of the ADAL algorithm, we discuss properties of the solutions to the subproblems $\mathcal{L}_p(x, p, u, \mu)$ and $\mathcal{L}_x(x, p, u, \mu)$.

The subproblem $\mathcal{L}_p(x^k, p, u^k, \mu)$ is separable. Defining

$$s_i^k := A_i x^k + b_i + \mu u_i^k \quad \forall\, i \in \mathcal{I},$$

the solution of $\mathcal{L}_p(x^k, p, u^k, \mu)$ can be written explicitly, for each $i \in \mathcal{I}$, as

$$p_i^{k+1} := \begin{cases} P_{C_i}(s_i^k) & \text{if } \operatorname{dist}_2\left(s_i^k \mid C_i\right) \leq \mu \\ s_i^k - \frac{\mu}{\operatorname{dist}_2\left(s_i^k \mid C_i\right)}(s_i^k - P_{C_i}(s_i^k)) & \text{if } \operatorname{dist}_2\left(s_i^k \mid C_i\right) > \mu. \end{cases} \tag{3.2.1}$$

Subproblem $\mathcal{L}_x(x, p^{k+1}, u^k, \mu)$, on the other hand, involves the minimization of a convex quadratic over X, which can be solved by matrix-free methods.

Along with the dual variable estimates $\{u_i^k\}$, we define the auxiliary estimates

$$\hat{u}^{k+1} := u^{k+1} - \tfrac{1}{\mu} q^k, \quad \text{where} \quad q^k := A(x^{k+1} - x^k) \quad \text{as in IRWA Step 2.}$$

First-order optimality conditions for (3.1.1) are then given by

$$0 \in \partial \mathrm{dist}\,(p \,|\, C) - u, \tag{3.2.2a}$$

$$0 \in \nabla\varphi(x) + A^T u + N(x|X), \tag{3.2.2b}$$

$$0 = Ax + b - p, \tag{3.2.2c}$$

or, equivalently,

$$0 \in \partial J_0(x) = \nabla\varphi(x) + A^T \partial \mathrm{dist}\,(\cdot \,|\, C)\,(Ax + b) + N\,(x \,|\, X)\,.$$

The next lemma relates the iterates to these optimality conditions.

Lemma 3.2.1. *Suppose that the sequence $\{(x^k, p^k, u^k)\}$ is generated by ADAL with initial point $x^0 \in X$. Then, for all $k \in \mathbb{N}$, we have*

$$\hat{u}^{k+1} \in \partial \mathrm{dist}\left(p^{k+1} \,|\, C\right) \ \text{and} \ -\tfrac{1}{\mu} A^T q^k \in \nabla\varphi(x^{k+1}) + A^T \hat{u}^{k+1} + N\left(x^{k+1} \,|\, X\right). \tag{3.2.3}$$

Therefore,

$$-\tfrac{1}{\mu} A^T q^k \in \nabla\varphi(x^{k+1}) + A^T \partial \mathrm{dist}\left(p^{k+1} \,|\, C\right) + N\left(x^{k+1} \,|\, X\right).$$

Moreover, for all $k \geq 1$, we have

$$\left\|\hat{u}^k\right\|_* \leq 1, \quad \left\|s^k\right\|_* \leq \mu, \quad \text{and} \quad \left\|p^k\right\|_* \leq \hat{\mu}, \tag{3.2.4}$$

where $\hat{\mu} := \max\{\mu, \sup_{\|s\|_* \leq \mu} \|P_C(s)\|_*\} < \infty$.

Proof. By ADAL Step 1, the auxiliary variable p^{k+1} satisfies

$$0 \in \partial \mathrm{dist}\left(p^{k+1} \,|\, C\right) - u^k - \tfrac{1}{\mu}(Ax^k + b - p^{k+1}),$$

which, along with ADAL Step 2, implies that

$$\begin{aligned} u^{k+1} &\in \partial \mathrm{dist}\left(p^{k+1} \,|\, C\right) + \tfrac{1}{\mu}(Ax^{k+1} + b - p^{k+1}) - \frac{1}{\mu}(Ax^k + b - p^{k+1}) \\ &= \partial \mathrm{dist}\left(p^{k+1} \,|\, C\right) + \tfrac{1}{\mu} q^k. \end{aligned}$$

Hence, the first part of (3.2.3) holds. Then, again by ADAL Step 1, x^{k+1} satisfies

$$0 \in \nabla\varphi(x^{k+1}) + \frac{1}{\mu}A^T(Ax^{k+1} + b - p^{k+1} + \mu u^k) + N(x^{k+1}|X).$$

which, along with ADAL Step 2, implies that

$$0 \in \nabla\varphi(x^{k+1}) + A^T u^{k+1} + N(x^{k+1}|X). \tag{3.2.5}$$

Hence, the second part of (3.2.3) holds.

The first bound in (3.2.4) follows from the first part of (3.2.3). The second bound in (3.2.4) follows from the first bound and the fact that for $k \in \mathbb{N}$ we have

$$s^k = Ax^k + b + \mu u^k = \mu u^{k+1} - q^k = \mu\hat{u}^{k+1}.$$

As for the third bound, note that if, for some $i \in \mathcal{I}$, we have $\text{dist}_2\left(s_i^{k-1} \mid C_i\right) \leq \mu$, then, by (3.2.1), we have $\left\|p_i^k\right\|_2 \leq \hat{\mu}$; on the other hand, if $\text{dist}_2\left(s_i^{k-1} \mid C_i\right) > \mu$ so that $0 < \xi := \mu/\text{dist}_2\left(s_i^{k-1} \mid C_i\right) < 1$, then, by (3.2.1) and the second bound in (3.2.4),

$$\left\|p_i^k\right\|_2 \leq (1-\xi)\left\|s_i^{k-1}\right\|_2 + \xi\hat{\mu} \leq \hat{\mu}.$$

Consequently, $\left\|p^k\right\|_* = \sup_{i\in\mathcal{I}}\left\|p_i^k\right\|_2 \leq \hat{\mu}$. □

For the remainder of our discussion of ADAL, we define the residuals

$$z^{k+1} := Ax^{k+1} + b - p^{k+1}.$$

Lemma 3.2.1 tells us that the deviation of $(p^{k+1}, \hat{u}^{k+1})$ from satisfying the first-order stationary conditions for (3.2.2) can be measured by

$$E^{k+1} = \max\{\left\|q^k\right\|, \left\|z^{k+1}\right\|_*\}. \tag{3.2.6}$$

3.3 Convergence of ADAL

In this section, we establish the global convergence properties of the ADAL algorithm. The proofs in this section follow a standard pattern for algorithms of this type (e.g., see [4]). We make use of the following standard assumption.

Assumption 3.3.1. *There exists a point* (x^*, p^*, u^*) *satisfying* (3.2.2).

Since (3.1.1) is convex, this assumption is equivalent to the existence of an optimal solution. Moreover, the optimality conditions imply that (x^*, p^*) is a minimizer of the convex function $L(x, p, u^*)$ over X. We begin our analysis by providing bounds on the optimal primal objective value.

Lemma 3.3.2. *Suppose that the sequence* $\{(x^k, p^k, u^k)\}$ *is generated by ADAL with initial point* $x^0 \in X$. *Then, under Assumption 3.3.1, we have for all* $k \in \mathbb{N}$ *that*

$$(u^*)^T z^{k+1} \geq \hat{J}(x^*, p^*) - \hat{J}(x^{k+1}, p^{k+1}) \geq (u^{k+1})^T z^{k+1} - \frac{1}{\mu}(q^k)^T(p^* - p^{k+1}). \tag{3.3.1}$$

Proof. Since (x^*, p^*, u^*) is a saddle point of L, it follows that $Ax^* + b - p^* = 0$, which implies by the fact that $x^{k+1} \in X$ that

$$\hat{J}(x^*, p^*) = L(x^*, p^*, u^*) \leq L(x^{k+1}, p^{k+1}, u^*).$$

Rearranging, we obtain the first inequality in (3.3.1).

We now show the second inequality in (3.3.1). Recall that Steps 1 and 2 of ADAL tell us that (3.2.5) holds for all $k \in \mathbb{N}$. Therefore, by convexity, x^{k+1} is an optimal solution to

$$\min_{x \in X} \ \varphi(x) + (u^{k+1})^T Ax.$$

Since this is a convex problem and $x^* \in X$, we have

$$\varphi(x^*) + (u^{k+1})^T Ax^* \geq \varphi(x^{k+1}) + (u^{k+1})^T Ax^{k+1}. \tag{3.3.2}$$

Similarly, by the first expression in (3.2.3) and convexity, p^{k+1} is an optimal solution to

$$\min_{p} \ \text{dist}\,(p \,|\, C\,) - (\hat{u}^{k+1})^T p.$$

Hence, by the convexity of this problem, we have

$$\text{dist}\,(p^* \,|\, C\,) - (\hat{u}^{k+1})^T p^* \geq \text{dist}\left(p^{k+1} \,|\, C\right) - (\hat{u}^{k+1})^T p^{k+1}. \tag{3.3.3}$$

By adding (3.3.2) and (3.3.3), we obtain

$$\begin{aligned}
&\hat{J}(x^*, p^*) - \hat{J}(x^{k+1}, p^{k+1}) \\
&\geq (\hat{u}^{k+1})^T(p^* - p^{k+1}) + (u^{k+1})^T A(x^{k+1} - x^*) \\
&= (u^{k+1})^T(p^* - p^{k+1}) - \tfrac{1}{\mu}(q^k)^T(p^* - p^{k+1}) + (u^{k+1})^T A(x^{k+1} - x^*) \\
&= (u^{k+1})^T \Big((p^* - Ax^*) - b) - (p^{k+1} - Ax^{k+1} - b)\Big) - \tfrac{1}{\mu}(q^k)^T(p^* - p^{k+1}) \\
&= (u^{k+1})^T z^{k+1} - \tfrac{1}{\mu}(q^k)^T(p^* - p^{k+1}),
\end{aligned}$$

which completes the proof. □

Consider the measure of distance to (x^*, u^*) defined by

$$\omega^k := \tfrac{1}{\mu}\left\|A(x^k - x^*)\right\|_2^2 + \mu\left\|u^k - u^*\right\|_2^2.$$

In our next lemma, we show that this measure decreases monotonically.

Lemma 3.3.3. *Suppose that the sequence $\{(x^k, p^k, u^k)\}$ is generated by ADAL with initial point $x^0 \in X$. Then, under Assumption 3.3.1, we have for all $k \geq 1$ that*

$$\tfrac{1}{\mu}(\left\|z^{k+1}\right\|_2^2 + \left\|q^k\right\|_2^2) + 2(x^{k+1} - x^k)^T H(x^{k+1} - x^k) \leq \omega^k - \omega^{k+1}. \tag{3.3.4}$$

Proof. By using the extremes of the inequality (3.3.1) and rearranging, we obtain

$$(u^{k+1} - u^*)^T z^{k+1} - \tfrac{1}{\mu}(q^k)^T(p^* - p^{k+1}) \leq 0.$$

Since (x^*, p^*, u^*) is a saddle point of L, and so $Ax^* + b = p^*$, this implies

$$(u^{k+1} - u^*)^T z^{k+1} - \tfrac{1}{\mu}(q^k)^T z^{k+1} + \tfrac{1}{\mu}(x^{k+1} - x^k)^T A^T A(x^{k+1} - x^*) \leq 0. \tag{3.3.5}$$

The update in Step 2 yields $u^{k+1} = u^k + \frac{1}{\mu}z^{k+1}$, so we have

$$(u^{k+1} - u^*)^T z^{k+1} = \left[(u_i^k - u^*)^T z^{k+1} + \tfrac{1}{2\mu}\left\|z^{k+1}\right\|_2^2\right] + \tfrac{1}{2\mu}\left\|z^{k+1}\right\|_2^2. \tag{3.3.6}$$

Let us now consider the first grouped term in (3.3.6). From ADAL Step 2, we have $z^{k+1} = \mu(u^{k+1} - u^k)$, which gives

$$\begin{aligned}
(u^k - u^*)^T z^{k+1} + \tfrac{1}{2\mu}\left\|z^{k+1}\right\|_2^2 &= \mu(u^k - u^*)^T(u^{k+1} - u^k) + \tfrac{\mu}{2}\|u^{k+1} - u^k\|_2^2 \\
&= \mu(u^k - u^*)^T(u^{k+1} - u^*) - \mu(u^k - u^*)^T(u^k - u^*) \\
&\quad + \tfrac{\mu}{2}\|(u^{k+1} - u^*) - (u^k - u^*)\|_2^2 \\
&= \tfrac{\mu}{2}(\|u^{k+1} - u^*\|_2^2 - \|u^k - u^*\|_2^2).
\end{aligned} \tag{3.3.7}$$

Adding the final term $\frac{1}{2\mu}\left\|z^{k+1}\right\|_2^2$ in (3.3.6) to the second and third terms in (3.3.5),

$$\begin{aligned}
&\tfrac{1}{\mu}\left(\tfrac{1}{2}\left\|z^{k+1}\right\|_2^2 - (q^k)^T z^{k+1} + (x^{k+1}-x^k)^T A^T A(x^{k+1}-x^*)\right)\\
&= \tfrac{1}{\mu}\left(\tfrac{1}{2}\left\|z^{k+1}\right\|_2^2 - (q^k)^T z^{k+1} + (x^{k+1}-x^k)^T A^T A((x^{k+1}-x^k)+(x^k-x^*))\right)\\
&= \tfrac{1}{\mu}\left(\tfrac{1}{2}\left\|z^{k+1}-q^k\right\|_2^2 + \tfrac{1}{2}\|q^k\|_2^2 + (x^{k+1}-x^k)^T A^T A(x^k-x^*)\right)\\
&= \tfrac{1}{\mu}(\tfrac{1}{2}\left\|z^{k+1}-q^k\right\|_2^2 + \tfrac{1}{2}\|A((x^{k+1}-x^*)-(x^k-x^*))\|_2^2\\
&\quad + ((x^{k+1}-x^*)-(x^k-x^*))^T A^T A(x^k-x^*))\\
&= \tfrac{1}{2\mu}\left(\left\|z^{k+1}-q^k\right\|_2^2 + \|A(x^{k+1}-x^*)\|_2^2 - \|A(x^k-x^*)\|_2^2\right) \qquad (3.3.8)
\end{aligned}$$

From (3.3.6), (3.3.7), and (3.3.8), we have that (3.3.5) reduces to

$$\omega^{k+1}-\omega^k \le -\frac{1}{\mu}\left\|z^{k+1}-q^k\right\|_2^2.$$

Since (3.2.5) holds for $k \ge 1$, we have

$$-(v^{k+1}-v^k) = H(x^{k+1}-x^k) + A^T(u^{k+1}-u^k),$$

for some $v^{k+1} \in N(x^{k+1}|X)$ and $v^k \in N(x^k|X)$. Therefore,

$$\begin{aligned}
(u^{k+1}-u^k)^T q^k &= -(v^{k+1}-v^k)^T(x^{k+1}-x^k) - (x^{k+1}-x^k)^T H(x^{k+1}-x^k)\\
&\le -(x^{k+1}-x^k)^T H(x^{k+1}-x^k), \qquad (3.3.9)
\end{aligned}$$

where the inequality follows since the normal cone operator $N(\cdot\,|\,C)$ is a monotone operator [59]. Using this inequality in the expansion of the right-hand side of (3.3.9) along with the equivalence $z^{k+1} = \mu(u^{k+1}-u^k)$, gives

$$\begin{aligned}
\omega^{k+1}-\omega^k &\le -\tfrac{1}{\mu}\left(\|z^{k+1}\|_2^2 - 2\mu(u^{k+1}-u^k)^T q^k + \|q^k\|_2^2\right)\\
&\le -\tfrac{1}{\mu}(\|z^{k+1}\|_2^2 + \|q^k\|_2^2) + 2(u_i^{k+1}-u_i^k)^T q_i^k\\
&\le -\tfrac{1}{\mu}(\|z^{k+1}\|_2^2 + \|q^k\|_2^2) - 2(x^{k+1}-x^k)^T H(x^{k+1}-x^k),
\end{aligned}$$

as desired. □

We now state and prove our main convergence theorem for ADAL.

Theorem 3.3.4. *Suppose that the sequence $\{(x^k, p^k, u^k)\}$ is generated by ADAL with initial point $x^0 \in X$. Then, under Assumption 3.3.1, we have*

$$\lim_{k\to\infty} q^k = 0, \quad \lim_{k\to\infty} z^{k+1} = 0, \quad \textit{and so} \quad \lim_{k\to\infty} E^k = 0.$$

Moreover, the sequences $\{u^k\}$ and $\{Ax^k\}$ are bounded and

$$\lim_{k\to\infty} \hat{J}(x^k, p^k) = \hat{J}(x^*, p^*) = J_0(x^*).$$

Proof. Summing (3.3.4) over all $k \geq 1$ yields

$$\sum_{k=1}^{\infty} \left(2(x^{k+1} - x^k)^T H(x^{k+1} - x^k) + \tfrac{1}{\mu}(\|z^{k+1}\|_2^2 + \|q^k\|_2^2) \right) \leq \omega^1,$$

which, since $H \succeq 0$, implies that $z^{k+1} \to 0$ and $q^k \to 0$. Consequently, $E^k \to 0$.

The sequence $\{u^k\}$ is bounded since $\hat{u}^{k+1} + (1/\mu)q^k = u^{k+1}$, where $\{\hat{u}^k\}$ is bounded by (3.2.4) and $q^k \to 0$. Similarly, the sequence $\{Ax^k\}$ is bounded since $\mu(u^{k+1} - u^k) + p^{k+1} - b = Ax^{k+1}$, where the sequence $\{p^k\}$ is bounded by (3.2.4). Finally, by (3.3.1), we have that $\hat{J}(x^k, p^k) \to \hat{J}(x^*, p^*)$ since both $z^k \to 0$ and $q^k \to 0$ while $\{p^k\}$ and $\{u^k\}$ are both bounded. □

Corollary 3.3.5. *Suppose that the sequence $\{(x^k, p^k, u^k)\}$ is generated by ADAL with initial point $x^0 \in X$. Then, under Assumption 3.3.1, every cluster point of the sequence $\{x^k\}$ is a solution to* (2.1.4).

Proof. Let $\bar{x}$ be a cluster point of $\{x^k\}$, and let $S \subset \mathbb{N}$ be a subsequence such that $x^k \xrightarrow{S} \bar{x}$. By (3.2.4), $\{p^k\}$ is bounded so we may assume with no loss in generality that there is a $\bar{p}$ such that $p^k \xrightarrow{S} \bar{p}$. Theorem 3.3.4 tells us that $A\bar{x} + b = \bar{p}$ and $\hat{J}(\bar{x}, \bar{p}) = J_0(x^*)$ so that $J_0(\bar{x}) = \hat{J}(\bar{x}, A\bar{x} + b) = \hat{J}(\bar{x}, \bar{p}) = J_0(x^*)$. □

We now address the question of when the sequence $\{x^k\}$ has cluster points. For the IRWA of the previous section this question was answered by appealing to Theorem 2.3.9 which provided necessary and sufficient conditions for the compactness of the lower level sets of the function $J(x, \epsilon)$. This approach also applies to the ADAL algorithm, but the assumptions of Theorem 2.3.9 in conjunction with Assumption 3.3.1 are more than necessary. In the next result we consider two alternative approaches to this issue.

Proposition 3.3.6. *Suppose that the sequence* $\{(x^k, p^k, u^k)\}$ *is generated by ADAL with initial point* $x^0 \in X$. *If either*

(a) $\left[\bar{x} \in X^\infty \cap \ker(H) \cap A^{-1}C^\infty \text{ satisfies } g^T\bar{x} \leq 0\right] \iff \bar{x} = 0$, *or*

(b) Assumption 3.3.1 holds and

$$\left[\tilde{x} \in X^\infty \cap \ker(H) \cap \ker(A) \text{ satisfies } g^T\tilde{x} \leq 0\right] \iff \tilde{x} = 0, \tag{3.3.10}$$

then $\{x^k\}$ *is bounded and every cluster point of this sequence is a solution to* (2.1.4).

Proof. Let us first assume that (a) holds. By Theorem 2.3.9, the condition in (a) (recall (2.3.16)) implies that the set $L(J(x^0, 0), 0)$ is compact. Hence, a solution x^* to (2.1.4) exists. By [58, Theorem 23.7], there exist p^* and u^* such that (x^*, p^*, u^*) satisfies (3.2.2), i.e., Assumption 3.3.1 holds. Since

$$\begin{aligned} J(x^k, 0) &= \varphi(x^k) + \operatorname{dist}\left(Ax^k + b \,|\, C\right) \\ &= \varphi(x^k) + \left\|(Ax^k + b) - P_C(Ax^k + b)\right\| \\ &\leq \varphi(x^k) + \left\|(Ax^k + b) - p^k\right\| + \left\|p^k - P_C(p^k)\right\| + \left\|P_C(p^k) - P_C(Ax^k + b)\right\| \\ &= \hat{J}(x^k, p^k) + 2\left\|z^k\right\|, \end{aligned}$$

the second inequality in (3.3.1) tells us that for all $k \in \mathbb{N}$ we hvae

$$J(x^{k+1}, 0) \leq \hat{J}(x^*, p^*) + 2\left\|z^k\right\| - (u^{k+1})^T z^{k+1} + \frac{1}{\mu}(q^k)^T(p^* - p^{k+1})).$$

By Lemma 3.2.1 and Theorem 3.3.4, the right-hand side of this inequality is bounded for all $k \in \mathbb{N}$, and so, by Theorem 2.3.9, the sequence $\{x^k\}$ is bounded. Corollary 3.3.5 then tells us that all cluster points of this sequence are solutions to (2.1.4).

Now assume that (b) holds. If the sequence $\{x^k\}$ is unbounded, then there is a subsequence $S \subset \mathbb{N}$ and a vector $\bar{x} \in X^\infty$ such that $\left\|x^k\right\|_2 \overset{S}{\to} \infty$ and $x^k/\left\|x^k\right\|_2 \overset{S}{\to} \bar{x}$ with $\|\bar{x}\|_2 = 1$. By Lemma 3.2.1, $\{p^k\}$ is bounded and, by Theorem 3.3.4, $z^k \to 0$. Hence, $(Ax^k + b - p^k)/\left\|x^k\right\|_2 = z^k/\left\|x^k\right\|_2 \overset{S}{\to} 0$ so that $A\bar{x} = 0$. In addition, the sequence $\{\hat{J}(x^k, p^k)\}$ is bounded, which implies $\hat{J}(x^k, p^k)/\left\|x^k\right\|_2^2 \overset{S}{\to} 0$ so that $H\bar{x} = 0$. Moreover,

since H is positive semi-definite, $g^T(x^k/\left\|x^k\right\|_2) \leq \hat{J}(x^k,p^k)/\left\|x^k\right\|_2 \stackrel{S}{\to} 0$ so that $g^T\bar{x} \leq 0$. But then (b) implies that $\bar{x} = 0$. This contradiction implies that the sequence $\{x^k\}$ must be bounded. The result now follows from Corollary 3.3.5. □

Note that, since $\ker(A) \subset A^{-1}C^\infty$, the condition given in (a) implies (3.3.10), and that (3.3.10) is strictly weaker whenever $\ker(A)$ is strictly contained in $A^{-1}C^\infty$.

We conclude this section by stating a result for the case when H is positive definite. As has been observed, in such cases, the function J_0 is strongly convex and so the problem (2.1.4) has a unique global solution x^*. Hence, a proof paralleling that provided for Theorem 2.4.3 applies to give the following result.

Theorem 3.3.7. *Suppose that H is positive definite and the sequence $\{(x^k,p^k,u^k)\}$ is generated by ADAL with initial point $x^0 \in X$. Then, the problem* (2.1.4) *has a unique global solution x^* and $x^k \to x^*$.*

3.4 Complexity of ADAL

In this subsection, we analyze the complexity of ADAL. As was done for IRWA in Theorem 2.5.1, we show that ADAL requires at most $O(1/\varepsilon^2)$ iterations to obtain an ε-optimal solution to the problem (2.1.4). In contrast to this result, some authors [31, 32] establish an $O(1/\varepsilon)$ complexity for ε-optimality for ADAL-type algorithms applied to more general classes of problems, which includes (2.1.4). However, the ADAL decomposition employed by these papers involves subproblems that are as difficult as our original problem (2.1.4), thereby rendering these approaches unsuitable for our purposes. On the other hand, under mild assumptions, the recent results in [68] show that for a general class of problems, which includes (3.1.1), the ADAL algorithm employed here has $\hat{J}(x^k,p^k)$ converging to an ε-optimal solution to (3.1.1) with $O(1/\varepsilon)$ complexity in an *ergodic sense* and $\|Ax+b-p\|_2^2$ converging to a value less than ε with $O(1/\varepsilon)$ complexity. This corresponds to an $O(1/\varepsilon^2)$ complexity for ε-optimality for problem (2.1.4). As of this writing, we know of no result that applies to our ADAL algorithm that establishes a better iteration complexity bound for obtaining an ε-optimal solution to (2.1.4).

We use results in [68] to establish the following result.

Theorem 3.4.1. *Consider the problem* (2.1.4) *with* $X = \mathbb{R}^n$ *and suppose that the sequence* $\{(x^k, p^k, u^k)\}$ *is generated by ADAL with initial point* $x^0 \in X$*. Then, under Assumption 3.3.1, in at most* $O(1/\varepsilon^2)$ *iterations we have an iterate* $x^{\bar{k}}$ *with* $k \leq \bar{k} \leq 2k-1$ *that is* ε*-optimal to* (2.1.4)*, i.e., such that* (2.5.1) *holds with* $\tilde{x} = x^{\bar{k}}$*.*

The key results from [68] used to prove this theorem follow.

Lemma 3.4.2. *[68, Lemma 2] Suppose that the sequence* $\{(x^k, p^k, u^k)\}$ *is generated by ADAL with initial point* $x^0 \in X$*, and, under Assumption 3.3.1, let* (x^*, p^*, u^*) *be the optimal solution of (3.1.1). Then, for all* $k \in \mathbb{N}$*, we have*

$$\begin{aligned}\hat{J}(x^{k+1}, p^{k+1}) - \hat{J}(x^*, p^*) \leq\ & \tfrac{\mu}{2}\left(\left\|u^k\right\|_2^2 - \left\|u^{k+1}\right\|_2^2\right) - \tfrac{1}{2\mu}\left\|Ax^k + b - p^{k+1}\right\|_2^2 \\ & + \tfrac{1}{2\mu}\left(\left\|Ax^* - Ax^k\right\|_2^2 - \left\|Ax^* - Ax^{k+1}\right\|_2^2\right).\end{aligned}$$

Lemma 3.4.3. *[68, Theorem 2] Suppose that the sequence* $\{(x^k, p^k, u^k)\}$ *is generated by ADAL with initial point* $x^0 \in X$*, and, under Assumption 3.3.1, let* (x^*, p^*, u^*) *be the optimal solution of (3.1.1). Then, for all* $k \in \mathbb{N}$*, we have*

$$\left\|Ax^k + b - p^k\right\|_2^2 + \left\|Ax^k - Ax^{k-1}\right\|_2^2 \leq \tfrac{1}{k}\left(\left\|A(x^0 - x^*)\right\|_2^2 + \mu^2\left\|u^0 - u^*\right\|_2^2\right),$$

i.e., in particular, we have

$$\left\|Ax^k + b - p^k\right\|_2^2 \leq \tfrac{1}{k}\left(\left\|A(x^0 - x^*)\right\|_2^2 + \mu^2\left\|u^0 - u^*\right\|_2^2\right).$$

Remark 3.4.4. To see how the previous two lemmas follow from the stated results in [68], the table below provides a guide for translating between our notation and that of [68], which considers the problem

$$\min_{x,z}\ f(x) + g(z) \text{ subject to } Ax + Bz = c. \tag{3.4.1}$$

Problem (3.1.1)	Problem (3.4.1)
(x, p)	(z, x)
φ	g
$\text{dist}\,(\cdot \mid C)$	f
A	B
$-I$	A
$-b$	c

For the results corresponding to our Lemmas 3.4.2 and 3.4.3, [68] requires f and g in (3.4.1) to be closed, proper, and convex functions. In our case, the corresponding functions $\text{dist}\,(\cdot \mid C)$ and φ satisfy these assumptions.

By Lemma 3.3.3, the sequence $\{\omega^k\}$ is monotonically decreasing, meaning that $\{\|Ax^k - Ax^*\|_2^2\}$ and $\{\|u^k\|_2^2\}$ are bounded by some $\tau_1 > 0$ and $\tau_2 > 0$, respectively. The proof of Theorem 3.4.1 now follows as a consequence of the following lemma.

Lemma 3.4.5. *Suppose that the sequence $\{(x^k, p^k, u^k)\}$ is generated by ADAL with initial point $x^0 \in X$, and, under Assumption 3.3.1, let (x^*, p^*, u^*) be the optimal solution of (3.1.1). Moreover, let $\bar{k} \in K := \{k, k+1, \ldots, 2k-1\}$ be such that $\hat{J}(x^{\bar{k}}, p^{\bar{k}}) = \min_{k \in K} \hat{J}(x^k, p^k)$. Then,*

$$J_0(x^{\bar{k}}) - J_0(x^*) \leq \sqrt{\frac{l(\|A(x^0 - x^*)\|_2^2 + \mu^2 \|u^0 - u^*\|_2^2)}{k}} + \frac{\mu\tau_2 + \tau_1/\mu}{k}.$$

Proof. Summing the inequality in Lemma 3.4.2 for $j = k-1, \ldots, 2(k-1)$ yields

$$\begin{aligned}
&\left(\sum_{j=k-1}^{2k-2} \hat{J}(x^{j+1}, p^{j+1})\right) - k\hat{J}(x^*, p^*) \\
&\leq \tfrac{\mu}{2}\left(\left\|u^{k-1}\right\|_2^2 - \left\|u^{2k-1}\right\|_2^2\right) + \tfrac{1}{2\mu}\left(\left\|Ax^* - Ax^{k-1}\right\|_2^2 - \left\|Ax^* - Ax^{2k-1}\right\|_2^2\right) \\
&\leq \mu\tau_2 + \tau_1/\mu. \qquad (3.4.2)
\end{aligned}$$

Therefore,

$$\begin{aligned}\hat{J}(x^{\bar{k}},p^{\bar{k}})-\hat{J}(x^*,p^*) &= \min_{k\le j\le 2k-1}\hat{J}(x^j,p^j)-\hat{J}(x^*,p^*)\\ &\le \tfrac{1}{k}\sum_{j=k-1}^{2k-2}\hat{J}(x^{j+1},p^{j+1})-\hat{J}(x^*,p^*)\\ &\le \tfrac{1}{k}(\mu\tau_2+\tau_1/\mu), \end{aligned} \tag{3.4.3}$$

where the last inequality follows from (3.4.2).

Next, observe that for any $x\in\mathbb{R}^n$ and p, we have

$$\begin{aligned}J_0(x)-\hat{J}(x,p) &= \varphi(x)+\operatorname{dist}(Ax+b\mid C)-(\varphi(x)+\operatorname{dist}(p\mid C))\\ &= \operatorname{dist}(Ax+b\mid C)-\operatorname{dist}(p\mid C)\\ &\le \|Ax+b-p\|\\ &= \sum_{i\in\mathcal{I}}\|A_ix+b_i-p_i\|_2\\ &\le \sqrt{l}\,\|Ax+b-p\|_2, \end{aligned} \tag{3.4.4}$$

where the first inequality follows since $|\operatorname{dist}(z\mid C)-\operatorname{dist}(w\mid C)|\le\|z-w\|$, and the second follows by Jensen's inequality. Combining (3.4.3) and (3.4.4) gives

$$\begin{aligned}J_0(x^{\bar{k}})-J_0(x^*) &= J_0(x^{\bar{k}})-\hat{J}(x^*,p^*)\\ &= J_0(x^{\bar{k}})-\hat{J}(x^{\bar{k}},p^{\bar{k}})+\hat{J}(x^{\bar{k}},p^{\bar{k}})-\hat{J}(x^*,p^*)\\ &\le \sqrt{l}\left\|Ax^{\bar{k}}+b-p^{\bar{k}}\right\|_2+\frac{\mu\tau_2+\tau_1/\mu}{k}\\ &\le \sqrt{\frac{l(\|A(x^0-x^*)\|_2^2+\mu^2\|u^0-u^*\|_2^2)}{k}}+\frac{\mu\tau_2+\tau_1/\mu}{k}, \end{aligned}$$

where the second inequality follows by Lemma 3.4.3 and the fact that $\bar{k}\ge k$. □

Chapter 4

IMPLEMENTATION OF IRWA AND ADAL

4.1 Nesterov Acceleration

In order to improve the performance of both IRWA and ADAL, one can use an acceleration technique due to Nesterov [49]. For the ADAL algorithm, we have implemented the acceleration as described in [33], and for the IRWA algorithm the details are given below. We conjecture that each accelerated algorithm requires $O(1/\varepsilon)$ iterations to produce an ε-optimal solution to (2.1.4), but this remains an open issue.

IRWA with Nesterov Acceleration

Step 0: (Initialization) Choose an initial point $x^0 \in X$, an initial relaxation vector $\epsilon^0 \in \mathbb{R}^l_{++}$, and scaling parameters $\eta \in (0,1)$, $\gamma > 0$, and $M > 0$. Let $\sigma \geq 0$ and $\sigma' \geq 0$ be two scalars which serve as termination tolerances for the stepsize and relaxation parameter, respectively. Set $k := 0$, $y^0 := x^0$, and $t_1 := 1$.

Step 1: (Solve the re-weighted subproblem for x^{k+1})

Compute a solution x^{k+1} to the problem

$$\mathcal{G}(y^k, \epsilon^k): \quad \min_{x \in X} \hat{G}_{(y^k, \epsilon^k)}(x).$$

Let

$$t_{k+1} := \frac{1+\sqrt{1+4(t^k)^2}}{2}$$
$$\text{and} \quad y^{k+1} := x^{k+1} + \frac{t^k - 1}{t_{k+1}}(x^{k+1} - x^k).$$

Step 2: (Set the new relaxation vector ϵ^{k+1})

Set

$$\tilde{q}_i^k := A_i(x^{k+1} - y^k) \quad \text{and} \quad \tilde{r}_i^k := (I - P_{C_i})(A_i y^k + b_i) \quad \forall\, i \in \mathcal{I}_0.$$

If

$$\left\|\tilde{q}_i^k\right\|_2 \leq M\Big[\left\|\tilde{r}_i^k\right\|_2^2 + (\epsilon_i^k)^2\Big]^{\frac{1}{2}+\gamma} \quad \forall\, i \in \mathcal{I},$$

then choose $\epsilon^{k+1} \in (0,\ \eta\epsilon^k]$; else, set $\epsilon^{k+1} := \epsilon^k$. If $J(y^{k+1}, \epsilon^{k+1}) > J(x^{k+1}, \epsilon^{k+1})$, then set $y^{k+1} := x^{k+1}$.

Step 3: (Check stopping criteria)

If $\left\|x^{k+1} - x^k\right\|_2 \leq \sigma$ and $\left\|\epsilon^k\right\|_2 \leq \sigma'$, then stop; else, set $k := k+1$ and go to Step 1.

In this algorithm, the intermediate variable sequence $\{y^k\}$ is included. If y^{k+1} yields an objective function value worse than x^{k+1}, then we re-set $y^{k+1} := x^{k+1}$. This modification preserves the global convergence properties of the original version since

$$\begin{aligned}
& J(x^{k+1}, \epsilon^{k+1}) - J(x^k, \epsilon^k) \\
=\ & J(x^{k+1}, \epsilon^{k+1}) - J(y^k, \epsilon^k) + J(y^k, \epsilon^k) - J(x^k, \epsilon^k) \\
\leq\ & J(x^{k+1}, \epsilon^k) - J(y^k, \epsilon^k) \\
\leq\ & -\tfrac{1}{2}(x^{k+1} - y^k)^T \tilde{A}^T W_k \tilde{A}(x^{k+1} - y^k) \qquad (4.1.1) \\
=\ & -\tfrac{1}{2}(\tilde{q}^k)^T W_k \tilde{q}^k,
\end{aligned}$$

where the inequality (4.1.1) follows from Lemma 2.3.6. Hence, $\frac{1}{2}(\tilde{q}^k)^T W_k \tilde{q}^k$ is summable, as was required for Lemma 2.4.1 and Theorem 2.4.2.

4.2 Application to Systems of Equations and Inequalities

In this section, we discuss how to apply the general results from §**??** and §**??** to the particular case when H is positive definite and the system $Ax+b \in C$ corresponds a system of equations and inequalities. Specifically, we take $l = m$, $X = \mathbb{R}^n$, $C_i = \{0\}$ for $i \in \{1, \ldots, s\}$, and $C_i = \mathbb{R}_-$ for $i \in \{s+1, \ldots, m\}$ so that $C := \{0\}^s \times \mathbb{R}_-^{m-s}$ and

$$\begin{aligned}
J_0(x) &= \varphi(x) + \operatorname{dist}_1(Ax + b \,|\, C) \\
&= \varphi(x) + \sum_{i=1}^{s} |A_i x + b_i| + \sum_{i=s+1}^{m} (A_i x + b_i)_+. \qquad (4.2.1)
\end{aligned}$$

The numerical performance of both IRWA and ADAL on problems of this type will be compared in the following section. For each algorithm, we examine performance relative to a stopping criterion based on percent reduction in the initial duality gap. It is straightforward to show that, since H is positive definite, the Fenchel-Rockafellar dual [58, Theorem 31.2] to (2.1.4) is

$$\begin{array}{ll} \underset{u}{\text{minimize}} & \frac{1}{2}(g + A^T u)^T H^{-1}(g + A^T u) - b^T u + \sum_{i \in \mathcal{I}} \delta^* \left(u_i \mid C_i \right) \\ \text{subject to} & u_i \in \mathbb{B}_2 \; \forall \, i \in \mathcal{I}, \end{array} \tag{4.2.2}$$

which in the case of (4.2.1) reduces to

$$\begin{array}{ll} \underset{u}{\text{minimize}} & \frac{1}{2}(g + A^T u)^T H^{-1}(g + A^T u) - b^T u \\ \text{subject to} & -1 \leq u_i \leq 1, \; i = 1, \ldots, s \\ & 0 \leq u_i \leq 1, \; i = s+1, \ldots, m. \end{array}$$

In the case of linear systems of equations and inequalities, IRWA can be modified to improve the numerical stability of the algorithm. Observe that if both of the sequences $|r_i^k|$ and ϵ_i^k are driven to zero, then the corresponding weight w_i^k diverges to $+\infty$, which may slow convergence by unnecessarily introducing numerical instability. Hence, we propose a modification that addresses those iterations and indices $i \in \{s+1, \ldots, m\}$ for which $(A_i x^k + b_i)_- < 0$, i.e., those inequality constraint indices corresponding inequality constraints that are strictly satisfied (inactive). For such indices, it is not necessary to set $\epsilon_i^{k+1} < \epsilon_i^k$. There are many possible approaches to address this issue, one of which is given in the algorithm given below.

IRWA for Systems of Equations and Inequalities

Step 0: (Initialization) Choose an initial point $x^0 \in X$, initial relaxation vectors $\hat{\epsilon}^0 = \epsilon^0 \in \mathbb{R}^l_{++}$, and scaling parameters $\eta \in (0,1)$, $\gamma > 0$, and $M > 0$. Let $\sigma \geq 0$ and $\sigma' \geq 0$ be two scalars which serve as termination tolerances for the stepsize and relaxation parameter, respectively. Set $k := 0$.

Step 1: (Solve the re-weighted subproblem for x^{k+1})

Compute a solution x^{k+1} to the problem

$$\mathcal{G}(x^k, \epsilon^k): \quad \min_{x \in X} \hat{G}_{(x^k, \epsilon^k)}(x).$$

Step 2: (Set the new relaxation vector ϵ^{k+1})

Set

$$q_i^k := A_i(x^{k+1} - x^k) \quad \text{and} \quad r_i^k := (I - P_{C_i})(A_i x^k + b_i) \quad \forall\, i = 0, \ldots, m.$$

If

$$\left\| q_i^k \right\|_2 \leq M \left[\left\| r_i^k \right\|_2^2 + (\epsilon_i^k)^2 \right]^{\frac{1}{2}+\gamma} \quad \forall\, i = 1, \ldots, m, \tag{4.2.3}$$

then choose $\hat{\epsilon}^{k+1} \in (0,\ \eta\hat{\epsilon}^k]$ and, for $i = 1, \ldots, m$, set

$$\epsilon_i^{k+1} := \begin{cases} \hat{\epsilon}_i^{k+1} & , i = 1, \ldots, s, \\ \epsilon_i^k & , i > s \text{ and } (A_i x^k + b_i)_- \leq -\hat{\epsilon}_i^k, \\ \hat{\epsilon}_i^{k+1} & , \text{otherwise.} \end{cases}$$

Otherwise, if (4.2.3) is not satisfied, then set $\hat{\epsilon}^{k+1} := \hat{\epsilon}^k$ and $\epsilon^{k+1} := \epsilon^k$.

Step 3: (Check stopping criteria)

If $\left\| x^{k+1} - x^k \right\|_2 \leq \sigma$ and $\left\| \hat{\epsilon}^k \right\|_2 \leq \sigma'$, then stop; else, set $k := k + 1$ and go to Step 1.

Remark 4.2.1. In Step 2 of the algorithm above, the updating scheme for ϵ can be modified in a variety of ways. For example, one can also take $\epsilon_i^{k+1} := \epsilon_i^k$ when $i > s$ and $(A_i x^k + b_i)_- < 0$.

This algorithm yields the following version of Lemma 2.4.1.

Lemma 4.2.2. *Suppose that the sequence $\{(x^k, \epsilon^k)\}$ is generated by IRWA for Systems of Equations and Inequalities with initial point $x^0 \in X$ and relaxation vector $\epsilon^0 \in \mathbb{R}^l_{++}$, and, for $k \in \mathbb{N}$, let q_i^k and r_i^k for $i \in \mathcal{I}_0$ be as defined in Step 2 of the algorithm with*

$$q^k := ((q_0^k)^T, \ldots, (q_l^k)^T)^T \quad \textit{and} \quad r^k := ((r_0^k)^T, \ldots, (r_l^k)^T)^T.$$

Moreover, for $k \in \mathbb{N}$, define

$$w_i^k := w_i(x^k, \epsilon^k) \text{ for } i \in \mathcal{I}_0 \quad \textit{and} \quad W_k := W(x^k, \epsilon^k),$$

and set $S := \left\{ k \;\middle|\; \epsilon^{k+1} \leq \eta\epsilon^k \right\}$. Then, the sequence $\{J(x^k, \epsilon^k)\}$ is monotonically decreasing. Moreover, either $\inf_{k\in\mathbb{N}} J(x^k, \epsilon^k) = -\infty$, in which case $\inf_{x\in X} J_0(x) = -\infty$, or the following hold:

(1) $\sum_{k=0}^{\infty}(q^k)^T W_k q^k < \infty$.

(2) $\hat{\epsilon}^k \to 0$ *and* $H(x^{k+1} - x^k) \to 0$.

(3) $W_k q^k \overset{S}{\to} 0$.

(4) $w_i^k r_i^k = r_i^k / \sqrt{\left\|r_i^k\right\|_2^2 + \epsilon_i^k} \in \mathbb{B}_2 \cap N\left(P_{C_i}(A_i x^k + b_i) \,|\, C_i\right),\ i \in \mathcal{I},\ k \in \mathbb{N}$.

(5) $-\tilde{A}^T W_k q^k \in (\nabla\varphi(x^k) + \sum_{i\in\mathcal{I}} A_i^T w_i^k r_i^k) + N\left(x^{k+1} \,|\, X\right),\ k \in \mathbb{N}$.

(6) If $\{\mathrm{dist}\left(Ax^k + b \,|\, C\right)\}_{k\in S}$ *is bounded, then* $q^k \overset{S}{\to} 0$.

Proof. Note that Lemma 2.3.6 still applies since it is only concerned with properties of the functions $\hat{G}$ and J. In addition, note that

$$\hat{\epsilon}^{k+1} \leq \hat{\epsilon}^k \quad \text{and} \quad \hat{\epsilon}^{k+1} \leq \epsilon^{k+1} \leq \epsilon^k \quad \forall\, k \geq 1.$$

With these observations, the proof of this lemma follows in precisely the same way as that of Lemma 2.4.1, except that in Part (2) $\{\hat{\epsilon}^k\}$ replaces $\{\epsilon^k\}$. □

With Lemma 4.2.2, it is straightforward to show that the convergence properties described in Theorems 2.4.2 and 2.4.3 also hold for the version of IRWA in this section.

4.3 Numerical Comparison of IRWA and ADAL

In this section, we compare the performance of our IRWA and ADAL algorithms in a set of three numerical experiments. The first two experiments involves cases where H is positive definite and the desired inclusion $Ax + b \in C$ corresponds to a system of equations and inequalities. Hence, for these experiments, we employ the version of IRWA as tailored for such systems given in the previous section. In the first experiment, we fix the problem dimensions and compare the behavior of the two algorithms over 500 randomly generated problems. In the second experiment, we investigate how the methods behave when we scale up the problem size. For this purpose, we compare performance over 20 randomly generated problems of increasing dimension. The algorithms were implemented in Python using the

NumPy and SciPy packages; in particular, we used the versions Python 2.7, Numpy 1.6.1, SciPy 0.12.0 [43, 52]. In both experiments, we examine performance relative to a stopping criterion based on percent reduction in the initial duality gap. In IRWA, the variables $\tilde{u}^k := W_k r^k$ are always dual feasible, i.e.,

$$\tilde{u}_i \in \mathbb{B}_2 \cap \operatorname{dom}\left(\delta^*\left(\cdot \mid C_i\right)\right) \quad \forall\, i \in \mathcal{I}$$

(recall Lemma 2.4.1(4)), and these variables constitute our kth estimate to the dual solution. On the other hand, in ADAL, the variables $\hat{u}^k = u^k - \frac{1}{\mu} q^k$ are always dual feasible (recall Lemma 3.2.1), so these constitute our kth estimate to the dual solution for this algorithm. The duality gap at any iteration is the sum of the primal and dual objectives at the current primal-dual iterates.

In both IRWA and ADAL, we solve the subproblems using CG, which is terminated when the ℓ_2-norm of the residual is less than 10% of the norm of the initial residual. At each iteration, the CG algorithm is initiated at the previous step x^{k-1}. In both experiments, we set $x^0 := 0$, and in ADAL we set $u^0 := 0$. It is worthwhile to note that we have observed that the performance of IRWA is sensitive to the initial choice of ϵ^0 while ADAL is sensitive to μ. We do not investigate this sensitivity in detail when presenting the results of our experiments, and we have no theoretical justification for our choices of these parameters. However, we empirically observe that these values should increase with dimension. For each method, we have chosen an automatic procedure for initializing these values that yields good overall performance. The details are given in the experimental descriptions. A more principled method for initializing and updating these parameters is the subject of future research.

In the third experiment, we apply both algorithms to an l_1 support vector machine (SVM) problem. Details are given in the experimental description. In this case, we use the stopping criteria as stated along with the algorithm descriptions in the paper, i.e., not a criterion based on a percent reduction in duality gap. In this experiment, the subproblems are solved as in the first two experiments with the same termination and warm-start rules.

First Experiment: In this experiment, we randomly generated 500 instances of problem (4.2.1). For each, we generated $A \in \mathbb{R}^{600\times 1000}$ and chose C so that the inclusion $Ax + b \in C$

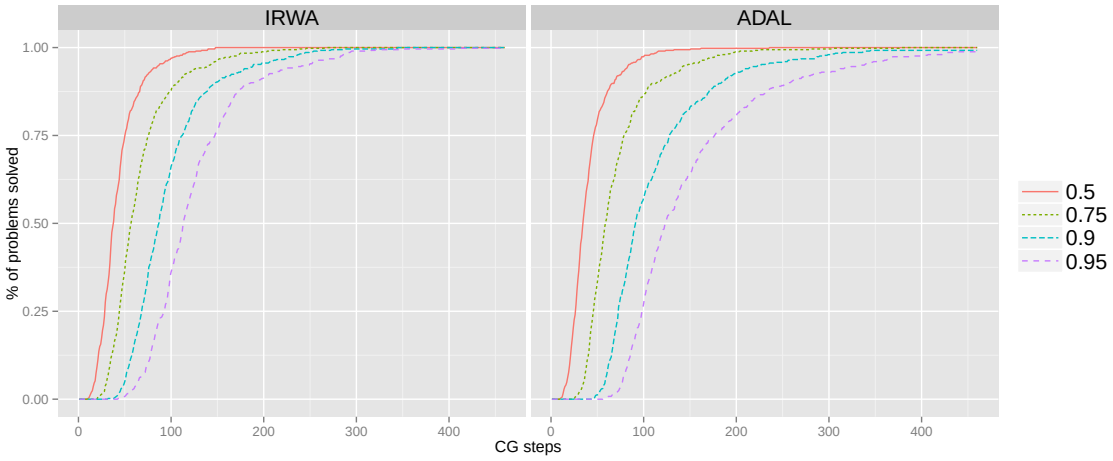

Figure 4.1: Efficiency curves for IRWA (left panel) and ADAL (right panel). The percentage of the 500 problems solved is plotted versus the total number of CG steps. IRWA terminated in fewer than 460 CG steps on all problems. ADAL required over 460 CG steps on 8 of the problems.

corresponded to 300 equations and 300 inequalities. Each matrix A is obtained by first randomly choosing a mean and variance from the integers on the interval $[1, 10]$ with equal probability. Then the elements of A are chosen from a normal distribution having this mean and variance. Similarly, each of the vectors b and g are constructed by first randomly choosing integers on the intervals $[-100, 100]$ for the mean and $[1, 100]$ for the variance with equal probability and then obtaining the elements of these vectors from a normal distribution having this mean and variance. Each matrix H had the form $H = 0.1I + LL^T$ where the elements of $L \in \mathbb{R}^{n \times n}$ are chosen from a normal distribution having mean 1 and variance 2. For the input parameters for the algorithms, we chose $\eta := 0.6$, $M := 10^4$, $\gamma := \frac{1}{6}$, $\mu := 100$, and $\epsilon_i^0 := 2000$ for each $i \in \mathcal{I}$. Efficiency curves for both algorithms are given in Figure 4.1, which illustrates the percentage of problems solved versus the total number of CG steps required to reduce the duality gap by 50, 75, 90 and 95 percent. The greatest number of CG steps required by IRWA was 460 when reducing the duality gap by 95%. ADAL stumbled at the 95% level on 8 problems, requiring $609, 494, 628, 674, 866, 467, 563, 856, 676$

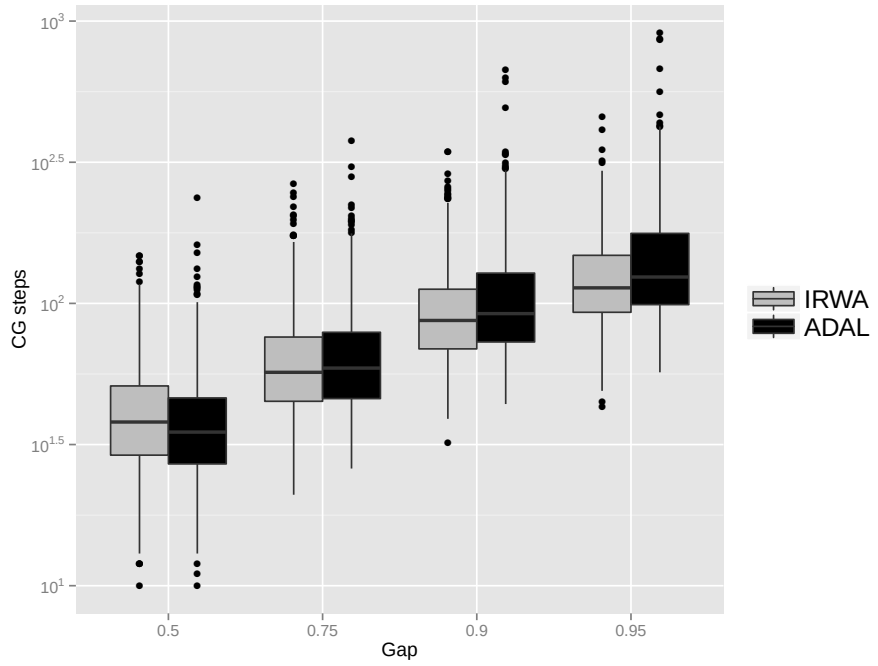

Figure 4.2: Box plot of CG steps for IRWA and ADAL for each duality gap threshold.

and 911 CG steps for these problems. Figure 4.2 contains a box plot for the log of the number of CG iterations required by each algorithm for each of the selected accuracy levels. Overall, in this experiment, the methods seem comparable with a slight advantage to IRWA in both the mean and variance of the number of required CG steps.

Second Experiment: In the second experiment, we randomly generated 20 problems of increasing dimension. The numbers of variables were chosen to be $n = 200 + 500(j-1),\ j = 1, \ldots, 20$, where for each we set $m := n/2$ so that the inclusion $Ax + b \in C$ corresponds to equal numbers of equations and inequalities. The matrix A was generated as in the first experiment. Each of the vectors b and g were constructed by first choosing integers on the intervals $[-200, 200]$ for the mean and $[1, 200]$ for the variance with equal probability and then obtaining the elements of these vectors from a normal distribution having this mean and variance. Each matrix H had the form $H = 40I + LDL^T$, where $L \in \mathbb{R}^{n \times k}$

Figure 4.3: Increase in CG steps for each duality gap (left panel) and CPU time (right panel) for IRWA and ADAL as dimension is increased.

with $k = 8$ and D was diagonal. The elements of L were constructed in the same way as those of A, and those of D were obtained by sampling from the inverse gamma distribution $f(x) := \frac{b^a}{\Gamma(a)} x^{-a-1} e^{-b/x}$ with $a = 0.5,\ b = 1$. We set $\eta := 0.5$, $M := 10^4$, and $\gamma := \frac{1}{6}$, and for each $j = 1, \ldots, 20$ we set $\epsilon_i^0 := 10^{2+1.3\ln(j+10)}$ for each $i = 1, \ldots, m$, and $\mu := 500(1+j)$. In Figure 4.3, we present two plots showing the number of CG steps and the log of the CPU times versus variable dimensions for the two methods. The plots illustrate that the algorithms performed similarly in this experiment.

Third Experiment: In this experiment, we solve the l_1-SVM problem as introduced in [75]. In particular, we consider the exact penalty form

$$\min_{\beta \in \mathbb{R}^n} \sum_{i=1}^{m} \left(1 - y_i \left(\sum_{j=1}^{n} x_{ij}\beta_j\right)\right)_+ + \lambda \|\beta\|_1, \tag{4.3.1}$$

where $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^m$ are the training data points with $\boldsymbol{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$ for each $i = 1, \ldots, m$, and λ is the penalty parameter. In this experiment, we randomly generated

40 problems in the following way. First, we sampled an integer on $[1, 5]$ and another on $[6, 10]$, both from discrete uniform distributions. These integers were taken as the mean and standard deviation of a normal distribution, respectively. We then generated an $m \times s$ component-wise normal random matrix T, where s was chosen to be $19+2j,\ j = 0, 1, \ldots, 39$ and m was chosen to be $200 + 10j,\ j = 0, 1, \ldots, 39$. We then generated an s-dimensional integer vector $\hat{\beta}$ whose components were sampled from the discrete uniform distribution on the integers between -100 and 100. Then, y_i was chosen to be the sign of the i-th component of $T\hat{\beta}$. In addition, we generated an $m \times t$ i.i.d. standard normal random matrix R, where t was chosen to be $200+30j,\ j = 0, 1, \ldots, 39$. Then, we let $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m]^T := [T, R]$. For all 40 problems, we fixed the penalty parameter at $\lambda = 50$. In this application, the problems need to be solved exactly, i.e., a percent reduction in duality gap is insufficient. Hence, in this experiment, we use the stopping criteria as described in Step 3 of both IRWA and ADAL. For IRWA, we set $\epsilon_i^0 := 10^4$ for all $i \in \mathcal{I}$, $\eta := 0.7$, $M := 10^4$, $\gamma := \frac{1}{6}$, $\sigma := 10^{-4}$ and $\sigma' := 10^{-8}$. For ADAL, we set $\mu := 1$, $\sigma := 0.05$ and $\sigma'' := 0.05$. We also set the maximum iteration limit for ADAL to 150. Both algorithms were initialized at $\beta := 0$. Figure 4.4 has two plots showing the objective function values of both algorithms at termination, and the total CG steps taken by each algorithm. These two plots show superior performance for IRWA when solving these 40 problems.

Based on how the problems were generated, we would expect the non-zero coefficients of the optimal solution β to be among the first $s = 19 + 2j,\ j = 0, \ldots, 39$ components corresponding to the matrix T. To investigate this, we considered "zero" thresholds of $10^{-3}, 10^{-4}$ and 10^{-5}; i.e., we considered a component as being "equal" to zero if its absolute value was less than a given threshold. Figure 4.5 shows a summary of the number of unexpected zeros for each algorithm. These plots show that IRWA has significantly fewer false positives for the nonzero components, and in this respect returned preferable sparse recovery results over ADAL in this experiment.

Finally, we use this experiment to demonstrate Nesterov's acceleration for IRWA. The effect on ADAL has already been shown in [33], so we only focus on the effect of accelerating IRWA. The 40 problems were solved using both IRWA and accelerated IRWA with the parameters stated above. Figure 4.6 shows the differences between the objective function

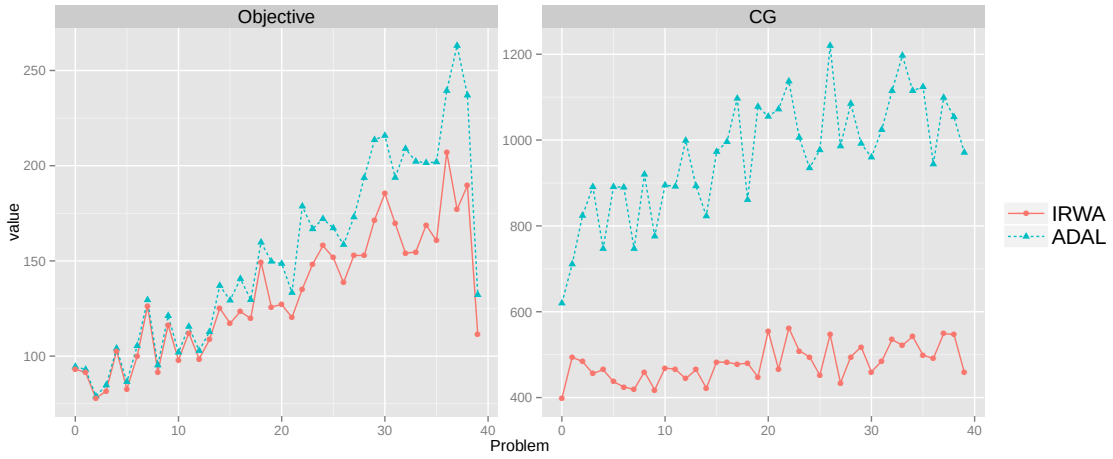

Figure 4.4: In all 40 problems, IRWA obtains smaller objective function values with fewer CG steps.

values ($\frac{\text{normal}-\text{accelerated}}{\text{accelerated}} \times 100$) and the number of CG steps (normal $-$ accelerated) needed to converge. The graphs show that accelerated IRWA performs significantly better than unaccelerated IRWA in terms of both objective function values obtained and CG steps required.

4.4 Conclusion

In this paper, we have proposed, analyzed, and tested two matrix-free solvers for approximately solving the exact penalty subproblem (2.1.4). The primary novelty of our work is a newly proposed iterative re-weighting algorithm (IRWA) for solving such problems involving arbitrary convex sets of the form (2.1.1). In each iteration of our IRWA algorithm, a quadratic model of a relaxed problem is formed and solved to determine the next iterate. Similarly, the alternating direction augmented Lagrangian (ADAL) algorithm that we present also has as its main computational component the minimization of a convex quadratic subproblem. Both solvers can be applied in large scale settings, and both can be implemented matrix-free.

Variations of our algorithms were implemented and the performance of these imple-

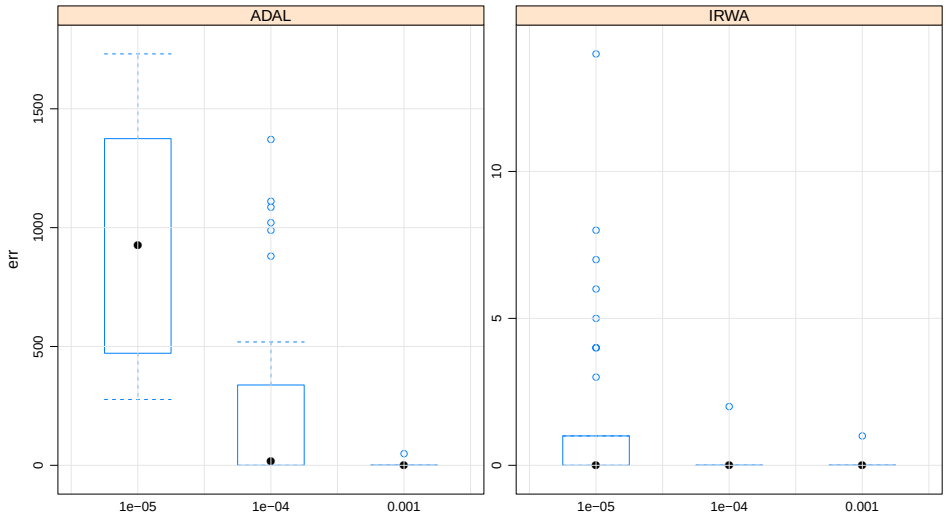

Figure 4.5: For both thresholds 10^{-4} and 10^{-5}, IRWA yields fewer false positives in terms of the numbers of "zero" values computed. The numbers of false positives are similar for the threshold 10^{-3}. At the threshold 10^{-5}, the difference in recovery is dramatic with IRWA always having fewer than 14 false positives while ADAL has a median of about 1000 false positives.

mentations were tested. Our test results indicate that both types of algorithms perform similarly on many test problems. However, a test on an ℓ_1-SVM problem illustrates that in some applications the IRWA algorithms can have superior performance. Overall, our investigation leads to a variety of open questions:

- How should the relaxation vector ϵ in IRWA and the penalty parameter μ in ADAL be initialized and updated in order for the methods to be most effective when solving a particular problem instance?

- Are there certain problem classes for which one should expect superior performance of IRWA over ADAL, or vice versa? If so, then what feature(s) of a given problem class leads to the different levels of performance? For example, although IRWA was

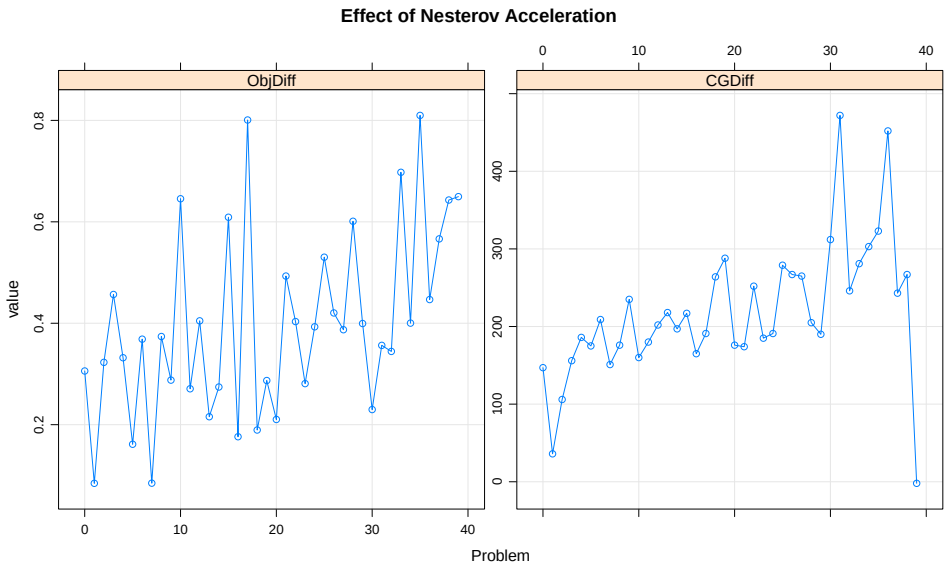

Figure 4.6: Differences in objective function values (left panel) obtained by normal and accelerated IRWA ($\frac{\text{normal}-\text{accelerated}}{\text{accelerated}} \times 100$), and differences in numbers of CG steps (right panel) required to converge to the objective function value in the left panel (normal − accelerated). Accelerated IRWA always converged to a point with a smaller objective function value, and accelerated IRWA typically required fewer CG steps. (There was only one exception, the last problem, where accelerated IRWA required two more CG steps.)

superior on the l_1-SVM problem presented in our third experiment (see Figures 4.4 and 4.5), we do not have a clear understanding of why this was the case. Conversely, are there instances within the problem class (2.1.4) where ADAL is clearly superior to IRWA?

- We have observed that, although IRWA and ADAL performed similarly on many of the problems in our first experiment, IRWA seemed to perform better as greater accuracy was requested, whereas ADAL occasionally stumbled under these requests (see Figures 4.1 and 4.2). Further numerical testing did not reveal a strategy for

consistently improving the performance of ADAL when higher accuracy was desired. Is there a way to tune ADAL to overcome this drawback, or is this perceived drawback simply an artifact of our experimental design?

- Our implementation of both IRWA and ADAL uses the Nesterov acceleration, and this innovation can have a dramatic impact on the performance of these methods (see Figure 4.6). However, we have not been successful in providing a complexity analysis for these implementations. We conjecture that the accelerated algorithms require $O(1/\varepsilon)$ iterations to produce an ε-optimal solution to (2.1.4).

Chapter 5

GENERALIZATION OF IRWA

5.1 Introduction

Consider the following optimization problem:

$$\min_x \sum_{i=1}^{N} f_i(x), \tag{5.1.1}$$

where $f_i : \mathbb{R}^n \to \bar{\mathbb{R}}$ are convex but not necessarily smooth. Define a smoothed approximation of the objective function using Moreau envelopes by

$$\min_x F_{\boldsymbol{\lambda}}(x) := \sum_{i=1}^{N} e_{\lambda_i} f_i(x), \tag{5.1.2}$$

where $e_{\lambda_i} f_i(x) := \inf_w (f_i(w) + \frac{1}{2\lambda_i} \|w - x\|^2)$ is the Moreau envelope function and $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_N]$. We first consider the case of fixed $\boldsymbol{\lambda}$. Instead of solving (5.1.2), at each x^k we solve a weighted least squares problem given by

$$x^{k+1} := \operatorname{argmin}_x \quad G_{(\boldsymbol{\lambda}, x^k)}(x) := \textstyle\sum_{i=1}^{N} f_i(p_i^k) + \frac{1}{2\lambda_i} \left\|x - p_i^k\right\|^2, \tag{5.1.3}$$

where $p_i^k := \operatorname{prox}_{\lambda_i f_i}(x^k)$. It is easily seen that $x^{k+1} = \frac{\sum_{i=1}^{N} \frac{1}{\lambda_i} p_i^k}{\sum_{i=1}^{N} \frac{1}{\lambda_i}}$. We have the following proposition about the iterates x^k obtained in this way.

Proposition 5.1.1. *Assume $F_{\boldsymbol{\lambda}}(x) > -\infty \ \forall x \in \mathbb{R}^n$. If $\{x^k\}_{k=1}^{\infty}$ are generated by solving* (5.1.3) *iteratively for fixed $\boldsymbol{\lambda} > 0$, then*

$$\sum_{k=1}^{\infty} \left\| x^{k+1} - x^k \right\|^2 < \infty. \tag{5.1.4}$$

Proof. First observe that

$$\begin{aligned}
&G_{(\boldsymbol{\lambda},x^k)}(x^{k+1}) - G_{(\boldsymbol{\lambda},x^k)}(x^k) \\
=&\sum_{i=1}^{N} \frac{1}{2\lambda_i}(\left\|x^{k+1} - p_i^k\right\|^2 - \left\|x^k - p_i^k\right\|^2) \\
=&-(\sum_{i=1}^{N} \frac{1}{2\lambda_i}) \left\|x^{k+1} - x^k\right\|^2.
\end{aligned}$$

In addition,

$$\begin{aligned}
e_{\lambda_i} f_i(x^{k+1}) &= \inf_w f_i(w) + \frac{1}{2\lambda_i} \left\|w - x^{k+1}\right\|^2 \\
&\leq f_i(p_i^k) + \frac{1}{2\lambda_i} \left\|p_i^k - x^{k+1}\right\|^2.
\end{aligned}$$

Hence,

$$\begin{aligned}
F_{\boldsymbol{\lambda}}(x^{k+1}) &= \sum_{i=1}^{N} e_{\lambda_i} f_i(x^{k+1}) \\
&\leq \sum_{i=1}^{N} f_i(p_i^k) + \frac{1}{2\lambda_i} \left\|p_i^k - x^{k+1}\right\|^2 \\
&= G_{(\boldsymbol{\lambda},x^k)}(x^{k+1}).
\end{aligned}$$

Notice that $F_{\boldsymbol{\lambda}}(x^k) = G_{(\boldsymbol{\lambda},x^k)}(x^k)$. Therefore

$$\begin{aligned}
&F_{\boldsymbol{\lambda}}(x^{k+1}) - F_{\boldsymbol{\lambda}}(x^k) \\
&\leq G_{(\boldsymbol{\lambda},x^k)}(x^{k+1}) - G_{(\boldsymbol{\lambda},x^k)}(x^k) \\
&\leq -(\sum_{i=1}^{N} \frac{1}{2\lambda_i}) \left\|x^{k+1} - x^k\right\|^2.
\end{aligned}$$

Summing over k from 1 to N, we have

$$\sum_{k=1}^{N} \left\|x^{k+1} - x^k\right\|^2 < F_{\boldsymbol{\lambda}}(x^1) - F_{\boldsymbol{\lambda}}(x^{N+1}) < \infty,$$

where the last inequality is due to $F_{\boldsymbol{\lambda}}(x) > -\infty$. Let $N \to \infty$, we have

$$\sum_{k=1}^{\infty} \left\|x^{k+1} - x^k\right\|^2 < \infty.$$

□

Theorem 5.1.2. *Assume $F_{\boldsymbol{\lambda}}(x) > -\infty \ \forall x \in \mathbb{R}^n$. If $\{x^k\}_{k=1}^{\infty}$ are generated by solving* (5.1.3) *iteratively for fixed $\boldsymbol{\lambda} > 0$, then every cluster point of $\{x^k\}_{k=1}^{\infty}$ is a global solution to the optimization problem* (5.1.2).

Proof. Proposition 5.1.1 shows that $x^{k+1} - x^k \to 0$. Assume $\{x^k\}_{k\in S}$ is a convergent subsequence, i.e. $x^k \overset{S}{\to} \bar{x}$. Then we have

$$p_i^k = \operatorname{prox}_{\lambda_i f_i}(x^k) \overset{S}{\to} \operatorname{prox}_{\lambda_i f_i}(\bar{x}) \quad i = 1, \cdots, N.$$

Optimality condition of (5.1.3) implies

$$\sum_{i=1}^{N} \frac{x^{k+1} - p_i^k}{\lambda_i} = 0, \tag{5.1.5}$$

i.e.

$$\sum_{i=1}^{N} \frac{x^k - p_i^k}{\lambda_i} + (\sum_{i=1}^{N} \frac{1}{\lambda_i})(x^{k+1} - x^k) = 0.$$

Let $k \overset{S}{\to} \infty$ and by $x^{k+1} - x^k \to 0$, we have

$$\sum_{i=1}^{N} \frac{\bar{x} - \operatorname{prox}_{\lambda_i f_i}(\bar{x})}{\lambda_i} = 0.$$

Hence $\bar{x}$ solves (5.1.2). □

Corollary 5.1.3 (Coercivity)**.** *Let $F_{\boldsymbol{\lambda}}$ and $\{x^k\}_{k=1}^{\infty}$ be as given in Theorem 5.1.2. If any one of the functions f_i $i \in \{1, \ldots, N\}$ is coercive, then $F_{\boldsymbol{\lambda}}$ is coercive and the sequence $\{x^k\}_{k=1}^{\infty}$ has cluster points that are global solutions to the optimization problem* (5.1.2).

Proof. The result follows from Theorem 5.1.2 once it is shown that $F_{\boldsymbol{\lambda}}$ is coercive. For this we need only show that $e_{\lambda_i} f_i$ is coercive for some $i \in \{1, \ldots, N\}$. Let $i \in \{1, \ldots, N\}$ be chosen so that f_i is coercive so that $0 \in \operatorname{intr}(\operatorname{dom}(f)_i^*)$ [59, Theorem 11.8]. Then $(e_{\lambda_i} f_i)^* = f_i^* + \lambda_i/2 \, \|\cdot\|_2^2$. Hence, $0 \in \operatorname{intr}(\operatorname{dom}((\,) e_{\lambda_i} f_i)^*)$, and so $e_{\lambda_i} f_i$ is coercive by [59, Theorem 11.8]. □

5.2 Continuation and Practical Version

A good guess of λ_i^k is $\left\|x^k - p_i^{k-1}\right\|_2 + \epsilon_i^k$, where ϵ_i^k is a non-negative constant. ϵ_i^k will be updated based on the relation between $\left\|x^{k+1} - x^k\right\|_2$ and $\left\|x^k - p_i^{k-1}\right\|_2 + \epsilon_i^k$. If $\left\|x^{k+1} - x^k\right\|_2 < \left\|x^k - p_i^{k-1}\right\|_2 + \epsilon_i^k$, then set $\epsilon_i^{k+1} = \eta\epsilon_i^k,\ \eta \in (0,1)$, otherwise $\epsilon_i^{k+1} = \epsilon_i^k$. Hence we have the following algorithm for (5.1.1), see Algorithm 1.

Algorithm 1 GIRWA

Initilization: $x^0, \eta \in (0,1), \lambda_i^0 > 0, \epsilon_i^0 > 0,\ i = 1, \cdots, N$ and $\varepsilon > 0$.

Step 1: Set $p_i^k := \text{prox}_{\lambda_i^k f}(x^k)$, and

$$x^{k+1} = \frac{\sum_{i=1}^N \frac{1}{\lambda_i^k} p_i^k}{\sum_{i=1}^N \frac{1}{\lambda_i^k}}.$$

Step 2: Update λ_i^k and ϵ_i^k. If $\left\|x^{k+1} - x^k\right\|_2 < \left\|x^k - p_i^{k-1}\right\|_2 + \epsilon_i^k$, then set $\epsilon_i^{k+1} = \eta\epsilon_i^k,\ \eta \in (0,1)$, otherwise $\epsilon_i^{k+1} = \epsilon_i^k$.
Set $\lambda_i^k = \left\|x^{k+1} - p_i^k\right\|_2 + \epsilon_i^{k+1}$.

Step 3: Check stopping criteria.

Nesterov's acceleration [49] could be combined with GIRWA naturally. Instead of using x^k to obtain p_i^k, we use a combination of x^k and x^{k-1} to calculate p_i^k. Then we have the accelerated GIRWA, see Algorithm 2.

5.3 Numerical Experiments

Experiment 1: Consider the l_1 regularized regression problem

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 + \beta \|x\|_1 .$$

A is an $m \times n$ random matrix with each element sampled i.i.d. from standard normal distribution. We set $m = 600$ and $n = 5000$. We test our algorithm with $\beta = 5, 10$. No warm start is used. We compare our algorithm with ADMM. Figure 5.3 shows that GIRWA is able to converge faster to a better solution than ADMM.

Algorithm 2 Accelerated GIRWA

Initilization: $x^0, y^0 = x^0, \eta \in (0,1), \lambda_i^0 > 0, t_0 = 1, \epsilon_i^0 > 0,\ i = 1, \cdots, N$ and $\varepsilon > 0$.

Step 1: Set $p_i^k := \text{prox}_{\lambda_i^k f}(y^k)$, and

$$x^{k+1} = \frac{\sum_{i=1}^N \frac{1}{\lambda_i^k} p_i^k}{\sum_{i=1}^N \frac{1}{\lambda_i^k}} \qquad t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$$

$$y^{k+1} = x^{k+1} + \frac{t_k - 1}{t_{k+1}}(x^{k+1} - x^k)$$

Step 2: Update λ_i^k and ϵ_i^k. If $\left\|x^{k+1} - x^k\right\|_2 < \left\|x^k - p_i^{k-1}\right\|_2 + \epsilon_i^k$, then set $\epsilon_i^{k+1} = \eta\epsilon_i^k,\ \eta \in (0,1)$, otherwise $\epsilon_i^{k+1} = \epsilon_i^k$.

Set $\lambda_i^k = \left\|x^{k+1} - p_i^k\right\|_2 + \epsilon_i^{k+1}$.

Step 3: Check stopping criteria.

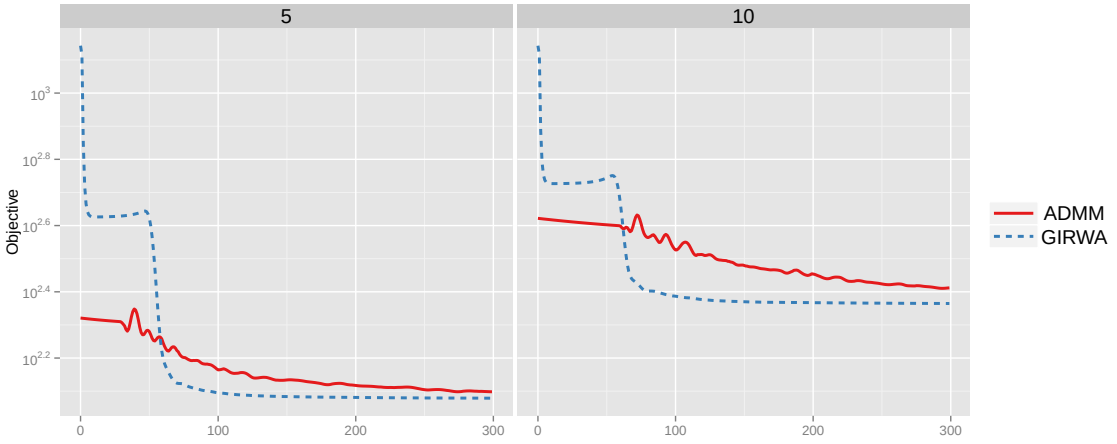

Figure 5.1: Left panel is for $\beta = 5$ and right panel is for $\beta = 10$.

Chapter 6

LINEARLY CONSTRAINED NONLINEAR PROGRAMMING

6.1 Introduction

Consider the linearly constrained nonlinear programming problem which is defined as

$$\begin{array}{lll} \text{(LNP)} & \min_x & f(x) \\ & \text{s.t.} & Ex = e \\ & & Gx \leq p, \end{array} \tag{6.1.1}$$

where $f(x)$ is twice differentiable, $E \in \mathbb{R}^{m_1 \times n}$ and $G \in \mathbb{R}^{m_2 \times n}$. We assume the linear constraints of (6.1.1) are consistent. Problem (6.1.1) can be approached by solving a sequence of quadratic subproblems. Taking the quadratic approximation of $f(x)$ at a feasible point x^k, the subproblem becomes the following linearly constrained quadratic programming problem

$$\begin{array}{ll} \min\limits_d & \frac{1}{2} d^T H_k d + \nabla f(x^k)^T d \\ \text{s.t.} & Ed = 0 \\ & Gd + Gx^k \leq p, \end{array} \tag{6.1.2}$$

where H_k is the Hessian approximation of $f(x)$ at x^k.

If (6.1.1) has only bound constraints, i.e. $l \leq x \leq c$, [14] applied limited memory BFGS together with gradient projection to solve it quite successfully. However gradient projection technique can not be employed to solve (6.1.2) for general linear constraints. Active set and interior point methods are the most widely used techniques to solve (6.1.2). These have been implemented in packages like [29, SNOPT] and [15, KNITRO]. The package [66, IPOPT] employs an interior point algorithm to solve (6.1.2) while [15, KNITRO] implements an interior point algorithm which either solves the Newton equation directly or through conjugate gradient (CG) method. The package [1, CVXOPT] is a conic program solver which uses a path-following algorithm to solve (6.1.2). All these methods solve the underlying linear systems of equation by (sparse) factorization or CG.

6.2 Review of Limited Memory BFGS

Suppose $\{x^k\}$ is a sequence generated to solve (6.1.1). Define

$$s^k = x^{k+1} - x^k, \quad y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$$
$$S_k = [s^0, s^1, \cdots, s^{k-1}], \quad Y_k = [y^0, y^1, \cdots, y^{k-1}].$$

Then the direct BFGS update formula is given by

$$H_{k+1} = H_k - \frac{H_k s^k s^{kT} H_k}{s^{kT} H_k s^k} + \frac{y^k y^{kT}}{y^{kT} s^k}, \tag{6.2.1}$$

where $s^{kT} := (s^k)^T$. The compact update of H_k can be summarized in the following theorem [16, Theorem 2.3].

Theorem 6.2.1. *[16, Theorem 2.3] Let H_0 be symmetric and positive definite and assume that the k pairs $\{s^i, y^i\}_{i=0}^{k-1}$ satisfy $s^{iT} y^i > 0$. Let H_k be obtained by updating H_0 k times with direct BFGS formula (6.2.1) using the pairs $\{s^i, y^i\}_{i=0}^{k-1}$. Then*

$$H_k = H_0 - \begin{bmatrix} H_0 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T H_0 S_k & P_k \\ P_k^T & -D_k \end{bmatrix}^{-1} \begin{bmatrix} S_k^T H_0 \\ Y_k^T \end{bmatrix}, \tag{6.2.2}$$

where P_k is the $k \times k$ matrix

$$(P_k)_{i,j} = \begin{cases} s^{(i-1)T} y^{j-1} & \text{if } i > j \\ 0 & \text{otherwise,} \end{cases}$$

and $D_k = \mathrm{diag}[s^{0T} y^0, \cdots, s^{(k-1)T} y^{k-1}]$.

In the limited memory setting, we only keep the r most recent s^k and y^k in S_k and Y_k, i.e.

$$S_k = [s^{k-r}, \cdots, s^{k-1}]$$
$$Y_k = [y^{k-r}, \cdots, y^{k-1}].$$

In this setting, H_k can be written as

$$H_k := \alpha I - L \Sigma^{-1} L^T, \tag{6.2.3}$$

where $L \in \mathbb{R}^{n\times 2r}$, and H_k^{-1} can be computed by the Sherman-Morrison-Woodbury formula

$$(\alpha I - L\Sigma^{-1}L^T)^{-1} = \frac{1}{\alpha}[I - L(\alpha\Sigma + L^TL)^{-1}L^T]. \tag{6.2.4}$$

Define

$$A = [E^T, G^T]^T, \quad b = [\mathbf{0}_{m_1}^T, (p - Gx^k)^T]^T, \quad \text{and} \quad \lambda = [u^T, v^T]^T,$$

where $u \in \mathbb{R}^{m_1}$ is the Lagrangian multiplier corresponding to the equality constrains and $v \in \mathbb{R}^{m_2}$ corresponds to inequality constraints (6.1.2) so that $\lambda \in \mathbb{R}^m$ with $m = m_1 + m_2$. A direct calculation shows the Lagrangian dual of (6.1.2) is

$$\begin{aligned} &\min_{u,v} \ \frac{1}{2}\lambda^T AH_k^{-1}A^T\lambda + \lambda^T(b + AH_k^{-1}\nabla f(x^k)) \\ &\text{s.t. } v \geq 0. \end{aligned} \tag{6.2.5}$$

If $\bar{\lambda}$ is a solution to (6.2.5), then the solution to (6.1.2) can be recovered as $d = -H_k^{-1}(A^T\bar{\lambda} + \nabla f(x^k))$.

Before introducing our algorithm for (6.1.1), we describe a coordinate descent algorithm for solving (6.2.5).

6.3 Coordinate Descent Algorithm

In this section, we describe a coordinate descent algorithm to solve bound constrained quadratic programming problems of the form

$$\begin{aligned} &\min\ d(u) = \frac{1}{2}u^T AH^{-1}A^T u + u^T\hat{b} \\ &\text{s.t. } l \leq u \leq c, \end{aligned} \tag{6.3.1}$$

where $A \in \mathbb{R}^{m\times n}$, $\hat{b} \in \mathbb{R}^m$, $H = \alpha I - L\Sigma^{-1}L^T \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{2r}$, $l_i \in [-\infty, \infty)$ and $c_i \in (-\infty, \infty]$ with $l_i \leq c_i,\ i = 1, 2, \ldots, m$. Plugging (6.2.4) into the objective function of (6.3.1) gives

$$d(u) = \frac{1}{2\alpha}u^T AA^T u - \frac{1}{2\alpha}u^T(AL)(\alpha\Sigma + L^TL)^{-1}(AL)^T u + u^T\hat{b}$$

We assume $L \in \mathbb{R}^{n\times 2r}$ can fit into memory, and we can compute and store $AL \in \mathbb{R}^{m\times 2r}$ and $(AL)(\alpha\Sigma + L^TL)^{-1} \in \mathbb{R}^{m\times 2r}$. Defining $Q := AL$ and $\tilde{Q} := AL(\alpha\Sigma + L^TL)^{-1}$, (6.3.1)

becomes

$$\begin{aligned} \min\ & d(u) = \frac{1}{2\alpha} u^T A A^T u - \frac{1}{2\alpha} u^T \tilde{Q} Q^T u + u^T \hat{b} \\ \text{s.t. } & l \leq u \leq c \end{aligned} \tag{6.3.2}$$

Coordinate descent algorithm introduced in [45] is applicable to solve (6.3.1). The algorithm in [45], see Algorithm 3, is designed to solve a problem like

$$\begin{aligned} \min_{u \in \mathbb{R}^m} \quad & f(u) := g(Mu) + w^T u \\ \text{s.t.} \quad & l \leq u \leq c \end{aligned} \tag{6.3.3}$$

Define $U := \{u |\ l \leq u \leq c\}$ and

$$\text{proj}(\nabla f(u)) := \begin{cases} \nabla_i f(u) & l_i < u_i < c_i \\ \min(\nabla_i f(u), 0) & u_i = l_i \\ \max(\nabla_i f(u), 0) & u_i = c_i, \end{cases}$$

where $\nabla_i f(x)$ denotes the i-th component of the gradient of $f(x)$.

Algorithm 3 Coordinate Descent Algorithm (CDA)

Step 1: Initialize u^0 and ε.

Step 2: While $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty > \varepsilon$

For $i = 1, \ldots, m$,

$$u_i^{k+1} = \underset{l_i \leq u_i \leq c_i}{\text{argmin}}\, f(u_1^{k+1}, \ldots, u_{i-1}^{k+1}, u_i, u_{i+1}^k, \ldots, u_n^k) \tag{6.3.4}$$

In order to prove the convergence of Algorithm 3, [45] imposed the following assumptions on (6.3.3)

(a) The set of optimal solutions for (6.3.3) denoted by $\mathcal{U}^*$ is non-empty.

(b) g has a open domain and is strictly convex twice continuously differentiable.

(c) $\nabla^2 g(Mu^*)$ is positive definite for all $u^* \in \mathcal{U}^*$.

Our problem (6.3.1) satisfies these assumptions by a direct verification. Defining

$$g(u) = \frac{1}{2}u^T H^{-1}u, \quad M = A^T, \quad w = \hat{b},$$

assumptions (a), (b), (c) can be trivially verified. Theorem 2.1 in [45] shows that under the above assumptions, at least a linear convergence can be guaranteed.

The main computation of CDA comes from solving sub-problem (6.3.4). Next, let us see how we can solve sub-problem (6.3.4) efficiently. This sub-problem minimizes the objective over one coordinate, for ease of presentation, we drop the sup index k and $k+1$ and assume u_i is the coordinate to be minimized over while fixing all the other coordinates at their current values. The i-th component of the gradient of $d(u)$ is

$$\nabla_i d(u) = \frac{1}{\alpha}\sum_{j=1}^{m}(a_i^T a_j - \tilde{q}_i^T q_j)u_j + \hat{b}_i, \tag{6.3.5}$$

where a_i^T, q_i^T and $\tilde{q}_i^T$ denotes the i-th row of A, Q, and $\tilde{Q}$ respectively. Then the solution to (6.3.4) is given by

$$u_i = \begin{cases} l_i & \text{if } a_i^T a_i - \tilde{q}_i^T q_i = 0 \text{ and } \nabla_i d(u) > 0 \\ [l_i, c_i] & \text{if } a_i^T a_i - \tilde{q}_i^T q_i = 0 \text{ and } \nabla_i d(u) = 0 \\ c_i & \text{if } a_i^T a_i - \tilde{q}_i^T q_i = 0 \text{ and } \nabla_i d(u) < 0 \\ \min(\max(\frac{\sum_{j\neq i}(a_i^T a_j - \tilde{q}_i^T q_j)u_j + \alpha\hat{b}_i}{-(a_i^T a_i - \tilde{q}_i^T q_i)}, l_i), c_i) & \text{if } a_i^T a_i - \tilde{q}_i^T q_i \neq 0. \end{cases} \tag{6.3.6}$$

Since $a_i^T a_i - \tilde{q}_i^T q_i$ is the i-th diagonal element of $AA^T - \tilde{Q}Q^T$, we can pre-compute the diagonal of $AA^T - \tilde{Q}Q^T$ and store it. The solution mapping (6.3.6) shows that the key computation is $\nabla_i d(u)$. Direct calculation of $\nabla_i d(u)$ takes $O(m(\bar{n}+r))$ operations, where $\bar{n}$ is the average number of non-zeros in the A's rows. If m is large, then so is $O(m(\bar{n}+r))$ and the computation is expensive. To avoid the expensive of this calculation, we use a key observation from [40] that reduces the cost of the update of $\nabla_i d(u)$ to $O(\bar{n}+r)$. The update technique takes advantage of the structure of $AA^T - \tilde{Q}Q^T$. Assume u^0 is the initial point for CDA. Let $w = \sum_{j=1}^{m} u_j^0 a_j$ and $v = \sum_{j=1}^{m} u_j^0 q_j$. With the track of w and v, the calculation

of $\nabla_i d(u)$ can be carried out as

$$\nabla_i d(u) = \frac{1}{\alpha}(a_i^T w - \tilde{q}_i^T v) + \hat{b}_i. \tag{6.3.7}$$

Besides updating u_i, we keep updating w and v as well. Suppose we change u_i^k to u_i^{k+1}, then we set $w = w + a_i(u_i^{k+1} - u_i^k)$ and $v = v + q_i(u_i^{k+1} - u_i^k)$. The computation in (6.3.7) only involves the inner product of two vectors whose complexity is $O(\bar{n} + k)$. Incorporating the update computation of w and v, the total complexity is still $O(\bar{n} + k)$ which is much cheaper than $O(m(\bar{n} + k))$.

6.4 LBFGS for Linearly Constrained Non-linear Programming

Now we are ready to introduce Algorithm 4 for (6.1.1).

Algorithm 4 LBFGS for LNP

Step 0: Initialize $\alpha^0 > 0$, tolerance constants $\varepsilon_1 > 0$, $\varepsilon_2 > 0$, maximum iteration number N and we keep r pairs of $\{s^i, y^i\}$ for limited memory BFGS.

Step 1: Initial feasible point. Use CDA to solve (6.2.5) with $H_k = \alpha^0 I$ and $b + AH_k^{-1}\nabla f(x^k) = 0$. Let λ^0 be the solution. Set $x^1 := -\frac{1}{\alpha^0}\lambda^0$. x^1 is the first feasible point. Set $k = 1$ and $H_1 = \alpha^0 I$.

Step 2: While $k < N$:

(a) Solve (6.2.5) with x^k and tolerance ε_2 to obtain λ^k, then $d^k := -H_k^{-1}(A^T\lambda^k + \nabla f(x^k))$.

(b) Check stopping criteria: if $\left\|\nabla f(x^k) + A^T\lambda^k\right\|_\infty \leq \varepsilon_1$, stop.

(c) Line search along d^k to find step length t_k satisfying Wolfe condition

$$\begin{aligned} f(x^k + td^k) &\leq f(x^k) + tc_1\nabla f(x^k)^T d^k \\ \nabla f(x^k + td^k)^T d^k &\geq c_2\nabla f(x^k)^T d^k, \end{aligned} \tag{6.4.1}$$

where $c_1 := 10^{-4}$ and $c_2 := 0.9$.

(d) Set $x^{k+1} = x^k + t_k d^k$.

(e) Set $s^k = t_k d^k$, $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$, $\alpha^{k+1} := \frac{y^{kT}y^k}{s^{kT}y^k}$ and $H_0 = \alpha^{k+1} I$. Then update H_k to H_{k+1} based on Theorem 6.2.1 with limited memory BFGS constant r.

Remark 6.4.1. 1. Unit step length is always initially tested for acceptability for two reasons. First, $x^k + d^k$ is always feasible, and secondly, it is always the case that $f(x^k + d^k) \leq f(x^k)$ with equality holds only if $d^k = 0$. If the weak Wolfe condition (6.4.1) is not satisfied for unit step length, backtracking line search is used to find a step length t_k. We observe that in the most iterations, a unit stepsize satisfies the Wolfe condition (6.4.1).

2. In practice, stopping criteria $\left\|\nabla f(x^k) + A^T\lambda^k\right\|_\infty$ may be hard to achieve in some cases. In our implementation, we also terminate if either a non-descent direction is generated or an insufficient decrease of objective $f(x)$ occurs in 4 successive iterations.

3. The great advantage in using CDA to solve (6.2.5) lies in its ability to take full advantage of the sparse structure of the constraint matrix A. Although other methods mentioned above could also benefit from the sparse structure of A, their performance will rely on sparse linear algebra techniques. Moreover, factorization is memory intensive, so memory limits curtail the ability to solve very large scale problem.

6.5 Numerical Experiments

In this section, we present five numerical experiments. The 1st and 2nd experiments solve (6.2.5) for varying sizes and different structures of A. The 3rd experiment is the entropy maximization problem taken from the CVXOPT package. The 4th experiment is a simulated linearly constrained least square problems, while the 5th experiment contains a set of simulated very large scale linearly constrained least square problems which are out of the reach of CVXOPT. The 3rd and 4th experiments contain a comparison with CVXOPT conic program and quadratic program solvers. We see a clear advantage of our proposed algorithm in terms of running time.

For the simulated study, a random normal matrix A from $N(\mu, \sigma)$ with density ρ means the elements of A are sampled independently from $N(\mu, \sigma)$ and the ratio between the number non-zero elements of A and the total number of elements of A is ρ.

All the experiments were run on MacBook Air with a 1.4 GHz Intel Core i5 processor and 4GB RAM.

First Experiment: In this experiment, we randomly generate 6 problems with increasing dimensions. Set $m = 5000, 10000, 50000, 100000, 300000, 500000$, $s = \frac{m}{2}$, $k = 20$, $n = 2m$, $\alpha = 10.0$, $\mu = 100.0$, $\rho = 10^{-2}, 10^{-2}, 5 \times 10^{-4}, 2 \times 10^{-4}, 3 \times 10^{-5}, 3 \times 10^{-5}$. A is a random normal matrix A from $N(10, 20)$ with density ρ. L is a random normal matrix from $N(5, 20)$ with density 1. Σ is a diagonal matrix with diagonal elements $i.i.d \sim$ gamma $(0.5, 1)$. b is sampled i.i.d from $N(0, 600)$ while g is sampled i.i.d from $N(0, 300)$. We use CDA to

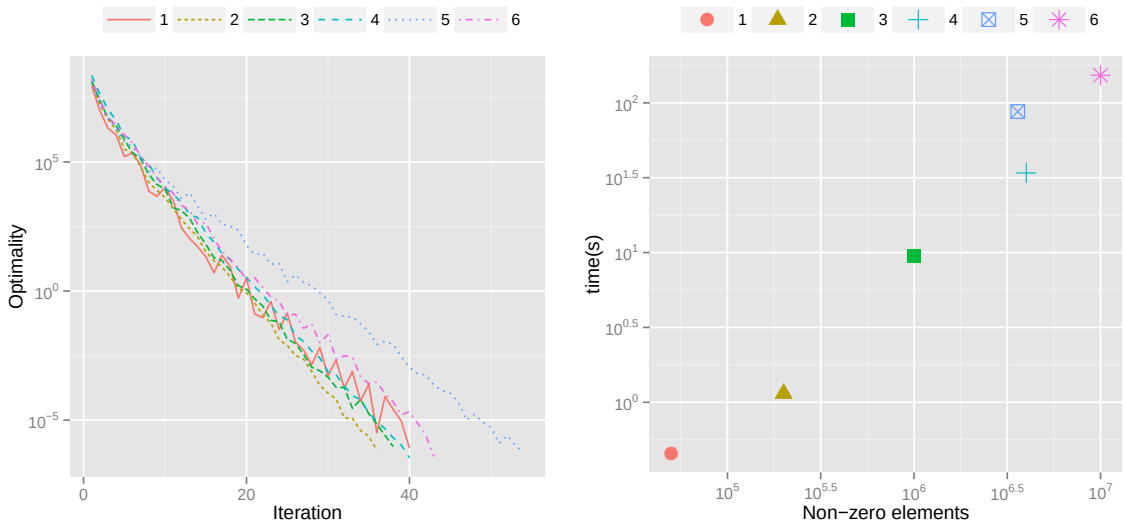

Figure 6.1: The left panel shows the trajectory of $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty$. For all the 6 problems, within less than 40 iterations, $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty$ reaches the required stopping tolerance $\epsilon = 10^{-6}$. The right panel presents the elapsed time of all 6 problems. It takes less than 1 minute to solve the largest problem.

solve these 6 problems with tolerance $\epsilon = 10^{-6}$. Figure 6.1 presents the trajectory of $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty$ and the elapsed time for all 6 problems. For all the 6 problems, within less than 40 iterations, $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty$ reaches the required tolerance $\epsilon = 10^{-6}$. The right panel presents the elapsed time of all 6 problems. It takes less than 1 minute to solve the largest problem.

Second Experiment: Set $m = 3000, 30000, 90000, 300000, 600000$, $n = m$ and $s = m$. Set

$$A = \begin{bmatrix} B & -I & 0 \\ -I & B & -I \\ 0 & -I & B \end{bmatrix},$$

where $B \in \mathbb{R}^{\frac{m}{3} \times \frac{m}{3}}$ with 4 on the main diagonal and -1 on two sub-diagonal, and $I \in \mathbb{R}^{\frac{m}{3} \times \frac{m}{3}}$ is the identity matrix. L, b, g, α, μ, k and ϵ are the same as in the first experiment.

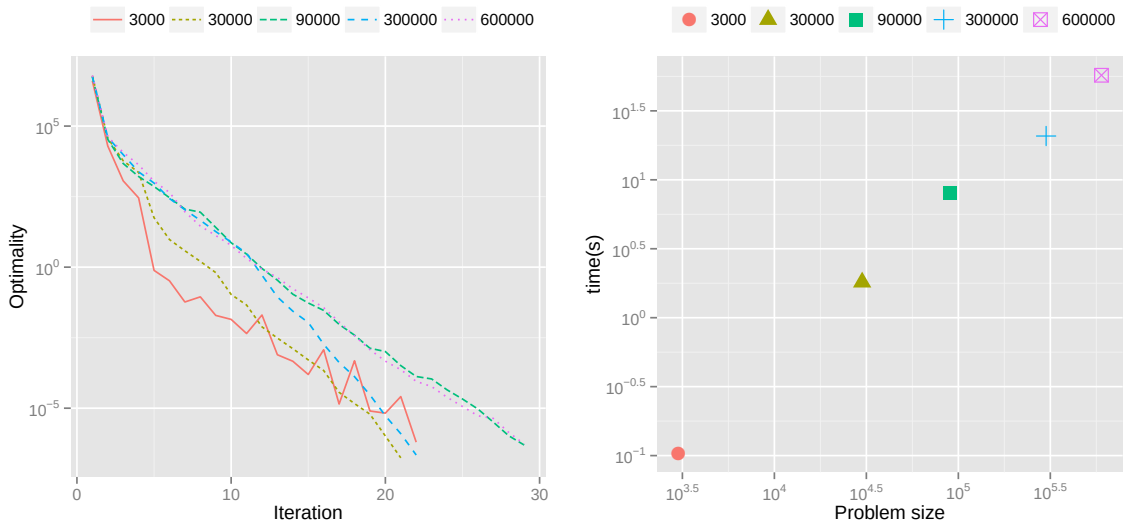

Figure 6.2: The left panel shows the trajectory of $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty$. For all the 5 problems, within less than 30 iterations, $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty$ reaches the required tolerance $\epsilon = 10^{-6}$. The right panel presents the elapsed time of all 5 problems. It takes less than 1 minute to solve the largest problem.

Figure 6.2 presents the trajectory of $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty$ and the elapsed time for all 5 problems. For all the 5 problems, within less than 30 iterations, $\left\|\text{proj}(\nabla f(u^k))\right\|_\infty$ reaches the required stopping tolerance $\epsilon = 10^{-6}$. The right panel presents the elapsed time of all 5 problems. It takes less than 1 minute to solve the largest problem.

Third Experiment (Entropy Maximization): Linearly constrained entropy maximization in [6] is defined as

$$\begin{aligned} \min_x \quad & \sum_{i=1}^{n} x_i \log(x_i) \\ \text{s.t.} \quad & Ex = e \\ & Gx \leq p, \end{aligned}$$

where the constraints are usually moment constraints for the distribution. We are go-

Table 6.1: Comparison of $f(x)$ and elapsed time

n	$f(x)$		Time(s)		$\max(\|Ex-e\|_\infty, (Gx-p)_+)$	
	LBFGS	CVXOPT	LBFGS	CVXOPT	LBFGS	CVXOPT
100	-4.386294	-4.386294	0.05	0.11	10^{-10}	10^{-9}
1000	-6.671064	-6.671064	0.29	0.83	10^{-10}	10^{-9}
2000	-7.363175	-7.363175	0.57	5.85	10^{-10}	10^{-9}
4000	-8.055801	-8.055801	2.52	45.71	10^{-10}	10^{-9}
6000	-8.461093	-8.461093	5.70	156.47	10^{-10}	10^{-9}

ing to solve one of the examples given in [1]. Let $a \in \mathbb{R}^n$ be made of n equidistant points in $[-1, 1]$. Define $h := \mathbf{0}_n$ and set $h_i = 1$ if $a_i < 0$. Set $E := \mathbf{1}_n^T$, $e := 1$, $p := [0.1, 0.1, 0.6, -0.5, -0.2, 0.3, 0.4, -0.3]^T$ and

$$A := [a, -a, a^2, -a^2, 3a^3 - 2a, -3a^3 + 2a, h, -h]^T,$$

where a^2 denotes the componentwise square of a.

We solve this problem using 'cp' solver in [1] and LBFGS for LNP with n =100, 1000, 2000, 4000, 6000. Parameters for 'cp' solver in [1] are all default except the absolute tolerance reset to 10^{-4}. We set $\alpha = 10^3$, $\varepsilon_1 = 10^{-2}$, $\varepsilon_2 = 10^{-10}$, $r = 10$ and $N = 1500$. If decrease of objective $f(x)$ is less than 10^{-5} 4 times in a row, we also terminate LBFGS for LNP. The results are summarized in Table 6.1. It shows that both algorithms converging to the same solution. LBFGS outperforms CVXOPT in all 5 problems in terms of running time. When $p = 6000$, LBFGS for LNP is 30 times faster than CVXOPT. Figure 6.3 shows the optimal solution comparison. Two solutions overlap with each other, i.e. both algorithms converge to the same solution.

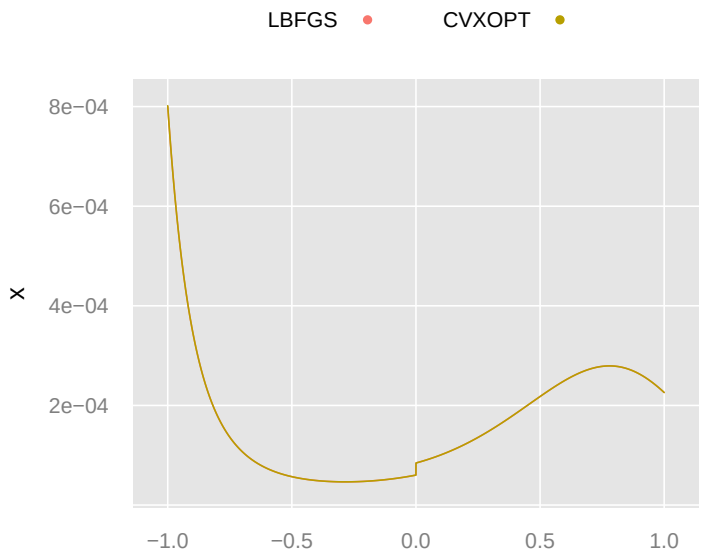

Figure 6.3: Optimal solution comparison

Fourth Experiment: Consider the following problem

$$\begin{aligned} \min_{x} \ & \frac{1}{2} \|Cx + g\|_2^2 \\ \text{s.t.} \ & Ex = e \\ & Gx \leq p. \end{aligned}$$

We generate 5 problems with number of variables $n = 2000, 3000, 4000, 5000, 6000$, and $\frac{n}{2}$ constraints with half equality and half inequality constraints. $C \in \mathbb{R}^{n \times n}$ is a random normal matrix from $N(5, 20)$ with density 0.2. E and G are random normal matrices from $N(2, 5)$ and $N(5, 10)$ respectively. Both E and G have a density 0.05. g is a random normal vector from $N(50, 200)$. To make sure the inequality constrains are always consistent, we generate random normal vectors z from $N(10, 50)$ and q from $N(10, 4)$ and let $p := Gz + q^T q$. e is a random vector from $N(5, 30)$. Set $\alpha = 10^3$, $\varepsilon_1 = 10^{-2}$, $\varepsilon_2 = 10^{-8}$, $r = 10$ and $N = 1500$. The condition number of C is approximately 69072.0, 89450.0, 166783.0, 66634.0 and 95756.0. CVXOPT qp solver takes all default parameters. Table 6.2 shows the comparison.

Table 6.2: Least Square Comparison of LBFGS and CVXOPT

n	$\frac{1}{2}\|Cx+g\|_2^2$		Time(s)		$\max(\|Ex-e\|_\infty, (Gx-p)_+)$	
	LBFGS	CVX	LBFGS	CVX	LBFGS	CVX
2000	3555229505.17	3555229505.52	5.04	6.66	10^{-10}	10^{-11}
3000	5961872832.29	5961872845.93	12.94	20.13	10^{-9}	10^{-12}
4000	3919393694.13	3919393694.25	24.56	48.40	10^{-11}	10^{-12}
5000	2073100245.36	2073100246.09	41.39	90.10	10^{-11}	10^{-12}
6000	6538672026.86	6538672026.94	62.24	159.84	10^{-11}	10^{-12}

Table 6.3: Over-determined Least Square Comparison of LBFGS and CVXOPT

n	$\frac{1}{2}\|Cx+g\|_2^2$		Time(s)		$\max(\|Ex-e\|_\infty, (Gx-p)_+)$	
	LBFGS	CVX	LBFGS	CVX	LBFGS	CVX
2000	358408646467.03	358408646468.23	1.29	10.27	10^{-10}	10^{-12}
3000	694034599282.74	694034599337.79	3.02	33.09	10^{-11}	10^{-12}
4000	1196359539210.19	1196359539230.58	4.82	77.82	10^{-11}	10^{-12}
5000	1595675771435.67	1595675771459.56	9.97	131.90	10^{-11}	10^{-12}
6000	2218234061219.69	2218234061220.45	22.36	254.30	10^{-11}	10^{-12}

LBFGS converges to a better objective function with less CPU time. However in terms of constraint violation, CVXOPT has smaller constraint violation.

We also test the case when $Cx + g = 0$ is an over-determined system, i.e. C has more rows than columns. Let $n = 2000, 3000, 4000, 5000, 6000$, $C \in \mathbb{R}^{(10n)\times n}$. The only difference is the dimension of C and g. The rest of this set of test problems is the same as previous one. Table 6.3 shows the comparison between LBFGS and CVXOPT. LBFGS is around 10 times faster than CVXOPT, and it is much more memory efficient.

Fifth Experiment: In this experiment, we try to scale up the scale of constrained least square as large as possible. We simulated 5 problems. Let n = 100000, 200000, 400000, 800000, 1600000 and ρ = 0.001, 0.001, 0.0001, 0.0001, 0.00001. And $C \in \mathbb{R}^{\frac{n}{20}\times n}$, $E \in \mathbb{R}^{\frac{n}{20}\times n}$

Table 6.4: Large scale test of LBFGS

n	Time(s)	$\left\|\nabla f(x) + A^T\lambda\right\|_\infty$	$\max(\|Ex-e\|_\infty, (Gx-p)_+)$
100000	15.66	0.008	10^{-8}
200000	67.25	0.009	10^{-8}
400000	122.87	0.022	10^{-8}
800000	558.619	0.012	10^{-8}
1600000	2301.06	0.074	10^{-8}

and $G \in \mathbb{R}^{\frac{n}{10}\times n}$. C is a random normal matrix with density ρ from $N(5,20)$. E is a random normal matrix with density ρ from $N(2,5)$. G is a random normal matrix with density ρ from $N(5,10)$. g is a random normal vector from $N(50,400)$, e is a random normal vector from $N(5,100)$. To make sure the inequality constraints is consistent, we sample two random normal vectors u from $N(3,10)$ and v from $N(3,5)$ and let $p := Au + v^2$. Set $\alpha = 10^5$, $\varepsilon_1 = 10^{-4}$, $\varepsilon_2 = 10^{-8}$, $r = 10$ and $N = 1500$. We only report the result for LBFGS, since CVXOPT does not work for this set of problems due to memory issue. Table 6.4 presents the results. It takes less than 1 hours to solve a problem of more than 1.5 million variables and half a million constraints.

Chapter 7

A DYNAMIC PENALTY PARAMETER UPDATING STRATEGY

7.1 A Penalty-SQP Framework

In this section, we formulate our problem of interest and outline the basic components of a penalty-SQP algorithm [26]. This method represents the framework in which we will define our dynamic penalty parameter updating strategy and matrix-free solvers whose purposes, respectively, are to guide the algorithm toward promising areas of the search space and approximately solve the direction-finding subproblems.

We formulate our problem of interest as the following nonlinear optimization problem with equality and inequality constraints for which we assume that the functions $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are continuously differentiable:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t. } & c_i(x) = 0, \ i \in \{1, \ldots, \overline{m}\} \\ & c_i(x) \leq 0, \ i \in \{\overline{m}+1, \ldots, m\}. \end{aligned} \tag{NLP}$$

For our penalty-SQP framework, we define two functions used in the implementation of the algorithm and for characterizing first-order stationary points. The first is our measure of infeasibility

$$v(x) := \sum_{i=1}^{\overline{m}} |c_i(x) \ + \sum_{i=\overline{m}+1}^{m} (c_i(x))_+$$

and the second is the associated exact penalty function

$$\phi(x; \rho) := \rho f(x) + v(x),$$

where $\rho \in \mathbb{R}_+$ is the penalty parameter. Generally speaking, our penalty-SQP framework aims to solve (NLP) through systematic minimization of $\phi(x; \rho)$ for appropriately chosen values of $\rho > 0$. However, if the constraints of (NLP) are infeasible, then the algorithm is designed to return an infeasibility certificate in the form of a stationary point for $\phi(x; 0) =$

$v(x)$. Second, we define the Fritz John function for (NLP) which, given $\rho \in \mathbb{R}_+$ and $\eta \in \mathbb{R}^m$, is

$$F(x; \rho, \eta) = \rho f(x) + \langle \eta, c(x) \rangle.$$

We remark that we have defined $\rho \in \mathbb{R}_+$ as having the dual role of penalty parameter in ϕ and objective multiplier in F. In fact, this makes sense from both theoretical and practical perspectives. First-order stationarity conditions for (NLP) can be written in terms of ∇F, the constraint function c, and bounds on the dual variables; see [11].

In the k-th iteration of our penalty-SQP framework, the search direction computation is based on a local model of the penalty function about the current primal iterate $x^k \in \mathbb{R}^n$ that makes use of the dual iterate $\eta^k \in \mathbb{R}^m$. We define this model as

$$\begin{aligned} J(d; x^k, \rho, \eta^k) := \rho \nabla f(x^k)^T d + \frac{1}{2} d^T H(x^k, \rho, \eta^k) d \\ + \sum_{i=1}^{\overline{m}} |c_i(x^k) + \nabla c_i(x^k)^T d| + \sum_{i=\overline{m}+1}^{m} \max\{c_i(x^k) + \nabla c_i(x^k)^T d, 0\}, \end{aligned}$$

where H represents an approximation of $\nabla^2_{xx} F$ so that

$$H(x^k, \rho, \eta^k) \approx \nabla^2_{xx} F(x^k, \rho, \eta^k) = \rho \nabla^2_{xx} f(x^k) + \sum_{i=1}^{m} \eta_i^k \nabla^2_{xx} c_i(x^k).$$

Ultimately, the search direction computed in the k-th iteration represents, for some $X \subset \mathbb{R}^n$ containing $\{0\}$ and penalty parameter $\rho^k > 0$, an approximate solution of the subproblem

$$\min_{d \in X} \; J(d; x^k, \rho^k, \eta^k). \tag{QP}$$

(We introduce the set X to allow for the possibility of employing, say, a trust region constraint; e.g., for some $\Delta > 0$, we may define X such that $X \subset \{d : \|d\|_2 \leq \Delta\}$). In fact, the value $\rho^k \in (0, \rho^{k-1}]$ is to be computed *during* the subproblem solve in order to satisfy two critical inequalities. First, as is typical in the context of a penalty-SQP method, we intend for the pair (d^k, ρ^k) to be computed such that the search direction d^k is a direction of sufficient descent for $\phi(\cdot; \rho^k)$ from x^k. This is guaranteed if the reduction in the local model $J(\cdot; x^k, \rho^k, \eta^k)$ yielded by d^k, namely,

$$\Delta J(d; x^k, \rho^k, \eta^k) = J(d^k; x^k, \rho^k, \eta^k) - J(0; x^k, \rho^k, \eta^k),$$

is sufficiently negative, and consequently steers the exact penalty function with penalty parameter ρ^k in a descent direction. This requirement—which we formulate concretely in §7.2.2—represents the first critical inequality satisfied by our search direction. The second critical inequality that we impose is similar, except that it relates to the reduction yielded by d^k in the local model $J(\cdot; x^k, 0, \eta^k)$ along d^k for feasibility alone, namely,

$$\Delta J(d; x^k, 0, \eta^k) = J(d; x^k, 0, \eta^k) - J(0; x^k, 0, \eta^k).$$

The dynamic strategy for setting ρ^k is designed so that this reduction is also sufficiently negative at the end of the search direction computation, implying that d^k is also a direction of sufficient descent from x^k for $\phi(\cdot; 0) = v$. It is important to note that since both v and ϕ are convex-composite function [7, 8, 13], we have

$$\phi'(x^k; d^k) \leq \Delta J(d; x^k, \rho^k, \eta^k) \quad \text{and} \quad v'(x^k; d^k) \leq \Delta J(d; x^k, 0, \eta^k).$$

Overall, the k-th iteration of our penalty-SQP strategy proceeds in the following manner. First, a search direction and penalty parameter pair (d^k, ρ^k) is computed by a subproblem solver such that the resulting search direction yields reductions in the models of the penalty function and measure of infeasibility that are sufficiently large. Then—potentially after additional updates of the penalty parameter—a line search may be performed for the merit function $\phi(\cdot; \rho^k)$ from x^k along the search direction d^k, yielding the step-size $\alpha^k \in \mathbb{R}_{++}$. Finally, the new iterate is set as $x^{k+1} \leftarrow x^k + \alpha^k d^k$ and the algorithm proceeds to the $(k+1)$-st iteration. We formally state this framework as Algorithm 5 below.

Algorithm 5 A Framework of *S*equential *Q*uadratic Optimization Algorithm

1. (Initialization) Choose $\gamma, \theta_l \in (0, 1)$. Set $k \leftarrow 0$ and choose (x^k, ρ^k, η^k).

2. (Subproblem Solution) Solve (QP) to obtain a primal-dual solution (d^k, η^k) and the penalty parameter ρ^k.

3. (Line Search) Let α^k be the largest value in $\{\gamma^0, \gamma^1, \gamma^2, \dots\}$ such that

$$\phi(x^k + \alpha^k d^k, \rho^k) - \phi(x^k, \rho^k) \leq -\theta_l \alpha^k \Delta J(d; x^k, \rho^k, \eta^k).$$

4. Set $x^{k+1} \leftarrow x^k + \alpha^k d^k$ and $k \leftarrow k + 1$ and go to step 1.

7.2 A Dynamic Penalty Parameter Updating Strategy

In this section, we present our new dynamic penalty parameter updating strategy. Our method is unique in that it is intended to be employed within a solver for the subproblem arising in our penalty-SQP framework. A potential pitfall of such an approach is that, since the penalty parameter dictates the weight between the objective terms in (QP), one may disrupt typical convergence guarantees of the subproblem solver by manipulating this weight during the solution process. However, under reasonable assumptions, we prove that for sufficiently small values of the penalty parameter, our updating strategy will no longer be triggered. Consequently, once the penalty parameter reaches a sufficiently small value, it will remain fixed and the subproblem solver will effectively be applied to solve (QP) for a fixed value of ρ^k. We state our proposed updating strategy in such a way that it can be incorporated into various subproblem solvers; see §7.3.

7.2.1 Preliminaries

As our penalty parameter updating strategy is to be employed in each iteration of our penalty-SQP framework, we can present our strategy generically by focusing on the k-th iteration of the framework. Thus, for ease of exposition in this section, we utilize the following shorthand notation to drop the dependence of certain quantities on the iteration number:

$$\begin{aligned} &g = \nabla f(x^k),\ a_i = \nabla c_i(x^k),\ b_i = c_i(x^k),\ A = [a_1, \cdots, a_m]^T, \\ &H_f \approx \nabla^2_{xx} f(x^k),\ H_0 \approx \sum_{i=1}^{m} \eta_i^k \nabla^2_{xx} c_i(x^k),\ H_\rho = \rho H_f + H_0. \end{aligned} \tag{7.2.1}$$

We also make the following assumption about the subproblem data.

Assumption 7.2.1. *The data matrices A, H_f, and H_0 for the subproblem are such that*

(i) H_ρ is positive definite for any $\rho \in [0, \rho^{k-1}]$; and

(ii) $\|a_i\|_2 > 0$ for all $i \in \{1, \ldots, m\}$.

We claim that this assumption is reasonable due to the following considerations. First, in large-scale contexts, it is typically impractical or inefficient to construct complete second-

derivative matrices. Hence, as indicated in (7.2.1), we assume that H_f and H_0 represent approximate (low-rank) Hessian matrices with at least H_0 being positive definite. Second, if $a_i = 0$ for any $i \in \{1, \ldots, m\}$, then the i-th constraint in the subproblem is superfluous and can be removed from consideration. The zero rows of A can be detected during a preprocessing phase for the subproblem, so for simplicity in our discussion we assume that each constraint gradient is nonzero. As such, for notational convenience, we define the scaled quantities $\bar{a}_i := a_i/\|a_i\|_2$ and $\bar{b}_i := b_i/\|a_i\|_2$ for all $i \in \{1, \ldots, m\}$.

Of central importance in the subproblems are the convex sets

$$C_i := \{d \in \mathbb{R}^n : a_i^T d + b_i = 0\}, \quad i \in \{1, \ldots, \overline{m}\}$$
$$\text{and} \quad C_i := \{d \in \mathbb{R}^n : a_i^T d + b_i \leq 0\}, \quad i \in \{\overline{m}+1, \ldots, m\}.$$

The penalty term in the model J can thus be written as

$$\sum_{i=1}^{m} \|a_i\|_2 \, \mathrm{dist}_2\left(d \mid C_i\right),$$

meaning that, without loss of generality (i.e., assuming $\|a_i\|_2 = 1$ for all $i \in \{1, \ldots, m\}$) we may rewrite (in shorthand) the penalty-SQP subproblem (QP) as

$$\min_{d \in \mathbb{R}^n} \; J(d;\rho), \quad \text{where} \quad \begin{cases} J(d;\rho) = \varphi(d;\rho) + \displaystyle\sum_{i=1}^{m} \mathrm{dist}_2\left(d \mid C_i\right) + \delta(d|X) \\ \varphi(d;\rho) = \rho g^T d + \dfrac{1}{2} d^T H_\rho d. \end{cases} \tag{QP$_\rho$}$$

We refer to (QP$_\rho$) with $\rho > 0$ as a *penalty subproblem*, whereas we refer to (QP$_\rho$) with $\rho = 0$ as the *feasibility subproblem*. Direct calculation shows the Fenchel–Rockafellar dual of subproblem (QP$_\rho$) is given by

$$\max_{\mathbf{u} \in \mathbb{R}^n \times \cdots \times \mathbb{R}^n} \; D(\mathbf{u};\rho) \quad \text{s.t.} \quad u_{m+1} = \sum_{i=0}^{m} u_i \quad \text{and} \quad u_i \in \mathbb{B}_2 \quad \text{for all} \quad i \in \{1, \ldots, m\}, \tag{DQP$_\rho$}$$

where the dual objective function is given by

$$D(\mathbf{u};\rho) = -\frac{1}{2}(u_0 - \rho g)^T H_\rho^{-1}(u_0 - \rho g) - \sum_{i=1}^{m} \delta^*(u_i|C_i) - \delta^*(u_{m+1}|X).$$

An interesting aspect of the dual subproblem (DQP$_\rho$) is that the penalty parameter appears only in the objective function; thus, if $\mathbf{u}$ satisfies the constraints of (DQP$_\rho$), then it is dual-feasible regardless of the value of ρ appearing in the subproblem. As a result, by weak

duality, we have for any primal-dual feasible pair $(d, \mathbf{u})$ that both

$$D(\mathbf{u}; 0) \leq J(d; 0) \quad \text{and} \quad D(\mathbf{u}; \rho) \leq J(d; \rho). \tag{7.2.2}$$

We close this subsection by noting that projections onto the set C_i for any $i \in \{1, \ldots, m\}$ are especially easy to compute; in particular,

$$\begin{aligned} P_{C_i}(x) &= x - (\bar{a}_i^T x + \bar{b}_i)\bar{a}_i, \qquad i \in \{1, \ldots, \overline{m}\} \\ \text{and} \quad P_{C_i}(x) &= x - (\bar{a}_i^T x + \bar{b}_i)_+ \bar{a}_i, \quad i \in \{\overline{m}+1, \ldots, m\}. \end{aligned}$$

Then

$$\begin{aligned} J(d; \rho) &= \varphi(d; \rho) + \sum_{i=1}^{\bar{m}} |\bar{a}_i^T d + \bar{b}_i| \ + \sum_{i=\bar{m}+1}^{m} (\bar{a}_i^T d + \bar{b}_i)_+ + \delta(d|X) \\ &= \varphi(d; \rho) + \operatorname{dist}_1 \left(\bar{A}d + \bar{b} \,\middle|\, \{0\}^{\bar{m}} \times \mathbb{R}_-^{m-\bar{m}} \right) + \delta(d|X). \end{aligned} \tag{7.2.3}$$

We also observe that if $X = \mathbb{R}^n$, then the Fenchel-Rockafellar dual of (QP_ρ) reduces to

$$\max_{\mathbf{u} \in \mathbb{R}^n \times \cdots \times \mathbb{R}^n} \ D(\mathbf{u}; \rho) \quad \text{s.t.} \quad 0 = \sum_{i=0}^{m} u_i \quad \text{and} \quad u_i \in \mathbb{B}_2 \quad \text{for all} \quad i \in \{1, \ldots, m\},$$

where

$$D(\mathbf{u}; \rho) = -\frac{1}{2}(u_0 - \rho g)^T H_\rho^{-1}(u_0 - \rho g) - \sum_{i=1}^{m} \delta^*(u_i | C_i).$$

7.2.2 Updating ρ

We now present our dynamic penalty parameter updating strategy. For a given $\rho > 0$, let $(d_\rho^*, \mathbf{u}_\rho^*)$ represent an optimal primal-dual pair for the penalty subproblem corresponding to ρ; in particular, $(d_0^*, \mathbf{u}_0^*)$ represents an optimal primal-dual pair for the feasibility subproblem. We present our algorithm in the context of a subproblem solver that generates two sequences of iterates: The first sequence of iterates, call it $\{(d^{(j)}, \mathbf{u}^{(j)})\}$, represents a sequence of primal-dual feasible solution estimates for a penalty subproblem, while the second sequence of iterates, call it $\{\mathbf{w}^{(j)}\}$, represents a sequence of dual feasible solution estimates for the feasibility subproblem. (In our strategy, we do not make separate use of a sequence of primal solution estimates for the feasibility subproblem; rather, the sequence $\{d^{(j)}\}$ plays this role as well.) Without loss of generality, we assume that the j-th dual solution estimate $\mathbf{w}^{(j)}$ represents a better (or no worse) dual solution for the feasibility subproblem than $\mathbf{u}^{(j)}$

in the sense that $D(\mathbf{w}^{(j)};0) \geq D(\mathbf{u}^{(j)};0)$. This is a reasonable assumption since if this inequality were not to hold, then one could simply replace $\mathbf{w}^{(j)}$ with $\mathbf{u}^{(j)}$ as the j-th dual feasible solution estimate for the feasibility subproblem.

Observe that, by the definition of the model J, we have

$$J^{(0)} := J(0;0) = J(0;\rho) = \sum_{i=1}^{\bar{m}} |\bar{b}_i| + \sum_{i=\bar{m}+1}^{m} (\bar{b}_i)_+ \geq 0$$

for any $\rho > 0$. We then define, for any given value of the penalty parameter $\rho > 0$, the following ratios corresponding to the j-th subproblem solver iteration:

$$r_v^{(j)} := \frac{J^{(0)} - J(d^{(j)};0)}{J^{(0)} - D(\mathbf{w}^{(j)};0)} \quad \text{and} \quad r_\phi^{(j)} := \frac{J^{(0)} - J(d^{(j)};\rho^{(j)})}{J^{(0)} - D(\mathbf{u}^{(j)};\rho^{(j)})}. \tag{7.2.4}$$

The critical property of these ratios is that, if they are sufficiently large, then the corresponding subproblem solver iterate must yield a reduction in the penalty function model that is proportional to that yielded by an exact subproblem solution. In particular, suppose that for some prescribed constant $\beta_v \in (0,1)$ we have

$$r_v^{(j)} \geq \beta_v. \tag{R_v}$$

We may then observe that the reduction in the linearized constraint violation model $J(\cdot;0)$ (relative to a zero step) yielded by the subproblem solver iterate $d^{(j)}$ satisfies

$$\begin{aligned} J^{(0)} - J(d^{(j)};0) &\geq \beta_v(J^{(0)} - D(\mathbf{w}^{(j)};0)) \\ &\geq \beta_v(J^{(0)} - D(\mathbf{u}_0^*;0)) \geq \beta_v(J^{(0)} - J(d_0^*;0)), \end{aligned}$$

where the first inequality follows by (R_v), the second follows by optimality of $\mathbf{u}_0^*$ with respect to the dual of the feasibility subproblem, and the last follows by weak duality. Similarly, if for some prescribed constant $\beta_\phi \in (0,1)$ and $\rho > 0$ we have

$$r_\phi^{(j)} \geq \beta_\phi, \tag{R_ϕ}$$

then it follows that

$$\begin{aligned} J^{(0)} - J(d^{(j)};\rho^{(j)}) &\geq \beta_\phi(J^{(0)} - D(\mathbf{u}^{(j)};\rho^{(j)})) \\ &\geq \beta_\phi(J^{(0)} - D(\mathbf{u}^*_{\rho^{(j)}};\rho^{(j)})) \geq \beta_\phi(J^{(0)} - J(d^*_{\rho^{(j)}};\rho^{(j)})). \end{aligned} \tag{7.2.5}$$

Our penalty parameter strategy is motivated by the desire to ensure that if the j-th iterate of the subproblem solver represents a sufficiently accurate solution of the penalty subproblem for $\rho > 0$, then it should also represent a sufficiently accurate solution of the feasibility subproblem; if not, then the penalty parameter should be reduced. Specifically, choosing parameters

$$0 < \beta_v < \beta_\phi < 1, \tag{7.2.6}$$

we initialize $\rho^{(0)} \leftarrow \rho^{k-1}$ (from the preceding iteration of the penalty-SQP framework) and apply the subproblem solver to (QP_ρ) to initialize the sequence $\{(d^{(j)}, \mathbf{u}^{(j)}, \mathbf{w}^{(j)})\}$. If, at the start of the j-th subproblem solver iteration we have that (R_ϕ) is not satisfied, then we continue the iteration to solve (QP_ρ) for the current value of ρ. Otherwise, if (R_ϕ) is satisfied but (R_v) is not, then we reduce the penalty parameter by setting

$$\rho^{(j)} \leftarrow \theta_\rho \rho^{(j-1)} \tag{7.2.7}$$

for some prescribed constant $\theta_\rho \in (0, 1)$. (The remaining case is that (R_ϕ) and (R_v) are both satisfied, in which case we do not change the penalty parameter and may either terminate the subproblem solver or continue to compute a more accurate solution of (QP_ρ). The determination of whether to terminate or continue the subproblem solver should be made based on the demands of the penalty-SQP method.)

We summarize our *d*ynamic *u*pdating *st*rategy below:

For $(d^{(j)}, \mathbf{u}^{(j)}, \mathbf{w}^{(j)})$, if (R_ϕ) holds but (R_v) does not, then apply (7.2.7).	(DUST)

We close this subsection by making a few practical remarks regarding the use of (DUST) within a subproblem solver for (QP_ρ). In particular, while we have defined the sequence $\{(d^{(j)}, \mathbf{u}^{(j)}, \mathbf{w}^{(j)})\}$ as being generated by (a single run of) the solver, it may be reasonable to reinitialize the solver—or at least perform some auxiliary computations—after any iteration in which (7.2.7) is invoked. That being said, it is reasonable to assume that, during any sequence of iterations in which ρ does not change, the subproblem solver would be applied as if it were being applied to a (static) instance of (QP_ρ). In such a manner, any convergence guarantees for the subproblem solver would hold if/when the penalty parameter stabilizes at a fixed value, as is guaranteed to occur in certain situations described next.

7.2.3 Finite Updates

The purpose of this subsection is to show that if (DUST) is employed within an algorithm for solving (QP_ρ), then, under reasonable assumptions on the subproblem data, for any $\rho^{(j)} \in (0, \tilde{\rho}]$ for some sufficiently small $\tilde{\rho} > 0$ whose value depends only on the subproblem data, if (R_ϕ) is satisfied, then (R_v) is also satisfied. In other words, after finite iterations, the updating strategy (7.2.7) will never be triggered. Let $\underline{\lambda}_0$ and $\bar{\lambda}_0$ be the smallest and largest eigenvalues of H_0, and so forth for $\underline{\lambda}_\rho$ and $\bar{\lambda}_\rho$ for H_ρ. Notice that

$$\underline{\lambda}_{\rho^{(j)}} \geq \min(\underline{\lambda}_{\rho^{(0)}}, \underline{\lambda}_0) \quad \text{and} \quad \bar{\lambda}_{\rho^{(j)}} \leq \max(\bar{\lambda}_{\rho^{(0)}}, \bar{\lambda}_0). \tag{7.2.8}$$

We formalize our assumptions for this analysis as the following.

Assumption 7.2.2. *Let $\{(d^{(j)}, \mathbf{u}^{(j)}, \mathbf{w}^{(j)})\}$ be a sequence such that, for all $j \in \mathbb{N}$, the vectors $\mathbf{u}^{(j)}$ and $\mathbf{w}^{(j)}$ are feasible for* (DQP_ρ). *Moreover, let $\{\rho^{(j)}\}$ be a sequence generated along with $\{(d^{(j)}, \mathbf{u}^{(j)}, \mathbf{w}^{(j)})\}$ by applying* (DUST). *Then, corresponding to the set*

$$\mathcal{U} = \{j : (d^{(j)}, \mathbf{u}^{(j)}) \text{ satisfies } (R_\phi)\},$$

the subsequences $\{\|\mathbf{u}^{(j)}\|_2\}_{k\in\mathcal{U}}$ and $\{\|\mathbf{w}^{(j)}\|_2\}_{k\in\mathcal{U}}$ are bounded by a constant $\kappa_0 > 0$ independent of $\{\rho^{(j)}\}$.

The boundedness assumption on dual estimates are reasonable since our subproblems are assumed to be strictly convex. We can also easily show the primal variables $\{d^{(j)}\}_{j\in\mathcal{U}}$ are also bounded.

Lemma 7.2.3. *If Assumption 7.2.1 and 7.2.2 hold, then we know*

$$\|d^{(j)}\|_2 \leq \kappa_1 := \frac{\rho^{(0)}\|g\|_2 + \sqrt{(\rho^{(0)})^2\|g\|_2^2 + 4\max(\bar{\lambda}_{\rho^{(0)}}, \bar{\lambda}_0)J^{(0)}}}{2\min(\underline{\lambda}_{\rho^{(0)}}, \underline{\lambda}_0)}, \quad j \in \mathcal{U}. \tag{7.2.9}$$

Proof. By Assumption 7.2.2 (i), $\{d^{(j)}\}_{j\in\mathcal{U}} \subset X$ which implies $\delta(d^{(j)}|X) = 0$ for $j \in \mathcal{U}$. By (R_ϕ), every $(d^{(j)}, \mathbf{u}^{(j)}, \rho^{(j)})$ must satisfies (7.2.5), which implies

$$J(d^{(j)}; \rho^{(j)}) = \rho^{(j)} g^T d^{(j)} + \frac{1}{2}(d^{(j)})^T H_{\rho^{(j)}} d^{(j)} \leq J^{(0)}.$$

It follows that

$$\underline{\lambda}_{\rho^{(j)}}\|d^{(j)}\|_2^2 \le J^{(0)} + |\rho^{(j)} g^T d^{(j)}| \le J^{(0)} + \rho^{(0)}\|g\|_2\|d^{(j)}\|_2,$$

a quadratic polynomial inequality in $\|d^{(j)}\|_2$. Consequently,

$$\|d^{(j)}\|_2 \le \frac{\rho^{(0)}\|g\|_2 + \sqrt{(\rho^{(0)})^2\|g\|_2^2 + 4\underline{\lambda}_{\rho^{(j)}} J^{(0)}}}{2\underline{\lambda}_{\rho^{(j)}} J^{(0)}},$$

which, together with (7.2.8), proves (7.2.9). □

The next lemma shows the difference between the primal values of the optimality subproblem and the feasibility subproblem at iteration $j \in \mathcal{U}$ depends on ρ, and consequently, so does the difference between dual values.

Lemma 7.2.4. *Suppose Assumption 7.2.1 and 7.2.2 hold. Then there exists a constant $\kappa_3 >$, independent of $j \in \mathcal{U}$, such that*

$$|J(d^{(j)};\rho^{(j)}) - J(d^{(j)};0)| \le \kappa_2\rho^{(j)}, \quad \textit{and} \tag{7.2.10a}$$

$$|D(\mathbf{u}^{(j)};\rho^{(j)}) - D(\mathbf{w}^{(j)};0)| \le \kappa_3\rho^{(j)}, \tag{7.2.10b}$$

with

$$\kappa_2 = \tfrac{1}{2}\|H_f\|_2\kappa_1^2 + \|g\|_2\kappa_1,$$
$$\kappa_3 = \frac{\kappa_0 + \rho^{(0)}\|g\|_2}{\min(\underline{\lambda}_{\rho^{(0)}}, \underline{\lambda}_0)}(\|H_0^{-1}H_f\|_2 + \frac{1}{2}\|g\|_2) + \|H_0^{-1}H_f\|_2\|g\|_2.$$

Proof. For primal value, we have

$$\begin{aligned} |J(d^{(j)};\rho^{(j)}) - J(d^{(j)};0)| &= |\rho^{(j)} g^T d^{(j)} + \frac{1}{2}(d^{(j)})^T H_{\rho^{(j)}} d^{(j)} - \frac{1}{2}(d^{(j)})^T H_0 d^{(j)}| \\ &= |\rho^{(j)} g^T d^{(j)} + \tfrac{\rho^{(j)}}{2}(d^{(j)})^T H_f d^{(j)}| \\ &\le \rho^{(j)}(\|g\|_2 + \tfrac{1}{2}\|H_f\|_2\|d^{(j)}\|_2)\|d^{(j)}\|_2, \end{aligned} \tag{7.2.11}$$

which combined with Lemma 7.2.3 immediately proves (7.2.10a).

Let $\hat{y}^{(j)} = H_{\rho^{(j)}}^{-1}(u_0^{(j)} - \rho^{(j)} g)$ and $\bar{y}^{(j)} = H_0^{-1}u_0^{(j)}$. By Lemma 7.2.3, one can show

$$\|\hat{y}^{(j)}\|_2 \le \frac{\kappa_0 + \rho^{(j)}\|g\|_2}{\underline{\lambda}_{\rho^{(j)}}} \le \frac{\kappa_0 + \rho^{(0)}\|g\|_2}{\min(\underline{\lambda}_{\rho^{(0)}}, \underline{\lambda}_0)},$$

and $\{\|\bar{y}^{(j)}\|_2\}_{j\in\mathcal{U}}$ are all bounded by constant

$$\|\bar{y}^{(j)}\|_2 \leq \frac{\kappa_0}{\underline{\lambda}_0}.$$

It follows that

$$\rho^{(j)}g = u_0^{(j)} - (u_0^{(j)} - \rho^{(j)}g) = H_0\bar{y}^{(j)} - H_{\rho^{(j)}}\hat{y}^{(j)} = H_0(\bar{y}^{(j)} - \hat{y}^{(j)}) - \rho^{(j)}H_f\hat{y}^{(j)},$$

which implies for any $j \in \mathcal{U}$,

$$\begin{aligned}\|\bar{y}^{(j)} - \hat{y}^{(j)}\|_2 &= \|\rho^{(j)}H_0^{-1}H_f(\hat{y}^{(j)} + g)\|_2 \\ &\leq \rho^{(j)}\|H_0^{-1}H_f\|_2\|\hat{y}^{(j)} + g\|_2 \\ &\leq \rho^{(j)}\|H_0^{-1}H_f\|_2\left(\frac{\kappa_0 + \rho^{(0)}\|g\|_2}{\min(\underline{\lambda}_{\rho^{(0)}}, \underline{\lambda}_0)} + \|g\|_2\right).\end{aligned} \tag{7.2.12}$$

The difference between the dual values of optimality subproblem and feasibility subproblem is then given by

$$\begin{aligned}&|D(\mathbf{u}^{(j)};\rho^{(j)}) - D(\mathbf{u}^{(j)};0)| \\ &= |-\frac{1}{2}(u_0^{(j)} - \rho^{(j)}g)^T H_{\rho^{(j)}}^{-1}(u_0^{(j)} - \rho^{(j)}g) + \frac{1}{2}(u_0^{(j)})^T H_0^{-1}u_0^{(j)}| \\ &= |\frac{1}{2}(\bar{y}^{(j)} - \hat{y}^{(j)})^T u_0^{(j)} + \frac{1}{2}\rho^{(j)}g^T\hat{y}^{(j)}| \\ &\leq \frac{1}{2}\|\bar{y}^{(j)} - \hat{y}^{(j)}\|_2\|u_0^{(j)}\|_2 + \frac{1}{2}\rho^{(j)}\|g\|_2\|\hat{y}^{(j)}\|_2 \\ &\leq \rho^{(j)}\left(\|H_0^{-1}H_f\|_2\left(\frac{\kappa_0 + \rho^{(0)}\|g\|_2}{\min(\underline{\lambda}_{\rho^{(0)}}, \underline{\lambda}_0)} + \|g\|_2\right) + \frac{1}{2}\|g\|_2\frac{\kappa_0 + \rho^{(0)}\|g\|_2}{\min(\underline{\lambda}_{\rho^{(0)}}, \underline{\lambda}_0)}\right) \\ &= \rho^{(j)}\left(\frac{\kappa_0 + \rho^{(0)}\|g\|_2}{\min(\underline{\lambda}_{\rho^{(0)}}, \underline{\lambda}_0)}(\|H_0^{-1}H_f\|_2 + \frac{1}{2}\|g\|_2) + \|H_0^{-1}H_f\|_2\|g\|_2\right),\end{aligned}$$

where the last inequality is by (7.2.12) and Lemma 7.2.3. This completes the proof of (7.2.10b). □

Now we are ready to prove our main result.

Theorem 7.2.5. *Given $g, H_f, H_0, \rho^{(j)}$ and set C satisfying Assumption 7.2.1. Consider an algorithm for solving* (QP_ρ) *such that Assumption 7.2.2 is satisfied. Let*

$$\tilde{\rho}^{(j)} := \frac{1 - \sqrt{\beta_v/\beta_\phi}}{\kappa_3}(J^0 - D(\mathbf{u}^{(j)};0)) \quad \textit{and} \quad \tilde{\rho}^* := \frac{1 - \sqrt{\beta_v/\beta_\phi}}{\kappa_3}(J^0 - D(\mathbf{u}_0^*;0)).$$

For (QP_ρ) *with any* $\rho^{(j)} \in (0, \tilde{\rho}^{(j)})$*, if the primal-dual iterate* $\{(d^{(j)}, \mathbf{u}^{(j)})\}$ *generated by the algorithm satisfies condition* R_ϕ*, then the primal-dual iterates* $\{(d^{(j)}, \mathbf{w}^{(j)})\}$ *also satisfy* R_v*. In particular, since* $0 < \tilde{\rho}^{(j)} \leq \tilde{\rho}^*$ *for all* j*, for any* $\rho^{(j)} \in (0, \tilde{\rho}^*)$*, the* (DUST) *update to* $\rho^{(j)}$ *is never triggered.*

Proof. By Lemma 7.2.4, we can see that for problem (QP_ρ) with sufficiently small $\rho^{(j)} > 0$, the primal-dual iterates in $\mathcal{U}$ generated by the given algorithm satisfy

$$\frac{J^0 - J(d^{(j)}; 0)}{J^0 - J(d^{(j)}; \rho^{(j)})} > \sqrt{\frac{\beta_v}{\beta_\phi}}, \tag{7.2.13a}$$

$$\frac{J^0 - D(\mathbf{u}^k; \rho^{(j)})}{J^0 - D(\mathbf{w}^k; 0)} \geq \frac{J^0 - D(\mathbf{u}^k; \rho^{(j)})}{J^0 - D(\mathbf{u}^k; 0)} > \sqrt{\frac{\beta_v}{\beta_\phi}}, \tag{7.2.13b}$$

by the fact that $\frac{\beta_v}{\beta_\phi} \in (0, 1)$. This implies that for $j \in \mathcal{U}$

$$\frac{r_v^{(j)}}{r_\phi^{(j)}} = \frac{J^0 - J(d^{(j)}; 0)}{J^0 - J(d^{(j)}; \rho^{(j)})} \frac{J^0 - D(\mathbf{u}; \rho^{(j)})}{J^0 - D(\mathbf{u}^{(j)}; 0)} \geq \frac{\beta_v}{\beta_\phi},$$

yielding

$$r_v^{(j)} \geq \frac{\beta_v}{\beta_\phi}\, r_\phi^{(j)} \geq \beta_v, \qquad j \in \mathcal{U}$$

where the last inequality follows from the fact that $r_\phi^{(j)} \geq \beta_\phi$ for all $j \in \mathcal{U}$.

Now we determine the values of ρ that guarantee (7.2.13b). For any $j \in \mathcal{U}$, we have from (7.2.10b)

$$-\kappa_3 \rho^{(j)} \leq D(\mathbf{u}^{(j)}; \rho^{(j)}) - D(\mathbf{u}^{(j)}; 0) \leq \kappa_3 \rho^{(j)},$$

which implies that

$$1 - \frac{\kappa_3 \rho^{(j)}}{J^0 - D(\mathbf{u}^{(j)}; 0)} \leq \frac{J^0 - D(\mathbf{u}^{(j)}; \rho^{(j)})}{J^0 - D(\mathbf{u}^{(j)}; 0)} \leq 1 + \frac{\kappa_3 \rho^{(j)}}{J^0 - D(\mathbf{u}^{(j)}; 0)}.$$

It follows that (7.2.13b) holds true for any

$$\rho^{(j)} < \frac{1 - \sqrt{\beta_v / \beta_\phi}}{\kappa_3} (J^0 - D(\mathbf{u}^{(j)}; 0)) = \tilde{\rho}^{(j)}.$$

Next recall that $D(\mathbf{u}^{(j)}; 0) \leq D(\mathbf{u}_0^*; 0)$ for all j, and so $0 < \tilde{\rho}^* \leq \tilde{\rho}^{(j)}$ for all j. Therefore, for any $\rho \in [0, \tilde{\rho}]$, the (DUST) is never triggered. □

7.3 Coordinate Descent Subproblem Solver

In this section, we present a coordinate descent algorithm (CDA) which solves (QP_ρ) with $X = \mathbb{R}^n$. We have the following two problems of interest:

$$\min_{x\in\mathbb{R}^n} J(x;\rho) := \frac{1}{2}x^T H_\rho x + \rho g^T x + \sum_{i=1}^{\bar{m}} |a_i^T x + b_i| + \sum_{i=\bar{m}+1}^{m} (a_i^T x + b_i)_+, \quad \text{and} \tag{7.3.1}$$

$$\min_{z\in\mathbb{R}^n} J(z;0) := \frac{1}{2}z^T H_0 z + \sum_{i=1}^{\bar{m}} |a_i^T z + b_i| + \sum_{i=\bar{m}+1}^{m} (a_i^T z + b_i)_+. \tag{7.3.2}$$

Note that this is the same objective function as appears in (7.2.3) but with the over bars removed.

Direct computation shows the Lagrangian dual problems of (7.3.1) and (7.3.2) are respectively

$$\max_{l\le\boldsymbol{\eta}\le c} D(\boldsymbol{\eta};\rho) := -\frac{1}{2}(A^T\boldsymbol{\eta} - \rho g)^T H_\rho^{-1}(A^T\boldsymbol{\eta} - \rho g) + \boldsymbol{\eta}^T\boldsymbol{b} \tag{7.3.3}$$

$$\max_{l\le\boldsymbol{\lambda}\le c} D(\boldsymbol{\lambda};0) := -\frac{1}{2}\boldsymbol{\lambda}^T A H_0^{-1} A^T\boldsymbol{\lambda} + \boldsymbol{\lambda}^T\boldsymbol{b} \tag{7.3.4}$$

where $l = [-\mathbf{1}_s^T, \mathbf{0}_{m-s}^T]^T$ and $c = \mathbf{1}_m$. Observe that if we let $u_i = \eta_i a_i$ $i = 1, \cdots, m$, $u_{m+1} = \mathbf{0}_n$, $u_0 = A^T\boldsymbol{\eta}$, then (7.3.3) is equivalent to (DQP_ρ). In particular, $\rho = 0$ gives the equivalence between (7.3.4) and (DQP_ρ). Then it is straightforward to verify that CDA satisfying Assumption 7.2.2 so that DUST can be incorporated. The solutions to (7.3.1) and (7.3.2) are recovered by $x = -H_\rho^{-1}(\rho g + A^T\boldsymbol{\eta})$ and $z = -H_0^{-1}A^T\boldsymbol{\lambda}$ respectively.

At iteration k, define

$$r_v^k := \frac{J^{(0)} - J(x^k;0)}{J^{(0)} - \max\{D(\boldsymbol{\eta}^k;0), D(\boldsymbol{\lambda}^k;0)\}}$$
$$r_\phi^k := \frac{J^{(0)} - J(x^k;\rho^{k-1})}{J^{(0)} - D(\boldsymbol{\eta}^k;\rho^{k-1})},$$

then we have an algorithm which combines the coordinate descent with dynamic penalty parameter updating, see Algorithm 6.

7.4 Numerical Experiments

In this section, we test our proposed dynamic penalty updating strategy on 20 Cuter problems [35] whose objective and constraints are both non-linear. Coordinate descent algorithm

Algorithm 6 Coordinate Descent Method for Penalty-SQP Subproblems

1: Initialization Set $\boldsymbol{\eta}^0$, β_v, β_ϕ, $\theta_\rho \in (0,1)$ and $k = 0$.

2: (Update $\boldsymbol{\eta}^k$ and $\boldsymbol{\lambda}^k$) For $i = 1, \ldots, m$

$$\boldsymbol{\eta}_i^k := \underset{l_i \le \eta_i \le c_i}{\operatorname{argmin}} D(\eta_1^k, \ldots, \eta_{i-1}^k, \eta_i, \eta_{i+1}^{k-1}, \ldots, \eta_m^{k-1}; \rho^{k-1}) \tag{7.3.5}$$

$$\boldsymbol{\lambda}_i^k := \underset{l_i \le \lambda_i \le c_i}{\operatorname{argmin}} D(\lambda_1^k, \ldots, \lambda_{i-1}^k, \lambda_i, \lambda_{i+1}^{k-1}, \ldots, \lambda_m^{k-1}; 0) \tag{7.3.6}$$

3: Update $x^k := -H_\rho^{-1}(\rho g + A^T \boldsymbol{\eta}^k)$, r_ϕ^k and r_v^k.

4: (Update ρ^k)(DUST) If $r_\phi^k > \beta_\phi$ and $r_v^k < \beta_v$, set $\rho^k \leftarrow \theta_\rho \rho^{k-1}$.

5: Set $k \leftarrow k + 1$.

described in §7.3 is used to solve the subproblems. We set $\rho = 1$, $\beta_\phi = 0.7$, $\beta_v = 0.05$, $\theta_\rho = 0.5$ and the maximum iteration to be 100 for SQP algorithm. Define the maximum constraint violation $v_\infty(x)$ and the optimality error $\epsilon_\infty(x)$ as

$$v_\infty(x) := \max\{|c_i(x)| \;\; i = 1, \cdots, \bar{m}, \;\; (c_i(x))_+ \;\; i = \bar{m}+1, \cdots, m\},$$
$$\epsilon_\infty(x) := \|\nabla f(x) + \nabla c(x)\eta\|_\infty .$$

We terminate the algorithm if $v_\infty(x) \le 10^{-7}$ and $\epsilon_\infty(x) \le 10^{-3}$, or the consecutive improvement of the sum of constraint violation and objective function value is less than 10^{-4}. The experiments were run on MacBook Air with a 1.4 GHz Intel Core i5 processor and 4GB RAM.

Figure 7.1 presents the update of ρ from iteration to iteration. Table 7.1 shows the summary of final objective function values, ϵ_∞, ρ, running time and v_∞ of our algorithm. 19 out of these 20 problems are solved to the specified accuracy. Only EG3 is terminated due to the consecutive improvement of constraint violation and objective function value is too small. All problems are solved successfully within 40 seconds, except EG3 takes around 80 seconds.

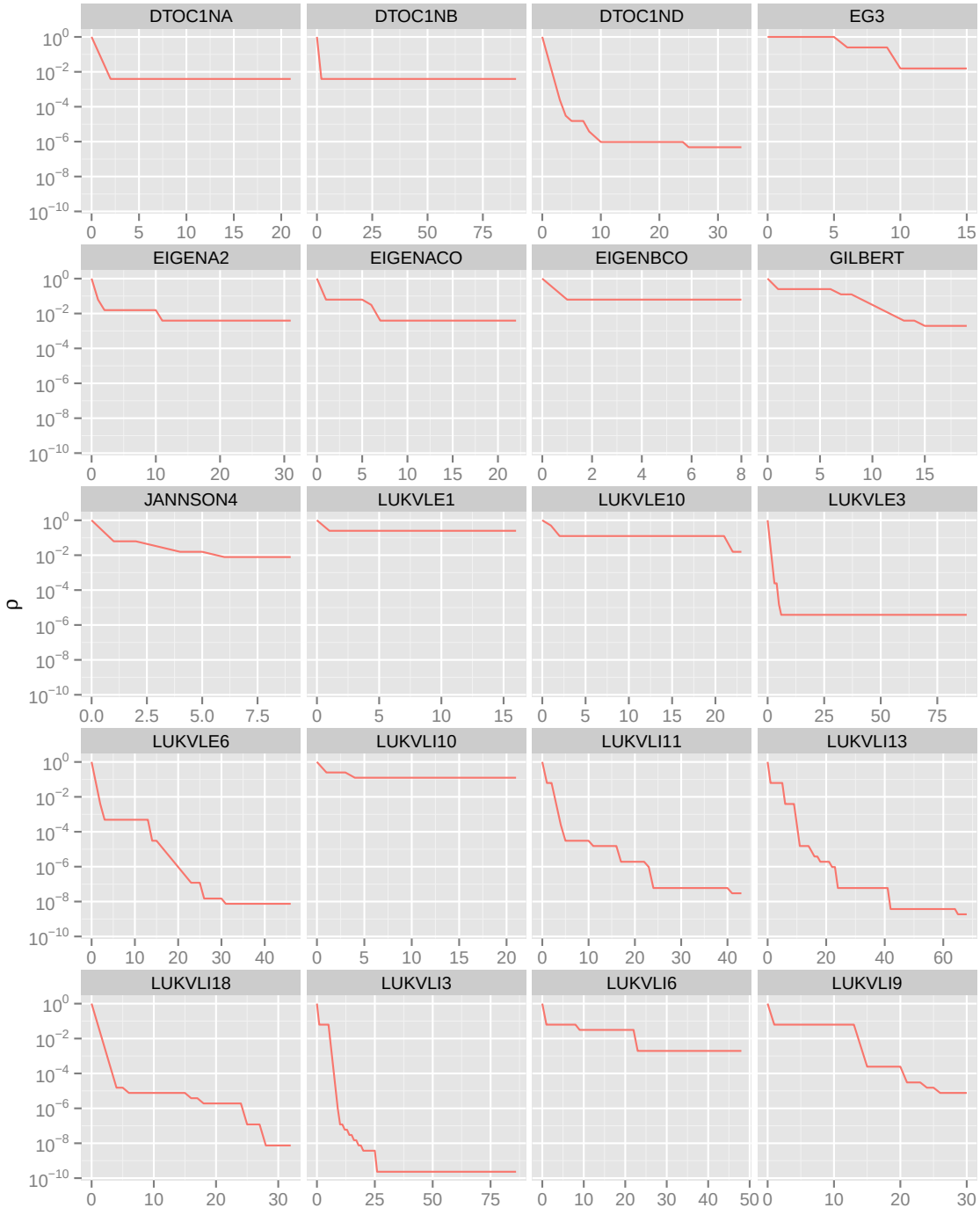

Figure 7.1: ρ update

	Name	Num_Con	Num_Eq	Num_Var	Obj	Opt_Err	ρ	Time	Violation
1	DTOC1NA	3996	3996	5998	4.1389E+00	3.8650E-05	3.9062E-03	3.6857E+00	9.9967E-08
2	DTOC1NB	3996	3996	5998	7.1388E+00	1.2129E-04	3.9062E-03	1.2981E+01	7.4492E-08
3	DTOC1ND	3996	3996	5998	4.7603E+01	5.5031E-05	4.7684E-07	5.4458E+00	1.9515E-08
4	EG3	20000	1	10001	3.4834E-04	2.6197E-02	1.5625E-02	7.0970E+01	5.2612E-10
5	EIGENA2	1275	1275	2550	3.0204E-12	3.3435E-08	3.9062E-03	1.9689E+00	0.0000E+00
6	EIGENACO	1275	1275	2550	4.5418E-09	6.2817E-07	3.9062E-03	2.1289E+00	1.7764E-15
7	EIGENBCO	1275	1275	2550	4.9000E+01	7.8237E-05	6.2500E-02	1.6198E+00	2.8207E-09
8	GILBERT	1	1	5000	2.4595E+03	3.8443E-05	1.9531E-03	5.5580E-01	9.8979E-09
9	JANNSON4	2	0	10000	9.8020E+03	2.7347E-09	7.8125E-03	1.6501E+00	4.4094E-11
10	LUKVLE1	9998	9998	10000	6.2325E+00	2.7028E-07	2.5000E-01	2.5797E+00	3.6903E-12
11	LUKVLE10	9998	9998	10000	3.5351E+03	5.6940E-04	1.5625E-02	7.4686E+00	9.0493E-08
12	LUKVLE3	2	2	10000	2.7587E+01	7.6157E-06	3.8147E-06	3.5032E+00	2.1330E-10
13	LUKVLE6	4999	4999	9999	6.2864E+05	9.2954E-06	7.4506E-09	5.1310E+00	3.0809E-12
14	LUKVLI10	9998	0	10000	3.5351E+03	2.3222E-03	1.2500E-01	4.9544E+00	2.4736E-09
15	LUKVLI11	6664	0	9998	5.0882E-05	8.3197E-04	2.9802E-08	4.7913E+00	2.6160E-08
16	LUKVLI13	6664	0	9998	1.3219E+02	4.8303E-04	1.8626E-09	9.9030E+00	4.8814E-08
17	LUKVLI18	7497	0	9997	2.8250E-04	7.0668E-04	7.4506E-09	3.3149E+01	8.4118E-09
18	LUKVLI3	2	0	10000	6.7819E+02	9.2782E-06	2.3283E-10	3.9321E+00	4.4420E-12
19	LUKVLI6	4999	0	9999	6.2864E+05	2.2232E-06	1.9531E-03	3.0957E+01	5.8365E-12
20	LUKVLI9	6	0	10000	9.9893E+02	3.9280E-04	7.6294E-06	1.6746E+00	0.0000E+00

Table 7.1: 20 Cuter problem results

7.5 Conclusion

In this chapter, we propose a dynamic penalty parameter updating strategy to be incorporated *within* the subproblem solver so that each computed search direction predicts progress toward both feasibility and optimality. We test our penalty parameter updating strategy with a coordinate descent subproblem solver on 20 Cuter problems with thousands of variables. Our numerical experiments demonstrate that our penalty updating strategy works well on those problems we have tested. In the future, we can examine other subproblem solvers to be combined with our dynamic penalty updating strategy.

Chapter 8

SVD-FREE BLOCK-COORDINATE DESCENT METHOD FOR LEARNING STRUCTURED GAUSSIAN GRAPHICAL MODELS

8.1 SVD-free Block Coordinate Descent Algorithm (SBCD)

Recall that our optimization model in (1.3.1) is given by:

$$\begin{aligned} \underset{V\in\mathbb{R}^{p\times p},\, Z\in\mathcal{S}^p}{\text{minimize}} \quad & f(V,Z)+g(V,Z) \\ \text{subject to} \quad & V+V^T \succeq 0 \quad (\text{or } \succ 0), \end{aligned} \tag{8.1.1}$$

where $f:\mathbb{R}^{p\times p}\times\mathcal{S}^p\to\mathbb{R}\cup\{+\infty\}$ is convex and continuously differentiable on its domain and $g:\mathbb{R}^{p\times p}\times\mathcal{S}^p\to\mathbb{R}\cup\{+\infty\}$ is convex and block separable in the columns of V and Z. A key bottle-neck in algorithms for (8.1.1) is checking for the positive-definiteness of the the expression $V+V^T$ or the implicit constraint of $\log\det$ term as shown in hub precision matrix learning and latent variable selection problem at each iteration. Many algorithms for learning Gaussian graphical models such as ADMM [48] or proximal Newton method (QUIC) [41] use either the singular value decomposition (SVD) or Cholesky decomposition to check for positive-definiteness. While the Cholesky decomposition is faster than SVD for larger problem sizes, it is still more expensive than a matrix-vector multiply. Here we propose a simple rule, depending only on matrix-vector multiplies, that preserves the positive definiteness of the intermediate iterates in our step-size selection procedure.

8.1.1 Step-size selection for positive definiteness

Our proposed algorithm for solving the problem (8.1.1) uses a block-coordinate descent update where the block is given by a row and a column of a matrix. This translates to a rank-two update to a positive definite matrix. We therefore ask the following question:
Given $\Theta\in\mathcal{S}^p_{++}$, for what range of α is $\Theta+\alpha(uw^T+wu^T)\in\mathcal{S}^p_{++}$, where $u,w\in\mathbb{R}^p$?
Answering this question will help us choose a step-size that maintains the positive definiteness of the iterates. Interestingly, we can compute this range of α in closed form. Choosing

such a step-size ensures that in each iteration of our approach, we stay within the positive definite cone while also making progress towards an optimal solution.

To arrive at a range for α, we need to examine the eigenvalues of $\Theta + \alpha(uw^T + wu^T)$. Assume that $\Theta \succ 0$ with eigenvalues $0 < \eta_p \leq \eta_{p-1} \cdots \eta_2 \leq \eta_1$. Before we examine the eigenvalues of $\Theta + \alpha(uw^T + wu^T)$, let us take a look at the eigenvalues of $B := uw^T + wu^T$. These are given in the following elementary lemma.

Lemma 8.1.1. *Let $B := uw^T + wu^T$ where u and w are not linearly dependent, then B has $p-2$ zero eigenvalues and two non-zero eigenvalues $w^T u \pm \|u\|\|w\|$ with corresponding eigenvectors $u/\|u\| \pm w/\|w\|$, respectively. We order the eigenvalues of B as*

$$w^T u - \|u\|\|w\| =: \beta_p < 0 = \beta_{p-1} = \beta_{p-2} \cdots = \beta_2 < \beta_1 := w^T u + \|u\|\|w\|. \tag{8.1.2}$$

Before we give a full characterization of the eigenvalues of $\Theta + \alpha B$, we need another lemma which is a consequence of the eigenvalue interlacing lemma due to Weyl.

Lemma 8.1.2. *[39, Theorem 4.3.1] Let Θ and B be symmetric and denote the ordered eigenvalues of Θ, B and $\Theta + B$, respectively, by $\eta_p \leq \eta_{p-1} \leq \cdots \leq \eta_1$, $\beta_p \leq \beta_{p-1} \leq \cdots \leq \beta_1$, and $\lambda_p \leq \lambda_{p-1} \leq \cdots \leq \lambda_1$. Then*

$$\eta_p \leq \lambda_i - \beta_i \leq \eta_1 \quad i = 1, \cdots, p \tag{8.1.3}$$

We now give our main result on the eigenvalues of $\Theta + \alpha(uw^T + wu^T)$.

Theorem 8.1.3. *Let $\Theta \succ 0$ and $B = uw^T + wu^T$ where u and w are not linearly dependent, then $\Theta + \alpha B \succ 0$ if and only if $\det(\Theta + \alpha B) > 0$, or equivalently,*

$$\delta^{-1}(w^T\Theta^{-1}u - \sqrt{(u^T\Theta^{-1}u)(w^T\Theta^{-1}w)}) \;<\; \alpha \;<\; \delta^{-1}(w^T\Theta^{-1}u + \sqrt{(u^T\Theta^{-1}u)(w^T\Theta^{-1}w)}, \tag{8.1.4}$$

where $\delta = (u^T\Theta^{-1}u)(w^T\Theta^{-1}w) - (u^T\Theta^{-1}w)^2$.

Proof. Denote the eigenvalues of Θ by $0 < \eta_p \leq \eta_{p-1} \leq \cdots \leq \eta_1$ and those of of $\Theta + \alpha B$ by $\lambda_p \leq \lambda_{p-1} \leq \cdots \leq \lambda_1$. Lemma 8.1.1 shows the eigenvalues of B satisfy

$$\beta_p < 0 = \beta_{p-1} = \beta_{p-2} \cdots = \beta_2 < \beta_1,$$

while Lemma 8.1.2 implies that

$$\eta_p \leq \lambda_i - \alpha\beta_i \leq \eta_1 \quad i = 1, \cdots, p. \tag{8.1.5}$$

In particular,

$$\lambda_1 \geq \eta_p + \alpha\beta_1 > 0 \quad \text{and} \quad \lambda_i \geq \eta_p + \alpha\beta_i = \eta_p > 0 \quad i = 2, \cdots, p-1.$$

This means $\Theta + \alpha B$ has at most one negative eigenvalue λ_p. Observe that

$$\det(\Theta + \alpha B) = \Pi_{i=1}^{p} \lambda_i,$$

hence $\det(\Theta + \alpha B) > 0$ if and only if $\lambda_p > 0$.

Note that

$$\begin{aligned} \det(X + uw^T) &= \det(X)\det(I + X^{-1}uw^T) \\ &= \det(X)(1 + w^T X^{-1}u), \end{aligned} \tag{8.1.6}$$

where we use the fact that the only non-zero eigenvalue of a rank one matrix ab^T is given $b^T a$. Using (8.1.6) twice, we have that

$$\det(\Theta + (uw^T + wu^T)) = \det(\Theta)[(1 + w^T\Theta^{-1}u)^2 - (w^T\Theta^{-1}w)(u^T\Theta^{-1}u)], \tag{8.1.7}$$

which, in turn, implies that

$$\det(\Theta + \alpha B) = \det(\Theta)[(1 + \alpha w^T\Theta^{-1}u)^2 - \alpha^2(w^T\Theta^{-1}w)(u^T\Theta^{-1}u)].$$

Therefore, $\det(\Theta + \alpha B) > 0$ if and only if

$$(1 + \alpha w^T\Theta^{-1}u)^2 - \alpha^2(w^T\Theta^{-1}w)(u^T\Theta^{-1}u) > 0. \tag{8.1.8}$$

From the Cauchy-Schwartz inequality, we have that

$$\delta := (w^T\Theta^{-1}w)(u^T\Theta^{-1}u) - (w^T\Theta^{-1}u)^2 \geq 0,$$

where equality holds if and only if u and w are linearly dependent. If we expand the expression on the left-hand side of (8.1.8) we get

$$-\delta\,\alpha^2 \;+\; 2\,(u^T\Theta^{-1}w)\,\alpha \;+\; 1\;,$$

where $\delta > 0$ since u and w are linearly independent. The result now follows from the quadratic formula. □

Remark 8.1.4. Theorem 8.1.3 can be proven for the special case of $w = e_i$ through the use of Schur complement conditions for the positive-definiteness of a matrix (see e.g. [72]). This special case is used to check for positive definiteness of iterates in hub structure. However Theorem 8.1.3 is a more general result that provides a range for the step-size α so that any symmetric rank two update of a positive definite matrix preserves positive definiteness. We require the general more result for the latent variable selection problem.

8.1.2 SVD-free block-coordinate descent

When applying SBCD [71] to solve (8.1.1), one alternates between the columns of V and those of Z. In SBCD, when one optimizes a column of V, all other columns of V and Z are kept fixed. In order to employ (8.1.4), we make the assumption that we modify V or Z through a rank 2 update. The SBCD is described in Algorithm 7.

Algorithm 7 SBCD algorithm

1: **Input:** f, g, q, ϵ.

2: **Output:** $(\hat{V}, \hat{Z})$.

3: V **update**: For $i \in \{1, 2, \ldots, p\}$

1. $\bar{d} \leftarrow \underset{d}{\operatorname{argmin}} \langle [\nabla_V f(V, Z)]_i, d\rangle + \frac{1}{2} d^T H d + \lambda g(V + de_i^T, Z)$
2. Choose α s.t. α satisfies Armijo rule as stated in Algorithm 8 and (8.1.4) with $\Theta = V + V^T$.
3. $V_i \leftarrow V_i + \alpha \bar{d}$

4: Z **update**: For $i \in \{1, 2, \ldots, p\}$

1. $\bar{d} \leftarrow \underset{d}{\operatorname{argmin}} \langle [\nabla_Z f(V, Z)]_i, d\rangle + \frac{1}{2} d^T H d + \lambda g(V, Z + de_i^T + e_i^T d)$
2. Choose α s.t. α satisfies Armijo rule as stated in Algorithm 8 (and (8.1.4) if necessary) with $\Theta = V + V^T$.
3. $Z \leftarrow Z + \alpha(\bar{d} e_i^T + e_i^T \bar{d})$

Remark 8.1.5. 1. Armijo rule for sufficient descent: In each iteration of Algorithm 7, we use the Armijo rule to select a step-size α that ensures sufficient descent in the

Algorithm 8 Armijo rule for minimizing the sum of a smooth function $l(x)$ and a non-smooth convex function $q(x)$

Input: Functions l, q, current iterate x, direction d and initial α_{init}

Output: $\hat{\alpha}$.

Initialize: Initial step size $\alpha = \alpha_{init}$, Armijo parameters $\sigma = 0.5, \beta = 0.5, \alpha_{\min} = 10^{-6}$.

1 Set $\Delta = q(x + d) - q(x) + \langle \nabla l(x), d \rangle$.

2 While $l(x + \alpha d) + q(x + \alpha d) - l(x) - q(x) \leq \alpha\sigma\Delta$ and $\alpha \geq \alpha_{\min}$, set $\alpha = \alpha\beta$.

3 If $\alpha < \alpha_{\min}$, set $\alpha = 0$.

4 Set $\hat{\alpha} = \alpha$.

objective value of (8.1.1). More formally, consider the following problem: $\min_x l(x) + q(x)$ where l is smooth and convex while q is convex. Given a direction d and an initial point x, the Armijo rule is used to select a step size α that ensures that $x + \alpha d$ gives sufficient decrease of the objective. The Armijo rule for this problem is summarized in Algorithm 8, and is an instance of the Armijo rule for *convex-composite functions* [7].

2. Updating the inverse: In each iteration of Algorithm 7, we need to compute the step-size α. To ensure that Θ stays positive definite, we need α to satisfy (8.1.4). This requires the computation of the inverse of Θ. Since the update to Θ involves a rank-2 update, the inverse of Θ can be easily updated using the Sherman-Morrison-Woodbury identity [70]. The cost of the inverse update is $O(p^2)$.

8.2 SBCD for Learning Graphs with Hubs

We consider two problems in this section: Learning a precision matrix with hub structure and learning a covariance matrix with hub structure.

8.2.1 SBCD for hub precision matrix

A simple version of the hub graphical lasso (see section 3 of [64]) (where we ignore the sparsity promoting regularizers for the sake of simplicity) is given by the following model:

$$\underset{V:V+V^T\succ 0}{\text{minimize}} \quad -\log\det(V+V^T)+\langle V+V^T,S\rangle+\lambda\|V-\text{Diag}(V)\|_{1,2}, \tag{8.2.1}$$

where we assume that the domain of $\log\det(\cdot)$ is the set of all positive definite matrices and $\|V\|_{1,2}:=\sum_{i=1}^{p}\|V_{\cdot i}\|_2$. Let $u^{\setminus i}$ denote a copy of the vector u with the i-th coordinate set to zero. In particular, we let $v^{\setminus i}$ denote $(V_{\cdot i})^{\setminus i}$.

Note that (8.2.1) is equivalent to:

$$\underset{V:V+V^T\succ 0}{\text{minimize}} \quad -\log\det(V+V^T)+2\langle S,V\rangle+\lambda\sum_{i=1}^{p}\left\|v^{\setminus i}\right\|_2. \tag{8.2.2}$$

We use Algorithm 7 to solve (8.2.2) with $f(V)=-\log\det(V+V^T)+2\langle S,V\rangle$ and $g(V)=\lambda\sum_{i=1}^{p}\left\|v^{\setminus i}\right\|_2$. To do this, we need to compute the search direction d of Algorithm 7 efficiently. We first note that optimizing $V_{\cdot i}$ while fixing other columns of V yields the following subproblem:

$$\underset{u}{\text{minimize}} \quad f(V+ue_i^T)+g(V+ue_i^T), \tag{8.2.3}$$

which is equivalent to

$$\begin{aligned}\underset{u}{\text{minimize}} \quad & -\log\det(V+V^T+ue_i^T+e_iu^T)+2\langle S,V+ue_i^T\rangle\\ & +\lambda\textstyle\sum_{i=1}^{p}\left\|(V+ue_i^T-\text{Diag}(V+ue_i^T))_{\cdot i}\right\|_2,\end{aligned}$$

and ignoring the constant terms gives

$$\underset{u}{\text{minimize}} \quad -\log\det(V+V^T+ue_i^T+e_iu^T)+2S_{\cdot i}^Tu+\lambda\left\|v^{\setminus i}+u^{\setminus i}\right\|_2. \tag{8.2.4}$$

The challenge in (8.2.4) is that evaluations of the objective function in the line search of Algorithm 7 involves the log-determinant which can be expensive. To address this difficulty, we give the following lemma which transforms (8.2.4) into a simpler problem.

Lemma 8.2.1. *Problem (8.2.4) is equivalent to the following optimization problem:*

$$\underset{u}{\text{minimize}}\ h(u):=l(u)+q(u), \tag{8.2.5}$$

where

$$l(u) = -\log[(1 + e_i^T \Theta^{-1} u)^2 - (e_i^T \Theta^{-1} e_i)(u^T \Theta^{-1} u)] + 2S_{\cdot i}^T u. \tag{8.2.6}$$

and

$$q(u) = \lambda \left\| v^{\setminus i} + u^{\setminus i} \right\|_2. \tag{8.2.7}$$

Proof. Let

$$\hat{l}(u) := f(V + ue_i^T) = -\log\det(\Theta + ue_i^T + e_i u^T) + 2S_{\cdot i}^T u, \tag{8.2.8}$$

and assume we keep track of the inverse of $\Theta := V + V^T$. Then using (8.1.7) we get that optimizing $\hat{l}(u) + q(u)$ with respect to u is equivalent to optimizing $l(u) + q(u)$ with respect to u.

□

Note that $l(u)$ as defined in (8.2.6) does not involve the computation of a determinant and hence is more efficient to compute. Now we are ready to describe the computation of searching direction d of Algorithm 7. Consider the following approximation to (8.2.5) at $\widetilde{u}$

$$\underset{u}{\text{minimize}}\ h(u; \widetilde{u}) := l(\widetilde{u}) + \langle \nabla l(\widetilde{u}), u - \widetilde{u} \rangle + \frac{1}{2}(u - \widetilde{u})^T H (u - \widetilde{u}) + \lambda \left\| v^{\setminus i} + u^{\setminus i} \right\|_2, \tag{8.2.9}$$

where H is the an approximation to the Hessian of $l(\widetilde{u})$. Consider a simple case $H = \beta I$ and define

$$\bar{u} = \left(1 - \frac{\lambda}{\left\| (\nabla l(\widetilde{u}))^{\setminus i} - \beta(v^{\setminus i} + \widetilde{u}^{\setminus i}) \right\|_2}\right) + \left(v^{\setminus i} + \widetilde{u}^{\setminus i} - \frac{1}{\beta}\nabla l(\widetilde{u})^{\setminus i}\right) - v^{\setminus i},$$

where $(x)_+ = \max\{0, x\}$, then the solution to (8.2.9) is

$$u^* = \begin{cases} \bar{u}_j & \text{if } j \neq i, \\ -\left(\nabla l\left(\widetilde{u}\right)\right)_j / \beta + \widetilde{u}_j & \text{if } j = i. \end{cases} \tag{8.2.10}$$

Furthermore $\nabla l(u)$ can be computed easily as:

$$\nabla l(u) = -2\frac{(1 + e_i^T \Theta^{-1} u)\Theta^{-1} e_i - (e_i^T \Theta^{-1} e_i)\Theta^{-1} u}{(1 + e_i^T \Theta^{-1} u)^2 - (e_i^T \Theta^{-1} e_i)(u^T \Theta^{-1} u)} + 2S_{\cdot i}, \tag{8.2.11}$$

where the main computation is $\Theta^{-1} u$.

Algorithm 9 SBCD algorithm for hub graphical lasso (8.2.1)

Input: Sample covariance matrix S, λ, optimality tolerance $\epsilon > 0$.

Output: ϵ-approximate optimal solution $\hat{V}$.

Initialization: Set $\sigma \in (0,1)$, $\mathcal{A} = \{\}$, $V = \frac{1}{2}(\mathrm{Diag}(S))^{-1}$, $\Theta = V + V^T$ and $\Theta^{-1} = \mathrm{Diag}(S)$.

While $\mathcal{A}^c \neq \{\}$

1 For $i = 1, 2, \ldots, p$

$$\mathbf{\Gamma}_{\cdot i} = \begin{cases} \lambda v^{\setminus i}/\|v^{\setminus i}\|_2 & \text{if } v^{\setminus i} \neq \mathbf{0} \\ (2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i} & \text{if } v^{\setminus i} = \mathbf{0} \text{ and } \|(2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i}\|_2 \leq \lambda \\ \lambda \frac{(2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i}}{\|(2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i}\|_2} & \text{if } v^{\setminus i} = \mathbf{0} \text{ and } \|(2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i}\|_2 > \lambda. \end{cases} \tag{8.2.12}$$

2 Identify active set: $\mathcal{A} = \left\{ i : \left\| -2\Theta_{\cdot i}^{-1} + 2S_{\cdot i} + \mathbf{\Gamma}_{\cdot i} \right\|_\infty \leq \epsilon \right\}$.

3 For $i \in \mathcal{A}^c$

(a) Set $d := \underset{u}{\mathrm{argmin}}\ h(u; 0)$. The solution in closed form is given by (8.2.10).

(b) Set α_{init} to be the upper bound in (8.1.4) with $u = d, w = e_i$.

(c) Set $V_{\cdot i} \leftarrow V_{\cdot i} + \alpha d$, where α is the output of Algorithm 8 with input $l, q, x = \mathbf{0}, d, \alpha_{\text{init}}$ where l, q are as in (8.2.6), (8.2.7).

(d) Update $\Theta^{-1} = (V + V^T)^{-1}$ using the Sherman-Morrison-Woodbury rank two update scheme.

8.2.2 Active set heuristic

Under the assumption that we have only k hubs in the true precision matrix, we would expect that most of the columns of V would converge to 0 after sufficiently many iterations. Thus, it would be more efficient to skip the columns that already attain optimality accuracy. The optimality condition of (8.2.1) is

$$2\Theta^{-1} - 2S \in \partial(\lambda g(V)).$$

For the i-th column of V, its optimality error η_i is computed as

$$\eta_i := \min_{\mathbf{\Gamma} \in \partial(\lambda g(V))} \left\| -2\Theta^{-1}_{\cdot i} + 2S_{\cdot i} + \mathbf{\Gamma}_{\cdot i} \right\|_\infty . \tag{8.2.13}$$

The minimum is achieved by $\mathbf{\Gamma}_{\cdot i}$ defined below

$$\mathbf{\Gamma}_{\cdot i} = \begin{cases} \lambda v^{\setminus i}/\|v^{\setminus i}\|_2 & \text{if } v^{\setminus i} \neq \mathbf{0} \\ (2\Theta^{-1}_{\cdot i} - 2S_{\cdot i})^{\setminus i} & \text{if } v^{\setminus i} = \mathbf{0} \text{ and } \|(2\Theta^{-1}_{\cdot i} - 2S_{\cdot i})^{\setminus i}\|_2 \leq \lambda \\ \lambda \frac{(2\Theta^{-1}_{\cdot i} - 2S_{\cdot i})^{\setminus i}}{\|(2\Theta^{-1}_{\cdot i} - 2S_{\cdot i})^{\setminus i}\|_2} & \text{if } v^{\setminus i} = \mathbf{0} \text{ and } \|(2\Theta^{-1}_{\cdot i} - 2S_{\cdot i})^{\setminus i}\|_2 > \lambda. \end{cases} \tag{8.2.14}$$

Define the active set $\mathcal{A} := \{i : \eta_i \leq \epsilon\}$, and $\mathcal{A}^c := \{1, \cdots, p\} \setminus \mathcal{A}$ be the complement of $\mathcal{A}$. In every iteration of Algorithm 9, we first identify $\mathcal{A}$ and only update the columns in $\mathcal{A}^c$ to obtain the next iterate V. An initialization of $\mathcal{A}$ which works well in practice is $\{i : \left\| S^{\setminus i}_{\cdot i} \right\|_2 \leq \frac{\lambda}{2}\}$. The SBCD algorithm with this active set heuristic for hub graphical lasso is given in Algorithm 9.

The active set heuristic allows for the efficient computation of the solution path (which refers to the set of solutions output by the algorithm as we sweep over λ) when combined with warm starts. Assume we have $\lambda \in [0, K]$, where $K = 2\|S - \mathrm{Diag}(S)\|_{\infty,2}$, where $\|X\|_{\infty,2}$ denotes the maximum $\|\cdot\|_2$ norm of the columns of the matrix X. We then pick m equally spaced points in the interval $[0, K]$ and denote them by $a_i = K - K/m \times i$, $i = 0, 1, \ldots, m-1$. In our numerical experiments, we sequentially solve (8.2.2) with $\lambda = a_i$ by initializing it with the solution of (8.2.2) with $\lambda = a_{i+1}$ for $i = m-2, m-1, \ldots, 0$. We discuss this further in Section 8.2.4 and Section 8.2.6.

8.2.3 HGlasso algorithm for hub graphical lasso

While SBCD is a SVD-free approach, there have been other SVD-free approaches which have been proposed for learning sparse graphical models. Indeed, Glasso [47] is a SVD-free algorithm proposed for graphical lasso. It is straightforward to extend Glasso to the problem of hub graphical lasso; see Algorithm 10 below. We refer to this algorithm as the HGlasso algorithm. We also note that active set heuristic in Algorithm 10 speeds up the algorithm considerably.

Remark 8.2.2. In HGlasso note that once v_{12} is updated, v_{22} is updated in step 3e) of Algorithm 10 as $v_{22} = \frac{1}{2w_{22}} + \frac{1}{2}\boldsymbol{\theta}_{12}^T(\Theta_{11})^{-1}\boldsymbol{\theta}_{12}$ (here $\boldsymbol{\theta}_{21} = \boldsymbol{\theta}_{12}$). This ensures the positive definiteness of the iterates. Indeed, since $\Theta_{11} \succ 0$, we require that $\theta_{22} - \boldsymbol{\theta}_{12}^T\Theta_{11}^{-1}\boldsymbol{\theta}_{12} > 0$. But since $\theta_{22} = 2v_{22} = \frac{1}{w_{22}} + \boldsymbol{\theta}_{12}^T(\Theta_{11})^{-1}\boldsymbol{\theta}_{12}$, it follows that $\theta_{22} - \boldsymbol{\theta}_{12}^T\Theta_{11}^{-1}\boldsymbol{\theta}_{12} = \frac{1}{w_{22}} > 0$. The inverse Θ_{11}^{-1} is easily computed from $W = \Theta^{-1}$. Indeed from the block matrix inversion lemma (see e.g. [70], [55]), we have that $\Theta_{11}^{-1} = W_{11} - w_{12}w_{12}^T/w_{22}$.

Remark 8.2.3. In our implementation of Algorithm 10, the subproblem in step 3b) is solved using the proximal gradient method. Whenever a column of V is updated, a row and a column of $\Theta = V + V^T$ is updated. This translates to a rank two update to Θ. Thus we can update $W = \Theta^{-1}$ in $O(p^2)$ by applying the Sherman-Morrison-Woodbury update scheme (see e.g. [70]).

8.2.4 Experimental results for hub graphical lasso

Synthetic data results: We generated a precision matrix Θ_0 with hubs similar to the Erdos-Renyi setup as described in [64]. For any given problem dimension p, we let the number of hubs k in Θ_0 equal $0.03 \times p$. We generated the sample covariance matrix S, from Θ_0 using a sample size, $n = 0.1 \times p \times k$. We use two metrics for comparison of algorithms: F-score and run-time in seconds. Let $\hat{\Theta} = \hat{V} + \hat{V}^T$ denote an approximate solution provided by an algorithm. Let $\mathcal{T}_0$ and $\hat{\mathcal{T}}$ be the support or set of non-zero entries of Θ_0 and $\hat{\Theta}$, respectively. Then the precision is given by $\frac{|\hat{\mathcal{T}} \cap \mathcal{T}_0|}{|\hat{\mathcal{T}}|}$ and the recall is given by $\frac{|\hat{\mathcal{T}} \cap \mathcal{T}_0|}{|\mathcal{T}_0|}$. The F-score equals the harmonic mean of the precision (P) and recall (R), i.e., F-score $= \frac{2 \times P \times R}{P+R}$. If the solution output by an algorithm has an F-score of 0, this implies

Algorithm 10 The HGlasso algorithm for hub graphical lasso (8.2.1).

Input: Sample covariance matrix S, λ, optimality tolerance $\epsilon > 0$

Output: ϵ-approximate optimal solution V

Initialize: $V = (\text{Diag}(S))^{-1}/2, \Theta = (\text{Diag}(S))^{-1}, W = \Theta^{-1} = \text{Diag}(S)$ and active set, $\mathcal{A} = \{\}$.

While $\mathcal{A}^c \neq \{\}$

1 For $i = 1, 2, \ldots, p$

$$\mathbf{\Gamma}_{\cdot i} = \begin{cases} \lambda v^{\setminus i}/\|v^{\setminus i}\|_2 & \text{if } v^{\setminus i} \neq \mathbf{0} \\ (2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i} & \text{if } v^{\setminus i} = \mathbf{0} \text{ and } \|(2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i}\|_2 \leq \lambda \\ \lambda \frac{(2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i}}{\|(2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i}\|_2} & \text{if } v^{\setminus i} = \mathbf{0} \text{ and } \|(2\Theta_{\cdot i}^{-1} - 2S_{\cdot i})^{\setminus i}\|_2 > \lambda. \end{cases} \tag{8.2.15}$$

2 Identify active set: $\mathcal{A} = \{i : \| - 2W_{\cdot i} + 2S_{\cdot i} + \mathbf{\Gamma}_{\cdot i}\|_\infty \leq \epsilon\}$.

3 For $i \in \mathcal{A}^c$

(a) Permute rows and columns of V so that the ith column of V before permutation is now the last column of V. Do the same for Θ, W. Let $V = \begin{bmatrix} V_{11} & v_{12} \\ v_{21}^T & v_{22} \end{bmatrix}$ where $V_{11} \in \mathbb{R}^{p-1 \times p-1}, v_{12} \in \mathbb{R}^{p-1}, v_{22} \in \mathbb{R}$. Let $\Theta, W = \Theta^{-1}, S$ have a similar block matrix decomposition.

(b) Fix V_{11}, v_{21}, v_{22} and optimize for v_{12}: $\hat{u} = \underset{u}{\text{argmin}}\, u^T(\Theta_{11})^{-1}u + u^T(2s_{12} + 2w_{22}(\Theta_{11})^{-1}v_{21}) + \lambda\|u\|_2$

(c) $v_{12} = \hat{u}s_{22}$.

(d) $\boldsymbol{\theta}_{12} = v_{12} + v_{21}$, $\boldsymbol{\theta}_{21} = v_{12} + v_{21}$.

(e) $v_{22} = \frac{1}{2w_{22}} + \frac{1}{2}\boldsymbol{\theta}_{21}^T(\Theta_{11})^{-1}\boldsymbol{\theta}_{12}$.

(f) Update $W = \Theta^{-1}$ using the Sherman-Morrison-Woodbury identity.

(g) Unpermute rows and columns of V, W, Θ.

that the algorithm has either a zero precision or zero recall. On the other hand, an F-score of 1 implies that both the precision and recall equal 1. Thus, the higher the F-score, the better the performance of the algorithm on this metric. We now describe our experimental results. All the algorithms in the following sections are implemented in C and use the BLAS and LAPACK linear algebra libraries. The experiments were run on MacBook Air with a 1.4 GHz Intel Core i5 processor and 4GB RAM.

1. In the first set of experiments in Figure 8.1, we plot the metrics against a range of regularization parameters λ for different choices of p. In particular, we consider $p = 500, 1000$. We vary λ as follows: Let $\lambda_{\max} = 2\|S - \mathrm{Diag}(S)\|_{\infty,2}$, where we define the $\|X\|_{\infty,2}$ norm as the maximum ℓ_2 norm of the columns of X. We then generate a λ grid by dividing the interval $[0, \lambda_{\max}]$ into 30 equal parts. Denote these points by $\{\lambda_i\}$, $i = 1, \ldots, 30$. The solution path for a given algorithm refers to the set of solutions output by the algorithm for regularization parameters λ_1 through λ_{30}. We use warm-starts to speed up the computation of the solution path for the algorithms, i.e. we start by running an algorithm for λ_{30} and then successively use the solution for λ_{i+1} to initialize the algorithm for the computation of solution for λ_i. Since the solution from a previous λ_i will be a better starting point than any random starting point, warm-starting speeds up the computation of the solution path. If for some λ_i, the solution output by the algorithm has more than 10% non-zero columns, then we stop the computation of the solution path at this point. This is done because, with more than 10% non-zero columns, the solution is no longer sparse. Since we are interested in structured sparse solutions, computing the remaining solution path wouldn't be of use.

   We set the optimality tolerance ϵ for all the algorithms to be 10^{-4}.
   As can be seen from the top panel of Figure 8.1, the F-scores match up for ADMM, HGlasso and SBCD. In the bottom panel of Figure 8.1, we see that in precisely this good regime, both HGlasso and SBCD are around 100 times faster than the SVD based ADMM algorithm for $p = 500, 1000$.

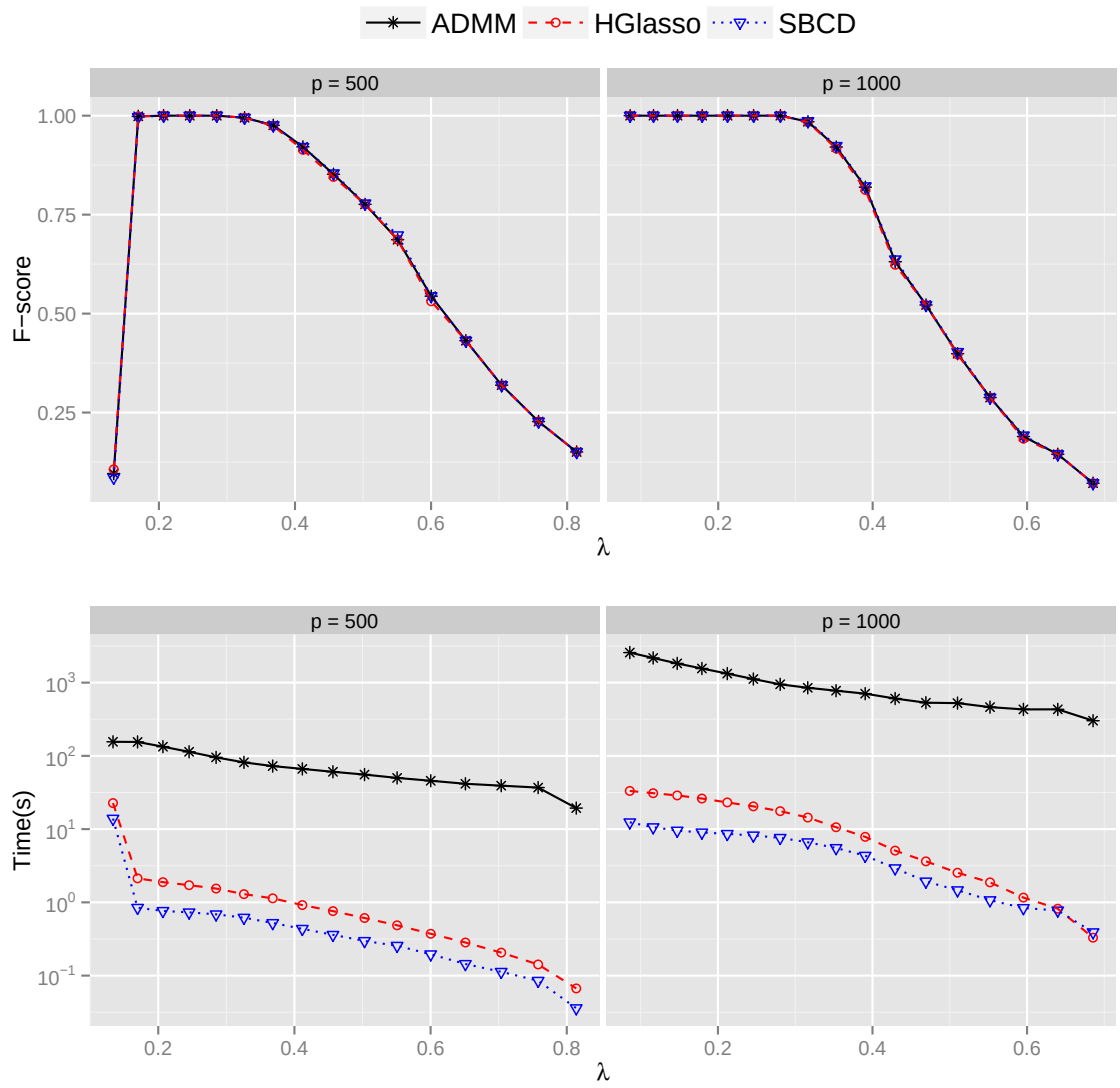

Figure 8.1: The figures compare three algorithms, ADMM, HGlasso and SBCD over two metrics, F-score and run time. The top panel shows F-score vs λ comparisons for $p = 500, 1000$ which match for all of the algorithms as expected. The top panel is useful in deciding the range of λ in which the F-score is high. The bottom panel shows the run time comparison against λ for $p = 500, 1000$. SBCD and HGlasso take much less time than ADMM in the range of interest where the F-score is high.

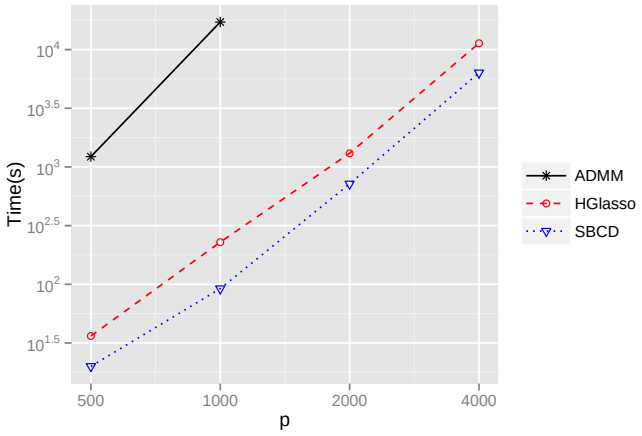

Figure 8.2: This plot compares the total solution path time of SBCD , HGlasso and ADMM across different problem dimensions p. The plot shows that SBCD is about 2 times faster than HGlasso and about 200 times faster than ADMM. We compute the solution path time of ADMM only for $p \leq 1000$ since ADMM timed out for p equal to 2 and 4 thousand.

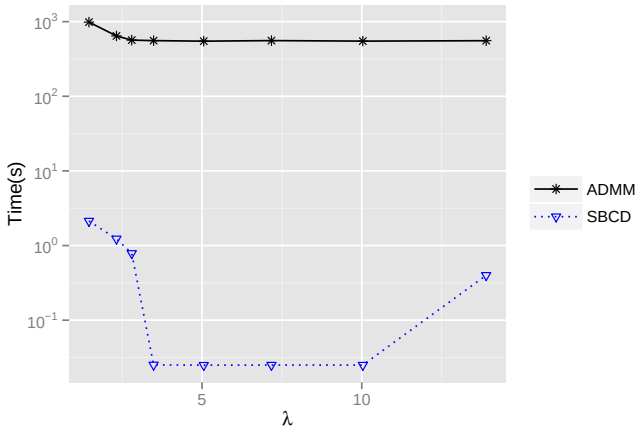

Figure 8.3: Run time (in seconds) comparison of ADMM and SBCD on a real data set with $p = 500, n = 401$. SBCD is around 1000 times faster than ADMM for a wide range of λ.

2. In the second set of experiments in Figure 8.2, we compare the solution path time of ADMM, HGlasso and SBCD against the problem dimension p. The solution path time for a given p and a given algorithm, is the total time taken by the algorithm to compute the solution path (i.e. the total run time across λ in Figure 8.2). For $p = 4000$, SBCD is 2 times faster than HGlasso. It is interesting to note that, for $p = 4000$, SBCD takes only 25% of the run time that ADMM takes for $p = 1000$. Although we haven't plotted the solution path time for ADMM for $p = 4000$, we expect it to be at least an order of magnitude slower than SBCD. The gains in run times are due to a combination of the absence of SVD factorizations in SBCD and the ability of SBCD to exploit the sparse solution structure (for example, through the active set heuristic) more readily than the ADMM algorithm.

Real data results: We next consider a real data experiment on a gliblastoma mutliforme (GBM) cancer data set over $p = 500$ genes. The sample covariance matrix S is generated from 401 samples using the procedure outlined in Section 6.2 of [64]. Figure 8.3 shows the run-time results on this data. We allow λ to range in a manner similar to the procedure used in synthetic data experiment of the previous section. We note that SBCD is around 1000 times faster than ADMM across this range of λ.

8.2.5 SBCD for hub covariance selection

A simpler version of the hub covariance selection problem [64] (where we ignore the regularization parameters that promote sparsity) is as follows:

$$\begin{array}{ll} \underset{\Theta \succ 0}{\text{minimize}} & \frac{1}{2}\|\Theta - S\|_F^2 + \lambda \|V - \text{Diag}(V)\|_{1,2} \\ \text{s.t.} & \Theta = V + V^T. \end{array} \tag{8.2.16}$$

An ADMM algorithm is proposed in [64] to solve (8.2.16). The ADMM algorithm requires an eigenvalue decomposition computation in each iteration. In this section, we describe the SBCD algorithm for hub covariance selection. The problem (8.2.16) can be reformulated as

$$\underset{V+V^T \succ 0}{\text{minimize}} \quad \frac{1}{2}\|V + V^T - S\|_F^2 + \lambda \|V - \text{Diag}(V)\|_{1,2}. \tag{8.2.17}$$

Due to the simplicity of the objective in (8.2.17), we can solve (8.2.17) in each column exactly (keeping all other columns fixed). The update for the i-th column is obtained by solving the problem

$$\min_u \frac{1}{2}\|V + V^T + ue_i^T + e_i u^T - S\|_F^2 + \lambda\|V + ue_i^T - \mathrm{Diag}(V + ue_i^T)\|_{1,2}. \qquad (8.2.18)$$

Let $u^{\setminus i}$ denote a copy of the vector u with the i-th coordinate set to zero. Then the optimization in (8.2.18) can be implemented by optimizing for u_i and $u^{\setminus i}$ separately. Let $\hat{u}$ denote the optimal solution to (8.2.18). Note that $\hat{u}_i = \frac{1}{2}(S_{ii} - 2V_{ii})$. Let $b = (V + V^T - S)_{\cdot i}$, $v = V_{\cdot i}$ and $y = v^{\setminus i} + u^{\setminus i}$. Then $\hat{u}^{\setminus i}$ can be obtained by solving

$$\underset{y}{\text{minimize}}\, \|b^{\setminus i} + y - v^{\setminus i}\|_2^2 + \lambda\|y\|_2. \qquad (8.2.19)$$

The optimal solution to (8.2.19) is given by

$$\hat{y} \;=\; \max\left(1 - \tfrac{\lambda}{2\|v^{\setminus i} - b^{\setminus i}\|_2}, 0\right)(v^{\setminus i} - b^{\setminus i}). \qquad (8.2.20)$$

Hence the solution $\hat{u}$ to (8.2.18) is given by

$$\hat{u}_j = \begin{cases} \frac{1}{2}(S_{ii} - 2V_{ii}) & \text{if } j = i \\ \hat{y}_j - v_j & \text{otherwise} \end{cases}. \qquad (8.2.21)$$

As for hub graphical lasso, in our implementation of Algorithm 11 below, we maintain and update the inverse $W = \Theta^{-1}$. Every time a column of V is updated, Θ incurs a rank-two update, and so, by using the Sherman-Morrison-Woodbury identity (see e.g. [70]), we can update W efficiently in $O(p^2)$.

8.2.6 Experimental results for hub covariance selection

We generated a covariance matrix Θ_0 with hubs similar to the Erdos-Renyi setup described in Section 4.2 of [64]. For any given problem dimension p, we let the number of hubs k in Θ_0 equal $0.03 \times p$. We generated the sample covariance matrix S from Θ_0 using a sample size $n = 0.1 \times p \times k$.

We use two metrics for comparison of algorithms: F-score and run-time in seconds.The F-score equals the harmonic mean of the precision (P) and recall (R), i.e., F-score $= \frac{2 \times P \times R}{P + R}$.

Algorithm 11 SBCD with active set heuristic for Hub covariance selection

Input: Sample covariance matrix, S, λ and optimality tolerance ϵ

Output: ϵ-approximate optimal solution $\hat{V}$

Initialize: $V = \text{Diag}(S)/2$, $\Theta = V + V^T = \text{Diag}(S)$, $W = (\Theta)^{-1} = (\text{Diag}(S))^{-1}$ and active set $\mathcal{A} = \{\}$.

While $\mathcal{A} \neq \{\}$

1 For $i = 1, 2, \ldots, p$

$$\mathbf{\Gamma}_{\cdot i} = \begin{cases} \lambda v^{\backslash i}/\|v^{\backslash i}\|_2 & \text{if } v^{\backslash i} \neq \mathbf{0} \\ (2\Theta_{\cdot i} - 2S_{\cdot i})^{\backslash i} & \text{if } v^{\backslash i} = \mathbf{0} \text{ and } \|(2\Theta_{\cdot i} - 2S_{\cdot i})^{\backslash i}\|_2 \leq \lambda \\ \lambda \frac{(2\Theta_{\cdot i} - 2S_{\cdot i})^{\backslash i}}{\|(2\Theta_{\cdot i} - 2S_{\cdot i})^{\backslash i}\|_2} & \text{if } v^{\backslash i} = \mathbf{0} \text{ and } \|(2\Theta_{\cdot i} - 2S_{\cdot i})^{\backslash i}\|_2 > \lambda. \end{cases} \tag{8.2.22}$$

2 Identify active set: $\mathcal{A} = \{i : \| - 2\Theta_{\cdot i} + 2S_{\cdot i} + \mathbf{\Gamma}_{\cdot i}\|_\infty \leq \epsilon\}$.

3 For $i \in \mathcal{A}^c$

(a) Set $d = \hat{u}$, where the $\hat{u}$ is given by (8.2.21).

(b) Choose a step size α that satisfies (8.1.4) with $u = d, w = e_i$, $\Theta^{-1} = W$.

(c) $V_{\cdot i} = V_{\cdot i} + \alpha d$.

(d) $\Theta = V + V^T$.

(e) Update $W = (\Theta)^{-1}$ using the Sherman-Morrison-Woodbury identity.

As mentioned earlier, higher the F-score, the better the performance of the algorithm on this metric. We now describe our experimental results.

1. In Figure 8.4, we plot the metrics against a range of regularization parameters λ for different choices of p. In particular, we consider $p = 500, 1000$. We vary λ as follows: Let $\lambda_{\max} = 2\|S - \mathrm{Diag}(S)\|_{\infty,2}$, where we define the $\|X\|_{\infty,2}$ norm as the maximum ℓ_2 norm of the columns of the X. We then generate a λ grid by dividing the interval $[0, \lambda_{\max}]$ into 30 equal parts. Denote these points by $\{\lambda_i\}$, $i = 1, \ldots, 30$. We compute the solution path starting at λ_{30} and ending at λ_1. We use warmstarts to speed up the computation of the solution path. If for some λ_i, the solution output by the algorithm has more than 10% non-zero columns, then we stop the computation of the solution path at this point. We set the optimality tolerance for all the algorithms to be 10^{-4}. As can be seen from the top panel of Figure 8.4, the F-scores match up for ADMM and SBCD. Indeed this makes sense, as the two algorithms are solving the same HGL formulation. The top panel reveals the regime of λ for which the F-score is high. In the bottom panel of Figure 8.4, we see that in precisely this good regime, SBCD is up to 100 times faster than the SVD based ADMM algorithm for a significant range of λ values.

2. We next compare the solution path time of ADMM and SBCD in Figure 8.5. As can be seen from the figure, SBCD is 200 times faster than ADMM for $p = 1000$. It is also interesting to note is that the solution path time taken by SBCD for $p = 4000$ is 3 times less than the time taken by ADMM for $p = 1000$. This illustrates that SBCD can be significantly faster than an SVD-based ADMM algorithm. The gains in run times are due to the absence of SVD factorizations in SBCD combined with the ability of SBCD to exploit the sparse solution structure as compared to the ADMM algorithm.

Our experimental results show that SBCD is indeed much faster than an SVD-based ADMM algorithm. However, we note that there could be potentially much faster algorithms

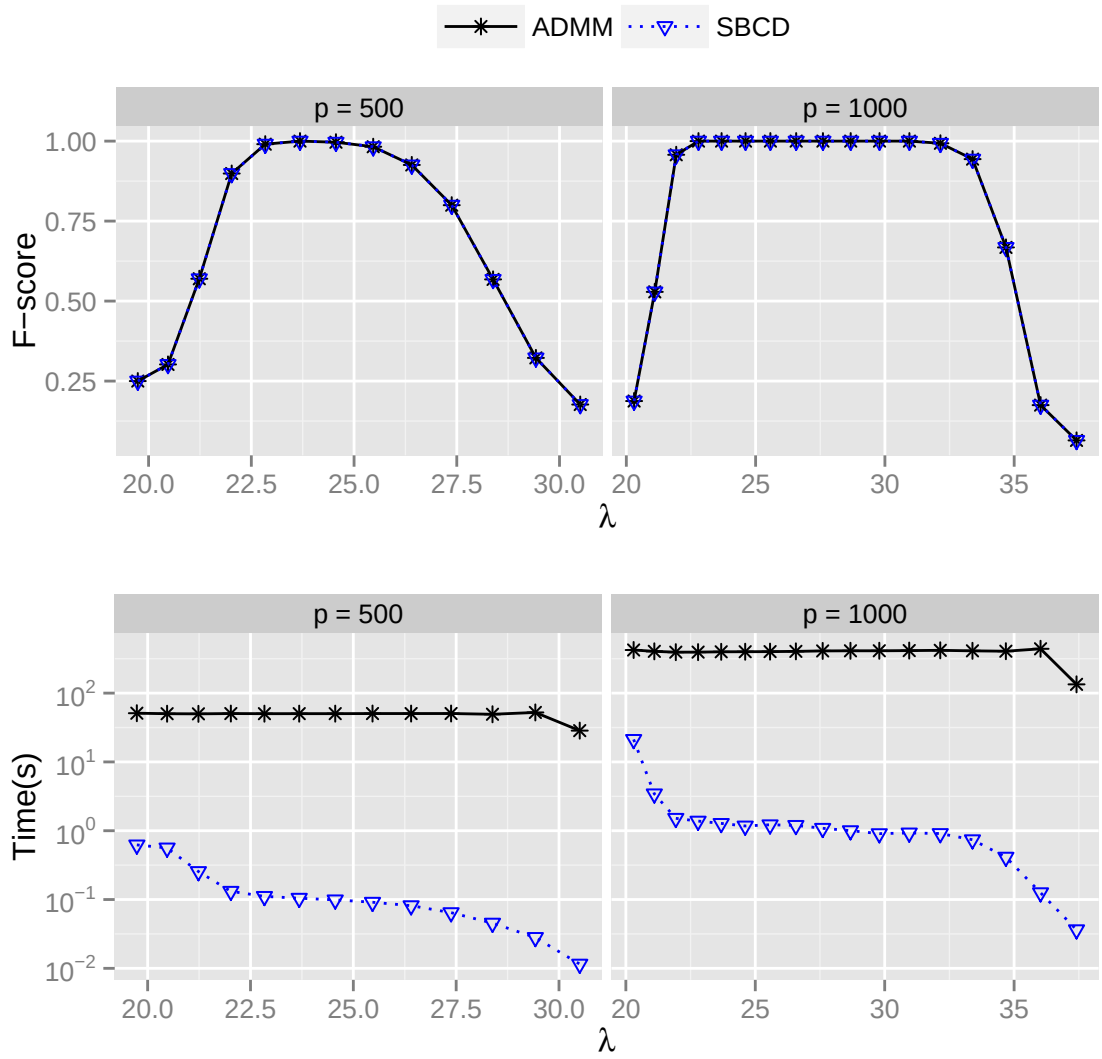

Figure 8.4: Comparison of ADMM and SBCD algorithms on the metrics of F-score and run time in seconds. The top panel shows F-score comparison of algorithms for $p = 500, 1000$ against λ. As expected, we see that the F-scores for the two algorithms match over the range of λ. The top panel is useful in deciding the range of λ in which the F-score is high. The bottom panel shows run time comparison of algorithms for $p = 500, 1000$ against λ. In the range of interest for λ, we see that SBCD is consistently faster than ADMM, sometimes by a factor of 100. The computation time for SBCD increases as λ decreases, while ADMM's computation time stays almost the same for all λ.

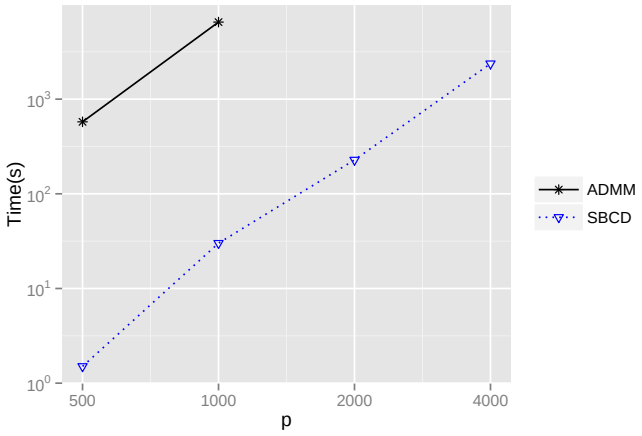

Figure 8.5: Solution path time of ADMM and SBCD vs p. SBCD is about 200 times faster than ADMM for $p = 1000$. Also note that SBCD for $p = 4000$ is 3 times for faster than ADMM for $p = 1000$.

than ADMM for the hub covariance selection problem (aside from our algorithm) and we leave more extensive comparison of algorithms for future work.

8.3 SBCD for Latent Variable Selection

Recall that the low-rank and sparse model proposed for latent variable selection problem [17] is given as follows:

$$\underset{Y,L\succeq 0}{\text{minimize}} \;\; \mathcal{F}(Y,L;S) := -\log\det(Y-L) + \langle Y-L, S\rangle + \lambda_1\|Y\|_1 + \lambda_2\,\text{trace}(L) \quad (8.3.1)$$

Our approach is to iteratively alternate the optimization of Y keeping L fixed and optimize L keeping Y fixed. The Y update follows from the Z update in Algorithm 7 with $g(V,Z) = \lambda_1\|Z\|_1 + 2\lambda_2(V)$ where $V + V^T := L$ and $Z := Y$. Our updating strategy for L is far more complex and depends on a low rank factorization.

8.3.1 Updating a low-rank factorization for L

Suppose we wish to update the ith row and column of L. Let T_i denote the $p \times p$ transposition matrix which exchanges the ith and pth rows of a matrix by left multiplication, i.e., TL is the matrix obtained from L by exchanging L's ith row with its last row. In particular, we have $T_i^{-1} = T_i = T_i^T$. Hence conjugation by T_i, $L \to T_i L T_i^T$, is also a similarity transformation on L wherein the objective of (8.3.1) is invariant under conjugation by T_i, i.e.

$$\mathcal{F}(Y, L; S) = \mathcal{F}(T_i Y T_i^T, T_i L T_i^T; T_i S T_i^T).$$

In the remaining, we will assume this transformation is applied when we update ith column and row of L where $i \neq p$. That is, given L and an index $i \in \{1, 2, \ldots, p\}$, we obtain an update to the last row and column of $L_i := T_i L T_i^T$, say ΔL_i, to obtain a new $\widetilde{L}_i := L_i + \Delta L_i$. Then the corresponding update to L is $T_i \Delta L_i T_i$. Hence, for every index $i \in \{1, 2, \ldots, p\}$, updates to the ith row and ith column of L can be completely described by describing a corresponding update to the last row and column of L.

Let

$$L := \begin{bmatrix} L_{11} & l_{12} \\ l_{12}^T & l_{22} \end{bmatrix},$$

with $L_{11} \in \mathcal{S}^p$, $l_{12} \in \mathbb{R}^p$, and $l_{22} \in \mathbb{R}$. We now describe an update to the last row and column of the form

$$\begin{pmatrix} l_{12} \\ l_{22} \end{pmatrix} \to \begin{pmatrix} l_{12} + \Delta l_{12} \\ l_{22} + \Delta l_{22} \end{pmatrix}. \tag{8.3.2}$$

We derived this update from a low-rank factorization of L. To this end let $L = WW^T$ be a low-rank factorization of L where W has full column rank and set

$$W = \begin{bmatrix} W_1 \\ w_2^T \end{bmatrix}.$$

We consider two cases depending on whether W_1 has full rank. If W_1 is not of full rank, then we need to derive from W another low rank factorization $L = \widetilde{W}\widetilde{W}^T$, where

$$\widetilde{W} := \begin{bmatrix} W_{11} & 0 \\ w_{21}^T & w_{22} \end{bmatrix}$$

with $\widetilde{W}$ of full column rank so that $w_{22} \neq 0$ and W_{11} is necessarily of full rank. The procedure for obtaining $\widetilde{W}$ is described below.

Low-rank factorization of L:

Let $L = WW^T$ be a low-rank factorization of L where W has full column rank.

Case (a): W_1 is full-rank. Then we set $\widetilde{W} = W$.

Case (b): W_1 is not full -rank. First compute the reduced QR factorization of W_1, say $W_1 = QR$, and the SVD of $RR^T = A\Sigma A^T$. Then the reduced SVD of $W_1 W_1^T = U\Sigma U^T$, where $U = QA$. Since W_1 is not full rank and W is full rank, then exactly one diagonal value of Σ is 0. Remove the 0 diagonal from Σ and the corresponding column of U and denote the resulting matrix as $\widetilde{\Sigma}$ and $\widetilde{U}$. Notice that $\widetilde{U}^T\widetilde{U} = I$. Let

$$\begin{aligned} \widetilde{W}_{11} &:= \widetilde{U}\widetilde{\Sigma}^{\frac{1}{2}}, \quad \widetilde{w}_{21} := \widetilde{\Sigma}^{-\frac{1}{2}}\widetilde{U}^T W_1 w_2 \\ \widetilde{w}_{12} &:= 0, \quad \widetilde{w}_{22} := \sqrt{w_2^T w_2 - \widetilde{w}_{21}^T \widetilde{w}_{21}}, \end{aligned} \tag{8.3.3}$$

it is easily verified that (8.3.3) satisfies that $L = WW^T = \widetilde{W}\widetilde{W}^T$.

Given this low-rank factorization set

$$(E, \widetilde{w}) = \begin{cases} (W_1, w_2), & \text{if } W_1 \text{ has full rank,} \\ (\widetilde{W}_{11}, \widetilde{w}_{21}), & \text{otherwise.} \end{cases} \tag{8.3.4}$$

Since $L = WW^T$ $(= \widetilde{W}\widetilde{W}^T)$, we have

$$L = \begin{bmatrix} L_{11} & l_{12} \\ l_{12}^T & l_{22} \end{bmatrix} = \begin{bmatrix} EE^T & E\widetilde{w} \\ (E\widetilde{w})^T & l_{22} \end{bmatrix}, \tag{8.3.5}$$

with $\mathrm{Ran}(L_{11}) = \mathrm{Ran}(E)$.

We construct the update (8.3.2) to maintain the positive semi-definiteness of

$$L_+ = \begin{bmatrix} L_{11} & l_{12} + \Delta l_{12} \\ (l_{12} + \Delta l_{12})^T & l_{22} + \Delta l_{22} \end{bmatrix}. \tag{8.3.6}$$

This places constraints on the choice of Δl_{12} and Δl_{22}. These constraints are derived from the well-known [] fact that

$$L \succeq 0 \text{ if and only if } L_{11} \succeq 0,\ l_{22} - l_{12}^T {L_{11}}^\dagger l_{12} \geq 0, \text{ and } l_{12} \in \mathrm{Ran}(L_{11}).$$

Since we assume that $L \succeq 0$, these conditions are satisfied by L and they establish the restrictions on Δl_{12} and Δl_{22} so that $L_+ \succeq 0$. Specifically, given $L \succeq 0$, then $L_+ \succeq 0$ if and only if

$$\Delta l_{12} \in \mathrm{Ran}(L_{11}) = \mathrm{Ran}(E) \quad \text{and} \tag{8.3.7}$$

$$l_{22} + \Delta l_{22} \geq (l_{12} + \Delta l_{12})^T L_{11}^\dagger (l_{12} + \Delta l_{12}). \tag{8.3.8}$$

We now use $(E, \widetilde{w})$ in (8.3.5) to re-express these conditions in a more convenient form. Note that $l_{12} = E\widetilde{w}$ and, since $\Delta l_{12} \in \mathrm{Ran}(E)$, we may write $l_{12} + \Delta l_{12} = E(\widetilde{w} + d_1)$ for some vector d_1. For consistency of notation, we define $d_2 := \Delta l_{22}$ giving

$$\begin{pmatrix} \Delta l_{12} \\ \Delta l_{22} \end{pmatrix} = \begin{bmatrix} E & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}.$$

If we now suppose that E has SVD $E = U\Sigma V^T$ with $VV^T = I$ (since E has full column rank), then $L_{11}^\dagger = U\Sigma^{-2}U^T$. With this notation, the inequality (8.3.8) becomes

$$l_{22} + d_2 \geq (\widetilde{w} + d_1)^T V\Sigma U^T U\Sigma^{-2}U^T U\Sigma V^T(\widetilde{w} + d_1) = \|\widetilde{w} + d_1\|_2^2 .$$

Hence the update L_+ in (8.3.6) can be written as

$$L_+ = L + D(d), \quad \text{where} \quad D(d) := \begin{bmatrix} 0 & Ed_1 \\ d_1^T E^T & d_2 \end{bmatrix} \quad \text{and} \quad d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix},$$

with the condition that

$$l_{22} + d_2 \geq \|\widetilde{w} + d_1\|_2^2 . \tag{8.3.9}$$

8.3.2 Computing L_+

Let

$$F(L) := -\log\det(Y - L) + \langle Y - L, S\rangle + \lambda_2 \,\mathrm{trace}(L). \tag{8.3.10}$$

For each $i = 1, \ldots, p$, define

$$d_H(L; i) := \operatorname*{argmin}_d \Big\{ f(d) := \langle \nabla F(L), D(d) \rangle + \frac{1}{2} d^T H d + \delta \left(d \mid L + D(d) \succeq 0 \right) \Big\}, \quad (8.3.11)$$

where i stands for the row-column index of L we are updating and H is a Hessian approximation to f at $d = 0$. When clear from the context, we will use d_H as the abbreviation for $d_H(L; i)$ and d^k for $d_{H^k}(L^k; i)$. Then we have the following descent lemma.

Lemma 8.3.1. *Given any $Y \succ L \succeq 0$ and $H \succ 0$, let d_H be as defined by* (8.3.11)*, then*

$$\langle \nabla F(L), D(d_H) \rangle \leq -d_H^T H d_H.$$

Proof. For any $\alpha \in (0, 1)$, first notice that $\delta \left(\alpha d_H \mid L + D(\alpha d_H) \succeq 0 \right) = 0$. Then we have

$$\begin{aligned} \langle \nabla F(L), D(d_H) \rangle + \frac{1}{2} d_H^T H d_H &\leq \langle \nabla F(L), D(\alpha d_H) \rangle + \frac{1}{2} (\alpha d_H)^T H (\alpha d_H) \\ &= \alpha \langle \nabla F(L), D(d_H) \rangle + \frac{1}{2} \alpha^2 d_H^T H d_H. \end{aligned}$$

Hence,

$$(1 - \alpha) \langle \nabla F(L), D(d_H) \rangle + \frac{1}{2} (1 - \alpha)(1 + \alpha) d_H^T H d_H \leq 0,$$

or, equivalently,

$$\langle \nabla F(L), D(d_H) \rangle + \frac{1}{2} (1 + \alpha) d_H^T H d_H \leq 0.$$

By letting $\alpha \uparrow 1$, we have

$$\langle \nabla F(L), D(d_H) \rangle \leq -d_H^T H d_H.$$

□

We now consider how to solve (8.3.11). By (8.3.9), problem (8.3.11) is equivalent to

$$\begin{array}{ll} \min\limits_d & \langle \nabla F(L), D(d) \rangle + \frac{1}{2} d^T H d \\ \text{s.t.} & d_2 + l_{22} \geq \| d_1 + \widetilde{w} \|^2 . \end{array} \quad (8.3.12)$$

Assume $\nabla F(L) := \begin{bmatrix} G & g \\ g^T & g_{22} \end{bmatrix}$, then $\langle \nabla F(L), D(d) \rangle = h^T d$ where $h := \begin{pmatrix} 2E^T g \\ g_{22} \end{pmatrix}$. In particular, if $H = \kappa I$, the objective in (8.3.12) can be written as

$$\begin{aligned} &\langle \nabla F(L), D(d) \rangle + \frac{1}{2} d^T H d \\ =& h^T d + \frac{\kappa}{2} d^T d = \frac{\kappa}{2} \left\| d + \frac{1}{\kappa} h \right\|_2^2 - \frac{1}{2\kappa} \| h \|_2^2 . \end{aligned}$$

Ignoring the constant term $-\frac{1}{2\kappa}\|h\|_2^2$ in the objective, (8.3.12) is equivalent to

$$\begin{aligned} \min_{d} \quad & \tfrac{\kappa}{2}\left\|d+\tfrac{1}{\kappa}h\right\|_2^2 \\ \text{s.t.} \quad & d_2 + l_{22} \geq \|d_1 + \widetilde{w}\|^2 . \end{aligned} \tag{8.3.13}$$

Now let $\tilde{d} := d + u$ with $u := \begin{pmatrix} \widetilde{w} \\ l_{22} \end{pmatrix}$. With this change of variable, we have (8.3.13) equivalent to

$$\begin{aligned} \min_{\tilde{d}} \quad & \tfrac{\kappa}{2}\left\|\tilde{d}-u+\tfrac{1}{\kappa}h\right\|_2^2 \\ \text{s.t.} \quad & \tilde{d}_2 \geq \left\|\tilde{d}_1\right\|^2 . \end{aligned} \tag{8.3.14}$$

Solving (8.3.14) is equivalent to evaluating the projection of $u-\frac{1}{\kappa}h$ onto $\Omega := \left\{ d \;\middle|\; d_2 \geq \|d_1\|^2 \right\}$.

We now show that the projection onto Ω can be computed in closed form. For this consider the projection problem

$$P_\Omega(\bar{d}_2, \bar{d}_1) := \operatorname*{argmin}_{d\in\Omega} \frac{1}{2}(d_2 - \bar{d}_2)^2 + \frac{1}{2}\|d_1 - \bar{d}_1\|_2^2. \tag{8.3.15}$$

The KKT conditions for (8.3.15) give

$$\begin{aligned} d_2 - \bar{d}_2 - \lambda &= 0 \\ d_1 - \bar{d}_1 + 2\lambda d_1 &= 0 \\ \lambda(d_1^T d_1 - d_2) &= 0 \\ \lambda &\geq 0. \end{aligned}$$

1. If $\bar{d}_2 \geq \|\bar{d}_1\|_2^2$, then $P_\Omega(\bar{d}_2, \bar{d}_1) = (\bar{d}_2, \bar{d}_1)$.

2. If $\bar{d}_2 < \|\bar{d}_1\|_2^2$, then $P_\Omega(\bar{d}_2, \bar{d}_1) = (\bar{d}_2 + \hat{\lambda}, \frac{\bar{d}_1}{1+2\hat{\lambda}})$, where $\hat{\lambda}$ is given by the positive root of the cubic equation $(\bar{d}_2 + \lambda)(1 + 2\lambda)^2 - \|\bar{d}_1\|_2^2 = 0$ (this cubic has two complex and one positive real root). If we set $\xi := 1/27 + \|\bar{d}_1\|_2^2 - \bar{d}_2$, then

$$\hat{\lambda} = \frac{1}{2}\left[\left[(\xi + \left(\xi^2 - \frac{1}{3^6}\right)^{1/2}\right]^{1/3} + \left[(\xi - \left(\xi^2 - \frac{1}{3^6}\right)^{1/2}\right]^{1/3}\right] - \frac{1}{3}.$$

After solving (8.3.11), we do a line search with Armijo rule to find the step size as stated in Algorithm 12.

Algorithm 12 Armijo rule for one row-column of L update

1: Initialization: Line search parameters $\beta \in (0,1)$, $\sigma \in (0,1)$, $\gamma \in [0,1)$, and update data $L^k \in \mathcal{S}^p_+$, $H^k \in \mathcal{S}^p_{++}$ and $d^k_H \in \mathbb{R}^p$ satisfying

$$\Delta^k := \langle \nabla F(L^k), D(d^k_H) \rangle + \gamma (d^k_H)^T H^k d^k_H < 0. \tag{8.3.16}$$

2: Set

$$\begin{aligned} \alpha^k \;\; := \;\; & \max_{s=0,1,\cdots} \beta^s \\ & \text{s.t. } F(L^k + \beta^s D(d^k_H)) \leq F(L^k) + \beta^s \sigma \Delta^k. \end{aligned}$$

3: Set $L^{k+1} = L^k + D(\alpha^k d^k_H)$.

8.3.3 Update of the low-rank factorization for L

Assume that we start with an L such that $\text{rank}(L) = r \ll p$ and its low rank factorization is $L = \widetilde{W}\widetilde{W}^T$ where $\widetilde{W} \in \mathbb{R}^{p \times r}$. The output of Algorithm 12 is

$$\hat{L} := \begin{bmatrix} EE^T & E\hat{w} \\ \hat{w}^T E^T & \hat{l}_{22} \end{bmatrix}, \tag{8.3.17}$$

where $\hat{w} = \widetilde{w} + \alpha d_1$, $\hat{l}_{22} = l_{22} + \alpha d_2$ and $(E, \widetilde{w})$ is as defined in (8.3.4). We now compute a low rank factorization of $\hat{L}$of the form

$$\hat{L} := \widehat{W}\widehat{W}^T = \begin{bmatrix} EE^T & E\hat{w} \\ \hat{w}^T E^T & \hat{l}_{22} \end{bmatrix}. \tag{8.3.18}$$

with $\widehat{W}$ of full column rank. To find $\widehat{W}$, there are two cases to consider.

Case (a): If $\hat{l}_{22} > \hat{w}^T \hat{w}$, then it is easy to verify that

$$\widehat{W} = \begin{bmatrix} E & 0 \\ \hat{w} & \sqrt{\hat{l}_{22} - \hat{w}^T \hat{w}} \end{bmatrix}, \tag{8.3.19}$$

satisfies $\hat{L} = \widehat{W}\widehat{W}^T$.

Case (b): If $\hat{l}_{22} = \hat{w}^T\hat{w}$, then it is easy to verify that

$$\widehat{W} = \begin{bmatrix} E \\ \hat{w} \end{bmatrix} \tag{8.3.20}$$

satisfies $\hat{L} = \widehat{W}\widehat{W}^T$.

Note that due to the way the matrix E is defined in (8.3.4), this update to $\widehat{W}$ can either leave the rank of L unchanged, or it can increase or decrease the rank by 1. The algorithm for updating one row-column of L is given in Algorithm 13. The overall algorithm for optimizing L is given in Algorithm 14. Next, we show that Algorithm 14 converges to a stationary point of a non convex problem.

Algorithm 13 Algorithm for the one row-column update of L

Input: Row-column index i of L to be updated. L and a low-rank factorization of $L = WW^T$. The inverse $Q := (Y - L)^{-1}$. Line search parameters $\beta \in (0, 1)$, $\sigma \in (0, 1)$, $\gamma \in [0, 1)$.

Output: $\hat{L} = \widehat{W}\widehat{W}^T$, $\widehat{Q} := (Y - \hat{L})^{-1}$ and $d_H(L; i)$.

1. Obtain a new factorization $L = \widetilde{W}\widetilde{W}^T$ as outlined in Section 8.3.1.

2. Solve the sub-problem (8.3.11) to get $d_H(L; i)$ and run Algorithm 12 to obtain $\hat{L}$.

3. Update the factorization for $\hat{L} = \widehat{W}\widehat{W}^T$.

   Case (a): $\hat{l}_{22} > \hat{w}^T\hat{w}$. Set $\widehat{W} = \begin{bmatrix} E & 0 \\ \hat{w}^T & \sqrt{\hat{l}_{22} - \hat{w}^T\hat{w}} \end{bmatrix}$.

   Case (b): $\hat{l}_{22} = \hat{w}^T\hat{w}$. Set $\widehat{W} = \begin{bmatrix} E \\ \hat{w}^T \end{bmatrix}$.

8.3.4 Convergence analysis for Algorithm 14

Let $L^0 \succeq 0$ be the initial value of L with $F(L^0) < \infty$. Denote

$$C := \left\{ L \,\middle|\, F(L) \leq F(L^0),\ L \succeq 0 \right\}.$$

Algorithm 14 SBCD for optimizing L

1: Initialization L^0, $\beta \in (0,1)$, $\sigma \in (0,1)$, $\gamma \in [0,1)$, $k=0$, $\tau=0$ and tolerance constant $\epsilon > 0$.

2: Compute $Q^k := (Y - L^k)^{-1}$ using matrix inversion lemma.

3: For $i = 1, \cdots, p$,

(i) Run Algorithm 13 with $L^k = W^k(W^k)^T$, index i, $Q^k := (Y - L^k)^{-1}$ and line search parameter $\beta \in (0,1)$, $\sigma \in (0,1)$, $\gamma \in [0,1)$ to obtain $L^{k+1} = W^{k+1}(W^{k+1})^T$ and $d^k := d_{H^k}(L^k; i)$.

(ii) Set $\tau = \tau + \left\|d^k\right\|$ and $k = k+1$.

4: If $\tau < \epsilon$, stop; Else set $\tau = 0$ and return to Step 2.

Proposition 8.3.2. *The mapping $\nabla F : \mathcal{S}^p \to \mathbb{R} \cup \{+\infty\}$ is Lipschitz continuous on C, i.e. $\exists K > 0$ such that*

$$\|\nabla F(L_1) - \nabla F(L_2)\| \leq K \|L_1 - L_2\| \quad \forall L_1, L_2 \in C.$$

Proof. It is easily seen that $\exists \epsilon > 0$ such that $\forall L \in C$, $Y \succeq Y - L \succeq \epsilon I$. Therefore,

$$\begin{aligned}
&\|\nabla F(L_1) - \nabla F(L_2)\| \\
&= \left\|(Y - L_1)^{-1} - (Y - L_2)^{-1}\right\| \\
&= \left\|(Y - L_1)^{-1}(L_2 - L_1)(Y - L_2)^{-1}\right\| \\
&\leq \left\|(Y - L_1)^{-1}\right\| \left\|(Y - L_2)^{-1}\right\| \|L_2 - L_1\| \\
&\leq K(\epsilon) \|L_2 - L_1\|,
\end{aligned}$$

where $K(\epsilon) \uparrow \infty$ as $\epsilon \downarrow 0$. □

In the rest of this chapter, it is assumed that $\nabla F(L)$ is Lipschitz continuous with constant K which depends on L^0 only. For an index i, let Y_i be the sub-matrix of Y obtained by deleting the i-th column and row from Y. Define

$$\eta = \max\{1, \lambda_{\max}(Y_i) \,:\, i \in \{1, \cdots, p\}\}.$$

Lemma 8.3.3. *Let $L \in C$ and $i \in \{1, \ldots, p\}$. Assume $H \succeq \underline{\lambda} I$ and $d := d_H(L; i)$ is given by* (8.3.11)*. If $\alpha \in [0, (1 - \sigma + \sigma\gamma)\underline{\lambda}/(K\eta)]$, then*

$$F(L + \alpha D(d)) \leq F(L) + \alpha\sigma\Delta,$$

where Δ is defined in (8.3.16)*.*

Proof. The Fundamental Theorem of Calculus and Lipschitz continuity of ∇F imply that

$$\begin{aligned}
&F(L + \alpha D(d)) - F(L) \\
=&\alpha\langle \nabla F(L), D(d)\rangle + \int_0^1 \langle \nabla F(L + t\alpha D(d)) - \nabla F(L), \alpha D(d)\rangle dt \\
\leq&\alpha\langle \nabla F(L), D(d)\rangle + \alpha \int_0^1 \|\nabla F(L + t\alpha D(d)) - \nabla F(L)\| \, \|D(d)\| \, dt \\
\leq&\alpha\langle \nabla F(L), D(d)\rangle + \frac{1}{2} K\alpha^2 \|D(d)\|^2 \\
=&\alpha\langle \nabla F(L), D(d)\rangle + \frac{1}{2} K\alpha^2 (2\|Ed_1\|^2 + d_2^2) \qquad \left(Note : d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \right) \\
\leq&\alpha\langle \nabla F(L), D(d)\rangle + \frac{1}{2} K\alpha^2 (\lambda_{\max}(Y_i) 2\|d_1\|^2 + d_2^2) \quad (Note : Y_i \succ EE^T) \\
\leq&\alpha\langle \nabla F(L), D(d)\rangle + \frac{1}{2} K\alpha^2 \eta (2\|d_1\|^2 + d_2^2) \\
\leq&\alpha\langle \nabla F(L), D(d)\rangle + K\alpha^2 \eta \|d\|^2 \\
=&\alpha\Delta - \alpha\gamma d^T H d + K\alpha^2 \eta \|d\|^2 .
\end{aligned}$$

If $\alpha \in [0, (1 - \sigma + \sigma\gamma)\underline{\lambda}/(K\eta)]$, then

$$\begin{aligned}
&- \gamma d^T H d + K\alpha\eta \|d\|^2 \\
\leq& - \gamma d^T H d + (1 - \sigma + \sigma\gamma) d^T H d \\
=&(1 - \sigma)(1 - \gamma) d^T H d \\
\leq& - (1 - \sigma)\Delta,
\end{aligned}$$

where the last inequality follows the definition of Δ in (8.3.16) and Lemma 8.3.1 so that

$$\Delta := \langle \nabla F(L), D(d)\rangle + \gamma d^T H d \leq -(1 - \gamma) d^T H d.$$

Hence

$$\begin{aligned}
&F(L+\alpha D(d))-F(L)\\
\leq&\alpha\Delta-\alpha\gamma d^THd+K\alpha^2\eta\left\|d\right\|^2\\
\leq&\alpha\Delta-\alpha(1-\sigma)\Delta\\
=&\alpha\sigma\Delta
\end{aligned}$$

□

Theorem 8.3.4. *Assume $\bar{\lambda}I \succeq H^k \succeq \underline{\lambda}I$, where $\bar{\lambda} \geq \underline{\lambda} > 0$. Let $\sigma \in (0,1)$, $\gamma \in [0,1)$, $\beta \in (0,1)$. Suppose $\{L^k, W^k, H^k d^k\}$ is generated by Algorithm 14. Let $(\bar{L}, \bar{W})$ be any cluster point of $\{(L^k, W^k)\}$, where, by construction, $L^k = W^k(W^k)^T$. Then the following hold:*

(i) $d^k \to 0$.

(ii) For $i \in \{1, \cdots, p\}$, let $\bar{H}^i$ denote a cluster point of $\{H^{i-1+jp}\}_{j=0}^{\infty}$, and

$$d_{\bar{H}^i}(\bar{L}; i) = \operatorname*{argmin}_d \left\{ \langle \nabla F(\bar{L}), D(d) \rangle + \frac{1}{2} d^T \bar{H}^i d + \delta\left(d \,\middle|\, \bar{L} + D(d) \succeq 0\right) \right\}, \quad (8.3.21)$$

then $d_{\bar{H}^i}(\bar{L}; i) = 0$.

(iii) For all d such that $\bar{L} + D(d) \succeq 0$, we have

$$\langle \nabla F(\bar{L}), D(d) \rangle \geq 0.$$

(iv) $\bar{W} \in \mathbb{R}^{p \times r}$ is a stationary point of

$$\min_{W \in \mathbb{R}^{p \times r}} G(W) := F(WW^T). \quad (8.3.22)$$

Proof. (i) Note that

$$F(L^{k+1}) - F(L^k) \leq \alpha^k \sigma \Delta^k \leq -\alpha^k \sigma(1-\gamma) d^k H^k d^k, \quad (8.3.23)$$

where the second inequality follows from Lemma 8.3.1. Summing (8.3.23) from $k = 0$ to N, we have

$$F(L^{N+1}) - F(L^0) \leq \sum_{k=0}^{N} -\alpha^k \sigma(1-\gamma) d^k H^k d^k.$$

Since $H^k \succeq \underline{\lambda} I$, we have

$$F(L^{N+1}) - F(L^0) \leq \sum_{k=0}^{N} -\alpha^k \sigma(1-\gamma)\underline{\lambda} \left\| d^k \right\|^2.$$

By Lemma 8.3.3 implies that $\hat{\alpha} := \inf \alpha^k > 0$, which gives

$$F(L^{N+1}) - F(L^0) \leq -\hat{\alpha}\sigma(1-\gamma)\underline{\lambda} \sum_{k=0}^{N} \left\| d^k \right\|^2. \tag{8.3.24}$$

Since $F(L) > -\infty$ on C, we can take the limit as $N \uparrow \infty$ to find that

$$\sum_{k=0}^{\infty} \left\| d^k \right\|^2 < \infty.$$

Hence, in particular, $d^k \to 0$ and

(ii)
$$\sum_{k=0}^{\infty} \left\| d^k \right\|^2 = \sum_{i=1}^{p} \sum_{j=0}^{\infty} \left\| d_H(L^{i-1+jp}; i) \right\|^2 < \infty,$$

hence, for any index i, $d_H(L^{i-1+jp}; i) \to 0$ as $j \to \infty$. Moreover,

$$\begin{aligned} &\langle \nabla F(L^{i-1+jp}), D(d_H(L^{i-1+kp}; i)) \rangle + \frac{1}{2}(d_H(L^{i-1+jp}; i))^T H^{i-1+jp} d_H(L^{i-1+jp}; i) \\ &\leq \langle \nabla F(L^{i-1+jp}), D(d) \rangle + \frac{1}{2} d^T H^{i-1+jp} + \delta\left(d \,\middle|\, L^{i-1+jp} + D(d) \succeq 0\right) \quad \forall\, d \in \mathbb{R}^p. \end{aligned}$$

By letting $j \to \infty$ along the subsequence for which $(L^k, W^k) \to (\bar{L}, \bar{W})$ and using continuity, we have

$$0 \leq \langle \nabla F(\bar{L}), D(d) \rangle + \frac{1}{2} d^T \bar{H}^i d + \delta\left(d \,\middle|\, \bar{L} + D(d) \succeq 0\right) \quad \forall\, d \in \mathbb{R}^p.$$

Therefore, by definition, $d_H(\bar{L}; i) = 0 \quad \forall i = 1, \cdots, p.$

(iii) Follows directly from (ii).

(iv) Assume not, then $\exists D_W \in \mathbb{R}^{p \times r}$ such that

$$\langle \nabla G(\bar{W}), D_W \rangle < 0.$$

Let $D_W^i \in \mathbb{R}^{p\times r}$ denote a zero matrix with the i-th row replaced by the i-th row of D_W. Then for at least one $i \in \{1, \cdots, p\}$, WLOG, say p, such that

$$\langle \nabla G(\bar{W}), D_W^p \rangle < 0.$$

Then $\exists \alpha > 0$, such that $G(\bar{W} + \alpha D_W^p) < G(\bar{W})$. We absorb α into D_W^p, i.e. $G(\bar{W} + D_W^p) < G(\bar{W})$. Let $\widetilde{L} := (\bar{W} + D_W^p)(\bar{W} + D_W^p)^T \succeq 0$ and $\widetilde{L}$ differs from $\bar{L}$ only on the last row and column, and $F(\widetilde{L}) < F(\bar{L})$. Contradiction to (iii).

□

Remark 8.3.5. The dimension of $\bar{W}$ is not pre-determined. Algorithm 14 is different from solving the non-convex formulation (8.3.22) with a fixed dimension for W.

8.3.5 Final tuning of rank by singular value shrinkage

Once Algorithm 14 has terminated, we apply a simple procedure for reducing the rank of L even further. For this, we compute the reduced eigenvalue decomposition of $L = U\Sigma U^T$ from its low rank factorization $L = WW^T$ as in §8.3.1, where $\Sigma = \mathrm{diag}\{\sigma_1^2, \sigma_2^2, \cdots, \sigma_r^2\}$, $U \in \mathbb{R}^{p\times r}$ and $U^TU = I_r$. Next consider the problem

$$\begin{array}{ll} \min\limits_{\alpha_i} & -\log\det(Y - L - \alpha_i u_i u_i^T) + \langle Y - L - \alpha_i u_i u_i^T, S\rangle + \lambda_2 \,\mathrm{trace}(L + \alpha_i u_i u_i^T) \\ \text{s.t.} & \alpha_i \geq -\sigma_i^2, \end{array}$$

or equivalently, the problem

$$\begin{array}{ll} \min\limits_{\alpha_i} & -\log(1 - \alpha_i u_i^T (Y - L)^{-1} u_i) + \alpha_i(\lambda_2 \left\|u_i\right\|_2^2 - u_i^T S u_i) \\ \text{s.t.} & \alpha_i \geq -\sigma_i^2, \end{array} \tag{8.3.25}$$

Define

$$\bar{\alpha} = \frac{1}{u_i^T (Y - L)^{-1} u_i} + \frac{1}{\lambda_2 \left\|u_i\right\|_2^2 - u_i^T S u_i}.$$

If $\bar{\alpha} \geq -\sigma_i^2$, then $\alpha_i = \bar{\alpha}$ solves (8.3.25). Otherwise $\alpha_i = -\sigma_i^2$ solves (8.3.25).

8.3.6 Practical version of SBCD (P-SBCD)

The main computation in Algorithm 14 occurs while updating $(Y - L)^{-1}$ and $(Y_{11} - L_{11})^{-1}W_{11}$ which is $O(rp^2)$ for a single row-column update of L. Hence one sweep of all

row-columns of L requires $O(rp^3)$ operations, which is even more expensive than an SVD based method.

To make our algorithm more efficient, we do not apply Algorithm 14. Instead we fix $Y - L$ and apply Algorithm 13 sweeping all p row-columns of L. At the end of the sweep, we have $\hat{L} = \widehat{W}\widehat{W}^T$. This gives a search direction $D_W := \widehat{W} - W$ which is a non-negative linear combination of descent directions for $G(W) := F(WW^T)$. Hence D_W is a descent direction for G as well. Consequently, we can think of D_W as the basis for a column update to W and perform this update column by column. Observe that each execution of Algorithm 13 requires $O(pr^2)$ operations, and so one sweep of all row-columns of L takes $O(p^2r^2)$ operations. Next we update W column by column with a line search (described below). This takes a total of $O(p^2r^2)$ operations, which is much better than $O(rp^3)$ when $r \ll p$.

We now describe the column-by-column line search procedure. Let $D_{\cdot i}$ denote the i-th column of D_W. A backtracking line search is employed to find a step size α such that an Armijo condition is satisfied. A key requirement is the need to preserve the positive definiteness of $Y - (W + \alpha D_{\cdot i}e_i^T)(W + \alpha D_{\cdot i}e_i^T)^T$. The following proposition addresses this issue.

Proposition 8.3.6. *Let $W \in \mathbb{R}^{p\times r}$ and $i \in \{1, \ldots, r\}$, and let w denote the i-th column of W. Assume $Y - WW^T \succ 0$, $Q := (Y - WW^T)^{-1}$, and $\bar{W} := W + de_i^T$, then $Y - \bar{W}\bar{W}^T \succ 0$ if and only if $(1 - d^TQw)^2 - (d^TQd)(w^TQw + 1) > 0$.*

Proof. Observe that

$$\begin{aligned} Y - \bar{W}\bar{W}^T &= Y - WW^T - dw^T - w^Td - dd^T \\ &= Y - WW^T - d(w + \frac{d}{2})^T - (w + \frac{d}{2})d^T. \end{aligned}$$

Theorem 8.1.3 implies that $Y - \bar{W}\bar{W}^T \succ 0$ if and only if $\det(Y - \bar{W}\bar{W}^T) > 0$. Direction computation shows that

$$\det(Y - \bar{W}\bar{W}^T) = \det(Y - WW^T)\Big[(1 - d^TQw)^2 - (d^TQd)(w^TQw + 1)\Big],$$

hence the result follows. □

Let us now examine the cost of one sweep of algorithm 13 across all p row-columns of L. During this sweep, the matrix L remains fixed, and so the initial low rank factorization of L remains fixed throughout the sweep. This reduces the total cost incurred. In particular, the computation of the Armijo stepsize depends on the evaluation of

$$F(L + \alpha D(d)) - F(L).$$

In our practical version of SBCD, this can be done in $O(r^2)$. Indeed, direct calculation shows that

$$\begin{aligned} F(L + \alpha D(d)) - F(L) = \quad & -\log[(1 - \alpha d^T B^T Q e_p)^2 - \alpha^2 (d^T B^T Q B d)(e_p^T Q e_p)] \\ & -2\alpha d^T B^T S e_p + 2\alpha\lambda_2 d^T B^T e_p, \end{aligned} \tag{8.3.26}$$

where $B = \begin{bmatrix} E & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$ and $Q = (Y - L)^{-1}$. The expression (8.3.26) shows that the evaluation of objective function $F(L + \alpha D(d)) - F(L)$ requires $B^T Q B$, $B^T Q e_p$ and $B^T S e_p$. These matrices can be obtained from $K := W^T Q W$ and $A := W^T Q$ with $O(r^2)$ operations since W remains fixed during the sweep. To see this, let $Q = \begin{bmatrix} Q_{11} & q_{12} \\ q_{12}^T & q_{22} \end{bmatrix}$, then

$$B^T Q B = \begin{bmatrix} E^T Q_{11} E & \frac{1}{2} E^T q_{12} \\ \frac{1}{2} q_{12}^T E & \frac{1}{4} q_{22} \end{bmatrix}, \quad B^T Q e_p = \begin{bmatrix} E^T q_{12} \\ \frac{1}{2} q_{22} \end{bmatrix}.$$

Hence we need only compute $E^T Q_{11} E$ and $E^T q_{12}$. Moreover,

$$\begin{aligned} E^T Q_{11} E &= K - A_{\cdot p} w_2^T - w_2^T A_{\cdot p} + q_{22} w_2 w_2^T \quad \text{and} \\ E^T q_{12} &= A_{\cdot p} - q_{22} w_2, \end{aligned}$$

that is $B^T Q B$ and $B^T Q e_p$ can be obtained from K and A with $O(r^2)$ operations.

8.3.7 Numerical Experiments

In this section, we compare P-SBCD with ADMM on a series of synthetic problems. We generate our data in the following way. Y is a symmetric $p \times p$ random Gaussian sparse matrix with density 2%, i.e. 2% non-zero elements. X is $p \times r$ random Gaussian matrix.

And let $L = XX^T/900$. Then add a multiple of identity αI to Y to make $Y - L \succ 0$, where α equals the absolute value of the minimum eigenvalue of $Y - L$ plus 1. We use $Y - L$ as the true precision matrix to generate a sample covariance matrix S with sample size N. We terminate both algorithms if the reduction of the objective function is less than 10^{-3}. We implement the ADMM with continuation as introduced in [46].

1. In the first set of experiments, we generate 5 problems with $p = 1000, 2000, 3000, 4000, 6000$ and $r = 10$ with sample size $N = p$. For $p = 6000$, we terminate ADMM after 10 hours, so we do not have the result for ADMM in this case. Figure 8.6 shows that P-SBCD is at least 10 times faster for the first 4 problems. Table 8.1 presents the objective function values of both algorithms. P-SBCD's objective is slightly better than ADMM. Both algorithms take around 50 iterations to stop. The speedup of P-SBCD comes from the low per-iteration computation cost.

p	λ_1	λ_2	ADMM	P-SBCD
1000	0.0080	0.3200	-1243.8724	-1243.8694
2000	0.0050	0.2800	-3040.2937	-3040.3240
3000	0.0028	0.2400	-5229.9635	-5230.0703
4000	0.0020	0.2100	-7565.1355	-7565.3030
6000	0.0015	0.1800	NA	-12516.9303

Table 8.1: Objective comparison of ADMM and P-SBCD for different dimension p.

2. In the second set of experiments, we test both algorithms on different sample sizes with $p = 2000$. Let $\gamma = 1, 2, 10, 50$, then we use $N = \gamma p$ as the sample size. Figure 8.8 shows that P-SBCD is at least 10 times faster than ADMM. Table 8.2 presents the objective function values of both algorithms. P-SBCD's objective is slightly better than ADMM. Figure 8.9 presents the sparsity of Y and rank of L for the final solution of both algorithms. We see that P-SBCD converges to solution with larger sparsity and smaller rank. Notice that as the sample size increases, the rank discrepancy between

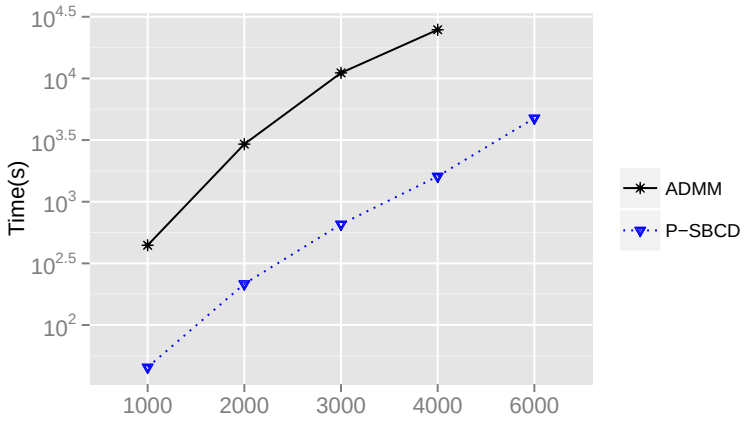

Figure 8.6: Time comparison of ADMM and P-SBCD for p = 1000, 2000, 3000, and 4000. From the picture, we see that P-SBCD is 10 times faster than ADMM. Notice the run-time of P-SBCD solving $p = 6000$ is only 20% of the CPU time of ADMM solving $p = 4000$.

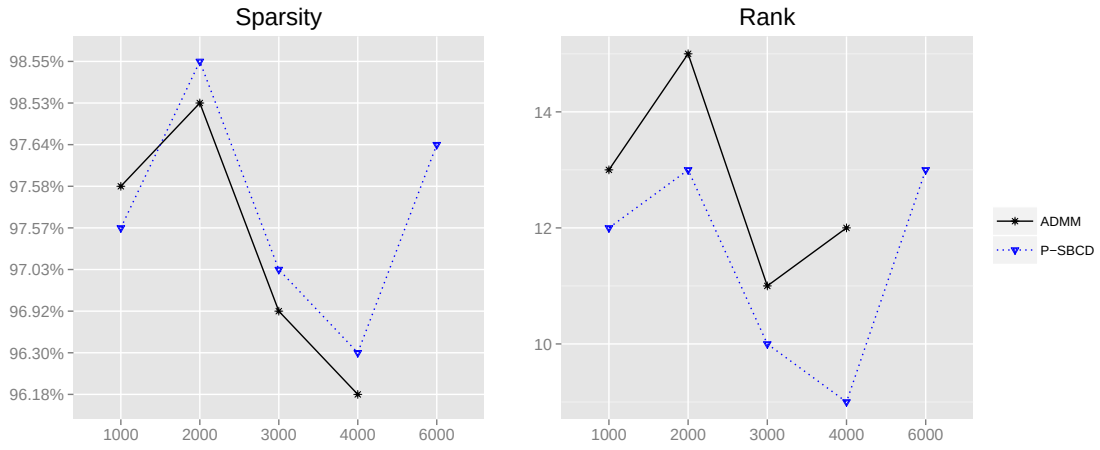

Figure 8.7: P-SBCD converges to a solution with smaller rank of L and larger sparsity of Y when $p = 2000, 3000, 4000$. When $p = 1000$, the sparsity of Y for P-SBCD is 97.57%, while the sparsity of Y for ADMM is 97.58%.

ADMM and P-SBCD is decreased. For sample sizes $10p$ and $50p$, both algorithms converge to an L with the same rank.

γ	λ_1	λ_2	ADMM	P-SBCD
1	0.0050	0.2800	-3040.2937	-3040.3240
2	0.0035	0.1800	-3164.0180	-3164.0235
10	0.0030	0.1300	-3144.5471	-3144.5878
50	0.0030	0.1200	-3122.1790	-3122.1955

Table 8.2: Objective comparison of ADMM and P-SBCD for different sample sizes of $p = 2000$

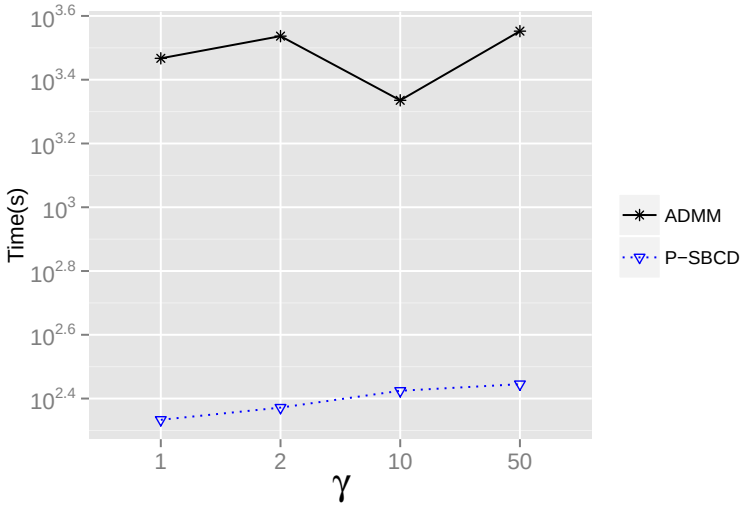

Figure 8.8: Time comparison of ADMM and P-SBCD for $\gamma = 1, 2, 10, 50$ and $p = 2000$. From the picture, we see that P-SBCD is around 10 times faster than ADMM.

3. In the third experiment, we test the effect of λ_2 by fixing λ_1 for $p = 2000, N = p$. Set $\lambda_1 = 0.005$ and $\lambda_2 = 0.27, 0.29, 0.31, 0.33, 0.35$. We solve these problems using

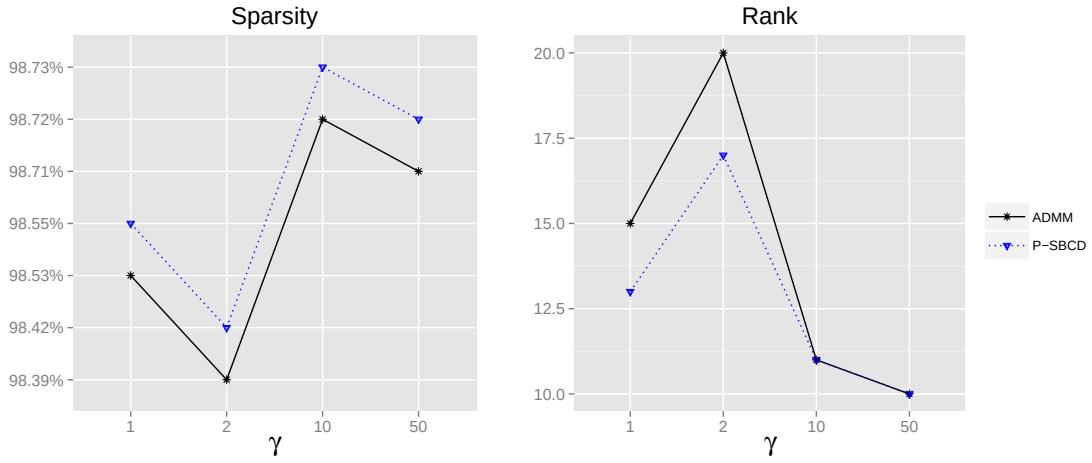

Figure 8.9: P-SBCD converges to solution with larger sparsity and smaller rank. Notice that as the sample size increases, the rank discrepancy is decreased. For sample size $10p$ and $50p$, both algorithms converge to an L with the same rank.

P-SBCD only. Figure 8.10 shows the rank of L and the sparsity of Y. As λ_2 increases, the rank of L and the sparsity of Y are decreased. The CPU time to solve each of these problems by P-SBCD is similar to the first and second experiments for $p = 2000$.

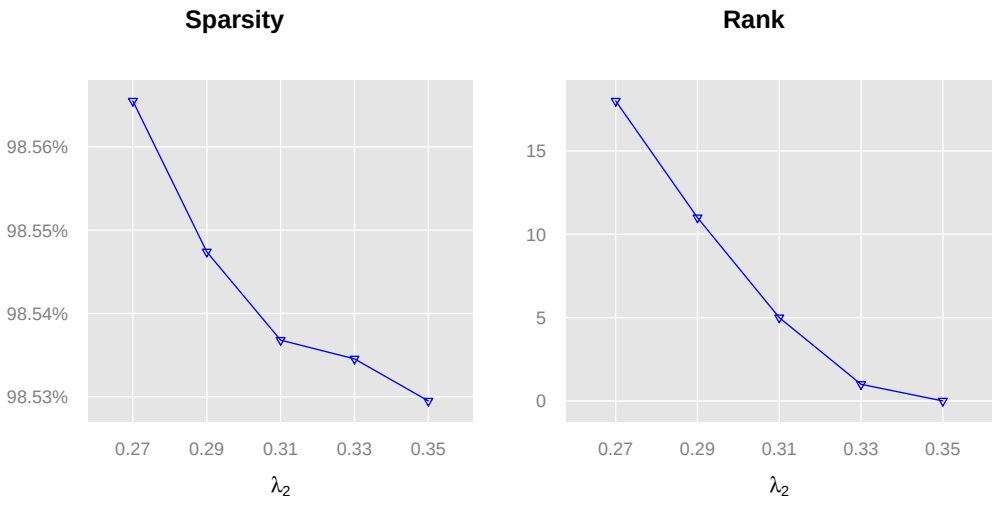

Figure 8.10: As λ_2 increases, the rank of L is decreased and the sparsity of Y as well.

BIBLIOGRAPHY

[1] M. S. Andersen, J. Dahl, and L. Vandenberghe. CVXOPT: A Python package for convex optimization, Version 1.1.6, 2013.

[2] D. H. Anderson and M. R. Osborne. Discrete, nonlinear approximation problems in polyhedral norms. *Numerische Mathematik*, 28:143–156, 1977.

[3] A. E. Beaton and J. W. Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectrographic data. *Technometrics*, 16:147–185, 1974.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2011.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and trends in machine learning*, 3(1):1–122, 2011.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 12th edition, 2013.

[7] J. V. Burke. Descent methods for composite nondifferentiable optimization problems. *Mathematical Programming*, 33:260–279, 1985.

[8] J. V. Burke. Second order necessary and sufficient conditions for convex composite NDO. *Mathematical Programming*, 38:287–302, 1987.

[9] J. V. Burke. A sequential quadratic programming method for potentially infeasible mathematical programs. *Journal of Mathematical Analysis and Applications*, 139:319–351, 1987.

[10] J. V. Burke. A sequential quadratic programming method for potentially infeasible mathematical programs. *J. Math. Analysis and Applications*, 139:319–351, 1989.

[11] J. V. Burke. An exact penalization viewpoint of constrained optimization. *SIAM J. Control & Opt.*, 29:968–998, 1991.

[12] J. V. Burke and S. P. Han. A robust sequential quadratic programming method. *Math. Program.*, 43:277–303, 1989.

[13] J.V. Burke and R. Poliquin. Optimality conditions for non-finite valued convex composite functions. *Mathematical Programming*, 57:103–120, 1992.

[14] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16:1190–1208, 1995.

[15] R. H. Byrd, J. Nocedal, and R. A. Waltz. Knitro: An integrated package for nonlinear optimization. *Large Scale Nonlinear Optimization.*, pages 35–59, 2006.

[16] R. H. Byrd, J. Norcedal, and R. B. Schnabel. Representation of quasi-Newton matrices and their use in limed memory methods. *Mathematical Programming*, 63:129–156, 1994.

[17] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky. Latent variable graphical model selection via convex optimization. *Annals of Statistics*, 40(4), 2012.

[18] F. E. Curti, T. Johnson, D. P. Robinson, and A. Wächter. An Inexact Sequential Quadratic Optimization Algorithm for Large-Scale Nonlinear Optimization. *SIAM Journal on Optimization*, 24(3):1041–1074, 2014.

[19] P. Danaher, P. Wang, and D. M. Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society: Series B*, 2013.

[20] A. Defazio and T. Caetano. A convex formulation for learning scale-free network via sub-modular relaxation. *NIPS*, 2012.

[21] Q. T. Dinh, A. Kyrillidis, and V. Cevher. A proximal Newton framework for composite minimization: Graph learning without Cholesky decompositions and matrix inversions. *30th International Conference on Machine Learning*, 2013.

[22] J. Eckstein. *Splitting methods for monotone operators with applications to parallel optimization*. PhD thesis, Massachusetts Institute of Technology, 1989.

[23] S. K. Eldersveld. Large-scale sequential quadratic programming algorithms. *Ph.D. thesis, Department of Operations Research, Stanford University, Stanford, CA*, pages 143–156, 1991.

[24] V. Filkov. Identifying gene regulatory networks from gene expression data. *Handbook of Computational Moleculary Biology*, 2005.

[25] R. Fletcher. A model algorithm for composite nondifferentiable optimization. *Mathematical Programming Study*, 17:67–76, 1982.

[26] R. Fletcher. A l_1 penalty method for nonlinear constraints. *Numerical Optimization*, pages 26–40, 1984.

[27] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, second edition, 1987.

[28] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2007.

[29] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM REVIEW*, 47(1):99–131, 2005.

[30] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. User's Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming. *Report SOL 86-2, Department of Operations Research, Stanford University, Stanford, CA*, pages 143–156, 1986.

[31] D. Goldfarb and S. Ma. Fast multiple-splitting algorithms for convex optimization. *SIAM Journal on Optimization*, 22(2):533–556, 2012.

[32] D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming*, 141(1-2):349–382, 2013.

[33] T. Goldstein, B. O'Donoghue, and S. Setzer. Fast alternating direction optimization methods. Technical report, CAM Report 12-35, UCLA, 2012.

[34] N. I. M. Gould, S. Lucidi, M. Roma, and P. L. Toint. Solving the trust-region subproblem using the lanczos method. *SIAM J. Optim.*, 9:504–525, 1999.

[35] N. I. M Gould, D. Orban, and P. Toint. CUTEr (and SifDec): a Constrained and Unconstrained Testing Environment, revisited. *ACM Transactions on Mathematical Software*, 29:373?394, 2003.

[36] A. Goyal, F. Bonchi, and L. V. S Lakshmanan. Learning influence probabilities in social networks. *WSDM 2010*, 2010.

[37] S. Hara and T. Washio. Learning a common substructure of multiple graphical Gaussian models. *Nural Networks*, 38:23–28, 2013.

[38] J. Honorio and D. Samaras. Multi-task learning of Gaussian graphical models. *Proc. of the 27th International Conference on Machine Learning*, 2010.

[39] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[40] C. Hsieh, K. Chang, C. Lin, S. Keerthi, and S. Sundararajan. On the limited memory bfgs method for large scale optimization. *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[41] C. Hsieh, M. Sustik, I. Dhillon, and P. Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. *Advances in Neural Information Processing Systems*, 2011.

[42] C. Hsieh, M. Sustik, I. Dhillon, P. Ravikumar, and R. Poldrack. BIG & QUIC: Sparse inverse covariance estimation for a million variables. *Advances in Neural Information Processing Systems*, 2012.

[43] E. Jones, T. Oliphant, P. Peterson, and Others. SciPy: Open source scientific tools for Python, 2001.

[44] Q. Liu and A. Ihler. Learning scale free networks by reweighted ℓ_1 regularization. *AISTATS*, 2011.

[45] Z. Luo and P. Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *J. Optim. Theory Appl.*, 72:7–35, 1992.

[46] S. Ma, L. Xue, and H. Zou. Alternating direction methods for latent variable Gaussian graphical model selection. *Journal Neural Computation*, 25:2172–2198, 2013.

[47] R. Mazumder and T. Hastie. The graphical lasso: New insights and alternatives. *Electronic Journal of Statistics*, pages 2125 – 2149, 2012.

[48] K. Mohan, P. London, M. Fazel, D. Witten, and S. Lee. Node-based learning of multiple Gaussian graphical models. *Journal of Machine Learning Research*, 2013.

[49] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269:543–547, 1983.

[50] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 2nd edition, 2006.

[51] D. P O'Leary. Robust regression computation using iteratively reweighted least squares. *SIAM J. Matrix Anal. Appl.*, 11:466–480, 1990.

[52] T. E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9:10–20, 2007.

[53] P. Olsen, F. Oztoprak, J. Nocedal, and S. Rennie. Newton-like methods for sparse inverse covariance estimation. *Advances in Neural Information Processing Systems*, 2012.

[54] M.R. Osborne. *Finite Algorithms in Optimization and Data Analysis*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1985.

[55] K. B. Petersen and M. S. Pedersen. The matrix cookbook. 11 2012.

[56] M. J. D. Powell. General algorithms for discrete nonlinear approximation calculations. In C.K. Chui, L.L. Schumaker, and J.D. Ward, editors, *Approximation Theory IV*, pages 187–218. Academic Press, N.Y., 1983.

[57] G. Robins, P. Pattison, Y. Kalish, and D. Lusher. An introduction to exponential random graph models for social networks. *Social Networks*, 29(2):173 – 191, 2007.

[58] R. T. Rockafellar. *Convex Analysis*. Priceton Landmarks in Mathematics. Princeton University Press, 1970.

[59] R. T. Rockafellar and R. J. B. Wets. *Variational Analysis*, volume 317 of *A Series of Comprehensive Studies in Mathematics*. Springer, 1998.

[60] B. Rolfs, B. Rajaratnam, D. Guillot, I. Wong, and A. Maleki. Iterative thresholding algorithm for sparse inverse covariance estimation. *Advances in Neural Information Processing Systems*, 2012.

[61] E. J. Schlossmacher. An iterative technique for absolute deviations curve fitting. *J. Am. Stat. Assoc.*, 68:857–865, 1973.

[62] M. Schmidt and K. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. *AISTATS*, 2010.

[63] J. E. Spingarn. Applications of the method of partial inverses to convex programming: decomposition. *Mathematical Programming*, 32, 1985.

[64] K. Tan, P. London, K. Mohan, M. Fazel, S. Lee, and D. Witten. Learning graphical models with hubs. *Journal of Machine Learning Research*, 15:3297–3331, 2014.

[65] G. Varoquaux, A. Gramfort, J. B. Poline, B. Thirion, R. Zemel, and J. Shawe-Taylor. Brain covariance selection: better individual functional connectivity models using population prior. *Advances in Neural Information Processing Systems*, 2010.

[66] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[67] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2), 2008.

[68] H. Wang and A. Banerjee. Online alternating direction method. *http://arxiv.org/abs/1306.3721*, 2013.

[69] R. Wolke and H. Schwetlick. Iteratively reweighted least squares: algorithms, convergence analysis, and numerical comparisons. *SIAM J. Sci. Stat. Comput.*, 9:907–921, 1988.

[70] M. A. Woodbury. Inverting modified matrices. *Memorandum Report 42, Statistical research group, Princeton university*, 1950.

[71] S. Yun, P. Tseng, and K. Toh. A block coordinate gradient descent method for regularized convex separable optimization and covariance selection. *Mathematical Programming*, 129:331–355, 2011.

[72] S. Yun, P. Tseng, and K.-C. Toh. A block coordinate gradient descent method for regularized convex separable optimization and covariance selection. *Mathematical programming*, pages 331 – 355, 2011.

[73] E.H. Zarantonello. *Projections on convex sets in Hilbert space and spectral theory.* Academic Press, New York, 1971.

[74] B. Zhang and Y. Wang. Learning structural changes of Gaussian graphical models in controlled experiments. *Proc. of the 26th Conference on Uncertainity in Artificial Intelligence*, 2010.

[75] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. *The Annual Conference on Neural Information Processing Systems*, 16:49–56, 2004.