On the Relationship between Conjugate Gradient and Optimal First-Order Methods for Convex Optimization

by

Sahar Karimi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In a series of work initiated by Nemirovsky and Yudin, and later extended by Nesterov, first-order algorithms for unconstrained minimization with optimal theoretical complexity bound have been proposed. On the other hand, conjugate gradient algorithms as one of the widely used first-order techniques suffer from the lack of a finite complexity bound. In fact their performance can possibly be quite poor. This dissertation is partially on tightening the gap between these two classes of algorithms, namely the traditional conjugate gradient methods and optimal first-order techniques. We derive conditions under which conjugate gradient methods attain the same complexity bound as in Nemirovsky-Yudin's and Nesterov's methods. Moreover, we propose a conjugate gradient-type algorithm named CGSO, for Conjugate Gradient with Subspace Optimization, achieving the optimal complexity bound with the payoff of a little extra computational cost.

We extend the theory of CGSO to convex problems with linear constraints. In particular we focus on solving l_1-regularized least square problem, often referred to as Basis Pursuit Denoising (BPDN) problem in the optimization community. BPDN arises in many practical fields including sparse signal recovery, machine learning, and statistics. Solving BPDN is fairly challenging because the size of the involved signals can be quite large; therefore first order methods are of particular interest for these problems. We propose a quasi-Newton proximal method for solving BPDN. Our numerical results suggest that our technique is computationally effective, and can compete favourably with the other state-of-the-art solvers.

Acknowledgements

First and foremost, I would like to express my sincere gratitude and appreciation to my supervisor, Stephen Vavasis. This thesis would have not been possible without his tremendous support, patience and unsurpassed knowledge. He has been a great mentor and I am extremely thankful to him.

I am also grateful to my examination committee, Henry Wolkowicz, Levent Tunçel, Hans De Sterck, and Michael Friedlander for their insightful comments and helpful suggestions.

Furthermore, I would like to acknowledge the role of the staff members, the faculty members and my fellow graduate students in the department of Combinatorics and Optimization. I wish to specially thank the members and organizers of the Continuous Optimization group for the wonderful seminars and useful discussions.

Thank you to all my friends who were there for me. Particularly, I owe a heartfelt thanks to my dear friend Ali who encouraged me and lightened me up whenever I needed it the most.

Last but certainly not least, my greatest, deepest and most special thanks goes to my parents, Mehdi and Fahimeh, and my brother, Soheil, who made finishing this thesis possible with their unconditional love, endless patience and continuous support. My gratitude is beyond words, but thank you for believing in me and encouraging me on this journey.

Dedication

To my parents

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The rapid progress in technology and the larger problem sizes arising in optimization call for fast, accurate and robust algorithms. For many large-scale problems, forming the Hessian is quite expensive so algorithms that rely only on function and gradient evaluations are of particular interest. These algorithms are often referred to as first-order algorithms (FOA); in other words, FOA refers to iterative algorithms that rely on information obtained by the first derivative of function in each iteration.

This dissertation is divided into two parts. The first part focuses on a class of first-order techniques for unconstrained problems called Conjugate Gradient algorithms (CG). The second part focuses on solving "Basis Pursuit Denoising Problem" (BPDN) arising in many applications including sparse recovery in compressive sensing. In the remainder of this chapter, we first present the notation and review the required background used throughout this thesis. CG methods are reviewed in Section 1.2. In Section 1.3 we give a brief overview of sparse recovery and BPDN problem. An outline and the contribution of this work is discussed in Section 1.4.

1.1 Notation and Background

1.1.1 Linear Algebra

The material of this section is covered in [63, 62] in extensive detail. We reserve the lowercase alphabet for vectors, and uppercase alphabet for matrices throughout this thesis. Scalars are denoted by lowercase Greek letters. We are mainly working with real Euclidean

vector space $\mathbb{R}^n$ equipped with inner product denoted by $\langle \cdot, \cdot \rangle$, and defined as $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$.

A linear transformation, matrix, is defined as $A : \mathbb{R}^n \to \mathbb{R}^m$. Its adjoint is denoted by A^t, and we have $\langle Ax, y \rangle = \langle x, A^t y \rangle$. We use the standard notion of "square", "symmetric", "nonsingular", "diagonal" and "orthonormal" for matrices of size $n \times n$. $\det(A)$, and A^{-1} stand for determinant and inverse of A, respectively. a_{ij}, $a_{[i]}$, and $a^{[j]}$ represent the ijth entry, ith row, and jth column of A, respectively.

Norms:

The p-norm of a vector is defined as $(\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$. The following special cases are particularly important in our discussions:

$$\|x\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}, \qquad \|x\|_1 := \sum_{i=1}^{n} |x_i|, \qquad \|x\|_\infty := \max_i |x_i|.$$

The function $\|x\|_2$ is often referred to as Euclidean norm, and $\|x\|_1$ as l_1-norm. The following inequalities play important roles regarding vector norms:

$$\|x + y\|_p \leq \|x\|_p + \|y\|_p \ , \tag{1.1}$$

$$|\langle x, y \rangle| \leq \|x\|_2 \|y\|_2 \ . \tag{1.2}$$

The first inequality is the "triangle inequality", which holds true for any norm; while the second one is "Cauchy-Schwarz inequality" and holds for Euclidean norm. In the remainder of this thesis we use $\|x\|$ for Euclidean norm unless otherwise is stated.

Matrix induced norms are natural extensions of vector norms. Associated with any vector norm, its corresponding matrix norm is defined as:

$$\|A\|_p = \sup_{\|x\|_p = 1} \|Ax\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}. \tag{1.3}$$

The induced l_1 and l_∞ norms are sometimes called maximum column sum and maximum row sum because the induced norm formulation essentially boils down to the following equivalent forms:

$$\|A\|_1 = \sup_{1 \leq j \leq n} \sum_{i=1}^{n} |a_{ij}| \quad \text{and} \quad \|A\|_\infty = \sup_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|. \tag{1.4}$$

The induced Euclidean or l_2-norm is called the "spectral norm" and equals to

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^t A)}, \tag{1.5}$$

where $\lambda_{\max}$ stands for the maximum eigenvalue (to be defined shortly).

Note that any $m \times n$ matrix can be considered as a vector of size mn, hence we can still define a component-wise matrix norm as

$$\|A\|_c = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^c \right)^{\frac{1}{c}} . \tag{1.6}$$

Frobenius norm refers to entry-wise 2-norm for matrices. In our discussion, we use induced norms for matrices unless otherwise is stated.

Eigenvalue Decomposition and Positive Definiteness:

Scalar λ and nonzero vector v are called eigenvalue and eigenvector of matrix $A \in \mathbb{R}^{n \times n}$ if $Av = \lambda v$. All eigenvalues of a symmetric matrix are real numbers. A symmetric matrix can be represented by its eigenvalues as $V \Lambda V^t$, where V is an orthogonal matrix consisting of eigenvectors of A, and Λ is a diagonal matrix with eigenvalues as the diagonal entries.

A symmetric matrix A is called positive semidefinite if $x^t A x \geq 0$ for all $x \in \mathbb{R}^n$; if the inequality holds strictly for all nonzero vectors x, i.e., $x^t A x > 0$, then we call A positive definite. Notations $A \succeq 0$ and $A \succ 0$ represent positive semidefinite and positive definite matrices. Also $A \succeq B$ and $A \succ B$ means $A - B \succeq 0$ and $A - B \succ 0$, respectively. All eigenvalues of a positive semidefinite matrix are nonnegative; similarly a positive definite matrix has only positive eigenvalues, so it is nonsingular. The eigenvalue decomposition enables us to define the square root of a positive semidefinite matrix A, as $A^{\frac{1}{2}} = V \Lambda^{\frac{1}{2}} V^t$, where $\Lambda^{\frac{1}{2}}$ stands for the diagonal matrix with square root of eigenvalues on the diagonal. $\mathbf{I}$ and $\mathbf{e}$ refer to the identity matrix and all ones vector, respectively.

Definition 1.1. *The scaled norm associated with a positive definite matrix H is defined as*

$$\|x\|_H = \sqrt{x^t H x} . \tag{1.7}$$

It is not too difficult to show that (1.7) satisfies all the axioms for a vector norm, namely

- Since $H \succ 0$, $x^t H x = 0 \Leftrightarrow x = 0$,
- $\|\alpha x\|_H = \sqrt{\alpha^2 x^t H x} = \alpha \|x\|_H$, and
- $\|x\|_H + \|y\|_H \geq \|x + y\|_H \Leftrightarrow (\|x\|_H + \|y\|_H \geq \|x + y\|_H)^2 \Leftrightarrow \sqrt{x^t H x y^t H y} \geq x^t H y$
  Let $a = H^{\frac{1}{2}} x$, and $b = H^{\frac{1}{2}} y$, then the final inequality is simply Cauchy-Schwarz inequality on a, and b, i.e., $\|a\|\|b\| \geq \langle a, b \rangle$.

Suppose a and b are defined as above, then

$$a^t b = x^t H y \leq \|a\|\|b\| = \sqrt{x^t H x}\sqrt{y^t H y} = \|x\|_H \|y\|_H;$$

in other words the notion of Cauchy-Schwarz inequality can be extended for scaled norms as

$$x^t H y \leq \|x\|_H \|y\|_H \ . \tag{1.8}$$

In addition, suppose λ_{min} and λ_{max} denote the minimum and maximum eigenvalues of a positive definite H, then for any x we have

$$\lambda_{min}\|x\| \leq \|x\|_H \leq \lambda_{max}\|x\| \ . \tag{1.9}$$

1.1.2 Calculus and Convex Analysis

In this section we present a brief overview of convex analysis and the notions we require throughout this thesis. For detailed discussion on convex analysis, one may refer to [97, 61]. The standard notations of $\nabla f(x)$ and $\nabla^2 f(x)$ are reserved for gradient and Hessian of function $f : \mathbb{R}^n \to \mathbb{R}$. Since the algorithms discussed in this thesis are iterative methods, we sometimes use the notation ∇f^k for $\nabla f(x^k)$, where k is the iteration counter. Moreover $\mathcal{D}(f)$ denotes the domain of a function $f : \mathbb{R}^n \to \mathbb{R} \cup \infty$, and is defined as

$$\mathcal{D}(f) = \{x : f(x) < \infty\}.$$

The signum function is denoted by sgn, and is defined as

$$\text{sgn}(x)_i = \begin{cases} -1 & \text{if } x_i < 0, \\ 0 & \text{if } x_i = 0, \\ 1 & \text{if } x_i > 0. \end{cases} \tag{1.10}$$

Rate of Convergence: Suppose $\{x^k\}$ is a sequence converging to x^*. The sequence $\{x^k\}$ is said to have convergence rate of order p if there exists a constant m such that

$$\frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^p} \leq m, \tag{1.11}$$

for sufficiently large k. In the special cases that $p = 1$ and $m \in (0, 1)$ the rate of convergence is called "linear". When $p = 2$, the sequence has "quadratic" rate of convergence. A sequence converges "superlinearly" if

$$\lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|} = 0. \tag{1.12}$$

Convex Sets and Convex Functions:

Set $\mathcal{C}$ is called convex if for any $x, y \in \mathcal{C}$ and $\lambda \in (0,1)$, we have $\lambda x + (1-\lambda)y \in \mathcal{C}$. Let $B_\epsilon(x)$ denote a ball of radius ϵ centred at x. The sets $\text{int}(\mathcal{C})$ and $\text{rint}(\mathcal{C})$ stand for the interior and relative interior of a convex set $\mathcal{C}$ which are defined as

$$\text{int}(\mathcal{C}) = \{x \in \mathcal{C} : \exists \epsilon > 0,\ B_\epsilon(x) \subseteq \mathcal{C}\}, \tag{1.13}$$
$$\text{rint}(\mathcal{C}) = \{x \in \mathcal{C} : \exists \epsilon > 0,\ B_\epsilon(x) \cap \text{aff}(\mathcal{C}) \subseteq \mathcal{C}\}. \tag{1.14}$$

A set of the form $\{x : Ax \leq b\}$ is called a polyhedron. It's not difficult to show that every polyhedron is a convex set.

Projection of a point $x \in \mathbb{R}^n$, onto a closed convex set $\mathcal{C}$ is defined as

$$\text{Proj}_{\mathcal{C}}(x) = \arg\min_{u \in \mathcal{C}} \|u - x\|. \tag{1.15}$$

Let $\mathcal{C} \in \mathbb{R}^n$ be a set, not necessarily convex. The dual cone of $\mathcal{C}$, which is always closed and convex, is denoted by $\mathcal{C}^*$ and defined as

$$\mathcal{C}^* = \{d : \langle d, c\rangle \geq 0,\ \forall c \in \mathcal{C}\}. \tag{1.16}$$

The polar cone of $\mathcal{C}$ is denoted by $\mathcal{C}^\circ$ and is defined as

$$\mathcal{C}^\circ = \{d : \langle d, c\rangle \leq 0,\ \forall c \in \mathcal{C}\}. \tag{1.17}$$

Clearly the polar cone is equal to the negative of the dual cone, i.e., $\mathcal{C}^\circ = -\mathcal{C}^*$.

Similarly, the dual cone and the polar cone of a cone $\mathcal{K}$ are defined as

$$\mathcal{K}^* = \{d : \langle d, c\rangle \geq 0,\ \forall c \in \mathcal{K}\}, \tag{1.18}$$
$$\mathcal{K}^\circ = \{d : \langle d, c\rangle \leq 0,\ \forall c \in \mathcal{K}\}. \tag{1.19}$$

Moreover, if the cone $\mathcal{K}$ is closed and convex, then $\mathcal{K} = \mathcal{K}^{**}$.

Theorem 1.1. *[60, Theorem 3.2.5 (Moreau's Decomposition Theorem)] Let $\mathcal{K}$ be a closed convex cone. For x, y, and z in $\mathbb{R}^n$, the following statements are equivalent:*

- *$z = x + y$, $x \in \mathcal{K}$, $y \in \mathcal{K}^\circ$ and $\langle x, y\rangle = 0$,*
- *$x = Proj_{\mathcal{K}} z$ and $y = Proj_{\mathcal{K}^\circ} z$.*

The cone of feasible directions, the tangent cone, and the normal cone of a set $\mathcal{C}$ at point $x \in \mathcal{C}$ are denoted by $F_{\mathcal{C}}(x)$, $T_{\mathcal{C}}(x)$, and $N_{\mathcal{C}}(x)$, respectively and defined as

$$F_{\mathcal{C}}(x) = \{d : \exists\ \bar{\epsilon} > 0 \text{ s.t. } \ x + \epsilon d \in \mathcal{C} \text{ for } \epsilon \in (0, \bar{\epsilon})\}, \tag{1.20}$$

$$T_{\mathcal{C}}(x) = \left\{d : \exists\{x^k\} \subseteq C, \{\tau^k\} \text{ s.t. } x^k \to x,\ \tau^k \to 0 \text{ and } \frac{x^k - x}{\tau^k} \to d\right\}, \tag{1.21}$$

$$N_{\mathcal{C}}(x) = \{d : \langle d, x - y\rangle \leq 0 \text{ for } \forall y \in \mathcal{C}\}. \tag{1.22}$$

A function f is said to be convex if

$$f\left(\lambda x + (1-\lambda)y\right) \leq \lambda f(x) + (1-\lambda)f(y),$$

for each $x,\ y \in \mathcal{D}(f)$ and $\lambda \in (0,1)$. Any convex function is continuous over the interior of its domain. If the above inequality holds true as strict inequality for all $x, y \in \mathcal{D}(f)$, $x \neq y$, and $\lambda \in (0,1)$, then the function is strictly convex.

The indicator function of a convex set $\mathcal{C}$ is convex and is defined as:

$$\mathcal{I}_{\mathcal{C}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C}, \\ \infty & \text{otherwise.} \end{cases} \tag{1.23}$$

The vector ξ is called a subgradient of a convex function f at $x \in \mathcal{D}(f)$ if

$$f(y) - f(x) \geq \langle \xi, y - x\rangle \quad \text{for all } y \in \mathcal{D}(f). \tag{1.24}$$

The set of all subgradients of f at x is denoted by $\partial f(x)$. The set $\partial f(x)$ is always nonempty, closed and convex. A point x is a minimizer of f if and only if 0 lies in $\partial f(x)$.

A function is said to be differentiable at x if $\nabla f(x)$ is the only element in $\partial f(x)$. A continuously differentiable function is a differentiable function whose $\nabla f(x)$ is a continuous function as well. Function f belongs to the class of k continuously differentiable functions if the first k derivatives of f $(\nabla f,\ \nabla^2 f, \ldots, \nabla^k f)$ exist and are continuous.

Let $\mathbf{C}^k$ stand for the class of k continuously differentiable convex functions; no superscript simply means that $k = 1$, i.e., the class of continuously differentiable functions. The following inequality holds for a function $f \in \mathbf{C}$ at every $x, y \in \mathcal{D}(f)$:

$$\langle \nabla f(x) - \nabla f(y), x - y\rangle \geq 0. \tag{1.25}$$

A function is called strongly convex if there exists a constant l such that for any $x,\ y \in \mathcal{D}(f)$ we have

$$f(y) \geq f(x) + \langle \xi, y - x\rangle + \frac{l}{2}\|x - y\|^2, \tag{1.26}$$

where $\xi \in \partial f(x)$. We say a function is "Lipschitz continuous with constant L" on $\mathcal{C} \subseteq \mathcal{D}(f)$ if for any $x,\ y \in \mathcal{C}$ and a constant L,

$$\|f(x) - f(y)\| \leq L\|x - y\|; \tag{1.27}$$

L is called the Lipschitz constant of f. For first-order algorithms, the Lipschitz continuity of the gradient is often desired. We use the notation $\mathbf{C}_L$ for the class of continuously differentiable convex functions with Lipschitz continuous gradient. In the first part of this thesis we are primarily focused on continuously differentiable strongly convex functions with Lipschitz continuous gradients, namely functions f for which we have

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \tag{1.28}$$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{l}{2}\|x - y\|^2. \tag{1.29}$$

We use the notation $\mathbf{C}_{l,L}$ for the aforementioned class of functions. As a result of (1.28), we have the following for a function $f \in \mathbf{C}_{l,L}$:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|x - y\|^2, \tag{1.30}$$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|^2, \tag{1.31}$$

$$\langle \nabla f(x) - \nabla f(y), x - y\rangle \geq \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|^2. \tag{1.32}$$

For derivation of the above inequalities one may refer to [85, Theorem 2.1.5].

Mean Value Theorem: Let f be differentiable on a convex set $\mathcal{C}$. Then for any $x,\ y \in \mathcal{C}$ we have

$$f(y) = f(x) + \nabla f\left(x + \lambda(y - x)\right)^t (y - x), \quad \text{for some } \lambda \in (0, 1). \tag{1.33}$$

Taylor's Theorem: Let f be twice differentiable on an open convex set $\mathcal{C}$. The second-order Taylor's expansion of f at x is

$$f(y) = f(x) + \nabla f(x)^t(y - x) + \frac{1}{2}(y - x)^t H(y - x) + \mathcal{O}(\|y - x\|^3).$$

If f is differentiable, one may aim for the first-order Taylor's expansion as

$$f(y) = f(x) + \nabla f(x)^t(y - x) + \mathcal{O}(\|y - x\|^2) \quad y \in \mathcal{C}.$$

Taylor's theorem generalizes the above equations to $k-$th differentiable functions. This is however beyond the scope of our work and needs additional definitions and notations. The interested reader may refer to [80, Theorem 3.3.1].

We use the term "composite function" to refer to a function of the following form:

$$F(x) := f(x) + g(x), \tag{1.34}$$

where f and g are both convex functions. Function g is possibly non-smooth, but $f \in \mathbf{C}_L$. The notion of composite function has been commonly used in the literature of first-order method for the summation of two convex functions, see for example [87, 74], and should not be confused with the composition of two functions.

Definition 1.2. *[30] The projection of the steepest descent direction, i.e., $-\nabla f(x)$, of a convex function $f : \mathcal{C} \to \mathbb{R}$ at $x \in \mathcal{C}$ is denoted by $\nabla_{\mathcal{C}} f(x)$ and defined as below:*

$$\nabla_{\mathcal{C}} f(x) = \arg\min_{v} \{\|v + \nabla f(x)\| \ : \ v \in T_{\mathcal{C}}(x)\}. \tag{1.35}$$

This is often referred to as the projected gradient.

Definition 1.3. *The "proximal mapping" or "proximal operator" of convex function g at y is defined as*

$$Prox_g(y) = \arg\min_{x} \left\{ \frac{1}{2}\|x - y\|^2 + g(x) \right\}. \tag{1.36}$$

Note that projection onto a convex set $\mathcal{C}$ corresponds to a proximal mapping in which $g(x)$ is the indicator function for $\mathcal{C}$, i.e.,

$$\text{Proj}_{\mathcal{C}}(x) = \text{Prox}_g(x), \tag{1.37}$$

where

$$g(x) = \begin{cases} 0 & x \in \mathcal{C}, \\ \infty & \text{otherwise.} \end{cases} \tag{1.38}$$

The proximal point, $\text{Prox}_g(y)$, exists and is unique because $\frac{1}{2}\|x - y\|^2 + g(x)$ is strictly convex. Moreover, the proximal operator is firmly nonexpansive, i.e.,

$$\|\text{Prox}_g(x) - \text{Prox}_g(y)\|^2 \leq \langle \text{Prox}_g(x) - \text{Prox}_g(y), x - y \rangle \quad \forall x, y \in \mathcal{D}\left(p(x)\right). \tag{1.39}$$

It is not too difficult to see why (1.39) is true. By the optimality of x, there exists a $\xi_g(x) \in \partial g(x)$ such that

$$\text{Prox}_g(y) - y + \xi_g\left(\text{Prox}_g(y)\right) = 0 \quad \Rightarrow \quad \xi_g\left(\text{Prox}_g(y)\right) = y - \text{Prox}_g(y). \tag{1.40}$$

Similarly, we have $\xi_g\left(\text{Prox}_g(x)\right) = x - \text{Prox}_g(x)$. Using the convexity of g we get

$$g\left(\text{Prox}_g(x)\right) \geq g\left(\text{Prox}_g(y)\right) + \langle y - \text{Prox}_g(y), \text{Prox}_g(x) - \text{Prox}_g(y)\rangle, \tag{1.41}$$
$$g\left(\text{Prox}_g(y)\right) \geq g\left(\text{Prox}_g(x)\right) + \langle x - \text{Prox}_g(x), \text{Prox}_g(y) - \text{Prox}_g(x)\rangle. \tag{1.42}$$

Adding the above two inequalities concludes that the proximal operator is nonexpansive. Also using (1.39) and Cauchy-Schwarz inequality we derive

$$\|\text{Prox}_g(x) - \text{Prox}_g(y)\| \leq \|x - y\|. \tag{1.43}$$

Definition 1.4. *"Shrinkage" or "Soft-thresholding" (commonly called thresholding) operator refers to a proximal mapping where $g(x)$ is the l_1-norm function, i.e.,*

$$\mathcal{S}_\lambda(y) = \arg\min_x \left\{ \frac{1}{2}\|x - y\|^2 + \lambda\|x\|_1 \right\}. \tag{1.44}$$

What makes thresholding operator interesting is the fact that it can be easily computed,

$$\mathcal{S}_\lambda(y)_i = \begin{cases} y_i - \lambda & y_i \geq \lambda, \\ 0 & |y_i| \leq \lambda, \\ y_i + \lambda & y_i \leq -\lambda. \end{cases} \tag{1.45}$$

One may easily check that (1.45) is actually equivalent to

$$\mathcal{S}_\lambda(y) = \text{sgn}(y) \odot \max\left\{|y| - \lambda, 0\right\}, \tag{1.46}$$

where $\odot$ denotes the entry-wise, or Hadamard product.

Using the definition of scaled norm (1.1), we can define the notion of a "scaled proximal mapping".

Definition 1.5. *The scaled proximal mapping (or operator) of a convex function g, associated with a positive definite matrix H, is defined as*

$$Prox_g^H(y) = \arg\min_x \left\{ \frac{1}{2}\|x - y\|_H^2 + g(x) \right\}. \tag{1.47}$$

The properties of proximal mapping can be generalized to scaled proximal mapping. The scaled proximal point, $\text{Prox}_g^H(y)$, exists and is unique because for a positive definite matrix H the proximity function, $\frac{1}{2}\|x - y\|_H^2 + g(x)$, is strictly convex. Besides we have

$$\begin{aligned} \xi_g\left(\text{Prox}_g^H\left(y\right)\right) &= H\left(y - \text{Prox}_g^H\left(y\right)\right), \\ \xi_g\left(\text{Prox}_g^H\left(x\right)\right) &= H\left(x - \text{Prox}_g^H\left(x\right)\right), \end{aligned} \tag{1.48}$$

by optimality conditions for (1.47). We may now use the convexity of g to obtain

$$g\left(\text{Prox}_g^H(x)\right) \geq g\left(\text{Prox}_g^H(y)\right) + \langle H\left(y - \text{Prox}_g^H(y)\right), \text{Prox}_g^H(x) - \text{Prox}_g^H(y)\rangle,$$
$$g\left(\text{Prox}_g^H(y)\right) \geq g\left(\text{Prox}_g^H(x)\right) + \langle H\left(x - \text{Prox}_g^H(x)\right), \text{Prox}_g^H(y) - \text{Prox}_g^H(x)\rangle.$$

Adding the above two inequalities gives us

$$\|\text{Prox}_g(x) - \text{Prox}_g(y)\|_H^2 \leq \langle \text{Prox}_g(x) - \text{Prox}_g(y), H(x-y)\rangle, \tag{1.49}$$

which shows that the scaled proximal map is firmly nonexpansive. Using the notion of scaled Cauchy-Schwarz inequality we attain

$$\|\text{Prox}_g(x) - \text{Prox}_g(y)\|_H \leq \|x - y\|_H. \tag{1.50}$$

This concludes the review on proximal operator. The interested reader can reach a more in-depth discussion on proximal operators in [7].

1.1.3 Convex Optimization

An optimization problem has the following general form:

$$\min_{x\in\mathcal{C}} f(x), \tag{1.51}$$

where $f(x)$ is the objective function and $\mathcal{C}$ is the feasible region. When f is a linear function and $\mathcal{C}$ is a polyhedron, the resulting problem is a linear program (LP); similarly, when f is quadratic, (1.51) is a quadratic program (QP).

An important class of problems in optimization is convex problems, in which $\mathcal{C}$ is a convex set and f is a convex function. In the absence of constraints and when $f \in \mathbf{C}$, it suffices to have $\nabla f(x^*) = 0$ for x^* to be the minimizer of f. The first order optimality condition for constrained convex problems requires the minimizer x^* to satisfy $-\nabla f(x^*) \in N_{\mathcal{C}}(x^*)$. In fact, for convex problems this condition is sufficient for optimality.

It is possible to state any convex problem in the format below:

$$\begin{array}{lll} \min & f(x) & \\ s.t. & g_i(x) \geq 0 & i \in \{1,\ldots,\kappa_{\mathcal{I}}\}, \\ & h_j(x) = 0 & j \in \{1,\ldots,\kappa_{\mathcal{E}}\}, \end{array} \tag{1.52}$$

where $f(x)$ and $-g_i(x)$ for all $i = 1, \ldots, \kappa_{\mathcal{I}}$ are convex functions and $h_j(x)$ for all $j = 1, \ldots, \kappa_{\mathcal{E}}$ are affine functions. The Lagrange function associated with problem (1.52) is defined as

$$L(x, \lambda, \mu) = f(x) - \sum_{i=1}^{\kappa_{\mathcal{I}}} \lambda_i g_i(x) - \sum_{i=1}^{\kappa_{\mathcal{E}}} \mu_i h_i(x), \tag{1.53}$$

where $\lambda \in \mathbb{R}^{\kappa_{\mathcal{I}}}$ and $\lambda \geq 0$ and $\mu \in \mathbb{R}^{\kappa_{\mathcal{E}}}$ are Lagrange multipliers of problem (1.52). The Lagrange function is quite important in deriving optimality conditions on the minimizer of (1.52). The point x^* is the minimizer of (1.52) if it solves

$$\min_{x} \max_{\lambda \geq 0,\ \mu} L(x, \lambda, \mu). \tag{1.54}$$

Problem (1.54) is called the primal problem. Its dual is

$$\max_{\lambda \geq 0,\ \mu} \min_{x} L(x, \lambda, \mu). \tag{1.55}$$

By weak duality theorem, any feasible solution of the dual problem provides a lower bound on the primal problem. Strong duality states that the value of the primal equals the value of the dual problem at optimality. Strong duality does not necessarily hold for any convex problem. There are, however, conditions under which strong duality holds. These conditions are called constraint qualifications [90, Chapter 12]. Suppose x^*, $\lambda^\star \geq 0$, and $\mu^\star$ are the optimal solutions of the primal-dual problem and the constraint qualifications hold. Then the following set of equations is true:

$$\nabla_x L(x^*, \lambda^\star, \mu^\star) = 0, \tag{1.56a}$$
$$g_i(x^*) \geq 0 \quad i \in \{1, \ldots, \kappa_{\mathcal{I}}\}, \tag{1.56b}$$
$$h_j(x^*) = 0 \quad j \in \{1, \ldots, \kappa_{\mathcal{E}}\}, \tag{1.56c}$$
$$\lambda^\star \geq 0, \tag{1.56d}$$
$$\lambda_i^\star g_i(x^*) = 0 \quad \forall i \in \{1, \ldots, \kappa_{\mathcal{I}}\}. \tag{1.56e}$$

Conditions (1.56) called KKT conditions are sufficient optimality conditions for the class of convex problems, i.e., if KKT conditions hold, then x^*, $\mu^\star$, and $\lambda^\star$ are optimal solutions to the primal and dual problems.

1.2 Conjugate Gradient Method

The method of Conjugate Gradient (CG) was introduced by Hestenes and Stiefel [59] for minimizing convex quadratic functions,

$$f(x) = \frac{1}{2}x^t Ax - b^t x, \tag{1.57}$$

where $A \succ 0$, or $A \succeq 0$ and $b \in \text{range}(A)$. The first order optimality conditions (i.e., $\nabla f(x) = Ax - b = 0$) imply that the solution to the above problem, x^*, solves the linear equation $Ax = b$; therefore we refer to this algorithm as "linear conjugate gradient". Hestenes and Stiefel's original linear CG has the following form:

$$x^{j+1} = x^j - \frac{(r^j)^t d^j}{(d^j)^t A d^j} d_j, \tag{1.58a}$$

$$d^{j+1} = -r^{j+1} + \frac{(r_{j+1})^t A d^j}{(d^j)^t A d_j} d_j. \tag{1.58b}$$

In the above equations r^j is $\nabla f(x^j) = Ax^j - b$ and $d^0 = -r^0$, where x^0 is the starting point. It is possible to show that the number of iterations in linear CG is bounded by the dimension of the problem, n. For more details on linear CG, one may refer to [50] or [90].

Conjugate gradient technique was soon generalized by Fletcher and Reeves [47] and Polak and Ribière [92], to the general problem of unconstrained minimization, i.e.,

$$\min_{x \in \mathbb{R}^n} f(x). \tag{1.59}$$

However, the theoretical basis for nonlinear CG is considerably weaker than that of the linear case. In the linear case, the successive gradients are mutually orthogonal and the search directions are mutually conjugate; these facts allow several strong convergence proofs including finite termination, convergence bounded in terms of problem condition number (i.e., $\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$), and superlinear convergence [50].

The general form of nonlinear CG is as follows:

$$x^{j+1} = x^j + \alpha^j d^j, \tag{1.60a}$$

$$d^{j+1} = -\nabla f(x^{j+1}) + \beta^j d^j. \tag{1.60b}$$

Here, d^j is the search direction at each iteration, and α^j is the step size, usually determined by a line search. Different updating rules for β^j gives us different variants of nonlinear CG. The most common formulas for computing β^j are:

$$\text{Fletcher-Reeves (1964):} \quad \beta_{FR} = \frac{\|g^{j+1}\|}{\|g^j\|}, \quad \textit{and}$$

$$\text{Polak-Ribière (1969):} \quad \beta_{PR} = \frac{\left(g^{j+1}\right)^t \left(g^{j+1} - g^j\right)}{\|g^j\|}.$$

Hager and Zhang [54] present a complete list of all updating rules in their survey on nonlinear CG. The convergence of nonlinear CG is highly dependent on the line search; for some, the exact line search is crucial. There are numerous papers devoted to the study of convergence properties of nonlinear CG, most of which discuss variants of nonlinear CG that do not rely on exact line search to be globally convergent. Al-Baali [3] shows the convergence of Fletcher-Reeves algorithm with inexact line search. Gilbert and Nocedal [48] establish the convergence of a variant of the Polak-Ribière nonlinear CG algorithm with no restart and no exact line search. Dai and Yuan [39] present a nonlinear CG for which the standard Wolfe condition suffices. A recent variant of CG has been proposed by Hager and Zhang [53] that relies on a line search satisfying the Wolfe Conditions. Furthermore this algorithm has the advantage that every search direction is a descent direction, which is not necessarily the case in nonlinear CG.

From Yuan and Stoer's perspective [116], CG is a technique in which the search direction d^{j+1} lies in the subspace of Span$\{g^{j+1}, d^j\}$. In their proposed algorithm, they compute the new search direction by minimizing a quadratic approximation of the objective function over the mentioned subspace. The idea of finding the next iterate of CG through a subspace optimization is the core of the variant of CG we proposed in the next chapter. It is also explored in the proposed CG techniques in [1] and [70]. A more generalized form of CG called Heavy Ball Method, was introduced by Polyak [93], in which x^{j+1} is $x^j + \alpha(-g^j) + \beta(x^j - x^{j-1})$. He proved a geometric progression rate for this algorithm when α and β belong to a specific range.

About two decades after nonlinear CG was originated, Nemirovsky and Yudin [82] proposed a variant of this algorithm with the worst-case complexity bound of $O(\ln(1/\epsilon)\sqrt{L/l})$. Here ϵ is the desired relative accuracy, that is, $(f(x^n) - f(x^*))/(f(x^0) - f(x^*)) \leq \epsilon$, where x^0 is the starting point, x^* is the optimizer, and x^n is the final iterate. This is the best achievable bound for this particular class of methods and functions [82, 85]. We refer to this algorithm as NY-CG, and will cover some more details about their method in the next chapter.

Although NY-CG algorithm can be regarded as a variant of conjugate gradient, it does not reduce to linear conjugate gradient when applied to a convex quadratic function. In fact, it can be much slower. In contrast, the FR and PR variants of nonlinear CG reduce to linear CG in the case of a convex quadratic and if an exact line search is used. Many would argue that this is a defining property of nonlinear CG. In addition, NY-CG requires an expensive subspace optimization step on every iteration. A later paper by Nesterov [84] remedied this drawback by achieving the same complexity without the need for subspace optimization.

The general form of Nesterov's algorithm is:

$$x^k = y^k - \alpha^k \nabla f(y^k), \tag{1.61a}$$

$$t^{k+1} = \frac{1 + \sqrt{1 + 4t^{k^2}}}{2}, \tag{1.61b}$$

$$y^{k+1} = x^k + \frac{t^k - 1}{t^{k+1}} \left(x^k - x^{k-1}\right), \tag{1.61c}$$

where $t^0 = 1$, and $\alpha^k \leq \frac{1}{L}$ is determined through a line search. If the Lipschitz constant is known, one may take the constant step size of $\frac{1}{L}$ in each iteration. Let us denote $\frac{t^k - 1}{t^{k+1}}$ by τ^k. Rewriting Nesterov's algorithm solely in terms of sequence $\{y^k\}$, we get

$$\begin{aligned} y^{k+1} &= y^k - \alpha^k \nabla f(y^k) + \tau^k \left(y^k - \alpha^k \nabla f(y^k) - y^{k-1} + \alpha^{k-1} \nabla f(y^{k-1})\right) \\ &= y^k + \tau^k (y^k - y^{k-1}) - (1 + \tau^k) \alpha^k \nabla f(y^k) + \tau^k \alpha^{k-1} \nabla f(y^{k-1}). \end{aligned}$$

Letting $d_y^{k+1} = y^{k+1} - y^k$, the above equation can equivalently be stated as

$$d_y^{k+1} = \tau^k d_y^k - (1 + \tau^k) \alpha^k \nabla f(y^k) + \tau^k \alpha^{k-1} \nabla f(y^{k-1}). \tag{1.62}$$

Assuming that nonlinear CG has the format mentioned in (1.60), d_y^{k+1} in nonlinear CG would be

$$d_y^{k+1} = \frac{\alpha^k \beta^k}{\alpha^{k-1}} d_y^k - \alpha^k \nabla f(y^k), \tag{1.63}$$

because

$$y^{k+1} = y^k + \alpha^k d^k,$$

where

$$d^k = \beta^k d^{k-1} - \nabla f(y^k) = \beta^k \left(\frac{y^k - y^{k-1}}{\alpha^{k-1}}\right) - \nabla f(y^k).$$

Comparing of (1.62) and (1.63) sheds more light on some similarities between CG and Nesterov's algorithm. In both algorithms d_y^{k+1} is a linear combination of the previous gradients; the difference lies in the coefficients used in the linear combinations.

Despite close similarities between Nesterov's, NY-CG, and traditional CG algorithms, the question of whether nonlinear CG can achieve the same complexity bound has remained unanswered. In Chapter 2 we derive conditions under which CG can achieve the same iteration bound of NY-CG. As a result of our study we propose an algorithm called CGSO that can achieve the same complexity bound with the cost of extra computational work per iteration.

We have already pointed out that the orthogonality of gradients does not hold for nonlinear CG, and in fact gradients may become nearly dependent causing a significant slow-down in the convergence of the algorithm. A natural measurement on the orthogonality of gradients is defined as $\frac{(\nabla f^k)^t \nabla f^{k-1}}{\|\nabla f^k\|^2} \geq \nu$; and a normal approach to fix the dependence of gradients is to restart CG by taking a steepest descent step. It is often believed in optimization community that the Polak-Ribière variant is more robust than Fletcher-Reeves nonlinear CG. Note that when $\nabla f^{k+1} \simeq \nabla f^K$, $\beta_{PR} \simeq 0$; thus the algorithm takes almost a steepest descent step at the next iteration. There is, however, no rigorous theory on when to restart a nonlinear CG, and how to prevent the dependence of gradients in subsequent iterations after a restart.

The theory of CGSO relies on two safeguard inequalities. Roughly speaking, one measures the amount of reduction in the objective value; while the other checks orthogonality of the gradients. Both these properties are crucial for reaching the optimal complexity bound of NY-CG. These safeguard conditions can be applied to any variant of nonlinear CG. In Section 2.2 we present how these conditions can effectively detect the loss of independence of gradients. Using CGSO, we also propose a technique for recovering the orthogonality of gradients. We refer to this procedure at "detection and correction" of loss of independence of gradients in CG.

1.3 Introduction to Sparse Recovery

Compressive Sensing (CS) [31, 5, 34] refers to the idea of encoding a large sparse signal through a relatively small number of linear measurements. This approach is essentially to apply a linear operator $A \in \mathbb{R}^{m \times n}$ to a signal $x \in \mathbb{R}^n$ and storing $\hat{b} = Ax$ instead. Naturally we want $\hat{b} \in \mathbb{R}^m$ to be of a smaller dimension than x; hence in practice $m \ll n$. The main question is how to decode $\hat{b}$ to recover signal x, i.e., finding the solution to the

underdetermined system of linear equations

$$Ax = \hat{b}. \tag{1.64}$$

Sparse recovery particularly aims at finding the sparsest solution to (1.64). The sparsest solution might be obtained by solving

$$\begin{array}{ll} \min & \|x\|_0 \\ s.t. & Ax = \hat{b}, \end{array} \tag{1.65}$$

where $\|x\|_0$ corresponds to the number of nonzero entries of x. Problem (1.65) is, however, NP-hard and difficult to solve, therefore the following relaxation was suggested for recovering the sparse solution:

$$\begin{array}{ll} \min & \|x\|_1 \\ s.t. & Ax = \hat{b}. \end{array} \tag{1.66}$$

The theory of compressive sensing has been well-established. Candes, Tao, Romberg, and Donoho are among the pioneers of compressive sensing theory, see [32, 33, 42]. In fact, they have shown that under some conditions, (1.66) can recover the solution to (1.65). Note that problem (1.66) can be reformulated as an LP; however general purpose LP solvers are not a suitable choice for solving it because of the excessive run time for large instances of the problem.

In the presence of the noise in computing and storing $\hat{b}$, the in-hand measurement is often $b = \hat{b} + \hat{\epsilon}$; hence it is customary to replace

$$Ax = \hat{b},$$

with

$$\|Ax - b\| \leq \epsilon,$$

in (1.66), where ϵ is an estimated upper bound on the noise. The resulting problem is

$$\begin{array}{ll} \min & \|x\|_1 \\ s.t. & \|Ax - b\| \leq \epsilon. \end{array} \qquad (\mathrm{BP}_\epsilon) \tag{1.67}$$

Problem (1.66) is usually referred to as "Basis Pursuit" (BP) problem, while BP_ϵ refers to its least-square constrained variant, i.e., (1.67).

Other common problems in sparse recovery are

$$\min \quad \tfrac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1, \qquad \text{(BPDN)} \tag{1.68}$$

and

$$\begin{array}{ll} \min & \|Ax-b\| \\ s.t. & \|x\|_1 \leq \tau. \end{array} \qquad \text{(LASSO)} \tag{1.69}$$

In the literature of compressive sensing, (1.68) is often called "Basis Pursuit Denoising Problem" (BPDN) or l_1-regularized least square problem, and (1.69) goes by the name of "LASSO" (Least Absolute Shrinkage and Selection Operator). Although in statistics, problem (1.68) is also referred to as LASSO, we stick with the customary labels used in compressive sensing.

Theoretically all the aforementioned formulations are equivalent, provided that some specific relations hold true for ϵ, λ, and τ. However, there is no simple manner to compute this relationship without already knowing the optimal solution. The algorithms proposed in this work are tailored for solving (1.68).

Compressive sensing has been a flourishing field of research for the last few decades. The formulations mentioned above (BP_ϵ, BPDN, and LASSO) rises in many applications including medical screening [79], machine learning [108], statistics [101, 118], missing data recovery [117, 99], and many other applications. One may refer to [103] for a vast list of references on the theory, applications, algorithms, and software of this field.

The key desired factors for an algorithm suited for solving any of the above formulations are speed and accuracy. Although efficient second order methods such as interior-point methods [89, 109, 94] can achieve high accuracy, solving large-scale instances can be challenging for these algorithms. First-order methods, on the other hand, are faster and more effective for solving large-scale problems rising in CS. There are numerous gradient-based first-order algorithms proposed for sparse recovery, see for example [83, 46, 21, 57, 18, 114, 43, 110, 11].

In [18, 17] an efficient root finding procedure has been employed for finding the solution of BP_ϵ through solving a sequence of LASSO problems. In other words a sequence of LASSO problems for different values of λ is solved using a spectral projected-gradient method [22]; and as $\lambda \to \lambda^\star$, the solution of the LASSO problem coincides with the solution of BP_ϵ. In [114], the solution of BP problem is recovered through solving a sequence of LASSO problems with an updated observation vector b. GPSR [46] is a gradient projection technique for solving the bound constrained QP reformulation of BPDN.

Many other state-of-the-art algorithms in CS are inspired by iterative thresholding/shrinkage idea [37, 38, 36]. The algorithms, however, can suffer from slow rate of convergence. Theoretically iterative thresholding algorithms have sublinear convergence in general and can achieve linear convergence only under some strict conditions [26]. The accelerated technique originally proposed by Nesterov [86] has been applied to iterative thresholding

algorithms to enhance their convergence rate, see [9] and references therein. Nesterov's accelerated proximal gradient algorithm has been adopted for solving different formulations in image processing such as BP_ϵ in [11], LASSO problem in [52], and total variation minimization in [40]. Other possible approaches for solving LASSO problem are augmented Lagrangian and penalty methods as in [72] and [73], respectively. Nesterov's algorithm is employed for solving the subproblem in both augmented Lagrangian and penalty methods.

In what follows we present iterative shrinkage idea and other relevant techniques to our work in more details. Similar to the algorithms we discussed for smooth unconstrained optimization problems, algorithms discussed in this section are also iterative techniques. Recall steepest descent method as the simplest gradient based method for unconstrained problems. The general format of the generated sequence by this algorithm is:

$$x^{k+1} = x - \alpha \nabla f(x^k). \tag{1.70}$$

We can consider the above sequence as solutions to the following quadratic approximations of function f, i.e.,

$$x^{k+1} = \arg\min_x \left\{ f(x^k) + \langle x - x^k, \nabla f(x^k) \rangle + \frac{1}{2\alpha} \|x - x^k\|^2 \right\}. \tag{1.71}$$

In the above model, one may think of

$$f(x^k) + \langle x - x^k, \nabla f(x^k) \rangle, \tag{1.72}$$

and

$$\frac{1}{2\alpha} \|x - x^k\|^2, \tag{1.73}$$

as the first order approximation to the objective function and the proximity term with weight $\frac{1}{\alpha}$, respectively. If $\alpha = \frac{1}{L}$, the model would have been the upper approximation of the objective function, and we would have the steepest descent with fixed step size.

"Iterative Shrinkage Thresholding Algorithm" (ISTA) is an extension of the steepest descent idea to composite functions using the thresholding operator.

Consider the problem of minimizing a composite function, i.e.,

$$\min \quad F(x) = f(x) + g(x). \tag{1.74}$$

The idea of steepest descent has been extended to minimizing a composite function by simply using the same approximation model as in (1.71) for $f(x)$. As a result we get the following iterative scheme:

$$x^{k+1} = \arg\min_x \left\{ f(x^k) + \langle x - x^k, \nabla f(x^k) \rangle + \frac{1}{2\alpha}\|x - x^k\|^2 + g(x) \right\}. \tag{1.75}$$

Shuffling the linear and quadratic terms and ignoring the constant terms in (1.75), it can equivalently be written as

$$x^{k+1} = \arg\min_x \left\{ \frac{1}{2\alpha}\|x - (x^k - \alpha\nabla f(x^k)\|^2 + g(x) \right\}. \tag{1.76}$$

Using the notion of Prox operator we conclude that

$$x^{k+1} = \text{Prox}_{\alpha g}\left(x^k - \alpha\nabla f(x^k)\right). \tag{1.77}$$

The iterative scheme of (1.77) is "Generalized Gradient Method" or "Proximal Gradient Method". Note that it actually coincides with steepest descent method in the absence of $g(x)$. Finding the proximal point may not be a trivial task in general, but for solving BPDN it can be computed efficiently because the l_1-norm is separable.

Iterative scheme (1.77) is often called "Forward-backward Splitting Method" [36, 111]. It gets its name from the two separate stages during each iteration while minimizing (1.74); the first stage is taking a forward step $x^k - \alpha\nabla f(x^k)$ involving only f, and the second stage is a backward step $\text{Prox}_{\alpha g}\left(x^k - \alpha\nabla f(x^k)\right)$ which involves only g.

The algorithm that goes by the name of ISTA in the literature of sparse recovery is derived from the same chain of arguments for composite functions in which $g(x) = \lambda\|x\|_1$. Using Definition 1.4 an ISTA step in its general form is

$$x^{k+1} = \mathcal{S}_{\lambda\alpha}\left(x^k - \alpha\nabla f(x^k)\right). \tag{1.78}$$

FISTA[1] [9] is the accelerated variant of ISTA that was built upon the Nesterov's idea [84, 86]. Each iteration of FISTA has the following format:

$$x^{k+1} = \mathcal{S}_{\lambda\frac{1}{L}}\left(y^k - \frac{1}{L}\nabla f(y^k)\right), \tag{1.79}$$

$$t^{k+1} = \frac{1 + \sqrt{1 + 4t^{k^2}}}{2}, \tag{1.80}$$

$$y^{k+1} = x^k + \left(\frac{t^k - 1}{t^{k+1}}\right)\left(x^{k+1} - x^k\right). \tag{1.81}$$

[1] Fast Iterative Shrinkage Thresholding Algorithm

When no knowledge of L is in hand, a backtracking line search is employed in (1.79). FISTA is not restricted to BPDN problem. In fact it was originally proposed for minimizing a general composite function. Replacing (1.79) with

$$x^{k+1} = \text{Prox}_{\lambda\frac{1}{L}}\left(y^k - \frac{1}{L}\nabla f(y^k)\right), \tag{1.82}$$

would generalize FISTA to an algorithm well-suited for minimizing composite functions.

In order to incorporate more information about the function without trading off the efficiency of the algorithms, Newton/quasi-Newton proximal methods [14, 74] has attracted researchers quite recently. Most of previous extensions on quasi-Newton methods are suited either for nonsmooth problems [75, 115], or for constrained problems with simple enough constraints [28, 98, 41]. For l_1-regularization, however, proximal quasi-Newton methods might be a more effective approach.

Recall that generalized gradient method was basically derived by approximating f with a quadratic model of the form

$$m_\alpha(x) = f(x^k) + \langle x - x^k, \nabla f(x^k)\rangle + \frac{1}{2\alpha}\|x - x^k\|^2 \tag{1.83}$$

$$= f(x^k) + \langle x - x^k, \nabla f(x^k)\rangle + \frac{1}{2}\left(x - x^k\right)^t\left(\frac{1}{\alpha}\mathbf{I}\right)\left(x - x^k\right), \tag{1.84}$$

and solving

$$\min \quad m_\alpha(x) + g(x), \tag{1.85}$$

in each iteration. A natural extension of the above algorithm is to replace the diagonal matrix $\frac{1}{\alpha}\mathbf{I}$ in the quadratic term of $m_\alpha(x)$ with a suitable positive definite matrix. In other words, one may define $m_H(x)$ as

$$m_H(x) = f(x^k) + \langle x - x^k, \nabla f(x^k)\rangle + \frac{1}{2}\left(x - x^k\right)^t H\left(x - x^k\right), \tag{1.86}$$

where $H \succ 0$, and solve

$$\min \quad m_H(x) + g(x), \tag{1.87}$$

instead of (1.85). Ignoring the constant terms and using definition of scaled norm, we can rewrite (1.86) as

$$m_H(x) = \frac{1}{2}\|x - \left(x^k - H^{-1}\nabla f(x^k)\right)\|_H^2. \tag{1.88}$$

By definition of scaled proximal mapping, we can now define the proximal quasi-Newton algorithm as

$$x^{k+1} = \text{Prox}_g^H \left(x^k - H^{-1} \nabla f(x^k) \right). \tag{1.89}$$

If $H = \nabla^2 f(x^k)$ in (1.89), then we obtain the proximal Newton method.

Alternating Direction Method (ADM), see [112, 113] and references therein, is also a technique that can be applied to BPDN. It is suited for minimizing the summation of (separable) convex functions, say $f(x) + g(y)$, over a linear set of constraints. The augmented Lagrangian technique then solves for x and y interchangeably while fixing the other variable. Alternating Linearization Method (ALM) [49] also applies to minimizing composite functions. In (1.75), we linearize f at every iteration to build the quadratic approximation model; in ALM a similar model based on g is also minimized at every iteration. Nesterov's accelerated technique has also been adopted and the resulting algorithm is called FALM, for fast ALM.

A randomized first-order technique is proposed in [65] suitable for solving bilinear saddle point problem. It is, also, shown that BP_ϵ formulation can be transformed into an equivalent bilinear saddle point formulation and solved through the proposed first-order scheme. The presented numerical results suggest that the randomization can enhance the practical performance of first-order techniques particularly for large-scale problems. This issue has attracted researchers lately and is discussed in [15] as well.

1.4 Outline of the Thesis

We mentioned in the introduction that a key question we aim to answer in this thesis is whether conjugate gradient can achieve the optimal complexity bound of Nemirovsky-Yudin's and Nesterov's algorithms. As a result of our study, we propose a variant of conjugate gradient algorithm named CGSO that achieves the desired complexity bound. Moreover, the analysis of CGSO suggests a new scheme for detecting and correcting the loss orthogonality of gradients in other variants of nonlinear conjugate gradient. CGSO and the detection and correction procedure for avoiding the slow-down in nonlinear conjugate gradient is presented in Chapter 2. We extend the theory of CGSO to convex problems with polyhedral constraints; and apply the resulting technique to solving the basis pursuit denoising problem (BPDN); this result is presented in Appendix A. In Chapter 3, we review Nesterov's optimal method for minimizing strongly convex functions. Despite its rich theory, Nesterov's method in some cases has slower convergence than conjugate gradient methods. We present a hybrid scheme to combine conjugate gradient and Nesterov's method that benefits from the theory of Nesterov's method as well as the practical

efficiency of conjugate gradient. In Chapter 4, we propose a novel and effective proximal quasi-Newton scheme called IMRO for solving BPDN. The established theory of IMRO is not restricted to BPDN, and can be employed to the general problem of minimizing a convex function. We also present an accelerated variant of this technique that is inspired by Nesterov's method. Chapter 5 concludes IMRO by presenting the computational experiment with this algorithm. Finally in Chapter 6 we conclude our discussion and present several venues for future work.

Chapter 2

Conjugate Gradient with Subspace Optimization

We introduced the conjugate gradient (CG) technique in Section 1.2, and as mentioned, nonlinear CG algorithms all have the property that when applied to a quadratic objective function and coupled with an exact line search, they reduce to linear CG (LCG). For LCG we have the following results.

Theorem 2.1. *[90, Theorem 5.2] Suppose $\{x^k\}$ is a sequence generated by the LCG for minimizing a convex quadratic function f, then*

$$(\nabla f^k)^t p^i = 0 \quad \forall i = 0, 1, \ldots, k-1,$$

where p^i is the search direction at i-th iteration. Moreover x^k is the minimizer of f over the space of

$$\left\{x : x = x^0 + Span\left\{p^0, p^1, ..., p^{k-1}\right\}\right\}.$$

Theorem 2.2. *[90, Theorem 5.2] At each iteration k, either x^k is the optimal solution or the properties below hold.*

$$\begin{aligned}
(\nabla f^k)^t \nabla f^i &= 0, \quad \text{for } \ i = 0, 1, \ldots, k-1, \\
Span\left\{\nabla f^0, \nabla f^1, \ldots, \nabla f^k\right\} &= Span\left\{\nabla f^0, A\nabla f^0, \ldots, A^k \nabla f^0\right\}, \\
Span\left\{p^0, p^1, \ldots, p^k\right\} &= Span\left\{\nabla f^0, A\nabla f^0, \ldots, A^k \nabla f^0\right\}, \\
(p^k)^t A p^i &= 0, \quad \text{for } \ i = 0, 1, \ldots, k-1.
\end{aligned}$$

Theorem 2.2 indicates that LCG finds the solution in at most n iterations. Nonlinear CG, however, has no known complexity bound when applied to nonquadratic functions. Indeed, Nemirovsky and Yudin [82] argue that their worst-case complexity for strictly convex functions is quite poor.

Nemirovsky and Yudin, on the other hand, found a different generalization of CG with worst-case complexity bound of $O(\ln(1/\epsilon)\sqrt{L/l})$ iterations for the class of $\mathbf{C}_{l,L}$ functions. Actually, their algorithm, which we refer to as NY-CG, is not a generalization of the CG algorithm itself but rather a derivation from some equations and inequalities that underlie it:

$$f(x^{k+1}) \leq f(x^k) - \frac{1}{2L}\|\nabla f(x^k)\|^2, \tag{2.1a}$$

$$\nabla f(x^{k+1}) \perp \left(\nabla f(x^k) + \cdots + \nabla f(x^0)\right), \tag{2.1b}$$

$$\nabla f(x^{k+1}) \perp x^{k+1} - x^0, \tag{2.1c}$$

$$\langle \nabla f(x^k), x^* - x^k \rangle \leq f(x^*) - f(x^k), \tag{2.1d}$$

$$\nu_f(x^0) := f(x^0) - f(x^*) \geq \frac{l}{2}\|x^* - x^0\|^2. \tag{2.1e}$$

Note that in linear CG, x^{k+1} is the minimizer of $f(x)$ on the space mentioned in Theorem 2.1, therefore the first condition is true because $f(x^{k+1}) \leq f(x^k - \frac{1}{L}\nabla f(x^k))$ and $f \in \mathbf{C}_{l,L}$. The next two conditions follows from Theorem 2.2. Fourth and fifth conditions hold true by convexity and strong convexity of function f, respectively.

As an algorithm that inherits all the mentioned properties, Nemirovsky and Yudin suggest to find iterate x^{k+1} by performing a two dimensional search on the plane passing through $x^0, q^k + x^0$, and $\bar{x}^k = x^k - \frac{1}{L}\nabla f^k$, where $q^k = \nabla f^0 + \ldots + \nabla f^k$, i.e., a general step of NY-CG is

$$x^{k+1} = \arg\min_{x \in E^k} f(x), \quad \text{where} \quad E^k = x^0 + \text{Span}\left\{q^k, \bar{x}^k - x^0\right\}. \tag{2.2}$$

Clearly all of the properties listed in (2.1) hold for NY-CG method. Inequality (2.1a) holds because $f(x^{k+1}) \leq f(\bar{x}^k)$; properties (2.1b) and (2.1c) are true because $(\nabla f^{k+1})^t q^k = 0$ and $(\nabla f^{k+1})^t \bar{x}^K = 0$; and the other two inequalities are properties of the function. Using (2.1), Nemirovski and Yudin [82] showed that their method achieves the complexity bound of $O(\ln(1/\epsilon)\sqrt{L/l})$, where ϵ is the desired accuracy. Moreover, they showed that this complexity bound is optimal and can not be improved further. In other words, no first-order algorithm in which $x^k \in x^0 + \text{Span}\left\{\nabla f^0, \nabla f^1, \ldots, \nabla f^k\right\}$, can achieve a better complexity bound for the class of $\mathbf{C}_{l,L}$ functions.

A disadvantage of NY-CG is that it has a relatively expensive computation on every iteration, namely, one must solve a two-dimensional convex optimization problem using the ellipsoid method[82, 105]. Although, as mentioned before, Nesterov's algorithm [84] remedies this disadvantage, both NY-CG and Nesterov's algorithm does not reduce to linear CG (and in fact, is much slower in practice). A second drawback is that both algorithms require prior knowledge of the ratio L/l. This is because they need to be restarted every $O(\sqrt{L/l})$ iterations in order to achieve the optimal complexity bound.

The purpose of conjugate gradient with subspace optimization (CGSO) is to avoid the disadvantages mentioned above. In particular, we seek a CG-like algorithm with the following properties:

1. The algorithm should reduce to linear CG when the objective function is quadratic.
2. The algorithm should have the complexity bound of $O(\ln(1/\epsilon)\sqrt{L/l})$ for the class of $\mathbf{C}_{l,L}$ functions.
3. The algorithm should not require prior knowledge of any parameters describing f.
4. The cost per iteration should not be excessive.

The algorithm we propose achieves goals 1–3, and mostly achieves goal 4. It is built upon NY-CG, and like NY-CG it must solve a convex optimization subproblem on every iteration. The dimension of the subproblem is always at least 2 and is determined adaptively by the algorithm. The worst-case upper bound we are able to prove for the dimension of this subproblem is $O(\log j)$, where j is the number of iterations so far. However, in our testing, the dimension of the subproblem was 2 in almost all instances; in one test case it reached the value 3 for a few iterations, but it never exceeded 3. Furthermore, we find that solving the subproblem is usually fairly efficient because we usually apply Newton's method when possible. Note that Newton's method is not even defined for $\mathbf{C}_{l,L}$ functions in general. In a case that Newton's method fails to be defined or fails to converge, CGSO falls back on a first-derivative method as mentioned below. However, Newton's method in the subspace proved to be a valuable technique in practice according to our computational experiments.

Goal 3, the condition of no prior knowledge of parameters, has the obvious benefit of making the algorithm easier to apply in practice. It also has a second subtler benefit. For some convex problems, e.g., minimization of a log-barrier function, there is no prior bound on L/l over the domain of the function since the derivatives tend to infinity at the

boundaries. However, as the minimizer is approached, the bad behavior at the boundaries becomes irrelevant, and there is a new smaller ratio L/l relevant for level sets in the neighborhood of the minimizer. In this case, CGSO automatically adapts to the improved value of L/l. Such adaptation is possible also with NY-CG, and Nesterov's algorithm too, but, as far as we know, the adaptation must be done by the user and cannot be easily automated.

2.1 CGSO

2.1.1 The Algorithm

CGSO, in its preliminary form, is as follows:

$x^0 =$ arbitrary;
for $j = 0, 1, \ldots$
$x^{j+1} = x^j + \alpha^j \nabla f(x^j) + \beta^j d^j$,
where $d^j = x^j - x^{j-1}$ and $\alpha^j, \beta^j = \arg\min_{\alpha,\beta} f(x^j + \alpha \nabla f(x^j) + \beta d^j)$

As we shall see, some modifications are necessary to achieve the desired complexity bound. The above algorithm is certainly a generalization of nonlinear CG, since each search direction is a combination of previous gradients. Moreover it reduces to LCG when applied to a quadratic function because x^{j+1} is the minimizer of f over the space of $x^j + \{\nabla f^j, d^j\}$ in both algorithms. Notice that the above algorithm is a descent method (i.e., $f(x^{j+1}) \leq f(x^j)$), and it performs at least as well as steepest descent in each iteration. Hence it converges to a stationary point, which, for the class of convex problems, coincides with the minimizer of the function.

Let $\nu_f(x)$ denote the residual of the function, i.e., $f(x) - f(x^*)$. By the fact that $f \in \mathbf{C}_{l,L}$, we get the following properties for the above algorithm

$$f(x^{j+1}) \leq f(x^j) - \frac{1}{2L}\|\nabla f^j\|^2, \tag{2.3a}$$

$$\langle \nabla f^j, x^* - x^j \rangle \leq f(x^*) - f(x^j), \tag{2.3b}$$

$$\nu_f(x^0) = f(x^0) - f^* \geq \frac{l}{2}(x^* - x^0)^2. \tag{2.3c}$$

Similar to NY-CG, property (2.3a) follows from the fact that $f(x^{j+1}) \leq f(x^j - \frac{1}{L}g^j)$ and $f \in \mathbf{C}_{l,L}$. Property (2.3b) is true by convexity of the function, and property (2.3c) is a

derivation from the strong convexity of f. We are now ready to present the main lemma on the complexity bound of this algorithm.

Lemma 2.1. *Suppose* $m \geq \left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil$ *and*

$$\frac{(f(x^{m-1}) - f(x^0))}{4}\left(\sum_{j=0}^{m-1}\lambda^j\right) + \sum_{j=0}^{m-1}\lambda^j \left\langle \nabla f(x^j), x^j - x^0 \right\rangle < 0, \tag{2.4}$$

and

$$\left\| \sum_{j=0}^{m-1}\lambda^j \nabla f(x^j) \right\| \leq \rho\sqrt{\sum_{j=0}^{m-1}(\lambda^j)^2 \left\|\nabla f(x^j)\right\|^2}, \tag{2.5}$$

are satisfied for a constant ρ*, where*

$$\lambda^j = \sqrt{\frac{f(x^j) - f(x^{j+1})}{\|\nabla f(x^j)\|^2}}. \tag{2.6}$$

Then the residual of the function is divided in half after m *iterations; i.e.,* $\nu_f(x^m) \leq \frac{1}{2}\nu_f(x^0)$.

Before we provide the proof of the above lemma, we would like to elaborate on parameter ρ. In the first glance at Lemma 2.1, the fact that m depends on ρ while ρ, in theory, can be as large as m might be confusing. Note that when ∇f^k is orthogonal to $\nabla f^0, \ldots, \nabla f^{k-1}$ (as in linear CG) or $\nabla f^0 + \ldots + \nabla f^k$ (as in NY-CG), ρ is 1. Although the orthogonality of the gradients are no longer guaranteed for CGSO, we expect that the gradients remain nearly orthogonal. Our experiment with CGSO suggests that ρ is normally a constant far below m, typically two at the beginning and converging to one as we get closer to the optimizer. We will discuss more details about ρ in Section 2.3.2; but for now, let us assume that ρ is a given parameter equal to a small finite number, say less than five.

Proof. Our proof is an extension of the proof in [82, Section 7.3] in which some derivations depend on inequalities (2.1b) and (2.1c), while here we rely on inequalities (2.4) and (2.5).

Suppose by contradiction that $m \geq \left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil$, (2.4) and (2.5) are satisfied, but $\nu_f(x^m) > \frac{\nu_f(x^0)}{2}$.

By definition of λ^j,

$$f(x^{j+1}) = f(x^j) - \left(\lambda^j\right)^2 \|\nabla f(x^j)\|^2,$$

hence

$$\nu_f(x^{j+1}) = \nu_f(x^j) - \left(\lambda^j\right)^2 \|\nabla f(x^j)\|^2.$$

Summing these equalities over $j = 0, \cdots, m-1$, we get

$$0 \leq \nu_f(x^m) = \nu_f(x^0) - \sum_{j=0}^{m-1} \left(\lambda^j\right)^2 \|\nabla f(x^j)\|^2,$$

or equivalently,

$$\sum_{j=0}^{m-1} \left(\lambda^j\right)^2 \|\nabla f(x^j)\|^2 \leq \nu_f(x^0). \tag{2.7}$$

By convexity of the function we have

$$\left\langle \nabla f(x^j), x^* - x^j \right\rangle \leq f(x^*) - f(x^j) = -\nu_f(x^j), \tag{2.8}$$

and so

$$\left\langle \nabla f(x^j), x^* - x^0 \right\rangle - \left\langle \nabla f(x^j), x^j - x^0 \right\rangle \leq -\nu_f(x^j) \leq -\nu_f(x^m) < \frac{-\nu_f(x^0)}{2}. \tag{2.9}$$

Let us consider the weighted sum of all the above inequalities for $j = 0, \ldots m-1$ with weights λ^j's to get

$$\left\langle \sum_{j=0}^{m-1} \lambda^j \nabla f(x^j), x^* - x^0 \right\rangle - \sum_{j=0}^{m-1} \lambda^j \left\langle \nabla f(x^j), x^j - x^0 \right\rangle < \frac{-\nu_f(x^0)}{2} \left(\sum_{j=0}^{m-1} \lambda^j \right),$$

which can be rearranged to the following form

$$\left\langle \sum_{j=0}^{m-1} \lambda^j \nabla f(x^j), x^* - x^0 \right\rangle < -\frac{\nu_f(x^0)}{2} \left(\sum_{j=0}^{m-1} \lambda^j \right) + \sum_{j=0}^{m-1} \lambda^j \left\langle \nabla f(x^j), x^j - x_0 \right\rangle.$$

Equivalently we can rewrite the above inequality as

$$\begin{aligned} \left\langle \sum_{j=0}^{m-1} \lambda^j \nabla f(x^j), x^* - x^0 \right\rangle < & -\frac{\nu_f(x^0)}{4} \left(\sum_{j=0}^{m-1} \lambda^j \right) \\ & + \left(\frac{f(x^*) - f(x^0)}{4} \left(\sum_{j=0}^{m-1} \lambda^j \right) + \sum_{j=0}^{m-1} \lambda^j \left\langle \nabla f(x^j), x^j - x^0 \right\rangle \right). \end{aligned}$$

Using inequality (2.4) along with the facts that $f(x^*) \leq f(x^j)$ and $\lambda^j \geq 0$ for all j, we get

$$\left\langle \sum_{j=0}^{m-1} \lambda^j \nabla f(x^j), x^* - x^0 \right\rangle < -\frac{\nu_f(x^0)}{4} \left(\sum_{j=0}^{m-1} \lambda^j \right). \tag{2.10}$$

By the Cauchy-Schwarz inequality we have

$$-\left\| \sum_{j=0}^{m-1} \lambda^j \nabla f(x^j) \right\| \left\| x^* - x^0 \right\| \leq \left\langle \sum_{j=0}^{m-1} \lambda^j \nabla f(x^j), x^* - x^0 \right\rangle < -\frac{\nu_f(x^0)}{4} \left(\sum_{j=0}^{m-1} \lambda^j \right),$$

hence

$$\left\| \sum_{j=0}^{m-1} \lambda^j \nabla f(x^j) \right\| \left\| x^* - x^0 \right\| > \frac{\nu_f(x^0)}{4} \left(\sum_{j=0}^{m-1} \lambda^j \right). \tag{2.11}$$

By property (2.3c) we have

$$\left\| x^* - x^0 \right\| \leq \sqrt{\frac{2\nu_f(x^0)}{l}}. \tag{2.12}$$

Furthermore, by inequalities (2.5) and (2.7) we get

$$\left\| \sum_{j=0}^{m-1} \lambda^j \nabla f(x^j) \right\| \leq \rho \sqrt{\sum_{j=0}^{m-1} (\lambda^j)^2 \left\| \nabla f(x^j) \right\|^2} \leq \rho \sqrt{\nu_f(x^0)}. \tag{2.13}$$

Replacing inequalities (2.12) and (2.13) in inequality (2.11), we get

$$\rho \sqrt{\nu_f(x^0)} \sqrt{\frac{2\nu_f(x^0)}{l}} > \frac{\nu_f(x^0)}{4} \left(\sum_{j=0}^{m-1} \lambda^j \right). \tag{2.14}$$

Notice that by definition of λ and property (2.3a), $\lambda^j \geq \sqrt{\frac{1}{2L}}$ for all j, so

$$\sum_{j=0}^{m-1} \lambda^j \geq \sqrt{\frac{1}{2L}}\, m. \tag{2.15}$$

Using this fact in inequality (2.14), we get

$$\rho \sqrt{\nu_f(x^0)} \sqrt{\frac{2\nu_f(x^0)}{l}} > \frac{\nu_f(x^0)}{4} \left(\sqrt{\frac{1}{2L}}\, m \right), \tag{2.16}$$

therefore

$$m < 8\rho\sqrt{\frac{L}{l}}, \tag{2.17}$$

which contradicts our assumption on the value of m. □

Lemma 2.1 shows that under conditions (2.4) and (2.5), the residual of the function is divided in half every $m = O\left(\sqrt{\frac{L}{l}}\right)$ iterations. For the next m iterations, a further reduction of $\frac{1}{2}$ is achieved provided that (2.4) and (2.5) hold with x^m substituted in place of x^0. Hence by letting x^m to be the new x^0 and repeating the same algorithm, we can find the ϵ-optimal solution in $\lceil \log_2 \frac{1}{\epsilon} \rceil \left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil$ iterations. Of course, m is not known to our algorithm, a point to which we return in the next section.

2.1.2 Restarting

We have already seen in the previous section that in CGSO, the residual of the function is divided in half every m iterations if (2.4) and (2.5) are satisfied. The algorithm, then, needs to be restarted with x^m substitutes for x^0 to get further reductions in the residual of the function. A closer look at the algorithm, however, clarifies that finding each iterate is independent of knowing x^0. Initial point x^0 is only required for checking inequalities (2.4) and (2.5). We use the term "restarting" to refer to the process of replacing x^0 in inequalities (2.4) and (2.5) with x^m for some $m > 0$. Notice that this process does not change any of the iterates, and it solely changes the interval of indices in which we check inequalities.

Knowing L and l, we can compute m directly; hence we would be able to determine exactly when we need to restart the algorithm. In many cases, however, finding the parameters of the function is a nontrivial problem itself. In order to have an algorithm which does not rely on any prior knowledge about the function, we propose the following technique.

Suppose p is an integer for which $2^p \leq m \leq 2^{p+1}$. By restarting the algorithm every 2^{p+1} iterations, we are guaranteed that the residual of the function is divided in half between every two consecutive restarts that contains at most $2m = \lceil 16\rho\sqrt{\frac{L}{l}} \rceil$ iterates. The last statement follows from the fact that $2^{p+1} \leq 2m$. Since the exact value of p is usually unknown, we repeat the above procedure for every value of $p \in \{\varphi_l, \ldots, \varphi_u\}$, where φ_l and φ_u are lower and upper bound estimates on the value of p. Typically several values of p suffice since the size of the intervals corresponding to p's grows exponentially. Furthermore p never exceeds $\lceil \log_2 j \rceil$, where j is the current iteration count; because inequalities (2.4)

and (2.5) are the same for all $p \geq \lceil \log_2 j \rceil$. We can now state the CGSO algorithm in a more complete form:

$x^0 =$ arbitrary, ρ given, $d^0 = 0$.
for $j = 0, 1, \ldots$
$x^{j+1} = \arg\min_{x \in E^j} f(x)$ where $E^j = x^j + \text{Span}\{\nabla f(x^j), d^j\}$
Let $\lambda^j = \sqrt{\frac{f(x^j) - f(x^{j+1})}{\|\nabla f(x^j)\|^2}}$
for $p = \varphi_l, \ldots, \lceil \log_2 j \rceil$
if $j + 1 = \kappa_p 2^p$ for some integer κ_p
Let $r_p = (\kappa_p - 1)\, 2^p$;
check
$$\frac{\left(f(x^{j+1}) - f(x^{r_p})\right)}{4} \left(\sum_{i=r_p}^{j} \lambda^i\right) + \sum_{i=r_p}^{j} \lambda^i \left\langle \nabla f(x^i), x^i - x^{r_p} \right\rangle < 0$$
and
$$\left\| \sum_{i=r_p}^{j} \lambda^i \nabla f(x^i) \right\| \leq \rho \sqrt{\sum_{i=r_p}^{j} (\lambda^i)^2 \, \|\nabla f(x^i)\|^2}$$
If any of the above inequalities fails, take the "correction step".
(which will be defined in the subsequent section)

2.1.3 Correction Step

For convenience, let us refer to the set of iterates between two consecutive restarts as a "block" of iterates. In other words for any p, the set of iterates $x^0, x^1, \ldots, x^{2^p - 1}$ is the first block of size 2^p which we refer to as first block of p; $x^{2^p}, x^{2^p+1}, \ldots, x^{2(2^p)-1}$ is the second block of p, and so on. At the end of each block we check inequalities (2.4) and (2.5). If they are satisfied and $2^p \geq \left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil$, then by Lemma 2.1 we know that the residual of the function is divided in half. On the other hand if any of these inequalities fails, then as mentioned in the previous section we need to take "correction step" for the next block of iterates. The correction step is basically computing the next block of iterates in a way that satisfaction of inequalities (2.4) and (2.5) is guaranteed for that particular block. Then the correction step is omitted in the subsequent blocks until the inequalities are violated again.

Recall that in an ordinary step, the new iterate, x^{j+1} is calculated by a two-dimensional search on the plane passing through x^j, and parallel to the two-dimensional subspace spanned by $\nabla f(x^j)$, and d^j. Suppose at least one of the inequalities (2.4) and (2.5) is violated for κth block of p; i.e., for the block of iterates $x^{r_p}, \ldots, x^{r_p + 2^p - 1}$, where $r_p =$

$(\kappa-1)2^p$. In other words, when checking the inequalities at iteration r_p+2^p-1 they fail. Then for the next block of p, i.e., for iterations $r_p+2^p(=\kappa 2^p), \kappa 2^p+1, \ldots, (\kappa+1)2^p-1$, we search for the new iterate x^{j+1} on the space of $x^j+\text{Span}\left\{\nabla f(x^j), d^j, q_p^j, x^j-x^{r_p}\right\}$, where $q_p^j=\sum_{i=r_p}^{j}\lambda^i\nabla f(x^i)$. Finding the new iterate x^{j+1} through a search on the space that in addition to $\nabla f(x^j)$ and d^j includes q_p^j and $x^j-x^{r_p}$ is what we refer to as "correction step". Notice that for each p with the violated constraints we increase the dimension of the search space by 2. However, the dimension of the search space never exceeds $O(\varphi_u)=2+2\lceil\log_2 j\rceil$, which happens to be the case when the inequalities are violated for all possible values of p.

It is quite easy to see that inequalities (2.4) and (2.5) are satisfied for the $(k+1)$th block of p when we take the correction step throughout it. By the first order optimality condition and the chain rule, we have $\langle\nabla f(x^j), x^j-x^{r_p}\rangle=0$ for all j in this block. Using this, along with the fact that $f(x^j)<f(x^{r_p})$ and non-negativity of λ^j for all j, we derive (2.4). Similarly one can argue that $\left\langle\nabla f(x^j), q_p^{j-1}\right\rangle=0$ for all j, hence

$$\left\|\sum_{i=r_p}^{r_p+2^p-1}\lambda^i\nabla f(x^i)\right\|=\sqrt{\sum_{i=r_p}^{r_p+2^p-1}(\lambda^i)^2\left\|\nabla f(x^i)\right\|^2},$$

which means inequality (2.5) is satisfied. After finding the iterates of one block through a correction step, the algorithm switches back to taking a regular step until the next failure of the inequalities. We can now present the algorithm in its entirety. To save space, we use the tabbing convention that the end of a code-block is denoted by a retraction of the indent-level.

Algorithm 2.1.

$x^0=$ *arbitrary*, ρ *given*, $d^0=0$, $S=\emptyset$.
for $j=0,1,\ldots$
$x^{j+1}=\arg\min_{x\in E^j} f(x)$ *where* $E^j=x^j+Span\left\{\nabla f(x^j), d^j, \cup_{p\in S}q_p^j, \cup_{p\in S}x^j-x^{r_p}\right\}$
Let $\lambda^j=\sqrt{\frac{f(x^j)-f(x^{j+1})}{\|\nabla f(x^j)\|^2}}$
for $p=\varphi_l,\ldots,\lceil\log_2 j\rceil$
if $j+1=\kappa_p 2^p$ *for some integer* κ_p
Let $r_p=(\kappa_p-1)\,2^p$
if $p\in S$
$S=S\setminus\{p\}$
else *(i.e., if* $p\notin S$*),*
check
$\frac{\left(f(x^{j+1})-f(x^{r_p})\right)}{4}\left(\sum_{i=r_p}^{j}\lambda^i\right)+\sum_{i=r_p}^{j}\lambda^i\left\langle\nabla f(x^i), x^i-x^{r_p}\right\rangle<0$

and
$\left\|\sum_{i=r_p}^{j} \lambda^i \nabla f(x^i)\right\| \leq \rho\sqrt{\sum_{i=r_p}^{j} (\lambda^i)^2 \left\|\nabla f(x^i)\right\|^2}$
if *any of the above inequalities fails,* $S = S \cup \{p\}$

We use Newton's method (with no line search) for solving the subspace optimization subproblem in algorithm 2.1 mostly because of its fast rate of convergence. It has been shown that under some assumptions, Newton's method convergences quadratically, see [90, Theorem 3.5]. Note that these assumptions are beyond strong convexity of the function and rely on twice differentiability of the function and Lipschitz continuity of the Hessian. Newton's method succeeds in many cases nonetheless. In the case of failure of Newton's algorithm, the ellipsoid method carries out the task of solving the subspace optimization problem. In other words we assign an upper bound to the number of iterations that Newton's method may take, and if it fails to converge within the given number of iterations CGSO switches to the ellipsoid method for finding the next iterate. The main reason that we do not solely rely on ellipsoid method for solving the subproblem is its convergence rate, see [105, Section 6.3]. It can, also, be much slower than Newton's method in practice. More details on implementation of the algorithm will be covered in Section 2.3.2.

We next turn to checking (2.4) and (2.5). It is apparent from their form that they can be checked efficiently by keeping running totals of all the summations appearing in them. This is how we have implemented them. The extra cost for tracking these summations is very low compared to the existing cost of evaluating f and ∇f in an ordinary CG iteration. For the storage, we need to store x^j and directions in the subspace E^j, i.e., ∇f^j, d^j, q_p^j for all $p \in S$, and $x^j - x^{r_p}$ for all $p \in S$. We also need to update and store x^{r_p}, $\sum_{i=r_p}^{j} \lambda^i$, $\sum_{i=r_p}^{j} \lambda^i \langle \nabla f^i, x^i - x^{r_p} \rangle$, $\sum_{i=r_p}^{j} \lambda^i \nabla f^i$, $\sum_{i=r_p}^{j} (\lambda^i)^2 \left\|\nabla f^i\right\|^2$ for all $p \in \{\varphi_l, \ldots, \log_2 j\}$. Let ω denote the dimension of the subproblem. Hence, the required storage space for the above elements is in $\mathcal{O}(n \max\{\omega, \log_2 j\})$. We have observed that ω is typically two. The maximum value we observed for ω was three. This is a point to which we return in Section 2.3.2.

Theorem 2.3. *Suppose* $m \geq \left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil$*, and* $\{x^j\}$ *is a sequence generated by Algorithm 2.1 for solving problem (1.59); then* $\nu_f(x^{(n+4)m}) \leq \frac{1}{2}\nu_f(x^{nm})$.

Proof. Let $\bar{p} \leq \varphi_u$ be the integer for which $2^{\bar{p}-1} \leq m \leq 2^{\bar{p}}$; and let $s_{\bar{p}}$ stand for $2^{\bar{p}}$. Consider any two consecutive blocks of size $s_{\bar{p}}$. Then either inequalities (2.4) and (2.5) hold at the end of the first block, or Algorithm 2.1 takes the correction step throughout the second

block which guarantees the satisfaction of the inequalities at the end of the second block. Therefore inequalities (2.4) and (2.5) are satisfied for at least one of any two consecutive blocks of size $s_{\bar{p}}$. The size of this block is $s_{\bar{p}} \geq m \geq \left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil$ and hence by Lemma 2.1 we have

$$\nu_f(x^{nm+2s_{\bar{p}}}) \leq \frac{1}{2}\nu_f(x^{nm}). \tag{2.18}$$

Since $2s_{\bar{p}} \leq 4m$, so $f(x^{nm+4m}) \leq f(x^{nm+2s_{\bar{p}}})$; hence

$$\nu_f(x^{nm+4m}) \leq \nu_f(x^{nm+2s_{\bar{p}}}). \tag{2.19}$$

(2.18) and (2.19) gives us the result we wanted to show. □

2.2 Detection and Correction of Loss of Independence

In the previous section we proposed a variant of nonlinear CG named CGSO, that achieves the optimal complexity bound of the first-order algorithms for the class of functions $f \in \mathbf{C}_{l,L}$. Each iteration of CGSO, requires a relatively expensive subspace optimization compared to traditional variants of nonlinear CG that may overshadow the theoretical advantages of the algorithm. As explained, the convergence of CGSO relies on two safeguard inequalities, and a correction procedure when these inequalities fail. In this section, we aim to combine the theory of CGSO with traditional nonlinear CG to obtain an algorithm that has lower computational cost than CGSO, but achieves the same optimal complexity bound.

Nonlinear CG has been widely used in practice, mainly because of its efficiency and low computational cost. The performance of nonlinear CG, however, can be affected by the loss of independence of gradients. In fact, the main drawback of nonlinear CG over the linear CG is the loss of orthogonality of search directions. Orthogonality of search directions is essentially the reason that safeguard inequalities, (2.4) and (2.5), always hold for linear CG.

The standard technique to combat loss of independence is restarting the method, i.e., occasionally taking a step of pure steepest descent. However, there is little rigorous theory that explains when to restart the method. The best known rigorous result in this direction is a proof that when an iterate is sufficiently close to the root, if one restarts every n iterations, one is guaranteed n-step quadratic convergence to the optimizer [35]. Here, n denotes the number of variables. This result is unsatisfying for at least two reasons.

First, there is no apparent method to detect when an iterate is sufficiently close to the root in order to apply this theorem. Second, restarting every n iterations does not seem to be practically motivated. The reason is that the convergence of conjugate gradient, both linear and nonlinear, is much more closely tied to the conditioning of the problem than to n, the number of variables. Thus, one would apparently prefer a rigorously supported restart strategy that is condition-dependent rather problem size-dependent.

It is clear from our analysis in the previous section, namely Lemma 2.1, that the convergence of CGSO mostly depends on inequalities (2.4) and (2.5). In fact, our approach for generating the sequence of iterates, i.e. the two dimensional subspace optimization, increases the chance that the aforementioned inequalities hold but does not guarantee it. The correction step is actually a consequence of their failure. Moreover, recall that if the gradients are orthogonal then both inequalities hold; therefore their failure is a fair indicator of loss of independence of gradients. This suggests a rigorous procedure for detecting and correcting the loss of independence in any variant of CG.

Suppose $\{x^k\}$ is a sequence of iterates generated by any variant of CG. We check inequalities (2.4) and (2.5) for $\{x^k\}$ in the same fashion that we check them for CGSO. We define the phrase "loss of independence" to mean the failure of these inequalities. When loss of independence is detected, we take the correction step according to what was described in Section 2.1.3. Similar to CGSO, we use Newton's and ellipsoid methods to solve the subspace subproblem when the correction step is taken. The detection procedure is relatively cheap; the correction procedure, however, is more expensive. Nonetheless, nonlinear conjugate gradient (any variant) augmented by our correction procedure in practice is sometimes the fastest method for solving the problem, according to our experiments detailed in Section 2.3.3.

Recall that another requirement for getting the optimal complexity bound of NY-CG through Lemma 2.1 is $f(x^{k+1}) \leq f(x^k) - \frac{1}{2L}\|\nabla f(x^k)\|^2$. For CGSO, this condition always holds because $f(x^{k+1}) \leq f\left(x^k - \frac{1}{L}\nabla f(x^k)\right)$; however for a general nonlinear CG algorithm this may not be the case. If we had prior knowledge of L, then this condition would be trivial to check since nonlinear CG already computes ∇f^k on every iteration. Without prior knowledge of L, we can still in principle check this condition by carrying out a Wolfe line-search [90] in the direction $-\nabla f^k$ on every iteration. It is known that, up to a constant factor depending on the parameters used in the line-search, the reduction guaranteed is at least as good as $\|\nabla f^k\|/(2L)$. However, it is quite expensive to carry out a line search in the steepest descent direction on every iteration in addition to the line search already required for the CG direction. Our computational experiments (not reported here) indicate that it is not necessary because there is little improvement in the behavior of the method. Hereafter, we will simply assume that this condition holds.

The subspace optimization needs a termination test. For this purpose, we again rely on inequalities (2.4) and (2.5). Although the lemma requires these inequalities to be checked only at an iteration at the end of a block, it is also possible to check them on intervening iterations. We use these inequalities to terminate the search for a subspace solution.

We can now present our detection and correction procedure in its entirety. In this procedure, S is a subset of $\{\varphi_l, \ldots, \varphi_{\max}\}$ and denotes the set of values of p for which correcting is currently active. Also let $m(j,p)$ denotes the largest multiple of 2^p less than or equal to the current iteration counter j. In the correction step, B is a matrix whose columns are the collection of the directions in the subspace optimization problem, i.e., ∇f^j, d^j, q_p^j for all $p \in S$, and $x^j - x^{m(j,p)}$ for all $p \in S$; and y^l denotes the variable of the subproblem, i.e., a vector whose entries are coefficients of columns of B. Recall that in CGSO, inequality (2.5) seems to hold for a modest value of ρ even without correcting for violations of (2.5). Therefore, in CGSO, we did not set a fixed value of ρ but simply tracked the behavior of the quotient $\frac{\|\sum_{i=r_p}^{j} \lambda^i \nabla f(x^i)\|}{\sqrt{\sum_{i=r_p}^{j} (\lambda^i)^2 \|\nabla f(x^i)\|^2}}$ in experiments to verify that it did not grow. In our implementation of the correction-detection procedure, however, this quotient grew large without correction steps. Therefore, we fixed $\rho = 1.2$ in the correction-detection procedure and took correction steps when either (2.4) or (2.5) was violated.

Algorithm 2.2. CGwDC[1]

SUBROUTINE: `verify_step`(x^j, s^j)
    for *each* $p \in S$
        if (2.4) *or* (2.5) *fail with* $x^{m(j,p)}$ *substituted for* x^0 *and* $x^j + s^j$ *substituted for* x^{m-1}
            return False*;*
    return True*;*
MAIN PROCEDURE: `CGwDC`(x^0)
$S = \emptyset$
for $j = 1, 2, \ldots$
    $d^j = -\nabla f^j + \beta^j d^{j-1}$*;*
        Remark: d^j *is the ordinary nonlinear CG direction.*
        Remark: take $\beta^j = 0$ *if either* $j = 1$ *or* d^{j-1} *was discarded.*
    $\alpha^j =$ `Wolfe_line_search`(f, x^j, d^j);
    stepfound = False*;*
    if `verify_step`$(x^j, \alpha^j d^j)$

[1] CG armed with Detection and Correction Procedure

stepfound = True*;*
$s^j = \alpha^j d^j$*;*
else
discard d^j*;*
if not *stepfound*
Apply Newton's method to solve:
$x^{j+1} = \arg\min_{x \in E^j} f(x)$
where $E^j = x^j + Span\left\{\nabla f(x^j), d^j, \cup_{p\in S} q_p^j, \cup_{p\in S} x^j - x^{m(j,p)}\right\}$
Terminate if either `verify_step`(x^j, By^l) *or iteration-max is attained.*
if `verify_step`(x^j, By^l)
stepfound = True*;*
$s^j = By^l$;
if not *stepfound*
Apply the ellipsoid method to solve:
$x^{j+1} = \arg\min_{x \in E^j} f(x)$
where $E^j = x^j + Span\left\{\nabla f(x^j), d^j, \cup_{p\in S} q_p^j, \cup_{p\in S} x^j - x^{m(j,p)}\right\}$
Terminate when `verify_step`(x^j, By^l)*.*
stepfound = True*;*
$s^j = By^l$;
$x^{j+1} = x^j + s^j$;
for $p = p_l, \ldots, \lceil \log_2 j \rceil$
if $j + 1 = k_p 2^p$ *for some integer* k_p
if $p \in S$
$S = S \setminus \{p\}$
elseif not `verify_step`(x^{j-1}, s^{j-1})*;*
$S = S \cup \{p\}$

As mentioned earlier, because the above algorithm enforces (2.4) and (2.5) for every value of 2^p (i.e. taking the correction step at every other iteration in the worst case), we get the optimal convergence bound using Lemma 2.1 and Theorem 2.3.

2.3 Computational Experiment

2.3.1 Remarks on Computational Divided Differences

In a line-search for conjugate gradient, it is necessary to accurately evaluate quantities of the form $f(x + \alpha d) - f(x)$. A similar quantity arises in the ratio test for the trust-region method [90]. It is well known to implementors of such methods that these divided differences are problematic near the root because of cancellation error between the two terms. A brief discussion of this issue appears in Hager and Zhang [53]. Failure to compute these quantities accurately can lead either to premature termination of an algorithm or to infinite loops.

A solution to this problem, perhaps not as widely known in the optimization literature as it should be, is "computational divided differences" by Rall and Reps [95]. The idea is to transform a source-code program for computing f into another source-code program for accurately computing divided differences of f. The technique is somewhat reminiscent of automatic differentiation.

To give a concrete example, consider the log-barrier function that will be used in Section 2.3.2 as a test case, which is written as $f(x) = \sum_{i=1}^{m} \log(a_i^T x - b_i)$, where each a_i is given vector in $\mathbb{R}^n$ and each b_i is a given scalar. This function is defined on the open polyhedron given by $Ax > b$ and strongly convex on this polyhedron provided that the polyhedron is bounded. Suppose x is our current iterate and δ is a small step. We have the following derivation:

$$\begin{aligned} f(x+\delta) - f(x) &= \sum_{i=1}^{m} \log(a_i^T(x+\delta) - b_i) - \sum_{i=1}^{m} \log(a_i^T x - b_i) \\ &= \sum_{i=1}^{m} \log\left(1 + \frac{a_i^T \delta}{a_i^T x - b_i}\right). \end{aligned}$$

Thus, to evaluate this divided difference accurately, one needs a function to compute $\log(1 + a)$ accurately when $|a|$ is small. One can develop a method for this computation using calculus. That effort is, however, unnecessary since Matlab and C++ both contain the built-in library function `log1p` for exactly this purpose.

We have used computational divided differences for all of our testing in Section 2.3.3. (We hand-coded the accurate divided differences rather than using a source-to-source translation tool; we are not sure if such a tool exists.) In addition to the line-search, our method uses computational divided differences for the evaluation of the left-hand side of (2.4).

Without them, all the methods would be less reliable and the test results harder to interpret. Indeed, we believe that computational divided differences deserve to be used much more widely in general nonlinear optimization than they are currently. See [104] for some additional comments on their use in optimization.

Because of our reliance on this technique, it is not possible to directly compare our results in the next section to well known packages like CG-DESCENT, or on benchmark test cases like CUTEr which do not use computational divided differences. For this reason, we compare only our own implementations against each other. A stated goal of CG in general (as well as of our variants of CG) is to use as little information about the function as possible. It may thus seem contradictory that we are assuming the availability of a source-code program for the objective function in order to obtain computational divided differences. But one should note that obtaining divided differences from f is in principle an automatic procedure given the source code, whereas obtaining parameters l and L would be quite difficult. Indeed, it is known that simply checking whether a 4th degree multivariate polynomial is convex (presumably an easier problem than obtaining l and L) is NP-hard [2].

2.3.2 Implementation of CGSO

Obviously the most important part in CGSO is solving the subspace optimization problem in each iteration. As mentioned, we used Newton's method for this task unless it fails to rapidly converge to the optimum in our implementation. In the case of failure of Newton's algorithm, the ellipsoid method carries out the task of solving the optimization problem. We would like to point out again that the assumption of strong convexity or the Lipschitz continuity of the gradient are not sufficient for convergence of Newton's method, but it succeeds in many cases nonetheless. In our experiment we assigned the upper bound of 15 to the number of iterations Newton's method may take. If it does not converge in 15 iterations, the algorithm switches to ellipsoid method.

Recall that x^{j+1} is the minimizer of the function over the space of vectors $x = x^j + \alpha \nabla f^j + \beta d^j + Qa + Rb$, where $Q \in \mathbb{R}^{n \times |S|}$ is the matrix formed by columns q_p^j for all $p \in S$; R is the matrix of the same dimension with columns $x^j - x^{r_p}$ for all $p \in S$; $\alpha, \beta \in \mathbb{R}$, and $a, b \in \mathbb{R}^{|S|}$ are coefficients that we want to find. Let y denote the variable of the subspace optimization problem, i.e., $y = [\alpha, \beta, a^t, b^t]^t$; in addition let $B = [\nabla f^j,\ d^j,\ Q,\ R]$ and $\omega = 2 + 2\,|S|$. The formal statement of the subspace optimization problem is

$$\min_{y \in \mathbb{R}^\omega} f\left(x^j + By\right). \tag{2.20}$$

As mentioned, we solve problem (2.20) with Newton's method. Letting $\tilde{f}(y) = f(x^j + By)$ and using the chain rule we get the following formulas for the gradient and Hessian of each Newton's iteration,

$$\nabla \tilde{f}(y) = B^t \nabla f(x), \tag{2.21}$$

$$\nabla^2 \tilde{f}(y) = B^t \nabla^2 f(x) B. \tag{2.22}$$

Notice that some second order information of the function comes into play in equation (2.22). For the subspace optimization subproblem, we solve a problem of size ω which is typically two. From (2.22) it is clear that we require $\nabla^2 fB$, i.e., Hessian times few (normally two) vectors, and we are not forming the Hessian explicitly. Since computing a Hessian-vector multiplication can often be a cheaper task as opposed to finding the Hessian, Newton's method is effective for solving the subproblem. The Newton equations are typically very low dimensional (say, dimension 2 or 3), so our implementation solves them using MATLAB's backslash operator, which is implemented via dense Cholesky factorization.

As we discussed in Section 2.3.1, a difficulty we need to overcome especially when we are trying to achieve high accuracy, is round-off error. In particular $f(x^j) - f(x^{j+1})$, which is required for computing λ^j, gets more and more inaccurate as the iterates approach the optimum. In Section 2.3.1 we explained how one may overcome this problem by computational divided differences. The experiments in this section, however, were performed before we knew about computational divided differences. Therefore, we took advantage of the second order Taylor series expansion to solve the problem here. We have a subroutine that analyses the absolute error of $f(x^j) - f(x^{j+1})$ computed directly and through Taylor series, i.e. $f(x^j) - f(x^{j+1}) \approx -\big(\nabla f(x^j)^t (x^{j+1} - x^j) + \frac{1}{2}(x^{j+1} - x^j)^t \nabla^2 f(x^j)(x^{j+1} - x^j)\big)$; the one with smaller error is accepted. In some cases the error analysis is not easy, hence a heuristic is used to choose the preferred formula. Computing the difference of two objective values appeared in inequality (2.4) as well, in $f(x^{m-1}) - f(x^0)$; the same subroutine is used to compute this term.

Test Cases:

We have tested our algorithm on the following classes of problem:

1. $f_1(x) = -\sum_{i=1}^m \log\left(a_i^t x - b_i\right)$
2. $f_2(x) = c^t x - \log\left(\det\left(C - \mathrm{Diag}(x)\right)\right)$
3. $f_3(x) = \sum_{i=1}^m \left(a_i^t x - b_i\right)^d$ where d is a given even integer.

Functions f_1 and f_2 are log-barrier functions, and f_3 is an approximation to the infinity norm of the vector $Ax - b$. Note that we need to restrict the degree, d, to even numbers to enforce the convexity of f_3. Suppose $./$ and $.*$ denote the component-wise division and multiplication, respectively; similarly $\cdot$ before the exponent denote the component-wise power. Using this notation, the gradient and the Hessian (times a vector) of the above functions are

1. $f_1(x) = -\sum_{i=1}^{m} \log\left(a_i^t x - b_i\right)$
   $\nabla f_1(x) = A^t\left(\mathbf{e}./s\right)$, where $s = Ax - b$
   $\nabla^2 f_1(x)p = A^t A p./s^{\cdot 2}$

2. $f_2(x) = c^t x - \log\left(\det\left(C - \mathrm{Diag}(x)\right)\right)$
   $\nabla f_2(x) = c + \mathrm{diag}(Z^{-1})$, where $Z = C - \mathrm{Diag}(x)$
   $\nabla^2 f_2(x)p = \mathrm{diag}\left(Z^{-1}\mathrm{Diag}(p)Z^{-1}\right) = (Z^{-1}.*Z^{-1})p$

3. $f_3(x) = \sum_{i=1}^{m}\left(a_i^t x - b_i\right)^d$ where d is a given even integer.
   $\nabla f_3(x) = dA^t s^{\cdot d-1}$, where $s = Ax - b$
   $\nabla^2 f_3(x)p = d(d-1)A^t\mathrm{Diag}(s^{\cdot(d-2)})Ap = d(d-1)A^t\left(s^{\cdot(d-2)}.*(Ap)\right)$

Before presenting the results, we need to comment on how the running time was measured. We measure time in "units", where we count as one unit an evaluation of a function or gradient or function/gradient pair (at the same point). In the line-search procedure, gradients are evaluated several times, so each outer iteration costs several units. We count the evaluation of $\nabla^2 f(x)p$, needed for Newton's method, as two units. Here, x and p are arbitrary vectors. From the presented formulas, it is clear that the computation of $\nabla^2 fp$ in our test cases costs twice as much as the gradient, namely two matrix-vector multiplication as opposed to one for the gradient. The main theorem of backward-mode automatic differentiation states that the evaluation of $\nabla^2 f(x)p$ should never cost more than 5 units [90, Chapter 8]. (None of our examples reach this upper bound of 5.) Note that each iteration in Newton's method needs two $\nabla^2 fp$ typically, so its cost counts as five units. Finally, one iteration of the ellipsoid method also counts as one unit since it involves one gradient evaluation.

Parameter ρ:

We already discussed that ρ is a parameter of CGSO required for checking inequality (2.5). We desire ρ to be close to one. In our experiment, we do not check inequality (2.5); instead

we gather the values of

$$\bar{\rho} := \frac{\left\|\sum_{i=r_p}^{j} \lambda^i \nabla f^i\right\|}{\sqrt{\sum_{i=r_p}^{j} (\lambda^i)^2 \|\nabla f^i\|^2}} \tag{2.23}$$

and study their trend. Our observation (to be presented shortly) suggests that a reasonably small bound for ρ should suffice, meaning that we may assign a relatively small value, say 5, to ρ for inequality (2.5) to always go through. In other words, we never require any correction regarding to (2.5), since it never fails.

Numerical Results:

We take φ_l, the lower bound on the range of p for checking inequality (2.4), to be 4. In fact, there is little harm in omitting the check on the conditions for very small values of p since the lemma will still guarantee convergence, albeit slightly more slowly, if we catch those corrections for larger values.

All the codes are written in MATLAB. The stopping criterion we used for both CGSO and its subproblem are $\|\nabla f(x^j)\| \leq \epsilon$ and $\left\|\nabla \tilde{f}(x^j)\right\| \leq \epsilon_N$, respectively. In our implementation, ϵ_N at iteration j is $\frac{\|B^t \nabla f^j\|}{100}$; however, if the subproblem is solved by Newton's method, the obtained solution is usually more accurate due to the fast local convergence of the Newton's method. Notice that for f_1 and f_2 we have some hidden constraints, namely $Ax - b > 0$ in f_1 and $C - \text{Diag}(x) \succ 0$ in f_2. Since in CGSO we use direct Newton's method without any line search, the algorithm switches to ellipsoid method if the iterates get infeasible. In addition, we have set the upper bound of 15 to the number of iteration Newton's method may take. After reaching 15 iterations in the Newton's method, the algorithm switches to ellipsoid method; this, however, never happened in our experiment.

Note that the termination criterion used in this experiment is not scale invariant. An ideal termination criterion would be a relative error on the residual of the objective value (or the solution), i.e.,

$$\frac{f(x^j) - f(x^*)}{f(x^*)} \leq \epsilon. \tag{2.24}$$

This, however, requires knowing the optimal solution. Other possible candidates for a scale

		HZ			CGSO				
m(=3n)	ds(A)	itr.	ls.	unit	itr.	New.	ellp.	unit	corr.
6000	1	404	2999	2999	261	402	394	2404	0
6000	0.5	357	2535	2535	213	323	422	2037	0

Table 2.1: Comparison of CGSO and HZ for linear log barrier function

invariant termination test might be

$$\|\nabla f(x^j)\| \leq \epsilon \|\nabla f(x^0)\|, \tag{2.25}$$

$$\frac{f(x^j) - f(x^{j-1})}{f(x^j)} \leq \epsilon. \tag{2.26}$$

Because of the hidden constraints for most of our test cases, $\|\nabla f(x^0)\|$ might be very large if x^0 is not sufficiently close to the optimizer. Therefore, termination criterion (2.25) can perform poorly. Criterion (2.26) may also result in early termination at a non-optimal point for CG methods due to loss of independence of the gradients. As we discussed earlier, when the independence of the gradients is lost in CG, two or more consecutive iterates may become almost identical. In fact, this phenomenon is the motivation for recovering the independence of the gradients or restarting in the CG techniques. We observed that the tolerance on the norm of the gradient is a better option for our experiment.

Hager and Zhang [53] compared their variant of CG with L-BFGS and PRP_+, and established the superior performance of their variant of CG. We compare our algorithm with their variant of CG, represented by HZ on a set of randomly generated instances. In the tables summarizing the results, "itr.", "ls.", "New." and "ellp." refer to the total number of (outer) iterations, line search iterations, Newton's iterations and ellipsoid iterations, respectively. The column "unit" represents the total cost (according to the presented definition of unit) for each algorithm. m and n are for the dimension of the problem. "ds(A)" denotes the sparsity of matrix A, i.e., ds=0.5 means that half of the entries of A are zeros. For the SDP log barrier function (log of determinant) we assume $c = \mu \mathbf{e}$ for some constant μ. The value of this parameter in Table 2.2 is stated by μ_f. The larger gets the value for μ, the more ill-conditioned the problem is. Finally, the number of times that inequality (2.4) has failed (i.e., the number of times the correction step was required) is listed under column "corr."

The number of iterations CGSO takes is less than HZ in all instances, and the number of iterations for solving subproblems is significantly less than the number of line search

		HZ			CGSO				
n	μ_f	itr.	ls.	unit	itr.	New.	ellp.	unit	corr.
500	100	3059	92682	92682	1080	1884	6195	15615	0
1000	10	928	14596	14596	419	600	1641	4641	0

Table 2.2: Comparison of CGSO and HZ for log of determinant function

				HZ			CGSO				
n	m	d	ds(A)	itr.	ls.	unit	itr.	New.	ellp.	unit	corr.
3000	1500	6	0.5	225	6220	6220	148	545	0	2725	0
5000	2500	4	0.5	203	7333	7333	136	410	0	2050	0
50	50	4	1	3442	53474	53474	658	1055	0	5723	7

Table 2.3: Comparison of CGSO and HZ for d-norm function

iterations. The "unit" cost of both algorithms indicates that CGSO outperforms HZ, especially on the ill-conditioned instances like the first SDP log barrier and the third d-norm function reported here. The correction step was almost never required. The only instance that correction step was taken is in the third row of Table 2.3. Actually the purpose of this instance is to show that correction step might be required for some problems. In this instance the condition number of A is in $O(10^3)$. The correction step is taken throughout 7 blocks corresponding to $\varphi_l = 4$. This, however, is consistent with the theory because the larger the condition number is, the larger $\frac{L}{l}$ is; therefore 2^{φ_l} may not be a good approximation of $\left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil$.

Earlier, we mentioned that we collect $\bar{\rho}$ and study their values. Table 2.4 summarizes the maximum value of $\bar{\rho}$ reached in each instance over all values of p. Figure 2.1 depicts $\bar{\rho}$ for the instances corresponding to the first rows of Tables 2.1 and 2.2. We chose these instances because they have higher ρ_{max} and iteration count in each category. As we expected the maximum value of $\bar{\rho}$ is fairly small (indeed much smaller than m), so talking $\rho = 5$ makes inequality (2.5) to hold for all our instances. Notice that the plotted values imply that $\bar{\rho}$ reaches its maximum for some $\bar{p}$ and then it decreases for larger values of p, so it does not grow with p. Furthermore, as we get closer to the optimum $\bar{\rho}$ gets closer to 1. This is a common pattern for all test problems we had. The fact that $\bar{\rho}$ decreases as we get closer to the optimum suggests an adaptive way to update this parameter instead of fixing it. In other words, we can assign an initial value to parameter ρ and if inequality (2.5) fails

	ρ_{max}
Linear Log Barrier	1.9848
	1.9803
Log of Determinant	3.2511
	2.144
d-norm	1.016
	1.0281
	1.9430

Table 2.4: Maximum value of quotient (2.23) ($\bar{\rho}$)

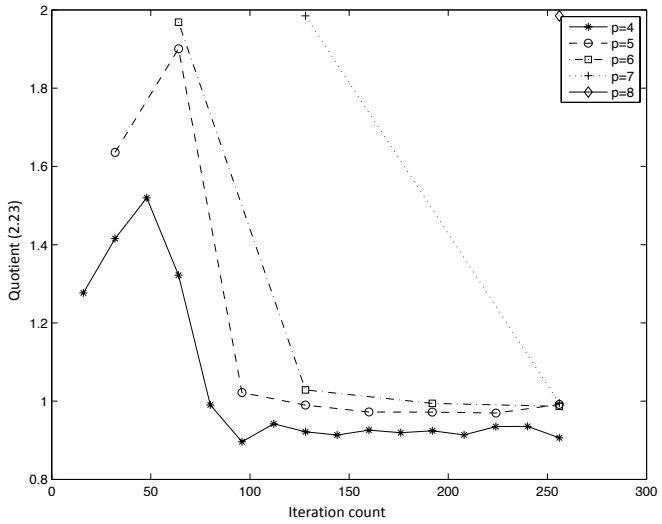
for more than a certain number of blocks, then we may increase ρ, and as the algorithm progresses towards the optimum we can decrease ρ.

2.3.3 Numerical Results for the Detection and Correction Procedure

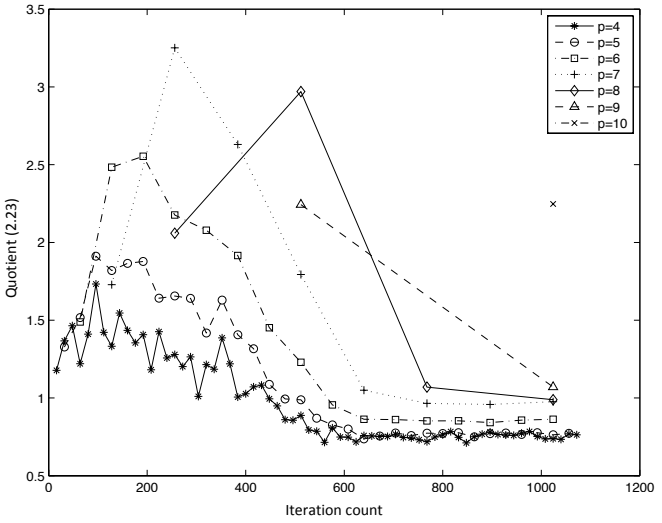
We have tested the correction method on four classes of problems, three convex and the fourth nonconvex. Our setup was as follows. We tried three different variants of conjugate gradient, namely FR, PR+, and HZ. Here, PR+ denotes the Polak-Ribière method in which the parameter β is replaced by 0 in the case that it becomes negative (thus forcing a restart), which is a recommended modification (see [90]).

Most of our test cases are small. This allowed us to perform more experiments in a reasonable amount of time. As mentioned earlier, the behavior of conjugate gradient is governed much more by conditioning of the problem than problem size. However, to illustrate that the method is also suitable for large problems, we have included two somewhat larger test cases.

The results of our experiments can be summarized as follows. For uncorrected methods, the HZ direction is usually the best, while the FR method is usually the worst, and sometimes FR is much worse. For corrected methods, all three directions perform about equally well. The corrected methods are typically slower than the uncorrected HZ method for well-conditioned problems. For ill-conditioned problems, however, the corrected method is sometimes much better than HZ (as well as the other two methods). Note that no forced restarts have been implemented. However, there are still restarts in some cases. As noted above, in our correction procedure, when a conjugate gradient search direction is discarded, the following step is, at least initially, the steepest descent direction. Also as noted above,



(a) First instance of linear log barrier



(b) First instance of log of determinant function

Figure 2.1: Quotient (2.23) ($\bar{\rho}$) with respect to iteration count

	Uncorrected			Corrected		
	HZ	FR	PR+	HZ	FR	PR+
$\text{cond}(A) = 10^5$	38,483	98,442	73,756	87,894	85,930	88,280
$\text{cond}(A) = 10^8$	*5,552,754	8,557,387	*27,669,107	2,407,560	2,181,492	2,517,924
cond unknown	149,543	66,373	400,112	115,698	110,230	86,200

Table 2.5: Number of units of computation for convex quadratic functions.

the PR+ method will sometimes restart automatically if it computes a negative β.

Similar to what was described for CGSO, we measure time in "units". Recall that one unit is an evaluation of a function or gradient or function/gradient pair, and the evaluation of $\nabla^2 f(x)p$, needed for Newton's method, counts as two units. Our line search is based on simple bisection and the Wolfe conditions [90, Inequality 3.6 and Algorithm 3.1]. An asterisk indicates a computation terminated due to an iteration limit.

We now present the results in more detail. The first test function is a simple quadratic, $f(x) = x^T Ax + b^T x$ for a positive definite matrix A. The first two rows are smaller instances with $n = 1000$, the last row is a larger problem with $n = 197,136$. Note that none of the methods reduce to linear CG in this case because we did not implement an exact line search. Therefore, there is no prior guarantee that independence of search directions is maintained. On the other hand, because the problem is quadratic, the Newton method on the subspace converges in a single iteration and the ellipsoid method is never used. In two cases we formed A by choosing 1000 geometrically spaced eigenvalues in a predetermined interval and then multiplying on the left and right by a random 1000×1000 orthogonal matrix. In this way, the condition number of A is determined exactly. In the third case we formed A as the assembled stiffness matrix of a finite-element discretization of Poisson's equation on the unit disk with a relatively uniform and well-behaved mesh. This problem has moderate ill-conditioning, but the matrix was too large to exactly measure its condition number. The results of these experiment are shown in Table 2.5.

The next class of experiments is with linear log-barrier functions, that is $f(x) = \mu \sum_{i=1}^{m} \log(a_i^T x - b_i) + c^T x$. The first three lines are smaller problems ($A \in R^{400 \times 100}$); the last line is a larger DIMACS graph problem ($A \in R^{91,756 \times 15,605}$). In the third line, the condition number of A was slightly worse. In these experiments we generated A randomly with known condition number for two smaller cases, and we took A to be the node-arc incidence matrix of an undirected graph (hence two copies of each edge, one for each direction) for a larger test case. This matrix A is relatively well conditioned. However, we can make the problem more ill-conditioned by decreasing μ (thus pushing the solution closer to the

	Uncorrected			Corrected		
	HZ	FR	PR+	HZ	FR	PR+
$\mu = .4$	292,012	1,496,650	963,968	461,036	394,382	420,808
$\mu = .1$	593,190	3,034,394	2,059,235	1,568,258	1,477,938	1,463,452
$\mu = .1$	*55,190,257	*58,728,472	*55,665,606	13,349,163	15,235,567	14,813,917
$\mu = 100$	1,298,292	*6,633,403	2,297,573	762,649	654,900	668,127

Table 2.6: Number of units of computation for log-barrier functions.

	Uncorrected			Corrected		
	HZ	FR	PR+	HZ	FR	PR+
$\lambda = 10^{-3}$ †	51,514	263,202	116,740	97,781	99,072	91,763
$\lambda = 10^{-4}$ ‡	986,314	5,049,449	3,397,063	810,887	926,389	879,384
$\lambda = 10^{-4}$ ‡	46,206,176	56,618,846	55,827,526	15,523,751	19,667,056	14,350,066

† $\text{cond}(A) = 10^5$
‡ $\text{cond}(A) = 10^6$

Table 2.7: Number of units of computation for regularized BPDN functions.

boundary of the feasible region). The graph in question came from a DIMACS challenge problem. The results are in Table 2.6.

The third test case consists of smoothed versions of the BPDN problem. The unsmoothed version of this problem has an objective function of the form $\|Ax - b\|^2 + \lambda\|x\|_1$, where A has fewer rows than columns. In the smoothed version we approximate the function $|x|$ by $(x^2 + \delta)^{1/2}$ which is convex (strongly convex on bounded intervals) and smooth. For each case, $A \in R^{100\times 400}$. For both rows, the regularization parameter δ is $5 \cdot 10^{-4}$. We did not try a large instance of this problem because typically A is taken to be a dense matrix, so a large problem would require too much computation time. The results are shown in Table 2.7.

The final test case is the nonconvex distance geometry problem. In this problem, there is a sequence of n points $(x_1, \ldots, x_n)$ each in R^d whose coordinates are mostly unknown. However, many pairs of interpoint distances are given. The problem is to find the positions of the points. This can be posed as a nonlinear least squares problem of minimizing $\sum_{(i,j)\in E} \left(d_{i,j}^2 - \|x_i - x_j\|^2\right)^2$ where E is a list of the pairs (i, j) whose distances are known,

	Uncorrected			Corrected		
	HZ	FR	PR+	HZ	FR	PR+
stretch=1	29,829	61,148	56,453	42,311	38,501	43,123
stretch=5	328,436	672,881	974,868	87,416	93,771	102,012

Table 2.8: Number of units of computation for distance geometry functions.

d_{ij} is the known distance, and the x_i's are unknown, except for a few, called 'anchors', which make the problem well posed (i.e. the problem has isolated unique solution provided that the graph is sufficiently well connected and that there are enough anchors [58, 4, 64]).

Because of the nonconvexity, it is possible for different algorithms to converge to different local optimizers; such a result would naturally make the running time estimates difficult to interpret. In order to prevent this inconsistency, the data was constructed so that there is an exact solution (i.e., the nonlinear least squares instance has a solution with zero residual), and then all the methods were initialized at a point close to that solution. With this device, we were able to ensure convergence to the same solution. The coordinates of the known solution were taken as random points in the plane, and a random subset of possible edges was used in the objective function. We do not attempt to experiment here with the global convergence behaviour of the detection-correction procedure for nonconvex objective functions, but this is a topic of possible future study.

A second issue with nonconvexity is that the ellipsoid method is no longer valid for solving the subspace problem. Therefore, our two methods for solving the subspace problem in this case were Newton, and, if it fails, a trust-region method [90]. Note that our Newton method used in this test case and all the others reported in this chapter was not globalized (i.e., no line search or trust region was employed), so the trust-region method can be regarded as recomputing a Newton step with a globalization. We define failure of pure Newton method to be any of the following conditions: excessive number of iterations, badly conditioned Jacobian, or a computed step with an excessive norm. It turned out that the trust-region method was never invoked, most likely because we started sufficiently close to the root. We can control the conditioning of the problem by stretching the random data points along one axis (x or y). The results of a well-conditioned and ill-conditioned problem are in Table 2.8. In each case the number of unknowns was 400 while the number of distances was 600.

2.4 CGSO for Constrained Problems

In Appendix A, we extend our results for CGSO to minimization of a strongly convex function over a polyhedron using the notion of projected gradient, Definition 1.2. The number of iterations required to decrease the residual of the function by a factor of two is $\lceil 8\rho\sqrt{\frac{L}{l}}\rceil + |\kappa^{\mathcal{A}}|$ in the constrained variant of CGSO (as opposed to $\lceil 8\rho\sqrt{\frac{L}{l}}\rceil$ in the unconstrained case), where $|\kappa^{\mathcal{A}}|$ denotes the number of iterations between each two restarts for which the active set has changed.

Currently we have no bound on the number of active set changes. In general, finding this bound is not possible; but the question of whether this bound exists for a special class of problems, say box-constrained QP, has remained unanswered. Moreover, a notion similar to the correction step presented in Section 2.1.3 is required for the constrained variant of CGSO as well, since it also relies on the two safeguard inequalities. How to perform a correction step in the constrained case is unclear at this point and needs further study.

In Appendix A we applied CGSO to the following reformulation of BPDN problem

$$\begin{array}{ll} \min & \frac{1}{2}\|Ax-b\|^2 + \lambda \mathbf{e}^t y, \\ s.t. & y \geq x, \\ & y \geq -x. \end{array} \tag{2.27}$$

Note that this may not be a suitable test case as the objective function is no longer strongly convex, because $Ax-b$ is underdetermined. But computing the projected gradient for this problem can be done quite efficiently.

As we expect, the solution of BPDN is typically very sparse. The sparsity of the solution means that many constraints are active at the optimal solution. Our observation in the experiment presented in Appendix A indicates that many iterations of CGSO correspond to active set changes. This influences the convergence rate of the algorithm considerably since it depends on $|\kappa^{\mathcal{A}}|$. For the BPDN problem, we present a quasi-Newton method in Chapter 4 that is more effective than CGSO in practice.

Chapter 3

On Nesterov's Technique

We presented a simple form of Nesterov's algorithm in Section 1.2, and we discussed some connections between this algorithm and CG. Similar to NY-CG, Nesterov's algorithm does not reduce to linear CG when applied to a quadratic function; in fact, our numerical experiment suggests that it can be much slower than linear CG. Unlike NY-CG, Nesterov's algorithm has not been motivated by the essence of CG. It is rather built on the idea of an estimate sequence proposed by Nesterov [85]. In this chapter we first give an overview of Nesterov's technique; then we explain how we can partially incorporate CG in Nesterov's scheme.

3.1 Nesterov's Optimal Method

A major difference between Nesterov's method and CG is that the former generates two sequences of iterates. This idea, albeit originated by Nesterov, has been adopted by other researchers as well, see [9, 102, 81] and references therein. Nesterov motivates his algorithm in [85, Section 2.2]. The idea is to generate a sequence of functions $\{\phi^k\}$ with certain properties, then update the sequence of iterates using the minimizer of ϕ^k and the previous iterate.

Definition 3.1. *[85, Definition 2.2.1] A pair of sequences $\{\phi^k(x)\}$ and $\{\lambda^k\}$, $\lambda^k \geq 0$ is called an estimate sequence of the function $f(x)$ if $\lambda^k \to 0$ and for any $x \in \mathbb{R}^n$ and all $k \geq 0$ we have*

$$\phi^k(x) \leq (1 - \lambda^k) f(x) + \lambda^k \phi^0(x).$$

The question of how to create an estimate sequence is nontrivial. Nesterov, however, proposed an iterative mechanism for generating an estimate sequence for any $f \in \mathbf{C}_{l,L}$.

Lemma 3.1. *[85, Lemma 2.2.2] Suppose $\phi^0(x)$ is an arbitrary function, and $\{y^k\}$ is an arbitrary sequence in $\mathbb{R}^n$. Let $\lambda^0 = 1$ and $\{\alpha^k\}$ be a sequence such that $\alpha^k \in (0,1)$ and $\sum_{k=0}^{\infty} \alpha^k = \infty$. Then the pair of $\{\phi^k(x)\}$ and $\{\lambda^k\}$ defined as*

$$\lambda_{k+1} = (1-\alpha^k)\lambda^k, \tag{3.1}$$

$$\phi^{k+1} = (1-\alpha^k)\phi^k(x) + \alpha^k \left[f(y^k) + \langle \nabla f(y^k), x - y^k \rangle + \frac{l}{2}\|x - y^k\|^2 \right], \tag{3.2}$$

is an estimate sequence.

As mentioned before, the minimizer of ϕ^k is required for the algorithm, so we would like it to be simple enough so that its minimizer can be computed efficiently. The following lemma handles this task.

Lemma 3.2. *[85, Lemma 2.2.3] Suppose $\phi^0 = \bar{\phi}^0 + \frac{\gamma^0}{2}\|x - v^0\|^2$. The sequence generated in (3.1) and (3.2) forms*

$$\phi^k(x) = \bar{\phi}^k + \frac{\gamma^k}{2}\|x - v^k\|^2, \tag{3.3}$$

where

$$\gamma^{k+1} = (1-\alpha^k)\gamma^k + \alpha^k l, \tag{3.4}$$

$$v^{k+1} = \frac{1}{\gamma^{k+1}} \left[(1-\alpha^k)\gamma^k v^k + l\alpha^k - \alpha^k \nabla f(y^k) \right], \tag{3.5}$$

$$\begin{aligned} \bar{\phi}^{k+1} = {} & (1-\alpha^k)\bar{\phi}^k + \alpha^k f(y^k) - \frac{(\alpha^k)^2}{2\gamma^{k+1}}\|\nabla f(y^k)\|^2 \\ & + \frac{\alpha^k(1-\alpha^k)\gamma^k}{\gamma^{k+1}} \left(\frac{l}{2}\|y^k - v^k\|^2 + \langle \nabla f(y^k), v^k - y^k \rangle \right). \end{aligned} \tag{3.6}$$

Using the above lemma, we may now present Nesterov's method.

Algorithm 3.1.
Let $x^0 \in \mathbb{R}^n$ be an arbitrary point, $\gamma^0 > 0$, and $v^0 = x^0$
for $k = 0, 1, \ldots$
Let $\gamma^{k+1} = (1-\alpha^k)\gamma^k + \alpha^k l$,
where α^k is the solution of the equation $L(\alpha^k)^2 = (1-\alpha^k)\gamma^k + \alpha^k l$.

Let $y^k = \frac{\alpha^k\gamma^k v^k + \gamma^{k+1}x^k}{\gamma^k + \alpha^k l}$,
compute $f(y^k)$ *and* $\nabla f(y^k)$.

Find x^{k+1} *such that* $f(x^{k+1}) \leq f(y^k) - \frac{1}{2L}\|\nabla f(y^k)\|^2$.

Set $v^{k+1} = \frac{1}{\gamma^{k+1}}\left[(1-\alpha^k)\gamma^k v^k + \alpha^k l y^k - \alpha^k \nabla f(y^k)\right]$.

Now that an efficient scheme for generating an estimate sequence is on hand, we can present the lemma required for the convergence of the algorithm.

Lemma 3.3. *[85, Lemma 2.2.1] If for an estimate sequence of* f, $(\lambda^k, \{\phi^k\})$, *and a sequence of iterates,* $\{x^k\}$, *we have*

$$f(x^k) \leq \bar{\phi}^k = \min_x \phi^k(x),$$

then

$$f(x^k) - f(x^*) \leq \lambda^k \left(\phi^0(x^*) - f(x^*)\right).$$

The above lemma indicates that the convergence of the presented algorithm is directly related to the rate of convergence of $\{\lambda^k\}$. Nesterov has constructed the updating rules for his algorithm in a smart fashion so that the optimal convergence is attained. For more details on the convergence, one may refer to [85, Section 2.2]. Note that other than the sequence $\{v^k\}$ which is the optimizer of ϕ^k, we are free in choosing $\{y^k\}$ and $\{x^k\}$ provided that

$$\bar{\phi}^k = \min \phi^k(x) \geq f(x^k), \tag{3.7}$$

$$f(x^k) \leq f(y^k) - \frac{1}{2L}\|\nabla f(y^k)\|^2, \tag{3.8}$$

are satisfied. The updating formulas for y^k and x^k were actually suggested such that the satisfaction of (3.7) and (3.8), respectively is guaranteed. As mentioned earlier, Nesterov's method can be much slower than CG in practice. Since the only restriction on the sequences x^k and y^k are inequalities (3.7) and (3.8) for achieving the optimal complexity bound, we may replace the iterates generated by CG for either x^k or y^k in Nesterov's scheme hoping to get a hybrid method that is more efficient than Nesterov's method in practice while it maintains the desired optimal complexity bound. This is the topic we cover in the next two sections.

3.2 Substituting CG for x^k

The investigation in this chapter concerns methods for combining CG and Nesterov's method. The purpose of combining them is to develop a hybrid method with the best properties of both methods. In particular, the combined method should reduce to linear conjugate gradient in the case that the objective is function is quadratic and an exact line search is used, but on the other hand Nesterov's complexity bound will apply no matter what input function in $\mathbf{C}_{l,L}$ is given.

As the title of this section implies, we can combine CG and Nesterov's method by running these two algorithms in parallel; checking inequality (3.8); if the inequality holds then accept the x^k_{CG}, else perform a line search as Nesterov's method suggests and restart the CG algorithm. A more rigorous presentation of this hybrid method is as follows.

Algorithm 3.2.

Let $x^0 \in \mathbb{R}^n$ be an arbitrary point, $\gamma^0 > 0$, and $v^0 = x^0$.

SUBROUTINE (any variant of CG): $(x^+_{CG}, p^+_{CG}) = CG(x_{CG}, p_{CG})$

Compute β using an updating rule in a variant of CG

$p^+_{CG} = -\nabla f(x_{CG}) + \beta p_{CG}$

$x^+_{CG} = x_{CG} + \alpha p^+_{CG}$, where α is a step size found through a line search.

return

for $k = 0, 1, \ldots$

Let $\gamma^{k+1} = (1-\alpha^k)\gamma^k + l\alpha^k$,

where α^k is obtained by solving $L(\alpha^k)^2 = (1-\alpha^k)\gamma^k + l\alpha^k$.

Let $y^k = \frac{\alpha^k\gamma^k v^k + \gamma^{k+1}x^k}{\gamma^k + l\alpha^k}$,

compute $f(y^k)$ and $\nabla f(y^k)$.

Let $(x^{k+1}_{CG}, p^{k+1}_{CG}) = CG(x^k_{CG}, p^k_{CG})$.

if $f(x^{k+1}_{CG}) \leq f(y^k) - \frac{1}{2L}\|\nabla f(y^k)\|^2$

accept x^{k+1}_{CG} for x^{k+1},

else

find x^{k+1} by a line search on direction $\nabla f(y^k)$.

Restart CG: set $x^{k+1}_{CG} = x^{k+1}$, and $p^{k+1}_{CG} = \mathbf{0}$.

Set $v^{k+1} = \frac{1}{\gamma^{k+1}}\left[(1-\alpha^k)\gamma^k v^k + l\alpha^k y^k - \alpha^k \nabla f(y^k)\right]$.

The above algorithm carries all the properties of Nesterov's method, so it has the same

complexity bound. A drawback of Algorithm 3.2 is that two gradients, namely $\nabla f(x^k)$ and $\nabla f(y^k)$, must be computed per iteration; while in Nesterov's algorithm only $\nabla f(y^k)$ is required. Note that $\nabla f(x^k)$ is needed for finding p_{CG}^+ in the CG subroutine. This issue can possibly overshadow the improvement in the performance of the algorithm, since it increases the cost of each iteration to two gradient evaluation. This extra computational burden can be avoided by substituting CG iterates for $\{y^k\}$ rather than $\{x^k\}$. This is the topic of the next section. Before moving on to the next section we show that in fact for a sequence $\{x_{CG}^k\}$ generated by linear CG the inequality always holds.

Proposition 3.1. *When f is a strongly convex quadratic function, and linear CG is used in Algorithm 3.2, the condition $f(x_{CG}^{k+1}) \leq f(y^k) - \frac{1}{2L}\|\nabla f(y^k)\|^2$ always holds, i.e., the CG sequence always gets accepted.*

Proof. Suppose $f(x) = \frac{1}{2}x^tAx - b^tx$. Let us define sets $\mathcal{S}^i$ as follows:

$$\mathcal{S}^i = \left\{x : x = x^0 + \text{Span}\left\{r^0, Ar^0, \ldots, A^{i-1}r^0\right\}\right\},$$

where $r^0 = \nabla f(x^0)$.

Our proof is by induction. Notice that by the choice of parameters $y^0 = x^0$. By linear CG, x_{CG}^1 is the minimizer of f over the space $\mathcal{S}^1$; therefore

$$f(x_{CG}^1) \leq f(y^0 - \frac{1}{L}r^0) \leq f(y^0) - \frac{1}{2L}\|r^0\|^2.$$

Furthermore $v^1 \in \mathcal{S}^1$, and as a result $y^1 \in \mathcal{S}^1$.

Suppose $y^k, x^k, v^k \in \mathcal{S}^k$, then $\left(y^k - \frac{1}{L}\nabla f(y^k)\right) \in \mathcal{S}^{k+1}$. Since x_{CG}^{k+1} is the minimizer of f over $\mathcal{S}^{k+1}$, the inequality $f(x_{CG}^{k+1}) \leq f\left(y^k - \frac{1}{L}\nabla f(y^k)\right)$ holds.

□

3.3 Substituting CG for y^k

As mentioned earlier, computing $\nabla f(y^k)$ in Nesterov's method is unavoidable. It is required both in finding x^{k+1} and updating v^{k+1}. Using the CG sequence in the place of $\{y^k\}$ in Nesterov's method minimizes the computational cost per iteration as opposed to substituting the sequence $\{x^k\}$ with the CG sequence. The trade off is that inequality (3.7) must be checked in each iteration as y^k is no longer constructed in a specific way that guarantees it. On the positive side, this inequality can be effectively calculated as presented in the equation (3.6) in Lemma 3.2. If the inequality fails, we update y^k based on Nesterov's formula and restart CG. The algorithm in its complete form is stated below.

Algorithm 3.3.

Let $x^0 \in \mathbb{R}^n$ be an arbitrary point, $\gamma^0 > 0$, and $y^0, v^0 = x^0$.

SUBROUTINE (any variant of CG): $(x^+_{CG}, p^+_{CG}) = CG(x_{CG}, p_{CG})$ as in Algorithm 3.2

for $k = 0, 1, \ldots$

Let $\gamma^{k+1} = (1-\alpha^k)\gamma^k + l\alpha^k$,

where α^k is obtained by solving $L(\alpha^k)^2 = (1-\alpha^k)\gamma^k + l\alpha^k$

For $k \geq 1$: $(y^k_{CG}, p^k_{CG}) = CG(y^{k-1}_{CG}, p^{k-1}_{CG})$, for $k = 0$ $y^0_{CG} = x^0$

compute $f(y^k_{CG})$ and $\nabla f(y^k_{CG})$.

Compute $\bar{\phi}^{k+1}$ from equation (3.6).

if $\bar{\phi}^{k+1} \geq f(y^k_{CG}) - \frac{1}{2L}\|\nabla f(y^k_{CG})\|^2$

accept y^k_{CG} for y^k,

else

Let $y^k = \frac{\alpha^k\gamma^k v^k + \gamma^{k+1}x^k}{\gamma^k + l\alpha^k}$,

Restart CG: set $y^k_{CG} = y^k$, and $p^k_{CG} = \mathbf{0}$

Let $x^{k+1} = y^k - \frac{1}{L}\nabla f(y^k)$.

Set $v^{k+1} = \frac{1}{\gamma^{k+1}}\left[(1-\alpha^k)\gamma^k v^k + l\alpha^k y^k - \alpha^k \nabla f(y^k)\right]$.

Note that choosing x^{k+1} as $y^k - \frac{1}{L}\nabla f(y^k)$ fulfils inequality (3.8). Also note that computing α^k requires solving a quadratic equation. It can be checked that the equation always has one positive real root.

Proposition 3.2. *When f is a strongly convex quadratic function, and linear CG is used in Algorithm 3.3, the condition $\bar{\phi}^{k+1} \geq f(y^k_{CG}) - \frac{1}{2L}\|\nabla f(y^k_{CG})\|^2$ always holds, i.e. the CG sequence always gets accepted.*

Proof. Let $\mathcal{S}^i$ be as defined earlier in the proof of Proposition 3.1. Our proof is by induction. Note that $\bar{\phi}^0$ is by user's choice. Nesterov's suggestion is to take $f(x^0)$, so

$$\bar{\phi}^1 = (1-\alpha^0)f(x^0) + \alpha^0 f(x^0) - \frac{(\alpha^0)^2}{2\gamma^{k+1}}\|\nabla f(y^0)\|^2 \geq f(y^0) - \frac{1}{2L}\|\nabla f(y^0)\|^2,$$

by the fact that $\frac{(\alpha^0)^2}{2\gamma^{k+1}} = \frac{1}{2L}$ and $v^0 = y^0$.

Suppose $\bar{\phi}^k \geq f(x^k)$. Note that $f(x^k) \geq f(y^k)$ because $f(x^k) = f\left(y^{k-1} - \frac{1}{L}\nabla f(y^{k-1})\right)$ and $y^{k-1} - \frac{1}{L}\nabla f(y^{k-1}) \in \mathcal{S}^k$; while y^k is the minimizer of f over $\mathcal{S}^k$. Furthermore since

$\{y^k\}$ is generated by linear CG, $\langle \nabla f(y^k), v^k - y^k \rangle = 0$. This is true by Theorem 2.2; i.e., because both $v^k, y^k \in \mathcal{S}^k$, so $v^k - y^k \in \text{Span}\left\{\nabla f^0, \ldots, \nabla f^{k-1}\right\}$; and ∇f^k is orthogonal to $\nabla f^0, \ldots, \nabla f^{k-1}$. Now using closed form of $\bar{\phi}^{k+1}$ obtained in Lemma 3.2 we conclude that

$$\begin{aligned}
\bar{\phi}^{k+1} &= (1-\alpha^k)\bar{\phi}^k + \alpha^k f(y^k) - \frac{(\alpha^k)2}{2\gamma^{k+1}}\|\nabla f(y^k)\|^2 \\
&\quad + \frac{\alpha^k(1-\alpha^k)\gamma^k}{\gamma^{k+1}}\left(\frac{l}{2}\|y^k - v^k\|^2 + \langle \nabla f(y^k), v^k - y^k \rangle\right) \\
&\geq f(y^k) - \frac{1}{2L}\|\nabla f(y^k)\|^2 + \frac{\alpha^k(1-\alpha^k)\gamma^k}{\gamma^{k+1}}\left(\frac{l}{2}\|y^k - v^k\|^2\right) \\
&\geq f(y^k) - \frac{1}{2L}\|\nabla f(y^k)\|^2.
\end{aligned}$$

□

We have implemented the above techniques for two test cases. The first one is a strongly quadratic function and the second one is the smoothed BPDN problem explained in Section 2.3.3, i.e., $f(x) = \frac{1}{2}\|Ax-b\|^2 + \lambda \sum_{i=1}^n \sqrt{x_i^2 + \delta}$, where δ is a small regularization parameter equal to $5 \cdot 10^{-4}$. Tables 3.1 and 3.2 summarize the obtained results. For the quadratic function, employing CG with no exact line search was not significantly helpful, but for the smoothed compressive sensing problem it was very helpful. Table 3.1 also clarifies that Nesterov's method does not reduce to LCG, and can be quite slower.

$n = 400$, cond(A)=1e2, tol=1e-5						
	Nst.[1] cst.[4]	Nst. bs. ls.[5]	Nst.+CG$_x$[2] bs. ls.	Nst.+LCG$_x$ ex. ls.[6]	Nst.+CG$_y$[3] bs. ls.	Nst.+LCG$_y$ ex. ls.
it.[7]	147	145	113	74	113	74
ls. [8]	146	793	637	73	637	73
rj. [9]	-	-	0	0	1	0

1: Nesterov's method
2: Nesterov's method with CG for $\{x^k\}$
3: Nesterov's method with CG for $\{y^k\}$
4: Constant step size of $\frac{1}{L}$
5: Bisection line search
6: Exact line search
7: Iteration count
8: Line search iteration count
9: rejection count for sequence $\{x^k\}$ or $\{y^k\}$

Table 3.1: Nesterov's method and CG for the convex quadratic function

$A \in \mathbb{R}^{100\times400}$, cond($A$)=1e6, tol=1e-5				
	Nst. cst.	Nst. bs. ls.	Nst.+CG$_x$ bs. ls.	Nst.+CG$_y$ bs. ls.
it.	DNC ($> 1e6$)	DNC ($> 1e6$)	87021	85843
ls.	($> 1e6$)	($> 1e6$)	506417	604050
rj.	-	-	1561	16

Table 3.2: Nesterov's method and CG for the smoothed BPDN problem

Chapter 4

IMRO: A Practical Proximal Quasi-Newton Method

In Section 1.3 we briefly introduced a quasi-Newton proximal technique. In this chapter we present a practical variant of this method for solving the BPDN problem. Recall that the BPDN problem is

$$\min_x \quad F(x) := \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1. \tag{4.1}$$

Let us denote the quadratic part of $F(x)$ with $f(x)$. As mentioned earlier in Chapter 1, we may substitute $f(x)$ with a quadratic approximation model

$$m_H(x, x^k) = f(x^k) + \langle x - x^k, \nabla f(x^k)\rangle + \frac{1}{2}\left(x - x^k\right)^t H \left(x - x^k\right), \tag{4.2}$$

to find the next iterate. Ignoring a constant additive term, an equivalent form of (4.2) is

$$m_H(x, x^k) = \frac{1}{2}\|x - \left(x^k - H^{-1}\nabla f(x^k)\right)\|_H^2. \tag{4.3}$$

We enforce the condition $H \succ 0$ on model (4.2) to make sure $m_H(x, x^k) + \lambda\|x\|_1$ is a strongly convex function; therefore its minimizer exists and is unique. The next iterate, x^{k+1}, is then defined as

$$x^{k+1} = \arg\min_x \ \left(m_H(x, x^k) + \lambda\|x\|_1\right). \tag{4.4}$$

The above scheme is what we refer to as a proximal quasi-Newton methods. In the algorithms presented in this chapter we propose the following format for H:

$$H = \sigma\mathbf{I} - uu^t. \tag{4.5}$$

In fact the term "IMRO" stands for "Identity Minus Rank One" which is the proposed format for matrix H. We will see shortly that one of the advantage of IMRO is the efficiency in computing x^{k+1}. In [14], Becker and Fadili suggest a proximal quasi-Newton method in which H is an identity plus rank one matrix. The methodology that we develop for selecting σ and u presented in Sections 4.2.1 and 4.2.1 does not seem to extend to the case of identity plus rank one according to our analysis, but this question may need future investigation. Lee et al. [74] have also recently proposed proximal Newton-type method, and generalized the superlinear convergence of quasi-Newton methods to this class of algorithms under some assumptions on H.

In the next section we present how we can effectively compute x^{k+1} in IMRO. Our discussion is then followed by two different variants of IMRO, their properties and convergence results.

4.1 Computing x^{k+1} in IMRO

In this section we explain how we can attain the solution of (4.4), x^{k+1}, in linearithmic time, i.e. $O(n \log n)$.

By (4.4) and (4.3) we observe that at optimality

$$H\left(x - (x^k - H^{-1}\nabla f^k)\right) + \lambda\xi = 0, \tag{4.6}$$

holds, where $\xi \in \partial\|\cdot\|_1(x^{k+1})$. Let us denote $x^k - H^{-1}\nabla f^k$ by x^c. Recall that in IMRO $H = \sigma\mathbf{I} - uu^t$; thus H^{-1} might be computed in closed form:

$$(\sigma\mathbf{I} - uu^t)^{-1} = \frac{1}{\sigma}\mathbf{I} - \frac{1}{\sigma(\|u\|^2 - \sigma)}uu^t, \tag{4.7}$$

so we are able to calculate x^c easily. Condition (4.6) may now be restated as

$$(\sigma\mathbf{I} - uu^t)(x - x^c) + \lambda\,[\pm 1]_n = 0, \tag{4.8}$$

where $[\pm 1]_n$ is a vector of size n with ± 1 entries. Actually $\xi_i \in [-1, 1]$ for i's corresponding to zero entries of x. However as will be discussed soon, we require condition (4.8) to find the breakpoints that x_i changes sign, and we do not directly solve this equation; therefore we may ignore this notation abuse for now.

Let $\mu(\text{to be found}) = \frac{u^t(x-x^c)}{\sigma}$, then (4.8) reduces to

$$x - x^c - u\mu + \frac{\lambda}{\sigma}\,[\pm 1]^n = 0. \tag{4.9}$$

By (4.9), we conclude that i-th entry of x is either 0 or $x_i^c + u_i\mu - \frac{\lambda}{\sigma}$ (for $x_i > 0$) or $x_i^c + u_i\mu + \frac{\lambda}{\sigma}$ (for $x_i < 0$). Using this and sign of u_i, we may now find the proper interval for μ so that the mentioned equations for x_i holds true; in other words:

$$x_i > 0 \;\Rightarrow x_i^c + u_i\mu - \frac{\lambda}{\sigma} > 0 \quad \begin{array}{l} \rightarrow \mu > \frac{\frac{\lambda}{\sigma} - x_i^c}{u_i} \text{ if } u_i > 0, \\ \rightarrow \mu < \frac{\frac{\lambda}{\sigma} - x_i^c}{u_i} \text{ if } u_i < 0, \end{array} \tag{4.10}$$

$$x_i < 0 \Rightarrow x_i^c + u_i\mu + \frac{\lambda}{\sigma} < 0 \quad \begin{array}{l} \rightarrow \mu < \frac{-\frac{\lambda}{\sigma} - x_i^c}{u_i} \text{ if } u_i > 0, \\ \rightarrow \mu > \frac{-\frac{\lambda}{\sigma} - x_i^c}{u_i} \text{ if } u_i < 0. \end{array} \tag{4.11}$$

Note that by definition of μ, we have

$$u^t x - \mu\sigma = u^t x^c. \tag{4.12}$$

Searching over all the breakpoints mentioned in (4.10) and (4.11) (i.e. $\frac{\frac{\lambda}{\sigma} - x_i^c}{u_i}$ and $\frac{-\frac{\lambda}{\sigma} - x_i^c}{u_i}$), enables us to find the proper value of μ for which (4.12) holds. It remains to note that

$$u^t x = u^t(x^c \,[\pm]\, \frac{\lambda}{\sigma}) + u^t u\mu, \tag{4.13}$$

hence equation (4.12) has the equivalent form of

$$\text{(lhs)} \quad u^t(x^c \,[\pm]\, \frac{\lambda}{\sigma}) + (u^t u - \sigma)\mu = u^t x^c \quad \text{(rhs)}. \tag{4.14}$$

Since $\sigma\mathbf{I} - uu^t \succ 0$, the slope $u^t u - \sigma < 0$. To find μ, we sort all the breakpoints (a vector of size $2n$); we start with an initial value of μ small enough such that lhs $>$ rhs; we then increment the value of μ over the sorted breakpoints until we reach the desired interval $[\mu_l, \mu_u]$ such that $\text{lhs}_{\mu_l} > \text{rhs}$ and $\text{lhs}_{\mu_u} < \text{rhs}$, or the value of μ^* for which $\text{lhs}_{\mu^*} = \text{rhs}_{\mu^*}$. In the case that we reach the interval, a simple interpolation solves (4.14). Note that we may effectively update the lhs when reaching a breakpoint, since only one of x_i's changes sign for each breakpoint. The following chart visualizes how the search process is actually carried out:

$$u_i > 0: \quad \underbrace{- - - - - -}_{x_i<0} \Big| \frac{-\frac{\lambda}{\sigma} - x_i^c}{u_i} \Big| \underbrace{- - - - - -}_{x_i=0} \Big| \frac{\frac{\lambda}{\sigma} - x_i^c}{u_i} \Big| \underbrace{- - - - - -}_{x_i>0}$$

$$u_i < 0: \quad \underbrace{------}_{x_i>0} \Big| \frac{\frac{\lambda}{\sigma} - x_i^c}{u_i} \Big| \underbrace{------}_{x_i=0} \Big| \frac{-\frac{\lambda}{\sigma} - x_i^c}{u_i} \Big| \underbrace{------}_{x_i<0}$$

The algorithm below summarizes all we said above for finding x^{k+1}. The presented pseudocode is in MATLAB notation. "slp" in the following algorithm stands for the slope of lhs in (4.14) of the current piece (i.e., the derivative with respect to μ).

Algorithm 4.1.

Input: σ, u, x^c, *and* λ

`slp-Update Subroutine`*:*

Let $i = |\bar{a}(j,2)|$

if $\bar{a}(j,2) < 0$

  if $u_i < 0$

    $slp = slp + u_i^2$

  else

    $slp = slp - u_i^2$

else

  if $u_i < 0$

    $slp = slp - u_i^2$

  else

    $slp = slp + u_i^2$

`Main Procedure`

Let $\mathcal{I} = \{i : u_i \neq 0\}$

Form $a \in \mathbb{R}^{2|\mathcal{I}|\times 2}$ *such that* $a(i,:) = [\frac{\frac{\lambda}{\sigma} - x_i^c}{u_i}, +i]$ *and* $a(|\mathcal{I}|+i,:) = [\frac{-\frac{\lambda}{\sigma} - x_i^c}{u_i}, -i]$

Let $\bar{a} :=$ *sorted "a" on first column*

Let $rhs := u^t x^c$

Choose $\mu < \bar{a}(1,1)$ *such that* $lhs := u^t x^\mu - \mu\sigma > rhs$,

  where x^μ *is derived by* (4.9)

  $slp = u_{\mathcal{I}^0}^t u_{\mathcal{I}^0} - \sigma$, *where* $\mathcal{I}^0 = \{i : x_i^\mu \neq 0\}$

for $j = 1, 2, \ldots 2|\mathcal{I}|$

  Let $\mu^+ = \bar{a}(j,1)$

  $lhs^+ = lhs + slp(\mu^+ - \mu)$

  Update slp using `slp-Update Subroutine` $(\bar{a}(j,:), slp)$

  if $lhs^+ \leq rhs$

    $\mu^* = \frac{(rhs - lhs^+)\mu}{lhs - lhs^+} + \frac{(lhs - rhs)\mu^+}{lhs - lhs^+}$,

    Find x_μ *by* (4.9)

$$\begin{array}{c} return \\ \mu = \mu^+ \ and \ lhs = lhs^+ \end{array}$$

The computation of x^{k+1} can actually be done in linear time, i.e., $O(n)$ (rather than $O(n \log n)$). The linear-time algorithm for finding μ is based on the fact that there is an algorithm to find the median of an unsorted array of size n in $O(n)$. So after computing the $2n$ breakpoints, we can find the median of the breakpoints and calculate the lhs and rhs of (4.14) in $O(n)$. If the lhs$\geq$rhs, then we can discard all the breakpoints below the median. Likewise, if lhs$\leq$rhs we can drop all the values above the median. This step can also be done in $O(n)$, and reduces the size of the problem to $\frac{n}{2}$. The same procedure can be applied to the remaining breakpoints until we reach the desired interval for μ (an interval $[\mu_l,\ \mu_u]$ such that $lhs_{\mu_l} \geq rhs$ and $lhs_{\mu_u} \leq rhs$). Thus, the total running time is of the form $O(n) + O(\frac{n}{2}) + O(\frac{n}{4}) + \cdots$ which is $O(n)$.

4.2 IMRO - The Algorithm

The general format of algorithm IMRO is:

x^0 arbitrary,
for k=0,1,...
    Find σ^k and u^k, let $H^k = \sigma^k \mathbf{I} - u^k u^{k^t}$
    Solve (4.4) for x^{k+1} via algorithm 4.1.

In this section, we present two variants of IMRO. The difference between these two variants lies in the derivation of σ^k and u^k. We refer to these variants as IMRO-2D, for IMRO on two-dimensional subspace, and IMRO-1D for IMRO on one-dimensional subspace.

4.2.1 IMRO-1D

In IMRO-1D, we find σ and u such that the approximation model $m_H(x, x^k)$ equals $f(x)$ on a one-dimensional affine space $x^k + \alpha v$. We will later discuss the possible choices for v. Moreover, we require $m_H(x, x^k)$ to be an upper approximation for $f(x)$. The latter property has some theoretical benefits in convergence of the algorithm as we shall see in Section 4.3. The formal statement of these imposed constraints is

$$m_H(x, x^k) = f(x) \quad \text{whenever} \quad x \in x^k + \text{Span}\{v\}, \tag{4.15}$$

$$m_H(x, x^k) \geq f(x) \quad \forall x. \tag{4.16}$$

Using (4.2), we deduce that (4.15) implies that

$$\frac{1}{2}v^t H v = \frac{1}{2}v^t A^t A v, \tag{4.17}$$

and condition (4.16) implies that

$$\frac{1}{2}(x - x^k)^t H (x - x^k) \geq \frac{1}{2}(x - x^k) A^t A (x - x^k). \tag{4.18}$$

Obviously (4.18) holds if $H \succeq A^t A$. By (4.17) and (4.18), the required conditions on H boils down to

$$v^t (H - A^t A) v = 0, \tag{4.19}$$

$$H \succeq A^t A. \tag{4.20}$$

In the rest of this section we show how we can compute σ and u such that the above conditions are satisfied.

Finding σ, and u in IMRO-1D

Conditions (4.19) and (4.20) imply that

$$v \in \mathcal{N}(H - A^t A). \tag{4.21}$$

Without loss of generality, we assume that v is normalized, i.e. $\|v\| = 1$. The following lemma gives us the formula for σ and u in IMRO-1D.

Lemma 4.1. (4.19) *and* (4.20) *are satisfied for*

$$\sigma = \|A\|^2, \tag{4.22}$$

and

$$u = \begin{cases} \frac{\sigma v - A^t A v}{\sqrt{\sigma - \|Av\|^2}} & \text{if } v \text{ is not a dominant singular vector of } A, \\ 0 & \text{otherwise.} \end{cases} \tag{4.23}$$

Proof. Note that $\|A\|^2 = \lambda_{\max}(A^t A) = \sigma^2_{\max}(A)$, where $\lambda_{\max}$ and $\sigma_{\max}$ stand for the maximum eigenvalue and maximum singular value, respectively. Let us first consider the case

where v is a dominant singular vector of A. In this case $H = \sigma\mathbf{I} = \|A\|^2\mathbf{I} \succeq A^tA$ and $(\sigma\mathbf{I} - A^tA)v = 0$, so both requirements hold.

Suppose v is not a dominant singular vector of A. Then the denominator in the formula for u is positive and u is defined. We, therefore, have

$$(H - A^tA)v = \sigma v - (u^tv)u - A^tAv = \sigma v - (\sqrt{\sigma - \|Av\|^2})\frac{\sigma v - A^tAv}{\sqrt{\sigma - \|Av\|^2}} - A^tAv = 0,$$

which concludes equality (4.19). It remains to show (4.20), that is $x^t(\sigma\mathbf{I} - uu^t)x \geq x^tA^tAx$ for all $x \in \mathbb{R}^n$. Equivalently, we will show that for all $x \in \mathbb{R}^n$ such that $\|x\| = 1$ we have $\sigma \geq x^tA^tAx + (u^tx)^2$, i.e.,

$$\sigma \geq \sup_{\|x\|=1} \left\| \begin{pmatrix} A \\ u^t \end{pmatrix} x \right\|^2 = \left\| \begin{pmatrix} A \\ u^t \end{pmatrix} \right\|^2.$$

In fact, we prove that $\sigma = \left\| \begin{pmatrix} A \\ u^t \end{pmatrix} \right\|^2$. Clearly

$$\left\| \begin{pmatrix} A \\ u^t \end{pmatrix} \right\| \geq \|A\|,$$

because

$$\left\| \begin{pmatrix} A \\ u^t \end{pmatrix} x \right\| = \left\| \begin{pmatrix} Ax \\ u^tx \end{pmatrix} \right\| \geq \|Ax\| \quad \forall x \in \mathbb{R}^n.$$

It remains to show that $\left\| \begin{pmatrix} A \\ u^t \end{pmatrix} \right\| \leq \|A\|$. By the value of σ, we have $\sigma\mathbf{I} - A^tA \succeq 0$, so we can define B such that $BB^t = \sigma\mathbf{I} - A^tA$. Note that

$$\begin{aligned} x^t \begin{pmatrix} A^t & u \end{pmatrix} \begin{pmatrix} A \\ u^t \end{pmatrix} x = x^tA^tAx + \left(u^tx\right)^2 &= x^tA^tAx + \frac{\left(x^t\left(\sigma\mathbf{I} - A^tA\right)v\right)^2}{\sigma - \|Av\|^2} \\ &= x^tA^tAx + \frac{\left(x^t\left(\sigma\mathbf{I} - A^tA\right)v\right)^2}{v^t\left(\sigma - A^tA\right)v} \\ &= x^tA^tAx + \frac{\left(x^tBB^tv\right)^2}{v^tBB^tv} \\ &\leq x^tA^tAx + x^tBB^tx = \sigma x^tx, \end{aligned} \tag{4.24}$$

where the last inequality is ensured by Cauchy-Schwarz inequality, i.e.,

$$\left(x^tBB^tv\right)^2 \leq \|B^tx\|^2\|B^tv\|^2 = \left(x^tBB^x\right)\left(v^tBB^tv\right).$$

Combining the definition of induced matrix norms and the result obtained in (4.24) we get

$$\left\| \begin{pmatrix} A \\ u^t \end{pmatrix} \right\| = \sup_{x:\|x\|=1} x^t \begin{pmatrix} A^t & u \end{pmatrix} \begin{pmatrix} A \\ u^t \end{pmatrix} x \leq \sup_{x:\|x\|=1} \sigma x^t x = \sigma = \|A\|,$$

which yields the result we wanted to show. □

4.2.2 IMRO-2D

IMRO-2D is a variant of IMRO algorithm in which the quadratic model $m_H(x, x^k)$ matches the function on the two dimensional space of $x^k + \text{Span}\left\{\nabla f^k, x^k - x^{k-1}\right\}$. For convenience, let us denote $(x^k - x^{k-1})$ with d^k; furthermore without loss of generality let us assume that ∇f^k and d^k are normalized.

The imposed condition for IMRO-2D requires

$$f(x^k) + \langle \nabla f^k, x - x^k \rangle + \frac{1}{2}(x - x^k)^t H(x - x^k) = \frac{1}{2}\|A\left(x^k + (x - x^k)\right)\|^2, \tag{4.25}$$

for all $x \in \left\{x^k + \text{Span}\left\{\nabla f^k, d^k\right\}\right\}$, that is when $x - x^k = \alpha \nabla f^k + \beta d^k$.

Condition (4.25), therefore, reduces to

$$\begin{aligned} \frac{1}{2}(x - x^k)^t H(x - x^k) &= \frac{1}{2}(x - x^k)^t A^t A(x - x^k), \text{ i.e.,} \\ \frac{1}{2}(\alpha \nabla f^k + \beta d^k)^t H(\alpha \nabla f^k + \beta d^k) &= \frac{1}{2}(\alpha \nabla f^k + \beta d^k)^t A^t A(\alpha \nabla f^k + \beta d^k), \end{aligned} \tag{4.26}$$

for all $\alpha, \beta \in \mathbb{R}$. The fact that $H = \sigma \mathbf{I} - uu^t$ in IMRO enables us to find σ, τ, and ρ such that (4.26) is satisfied for σ and $u = \tau \nabla f^k + \rho d^k$. This is the topic covered in the remainder of this section.

Finding σ and u in IMRO-2D

By (4.26), we need to solve

$$(\alpha \nabla f^k + \beta d^k)^t A^t A(\alpha \nabla f^k + \beta d^k) = (\alpha \nabla f^k + \beta d^k)^t (\sigma I - uu^t)(\alpha \nabla f^k + \beta d^k), \tag{4.27}$$

for σ and u. We first derive σ, then using σ we will compute vector u.

Let S be the following matrix

$$S = \begin{pmatrix} \nabla f^k & d^k \end{pmatrix}^t A^t A \begin{pmatrix} \nabla f^k & d^k \end{pmatrix} = \begin{pmatrix} (\nabla f^k)^t A^t A \nabla f^k & (\nabla f^k)^t A^t A d^k \\ (d^k)^t A^t A \nabla f^k & (d^k)^t A^t A d^k \end{pmatrix}. \tag{4.28}$$

Then (4.27) impose the following equations on σ, and u

$$\begin{aligned} S_{11} &= \sigma - (\nabla f^k)^t u u^t \nabla f^k, \\ S_{12} &= \sigma (\nabla f^k)^t d^k - (\nabla f^k)^t u u^t d^k, \\ S_{22} &= \sigma - (d^k)^t u u^t d^k. \end{aligned} \tag{4.29}$$

Let ϵ be $(\nabla f^k)^t d^k$, an easily computable constant in each iteration, then

$$\begin{aligned} \det(S) = S_{11}S_{22} - S_{12}^2 &= \sigma^2 \left(1 - \epsilon^2\right) + \sigma \left(-(\nabla f^k)^t u u^t \nabla f^k - (d^k)^t u u^t d^k + 2\epsilon (\nabla f^k)^t u u^t d^k\right) \\ &= \sigma^2 (1 - \epsilon^2) + \sigma \left(S_{11} - \sigma + S_{22} - \sigma + 2\epsilon(\sigma\epsilon - S_{12})\right), \end{aligned}$$

using the set of equations in (4.29). Hence σ can be calculated by solving the following quadratic equation

$$\sigma^2 \underbrace{(1 - \epsilon^2)}_{\eta_1} + \sigma \underbrace{(-S_{11} - S_{22} + 2\epsilon S_{12})}_{\eta_2} + \underbrace{\det(S)}_{\eta_3} = 0. \tag{4.30}$$

Suppose $u = \tau \nabla f^k + \rho d^k$; using (4.29) we get

$$\begin{aligned} \sigma - S_{11} &= \tau^2 + 2\epsilon\tau\rho + \epsilon^2\rho^2 = (\tau + \epsilon\rho)^2, \\ \epsilon\sigma - S_{12} &= \epsilon\tau^2 + \tau\rho + \epsilon^2\tau\rho + \epsilon\rho^2 = (\tau + \epsilon\rho)(\epsilon\tau + \rho), \\ \sigma - S_{22} &= \epsilon^2\tau^2 + 2\epsilon\tau\rho + \rho^2 = (\epsilon\tau + \rho)^2, \end{aligned} \tag{4.31}$$

so $(\tau + \epsilon\rho) = \sqrt{\sigma - S_{11}}$ and $(\epsilon\tau + \rho) = \sqrt{\sigma - S_{22}}\ \mathrm{sgn}(\epsilon\sigma - S_{12})$. Therefore τ and ρ are the solutions of the linear system below

$$\begin{pmatrix} 1 & \epsilon \\ \epsilon & 1 \end{pmatrix} \begin{pmatrix} \tau \\ \rho \end{pmatrix} = \begin{pmatrix} \sqrt{\sigma - S_{11}} \\ \sqrt{\sigma - S_{22}}\ \mathrm{sgn}(\epsilon\sigma - S_{12}) \end{pmatrix}. \tag{4.32}$$

In what follows, we show that IMRO-2D is a valid algorithm, namely $\exists \sigma, \tau, \rho \in \mathbb{R}$ that solve (4.30) and (4.32).

Validity of IMRO-2D

Property 4.1. *Let η_1, η_2, and η_3 be defined as in (4.30). Then*

$$\eta_1 \geq 0, \tag{4.33}$$
$$\eta_2 \leq 0, \tag{4.34}$$
$$\eta_3 \geq 0. \tag{4.35}$$

Proof.

- Note that $\epsilon^2 \leq \epsilon \leq 1$ because $\epsilon = (\nabla f^k)^t d^k$ and $\|\nabla f^k\| = \|d^k\| = 1$, therefore $\eta_1 = 1 - \epsilon^2 \geq 0$.
- $\eta_3 = \det(S)$ and $S \succeq 0 \Rightarrow \eta_3 \geq 0$.
- $(S_{11} - S_{22})^2 \geq 0 \;\Leftrightarrow\; S_{11}^2 + S_{22}^2 + 2S_{11}S_{22} \geq 4S_{11}S_{22} \geq 4S_{12}^2 \geq 4\epsilon^2 S_{12}^2$, where the last two inequalities hold by $S \succeq 0$ and $\epsilon^2 \leq 1$, respectively; therefore $(S_{11} + S_{22})^2 \geq 4\epsilon^2 S_{12}^2 \;\Rightarrow S_{11} + S_{22} \geq 2\epsilon S_{12} \Rightarrow \eta_2 \leq 0$.

□

Claim 4.1. *Equation (4.30) has a real solution, i.e., σ exists.*

Proof. Basically we want to show that $\eta_2^2 - 4\eta_1\eta_3 \geq 0$. Note that

$$\begin{aligned}\eta_2^2 - 4\eta_1\eta_3 = &\left(S_{11}^2 + S_{22}^2 + 4\epsilon^2 S_{12}^2 + 2S_{11}S_{22} - 4\epsilon S_{11}S_{12} - 4\epsilon S_{22}S_{12}\right) \\ &+ \left(-4S_{11}S_{22} + 4S_{12}^2 + 4\epsilon^2 S_{11}S_{22} - 4\epsilon^2 S_{12}^2\right).\end{aligned} \tag{4.36}$$

Now let us make the following substitutions in (4.36)

$$S_{11}^2 = (1-\epsilon^2)S_{11}^2 + \epsilon^2 S_{11}^2, \tag{4.37}$$
$$S_{22}^2 = (1-\epsilon^2)S_{22}^2 + \epsilon^2 S_{22}^2, \tag{4.38}$$
$$4\epsilon^2 S_{11}S_{22} = 2\epsilon^2 S_{11}S_{22} + 2\epsilon^2 S_{11}S_{22}, \tag{4.39}$$

to get

$$\begin{aligned}\eta_2^2 - 4\eta_1\eta_3 &= (1-\epsilon^2)S_{11}^2 + \underline{\epsilon^2 S_{11}^2} + (1-\epsilon^2)S_{22}^2 + \underline{\epsilon^2 S_{22}^2} - \underline{4\epsilon S_{11}S_{12}} - \underline{4\epsilon S_{22}S_{12}} \\ &\quad -2S_{11}S_{22} + \underline{4S_{12}^2} + \underline{2\epsilon^2 S_{11}S_{22}} + 2\epsilon^2 S_{11}S_{22} \\ &= (\epsilon S_{11} + \epsilon S_{22} - 2S_{12})^2 + (1-\epsilon^2)(S_{11} - S_{22})^2 \geq 0.\end{aligned}$$

□

Claim 4.2. *$\sigma \geq S_{11}$ and $\sigma \geq S_{22}$; therefore u exists.*

Proof. We will prove it for S_{11}, the proof for S_{22} would be similar.

$$\begin{aligned}
(\epsilon S_{11} - S_{12})^2 \geq 0 &\Rightarrow S_{12}^2 \geq -\epsilon^2 S_{11}^2 + 2\epsilon S_{11} S_{12}, \\
&\Rightarrow S_{12}^2 - S_{11}S_{22} \geq -\epsilon^2 S_{11}^2 + 2\epsilon S_{11} S_{12} - S_{11}S_{22} + S_{11}^2 - S_{11}^2, \\
&\Rightarrow -\eta_3 \geq S_{11}^2 \eta_1 + S_{11}\eta_2, \\
&\Rightarrow -4\eta_1\eta_3 \geq 4\eta_1 \left(S_{11}^2\eta_1 + S_{11}\eta_2\right) = 4S_{11}^2\eta_1^2 + 4S_{11}\eta_1\eta_2, \\
&\Rightarrow \eta_2^2 + -4\eta_1\eta_3 \geq 4S_{11}^2\eta_1^2 + 4S_{11}\eta_1\eta_2 + \eta_2^2 = \left(2S_{11}\eta_1 + \eta_2\right)^2, \\
&\Rightarrow \sqrt{\eta_2^2 + -4\eta_1\eta_3} \geq 2S_{11}\eta_1 + \eta_2, \\
&\Rightarrow \sigma = \frac{-\eta_2 + \sqrt{\eta_2^2 + -4\eta_1\eta_3}}{2\eta_1} \geq S_{11}.
\end{aligned}$$

□

Claim 4.3. *Suppose σ and u are as defined in IMRO-2D by (4.30) and (4.32). Then $H = (\sigma \mathbf{I} - uu^t) \succeq 0$.*

Proof. We will prove that $\sigma \geq \|u\|^2$, which implies that $\sigma\|x\|^2 \geq \|u\|^2\|x\|^2 \geq (u^t x)^2$ for all $x \in \mathbb{R}^n$; thus $H \succeq 0$.

Recall that $u = \tau \nabla f^k + \rho d^k$, $\|\nabla f^k\| = \|d^k\| = 1$ and $\epsilon = (\nabla f^k)^t d^k$ by definition, so

$$\|u\|^2 = \tau^2 + \rho^2 + 2\tau\rho\epsilon. \tag{4.40}$$

In addition, recall (4.31) in which we had

$$\begin{aligned}
\sigma - S_{11} &= \tau^2 + 2\epsilon\tau\rho + \epsilon^2\rho^2, \\
\epsilon\sigma - S_{12} &= \epsilon\tau^2 + \tau\rho + \epsilon^2\tau\rho + \epsilon\rho^2, \\
\sigma - S_{22} &= \epsilon^2\tau^2 + 2\epsilon\tau\rho + \rho^2.
\end{aligned}$$

Let us multiply the second equation by -2ϵ and add the result to the summation of the

other two equations to get

$$\begin{aligned}
\text{lhs} &= \sigma - S_{11} - 2\epsilon^2\sigma + 2\epsilon S_{12} + \sigma - S_{22} = 2(1-\epsilon^2)\sigma + \eta_2 = 2\eta_1\sigma + \eta_2,\\
\text{rhs} &= \underline{\underline{\tau^2}} + \cancel{2\epsilon\tau\rho} + \underbrace{\epsilon^2\rho^2} - \underline{\underline{2\epsilon^2\tau^2}} - \cancel{2\epsilon\tau\rho} - \underline{2\epsilon^3\tau\rho} - \underbrace{2\epsilon^2\rho^2} + \underline{\underline{\epsilon^2\tau^2}} + \underline{2\epsilon\tau\rho} + \underbrace{\rho^2}\\
&= \tau^2(1-\epsilon^2) + 2\epsilon\tau\rho(1-\epsilon^2) + \rho^2(1-\epsilon^2) = (1-\epsilon^2)\|u\|^2 = \eta_1\|u\|^2,\\
&\Rightarrow 2\eta_1\sigma + \eta_2 = \eta_1\|u\|^2,\\
&\Rightarrow \cancel{-\eta_2} + \sqrt{\eta_2^2 - 4\eta_1\eta_3} + \cancel{\eta_2} = \eta_1\|u\|^2,\\
&\Rightarrow \|u\|^2 = \frac{\sqrt{\eta_2^2 - 4\eta_1\eta_3}}{\eta_1}.
\end{aligned}$$

By the value of σ, we have

$$\sigma - \|u\|^2 = \frac{-\eta_2 + \sqrt{\eta_2^2 - 4\eta_1\eta_3}}{2\eta_1} - \frac{\sqrt{\eta_2^2 - 4\eta_1\eta_3}}{\eta_1} = \frac{-\eta_2 - \sqrt{\eta_2^2 - 4\eta_1\eta_3}}{2\eta_1} \geq 0,$$

where the final inequality holds by property (4.1), i.e.,

$$\eta_1 \geq 0,\ \eta_3 \geq 0 \Rightarrow -4\eta_1\eta_3 \leq 0,$$

hence

$$\eta_2^2 - 4\eta_1\eta_3 \leq \eta_2^2 \Rightarrow \sqrt{\eta_2^2 - 4\eta_1\eta_3} \leq |\eta_2| = -\eta_2.$$

□

Note that $\sigma > \|u\|^2$ unless $\eta_1 = 0$ (i.e. $\epsilon = 0$) or $\eta_3 = 0$ (i.e. $\det(S) = 0$). Both cases happen only if ∇f^k and d^k are parallel, otherwise $H = \sigma\mathbf{I} - uu^t \succ 0$.

Before we start the analysis on the convergence of IMRO, we would like to point out that IMRO-2D reduces to linear CG (LCG) in the absence of $\lambda\|x\|_1$ term. In fact, IMRO-2D was slightly motivated by CG algorithms and in particular CGSO. The following theorem explains why IMRO-2D is essentially linear CG when the regularized term is missing.

Theorem 4.1. *Suppose IMRO-2D is applied to minimizing the quadratic function $\frac{1}{2}\|Ax - b\|^2$, then the sequence of iterates generated by IMRO-2D is the same as iterates generated in linear CG.*

Proof. Notice that

$$f(x) = \frac{1}{2}\|Ax - b\|^2 = \frac{1}{2}x^t \overbrace{A^tA}^{Q} x - x^t \overbrace{A^tb}^{c} + \frac{1}{2}b^tb.$$

The proof is by induction. Let x^0 and $r^0 = \nabla f(x^0) = Qx - c$ be the starting point for both algorithms. The subscript CG, classifies iterates for LCG from iterates obtained by IMRO-2D. For IMRO-2D at the first iteration we have

$$\frac{1}{2}(r^0)^t(\sigma \mathbf{I})r^0 = \frac{1}{2}(r^0)^t A^t A r^0 \quad \Rightarrow \quad \sigma = \frac{(r^0)^t A^t A r^0}{(r^0)^t r^0},$$

and

$$x^1 = x^0 + \frac{1}{\sigma}(-r^0).$$

By the fact that for LCG, $p^0_{CG} = -r^0$ and $\alpha^0 = \frac{(r^0)^t r^0}{(r^0)^t A^t A r^0} = \frac{1}{\sigma}$, we get $x^1_{CG} = x^1$. Suppose this holds true for k, i.e. $x^k_{CG} = x^k$. To ensure that $x^{k+1}_{CG} = x^{k+1}$ it suffices to show that $x^{k+1} \in x^k + \text{Span}\left\{r^k, p^k\right\}$. Because $m_H(x, x^k) = f(x)$ on the space of $x^k + \text{Span}\left\{r^k, p^k\right\}$, hence x^{k+1} must be the minimizer of $f(x)$ over the space $x^k + \text{Span}\left\{r^k, p^k\right\}$ which is x^{k+1}_{CG}.

Using optimality condition for our model $m_H(x)$ we get that

$$\begin{aligned} x^{k+1} &= x^k - H^{-1}r^k = x^k - \left(\sigma \mathbf{I} - uu^t\right)^{-1} r^k \\ &= x^k - \left(\frac{1}{\sigma}\mathbf{I} - \frac{1}{\sigma(\|u\|^2 - \sigma)}uu^t\right) r^k \\ &= x^k - \frac{1}{\sigma}r^k + \frac{u^t r^k}{\sigma(\|u\|^2 - \sigma)}(\tau r^k + \rho p^k) \in \left\{x^k + \text{Span}\{r^k, p^k\}\right\}. \end{aligned}$$

□

4.3 Convergence of IMRO

The difference of IMRO and other proximal quasi-Newton methods is the special structure of H. The format of H in IMRO facilitates computation of the next iterate as mentioned earlier in this chapter. The convergence properties of IMRO, however, can mostly be generalized to other variants of proximal quasi-Newton methods. In the preceding sections, we established that $H \succeq 0$ for both IMRO-1D and IMRO-2D. Furthermore, the conditions under which H is singular are apparently unusual (that v^k is a dominant singular vector of A in the case of IMRO-1D; that ∇f^k and d^k are parallel in the case of IMRO-2D) and never arose in our computational experiments. Therefore, for the remainder of this section, we assume $H \succ 0$. If one of these unusual cases arose in practice, we could simply modify the algorithm by replacing σ by $\sigma + \epsilon$ for some small $\epsilon > 0$ to ensure that $H \succ 0$.

Let us review the general scheme of these algorithms. The problem we aim to solve is

$$\min \quad F(x) := f(x) + p(x), \tag{4.41}$$

where $f(x) \in \mathbf{C}_L$ (such as $f(x) = \frac{1}{2}\|Ax - b\|^2$) and $p(x)$ is a convex function (such as $\lambda\|x\|_1$). We find each iterate by solving

$$\begin{aligned} \min_x \quad M_H(x, x^k) :&= m_H(x, x^k) + p(x) \\ &= \frac{1}{2}\|x - x^k\|_H^2 + \langle \nabla f^k, x - x^k \rangle + f(x^k) + \lambda\|x\|_1, \end{aligned} \tag{4.42}$$

where $H \succ 0$. Let us fix the following notation:

$$M_H^*(x^k) = \min \quad M_H(x, x^k), \tag{4.43}$$

$$x_H^*(x^k) = \arg\min \quad M_H(x, x^k), \tag{4.44}$$

$$g_H^k(x^k) = H(x^k - \bar{x}_H^k). \tag{4.45}$$

Throughout this section, we use the compact notation of $M_H^k(x)$, $\bar{x}_H^k$ and g_H^k for $M_H(x, x^k)$, $x_H^*(x^k)$ and $g_H^k(x^k)$, respectively.

Note that optimality conditions for (4.42) imply that

$$g_H^k = \nabla f^k + \xi^{k+}, \tag{4.46}$$

where $\xi^{k+} \in \partial\left(p(\bar{x}_H^k)\right)$. We will see in this section that the notion of the scaled gradient, g_H^k, mimics some of the properties of the gradient in unconstrained optimization. An important property of g_H^k is captured in the following property.

Property 4.2. *$g_H^k = 0$ if and only if x^k is the optimizer of the problem.*

Proof. Note that if $g_H^k = 0$, then $x^k - \bar{x}_H^k = 0$ because $H \succ 0$ (thus invertible). This implies that $x^k = \bar{x}_H^k$. Therefore (4.46) reduces to optimality condition for (4.41). Likewise if x^k is the optimal solution of (4.41), then $\nabla f^k + \xi^k = 0$ implies that $\bar{x}_H^k = x^k$; thus $g_H^k = 0$. □

The following lemma, presented in [74], shows that in fact direction $\bar{x}_H^k - x^k = -H^{-1}g_H^k$ is a descent direction; in other words using this direction armed with a line search we attain the next iterate, x^{k+1}, for which we have $F(x^{k+1}) < F(x^k)$.

Lemma 4.2. *Suppose the scheme of (4.42) with some $H \succ 0$ has been applied to problem (4.41). Let $x^{k+1} = x^k + \alpha(\bar{x}_H^k - x^k)$. Then $F(x^{k+1}) < F(x^k)$, for sufficiently small step size α.*

Proof. Let us denote $\bar{x}_H^k - x^k$ by d^k. Since $\bar{x}_H^k$ is the unique optimizer of (4.42), we have

$$\begin{aligned}\frac{1}{2}\|d^k\|_H^2 + \langle \nabla f^k, d^k\rangle + f(x^k) + p(\bar{x}_H^k) &< \frac{\alpha^2}{2}\|d^k\|_H^2 + \alpha\langle \nabla f^k, d^k\rangle + f(x^k) + p(x^{k+1}) \\ &\leq \frac{\alpha^2}{2}\|d^k\|_H^2 + \alpha\langle \nabla f^k, d^k\rangle + f(x^k) + \alpha p(\bar{x}_H^k) + (1-\alpha)p(x^k),\end{aligned}$$

where the last inequality follows from convexity of $p(x)$. Rearranging the terms and dividing by $1-\alpha$ we get

$$\langle \nabla f^k, d^k\rangle + p(\bar{x}_H^k) - p(x^k) < -\frac{1+\alpha}{2}\|d^k\|_H^2. \tag{4.47}$$

Using the convexity of $p(x)$ and the Taylor expansion for $f(x)$, we derive

$$\begin{aligned}F(x^{k+1}) - F(x^k) &= f(x^{k+1}) - f(x^k) + p(x^{k+1}) - p(x^k) \\ &\leq \alpha\langle \nabla f^k, d^k\rangle + O(\alpha^2) + \alpha p(\bar{x}_H^k) + (1-\alpha)p(x^k) - p(x^k) \\ &= \alpha\left[\langle \nabla f^k, d^k\rangle + p(\bar{x}_H^k) - p(x^k)\right] + O(\alpha^2) < 0,\end{aligned} \tag{4.48}$$

by (4.47) for sufficiently small values of α. □

The above lemma indicates that both variants of IMRO coupled with a proper line search generate a decreasing sequence. The following lemma shows that under certain condition (that always holds for IMRO-1D), no line search is required to get a descent algorithm.

Lemma 4.3. *Suppose the scheme of* (4.42) *with some* $H \succ 0$ *has been applied to problem* (4.41)*, and*

$$F(\bar{x}_H^k) \leq M_H^k(\bar{x}_H^k). \tag{4.49}$$

Let $x^{k+1} = x^k + \alpha(\bar{x}_H^k - x^k)$*, where* $\alpha \in (0,1)$*. Then*

$$F(x^{k+1}) < F(x^k).$$

Proof. By the hypothesis and the fact that $\bar{x}_H^k$ is the unique minimizer of M_H^k we get

$$F(\bar{x}_H^k) \leq M_H^k(\bar{x}_H^k) < M_H^k(x^k) = F(x^k).$$

We now attain the desired result using the convexity of $F(x)$:

$$F(x^{k+1}) \leq \alpha F(\bar{x}_H^k) + (1-\alpha)F(x^k) < F(x^k).$$

□

Note that for IMRO-1D condition (4.49) always hold because $H \succeq A^t A$.

The following lemma which is analogous to [9, Lemma 2.3] plays a key role in the convergence of IMRO.

Lemma 4.4. *Suppose $x^{k+1} = \bar{x}_H^k$ and $F(x^{k+1}) \leq M_H^k(x^{k+1})$. For $\forall x \in \mathbb{R}^n$ we have*

$$F(x) - F(x^{k+1}) \geq \frac{1}{2}\|x^{k+1} - x^k\|_H^2 + \langle g_H^k, x - x^k \rangle.$$

Proof. Recall that we have

$$H(x^{k+1} - x^k) + \nabla f(x^k) + \xi = 0, \tag{4.50}$$

where $\xi \in \partial\left(p(x^{k+1})\right)$. By hypothesis we have

$$F(x) - F(x^{k+1}) \geq F(x) - M_H^k(x^{k+1}), \tag{4.51}$$

and by convexity of $f(x)$ and $p(x)$ we derive

$$f(x) \geq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle, \tag{4.52}$$
$$p(x) \geq p(x^{k+1}) + \langle \xi, x - x^{k+1} \rangle. \tag{4.53}$$

Summing the above inequalities, we get

$$F(x) \geq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + p(x^{k+1}) + \langle \xi, x - x^{k+1} \rangle. \tag{4.54}$$

Substituting (4.54) in (4.51) gives us

$$\begin{aligned} F(x) - F(x^{k+1}) \geq\ & \cancel{f(x^k)} + \langle \nabla f(x^k), x - x^k \rangle + \cancel{p(x^{k+1})} + \langle \xi, x - x^{k+1} \rangle \\ & - \frac{1}{2}\|x^{k+1} - x^k\|_H^2 - \langle \nabla f(x^k), x^{k+1} - x^k \rangle - \cancel{f(x^k)} - \cancel{p(x^{k+1})} \\ =\ & -\frac{1}{2}\|x^{k+1} - x^k\|_H^2 + \langle \nabla f(x^k) + \xi, x - x^{k+1} \rangle \\ =\ & -\frac{1}{2}\|x^{k+1} - x^k\|_H^2 + \langle H(x^k - x^{k+1}), x - x^k + (x^k - x^{k+1}) \rangle \\ =\ & \frac{1}{2}\|x^{k+1} - x^k\|_H^2 + \langle g_H^k, x - x^k \rangle. \end{aligned}$$

□

We may obtain a stronger bound on $F(x) - F(x^{k+1})$ in Lemma 4.4 for strongly convex functions. Suppose $f \in \mathbf{C}_{l,L}$ in problem (4.41), then inequality (4.52) might be replaced with

$$f(x) \geq f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{l}{2}\|x - x^k\|^2,$$

to get the following lemma.

Lemma 4.5. *Suppose $f \in \mathbf{C}_{l,L}$ in problem (4.41), $x^{k+1} = \bar{x}_H^k$ and $F(x^{k+1}) \leq M_H^k(x^{k+1})$. Then for $\forall x \in \mathbb{R}^n$*

$$F(x) - F(x^{k+1}) \geq \frac{l}{2}\|x - x^k\|^2 + \frac{1}{2}\|x^{k+1} - x^k\|_H^2 + \langle g_H^k, x - x^k \rangle.$$

As mentioned earlier, in IMRO we have $H \preceq \sigma \mathbf{I}$, thus $H^{-1} \succeq \frac{1}{\sigma}\mathbf{I}$ and

$$\|x\|_{H^{-1}}^2 \geq \frac{1}{\sigma}\|x\|^2. \tag{4.55}$$

Also recall that $\bar{x}_H^k - x^k = H^{-1} g_H^k$. As a result we get the following corollaries.

Corollary 4.1. *Suppose $x^{k+1} = \bar{x}_H^k$ and $F(x^{k+1}) \leq M_H^k(x^{k+1})$. For $\forall x \in \mathbb{R}^n$ we have*

$$F(x) \geq F(x^{k+1}) + \langle g_H^k, x - x^k \rangle + \frac{1}{2}\|g_H^k\|_{H^{-1}}^2 \tag{4.56}$$

$$\geq F(x^{k+1}) + \langle g_H^k, x - x^k \rangle + \frac{1}{2\sigma}\|g_H^k\|^2. \tag{4.57}$$

Proof. Immediately follows from Lemma 4.4 and inequality (4.55). □

Corollary 4.2. *Let $f \in \mathbf{C}_{l,L}$ in problem (4.41), $x^{k+1} = \bar{x}_H^k$ and $F(x^{k+1}) \leq M_H^k(x^{k+1})$. Then for $\forall x \in \mathbb{R}^n$ we have*

$$F(x) \geq F(x^{k+1}) + \langle g_H^k, x - x^k \rangle + \frac{1}{2}\|g_H^k\|_{H^{-1}}^2 + \frac{l}{2}\|x - x^k\|^2 \tag{4.58}$$

$$\geq F(x^{k+1}) + \langle g_H^k, x - x^k \rangle + \frac{1}{2\sigma}\|g_H^k\|^2 + \frac{l}{2}\|x - x^k\|^2. \tag{4.59}$$

Proof. Immediately follows from Lemma 4.5 and inequality (4.55). □

Corollary 4.3. *Let $x^{k+1} = \bar{x}_H^k$ and $F(x^{k+1}) \leq M_H^k(x^{k+1})$. Then*

$$F(x^{k+1}) \leq F(x^k) - \frac{1}{2\sigma}\|g_H^k\|^2. \tag{4.60}$$

Proof. It is derived by applying Corollary 4.1 at $x = x^k$. □

Inequality (4.60) clarifies more similarities between the scaled gradient, g_H, and the notion of the gradient in smooth unconstrained problems. Suppose that $\sigma = L$ as in IMRO-1D, then inequality (4.60) implies

$$F(x^{k+1}) \leq F(x^k) - \frac{1}{2L}\|g_H^k\|^2, \tag{4.61}$$

which is similar to the inequality we had in unconstrained smooth optimization for sufficient reduction in the objective value at each iteration.

Corollary 4.4. *Suppose $f \in \mathbf{C}_{l,L}$ in problem (4.41), $x^{k+1} = \bar{x}_H^k$ and $F(x^{k+1}) \leq M_H^k(x^{k+1})$. Then*

$$\langle g_H^k, x^k - x^* \rangle \geq \frac{1}{2\sigma}\|g_H^k\|^2 + \frac{l}{2}\|x^* - x^k\|^2, \tag{4.62}$$

where x^* is the minimizer of $F(x)$.

Proof. Applying Corollary 4.2 at $x = x^*$ along with the fact that $F(x^*) \leq F(x^{k+1})$ concludes the desired result. □

As explained, our gradient based method, IMRO, is as follows.

Algorithm 4.2.

Let $x^0 \in \mathbb{R}^n$ be an arbitrary starting point and $x^1 = Prox_p(x^0 - \nabla f^0/\sigma)$.

for $i = 1, 2, \ldots$

Find σ and u:

equations (4.22) and (4.23) for IMRO-1D

equations (4.30) and (4.32) for IMRO-2D

Find x^{k+1} using Algorithm 4.1

Update ∇f^k and d^k

The following theorem states the linear convergence of Algorithm 4.2 for strongly convex functions.

Theorem 4.2. *Suppose $f \in \mathbf{C}_{l,L}$ in problem (4.41), $\sigma \geq L$, and $\alpha \leq \frac{1}{\sigma}$ in Algorithm 4.2. Then*

$$\|x^k - x^*\|^2 \leq (1 - \alpha l)^k \|x^0 - x^*\|^2. \tag{4.63}$$

Proof. Proof is by induction. The base case, i.e., $k = 0$ is true. Assume it holds for k; then

$$\|x^{k+1} - x^*\|^2 = \|x^k - \alpha g_H^k - x^*\|^2 \tag{4.64}$$
$$= \|x^k - x^*\|^2 - 2\alpha\langle g_H^k, x^k - x^*\rangle + \alpha^2\|g_H^k\|^2 \tag{4.65}$$
$$\leq (1 - \alpha l)\|x^k - x^*\|^2 + (\alpha^2 - \frac{\alpha}{\sigma})\|g_H^k\|^2 \tag{4.66}$$
$$= (1 - \alpha l)\|x^k - x^*\|^2, \tag{4.67}$$

where (4.66) follows from Corollary 4.4. □

Since in IMRO-1D, $\sigma \geq L$, we can conclude that it has linear convergence by the above theorem for the class of strongly convex functions. Moreover, because BPDN problem is not strongly convex, l might be zero in which case Theorem 4.2 does not apply. In what follows we derive a lemma that leads us to the sublinear convergence of IMRO-1D for convex problems.

Lemma 4.6. *Suppose $x^{k+1} = \bar{x}_H^k$, and $F(x^{k+1}) \leq M_H^k(x^{k+1})$. Then*

$$F(x^k) - F(x^{k+1}) \geq \frac{\big(F(x^{k+1}) - F(x^*)\big)^2}{2\sigma^2\delta^2}, \tag{4.68}$$

where δ is the diameter of the level set of x^0.

Proof. By Lemma 4.4 at x^* we have

$$\begin{aligned}
F(x^*) - F(x^{k+1}) &\geq \frac{1}{2}\|x^{k+1} - x^k\|_H^2 + \langle H(x^k - x^{k+1}), x^* - x^k\rangle \\
&= \frac{1}{2}\left(\langle H(x^k - x^{k+1}), x^k - x^{k+1} + 2x^* - 2x^k\rangle\right) \\
&= \frac{1}{2}\left(\langle H(x^* - x^{k+1}) - H(x^* - x^k), (x^* - x^{k+1}) + (x^* - x^k)\rangle\right) \\
&= \frac{1}{2}\left(\|x^* - x^{k+1}\|_H^2 - \|x^* - x^k\|_H^2\right) \\
&= \frac{1}{2}\left(\|x^* - x^{k+1}\|_H - \|x^* - x^k\|_H\right)\left(\|x^* - x^{k+1}\|_H + \|x^* - x^k\|_H\right) \\
&\geq \frac{1}{2}\left(-\|x^{k+1} - x^k\|_H\right)\left(\|x^* - x^{k+1}\|_H + \|x^* - x^k\|_H\right),
\end{aligned}$$

where the last line is by triangle inequality. Therefore we have

$$\|x^{k+1} - x^k\|_H \geq \frac{2\left(F(x^{k+1}) - F(x^*)\right)}{\|x^* - x^{k+1}\|_H + \|x^* - x^k\|_H} \geq \frac{F(x^{k+1}) - F(x^*)}{\sigma\delta}, \tag{4.69}$$

by the fact that $\|x^k - x^*\|, \|x^{k+1} - x^*\| \leq \delta$ because IMRO is a descent method, and $\|x\|_H \leq \sigma\|x\|$ by the choice of H in IMRO.

Moreover, by Lemma 4.4 at x^k we have

$$F(x^k) - F(x^{k+1}) \geq \frac{1}{2}\|x^{k+1} - x^k\|_H^2. \tag{4.70}$$

Applying inequality (4.69) concludes the result we wanted to show. □

The sublinear convergence of IMRO-1D is established using the following lemma.

Lemma 4.7. *Suppose $\{\omega^k\} \to \omega^*$ is a decreasing sequence, $\omega^k - \omega^{k+1} \geq \frac{(\omega^{k+1}-\omega^*)^2}{\mu}$ for all k, and $\omega_1 - \omega^* \leq 4\mu$. For all k we have*

$$\omega^k - \omega^* \leq \frac{4\mu}{k}.$$

Proof. Proof is by induction. For $k = 1$, the result holds by hypothesis. Let $p_k = \frac{4\mu}{k}$, then

$$\begin{aligned}\omega^{k+1} - \omega^* &= \omega^k - \omega^* + \omega^{k+1} - \omega^k \\ &\leq \omega^k - \omega^* - \frac{(\omega^{k+1} - \omega^*)^2}{\mu} \\ &\leq p_k - \frac{(\omega^{k+1} - \omega^*)^2}{\mu}.\end{aligned}$$

Let $\nu = \omega^{k+1} - \omega^*$; then the above inequality is

$$\frac{\nu^2}{\mu} + \nu - p_k \leq 0, \tag{4.71}$$

which has nonnegative solution by

$$\nu \leq \frac{-\mu + \sqrt{\mu^2 + 4p_k\mu}}{2} = \frac{2p_k}{1 + \sqrt{1 + \frac{4p_k}{\mu}}}. \tag{4.72}$$

Note that function $f(x) = \frac{1}{1+\sqrt{1+x}}$ is convex; thus on any interval $[0, a]$, it is bounded above by its secant interpolant.

We now consider two separate cases; when $k = 1$ to show that lemma holds for $k + 1 = 2$, and when $k \geq 2$ to show that the lemma holds for $k + 1 \geq 3$. For $k = 1$ we have $p_1 \leq 4\mu$, so $\frac{4p_1}{\mu} \leq 16$. Therefore

$$\nu \leq 2p_1 \left(\frac{1}{2} + \frac{\frac{1}{1+\sqrt{17}} - \frac{1}{2}}{16} \frac{4p_1}{\mu} \right) \leq \frac{4\mu}{2}, \tag{4.73}$$

i.e., the lemma holds for $k = 2$. For $k \geq 2$, $p_k \leq \frac{4\mu}{k} \leq 2\mu$, so $\frac{4p_k}{\mu} \leq 8$; hence

$$\begin{aligned} \nu &\leq 2p_k \left(\frac{1}{2} + \frac{\frac{1}{1+\sqrt{9}} - \frac{1}{2}}{8} \frac{4p_k}{\mu} \right) \\ &= 4\mu \left(\frac{1}{k} + 4 \left(\frac{1}{4} - \frac{1}{2} \right) \frac{1}{k^2} \right) \leq \frac{4\mu}{k+1}, \end{aligned} \tag{4.74}$$

where the last inequality follows from the fact that $\frac{1}{k} - \frac{1}{k^2} \leq \frac{1}{k+1}$. □

The sublinear convergence of IMRO-1D is a direct conclusion of applying the previous lemma to $\omega^k = F(x^k)$ and $\mu = 2\sigma^2\delta^2$ along with Lemma 4.6.

To sum up this section we would like to point out that in IMRO-1D we fix σ; however, even if it was changing we could set $\sigma = \max\{\sigma_i\}$ and the previous results hold.

4.4 FIMRO - Accelerated Variant of IMRO

We briefly introduced the notion of an estimate sequence in Definition 3.1, and mentioned some ground properties of Nesterov's accelerated technique for minimizing function $f \in \mathbf{C}_{l,L}$ in Chapter 3. In this section, we present in details how we can extend the theory of Nesterov's method to IMRO, particularly IMRO-1D, for minimizing a composite function. The resulting algorithm is named FIMRO for fast IMRO.

As in Nesterov's method, we have two sequences $\{y^k\}$ and $\{x^k\}$ in this section. The model $M_H(x, y^k)$ is built using y^k, while its solution generates $\{x^k\}$. In other words, we have the following:

$$M_H{}^*(y^k) = \min \quad M_H(x, y^k) \tag{4.75}$$

$$x_H^*(y^k) = \arg\min \quad M_H(x, y^k) \tag{4.76}$$

$$g_H^k(y^k) = H(y^k - \bar{x}_H^k) \equiv g_H^k \ . \tag{4.77}$$

Similar to what was described in Definition 3.1, a pair of sequences $\{\phi^k(x)\}$ and $\{\lambda^k\}$, $\lambda^k \geq 0$ are called an estimate sequence of a composite function $F(x)$ if $\lambda^k \to 0$ and for any $x \in \mathbb{R}^n$ and all $k \geq 0$ we have

$$\phi^k(x) \leq (1 - \lambda^k)F(x) + \lambda^k \phi^0(x).$$

The following two lemmas which are analogous to [85, Lemmas 2.2.2 and 2.2.3] summarize how we can construct an estimate sequence for the composite function F. The first one recursively generates the estimate sequence using the scaled gradient at each iterate; while the second one presents a closed form for the recursively generated sequence.

Lemma 4.8. *Suppose $\{y^k\}$ is an arbitrary sequence, $\{\alpha^k\}$ is a sequence such that $\alpha^k \in (0, 1)$ and $\sum_{k=0}^{\infty} \alpha^k = \infty$, and $\lambda^0 = 1$. Then the pair of sequences $\{\lambda^k\}$ and $\{\phi^k(x)\}$ generated as*

$$\begin{aligned}
\lambda^{k+1} &= (1 - \alpha^k)\lambda^k, \\
\phi^{k+1}(x) &= (1 - \alpha^k)\phi^k(x) + \alpha^k \left[\underbrace{F(x_H^*(y^k)) + \langle g_H^k, x - y^k \rangle + \frac{1}{2\sigma}\|g_H^k\|^2}_{\psi(x)} \right],
\end{aligned}$$

is an estimate sequence for $F(x)$.

Proof. Note that by Corollary 4.1, $F(x) \geq \psi(x)$ for $\forall x \in \mathbb{R}^n$. Our proof is by induction. The base case holds true for $k = 0$. Suppose it holds true for k, then for $k + 1$ we have

$$\begin{aligned}
\phi^{k+1}(x) &= (1 - \alpha^k)\phi^k(x) + \alpha^k \psi(x) \leq (1 - \alpha^k)\phi^k(x) + \alpha^k F(x) \\
&= (1 - (1 - \alpha^k)\lambda^k)F(x) + (1 - \alpha^k)\left(\phi^k(x) - (1 - \lambda^k)F(x)\right) \\
&\leq (1 - (1 - \alpha^k)\lambda^k)F(x) + (1 - \alpha^k)\lambda^k \phi^0(x) \\
&= (1 - \lambda^{k+1})F(x) + \lambda^{k+1}\phi^0(x).
\end{aligned}$$

□

In the following lemma we show how we may write $\phi^{k+1}(x)$ in closed form.

Lemma 4.9. *Suppose $\phi^0(x) = F(x^0) + \frac{\gamma^0}{2}\|x - z^0\|^2$. Then $\phi^{k+1}(x)$ generated by the recursive formulation of the previous lemma is*

$$\phi^{k+1}(x) = \bar{\phi}^{k+1} + \frac{\gamma^{k+1}}{2}\|x - z^{k+1}\|^2,$$

where

$$\gamma^{k+1} = (1-\alpha^k)\gamma^k,$$
$$z^{k+1} = z^k - \frac{\alpha^k}{\gamma^{k+1}} g_H^k,$$
$$\bar{\phi}^{k+1} = (1-\alpha^k)\bar{\phi}^k + \alpha^k F(x_H^*(y^k)) + \left(\frac{\alpha^k}{2\sigma} - \frac{(\alpha^k)^2}{2\gamma^{k+1}}\right)\|g_H^k\|^2 + \alpha^k\langle g_H^k, z^k - y^k\rangle.$$

Proof. The proof is by induction. The base case for $k = 0$ holds. Suppose for k we have

$$\phi^k(x) = \bar{\phi}^k + \frac{\gamma^k}{2}\|x - z^k\|^2,$$

then by the previous lemma we have

$$\phi^{k+1}(x) = (1-\alpha^k)\left[\bar{\phi}^k + \frac{\gamma^k}{2}\|x - z^k\|^2\right] + \alpha^k\left[F(x_H^*(y^k)) + \langle g_H^k, x - y^k\rangle + \frac{1}{2\sigma}\|g_H^k\|^2\right].$$

Using the fact that ϕ is a quadratic function we get

$$\nabla^2\phi^{k+1}(x) = \gamma^{k+1} = (1-\alpha^k)\gamma^k.$$

By $\nabla\phi^{k+1}(x) = 0$, we get the minimizer of ϕ^{k+1} which is $\frac{1}{\gamma^{k+1}}z^{k+1}$, so

$$\nabla\phi^{k+1}(x) = (1-\alpha^k)\gamma^k(x - z^k) + \alpha^k g_H^k,$$
$$\Rightarrow x = z^{k+1} = \arg\min \phi^{k+1}(x) = z^k - \frac{\alpha^k}{\gamma^{k+1}} g_H^k.$$

To find $\bar{\phi}^{k+1}$ we set equal the $\phi^{k+1}(y^k)$ in both formulations of ϕ^{k+1}. We, therefore, have

$$\bar{\phi}^{k+1} + \frac{\gamma^{k+1}}{2}\|y^k - z^{k+1}\|^2 = \bar{\phi}^{k+1} + \frac{\gamma^{k+1}}{2}\|z^k - y^k\|^2 - \alpha^k\langle g_H^k, z^k - y^k\rangle + \frac{(\alpha^k)^2}{2\gamma^{k+1}}\|g_H^k\|^2$$
$$= (1-\alpha^k)\bar{\phi}^k + \frac{(1-\alpha^k)\gamma^k}{2}\|y^k - z^k\|^2 + \alpha^k F(x_H^*(y^k)) + \frac{\alpha^k}{2\sigma}\|g_H^k\|^2,$$

i.e.,

$$\bar{\phi}^{k+1} = (1-\alpha^k)\bar{\phi}^k + \alpha^k F(x_H^*(y^k)) + \left(\frac{\alpha^k}{2\sigma} - \frac{(\alpha^k)^2}{2\gamma^{k+1}}\right)\|g_H^k\|^2 + \alpha^k\langle g_H^k, z^k - y^k\rangle. \quad (4.78)$$

□

Similar to what we mentioned for unconstrained smooth case in Chapter 3, we would like to construct y^k such that $\bar{\phi}^{k+1} \geq F(x_H^*(y^k)) = F(x^{k+1})$. The benefit of this condition will be clear in Theorem 4.3. Note that for $k = 0$, $\bar{\phi}^0 = F(x^0)$, so the condition holds. Let $x^{k+1} = x_H^*(y^k)$, and suppose the required condition is satisfied for k, i.e., $\bar{\phi}^k \geq F(x^k)$. Using Corollary 4.1 at $x^k = y^k$ and $x = x^k$ we derive

$$\bar{\phi}^k \geq F(x^k) \geq F(x_H^*(y^k)) + \langle g_H^k x^k - y^k \rangle + \frac{1}{2\sigma}\|g_H^k\|^2. \tag{4.79}$$

Substituting inequality (4.79) in the equation (4.78) we get

$$\begin{aligned}\bar{\phi}^{k+1} \geq F(x_H^*(y^k)) + \left(\frac{1}{2\sigma} - \frac{(\alpha^k)^2}{2\gamma^{k+1}}\right)\|g_H^k\|^2 \\ + \langle g_H^k, \alpha^k(z^k - y^k) + (1-\alpha^k)(x^k - y^k)\rangle.\end{aligned} \tag{4.80}$$

To make sure that $\bar{\phi}^{k+1} \geq F(x_H^*(y^k))$, we need to set

$$\sigma(\alpha^k)^2 = (1-\alpha^k)\gamma^k \equiv \gamma^{k+1}, \tag{4.81}$$

$$y^k = \alpha^k z^k + (1-\alpha^k)x^k. \tag{4.82}$$

Equation (4.81) ensures that the coefficient of $\|g_H^k\|^2$ is zero, and (4.82) makes the linear term vanish. The proposed accelerated scheme is summarized as follows.

Algorithm 4.3.

Let $z^0 = x^0$ be arbitrary initial points, $\gamma^0 \geq L$

for *i=0,1,...*

Compute α^k as $\sigma(\alpha^k)^2 = (1-\alpha^k)\gamma^k$

Let $\gamma^{k+1} = \sigma(\alpha^k)^2$

Let $y^k = \alpha^k z^k + (1-\alpha^k)x^k$

Compute $f(y^k)$ and $\nabla f(y^k)$

Find σ^k and u^k:

equations (4.22) and (4.23) (as in IMRO-1D)

Find $x^{k+1} = x_H^*(y^k)$ using algorithm 4.1

(Note that for this choice of x we get $F(x^{k+1}) \leq F(y^k) - \frac{1}{2\sigma}\|g_H^k\|^2$)

Let $z^{k+1} = z^k - \frac{\alpha^k}{\gamma^{k+1}} g_H^k$

The following theorem reveals the importance of condition $\bar{\phi}^{k+1} \geq F(x_H^*(y^k))$.

Theorem 4.3. *Suppose $F(x^k) \leq \bar{\phi}^k$ holds true for a sequence $\{x^k\}$, then*

$$F(x^k) - F(x^*) \leq \lambda^k \left[F(x^0) + \frac{\gamma^0}{2}\|x^0 - x^*\|^2 - F(x^*)\right].$$

Proof. By definition of an estimate sequence we get

$$F(x^k) \leq \bar{\phi}^k \leq \min_x (1-\lambda^k)F(x) + \lambda^k \phi^0(x) \leq (1-\lambda^k)F(x^*) + \lambda^k \phi^0(x^*),$$

$$\Rightarrow F(x^k) - F(x^*) \leq \lambda^k [\phi^0(x^*) - F(x^*)] = \lambda^k \left[F(x^0) + \frac{\gamma^0}{2}\|x^0 - x^*\|^2 - F(x^*) \right].$$

□

The beauty of the above theorem lies in the fact that the convergence of $\{x^k\}$ follows the convergence rate of $\{\lambda^k\}$. It remains to find the convergence rate of $\{\lambda^k\}$.

Lemma 4.10. *[85, 2.2.4] Suppose $\gamma^0 \geq L$, then*

$$\lambda^k \leq \frac{4\sigma}{(2\sqrt{\sigma} + k\sqrt{\gamma^0})^2}.$$

Proof. Inductively, we first show that $\gamma^0 \lambda^k \leq \gamma^k$. It holds by our assumption that $\lambda^0 = 1$ for 0. Suppose it holds for k, then for $k+1$ we get

$$\gamma^0 \lambda^{k+1} = \gamma^0 \lambda^k (1 - \alpha^k) \leq \gamma^k (1 - \alpha^k) = \gamma^{k+1}.$$

Using the fact $\gamma^{k+1} = \sigma(\alpha^k)^2$ we conclude that $\alpha^k \geq \sqrt{\frac{\gamma^0 \lambda^{k+1}}{\sigma}}$.

Let $\tau^k = \frac{1}{\sqrt{\lambda^k}}$, be an increasing sequence; then we have

$$\begin{aligned} \tau^{k+1} - \tau^k &= \frac{\sqrt{\lambda^k} - \sqrt{\lambda^{k+1}}}{\sqrt{\lambda^k \lambda^{k+1}}} = \frac{\lambda^k - \lambda^{k+1}}{\sqrt{\lambda^k \lambda^{k+1}}\sqrt{\lambda^k} + \sqrt{\lambda^{k+1}}} \\ &\geq \frac{\lambda^k - \lambda^{k+1}}{2\lambda^k \sqrt{\lambda^{k+1}}} = \frac{\lambda^k - (1-\alpha^k)\lambda^k}{2\lambda^k \sqrt{\lambda^{k+1}}} = \frac{\alpha^k}{2\sqrt{\lambda^{k+1}}} \geq \frac{1}{2}\sqrt{\frac{\gamma^0}{\sigma}}. \end{aligned}$$

Hence,

$$\tau^k \geq 1 + \frac{k}{2}\sqrt{\frac{\gamma^0}{\sigma}} \;\Rightarrow\; \lambda^k \leq \frac{4\sigma}{(2\sqrt{\sigma} + k\sqrt{\gamma^0})^2}.$$

□

The following corollary immediately follows from Theorem 4.3, Lemma 4.10, and the fact that $(F(x^0) - F(x^*) \leq \frac{L}{2}\|x^0 - x^*\|^2)$.

Corollary 4.5. *Suppose $\gamma^0 \geq l$ then the generated sequence by Algorithm 4.3 satisfies*

$$F(x^k) - F(x^*) \leq \frac{\gamma^0 + L}{2} \left(\frac{4\sigma}{(2\sqrt{\sigma} + k\sqrt{\gamma^0})^2} \right) \|x^0 - x^*\|^2.$$

Note that in IMRO-1D, $\sigma \geq L$. Under the assumption that $\sigma = \gamma^0 = L$, we have

$$F(x^k) - F(x^*) \leq \frac{4L}{(2+k)^2} \|x^0 - x^*\|^2.$$

Chapter 5

Computational Experiment on IMRO

In this chapter we test the performance of IMRO in practice. We compare speed and accuracy of IMRO with other available solvers summarized in the succeeding section.

5.1 Related Software

- **GPSR (Gradient Projection for Sparse Reconstruction)**[46] This gradient projection based algorithm, first reformulates BPDN problem (1.68) into a bound constrained quadratic program (BCQP) as

$$\begin{array}{ll} \min & \frac{1}{2}\|b - [A, -A]\, z\|^2 + \lambda \mathbf{1}^t z, \\ s.t. & z = \begin{bmatrix} u \\ v \end{bmatrix} \geq 0. \end{array}$$

  The BCQP formulation is then solved through a projected gradient technique. In other words

$$z^{k+1} = z^k + \lambda^k \left(\text{Proj}_+ \left(z^k - \alpha^k \nabla F(z^k) \right) - z^k \right),$$

  where λ^k and α^k are step sizes and Proj_+ is the projection on nonnegative orthant. Two variants of the algorithm have been proposed. In the basic variant $\lambda^k = 1$ and α^k is determined through a backtracking line search. The BB version finds $\lambda^k \in [0, 1]$ using the technique due to Barzilai and Borwein [6], then updates α^{k+1} accordingly. GPSR software is available at [44].

- **l1-ls** [69] l1-ls solves the same reformulation of BPDN problem as in GPSR through a truncated Newton interior point method. In l1-ls, preconditioned conjugate gradient (PCG) has been adopted for finding the search direction. Although forming the Newton system explicitly requires A^tA, the computational cost of each iteration of PCG is reduced to a matrix vector multiplication by the proper choice of preconditioner. The MATLAB code of this solver is available at [71].

- **FPC and FPC-AS** [56, 57] Recall that first order optimality conditions implies that x^* is the minimizer of a composite function (defined in (1.34)) if and only if

$$x^* = \text{Prox}_{\alpha g}\left(x^* - \alpha \nabla f(x^*)\right), \tag{5.1}$$

because

$$\begin{aligned} & 0 \in \nabla f(x^*) + \partial g(x^*), \\ \Leftrightarrow \quad & -\nabla f(x^*) \in \partial g(x^*), \\ \Leftrightarrow \quad & (x^* - \alpha \nabla f(x^*)) - x^* \in \alpha \partial g(x^*), \\ \Leftrightarrow \quad & x^* = \text{Prox}_{\alpha g}\left(x^* - \alpha \nabla f(x^*)\right). \end{aligned}$$

Equation (5.1) is called "fixed point equation". "Fixed Point Continuation" (FPC) aims to solve equation (5.1) through a proximal gradient method. The resulting algorithm has the following general scheme

$$x^{k+1} = \text{Prox}_{\alpha g}\left(x^k - \alpha \nabla f(x^k)\right). \tag{5.2}$$

The developed theory on the convergence of this method suggests that the algorithm converges faster for larger values of λ. "Continuation" strategy is essentially solving BPDN problem for a decreasing values of $\bar{\lambda} \to \lambda$ and warm starting the algorithm from the terminating solution corresponding to the previous value of $\bar{\lambda}$. FPC has later been extended to FPC-AS [107]. For each continuation interval, FPC-AS first solves the problem through FPC, then hard-theshold the solution for nonzero entries. The $\|x\|_1$ is replaced with $\text{sgn}(x)^t x$, and the smaller smooth problem is minimized to attain the final solution. FPC and FPC_AS software are available at [55, 106].

- **SpaRSA (Sparse Reconstruction by Seperable Approximation)** [110] SpaRSA is a proximal gradient framework for composite functions in which the nonsmooth part, $g(x)$, is separable. When $g(x) = \|x\|_1$ as is BPDN problem, SpaRSA reduces to an ISTA algorithm. The Barzilai and Borwein [6] and a continuation scheme has been applied to enhance the performance of the algorithm. The MATLAB code is available at [45].

- **NestA**[11] NestA is an algorithm built upon the Nesterov's accelerated technique for minimizing convex functions with Lipschitz continuous gradients over simple convex sets [85]. NestA is tailored for solving problem BP_ϵ, i.e., (1.67), with orthonormal matrix A. A continuation scheme has been adapted to improve the performance of NestA. The MATLAB package for NestA might be reached at [10].

- **SPGL1** [18] This method solves BP_ϵ formulation, (1.67), through solving a sequence of LASSO problems, (1.69). Each LASSO problem is solved using a spectral projected gradient method [23]. In this technique, a single parameter function for LASSO problem is defined as $\phi(\tau) = \|Ax_\tau^* - b\|$. Using the dual information of the LASSO problem, one may recover derivative of ϕ; hence Newton method is applied to find the root of $\phi(\tau) = \epsilon$, where ϵ is the parameter of BP_ϵ problem. At termination the solution of LASSO_{τ^*} coincidences with the solution of BP_ϵ. The MATLAB package for SPGL1 is available at [16].

- **TwIST**[21] Recall that ISTA algorithm as presented in Section 1.3 had the general form of $\mathcal{S}_{\lambda\alpha}\left(x^k - \alpha\nabla f(x^k)\right)$ or $(1-\beta)x^k + \beta\mathcal{S}_{\lambda\alpha}\left(x^k - \alpha\nabla f(x^k)\right)$ for $\beta = 1$. TwIST is a two step ISTA technique in which each iterate depends on the last two previous iterates. The general form of TwIST is $(1-\alpha)x^{k-1} + (\alpha-\beta)x^k + \beta\mathcal{S}_{\lambda\alpha}\left(x^k - \alpha\nabla f(x^k)\right)$. Note that α and β are not necessarily smaller than 1. For our experiment we have used the default setting on these parameters. The MATLAB package of this algorithm is available at [20].

- **FISTA**[9] We presented FISTA in Section 1.3. The algorithm has been proposed by Beck and Teboulle in [9] and by Nesterov in [84]. The algorithm is part of the TFOCS package [13] that is available at [12].

5.2 Numerical Results

The termination criterion used for IMRO is the measurement on the norm of subgradient of function $F(x)$, ξ, which at iteration k is

$$\begin{aligned}
\xi_i &= \lambda + \nabla f_i^k \qquad \text{if } x_i > 0,\\
\xi_i &= -\lambda + \nabla f_i^k \qquad \text{if } x_i < 0,\\
\xi_i &= -\lambda\alpha + \nabla f_i^k \qquad \text{if } x_i = 0 \ \text{ for some } \ \alpha \in [-1, 1].
\end{aligned}$$

Note that for zero entries, if $|\nabla f_i^k| \leq \lambda$, then $\exists \alpha$ such that $\xi_i = 0$. In case that $|\nabla f_i^k| > \lambda$, then $\xi_i \neq 0$; so we take α such that ξ_i is minimized, i.e., $\xi_i = |\nabla f_i^k| - \lambda$. Therefore, The norm of subgradient at x^k is easily calculated with no extra computational cost.

In both variants of IMRO, we took x^{k+1} as the minimizer of the approximation model, i.e., no line search has been employed. Our experiment (not reported here) with IMRO coupled with a bisection line search suggested that there is no significant advantage in applying the line search. Other techniques such as Barzilai and Borwein [6], however, might be advantageous and needs further investigation. One of the key questions regarding the implementation of IMRO-1D is what direction v should be used. In our experiment, we have used the previous direction for v, i.e., $v^k = x^k - x^{k-1}$. The choice of v, however, needs further study. Our measurement on the computational cost of each algorithm is the number of matrix-vector multiplications, i.e., the number of calls to A or A^t.

As mentioned earlier, NestA only runs on instance with orthonormal A. In the first part of our experiment we compare IMRO with NestA. For increasing the accuracy of IMRO, we can simply use a smaller tolerance on the norm of subgradient. For NestA, parameter μ controls the accuracy of the solution; the smaller parameter μ gets, the more accurate the solution becomes. We have used the default value of $\mu = 0.02$ suggested by the authors, and $\mu = 0.002$ in our experiment. All other settings are set to their default value. The number of continuation steps in the default setting is five.

The information on test cases with orthonormal A is summarized in Table 5.1. These instances are generated by L1TestPack package [78]. "Ent. type" stands for the type of entries in matrix A or vector x^*, the optimal solution of the problem. "Gaussian" entries are generated using MATLAB's "randn" function and "dynamic 3" gives random entries with a dynamic range of 10^3. In other words, when entry type of x^* is "dynamic 3", each nonzero entry of x^* is generated as

$$\text{sgn}(r_n)10^{3r},$$

where r_n is a normally distributed (Gaussian) random number and r is a uniformly distributed random number in $(0, 1)$. The dynamic range typically varies between zero to five. It is often believed that instances with higher dynamic range are harder to solve.

In Table 5.2, we have presented the number of A or A^t calls each algorithm takes to converge. We also include the residual of the solution at termination for each algorithm (i.e., $\|x_t - x^*\|$); this quantity is depicted in Figure 5.1 as well. We can see from the graph that in IMRO, especially IMRO-2D, the decrease of the residual is significantly sharper than NestA.

	m	n	Ent. type of A	Ent. type of x	λ
Ins 1	2500	10000	Gaussian	Gaussian	0.5
Ins 2	2500	10000	Gaussian	Gaussian	0.05
Ins 3	2500	10000	Gaussian	dynamic 3	0.5
Ins 4	2500	10000	Gaussian	dynamic 3	0.1

Table 5.1: Test cases with orthonormal A

		IMRO-2D		IMRO-1D		NestA		
	tol	it.	$\|x_t - x^*\|$	it.	$\|x_t - x^*\|$	it.	$\|x_t - x^*\|$	μ
Ins 1	1e-2	51	0.047	54	0.057	684	1.539	0.02
	1e-6	138	7.119e-6	214	9.102e-06	1010	0.474	0.002
Ins 2	1e-2	60	0.048	60	0.058	424	0.173	0.02
	1e-6	120	6.755e-6	220	8.759e-06	572	0.026	0.002
Ins 3	1e-2	198	0.051	208	0.061	532	1.488	0.02
	1e-6	267	7.169e-6	347	7.515e-06	580	0.182	0.002
Ins 4	1e-2	393	0.055	630	0.06	504	1.487	0.02
	1e-6	474	7.194e-06	796	1.160e-05	552	0.646	0.002

Table 5.2: Numerical results (number of A/A^t calls) for comparison of IMRO and NestA

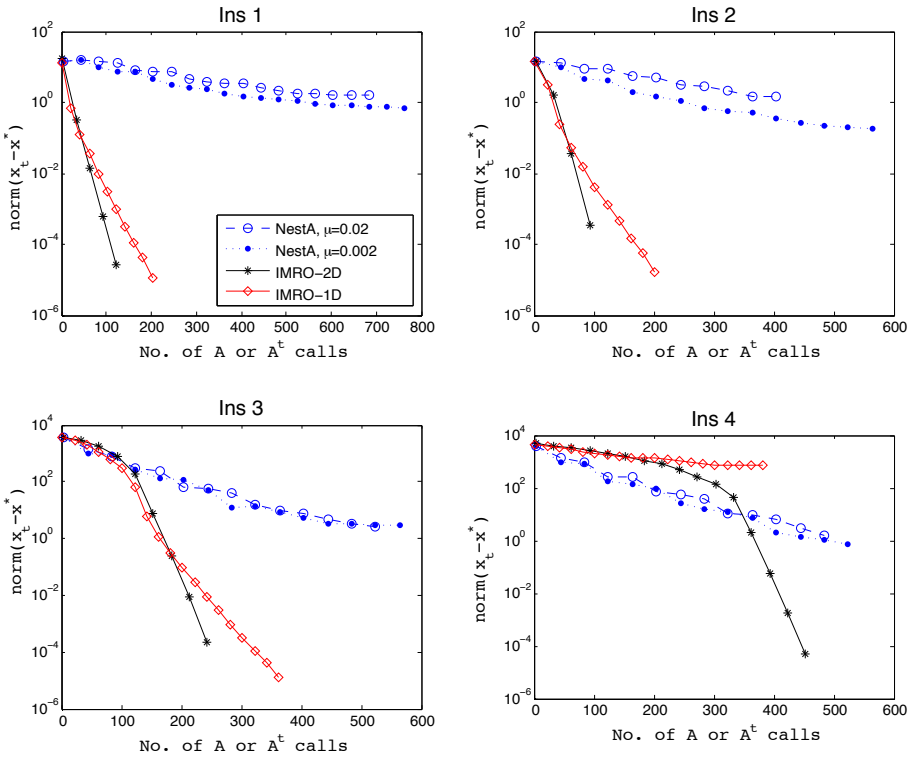


Figure 5.1: Accuracy of solution in IMRO and NestA

The comparison of IMRO and solvers other than NestA has been separated into two phases. The first one is testing available software for solving BPDN formulation on L1TestPack generated instance. The second one is testing IMRO and couple of other state-of-the-art solvers in sparse signal recovery on a number of examples from Sparco [19]. Table 5.3 summarizes the information on the test cases used for our computational experiment. Before presenting the numerical results, we would like to elaborate few details on the data structure of Sparco problems. In Sparco problems, the operator A has the format of

$$A = MB, \tag{5.3}$$

where M is the measurement matrix and B is the sparsity basis. The sparsity basis enables the original signal, y, to accept a sparse representation, i.e.,

$$y = Bx, \tag{5.4}$$

has a solution x^* with only few nonzero entries. The measurement matrix describes the sampling procedure. Recall that in compressive sensing (sampling), we desire to construct the signal through a relatively few measurements. Therefore the problem we would like to solve is essentially finding a sparse x such that

$$b := My = Ax. \tag{5.5}$$

After finding the sparse solution of (5.5), we can reconstruct the original signal as

$$y = Bx^*. \tag{5.6}$$

One may refer to [19] for more information on Sparco package and specifics of each problem.

When comparing BPDN solvers, we run IMRO-2D for a given tolerance to get $\tilde{x}$; and the other solvers terminate when their solution $\hat{x}$ satisfies

$$\frac{1}{2}\|A\hat{x} - b\|^2 + \lambda\|\hat{x}\|_1 \leq \frac{1}{2}\|A\tilde{x} - b\|^2 + \lambda\|\tilde{x}\|_1. \tag{5.7}$$

		cond(A)	Ent. type of A	Ent. type of x	λ
L1TestPack $A \in \mathbb{R}^{2500\times10000}$	Ins 1	$O(1)$	Gaussian	Gaussian	0.5
	Ins 2	$O(1)$	Gaussian	dynamic 3	0.5
	Ins 3	$O(1)$	Gaussian	Gaussian	0.1
	Ins 4	$O(1)$	Gaussian	dynamic 3	0.1
	Ins 5	$O(10^3)$	Gaussian	Gaussian	0.1
	Ins 6	$O(10^3)$	Gaussian	dynamic 3	0.1
	ID	m	n	Operator	λ
Sparco	5	300	2048	Gaussian, DCT[1]	0.1
	9	128	128	Heaviside	0.1
	10	1024	1024	Heaviside	0.1
	11	256	1024	Gaussian	0.1
	401	29166	57344	Windowed (DCT)	0.1
	402	29166	86016	Windowed (DCT)	0.1

1: Discrete Cosine Transform

Table 5.3: Information on test cases

The number of calls to A or A^t, and the residual of the solution (i.e., $\|x_t - x^*\|$) are presented in Tables 5.4 and 5.5, respectively. In almost all cases IMRO-2D outperforms other algorithms, especially when a highly accurate solution is desired. The performance of some of the techniques like GPSR or SpaRSA has been affected considerably by the increase in the dynamic range. Moreover, some of the solvers failed to converge when A is ill-conditioned, or failed to reach high accuracy as in FPC-AS. IMRO-2D on the other hand, performs consistently well in all our test cases. Graph 5.2 visualizes the residual of the solution with respect to the number of A or A^t calls.

	tol	IMRO-2D	IMRO-1D	FISTA	GPSR-BB	SpaRSA	TwIST	FPC-AS	L1ls(pcg)
Ins 1	1e-2	37	52	32	24	24	60	23	368
	1e-6	103	232	106	64	48	153	DNRT†	DNC‡
Ins 2	1e-2	175	190	168	230	180	187	142	1311
	1e-6	253	392	258	264	203	277	DNRT	DNC
Ins 3	1e-2	52	54	36	26	30	53	25	504
	1e-6	136	242	118	56	53	136	DNRT	DNC
Ins 4	1e-2	322	416	382	1208	880	313	160	2201
	1e-6	394	616	480	1242	904	387	DNRT	DNC
Ins 5	1e-2	136	DNC	225	380	DNC	356	DNC	502
	1e-6	277	DNC	997	884	DNC	883	DNC	DNC
Ins 6	1e-2	202	540	399	820	DNC	554	DNRT	2550
	1e-6	538	1292	1741	1416	DNC	1033	DNRT	DNC

† Did Not Reach the Tolerance (DNRT): The solver terminates before reaching the desired tolerance.

‡ Did Not Converge (DNC): The solver reaches the maximum iteration count of $5 \times k_{2D}$, where k_{2D} is the number of iterations IMRO-2D takes to converge.

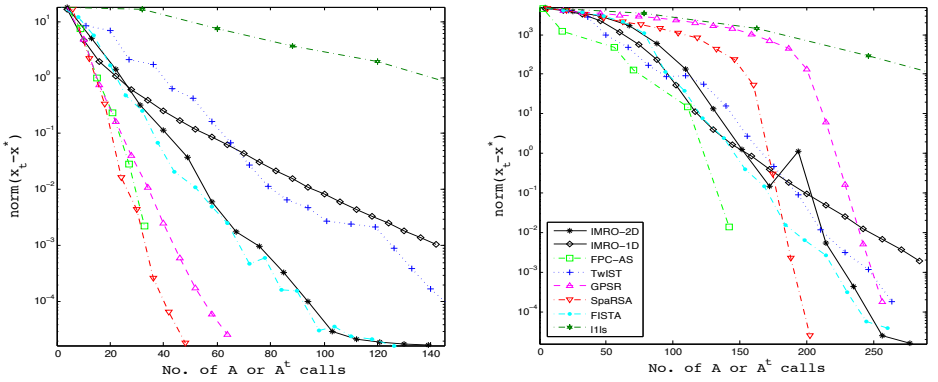
Table 5.4: Numerical results (number of A/A^t calls) for BPDN solvers

	tol	IMRO-2D	IMRO-1D	FISTA	GPSR-BB	SpaRSA	TwIST	FPC-AS	L1ls(pcg)
Ins 1	1e-2	1.4e-01	1.2e-01	2.5e-01	1.1e-01	1.6e-02	1.3e-01	5.7e-02	1e-02
	1e-6	2.9e-05	2e-05	2.2e-05	2.7e-05	1.8e-05	4.5e-05	2.3e-03	1.9e-05
Ins 2	1e-2	1.1e-01	1.5e-01	1.5e-01	1.1e-01	3.7e-02	1e-01	1.4e-02	7.3e-03
	1e-6	2.6e-05	2.5e-05	3.7e-05	4.4e-05	2.5e-05	6.3e-05	1.3e-04	2.9e-03
Ins 3	1e-2	8.8e-02	1.3e-01	1.2e-01	8.4e-02	2.2e-02	8.5e-02	3.9e-02	5e-03
	1e-6	2.3e-05	2.5e-05	3.1e-05	3.2e-05	2.1e-05	3e-05	4.4e-03	1.8e-05
Ins 4	1e-2	9.3e-02	1.6e-01	1.2e-01	1.2e-01	2.7e-02	8.1e-02	3.9e-02	7.6e-03
	1e-6	3.4e-05	2.1e-05	4.2e-05	5.2e-05	2.6e-05	1.8e-04	1.4e-04	2.7e-05
Ins 5	1e-2	1e-01	1e-01	1e-01	1.1e-01	9.2e+00	8.8e-02	1.7e+03	5.9e-03
	1e-6	2.4e-05	2.2e-05	4.3e-05	2.7e-05	9.2e+00	2.6e-05	1.7e+03	1.7e-05
Ins 6	1e-2	1e+01	7.6e+00	9.8e+00	1e+01	1.8e+03	1e+01	6.4e+03	5.2e-01
	1e-6	2.2e-03	1.9e-03	1.8e-03	2.2e-03	1.8e+03	2.1e-03	6.4e+03	1.5e-03

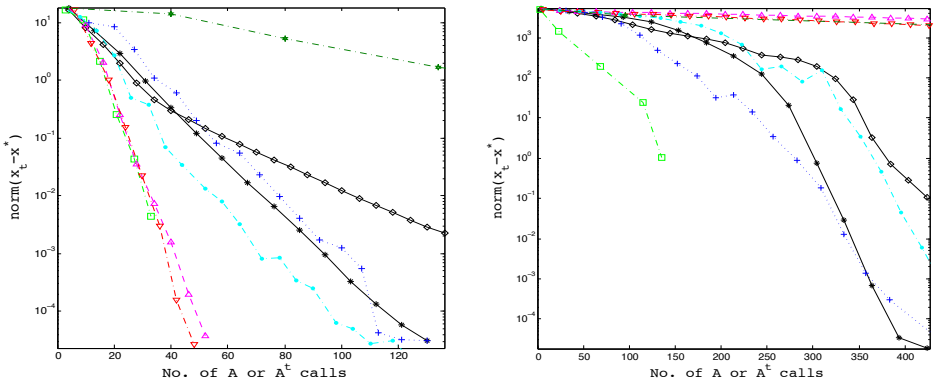
† For instance with DNRT or DNC in Table 5.4, we report the error of the final iterate.

Table 5.5: Accuracy of the solution, i.e. $\|x_t - x^*\|$, in IMRO and other BPDN solvers†

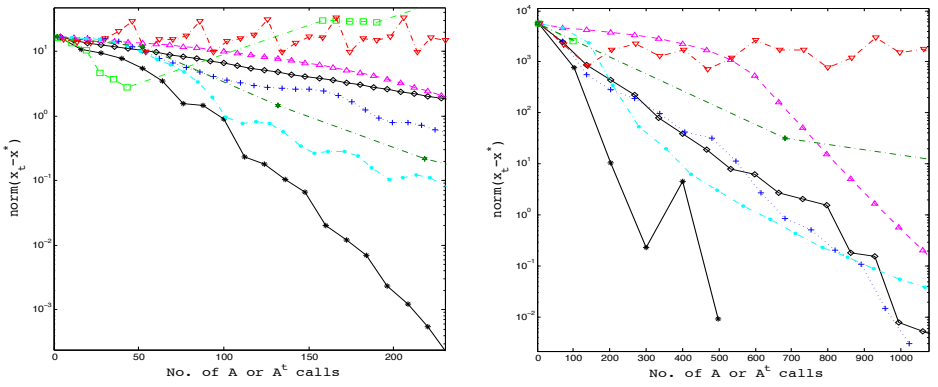
When testing the algorithms on Sparco test cases, we run all the solvers to a fixed number of iterations, and plot the signal at termination. We have compared IMRO-2D with several other state-of-the-art techniques in sparse signal recovery. Figures 5.3 to 5.9 demonstrate the results. In figures 5.3 to 5.7, the original signal (i.e., Bx^* by (5.6)) is plotted in blue and the recovered signal (i.e., Bx_t) is in red. Figures 5.8 and 5.9 are on Sparco(401) and Sparco(402). The signals in these two problems are mixed audio signals. Hence, for these two instances we plot both original and the recovered signals in each subplot. The plotted signal on the top of each subplot is the original signal (i.e. Bx^*), while the one in the bottom is the recovered signal (i.e., Bx_t). An asterisk next to the name of a solver indicates that the solver had premature termination, or found the optimum before the assigned iteration count. A double asterisk, stands for cases that the solver did not converge (terminate) in 10000 iterations. obviously, IMRO-2D has a stable performance and has almost perfect recovery in all test cases. The difference between IMRO-2D and other solvers is more noticeable in test cases Sparco(9) and Sparco(10) (figures 5.4, 5.5 and 5.6). Note that for Sparco(10), the signal recovered by IMRO-2D after 300 iterations outperforms the signal recovered by other solvers at iteration 1000.



(a) Ins 1 (b) Ins 2



(c) Ins 3 (d) Ins 4



(e) Ins 5 (f) Ins 6

Figure 5.2: Accuracy of the solution for BPDN solvers

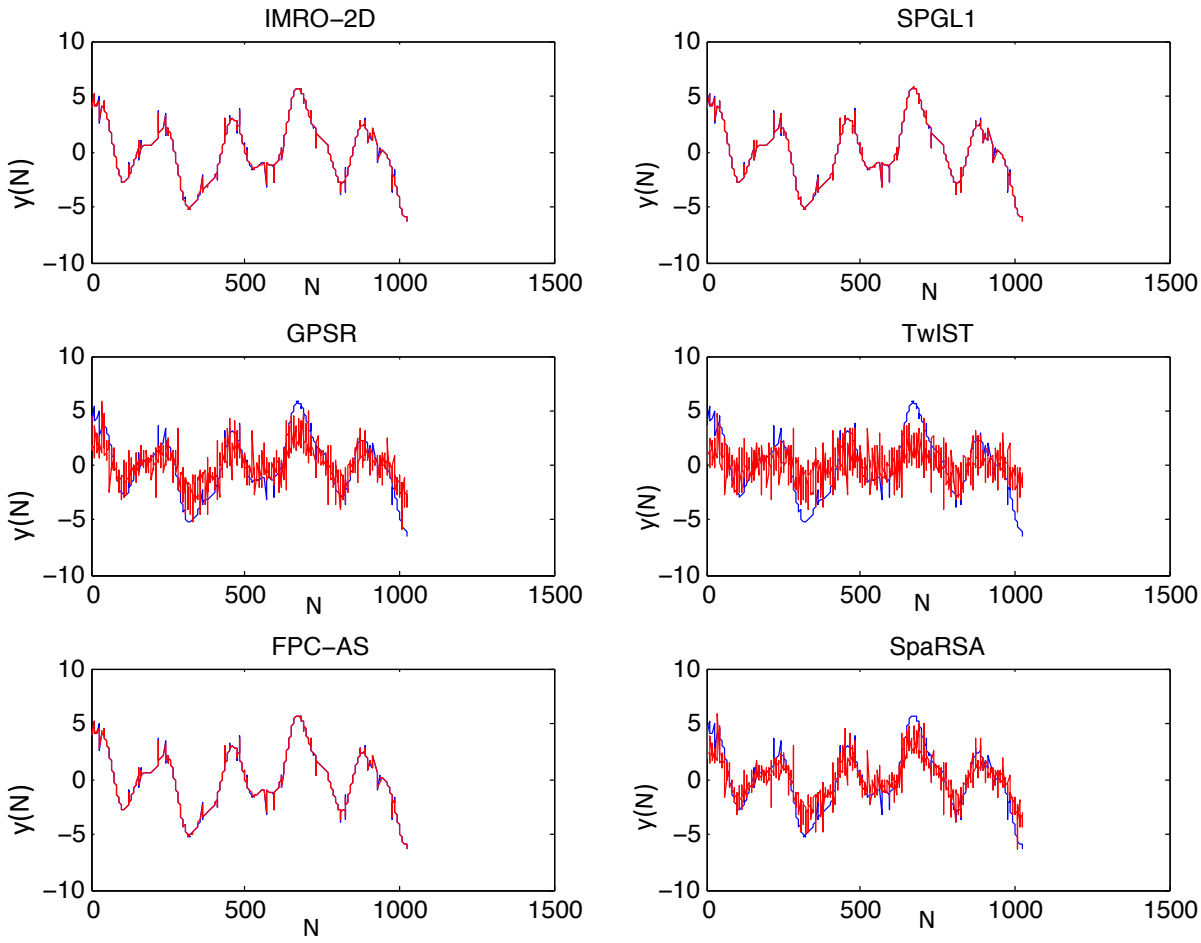


Figure 5.3: Recovered signal after 100 iteration for Sparco(5)

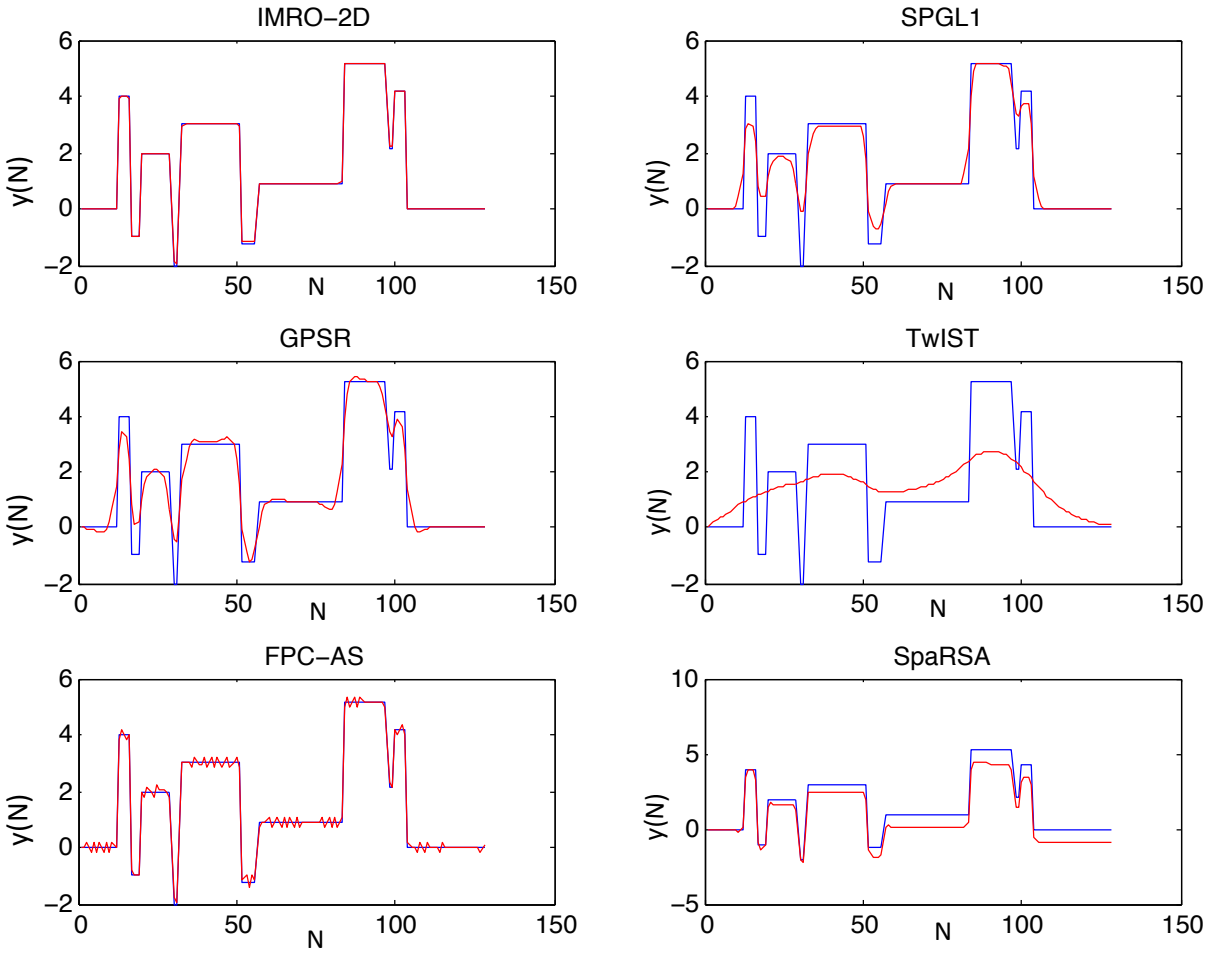


Figure 5.4: Recovered signal after 300 iteration for Sparco(9)

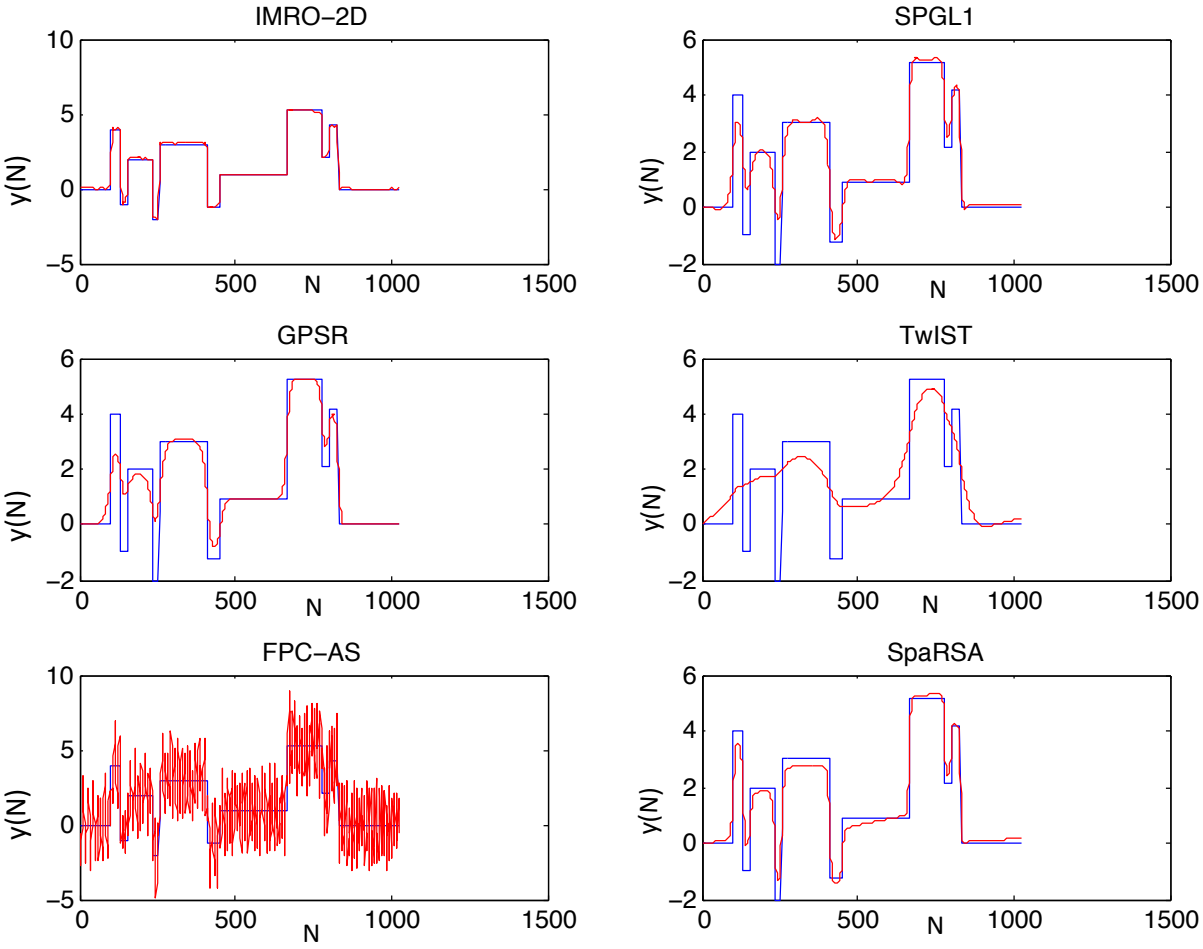


Figure 5.5: Recovered signal after 300 iteration for Sparco(10)

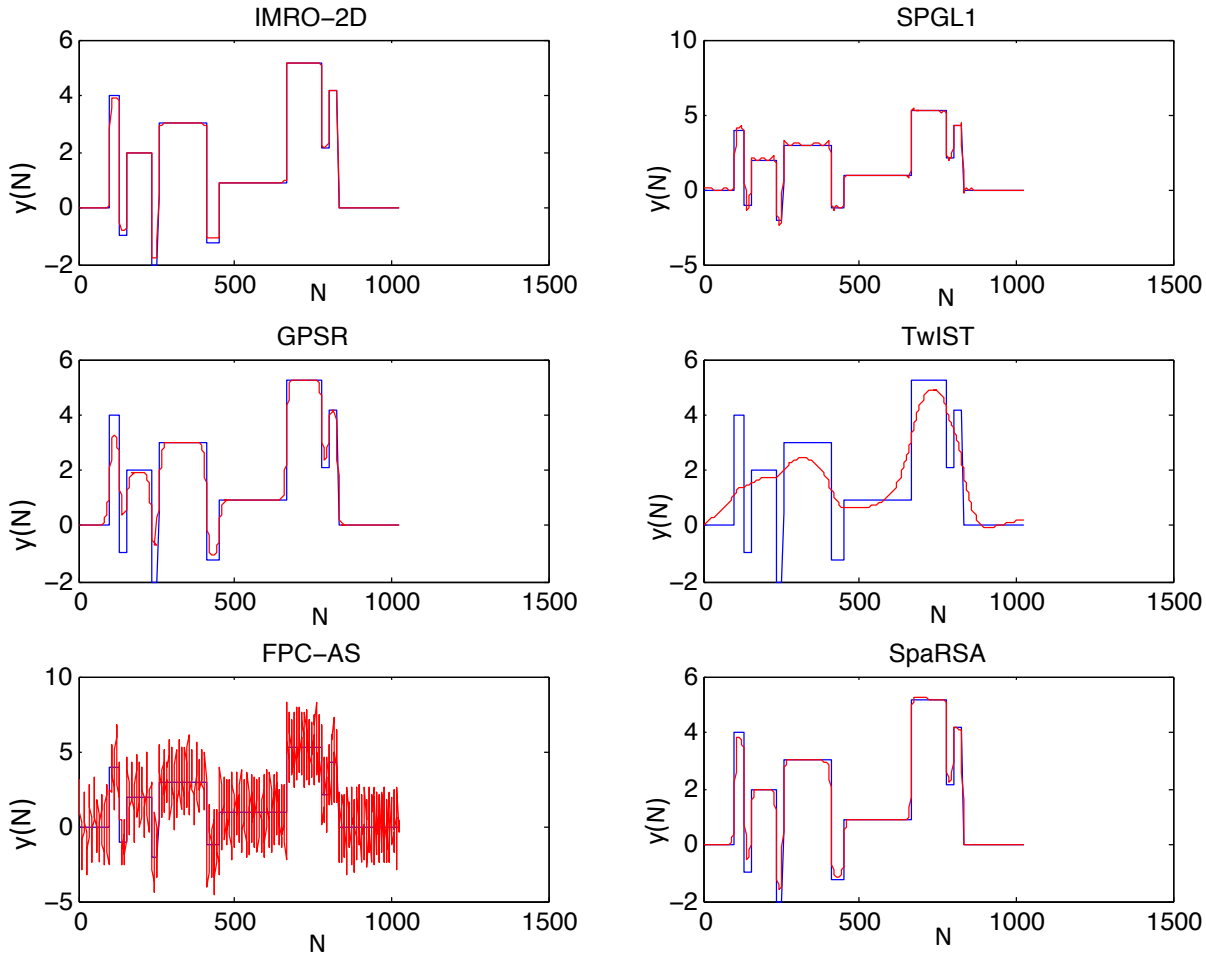


Figure 5.6: Recovered signal after 1000 iteration for Sparco(10)

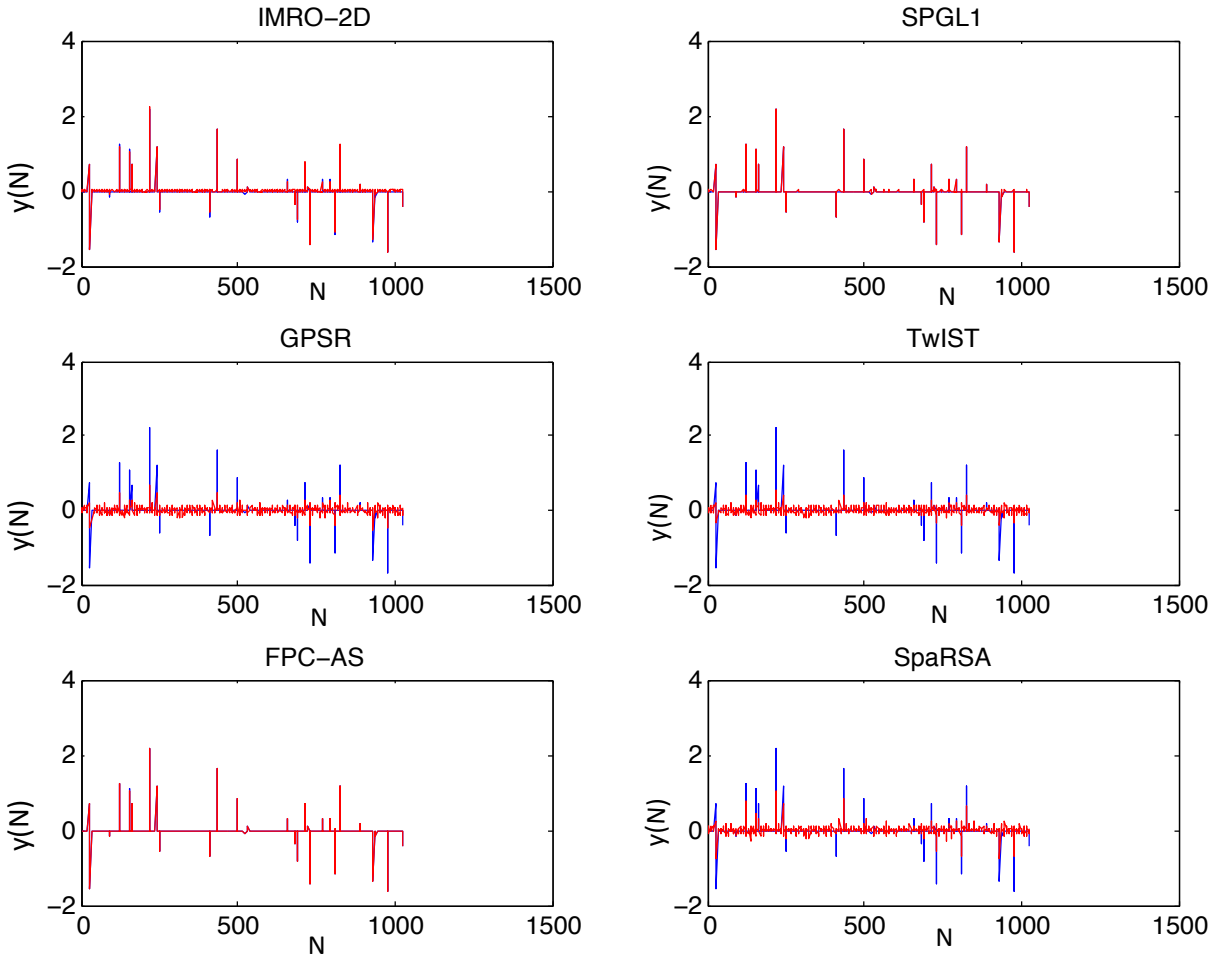


Figure 5.7: Recovered signal after 200 iteration for Sparco(11)

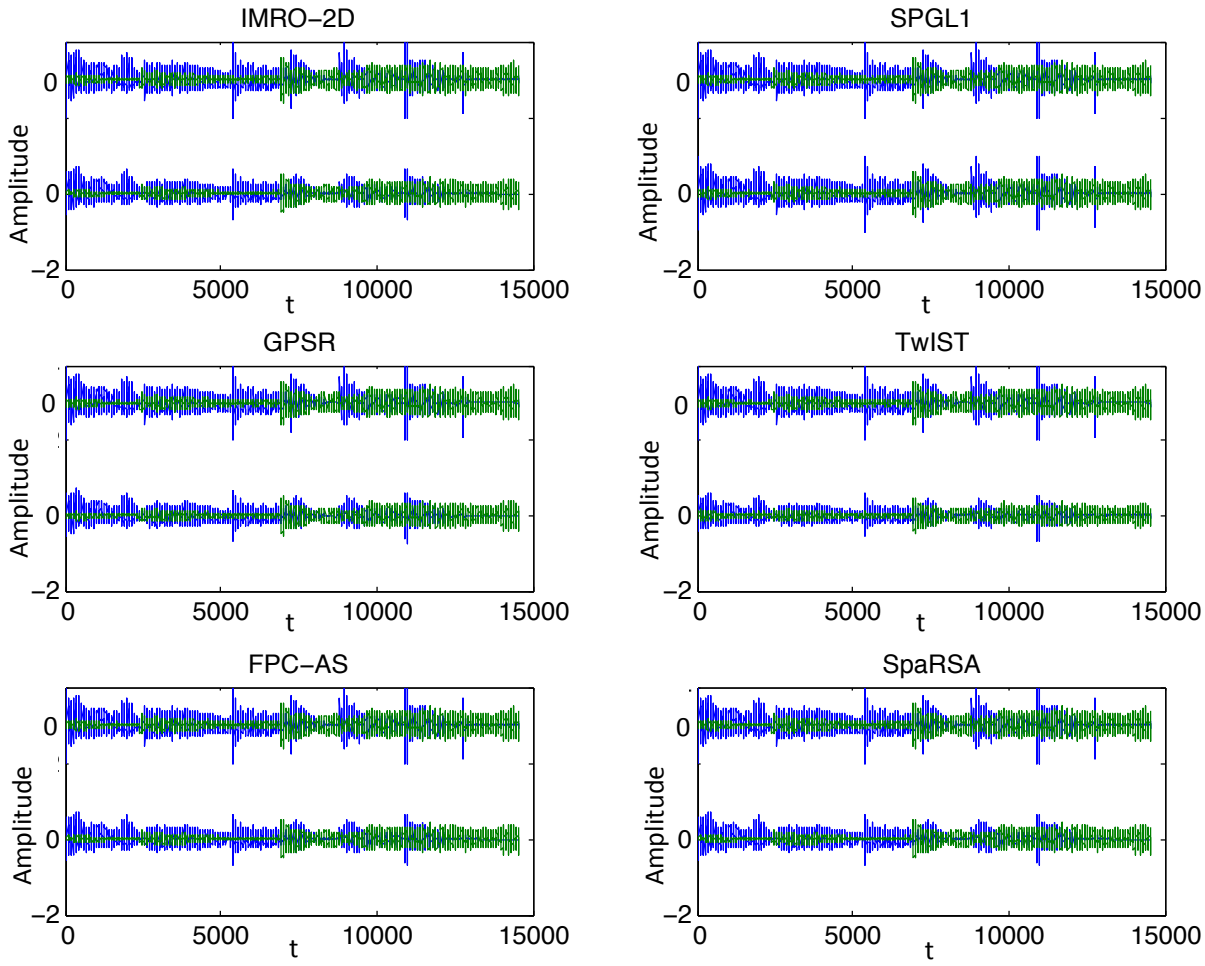


Figure 5.8: Recovered signal after 300 iteration for Sparco(401)

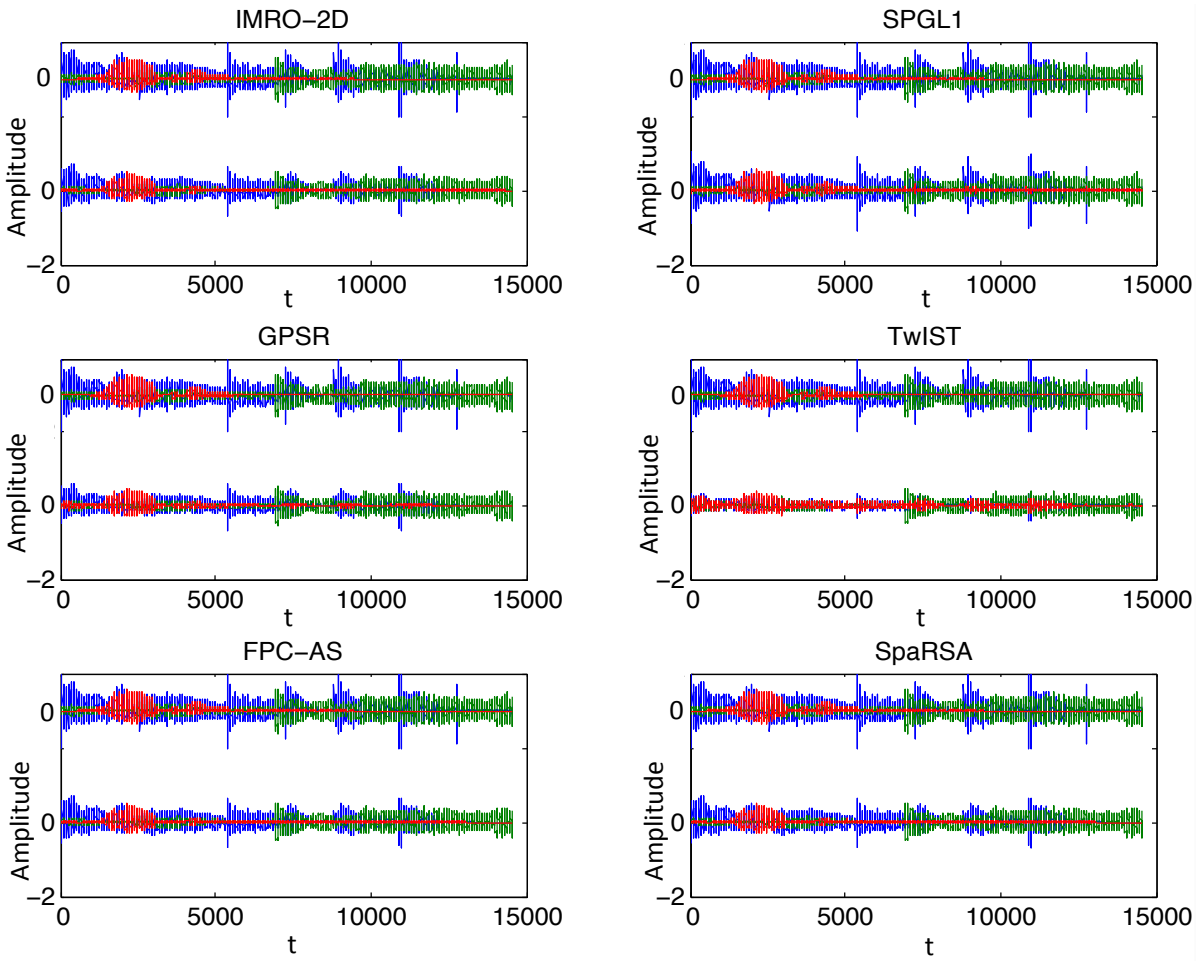


Figure 5.9: Recovered signal after 500 iteration for Sparco(402)

Chapter 6

Conclusion and Future Work

The low computational cost of first-order methods makes this class of algorithms a popular choice and an efficient technique for solving large-scale problems. These techniques, however, may suffer from slow rates of convergence. In particular, the conjugate gradient method is a practical technique that sometimes encounters slow convergence due to the loss of independence of search directions. The question of whether CG can achieve the optimal complexity bound for the class of strongly convex functions has remained open for decades. In this thesis, we partially answer this question, namely we derive conditions under which CG achieves the complexity bound proved to be optimal for the class of strongly convex functions. These conditions almost always hold for our proposed algorithm, CGSO. We also suggest a firm scheme for detecting and correcting the loss orthogonality of gradients in other variants of CG. The analysis of CGSO has also been extended to linearly constrained problems in Appendix A. Our experiment suggests that CGSO is a stable technique and can often outperform other methods, especially on ill-conditioned problems.

Another algorithm proposed in this thesis is IMRO, a proximal quasi-Newton method for minimizing composite functions. The theory of IMRO implies that choosing a suitable matrix H in proximal quasi-Newton methods enables us to incorporate more information about the function in the approximation quadratic model, making IMRO superior to its predecessor techniques. IMRO is proposed and applied to BPDN problem in this thesis. Two variants of this algorithm have been proposed, IMRO-1D and IMRO-2D. Our numerical experiment indicates that IMRO is very effective in solving BPDN, and outperforms other state-of-the-art solvers in most cases. One of the advantages of IMRO, especially IMRO-2D, is that it can attain a highly accurate solution. This important property is quite crucial in retrieving an almost sparse signal, i.e. a signal with large number of entries

close to zero. Moreover IMRO has shown to be more stable than other techniques, and less sensitive to parameters of the problem.

This thesis has raised many questions, some remained unanswered:

- The analysis of CGSO relies on two conditions. These conditions must be checked every m iteration, where m is a constant dependent on parameters of the function. Satisfaction of these conditions concludes the optimal complexity bound for the class of functions $f \in \mathbf{C}_{l,L}$. Since we normally do not know the parameters of the function, we check these conditions for all possible choices of m. In our experiment, these conditions are always satisfied except in one instance. This particular instance, however, is ill-conditioned and the required inequalities failed for the smallest estimate on m. This, in fact, suggests that the smallest pick for m might have been an underestimate, promoting the idea that there might exists a variant of conjugate gradient method that attains the optimal complexity bound.

- Checking the conditions derived in the analysis of CGSO has been applied to the traditional variants of CG as a technique for detection and correction of the loss of independence of gradients. As reported, the performance of CG equipped with the detection and correction procedure is similar for all the tested variants of CG. It is still unclear as to why the three variants of CG in our experiment have almost the same behaviour after the detection and correction procedure, while their performance is significantly different prior to it.

- We also proposed a hybrid technique for combining Nesterov's algorithm and conjugate gradient method. Two variants of this method is proposed, one is substituting iterates of CG for x in Nesterov's method, and the other replaces y in Nesterov's method by the iterates of CG. The proposed methods need further computational experiments. Our preliminary experiment shows promising result for the smoothed compressive sensing problem. It is, however, unclear as to which hybrid technique performs better in general. The question of how this hybrid technique compares with CGSO or other variants of nonlinear CG has also remained unanswered.

- As described, in IMRO-1D the quadratic model matches the function on direction v. A key question to answer is which choice of direction v is preferred. An extensive study is required on different choices of v. It might also be helpful to study whether combining different directions, say the previous gradient and the previous direction, can enhance the performance of the algorithm. FIMRO (the accelerated variant of IMRO) needs to be extensively tested and compared to IMRO. All the proposed

techniques in Chapter 4 can potentially be applied to other suitable test cases. This is also another venue for future work.

Appendix A

CGSO for Convex Problems with Linear Constraints

We extend CGSO to solving the problem

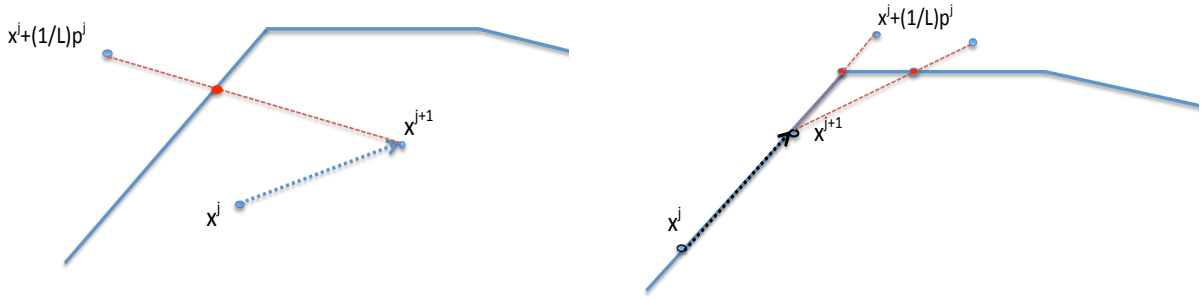
$$\min_{x\in\mathcal{P}} f(x), \tag{A.1}$$

where $f(x) \in \mathbf{C}_{l,L}$ and $\mathcal{P}$ is a polyhedron. We first present the algorithm in the next section. Then we apply this method to l_1-regularized least square problem in Section A.2. The numerical results are presented in Section A.3.

A.1 The Algorithm

Let p^j denote the projection of the steepest descent direction of f at x^j on $\mathcal{P}$, i.e., $p^j = \nabla_{\mathcal{P}} f^j$ which is defined as $\mathrm{Proj}_{T_{\mathcal{P}}(x^j)}(-\nabla f^j)$ according to Definition 1.2. We refer to p^j as the projected gradient throughout this appendix. Substituting p^j for $\nabla f(x^j)$, we derive the following natural extension of CGSO:

Algorithm A.1.
$x^0 = \textit{arbitrary}$;
for $j = 1, 2, \ldots$
$x^{j+1} = x^j + \alpha^j p^j + \beta^j d^j$
$\textit{where}\ \ d^j = x^j - x^{j-1}\ \textit{and}$
$\alpha^j, \beta^j = \arg\min_{\alpha,\beta} f(x^j + \alpha p^j + \beta d^j)\ \textit{s.t.}\ \ x^j + \alpha p^j + \beta d^j \in \mathcal{P}$



(a) Case 1 (b) Case 2

Figure A.1: Illustration of the sufficient reduction in objective value when $\mathcal{A}(x^{j+1}) = \mathcal{A}(x^j)$

Let $\mathcal{A}(x^j)$ stand for the set of active constraints at x^j.

Claim A.1. *In each iteration of the Algorithm A.1, either* $f(x^{j+1}) \leq f\left(x^j + \frac{1}{L}p^j\right)$ *or* $\mathcal{A}(x^{j+1}) \neq \mathcal{A}(x^j)$.

Proof. Let $\mathcal{H}^j = x^j + \text{Span}\{p^j, d^j\}$. Then the feasible region of the two dimensional optimization subproblem in each iteration is $\mathcal{P}^j = \mathcal{H}^j \cap \mathcal{P}$; hence at most a two-dimensional polyhedron.

If $x^j + \frac{1}{L}p^j \in \mathcal{P}^j$, or $\mathcal{A}(x^j) \neq \mathcal{A}(x^{j+1})$, then the claim holds. Suppose $x^j + \frac{1}{L}p^j \notin \mathcal{P}^j$, and $\mathcal{A}(x^j) = \mathcal{A}(x^{j+1})$. Two cases may occur; either x^j and x^{j+1} are both in the $\text{int}(\mathcal{P}^j)$ as in Figure A.1(a), or in the relative interior of a one dimensional face of $\mathcal{P}^j$ which we refer to as $\mathcal{F}$ as demonstrated in Figure A.1(b).

In both cases the line segment connecting x^{j+1} and $x^j + \frac{1}{L}p^j$ intersects $\mathcal{P}^j$ at the point $\bar{x} = \lambda x^{j+1} + (1-\lambda)\left(x^j + \frac{1}{L}p^j\right)$ for some $\lambda > 0$. This is true in the first case by the fact that $x^{j+1} \in \text{int}(\mathcal{P}^j)$. In the second case, it follows from the fact that $p^j \in T_{\mathcal{P}}(x^j)$ and so $x^j + \frac{1}{L}p^j \in \mathcal{S}$ where $\mathcal{S}$ is the half-space containing $\mathcal{P}^j$ and supported by $\mathcal{F}$.

Suppose by contradiction that $f(x^j + \frac{1}{L}p^j) < f(x^{j+1})$. By convexity of the function we have,

$$f(\bar{x}) \leq \lambda f(x^{j+1}) + (1-\lambda) f(x^j + \frac{1}{L}p^j) < f(x^{j+1}),$$

which contradicts the optimality of x^{j+1} on $\mathcal{P}^j$. □

Lemma A.1. *Let* p^j *be the projected gradient of* f. *Then* $\langle p^j, -\nabla f(x^j)\rangle = \|p^j\|^2$

Proof. By definition, $p^j = \text{Proj}_{T_{\mathcal{P}}(x^j)}(-\nabla f^j)$. $T_{\mathcal{P}}(x^j)$ is a closed convex cone, and its polar is $N_{\mathcal{P}}(x^j)$[60, Proposition 5.2.4]. Let $o^j = \text{Proj}_{N_{\mathcal{P}}(x^j)}(-\nabla f^j)$. By Moreau's decomposition theorem, Theorem 1.1, $-\nabla f^j = p^j + o^j$, and $\langle p^j, o^j \rangle = 0$. Thus

$$\langle p^j, -\nabla f^j \rangle = \langle p^j, p^j + o^j \rangle = \|p^j\|^2.$$

□

We are now ready to list the properties of Algorithm A.1.

1. If $\mathcal{A}(x^j) = \mathcal{A}(x^{j+1})$, then $f(x^{j+1}) \leq f(x^j) - \frac{1}{2L} \parallel p^j \parallel^2$.

   Proof. By Claim A.1 we have

   $$f(x^{j+1}) \leq f\left(x^j + \frac{1}{L}p^j\right).$$

   Since $f \in \mathbf{C}_L$ we get

   $$f\left(x^j + \frac{1}{L}p^j\right) \leq f(x^j) - \langle \frac{1}{L}p^j, -\nabla f(x^j) \rangle + \frac{L}{2}\|\frac{1}{L}p^j\|^2 = f(x^j) - \frac{1}{2L} \parallel p^j \parallel^2,$$

   where the last equation holds by Lemma A.1, i.e., $\langle p^j, -\nabla f(x^j) \rangle = \|p^j\|^2$. □

2. $\langle -p^j, x^* - x^j \rangle \leq f(x^*) - f(x^j)$.

   Proof. Note that $x^* - x^j \in T_{\mathcal{P}}(x^j)$; and recall that by definition, the projected gradient is

   $$p^j = \arg\min \left\{ \|v + \nabla f(x^j)\| : v \in T_{\mathcal{P}}(x^j) \right\}. \tag{A.2}$$

   It is easy to observe that derivative of $\|v + \nabla f(x^j)\|$ is $\frac{\partial \|v+\nabla f(x)\|}{\partial v} = \frac{v+\nabla f(x)}{\|v+\nabla f(x)\|}$. If $-\nabla f(x^j) \in T_{\mathcal{P}}(x^j)$, then $p^j = -\nabla f(x^j)$ and the inequality holds by the convexity of f. When $-\nabla f(x^j) \notin T_{\mathcal{P}}(x^j)$, by optimality condition for (A.2) we get

   $$\langle p^j + \nabla f(x^j), (x^* - x^j) - p^j \rangle \geq 0.$$

   Therefore

   $$\langle p^j + \nabla f(x^j), x^* - x^j \rangle \geq \|p^j\|^2 - \langle -\nabla f(x^j), p^j \rangle = 0. \tag{A.3}$$

Again the last equation follows from Lemma A.1. By inequality (A.3) we get

$$\langle \nabla f(x^j), x^* - x^j \rangle \geq \langle -p^j, x^* - x^j \rangle,$$

and by convexity of the function we have

$$\langle \nabla f(x^j), x^* - x^j \rangle \leq f(x^*) - f(x^j).$$

This concludes the property we wanted to show. □

3. $\nu_f(x^0) = f(x^0) - f(x^*) \geq \frac{l}{2}\|x^* - x^0\|^2$.

Proof. By strong convexity of f we have,

$$f(x^0) \geq f(x^*) + \langle \nabla f(x^*), x^0 - x^* \rangle + \frac{l}{2}\|x^0 - x^*\|^2.$$

Using the first order optimality condition (i.e., $-\nabla f(x^*) \in N_{\mathcal{P}}(x^*)$) we get

$$\langle \nabla f(x^*), x^0 - x^* \rangle \geq 0,$$

hence

$$f(x^0) \geq f(x^*) + \frac{l}{2}\|x^0 - x^*\|.$$

□

The above properties essentially are the same as those we had for CGSO. In the following lemma, we extend Lemma 2.1 to constrained problems.

Lemma A.2. *Suppose $\kappa^{\mathcal{A}}$ represents the set of iterations in which active set changes; i.e., j's for which $\mathcal{A}(x^j) \neq \mathcal{A}(x^{j+1})$. Let $m \geq \left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil + \left|\kappa^{\mathcal{A}}\right|$ and suppose*

$$\frac{(f(x^{m-1}) - f(x^0))}{4} \left(\sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right) + \sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \lambda^j \left\langle -p^j, x^j - x^0 \right\rangle < 0, \tag{A.4}$$

and

$$\left\| \sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \lambda^j p^j \right\| \leq \rho \sqrt{\sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} (\lambda^j)^2 \left\|p^j\right\|^2}, \tag{A.5}$$

are satisfied for a constant ρ, and $\lambda^j = \sqrt{\frac{f(x^j) - f(x^{j+1})}{\|p^j\|^2}}$. Then the residual of the function is divided in half after m iterations; i.e., $\nu_f(x^m) \leq \frac{1}{2}\nu_f(x^0)$.

Proof of the above lemma is along the same ideas used in Section 2.1 and very similar to Proof 2.1.1.

Proof. Suppose by contradiction that $m \geq \left\lceil 8\rho\sqrt{\frac{L}{l}} \right\rceil + \left|\kappa^{\mathcal{A}}\right|$, (A.4) and (A.5) are satisfied; but $\nu_f(x^m) > \frac{\nu_f(x^0)}{2}$.

By definition of λ^j,

$$f(x^{j+1}) = f(x^i) - \left(\lambda^j\right)^2 \|p^j\|^2,$$

hence

$$\nu_f(x^{j+1}) = \nu_f(x^j) - \left(\lambda^j\right)^2 \|p^j\|^2.$$

Summing these inequalities over $j \leq m-1$, we get:

$$0 \leq \nu_f(x^m) = \nu_f(x^0) - \sum_{j=0}^{m-1} \left(\lambda^j\right)^2 \|p^j\|^2 \leq \nu_f(x^0) - \sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \left(\lambda^j\right)^2 \|p^j\|^2,$$

or equivalently,

$$\sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \left(\lambda^j\right)^2 \|p^j\|^2 \leq \nu_f(x^0). \tag{A.6}$$

By property (2) of the function we have,

$$\left\langle -p^j, x^* - x^j \right\rangle \leq f(x^*) - f(x^j) = -\nu_f(x^j),$$

and so

$$\left\langle -p^j, x^* - x^0 \right\rangle - \left\langle -p^j, x^j - x^0 \right\rangle \leq -\nu_f(x^j) \leq -\nu_f(x^m) < \frac{-\nu_f(x^0)}{2}.$$

Let's consider the weighted sum of all the above inequalities for $j \in \{0, \dots m-1\} \setminus \kappa^{\mathcal{A}}$ with weights λ^j's to get:

$$\left\langle \sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} -\lambda^j p^j, x^* - x^0 \right\rangle - \sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \lambda^j \left\langle -p^j, x^j - x^0 \right\rangle < \frac{-\nu_f(x^0)}{2} \left(\sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right),$$

which can be rearranged to the following form,

$$\left\langle \sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} -\lambda^j p^j, x^* - x^0 \right\rangle < -\frac{\nu_f(x^0)}{2} \left(\sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right) + \sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j \left\langle -p^j, x^j - x_0 \right\rangle.$$

Equivalently, we can rewrite the above inequality as

$$\begin{aligned}\left\langle \sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} -\lambda^j p^j, x^* - x^0 \right\rangle < & -\frac{\nu_f(x^0)}{4} \left(\sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right) \\ & + \left(\frac{f(x^*) - f(x^0)}{4} \left(\sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right) + \sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j \left\langle -p^j, x^j - x^0 \right\rangle \right).\end{aligned}$$

Using inequality (A.4) along with the facts that $f(x^*) \leq f(x^j)$ and $\lambda^j \geq 0$ for all j, we get

$$\left\langle \sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} -\lambda^j p^j, x^* - x^0 \right\rangle < -\frac{\nu_f(x^0)}{4} \left(\sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right). \tag{A.7}$$

By the Cauchy-Schwarz inequality we have

$$-\left\| \sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j p^j \right\| \left\| x^* - x^0 \right\| \leq \left\langle \sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} -\lambda^j p^j, x^* - x^0 \right\rangle < -\frac{\nu_f(x^0)}{4} \left(\sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right),$$

hence

$$\left\| \sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j p^j \right\| \left\| x^* - x^0 \right\| > \frac{\nu_f(x^0)}{4} \left(\sum_{\substack{j=0 \\ j\notin\kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right). \tag{A.8}$$

By property (3) we have

$$\left\| x^* - x^0 \right\| \leq \sqrt{\frac{2\nu_f(x^0)}{l}}. \tag{A.9}$$

Furthermore, by inequalities (A.5) and (A.6) we get:

$$\left\| \sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \lambda^j p^j \right\| \leq \rho \sqrt{\sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} (\lambda^j)^2 \left\| p^j \right\|^2} \leq \rho \sqrt{\nu_f(x^0)}. \tag{A.10}$$

Replacing inequalities (A.9) and (A.10) in inequality (A.8), we get

$$\rho \sqrt{\nu_f(x^0)} \sqrt{\frac{2\nu_f(x^0)}{l}} > \frac{\nu_f(x^0)}{4} \left(\sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \lambda^j \right). \tag{A.11}$$

Notice that by definition of λ and property (1), $\lambda^j \geq \sqrt{\frac{1}{2L}}$ for all $j \in \{0, \ldots, m-1\} \setminus \kappa^{\mathcal{A}}$, so

$$\sum_{\substack{j=0 \\ j \notin \kappa^{\mathcal{A}}}}^{m-1} \lambda^j \geq \sqrt{\frac{1}{2L}} \left(m - |\kappa^{\mathcal{A}}| \right).$$

Using the above inequality in (A.11), we get

$$\rho \sqrt{\nu_f(x^0)} \sqrt{\frac{2\nu_f(x^0)}{l}} > \frac{\nu_f(x^0)}{4} \left(\sqrt{\frac{1}{2L}} \left(m - |\kappa^{\mathcal{A}}| \right) \right).$$

Therefore we have

$$m - |\kappa^{\mathcal{A}}| < 8\rho \sqrt{\frac{L}{l}}, \tag{A.12}$$

or

$$m < \left\lceil 8\rho \sqrt{\frac{L}{l}} \right\rceil + |\kappa^{\mathcal{A}}|,$$

which contradicts our assumption on the value of m. □

A.2 Implementation of CGSO for BPDN Problem

In this section, we apply Algorithm A.1 to the BPDN problem, i.e., formulation (1.68). The main task in CGSO approach is computing the projected gradient which will be

discussed shortly. Before getting into details of computing the projected gradient, we need to point out that BPDN is actually not a strongly convex function because $Ax - b$ is underdetermined, hence A^tA is not full rank.

Using the standard linear programming technique for (linearly) expressing the absolute value, we are able to reformulate the BPDN problem as the following QP:

$$\begin{array}{ll} \min & f(y,\ x) := \frac{1}{2}\|Ax - b\|^2 + \lambda \mathbf{e}^t y, \\ s.t. & y \geq x, \\ & y \geq -x. \end{array} \tag{A.13}$$

The feasible region of the above problem is $\mathcal{F} = \{(y\ ,\ x)^t : y \geq x,\ y \geq -x\}$. Since the feasible region is a polyhedron, the tangent cone and the feasible directions cone of $\mathcal{F}$ at x^j coincide.

A.2.1 Computing the Projected Gradient

Note that at each iteration if $y_i > |x_i|$, we would be able to improve the solution by decreasing y_i. Hence for each i the list of active constraints is as follows:

$$\begin{array}{llll} y_i = x_i & \text{and} & y_i > -x_i & \text{if } x_i > 0, \\ y_i > x_i & \text{and} & y_i = -x_i & \text{if } x_i < 0, \\ y_i = x_i & \text{and} & y_i = -x_i & \text{if } x_i = 0. \end{array} \tag{A.14}$$

The set of all directions in the tangent cone of $\mathcal{F}$ at $(y\ ,\ x)$ is

$$T_{\mathcal{F}}(y\ ,\ x) = \left\{p = (p^y\ ,\ p^x)^t : (y\ ,\ x)^t + \epsilon(p^y\ ,\ p^x) \in \mathcal{F}, \quad \text{for some } \epsilon > 0\right\}.$$

This implies that for each i,

$$\begin{array}{l} y_i + \epsilon p_i^y \geq x_i + \epsilon p_i^x, \\ y_i + \epsilon p_i^y \geq -(x_i + \epsilon p_i^x). \end{array} \tag{A.15}$$

We may now derive conditions on $(p^y\ ,\ p^x) \in T_{\mathcal{F}}(y, x)$ using (A.14). Clearly for all non-active constraints it is always possible to choose ϵ small enough such the inequalities in (A.15) are satisfied. For active constraints, however, certain conditions must hold:

$$\begin{array}{rl} \cancel{y_i} + \epsilon p_i^y \geq \cancel{x_i} + \epsilon p_i^x & \text{if } \; x_i > 0, \\ \cancel{y_i} + \epsilon p_i^y \geq -(\cancel{x_i} + \epsilon p_i^x) & \text{if } \; x_i < 0, \\ \cancel{y_i} + \epsilon p_i^y \geq \cancel{x_i} + \epsilon p_i^x \\ \cancel{y_i} + \epsilon p_i^y \geq -(\cancel{x_i} + \epsilon p_i^x) & \text{if } \; x_i = 0. \end{array} \tag{A.16}$$

From (A.16), we conclude the following required inequalities for a feasible direction:

$$\begin{array}{rl} p_i^y - p_i^x \geq 0 & \text{if } \; x_i > 0, \\ p_i^y + p_i^x \geq 0 & \text{if } \; x_i < 0, \\ p_i^y - p_i^x \geq 0 \;\text{ and }\; p_i^y + p_i^x \geq 0 & \text{if } \; x_i = 0. \end{array} \tag{A.17}$$

The above inequalities conclude the required specifications for a vector to be in $T_{\mathcal{F}}(y\ ,\ x)$. The projection of $-\nabla f(y\ ,\ x)$ may now be obtained by solving the following problem

$$\begin{array}{lll} \min & \|p + \nabla f(y\ ,\ x)\|^2, & \\ s.t. & p_i^y - p_i^x \geq 0 & \forall i: \; x_i > 0, \\ & p_i^y + p_i^x \geq 0 & \forall i: \; x_i < 0, \\ & p_i^y - p_i^x \geq 0 & \forall i: \; x_i = 0. \\ & p_i^y + p_i^x \geq 0 & \end{array} \tag{A.18}$$

For convenience in notation, let us denote $\nabla f(y\ ,\ x)$ by $g = (g^y\ ,\ g^x)^t$. For problem (A.13), the gradient of the objective function is

$$g = \begin{pmatrix} g^y \\ g^x \end{pmatrix} = \begin{pmatrix} \lambda \mathbf{e} \\ g^x \end{pmatrix} = \begin{pmatrix} \lambda \mathbf{e} \\ A^t(Ax - b) \end{pmatrix}.$$

Therefore (A.18) reduces to

$$\begin{array}{lll} \min & \sum_{i=1}^n \left((p_i^y + \lambda)^2 + (p_i^x + g_i^x)^2\right), & \\ s.t. & p_i^y - p_i^x \geq 0 & \forall i: \; x_i > 0, \\ & p_i^y + p_i^x \geq 0 & \forall i: \; x_i < 0, \\ & p_i^y + p_i^x \geq 0 & \forall i: \; x_i = 0. \\ & p_i^y - p_i^x \geq 0 & \end{array} \tag{A.19}$$

Let $i_0 \leq n$ be the number of zero entries of x. We define γ_i for $i = 1, \ldots, n$ and μ_i for $i = 0, \ldots, i_0$ as the Lagrangian multipliers for inequalities in (A.19); the former one refers to the multipliers of the first three set of inequalities, and the latter one stands for the last

set of inequalities, i.e.,

$$\begin{array}{lll} \min & \sum_{i=1}^{n}\left((p_i^y+\lambda)^2+(p_i^x+g_i^x)^2\right), & \\ s.t. & (\gamma_i) \quad p_i^y - p_i^x \geq 0 & \forall i: \ x_i > 0, \\ & (\gamma_i) \quad p_i^y + p_i^x \geq 0 & \forall i: \ x_i < 0, \\ & (\gamma_i) \quad p_i^y + p_i^x \geq 0 & \forall i: \ x_i = 0. \\ & (\mu_i) \quad p_i^y - p_i^x \geq 0 & \end{array}$$

The above problem is separable in i, thus we are able to find the projected gradient in the closed form by solving n separate one-dimensional problems. We categorize these problems based on $\text{sgn}(x_i)$; and in the remainder of this section we describe the mechanism for obtaining the solution of (A.19).

$\mathbf{i : x_i > 0}$

When x_i is positive, the one dimensional problem for i is

$$\begin{array}{ll} \min & (p_i^y+\lambda)^2+(p_i^x+g_i^x)^2, \\ s.t. & p_i^y - p_i^x \geq 0. \end{array} \tag{A.20}$$

If $g_i^x \geq \lambda$ then $p_i^y = -\lambda$ and $p_i^x = -g_i^x$ solves the problem. Suppose $g_i^x < \lambda$ then by KKT conditions we get

$$p_i^y - p_i^x \geq 0, \tag{A.21}$$
$$\gamma_i \geq 0, \tag{A.22}$$
$$\gamma_i(p_i^y - p_i^x) = 0, \tag{A.23}$$
$$\begin{pmatrix} \gamma_i \\ -\gamma_i \end{pmatrix} = \begin{pmatrix} 2(p_i^y+\lambda) \\ 2(p_i^x+g_i^x) \end{pmatrix}. \tag{A.24}$$

By (A.24), we have $p_i^y = \frac{\gamma_i - 2\lambda}{2}$ and $p_i^x = \frac{-\gamma_i - 2g_i^x}{2}$. Substituting these in (A.23) gives us

$$\gamma_i\left(\gamma_i + (g_i^x - \lambda)\right) = 0.$$

Hence either $\gamma_i = 0$ or $\gamma_i = \lambda - g_i^x > 0$ must hold. If $\gamma_i = 0$, then $p_i^y = -\lambda$ and $p_i^x = -g_i^x$ which is not possible by the assumption that $g_i^x < \lambda$. If $\gamma_i = \lambda - g_i^x > 0$, then

$$p_i^y = p_i^x = \frac{-\lambda - g_i^x}{2}. \tag{A.25}$$

$\mathbf{i : x_i < 0}$

For negative entries, we acquire the corresponding entries of projected gradient by solving

$$\begin{array}{ll} \min & (p_i^y + \lambda)^2 + (p_i^x + g_i^x)^2, \\ s.t. & p_i^y + p_i^x \geq 0. \end{array} \tag{A.26}$$

If $g_i^x \leq -\lambda$ then $p_i^y = -\lambda$ and $p_i^x = -g_i^x$ solves the problem. Suppose $g_i^x > -\lambda$; by KKT conditions we have:

$$p_i^y + p_i^x \geq 0 \tag{A.27}$$
$$\gamma_i \geq 0 \tag{A.28}$$
$$\gamma_i(p_i^y + p_i^x) = 0 \tag{A.29}$$
$$\begin{pmatrix} \gamma_i \\ \gamma_i \end{pmatrix} = \begin{pmatrix} 2(p_i^y + \lambda) \\ 2(p_i^x + g_i^x) \end{pmatrix}. \tag{A.30}$$

Similar to what we described for $x_i > 0$, we get $p_i^y = \frac{\gamma_i - 2\lambda}{2}$ and $p_i^x = \frac{\gamma_i - 2g_i^x}{2}$ by (A.30). Replacing these equations in (A.29) we get

$$\gamma_i \left(\gamma_i + (-g_i^x - \lambda)\right) = 0.$$

Therefore either $\lambda_i = 0$ or $\gamma_i = \lambda + g_i^x > 0$ must hold. In the former case, we derive $p_i^y = -\lambda$ and $p_i^x = -g_i^x$ which contradicts $g_i^x > -\lambda$; in the latter case we deduce

$$\begin{aligned} p_i^y &= \frac{-\lambda + g_i^x}{2}, \\ p_i^x &= \frac{\lambda - g_i^x}{2}. \end{aligned} \tag{A.31}$$

$\mathbf{i : x_i = 0}$

Finally, for zero entries we need to solve the following problem

$$\begin{array}{ll} \min & (p_i^y + \lambda)^2 + (p_i^x + g_i^x)^2, \\ s.t. & p_i^y - p_i^x \geq 0, \\ & p_i^y + p_i^x \geq 0. \end{array} \tag{A.32}$$

In this case $p_i^y = -\lambda$ and $p_i^x = -g_i^x$ is an impossible solution because

$$-\lambda + g_i^x \geq 0 \text{ and } -\lambda - g_i^x \geq 0 \Rightarrow -2\lambda \geq 0 \quad \Rightarrow\Leftarrow$$

The KKT conditions for Problem (A.32) are

$$p_i^y + p_i^x \geq 0, \tag{A.33}$$
$$p_i^y - p_i^x \geq 0, \tag{A.34}$$
$$\gamma_i, \mu_i \geq 0, \tag{A.35}$$
$$\gamma_i(p_i^y + p_i^x) = 0, \tag{A.36}$$
$$\mu_i(p_i^y - p_i^x) = 0, \tag{A.37}$$
$$\begin{pmatrix} \gamma_i + \mu_i \\ \gamma_i - \mu_i \end{pmatrix} = \begin{pmatrix} 2(p_i^y + \lambda) \\ 2(p_i^x + g_i^x) \end{pmatrix}. \tag{A.38}$$

By (A.38) we have $p_i^y = \frac{\gamma_i + \mu_i - 2}{2}$ and $p_i^x = \frac{\gamma_i - \mu_i - 2g_i^x}{2}$. Replacing these in (A.37), we conclude

$$\gamma_i \left(\gamma_i + (-g_i^x - \lambda)\right) = 0,$$

so either $\gamma_i = 0$; or $\gamma_i = \lambda + g_i^x > 0$ provided that $g_i^x > -\lambda$.
Likewise by using (A.36) we get

$$\mu_i \left(\mu_i + (g_i^x - \lambda)\right) = 0$$

which means either $\mu_i = 0$; or $\mu_i = \lambda - g_i^x > 0$ if $g_i^x < \lambda$. Depending on the interval that g_i^x belongs to, we can solve the Problem (A.32) as follows

$$\text{if} \quad g_i^x \leq -\lambda \rightarrow \quad \gamma_i = 0 \ , \ \mu_i = \lambda - g_i^x \quad \Rightarrow p_i^y = p_i^x = \frac{-\lambda - g_i^x}{2}, \tag{A.39}$$
$$\text{if} \quad |g_i^x| < \lambda \ \rightarrow \quad \gamma_i = \lambda + g_i^x \ , \ \mu_i = \lambda - g_i^x \quad \Rightarrow \ p_i^y = p_i^x = 0, \tag{A.40}$$
$$\text{if} \quad g_i^x \geq \lambda \quad \rightarrow \quad \gamma_i = \lambda + g_i^x \ , \ \mu_i = 0 \quad \Rightarrow \ \begin{matrix} p_i^y = \frac{-\lambda + g_i^x}{2}, \\ p_i^x = \frac{\lambda - g_i^x}{2}. \end{matrix} \tag{A.41}$$

A.3 Computational Experiment

In this section we test CGSO on few instances of BPDN problem. We compared CGSO with FISTA, GPSR, and l1_ls. We explained FISTA in Section 1.3; GPSR is a projected gradient type algorithm; finally l1_ls is an interior point algorithm. In l1_ls, the linear system of equations which is the bottle neck of the algorithm is solved by PCG (preconditioned CG). These algorithms are presented in more details in Chapter 5. The results presented here are simply for demonstration. In Chapter 4 we presented a more efficient technique for solving BPDN that outperforms CGSO for this class of problems.

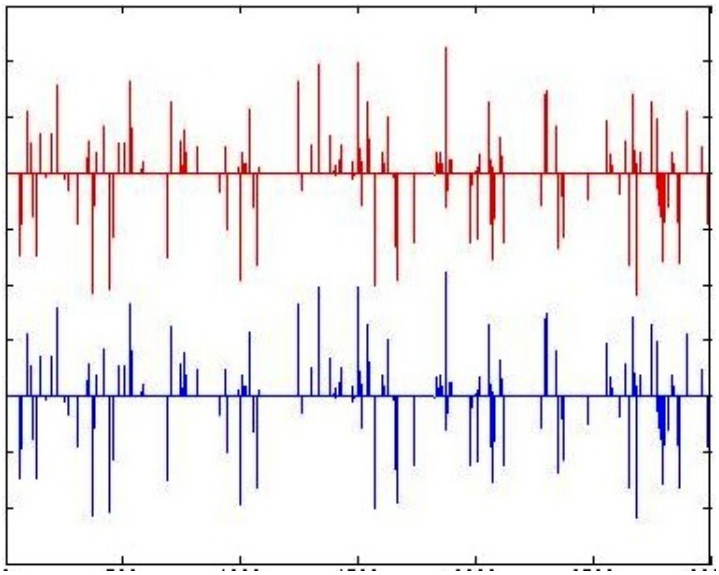


Figure A.2: Signal recovery for CGSO

Graph A.2 is the plot of the original signal in blue and the recovered signal in red for an instance generated through L1TestPack [77, 78]. Table A.1 shows the number of iterations each algorithm takes to recover the solution. The first column is the tolerance used on the norm of projected gradient. The first instance is the one plotted in the Graph A.2; while the second one is randomly generated. For CGSO, we have also reported the number of iteration for which the set of active constraints did not change under the column AS. Table A.2 presents the distance from the optimal solution versus iteration counts for another instance generated by L1TestPack.

Our numerical experience with CGSO suggests that CGSO for the class of problems rising in sparse recovery acts similar to an active set method rather than a conjugate gradient method. It soon reaches the boundary and travels along the boundary. The positive effect of this behaviour is that it can achieve high accuracy. On the negative side, this slows down the algorithm; since at optimality a great share of constraints are active, and we have no theory advocating the idea that the algorithm will not revisit any set of already travelled active sets. Recall that for each zero entry two constraints are active, so the number of nonactive constraints at optimality is essentially the number of nonzeros at the solution.

In Chapter 4, we presented an algorithm that similar to CGSO combines the previous gradient and direction to reach the next iterate. It, however, performs much faster than CGSO and proves to be stronger in practice.

		CGSO		l1-ls		GPSR	FISTA
		it.	AS	it.	PCG		
$A \in \mathbb{R}^{1000\times3000}$ l1TestPack	1e-3	20	2	12	41	7	13
	1e-6	30	10	30	177	25	44
	1e-10	39	19	37	281	31	209
$A \in \mathbb{R}^{500\times1500}$ Random	1e-3	4942	1158	19	2959	11279	417
	1e-6	6149	2207	30	8076	-	3821
	1e-10	6619	2552	46	16988	-	-

Table A.1: Results on CGSO for constrained problems

$A \in \mathbb{R}^{200\times1000}$

it.	CGSO	FISTA	GPSR	l1_ls(pcg)
10	0.83722	0.01673	0.01523	0.10299(42)
20	2.50265e-05	4.25162e-4	7.73314e-06	9.384300e-05(144)
30	4.97380e-10	7.29341e-06	6.28312e-09	1.05611e-07(264)
40	-	2.37749e-07	5.45133e-10	7.77589e-10(459)
50	-	3.84988e-08	4.92008e-10	-
60	-	1.62263e-09	4.92008e-10	-

Table A.2: Comparing the error of the solution, $\|x^k - x^*\|$

References

[1] P. A. Absil and K. A. Gallivan. Accelerated line-search and trust-region methods. *SIAM Journal on Numerical Analysis*, 47(2):997–1018, 2009.

[2] A. Ahmadi, A. Olshevsky, P. A. Parrilo, and J. N. Tsitsiklis. NP-hardness of deciding convexity of quartic polynomials and related problems. *Mathematical Programming*, 137(1-2):453–476, 2013.

[3] M. Al-Baali. Descent property and global convergence of the Fletcher-Reeves method with inexact line search. *IMA Journal of Numerical Analysis*, 5:121–124, 1985.

[4] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. N. Belhumeur. A theory of network localization. *IEEE Transactions on Mobile Computing*, 5(12):1663 – 1678, 2006.

[5] R. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–121, 2007.

[6] J. Barzilai and J. Borwein. Two point step size gradient method. *IMA Journal Numerical Analysis*, 8:141–148, 1988.

[7] H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

[8] M.S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, 2 edition, 1993.

[9] A. Beck and M. Teboulle. Fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.

[10] S. Becker, J. Bobin, and E. Candès. NestA: a fast and accurate first-order method for sparse recovery. Available from: http://www-stat.stanford.edu/~candes/nesta/#code.

[11] S. Becker, J. Bobin, and E. Candès. NestA: a fast and accurate first-order method for sparse recovery. *SIAM Journal Imaging Sciences*, 4(1):1–39, 2011.

[12] S. Becker, E. J. Candès, and M. Grant. Software: Templates for convex cone problems with applications to sparse signal recovery (TFOCS), 2011. Available from: http://cvxr.com/tfocs/paper/.

[13] S. Becker, E. J. Candès, and M. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3), 2011.

[14] S. Becker and M. J. Fadili. A quasi-Newton proximal splitting method. *arXiv.org*, math.OC, June 2012. Available from: http://arxiv.org/abs/1206.1156v1, arXiv:1206.1156v1.

[15] A. Ben-Tal and A. Nemirovski. On solving large scale polynomial convex problems by randomized first-order algorithms, 2013. Available from: http://arxiv.org/abs/1210.6853.

[16] E. van den Berg and M. P. Friedlander. SPGL1: A solver for large-scale sparse reconstruction. Available from: http://www.cs.ubc.ca/~mpf/spgl1/.

[17] E. van den Berg and M. P. Friedlander. In pursuit of a root. Tech. Rep. TR-2007-19, Department of Computer Science, University of British Columbia, Vancouver, June 2007. Available from: http://www.optimization-online.org/DB_HTML/2007/06/1708.html.

[18] E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

[19] E. van den Berg, M. P. Friedlander, G. Hennenfent, F. Herrmann, R. Saab, and Ö. Yılmaz. Sparco: A testing framework for sparse reconstruction. Technical Report TR-2007-20, Dept. Computer Science, University of British Columbia, Vancouver, October 2007. Available from: http://www.cs.ubc.ca/labs/scl/sparco/.

[20] J. M. Bioucas-Dias and M. A. T. Figueiredo. TwIST: Two-step iterative shrinkage/thresholding algorithm for linear inverse problems. Available from: http://www.lx.it.pt/~bioucas/TwIST/TwIST.htm.

[21] J. M. Bioucas-Dias and M. A. T. Figueiredo. A new TwIST: Two-step iterative shrinkage/ thresholding algorithms for image restoration. *IEEE Transactions Image Process.*, 16:2992–3004, 2007.

[22] E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10:1196–1211, 2000.

[23] E. G. Birgin, J. M. Martínez, and M. Raydan. Inexact spectral projected gradient methods on convex sets. *IMA Journal of Numerical Analysis*, 23(4):539–559, 2003.

[24] J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. Springer-Verlag, 2000.

[25] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[26] K. Bredies and D. A. Lorenz. Linear convergence of iterative soft-thresholding. *Journal of Fourier Analysis and Applications*, 14:813–837, 2008.

[27] L. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

[28] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *AISTATS*, 2009.

[29] J. F. Cai, S. Osher, and Z. Shen. Convergence of the linearized bregman iterations for l_1-norm minimization. *Mathematics of Computation*, 78:2127–2136, 2009.

[30] P. H. Calamai and J. J. More. Projected gradient methods for linearly constrained problems. *Mathematical Programming*, 39:93–116, 1987.

[31] E. Candès. Compressive sampling. *International Congress of Mathematics*, 3:1433–1452, 2006.

[32] E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[33] E. Candès and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies. *IEEE Transactions on Information Theory*, 52(12):5406 – 5425, 2006.

[34] E. Candès and M. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[35] A. Cohen. Rate of convergence of several conjugate gradient algorithms. *SIAM Journal on Numerical Analysis*, 9(2):248–259, 1972.

[36] P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4:1168–1200, 2005.

[37] P. L. Combettes and J. C. Pesquet. Proximal thresholding algorithm for minimization over orthonormal bases. *SIAM Journal on Optimization*, 18:1351–1376, 2007.

[38] P. L. Combettes and J. C. Pesquet. Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212, 2011.

[39] Y. H. Dai and Y. Yuan. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM Journal on Optimization*, 10(1):177–182, 1999.

[40] C.D. Dang, K. Dai, and G. Lan. A linearly convergent first-order algorithm for total variation minimization in image processing. *International Journal of Bioinformatics Research and Applications*, 10(1):4 – 26, 2014.

[41] I. Dhillon, D. Kim, and S. Sra. Tackling box-constrained optimization via a new projected quasi-Newton approach. *SIAM Journal Scientific Computing*, 32(6):3548–3563, 2010.

[42] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289 – 1306, 2006.

[43] M. A. T. Figueiredo and R. D. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

[44] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Software: GPSR (gradient projection for sparse reconstruction). Available from: http://www.lx.it.pt/~mtf/GPSR/.

[45] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Software: Sparse reconstruction by separable approximation. Available from: http://www.lx.it.pt/~mtf/SpaRSA/.

[46] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1:586–597, 2007.

[47] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

[48] J. C. Gilbert and J. Nocedal. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on Optimization*, 2(1):21–42, 1992.

[49] D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming*, 141(1):349–382, 2013.

[50] G. H. Golub and C. F. Van loan. *Matrix Computations*. Hopkins Fulfillment Service, 1996.

[51] N. Gould, D. Orban, and P. Toint. Numerical methods for large-scale nonlinear optimization. *Acta Numerica*, pages 299–361, 2005.

[52] M. Gu, L. Lim, and C. Wu. ParNes: a rapidly convergent algorithm for accurate recovery of sparse and approximately sparse signals. *Numerical Algorithms*, pages 1–27, 2012.

[53] W. W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on Optimization*, 16(1):170–192, 2005.

[54] W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*, 2(1):35–58, 2006.

[55] E. T. Hale, W. Yin, and Y. Zhang. Software: Fixed point continuation (FPC). Available from: http://www.caam.rice.edu/~optimization/L1/fpc/#soft.

[56] E. T. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for l_1-regularized minimization with applications to compressed sensing. Technical report, Rice University, 2007.

[57] E. T. Hale, W. Yin, and Y. Zhang. Fixed-point continuation for l_1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19:1107–1130, 2008.

[58] B. Hendrickson. Conditions for unique graph realizations. *SIAM Journal Computing*, 21:65–84, 1992.

[59] M. R. Hestenes and E. Stiefel. Methods of conjugate gradient for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[60] J. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I*. Springer-Verlag, 1996.

[61] J. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer, 2004.

[62] K. Hoffman and R. Kunze. *Linear Algebra*. Prentice Hall, 2 edition, 1971.

[63] R.A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[64] B. Jackson and T. Jordán. Graph theoretic techniques in the analysis of uniquely localizable sensor networks. *Localization Algorithms and Strategies for Wireless Sensor Networks*, pages 146–173, 2009.

[65] A. Juditsky, F. K. Karzan, and A. Nemirovski. Randomized first order algorithms with applications to l_1-minimization. *Mathematical Programming*, 142(1-2):269–310, 2013.

[66] S. Karimi and S. A. Vavasis. Conjugate gradient with subspace optimization, 2012. Available from: http://arxiv.org/abs/1202.1479v1.

[67] S. Karimi and S. A. Vavasis. Detecting and correcting the loss of independence in nonlinear conjugate gradient, 2013. Available from: http://arxiv.org/abs/1202.1479v2.

[68] S. Karimi and S. A. Vavasis. IMRO: A proximal quasi-Newton method for solving l_1-regularized least square problem, 2013.

[69] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l_1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, 2007.

[70] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing*, 23(2):517–541, 2001.

[71] K. Koh, S. Kim, S. Boyd, and Y. Lin. l_1_ls: Simple MATLAB solver for l_1-regularized least squares problems. Available from: http://www.stanford.edu/~boyd/l1_ls/.

[72] G. Lan and R. D. C. Monteiro. Iteration-complexity of first-order augmented lagrangian methods for convex programming, 2013.

[73] G. Lan and R. D. C. Monteiro. Iteration-complexity of first-order penalty methods for convex programming. *Mathematical Programming*, 138(1):115–139, 2013.

[74] J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for minimizing composite functions, 2013. Available from: arXiv:1206.1623v11.

[75] A. S. Lewis and M. L. Overton. Nonsmooth optimization via quasi-Newton methods. *Mathematical Programming*, 141(1):135–163, 2013.

[76] E. Liu and V. N. Temlyakov. The orthogonal super greedy algorithm and applications in compressed sensing. *IEEE Transactions on Information Theory*, 58(4):2040–2047, 2012.

[77] D. A. Lorenz. Constructing test instances for basis pursuit denoising. arXiv.org/abs/1103.2897, 2011. Available from: http://arxiv.org/abs/1103.2897.

[78] D. A. Lorenz. L1TestPack: A software to generate test instances for l_1 minimization problems., 2011. http://www.tubraunschweig.de/iaa/personal/lorenz/l1testpack.

[79] M. Lustig, D. Donoho, and J. M. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[80] P. Mikusinski and M. Taylor. *An Introduction to Multivariable Analysis from Vector to Manifold*. Birkhäuser, 2002.

[81] R. D. C. Monteiro, C. Ortiz, and B. F. Svaiter. An adaptive accelerated first-order method for convex optimization, 2012. Available from: http://www.optimization-online.org/DB_HTML/2012/08/3554.html.

[82] A. S. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley & Sons, 1983.

[83] Y. Nesterov and A. Nemirovski. On first-order algorithms for l_1/nuclear norm minimization. *Acta Numerica*, 22:509–575, 2013.

[84] Y. E. Nesterov. A method of solving a convex programming problem with convergence rate $o(\frac{1}{k^2})$. *Soviet Mathematics, Doklady*, 27(2):372–376, 1983.

[85] Y. E. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

[86] Y. E. Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103:127–152, 2005.

[87] Y. E. Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE, Université Catholique de Louvain, 2007.

[88] Y. E. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[89] Y. E. Nesterov and A. S. Nemirovsky. *Interior point polynomial algorithms in convex programming*. SIAM, 1994.

[90] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Science, 2006.

[91] S. Osher, Y. Mao, B. Dong, and W. Yin. Fast linearized bregman iteration for compressive sensing and sparse denoising. *Communications in Mathematical Sciences*, 8(1):93–111, 2010.

[92] E. Polak and G. Ribiére. Note sur la convergence de méthodes de directions conjuguées. *Revue Francaise d'informatique et de Recherche Opeŕationnelle*, 16:35–43, 1969.

[93] B. T. Polyak. *Introduction to Optimization*. Optimization Software Inc., 1987.

[94] J. Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*. MPS-SIAM Series on Optimization, 2001.

[95] T. W. Reps and L. B. Rall. Computational divided differencing and divided difference arithmetics. *Higher-Order and Symbolic Computation*, 16:93–149, 2003.

[96] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal Control and Optimization*, 14(5):877–898, 1976.

[97] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[98] M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. *SIAM Journal Scientific Computing*, 16(5):1190–1208, 1995.

[99] I. Selesnick, R. Van Slyke, and O. Guleryuz. Pixel recovery via l_1 minimization in the wavelet domain. *Proceedings of the 2004 International Conference on Image Processing*, 3:1819–1822, 2004.

[100] J. Stewart. *Multivariable Calculus*. Brooks Cole, 2011.

[101] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.

[102] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization, 2008.

[103] Rice University. Compressive sensing resources, 2011. Available from: http://dsp.rice.edu/cs.

[104] S. Vavasis. Some notes on applying computational divided differencing in optimization, 2013. Available from: http://www.optimization-online.org/DB_HTML/2013/07/3957.html.

[105] S. A. Vavasis. *Nonlinear Optimization- Complexity Issues*. Oxford University Press, 1991.

[106] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang. FPC_AS: A MATLAB solver for l1-regularization problems. Available from: http://www.caam.rice.edu/~optimization/L1/FPC_AS/.

[107] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation. *SIAM Journal on Scientific Computing*, 32:1832–1857, 2010.

[108] D. P. Wipf and B. D. Rao. Sparse bayesian learning for basis selection. *IEEE Transactions on Signal Processing, Special Issue on Machine Learning Methods in Signal Processing*, 52:2153–2164, 2004.

[109] S. J. Wright. *Primal Dual Interior Point Methods*. SIAM, 1997.

[110] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[111] T. Yamamoto, M. Yamagishi, and I. Yamada. Adaptive proximal forward-backward splitting for sparse system identification under impulsive noise. *20th European Signal Processing Conference (EUSIPCO)*, pages 2620 – 2624, 2012.

[112] J. Yang and Y. Zhang. Alternating direction algorithms for l_1-problems in compressive sensing. Technical report, Rice University, 2009.

[113] J. Yang, Y. Zhang, and W. Yin. A fast alternating direction method for TVl1-l2 signal reconstruction from partial Fourier data. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):228–297, 2010.

[114] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for l_1-minimization with applications to compressed sensing. *SIAM Journal Imaging Sciences*, 1(1):143–168, 2008.

[115] J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to nonsmooth convex optimization problems in machine learning. *Journal of Machine Learning Research*, 11:1145–1200, March 2010. Available from: http://dl.acm.org/citation.cfm?id=1756006.1756045.

[116] Y. Yuan and J. Stoer. A subspace study on conjugate gradient algorithms. *ZAMM (Zeitschrift fr angewandte Mathematik und Mechanik)*, 11:69–77, 1995.

[117] Y. Zhang. When is missing data recoverable? Caam technical report tr06-15, Rice University, 2006.

[118] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 67(2):301–320, 2005.