



universität
wien

DISSERTATION

Titel der Dissertation

High-Dimensional Nonsmooth Convex
Optimization via Optimal Subgradient Methods

Verfasser

Masoud Ahookhosh

angestrebter akademischer Grad

Doktor der Naturwissenschaften (Dr. rer. nat.)

Wien, im Juni 2015

Studienkennzahl lt. Studienblatt: A 796 605 405

Dissertationsgebiet lt. Studienblatt: Mathematik

Betreuer: o. Univ.-Prof. Dr. Arnold Neumaier

**To my love,
Kiumars, Soraya, Paria, and Kaveh**

Abstract

Over the past few decades, convex optimization has obtained tremendous attention and grown very quickly due to providing a rich theoretical framework and computationally reliable and tractable schemes. Designing iterative schemes in convex optimization are typically based of zero-, first-, or second-order information of the objective. The first-order information (function values and subgradients) has a reputation of providing the most promising schemes for solving problems involving high-dimensional or big data. Among first-order methods, the subgradient methods have a simple form and can deal with general convex optimization (without considering the structure of problems, in contrast to proximal-based and Nesterov-type optimal methods). However, they are too slow to be able handling real-life problems appearing in applied sciences and engineering.

In this thesis, motivated by the need for fast and reliable methods to solve general nonsmooth convex optimization, we develop some subgradient methods obtaining the optimal complexity of first-order methods that are known to be fast enough to deal with applications involving high-dimensional or big data. More specifically, the novel subgradient framework (OSGA) depends on solving efficiently a related nonconvex subproblem. We show that this subproblem can be solved efficiently for unconstrained (multi-term affine composite functions and objective involving costly linear operators), simply constrained (bound-constrained and simple domains with available projection), and simply functional constrained (sublevel set of simple convex function) problems. In addition, if the nonsmoothness of the objective is manifested in an appropriately structured form, a novel optimal subgradient method is presented that can attain the complexity $O(\epsilon^{-1/2})$, the same optimal complexity as for smooth problems with Lipschitz continuous gradients.

OSGA is released as a software package available freely for academic use. Numerical results and comparisons with state-of-the-art schemes regarding a number of interesting problems in application are reported.

Zusammenfassung

In den letzten Jahrzehnten hat die konvexe Optimierung enorme Aufmerksamkeit erhalten und sich aufgrund ihres reichen theoretischen Rahmens sowie der rechnerischen Zuverlässigkeit sehr rasch entwickelt. Die iterativen Verfahren in der konvexen Optimierung basieren typischerweise auf Informationen nullter, erster oder zweiter Ordnung über die Zielfunktion. Dabei bilden Informationen erster Ordnung (Funktionswerte und Subgradienten) die meistversprechenden Methoden zum Lösen von hoch dimensionalen oder „big data“-Problemen. Unter diesen Methoden bieten sich sogenannte Subgradienten-Methoden an, da diese eine einfache Form haben und mit allgemeinen konvexen Problemen umgehen können, ohne dabei die Struktur des Problems zu berücksichtigen (im Gegensatz zu Optimierungsmethoden, die auf proximalen Punkten basieren oder solchen vom Nesterov-Typ). Der Nachteil dieser Subgradienten-Verfahren ist jedoch, dass sie zu langsam sind, um reale Probleme, die in Technik oder angewandter Wissenschaft auftreten, effektiv lösen zu können.

Motiviert durch den Bedarf an schnellen und zuverlässigen Methoden für das Lösen von allgemeinen nicht-glaten konvexen Optimierungsproblemen wurden im Zuge dieser Arbeit spezielle Subgradienten-Verfahren entwickelt, die zum einen die optimale Komplexität von bekannten Verfahren erster Ordnung haben und zum anderen schnell genug sind, um hoch-dimensionale und „big data“-Anwendungsprobleme lösen zu können. Das neue Subgradienten-Verfahren (OSGA) beruht auf der Lösbarkeit eines nicht-konvexen Teilproblems. In dieser Arbeit wird gezeigt, dass man dieses Teilproblem für unbeschränkte (multi-term affine zusammengesetzte Funktionen, sowie Zielfunktionen, die kostenintensive lineare Operatoren beinhalten) und einfach beschränkte Probleme (auf einfachen Gebieten mit effektiver Projektion), sowie Probleme beschränkt durch einfache Funktionen (Sublevel set von einfachen konvexen Funktionen) effizient lösen kann. Wenn zusätzlich die Nicht-Glattheit der Zielfunktion in einer passenden Struktur formuliert ist, kann eine neue Subgradienten-Methode, die ebenfalls hier vorgestellt wird, eine Komplexität von $O(\varepsilon^{-1/2})$ erreichen, welche dem optimalen Aufwand für glatte Probleme mit Lipschitz-stetigen Gradienten entspricht.

Numerische Ergebnisse und Vergleiche mit anderen state-of-the-art-Verfahren werden in dieser Arbeit anhand von verschiedenen interessanten Anwendungsproblemen präsentiert. OSGA wurde als Software-Paket veröffentlicht und ist für die Nutzung zu akademischen Zwecken frei verfügbar.

Acknowledgement

I am indebted to many people who encouraged and supported me through my study in University of Vienna.

First and foremost, I wish to express my deepest sincere gratitude to my supervisor Prof. Arnold Neumaier for his invaluable insight, encouragement, and continuous guidance during this research. Indeed, without his mentoring this dissertation would have not been possible. I feel very fortunate to have had him as a knowledgeable teacher and a precise mathematician.

I would like to thanks Prof. Yurii Nesterov, Prof. Marko M. Mäkelä, and Prof. Radu Ioan Boț for refereeing this thesis. I would like to thanks Prof. Hermann Schichl and Prof. Radu Ioan Boț who taught me a lot during their Nonsmooth Analysis, Selected Topics in Nonsmooth Analysis, and Convex Optimization courses and Optimization seminars. I also thankful to Prof. Keyvan Amini, Prof. Shapour Heidarkhani, and Prof. M. Reza Peyghami for their support.

I would like to thanks my colleagues in IK Computational Optimization for sharing their knowledge in these three years. In particular, I am grateful of Prof. Georg Pflug for his kindness, understanding, and support. It has been my pleasure to thank the guest professors of our group Prof. Yurii Nesterov, Prof. Margaret Wright, Prof. Florian Jarre, and Prof. Nick Sahinidis for their enthusiasm, energy, and great teaching during their intensive courses in University of Vienna. I also would like to thanks to Gerald Kamhuber and Svetlana Mihajlovic for their help and support.

I feel very grateful to my friends Simon Konzett, Peter Gross, Eric Laas-Nesbitt, Dmytro Dzhunzhuk, Harald Schilly, Bettina Ponleitner, Iraj Yadegari, Susan Ghaderi, Bitu Analui, Razieh Taasob, Babak Ghadiri, Arash Ghaani Farashahi, and Behzad Azmi who have supported me throughout these years in Vienna and have made my time very enjoyable. A very spacial thanks goes to my friends Arash Ghaani Farashahi and Susan Ghaderi for proofreading the thesis.

My deepest gratitude and endless love goes to my lovely parents Kiumars and Soraya, my sister Paria, and my brother Kaveh for their pure and unconditional love, support, and inspiration during my life.

Contents

Abstract	iv
Zusammenfassung	v
Acknowledgement	vi
List of Figures	xi
List of Tables	xiii
I Convex Optimization and Subgradient Methods	1
1 Introduction	2
2 Background of convex analysis and optimization	5
2.1 Convex analysis tools	6
2.2 Optimality conditions	12
2.3 Information-based optimization	14
2.4 First-order methods	15
2.5 Optimal first-order methods	17
3 Optimal subgradient algorithms: basic idea & derivation	20
3.1 Novel subgradient framework	20
3.1.1 Linear relaxations	22
3.1.2 Step-size selection	23
3.1.3 Strongly convex relaxations	24
3.2 Optimal subgradient algorithms	25
3.2.1 Single-projection OSGA	25
3.2.2 Double-projections OSGA	28
3.3 Convergence analysis	31

II	Optimal Subgradient Algorithms: Applicability & Developments	35
4	Unconstrained convex optimization	36
4.1	Multi-terms affine composite problems	36
4.1.1	Convergence analysis	39
4.1.2	Solving the auxiliary subproblem	40
4.2	Problems involving costly linear operators	44
4.2.1	Multi-dimensional subspace search	46
4.2.2	Solving the subspace subproblem by OSGA	48
5	Convex optimization with simple constraints	51
5.1	Convex optimization with simple domains	51
5.1.1	Convex problems with simple domains	51
5.1.2	Structured problems with a simple functional constraint	59
5.2	Bound-constrained convex optimization	65
5.2.1	Explicit solution of OSGA's rational subproblem	67
5.2.2	Inexact solution of OSGA's rational subproblem	75
6	Solving nonsmooth convex problems with complexity $O(\varepsilon^{-1/2})$	76
6.1	Structured convex optimization problems	76
6.1.1	Description of OSGA's new setup	78
6.1.2	Convergence analysis	83
6.2	Solving proximal-like subproblem	84
6.2.1	Unconstrained examples ($C = V$)	88
6.2.2	Constrained examples ($C \neq V$)	95
III	Numerical Results, Applications, and Future Work	100
7	OSGA software package	101
7.1	Inputs and outputs	102
7.2	Forward and adjoint operations	103
7.3	Stopping criteria	104
7.4	Building your own problem	104
7.5	How to call OSGA	105
8	Unconstrained convex optimization	106
8.1	Multi-term affine composite problems	106
8.1.1	Image restoration	106
8.1.2	A comparison among first-order methods	125
8.1.3	Compressed sensing and sparse optimization	130

8.2	Problems involving costly linear operators	134
9	Convex optimization with simple constraints	143
9.1	Convex optimization with simple constraints	143
9.1.1	Ridge regression	143
9.1.2	Image deblurring with nonnegativity constraint	144
9.2	Bound-constrained convex optimization problems	149
9.2.1	Experiment with artificial data	152
9.2.2	Image deblurring/denoising	154
10	Solving nonsmooth convex problems with complexity $O(\varepsilon^{-1/2})$	163
10.1	ℓ_1 minimization	164
10.2	Elastic Net minimization	167
11	Summary and future work	171
11.1	Extended summary	171
11.2	Directions for future research	173
	Bibliography	175

List of Figures

8.1	Denoising of the 1024×1024 Pirate image (comparison)	110
8.2	Denoising of the 1024×1024 Pirate image (visualization)	111
8.3	Inpainting of the 512×512 Head CT image (comparison)	114
8.4	Inpainting of the 512×512 Head CT image (visualization)	115
8.5	Deblurring of the 512×512 Elaine image (comparison)	117
8.6	Deblurring of the 512×512 Elaine image (visualization)	118
8.7	Deblurring of the 512×512 Mandril image (comparison)	119
8.8	Deblurring of the 512×512 Mandril image (visualization)	120
8.9	Performance profiles for function values and PSNR	122
8.10	Comparison among PGA, FISTA, NESCO, NESUN, and OSGA	128
8.11	Comparison among NSDSG, NES83, NESCS, NES05, and OSGA	129
8.12	Sparse recovery (comparison)	132
8.13	Sparse recovery (visualization)	133
8.14	Comparison I between OSGA and OSGA-S (iterations)	136
8.15	Comparison II between OSGA and OSGA-S (iterations)	137
8.16	Comparison I between OSGA and OSGA-S (running time)	138
8.17	Comparison II between OSGA and OSGA-S (running time)	139
8.18	Comparison I among SGA-1, SGA-2, OSGA, and OSGA-S	141
8.19	Comparison II among SGA-1, SGA-2, OSGA, and OSGA-S	142
9.1	Ridge regression with PGA, SPG-G, SPG-A, and OSGA	145
9.2	Deblurring by L22ITVR (nonnegativity constraint, comparison)	147
9.3	Deblurring by L22ITVR (nonnegativity constraint, visualization)	148
9.4	Deblurring by L1ITVR (nonnegativity constraint, comparison)	150
9.5	Deblurring by L1ITVR (nonnegativity constraint, visualization)	151
9.6	Comparison I among PSGA-1, PSGA-2, OSGA-1, and OSGA-2	154
9.7	Comparison II among PSGA-1, PSGA-2, OSGA-1, and OSGA-2	155
9.8	Deblurring by L22ITVR (Bound-constrained domain, comparison)	157
9.9	Deblurring by L22ITVR (Bound-constrained domain, visualization)	158
9.10	Deblurring by L1ITVR (Bound-constrained domain, comparison)	160
9.11	Deblurring by L1ITVR (Bound-constrained domain, visualization)	161

10.1	Comparison among PGA, FISTA, NESCO, NESUN, OSGA, and OSGA-O	165
10.2	Comparison among NSDSG, NES83, NESCS, NES05, OSGA, and OSGA-O	166
10.3	Comparison among PGA, FISTA, NESCO, NESUN, OSGA, and OSGA-O	169
10.4	Comparison among NSDSG, NES83, NESCS, NES05, OSGA, and OSGA-O	170

List of Tables

2.1	Convex optimization problems and their complexity	18
4.1	Minimization problems for solving overdetermined systems	45
5.1	List of some available projection operators for $C = \{x \in V \mid c(x)\}$	55
5.2	List of domains C where $\varphi(e) = 0$ can be solved explicitly	56
8.1	Denoising of the 1024×1024 Pirate image	109
8.2	Inpainting of the 512×512 Head CT image	113
8.3	Deblurring of the 512×512 Elaine image (L22ITVR)	121
8.4	Deblurring of the 512×512 Mandril image (L1ITVR)	122
8.6	Comparison among PGA, FISTA, NESCO, NESUN, and OSGA	130
8.7	Comparison among NSDSG, NES83, NESCS, NES05, and OSGA	130
8.8	Comparison between OSGA and OSGA-S	135
8.9	Comparison among SGA-1, SGA-2, OSGA, and OSGA-S	140
9.1	Result summary for the ridge regression	144
9.2	Result summary for L22ITV	146
9.3	Results summary for L1ITV	149
9.4	Result summary for L22L22R and L22L1R	153
9.5	Result summary for L1L22R and L1L1R	153
9.6	Result summary for the ℓ_2^2 isotropic total variation	159
9.7	Results for the ℓ_1 isotropic total variation	159
10.1	Comparison among PGA, FISTA, NESCO, NESUN, OSGA, and OSGA-O	167
10.2	Comparison among NSDSG, NES83, NESCS, NES05, OSGA, and OSGA-O	167
10.3	Comparison among PGA, FISTA, NESCO, NESUN, OSGA, and OSGA-O	168
10.4	Comparison among NSDSG, NES83, NESCS, NES05, OSGA, and OSGA-O	168

Part I

**Convex Optimization and
Subgradient Methods**

Chapter 1

Introduction

Optimization (nonlinear programming) is a branch of mathematics expanding at an overwhelming rate in many fields of applied sciences, medicine, economics, and engineering. It deals with finding the local or global optimizer of an objective function within a given set of feasible points (feasible domain) that is specified by some constraints. One of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field (see the introductory book BOYD & VANDENBERGHE [43]). For general nonlinear optimization problems, one can typically find a local optimizer and it is hard to find a global optimizer, especially if the dimension of the problem is large. Hence finding a global solution needs much more sophisticated techniques that cannot be applied for very large problems (see the survey by NEUMAIER in [129] and references therein). If the objective is smooth, there exists a mature theory and many numerical schemes guaranteeing to find a stationary point. However, for nonsmooth problems, designing efficient schemes and checking the optimality conditions becomes much more difficult. In the case of nonsmooth nonconvex problems, one cannot even guarantee the existence of a descent direction in general.

Apart from linear programming, the most tractable class of problems is that of convex problems. They share very good features: (i) every local optimizer is also a global optimizer; (ii) a rich theory is available that provides us necessary and sufficient optimality conditions; (iii) efficient numerical schemes are available that can solve problems involving high-dimensional data in reasonable computational cost. Information-based convex optimization gives the possibility to handle many interesting applications, where only zero-, first-, and second-order information is available. If the required accuracy is not too high, first-order methods provide responsible schemes due to their cheap computational cost, low memory requirement, and a rich convergence theory. First-order methods are generally designed for structural classes of problems with a given level of smoothness and convexity of the objective, and the corresponding convergence analysis is valid only for

this class of problems. However, there exist some schemes such as subgradient, cutting plan, and bundle methods that are able to handle general convex problems.

In this thesis we consider smooth and nonsmooth convex optimization problems involving high-dimensional or big data. Since the dimension of the problems considered is large, we investigate schemes that the cost of each iteration is only weakly dependent on the problem dimension (cf. NEMIROVSKY & YUDIN [117]). We are interested in fast methods that can be applied for general convex optimization without considering the structure of problems. Nesterov-type optimal methods obtain the best possible complexity with weakly dependence to the problem dimension, however, they apply to special classes of convex problems. Lan's accelerated bundle-level method [105] has optimal complexity for several classes of convex problems and applies to general convex optimization, but the cost of each iteration is typically too large to be applicable in general. These facts motivate the quest to develop some subgradient schemes for solving general convex optimization problems such that the complexity for finding an ε -solution is only weakly dependent on the dimension.

This thesis presents in detail the design and analysis of two iterative schemes based on an optimal subgradient framework originally proposed by NEUMAIER in [130]. The thesis is divided into three parts that are described in the following:

1. The first part of the thesis consists of three chapters. The first chapter gives an introduction to the subject of the thesis. In Chapter 2 we review some introductory results, required definitions and results from convex analysis, optimality conditions, information-based optimization, and first-order methods. In Chapter 3 we give a review of the OSGA optimal subgradient framework and then introduce two iterative schemes, namely the single projection OSGA and the double projection OSGA requiring one and two times solving of OSGA's subproblem, respectively. We then establish the convergence analysis of both methods.
2. The second part of the thesis consists of three chapters in which the applicability and some developments of optimal subgradient algorithms for solving convex optimization problems are investigated. In Chapter 4 we show how OSGA's subproblem can be solved in a closed form for unconstrained problems. We develop an accelerated version of OSGA using a multi-dimensional subspace search for solving unconstrained problems involving costly linear mappings and cheap nonlinear terms. In Chapter 5 we discuss solving convex problems in simple domains with OSGA, where simple means that either the orthogonal projection onto these domains is efficiently available or the domain is defined by a sublevel set of a simple convex function. In particular, we consider bound-constrained convex problems and show that OSGA's subproblem can be solved globally by a simple

iterative scheme. In Chapter 6 we consider a class of structured nonsmooth convex problems, reformulate it in such a way that nonsmooth terms appear as a single constraint in the feasible domain and the new objective is smooth with Lipschitz continuous gradients, and propose a new setup of OSGA that can minimize the reformulated problem with the complexity $O(\varepsilon^{-1/2})$.

3. The third part of the thesis consists of five chapters. They consider applications, apply OSGA and some state-of-the-art schemes and solvers to such applications, and give some comparisons. In Chapter 7 we first introduce the software package implementing OSGA for several classes of convex problems such as unconstrained, bound-constrained, simple constraints, and simple functional constraints. In Chapter 8, we consider several unconstrained problems in applications such as signal and image processing, compressed sensing, sparse optimization, and statistics. We conduct numerical experiments with such applications and compare with state-of-the-art schemes or solvers such as FISTA, several Nesterov-type optimal methods, ADMM (alternating direction method of multipliers), accelerated forward-backward splitting, accelerated primal-dual methods, and Douglas-Rachford splitting methods. In Chapter 9 we consider problems with simple constraints (bound-constrained, simple domains, and simple functional constraints) and report numerical results for OSGA and some state-of-the-art solvers. In Chapter 10 we report numerical results and comparison regarding structured nonsmooth problems discussed in Chapter 6. In Chapter 11 we give some conclusions and directions for future work.

The main contribution of the thesis are based on:

1. The articles AHOOKHOSH [1] (for unconstrained multi-term affine composite problems), AHOOKHOSH & NEUMAIER [9, 12] (for unconstrained problems involving costly linear operators), AHOOKHOSH & NEUMAIER [10] (for bound-constrained problems), AHOOKHOSH & NEUMAIER [11] (for problems with simple domains or simple functional constraints), and AHOOKHOSH & NEUMAIER [13] (for solving structured nonsmooth problems with the complexity $O(\varepsilon^{-1/2})$). However, a single projection version of OSGA that will be presented in Chapter 3 does not involve the mentioned articles that will be presented in a separate paper.
2. A software package implementing OSGA and the corresponding user guide [2] that are publicly available.

Chapter 2

Background of convex analysis and optimization

Convexity and smoothness are key features of objective functions in modern optimization playing an important role in designing schemes and developing efficient solvers for such problems. For convex problems, if a local minimizer exists, it is a global minimizer as well, however, for nonconvex problems, it is possible to have several local stationary points that none of them are a global minimizer. Moreover, if the objective is strictly convex and the feasible domain is convex, the minimizer is unique. It is also known that solving a nonsmooth problem is usually much more difficult and costly than a smooth one. In addition, for nonsmooth nonconvex problems, even resolving the question of whether there exists a descent direction from a point is NP-hard, cf. [123]. Apart from linear programming, the simplest class of problems with good features for developing iterative schemes is smooth or nonsmooth convex problems.

Convex optimization has been shown to provide efficient algorithms for computing reliable solutions in a broad range of applications. Many applications arising in applied sciences and engineering such as automatic control systems, estimation and signal and image processing, communications and networks, electronic and circuit design, data analysis and modeling, machine learning, statistics, geophysics, finance, and general inverse problems can be addressed by a convex optimization problem involving high-dimensional data. With the immense growth of available data, developing algorithmic schemes for large datasets has become increasingly important [52, 82, 83]. Thanks to recent developments in optimization theory and computing, convex optimization have provided lots of schemes to handle problems involving big data. Hence the convexification of nonlinear problems received much attention during recent years. Indeed, there exist lots of iterative schemes for solving convex optimization problems that can be categorized with respect to the information needed to perform these schemes (see Section 2.3).

2.1 Convex analysis tools

Let V be a finite-dimensional vector space endowed with the norm $\|\cdot\|$, and V^* denotes its dual space, which is formed by all linear functional on V where the bilinear pairing $\langle g, x \rangle$ denotes the value of the functional $g \in V^*$ at $x \in V$. The associated dual norm of $\|\cdot\|$ is defined by

$$\|g\|_* = \sup_{z \in V} \{\langle g, z \rangle : \|z\| \leq 1\}.$$

If $V = \mathbb{R}^n$, then, for $1 \leq p \leq \infty$,

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}, \quad \|x\|_{1,p} = \sum_{i=1}^m \|x_{g_i}\|_p, \quad (2.1)$$

where $x = (x_{g_1}, \dots, x_{g_m}) \in \mathbb{R}^{n_1} \times \dots \times \mathbb{R}^{n_m}$ in which $n_1 + \dots + n_m = n$. $N(x, r)$ denotes a open ball around x with the radius r , i.e.,

$$N(x, r) = \{z \in V \mid \|z - x\| < r\}.$$

Definition. 2.1.1. A set $C \subseteq V$ is convex if for any $x, y \in C$ and for any $\theta \in [0, 1]$

$$\theta x + (1 - \theta)y \in C.$$

In particular, if C is a box, we denote it by $\mathbf{x} = [\underline{x}, \bar{x}]$, where in which $\underline{x}$ and $\bar{x}$ are the vectors of lower and upper bounds on the components of $\mathbf{x}$, respectively.

Definition. 2.1.2. Let $C \subset V$. The convex hull of a set C , denoted by $\text{conv}(C)$, is the intersection of all the convex subset of V containing C so that the smallest convex subset of V containing C , i.e., is a set of all convex combination of points in C (Proposition 3.4 in [24])

$$\text{conv}(C) := \left\{ \sum_{i \in I} \theta_i x_i \mid I \text{ is finite, } \{x_i\} \subset C, \{\theta_i\} \subset]0, 1[, \sum_{i \in I} \theta_i = 1 \right\}.$$

Definition. 2.1.3. The **conical hull** of a set $C \subseteq V$, denoted by $\text{cone}(C)$, is the intersection of all the cones in V containing C , i.e., the smallest cone in V containing C .

Definition. 2.1.4. The **interior** of a set $C \subseteq V$ can be expressed as

$$\text{int}(C) = \{x \in C \mid \exists r > 0 : N(0, r) \subset C - x\},$$

and the **relative interior** of C is

$$\text{ri}(C) = \{x \in C \mid \text{cone}(C - x) = \text{span}(C - x)\}.$$

Definition. 2.1.5. The *indicator function* of the set C is $\delta_C : V \rightarrow \overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$ defined by

$$\delta_C(x) = \begin{cases} 0 & \text{if } x \in C, \\ +\infty & \text{otherwise,} \end{cases}$$

which is a convex function if and only if C is a convex set.

Definition. 2.1.6. The *normal cone* of the set C at x is denoted by $N_C(x)$ and defined by

$$N_C(x) := \{p \in V \mid \langle p, x - z \rangle \geq 0 \quad \forall z \in C\}. \quad (2.2)$$

This definition plays a key role for giving optimality condition of a convex function over the domain C (see (2.8)).

Definition. 2.1.7. Let $f : V \rightarrow \overline{\mathbb{R}}$. The *effective domain* of f is defined by

$$\text{dom } f := \{x \in V \mid f(x) < +\infty\}$$

and f is called **proper** if $\text{dom } f \neq \emptyset$ and $f(x) > -\infty$, for all $x \in V$.

Definition. 2.1.8. Let $f : C \rightarrow \overline{\mathbb{R}}$ be a given function and $C \subseteq V$ be a nonempty convex set. The function f is called **convex** on C if

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) \quad (2.3)$$

for all $x, y \in C$ and for all $\theta \in [0, 1]$. A function f is **strictly convex** if strict inequality holds in (2.3) whenever $x \neq y$ and $\theta \in (0, 1)$. Moreover, f is called **strongly convex** with the convexity parameter σ if and only if

$$f(\theta x + (1 - \theta)y) + \theta(1 - \theta) \frac{\sigma}{2} \|x - y\|^2 \leq \theta f(x) + (1 - \theta)f(y), \quad (2.4)$$

for all $x, y \in C$ and $\lambda \in (0, 1)$, where $\|\cdot\|$ is an arbitrary norm of the vector space V (see also (2.5)).

Geometrically, the convexity of f means that the straight line linking the points $(x, f(x))$ and $(y, f(y))$ is above the graph of f on the interval $[x, y]$.

Definition. 2.1.9. Let $C \subseteq V$ be open and $f : V \rightarrow \mathbb{R}$. Then f is called *Lipschitz continuous* if there exists a constant $L_1 > 0$ such that

$$|f(x) - f(y)| \leq L_1 \|x - y\|,$$

for all $x, y \in C$. Moreover, f has *Lipschitz continuous gradients* if there exists a constant $L_2 > 0$ such that

$$\|g(x) - g(y)\|_* \leq L_2 \|x - y\|,$$

for all $x, y \in C$, where $g(x) = \nabla f(x)$.

Definition. 2.1.10. Let $C \subseteq V$ be open and $f : V \rightarrow \mathbb{R}$. Then f has Hölder continuous gradients if there exists a constant $L_v > 0$, for $v \in [0, 1]$, such that

$$\|g(x) - g(y)\|_* \leq L_v \|x - y\|^v,$$

for all $x, y \in C$, where $g(x) = \nabla f(x)$.

The class of problems with Hölder continuous gradients cover the classes of nonsmooth ($v = 0$), weakly smooth ($v \in]0, 1[$), and smooth with Lipschitz continuous gradients ($v = 1$) problems.

Definition. 2.1.11. The vector $g \in V^*$ is called a **subgradient** of f at x if $f(x) \in \mathbb{R}$ and

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

for all $y \in V$. The set of all subgradients is called the **(convex) subdifferential** of f at x and is denoted by $\partial f(x)$. The set-valued mapping $\partial f : x \rightarrow \partial f(x)$ is called the subdifferential of f .

In general, the subdifferential $\partial f(x)$ may be empty, or it may consist of just one vector. If $\partial f(x)$ is not empty, then f is called **subdifferentiable** at x . The next result guarantees the existence of subgradients for convex functions under specific conditions on $\text{dom } f$.

Theorem. 2.1.12. (Theorem 23.4 in [147]) Let $f : V \rightarrow \overline{\mathbb{R}}$ be a proper convex function. For $x \notin \text{dom } f$, $\partial f(x)$ is empty. For $x \in \text{ri}(\text{dom } f)$, $\partial f(x)$ is nonempty. Finally, $\partial f(x)$ is nonempty bounded set if and only if $x \in \text{int}(\text{dom } f)$.

Proposition 2.1.13. (SUBDIFFERENTIAL CALCULUS) Let $f : V \rightarrow \overline{\mathbb{R}}$, $f_i : V \rightarrow \overline{\mathbb{R}}$, for $i = 1, \dots, m$ be proper convex functions on V . Then we have

1. (Lemma 3.1.9 in [119]) If $\beta > 0$, then

$$\partial \beta f(x) = \beta \partial f(x).$$

2. (Theorem 23.8 in [147]) Let $f = f_1 + \dots + f_m$. Then

$$\partial f(x) \supseteq \partial f_1(x) + \dots + \partial f_m(x), \quad \text{for all } x,$$

and if $\bigcap_{i=1}^m \text{ri}(\text{dom } f_i) \neq \emptyset$, then

$$\partial f(x) = \partial f_1(x) + \dots + \partial f_m(x), \quad \text{for all } x.$$

3. (Theorem 23.9 in [147]) Let $f(x) = h(Ax)$, where $h : U \rightarrow \overline{\mathbb{R}}$ be a proper convex function and let $A : V \rightarrow U$ be a linear operator. Then

$$\partial f(x) \supseteq A^* \partial h(Ax), \quad \text{for all } x,$$

and if the range of A contains a point of $\text{ri}(\text{dom} h)$, then

$$\partial f(x) = A^* \partial h(Ax), \quad \text{for all } x.$$

4. Let $f(x) = \max_{i=1, \dots, n} f_i(x)$ be pointwise maximum of $f_1, f_2, \dots, f_n$ in x . Then for any

$$x \in \text{int}(\text{dom} f) = \bigcap_{i=1}^m \text{int}(\text{dom} f_i)$$

we have

$$\partial f(x) = \text{conv} \{ \partial f_i(x) \mid i \in I(x) \},$$

where $I(x) = \{i \in 1, \dots, m \mid f_i(x) = f(x)\}$.

5. (see, for example, [96]) Let $f_j : X \rightarrow \overline{\mathbb{R}}$, for $j \in J$, be proper and convex, where J is a finite index set and X is a separated locally convex space. Let $f(x) = \sup_{i \in J} f_i(x)$. Then

$$\partial f(x) = \text{conv} \left(\bigcup_{j \in I(x)} \partial f_j(x) \right),$$

where $I(x) = \{j \in J \mid f_j(x) = f(x)\}$.

We invite the interested readers to see Theorem 2.4.18 in [165] and [91] for more formula regarding the subdifferential of the supremum over an infinite number of functions.

The strongly convex condition (2.4) is equivalent to

$$f(y) \geq f(x) + \langle g, y - x \rangle + \frac{\sigma}{2} \|y - x\|^2, \quad (2.5)$$

for all $x, y \in C$, where g is any subgradient of f at x , i.e., $g \in \partial f(x)$.

Definition. 2.1.14. Let $f : V \rightarrow \overline{\mathbb{R}}$ be a given function. Then the **(Fenchel) conjugate function** of f is defined by

$$f^* : V^* \rightarrow \overline{\mathbb{R}}, \quad f^*(g) = \sup_{x \in V} \{ \langle g, x \rangle - f(x) \}.$$

If f is proper and convex, then f^* is also proper and convex. The next result will be used in Proposition 2.1.17 to derive the subdifferential of some desired functions.

Proposition 2.1.15. (*Fenchel-Young equality, Proposition 13.13 in [24]*) *Let $f : V \rightarrow \overline{\mathbb{R}}$ be proper, and let $x \in V$ and $g \in V^*$. Then*

$$f(x) + f^*(g) \geq \langle g, x \rangle.$$

Moreover, $g \in \partial f(x)$ if and only if

$$f(x) + f^*(g) = \langle g, x \rangle. \quad (2.6)$$

The next result is necessary to give the subdifferential of $\phi(x) = \|Wx\|$.

Lemma. 2.1.16. *Let $\phi : V \rightarrow \overline{\mathbb{R}}$, $\phi(x) = \|Wx\|$, where $\|\cdot\|$ is any norm in the vector space V and $W : U \rightarrow V$ is a linear continuous invertible operator. Then we have*

$$\phi^*(g) = \begin{cases} 0 & \text{if } \|(W^{-1})^* g\|_* \leq 1, \\ +\infty & \text{otherwise.} \end{cases}$$

Proof. By setting $y = Wx$ and the Cauchy-Schwarz inequality, we get

$$\begin{aligned} \phi^*(g) &= \sup_{x \in V} \{ \langle g, x \rangle - \|Wx\| \} = \sup_{y \in U} \{ \langle g, W^{-1}y \rangle - \|y\| \} \\ &= \sup_{y \in U} \{ \langle (W^{-1})^* g, y \rangle - \|y\| \} \leq \sup_{y \in U} \{ \|(W^{-1})^* g\|_* \|y\| - \|y\| \} \\ &= \sup_{y \in U} \left\{ \left(\|(W^{-1})^* g\|_* - 1 \right) \|y\| \right\}. \end{aligned}$$

If $\|(W^{-1})^* g\|_* \leq 1$, then $\phi^*(g) = 0$. If $\|(W^{-1})^* g\|_* > 1$, we have

$$\|(W^{-1})^* g\|_* = \sup_{\|y\| \leq 1} \langle (W^{-1})^* g, y \rangle > 1.$$

Thus there exists $\tilde{y} \in U$ such that $\|\tilde{y}\| \leq 1$ and $\langle (W^{-1})^* g, \tilde{y} \rangle > 1$ leading to

$$\begin{aligned} \phi^*(g) &= \sup_{x \in V} \{ \langle g, x \rangle - \|Wx\| \} = \sup_{y \in U} \{ \langle g, W^{-1}y \rangle - \|y\| \} \\ &= \sup_{y \in U} \{ \langle (W^{-1})^* g, y \rangle - \|y\| \} \geq \sup_{t > 0} \{ \langle (W^{-1})^* g, t\tilde{y} \rangle - \|t\tilde{y}\| \} \\ &= \sup_{t > 0} \left\{ t \left(\langle (W^{-1})^* g, \tilde{y} \rangle - \|\tilde{y}\| \right) \right\} = +\infty, \end{aligned}$$

giving the result. □

We use Lemma 2.1.16 to derive the subdifferential of $\phi(x) = \|Wx\|$ for an arbitrary norm $\|\cdot\|$ in the vector space V and a linear continuous invertible operator W .

Proposition 2.1.17. *Let $\phi : V \rightarrow \overline{\mathbb{R}}$, $\phi(x) = \|Wx\|$, where $W : U \rightarrow V$ is a linear continuous invertible operator and $\|\cdot\|$ is any norm in the vector space V . Then*

$$\partial\phi(x) = \begin{cases} \{g \in V^* \mid \|(W^{-1})^* g\|_* \leq 1\} & \text{if } x = 0, \\ \{g \in V^* \mid \|(W^{-1})^* g\|_* = 1, \langle g, x \rangle = \|Wx\|\} & \text{if } x \neq 0. \end{cases}$$

In particular, if $\|\cdot\|$ is self-dual ($\|\cdot\| = \|\cdot\|_*$), we have

$$\partial\phi(x) = \begin{cases} \{g \in V^* \mid \|(W^{-1})^* g\|_* \leq 1\} & \text{if } x = 0, \\ W^* \frac{Wx}{\|Wx\|} & \text{if } x \neq 0. \end{cases}$$

Proof. If $x = 0$, the Fenchel-Young equality (2.6) and Lemma 2.1.16 imply

$$\phi(0) + \phi^*(g) = \phi^*(g) = \langle g, 0 \rangle = 0,$$

leading to

$$\partial\phi(0) = \left\{ g \in V^* \mid \|(W^{-1})^* g\|_* \leq 1 \right\}.$$

If $x \neq 0$, the Fenchel-Young equality (2.6) implies

$$\phi(x) + \phi^*(g) = \|Wx\| + \phi^*(g) = \langle g, x \rangle,$$

leading to

$$\phi^*(g) = 0, \quad \langle g, x \rangle = \|Wx\|.$$

By setting $y = Wx$ and using the Cauchy-Schwarz inequality, we get

$$\begin{aligned} \|y\| &= \|Wx\| = \langle g, x \rangle = \langle g, W^{-1}y \rangle \\ &= \langle (W^{-1})^* g, y \rangle \leq \|(W^{-1})^* g\|_* \|y\| \leq \|y\| \end{aligned} \tag{2.7}$$

implying

$$\|(W^{-1})^* g\|_* = 1,$$

leading to

$$\partial\phi(x) = \left\{ g \in V^* \mid \|(W^{-1})^* g\|_* = 1, \langle g, x \rangle = \|Wx\| \right\}, \quad \text{for } x \neq 0$$

If $\|\cdot\|$ is self-dual then (2.7) implies

$$\langle g, x \rangle = \langle g, W^{-1}y \rangle = \langle (W^{-1})^* g, y \rangle = \|(W^{-1})^* g\|_* \|y\|,$$

hence $(W^{-1})^* g = \alpha y$ for $\alpha > 0$. Since $\|\cdot\|$ is self-dual and $\|(W^{-1})^* g\|_* = \|(W^{-1})^* g\| = 1$, we obtain

$$1 = \|(W^{-1})^* g\| = \alpha \|y\|,$$

leading to

$$g = W^* \frac{1}{\|y\|} y = W^* \frac{Wx}{\|Wx\|},$$

giving the result. $\square$

In the following example we show how Proposition 2.1.17 is applied for $\phi = \|\cdot\|_\infty$, which will be needed in Chapter 6. The subdifferential of other norms of V can be computed with Proposition 2.1.17 in the same way.

Example. 2.1.18. We use Proposition 2.1.17 to derive the subdifferential of $\phi = \|\cdot\|_\infty$ at arbitrary point x . We first recall that the dual norm $\|\cdot\|_\infty$ is $\|\cdot\|_1$. If $x = 0$, Proposition 2.1.17 implies

$$\partial\phi(0) = \{g \in V^* \mid \|g\|_1 \leq 1\} = \left\{ g \in V^* \mid g = \sum_{i=1}^n \beta_i e_i, \beta \in [-1, 1], \sum_{i=1}^n |\beta_i| \leq 1 \right\}.$$

Then we have

$$\begin{aligned} \partial\phi(x) &= \left\{ g \in V^* \mid \|g\|_1 = 1, \langle g, x \rangle = \|x\|_\infty = \max_{1 \leq i \leq n} |x_i| \right\} \\ &= \left\{ g \in V^* \mid \sum_{j=1}^n |g_j| = 1, \sum_{j=1}^n g_j x_j = \|x\|_\infty \right\} \end{aligned}$$

If $x \neq 0$, we set

$$I := \{i \in \{1, \dots, n\} \mid \|x\|_\infty = |x_i|\}$$

and we have $\|x\|_\infty = \sum_{i \in I} \beta_i |x_i|$ and $\sum_{i \in I} \beta_i = 1$ leading to

$$\partial\phi(x) = \left\{ g \in V^* \mid g = \sum_{i \in I} \beta_i \operatorname{sign}(x_i) e_i, \sum_{i \in I} \beta_i = 1 \right\}.$$

2.2 Optimality conditions

Let $f : V \rightarrow \overline{\mathbb{R}}$ be a proper, convex, and nonsmooth function. Then **Fermat's optimality condition** for the nonsmooth convex optimization problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in C \end{aligned} \tag{2.8}$$

is given by

$$0 \in \partial f(x) + N_C(x), \quad (2.9)$$

where $N_C(x)$ is the normal cone of C at x , see, for example, Theorem 4.8 by BAGIROV et al. in [19]. Note that a global minimizer of (2.8) is a $\hat{x} \in C$ such that

$$f(\hat{x}) \leq f(x) \quad \text{for all } x \in C.$$

The minimum is $\hat{f} := f(\hat{x})$, and the optimal set is defined by

$$\hat{X} := \{x \in C \mid f(x) = \hat{f}\}.$$

If in addition f is strictly convex, the minimizer $\hat{x}$ is unique. In particular, the **proximal-like** operator $\text{prox}_{\lambda f}^C(y)$ is defined by the unique minimizer of the optimization problem

$$\text{prox}_{\lambda f}^C(y) := \underset{x \in C}{\operatorname{argmin}} \frac{1}{2} \|x - y\|_2^2 + \lambda f(x), \quad (2.10)$$

where $\lambda > 0$. The first-order optimality condition (2.9) for the problem (2.10) is given by

$$0 \in x - y + \lambda \partial f(x) + N_C(x). \quad (2.11)$$

If $C = V$, then (2.11) is simplified to

$$0 \in x - y + \lambda \partial f(x), \quad (2.12)$$

giving the classical proximity operator. For a nonempty, closed, and convex set $C \subseteq V$, the **orthogonal projection** of $y \in V$ onto C is defined by

$$P_C(y) := \underset{x \in C}{\operatorname{argmin}} \frac{1}{2} \|x - y\|^2, \quad (2.13)$$

where the first-order optimality condition (2.9) for the problem (2.13) is given by

$$0 \in x - y + N_C(x). \quad (2.14)$$

If the orthogonal projection onto the set C is efficiently available in a closed form or by a simple iterative scheme, then C is called **simple convex domain**. Since the term $\frac{1}{2} \|x - y\|^2$ is a strongly convex function, then $P_C(y)$ is unique.

For smooth problems, since $\partial f(x_b)$ is singleton (see Corollary 2.4.10 in [165]), we can check the optimality of the current point x_b by checking the optimality condition (2.9). However, for nonsmooth problem $\partial f(x)$ is a set of vectors which is not typically available, i.e., it cannot be used for checking the optimality of the point x_b . Therefore, it is natural to check the condition

$$0 \leq f(x_b) - \hat{f} \leq \varepsilon,$$

for a given accuracy parameter ε . The minimizer of the problem (2.8) is called **ε -solution**, which is not necessarily unique if f is not strictly convex. In the following we give a definition that is useful to compare the efficiency of iterative schemes.

Definition. 2.2.1. *The number of calls of oracle $\mathcal{O}(x)$ required to attain an ε -solution is called **analytical complexity**. In addition, The total number of arithmetic operations (including work of oracle and work of the method) to reach an ε -solution is called **arithmetical complexity**.*

2.3 Information-based optimization

Let us consider an optimization problem of the form (2.8), where the objective function is not analytically known, i.e., the minimizer can not be computed analytically. More precisely, the objective function is like a black box that we do not know about its structure and we can only observe which parameters pass to the function and what come out. It is worth noting while information-based optimization are originally developed for problems with unknown objective functions, they are sometimes applied successfully to problems with available analytical form.

In order to recognize the objective function of (2.8) and solve it, we need a unit to get the parameters and collect outcomes of the objective function, which is called **oracle** denoted by $\mathcal{O}(x)$. Therefore, oracle is the only source of information about the objective function. This restricts our possibilities to design a suitable solver. On the basis of the level of information given by oracle, the information-based optimization problems are typically categorized as follows:

1. Problems with **zero-order oracle** $\mathcal{O}(x) = (f_x)$ (only function values are available);
2. Problems with **first-order oracle** $\mathcal{O}(x) = (f_x, g_x)$ (function values and subgradients (gradients) are available);
3. Problems with **second-order oracle** $\mathcal{O}(x) = (f_x, g_x, H_x)$ (function values, subgradients (gradients), and Hessian are available).

The algorithms that can handle the above-mentioned classes of problems are called zero-order (black-box, derivative-free), first-order, and second-order methods, respectively. In the following we give the most famous examples of each class:

1. **Zero-order methods:** direct search methods [67, 146]; line search methods based on simplex derivation [67]; trust-region methods (derivative-free or interpolation-based models) [67];

2. **First-order methods:** we divide this class of solvers into two subclasses with respect to the differentiability of objectives as:
 - **Smooth methods:** gradient methods [36, 134]; gradient projection methods [36, 134]; incremental gradient methods [35, 149]; conjugate gradient methods [134]; spectral gradient methods [38, 144]; Nesterov-type optimal methods [118, 119];
 - **Nonsmooth methods:** subgradient methods [36, 119, 122, 139, 148]; subgradient projection methods [36, 44, 119, 139, 148]; dual subgradient methods [36, 68, 94, 139]; incremental subgradient methods [116]; cutting-plane methods [119]; bundle methods [62, 94, 99, 100, 105, 112]; proximal methods [138]; primal-dual methods [24, 55, 61]; Mirror descent methods [25, 26, 117]; Nesterov-type optimal methods (composite problems) [123, 124, 125]; forward-backward splitting methods [24, 28]; forward-backward-forward splitting methods [24]; Douglas-Rachford methods [24, 41]; Smoothing methods [29, 40, 42, 73];
3. **Second-order methods:** Newton methods [134]; Newton-type methods [134]; quasi-Newton methods [134]; interior-point methods [126]; trust-region methods [3, 8, 66];

Over the past few decades, due to the dramatic increase in the size of data in applications, developing optimization schemes that can handle such problems get lots of attention. Among the above-mentioned methods, second-order methods need to compute and save Hessian or its approximation, and so they are not appropriate to deal with problems involving high-dimensional or big data. However, zero-order and first-order methods have simple structures and low memory requirement that makes them suitable to handle large problems. Between zero-order and first-order methods, first-order methods typically produce much more reliable analysis and computational results. Therefore, in this thesis, we only address first-order methods and assume that the first-order oracle (function values and subgradients) of the objective function is available.

2.4 First-order methods

As discussed in Section 2.3, first-order methods are a class of iterative methods for solving optimization problems that need to access the first-order oracle $\mathcal{O}(x) = (f_x, g_x)$, where $f_x = f(x)$ and $g_x \in \partial f(x)$. If the objective is convex, they are sufficient to achieve a minimizer. Generally, first-order methods have simple structures with the low memory requirement. Thanks to these features, first-order

methods have received much attention during the past few decades. We already addressed several first-order methods in Section 2.3 for both smooth and nonsmooth problems. Historically, gradient descent and subgradient methods were the first numerical schemes proposed to solve optimization problems with smooth and nonsmooth convex objective functions, respectively. The gradient descent method generates a sequence of iteration of the form

$$x_{k+1} = x_k - \alpha_k g_{x_k}, \quad (2.15)$$

where k is the iteration counter and $\alpha_k \in (0, 1]$ is a step-size computed by a monotone or nonmonotone line search, see, for example, [4, 7, 14, 134]. In practice, first-order methods are too slow and perform very poor for badly-scaled and ill-conditioned problems.

Subgradient methods have been developed to solve convex nonsmooth problems, dating back to 1960, see, for example, [139, 148]. Subgradient methods generate the same sequence as (2.15) where $g_{x_k} \in \partial f(x_k)$ and α is a step-size. Subgradient projection iterations for a problem of the form (2.8) are given by

$$x_{k+1} = P_C(x_k - \alpha_k g_{x_k}) \quad (2.16)$$

where $g_{x_k} \in \partial f(x_k)$ and α for both subgradient and subgradient projection methods is typically computed by one of the following terms:

1. **Constant step-size**, i.e., $\alpha_k = \alpha$ for $\alpha \in \mathbb{R}$;
2. **Constant step-length**, i.e., $\alpha_k = \alpha / \|g_{x_k}\|$ for $\alpha \in \mathbb{R}$;
3. **Square summable but not summable step-size**, i.e.,

$$\alpha_k \geq 0, \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

4. **Nonsummable diminishing step-size (NDSS)**, i.e.,

$$\alpha_k \geq 0, \quad \lim_{k \rightarrow \infty} \alpha_k = 0, \quad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

5. **Nonsummable diminishing step-length (NDSL)**, i.e.,

$$\alpha_k = \gamma_k / \|g_{x_k}\|, \quad \gamma \geq 0, \quad \lim_{k \rightarrow \infty} \gamma_k = 0, \quad \sum_{k=1}^{\infty} \gamma_k = \infty.$$

It is notable that step-sizes for subgradient methods do not need any line search in contrast to those of the gradient descent. Subgradient methods only need function values and subgradients, and not only inherit the basic features of general first-order methods such as low memory requirement and simple structure but also able to deal with general convex optimization problems without considering their structures (which is not the case for proximal or Nesterov-type optimal methods). This fact yields that they are suitable for solving convex problems in applications involving high-dimensional data, say several millions of variables. In practice, subgradient methods with the step-sizes computed by NDSS and NDSL perform better than the others. Since the step-sizes of NDSS and NDSL diminishes by increasing the number of iterations, the associated subgradient methods suffer from slow convergence rate, which finally limits the attainable accuracy.

2.5 Optimal first-order methods

It is known that the efficiency of first-order methods are highly related to the features of the underlying objective function and the feasible set C . For example solving nonsmooth problems is much harder than solving smooth problems, and projection into some convex set is expensive. This means that the efficiency of first-order methods is poor for solving the class of convex problems, because this class includes “bad guys”¹ whose minimum cannot be approximated without computing a large number of function values and subgradients. As a result, to have practically appealing efficiency results, it is necessary to make additional restriction on class of problems.

In 1983, NEMIROVSKI & YUDIN in [117] derived optimal worst-case complexity bounds of first-order methods to achieve an ε -solution for several classes of problems. If an algorithm attains the worst-case complexity bound for a class of problems, it is called *optimal*. In Table 2.1, we consider some classes of problems and give the corresponding optimal complexity bounds. In [117], it was proved that subgradient, subgradient projection, and mirror descent methods attain the optimal complexity of first-order methods for solving Lipschitz continuous nonsmooth problems, where the mirror decent method is a generalization of the subgradient projection method, see [26, 25].

The pioneering work on optimal first-order methods dates back to NESTEROV [118] in 1983 (see Algorithm 1). This method is optimal for smooth problems with Lipschitz continuous gradients producing interesting theoretical and computational results.

¹HIRIART-URRUTY & LEMARECHAL [94], “bad guy” constructed by Nemirovskii

Table 2.1: List of several classes of problems and associated optimal complexity bounds for first-order methods to reach an ε -solution

Problem's class	Complexity bound
Lipschitz continuous nonsmooth problems	$O(\varepsilon^{-2})$
Lipschitz continuous smooth problems	$O(\varepsilon^{-2})$
Strongly convex problems	$O(\log(\varepsilon^{-1}))$
Weakly smooth problems with Hölder continuous gradients ($\nu \in (0, 1)$)	$O(\varepsilon^{-2/(1+3\nu)})$
Smooth problems with Lipschitz continuous gradients	$O(\varepsilon^{-1/2})$

Algorithm 1: NES83 (Nesterov's 1983 optimal method)

Input: select $z \neq y_0$ and $g_{y_0} \neq g_z$, $\varepsilon > 0$;

Output: x_k, f_{x_k} ;

```

1 begin
2    $a_0 = 0$ ;  $x_{-1} = y_0$ ;  $\alpha_{-1} = \|y_0 - z\| / \|g_{y_0} - g_z\|$ ;
3   while stopping criteria do not hold do
4      $\hat{\alpha}_k = \alpha_{k-1}$ ;  $\hat{x}_k = y_k - \hat{\alpha}_k g_{y_k}$ ;
5     while  $f_{\hat{x}_k} < f_{y_k} - \frac{1}{2} \hat{\alpha}_k \|g_{y_k}\|^2$  do
6        $\hat{\alpha}_k = \rho \hat{\alpha}_k$ ;  $\hat{x}_k = y_k - \hat{\alpha}_k g_{y_k}$ ;
7     end
8      $x_k = \hat{x}_k$ ;  $f_{x_k} = f_{\hat{x}_k}$ ;  $\alpha_k = \hat{\alpha}_k$ ;
9      $a_{k+1} = (1 + \sqrt{4a_k^2 + 1}) / 2$ ;
10     $y_{k+1} = x_k + (a_k - 1)(x_k - x_{k-1}) / a_{k+1}$ ;
11  end
12 end
```

NESTEROV in [119] proposed some gradient methods for solving smooth problems with Lipschitz continuous gradient attaining the complexity $O(\varepsilon^{-1/2})$. He later in [123, 124] proposed two gradient-type methods for minimizing a sum of two functions (composite problems) with the complexity $O(\varepsilon^{-1/2})$, where, for the first method, the smooth part of the objective needs to have Lipschitz continuous gradients and, for the second one, the smooth part of the objective needs to have Hölder continuous gradients. Since 1983 many researchers have worked in the development of optimal schemes, see, for example AUSLANDER & TEBoulLE [16], BAES [17], BAES & BÜRGISSEr [18], BECK & TEBoulLE [28], DEVOLDEr et al. [74], GONZAGA et al. [86, 87], LAN [105], LAN et al. [106], NESTEROV [121, 120, 123], NEUMAIEr [130] and TSENG [153]. Compu-

tational comparisons for composite functions show that optimal Nesterov-type first-order methods are substantially superior to the gradient descent and subgradient methods, see, for example, AHOOKHOSH [1] and BECKER et al. [30]. NESTEROV also in [121, 120] proposed some smoothing methods for structured nonsmooth problems attaining the complexity of the order $O(\varepsilon^{-1/2})$. Smoothing methods also have been studied by many authors, see, for example, BECK & TEBoulLE in [28], DEVOLDER et al. [73], and BOT & HENDRICH in [42, 40].

NESTEROV in [119, 122] proposed primal-dual subgradient schemes, which attain the the complexity bound $O(\varepsilon^{-2})$ for Lipschitz continuous nonsmooth problems. JUDITSKY & NESTEROV in [97] proposed a primal-dual subgradient scheme for uniformly convex functions with an unknown convexity parameter, which attains the complexity close to the optimal bound. The slow convergence rates of gradient and subgradient methods can be addressed by their worst-case complexity bounds to achieve an ε -solution, where the gradient descent method attains the complexity $O(\varepsilon^{-1})$ for smooth problems and the subgradient methods attain the complexity $O(\varepsilon^{-2})$.

Recently, NEUMAIER in [130] proposed an optimal subgradient algorithm (OSGA) attaining the complexity $O(\varepsilon^{-2})$ for Lipschitz continuous nonsmooth problems and $O(\varepsilon^{-1/2})$ for smooth problems with Lipschitz continuous gradients at the same time. We describe OSGA and a variant of it, together with their convergence analysis, in Chapter 3.

Chapter 3

Optimal subgradient algorithms: basic idea & derivation

In this chapter we describe two subgradient algorithms with optimal complexity for solving convex optimization problems based on the optimal subgradient framework introduced by NEUMAIER in [130]. More specifically, a rational auxiliary problem is defined by a fraction of a linear relaxation of the objective over a prox-function. The first algorithm (single-projection OSGA) needs a single solving the auxiliary problem and the second one (double-projection OSGA) needs two times solving the auxiliary problem. It is proved that the single-projection OSGA is optimal for nonsmooth Lipschitz continuous problems, where the double-projection OSGA is optimal for both nonsmooth Lipschitz continuous problems and smooth problems with Lipschitz continuous gradients. The convergence of the sequence of iterations $\{x_k\}$ for both algorithms is shown if the objective is strictly convex.

3.1 Novel subgradient framework

In this section we give the basic idea behind of the optimal subgradient framework for solving convex constrained optimization problems.

Let us consider the convex constrained minimization problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in C, \end{aligned} \tag{3.1}$$

where $f : C \rightarrow \overline{\mathbb{R}}$ is a proper and convex function defined on a nonempty, closed, and convex subset C of V . The main objective is to find a solution $\hat{x} \in C$ by using the first-order information, i.e., function values and subgradients.

We consider linear relaxations of f at $x \in C$ given by

$$f(x) \geq \gamma + \langle h, x \rangle \quad \text{for all } x \in C, \tag{3.2}$$

where $\gamma \in \mathbb{R}$ and $h \in V^*$. This gives a global underestimator of f at $x \in C$, which is readily available due to the definition of subgradients at x . We also consider a continuously differentiable prox-function $Q : C \rightarrow \mathbb{R}$ that is strongly convex function (2.5) with the convexity parameter $\sigma = 1$ and satisfying

$$Q_0 := \inf_{x \in C} Q(x) > 0 \quad (3.3)$$

Afterwards, a sequence of minimization problems of the form

$$\begin{aligned} \sup \quad & E_{\gamma,h}(x) \\ \text{s.t.} \quad & x \in C, \end{aligned} \quad (3.4)$$

is solved, where it is known that the supremum is positive. The function $E_{\gamma,h} : C \rightarrow \mathbb{R}$ is defined by

$$E_{\gamma,h}(x) := -\frac{\gamma + \langle h, x \rangle}{Q(x)}. \quad (3.5)$$

The function $E_{\gamma,h}$ is differentiable and the solution of the subproblem (3.4) can be computed cheaply for many domains C . If $u := U(\gamma, h) \in C$ is the solution of this problem, then it is assumed that $e := E(\gamma, h)$ and $u = U(\gamma, h)$ are readily computable.

Proposition 3.1.1. *Let $\gamma_b = \gamma - f(x_b)$. Then we have*

$$0 \leq f(x_b) - \hat{f} \leq \eta Q(\hat{x}), \quad (3.6)$$

where $\hat{x}$ is the maximizer, $\hat{f}$ is the maximum, and x_b is the best known point. In particular, if x_b is not yet optimal, then we have $E(\gamma_b, h) > 0$.

Proof. See Proposition 2.1 in [130]. $\square$

In view of Proposition 3.1.1, if an upper bound for $Q(\hat{x})$ is known or assumed, the bound (3.6) translates into a computable error estimate for the minimal function value. But even in the absence of such an upper bound, the optimization problem (3.1) can be solved to a target accuracy

$$0 \leq f(x_b) - \hat{f} \leq \varepsilon Q(\hat{x}) \quad (3.7)$$

if one manages to decrease the error factor η from its initial value until $\eta \leq \varepsilon$ for some target tolerance $\varepsilon > 0$. In the remainder of this section, we introduce two algorithms that decrease the error factor η monotonically. For these algorithms, we shall prove the complexity bounds on the number of iterations needed to attain an ε -solution that are independent of the dimension of V (which may be infinite), and – apart from a constant factor – best possible under a variety of assumptions on the objective function, cf. [117, 119].

Proposition 3.1.2. *Let $\eta = E(\gamma, h) > 0$ and $u = U(\gamma, h)$. Then*

$$\gamma + \langle h, u \rangle = -\eta Q(u), \quad (3.8)$$

$$\langle \eta g_Q(u) + h, z - u \rangle \geq 0 \quad \text{for all } z \in C, \quad (3.9)$$

$$\gamma + \langle h, z \rangle \geq \eta \left(\frac{1}{2} \|z - u\|^2 - Q(z) \right) \quad \text{for all } z \in C. \quad (3.10)$$

Proof. See Proposition 2.2 in [130]. $\square$

In the remainder of this section we explain how the linear relaxations (3.2) is constructed for convex and strongly convex objectives and describe a scheme to determine step-sizes, which guarantees attaining the optimal complexity bounds. We will propose several prox-functions due to features of domains C guaranteeing that the subproblem (3.4) is solved in closed form or by a simple scheme.

3.1.1 Linear relaxations

The definition of subgradients of the objective f at the point $x_b \in C$ implies

$$f(x) \geq f(x_b) + \langle g(x_b), x - x_b \rangle \quad \text{for all } x \in C, \quad (3.11)$$

where $g(x_b)$ denotes a subgradient of f at x_b . By setting $f_{x_b} := f(x_b)$, $g_{x_b} := g(x_b)$, and

$$\gamma := f(x_b) - \langle g_{x_b}, x_b \rangle, \quad h := g_{x_b}, \quad (3.12)$$

the condition (3.2) is always valid. More general relaxations can be found by accumulating past information. The convex combination of γ and h used in (3.2) and those in (3.12) at x gives

$$\bar{\gamma} := \gamma + \alpha(f(x) - \langle g_x, x \rangle - \gamma), \quad \bar{h} := h + \alpha(g_x - h), \quad (3.13)$$

where $\alpha \in [0, 1]$ is a step-size. From (3.2) and (3.13), we obtain

$$\begin{aligned} f(x) &= (1 - \alpha)f(x) + \alpha f(x) \geq (1 - \alpha)(\gamma + \langle h, x \rangle) + \alpha(f(x) + \langle g_x, x - x \rangle) \\ &= (1 - \alpha)\gamma + \alpha(f(x) - \langle g_x, x \rangle) + \langle (1 - \alpha)h + \alpha g_x, x \rangle \\ &= \bar{\gamma} + \langle \bar{h}, x \rangle, \end{aligned}$$

which gives a linear underestimator of f in the form (3.2) at x .

3.1.2 Step-size selection

The step-size parameter α controls the fraction of the new information (3.11) incorporated into the new relaxation. It is chosen with the hope for a reduction factor of approximately $1 - \alpha$ in the current error factor η , and must therefore be adapted to the actual progress made.

First we note that in practice, $Q(\hat{x})$ is unknown; hence the numerical value of η is meaningless in itself. However, quotients of η at different iterations have a meaning, quantifying the amount of progress made.

In the following, we use bars to denote quantities tentatively modified in the current iteration, but they replace the current values of these quantities only if an acceptance criterion is met that we now motivate. We measure progress in terms of the quantity

$$R := \frac{\eta - \bar{\eta}}{\delta \alpha \eta}, \quad (3.14)$$

where $\delta \in]0, 1[$ is a fixed number. A value $R \geq 1$ indicates that we made sufficient progress in that

$$\bar{\eta} := (1 - \lambda R \alpha) \eta \quad (3.15)$$

was reduced at least by a fraction δ of the designed improvement of η by $\alpha \eta$; thus the step-size is acceptable or may even be increased if $R > 1$. On the other hand, if $R < 1$, the step-size must be reduced significantly to improve the chance of reaching the design goal. Introducing a maximal step-size $\alpha_{\max} \in]0, 1[$ and two parameters with $0 < \kappa' \leq \kappa$ to control the amount of increase or decrease in α , we update the step-size according to

$$\bar{\alpha} := \begin{cases} e^{-\kappa} & \text{if } R < 1 \\ \min(\alpha e^{\kappa'(R-1)}, \alpha_{\max}) & \text{if } R \geq 1. \end{cases} \quad (3.16)$$

Updating the linear relaxation and u makes sense only when η was improved. This suggests the following update scheme, in which α is always modified, while h, γ, η , and u are changed only when $\bar{\eta} < \eta$; if this is not the case, the barred quantities are simply discarded.

If $\alpha_{\min}$ denotes the smallest actually occurring step-size (which is not known in advance), we have global linear convergence with a convergence factor of $1 - e^{-\kappa} \alpha_{\min}$. However, $\alpha_{\min}$ and hence this global rate of convergence may depend on the target tolerance ε ; thus the convergence speed in the limit $\varepsilon \rightarrow 0$ may be linear or sublinear depending on the properties of the specific function minimized.

Algorithm 2: PUS (parameters updating scheme)**Input:** $\delta, \alpha_{\max} \in]0, 1[, 0 < \kappa' \leq \kappa, \alpha, \eta, \bar{h}, \bar{\gamma}, \bar{\eta}, \bar{u};$ **Output:** $\alpha, h, \gamma, \eta, u;$

```

1 begin
2    $R \leftarrow (\eta - \bar{\eta})/(\delta\alpha\eta);$ 
3   if  $R < 1$  then
4      $h \leftarrow \bar{h};$ 
5   else
6      $\bar{\alpha} = \min(\alpha e^{\kappa'(R-1)}, \alpha_{\max});$ 
7   end
8    $\alpha \leftarrow \bar{\alpha};$ 
9   if  $\bar{\eta} < \eta$  then
10     $h \leftarrow \bar{h}; \gamma \leftarrow \bar{\gamma}; \eta \leftarrow \bar{\eta}; u \leftarrow \bar{u};$ 
11  end
12 end

```

3.1.3 Strongly convex relaxations

If f is a strongly convex function, then we may know $\mu > 0$ such that $f - \mu Q$ is convex. For this function, the definition of the subgradient at x_b implies

$$f(z) - \mu Q_z \geq f(x_b) - \mu Q(x_b) + \langle g_{x_b} - \mu g_Q(x_b), z - x_b \rangle \quad \text{for all } z \in C, \quad (3.17)$$

where $g_Q(x_b)$ denotes the gradient of Q at x_b . This leads to strongly convex relaxations of the form

$$f(z) \geq \gamma + \langle h, z \rangle + \mu Q(z) \quad \text{for all } z \in C \quad (3.18)$$

with

$$\gamma = f(x_b) - \mu Q(x_b) - \langle g_{x_b}, x_b \rangle, \quad h = g_{x_b} - \mu g_Q(x_b). \quad (3.19)$$

In this case a more general linear relaxation with accumulated information is defined similar to (3.13). The details are summarized in the next result.

Proposition 3.1.3. *Let $x \in C$, $\alpha \in [0, 1]$, and*

$$\bar{\gamma} := \gamma + \alpha(f(x) - \mu Q(x) - \langle g_x, x \rangle - \gamma), \quad \bar{h} := h + \alpha(g - h), \quad g = g_x - \mu g_Q(x).$$

If (3.18) holds and $f - \mu Q$ is convex, then we have

$$f(z) \geq \bar{\gamma} + \langle \bar{h}, z \rangle + \mu Q(z) \quad \text{for all } z \in C. \quad (3.20)$$

Proof. See Proposition 3.2 of [130]. □

Using the relaxations (3.18), we give an error bound in the following result.

Proposition 3.1.4. *Let*

$$\gamma_b := \gamma - f(x_b), \quad \eta := E(\gamma_b, h) - \mu.$$

Then (3.18) implies

$$0 \leq f(x_b) - \hat{f} \leq \eta Q(\hat{x}).$$

Proof. See Proposition 3.3 of [130]. $\square$

Notice that the strongly convex parameter μ is positive, but we used $\mu \geq 0$ to cover the case that either the objective is only convex or for a strongly convex function f that μ is not available, where we set $\mu = 0$.

3.2 Optimal subgradient algorithms

In the context of nonsmooth optimization, it is known that the subgradient at the current point x_b does not always determine a descent direction (converse to the gradient that always gives a decent direction), see, for example, [139, 148]. However, we may hope to find better points by moving along subgradient directions. Therefore, it is typical to keep track on the best point so far, i.e., the one with the smallest function value. In view of this and the basic idea given the previous section, we describe two subgradient algorithms with the optimal complexity. More precisely, the first algorithm (single-projection OSGA) only needs a single solving of the subproblem (3.4) and the second one (double-projections OSGA) needs two times solving the subproblem (3.4).

3.2.1 Single-projection OSGA

In this section we give an algorithm for solving the optimization problems of the form (3.1), which needs a single solving of the rational subproblem (3.4).

Suppose that the solution u of the subproblem (3.4) is given. We generate the new point by a convex combination of x_b and u , i.e.,

$$x := x_b + \alpha(u - x_b), \tag{3.21}$$

where $\alpha \in [0, 1]$ is a step-size given by Algorithm 2. After producing the point x , we should update the linear relaxation given in Proposition 3.1.3. Since our linear relaxations (3.20) (and relatively the function $E_{\gamma, h}$ (3.5)) are constructed based on the subgradient information, we cannot guarantee that the direction $x_b - u$ is descent. Hence, similar to classical subgradient methods, we should keep track the best point so far leading to the point

$$x'_b := \operatorname{argmin}_{z \in \{x_b, x\}} f(z).$$

Afterwards, we update the linear relaxation information based on the new point x'_b given in Proposition 3.1.3, solve the subproblem (3.4) to attain the new trial step u' , and produce the new point

$$x' := x'_b + \alpha(u - x'_b),$$

which is a convex combination of x'_b and u' . The new x_b is produced in such a way guaranteeing

$$f_{x_b} \leq \min\{f_{x'_b}, f_{x'}\}.$$

Then we update the parameters α , h , γ , η , and u using Algorithm 2 and continue the procedure until a stopping criterion holds.

Summarizing the above-mentioned discussion, we give the single-projection optimal subgradient algorithm as follows:

Algorithm 3: OSGA-V (single-projection optimal subgradient algorithm)

Input: $\delta, \alpha_{\max} \in]0, 1[$, $0 < \kappa' \leq \kappa$; local parameters: $x_0, \mu \geq 0, f_{\text{target}}$;

Output: x_b, f_{x_b} ;

1 **begin**

2 choose an initial best point x_b ; compute f_{x_b} and g_{x_b} ;

3 **if** $f_{x_b} \leq f_{\text{target}}$ **then**

4 stop;

5 **else**

6 $h = g_{x_b} - \mu g_Q(x_b)$; $\gamma = f_{x_b} - \mu Q(x_b) - \langle h, x_b \rangle$;

7 $\gamma_b = \gamma - f_{x_b}$; $u = U(\gamma_b, h)$; $\eta = E(\gamma_b, h) - \mu$;

8 **end**

9 $\alpha = \alpha_{\max}$;

10 **while** *stopping criteria do not hold* **do**

11 $x = x_b + \alpha(u - x_b)$; compute f_x and g_x ;

12 $g = g_x - \mu g_Q(x)$; $\bar{h} = h + \alpha(g - h)$;

13 $\bar{\gamma} = \gamma + \alpha(f_x - \mu Q(x) - \langle g, x \rangle - \gamma)$; $x'_b = \operatorname{argmin}_{z \in \{x_b, x\}} \{f(z)\}$;

14 $\gamma'_b = \bar{\gamma} - f_{x'_b}$; $u' = U(\gamma'_b, \bar{h})$; $\eta = E(\gamma'_b, \bar{h}) - \mu$; $x' = x'_b + \alpha(u' - x'_b)$;

15 choose x_b in such a way that $f_{x_b} \leq \min\{f_{x'_b}, f_{x'}\}$;

16 **if** $f_{x_b} \leq f_{\text{target}}$ **then**

17 stop;

18 **else**

19 update the parameters α , h , γ , η and u using UPS;

20 **end**

21 **end**

22 **end**

The following inequality is needed in Section 3.3 to establish the complexity of Algorithm 3.

Theorem. 3.2.1. *In Algorithm 3, the error factors are related by*

$$\bar{\eta} - (1 - \alpha)\eta \leq \frac{\alpha^2 \|g(x)\|_*^2}{2(1 - \alpha)(\eta + \mu)Q_0}, \quad (3.22)$$

where $\|\cdot\|_*$ denotes the dual norm of $\|\cdot\|$.

Proof. We first establish some inequalities needed for the later estimation. By convexity of Q and the definition of $\bar{h}$,

$$\begin{aligned} \alpha\mu \left(Q(u') - Q(x) + \langle g_Q(x), x \rangle \right) &\geq \alpha\mu \langle g_Q(x), u' \rangle = \langle h - \bar{h} + \alpha(g(x) - h), u' \rangle \\ &= (1 - \alpha) \langle h, u' \rangle + \langle \alpha g(x) - \bar{h}, u' \rangle. \end{aligned}$$

By the definition of x , we have

$$(1 - \alpha)(x_b - x) = -\alpha(u - x).$$

Hence (3.17) (with $\mu = 0$) implies

$$(1 - \alpha)(f(x_b) - f(x)) \geq (1 - \alpha) \langle g(x), x_b - x \rangle = -\alpha \langle g(x), u - x \rangle.$$

By the definition of $\bar{\gamma}$, we conclude from these two inequalities that

$$\begin{aligned} \bar{\gamma} - f(x) + \alpha\mu Q(u') &= (1 - \alpha)(\gamma - f(x)) - \alpha \langle g(x), x \rangle \\ &\quad + \alpha\mu \left(Q(u') - Q(x) + \langle g_Q(x), x \rangle \right) \\ &\geq (1 - \alpha) \left(\gamma - f(x) + \langle h, u' \rangle \right) + \alpha \langle g(x), u' - x \rangle - \langle \bar{h}, u' \rangle \\ &\geq (1 - \alpha) \left(\gamma - f(x_b) + \langle h, u' \rangle \right) + \alpha \langle g(x), u' - u \rangle - \langle \bar{h}, u' \rangle. \end{aligned}$$

Then this, (3.8) (with $\bar{\gamma}_b = \bar{\gamma} - f(x'_b)$ in place of γ and $\bar{h}$ in place of h), and $E(\bar{\gamma}_b, \bar{h}) = \bar{\eta} + \mu$ give

$$\begin{aligned} (\bar{\eta} + \mu - \alpha\mu)Q(u') &= f(x'_b) - \bar{\gamma} - \langle \bar{h}, u' \rangle - \alpha\mu Q(u') \\ &\leq f(x'_b) - f(x) - \alpha \langle g(x), u' - u \rangle \\ &\quad - (1 - \alpha) \left(\gamma - f(x_b) + \langle h, u' \rangle \right). \end{aligned} \quad (3.23)$$

Since $E(\gamma_b, h) > 0$ by Proposition 3.1.2, we may use (3.10) with $\gamma_b = \gamma - f(x_b)$ in place of γ and $\eta + \mu = E(\gamma_b, h)$, and find

$$(\eta + \mu)Q(u') \geq f(x_b) - \gamma - \langle h, u' \rangle + \frac{\eta + \mu}{2} \|u' - u\|^2. \quad (3.24)$$

Now (3.23) and (3.24) imply

$$\begin{aligned}
(\bar{\eta} - (1 - \alpha)\eta)Q(u') &= (\bar{\eta} + \mu - \alpha\mu)Q(u') - (1 - \alpha)(\eta + \mu)Q(u') \\
&\leq f(x'_b) - f(x) - (1 - \alpha)\left(\gamma - f(x_b) + \langle h, u' \rangle\right) \\
&\quad - \alpha\langle g(x), u' - u \rangle \\
&\quad - (1 - \alpha)\left(f(x_b) - \gamma - \langle h, u' \rangle + \frac{\eta + \mu}{2}\|u' - u\|^2\right) \\
&= f(x'_b) - f(x) + \bar{S},
\end{aligned}$$

where

$$\begin{aligned}
\bar{S} &:= -\alpha\langle g(x), u' - u \rangle - \frac{(1 - \alpha)(\eta + \mu)}{2}\|u' - u\|^2 \\
&\leq \alpha\|g(x)\|_*\|u' - u\| - \frac{(1 - \alpha)(\eta + \mu)}{2}\|u' - u\|^2 \\
&= \frac{\alpha^2\|g(x)\|_*^2 - (\alpha\|g(x)\|_* + (1 - \alpha)(\eta + \mu)\|u' - u\|)^2}{2(1 - \alpha)(\eta + \mu)} \\
&\leq \frac{\alpha^2\|g(x)\|_*^2}{2(1 - \alpha)(\eta + \mu)}.
\end{aligned} \tag{3.25}$$

If $\bar{\eta} \leq (1 - \alpha)\eta$ then (3.22) holds trivially. Now let $\bar{\eta} > (1 - \alpha)\eta$. Then

$$(\bar{\eta} - (1 - \alpha)\eta)Q_0 \leq (\bar{\eta} - (1 - \alpha)\eta)Q(u') \leq f(x'_b) - f(x) + \bar{S}. \tag{3.26}$$

Since $f(x'_b) \leq f(x)$, we conclude that (3.22) holds. Thus (3.22) holds generally. $\square$

3.2.2 Double-projections OSGA

In this section we describe the optimal subgradient algorithms proposed by NEUMAIER in [130] for solving the optimization problems of the form (3.1) needing two times solving of the rational subproblem (3.4).

Let us assume that u is the solution of the subproblem (3.4). We generate x by (3.21) and update the linear relaxation given in Proposition (3.1.3). We set

$$x'_b := \operatorname{argmin}_{z \in \{x_b, x\}} f(z)$$

and update the linear relaxation given in Proposition 3.1.3. We then solve the subproblem (3.4) to generate u' and produce the new point

$$x' := x_b + \alpha(u' - x_b),$$

which is a convex combination of x_b and u' . Then we produce $\bar{x}_b$ by

$$\bar{x}_b := \operatorname{argmin}_{z \in \{x'_b, x'\}} f(z),$$

and update the linear relaxation given in Proposition 3.1.3. The new x_b is produced by

$$x_b := \operatorname{argmin}_{z \in \{\bar{x}_b, \bar{x}\}} f(z).$$

Then we update the parameters α , h , γ , η , and u using Algorithm 2 and continue this procedure until a stopping criterion holds.

Summarizing the above-mentioned discussion, we give the double-projection optimal subgradient algorithm as follows:

Algorithm 4: OSGA (double-projection optimal subgradient algorithm)

Input: $\delta, \alpha_{\max} \in]0, 1[$, $0 < \kappa' \leq \kappa$; local parameters: $x_0, \mu \geq 0$, f_{target} ;

Output: x_b, f_{x_b} ;

```

1 begin
2   choose an initial best point  $x_b$ ; compute  $f_{x_b}$  and  $g_{x_b}$ ;
3   if  $f_{x_b} \leq f_{\text{target}}$  then
4     stop;
5   else
6      $h = g_{x_b} - \mu g_Q(x_b)$ ;  $\gamma = f_{x_b} - \mu Q(x_b) - \langle h, x_b \rangle$ ;
7      $\gamma_b = \gamma - f_{x_b}$ ;  $u = U(\gamma_b, h)$ ;  $\eta = E(\gamma_b, h) - \mu$ ;
8   end
9    $\alpha = \alpha_{\max}$ ;
10  while stopping criteria do not hold do
11     $x = x_b + \alpha(u - x_b)$ ; compute  $f_x$  and  $g_x$ ;
12     $g = g_x - \mu g_Q(x)$ ;  $\bar{h} = h + \alpha(g - h)$ ;
13     $\bar{\gamma} = \gamma + \alpha(f_x - \mu Q(x) - \langle g, x \rangle - \gamma)$ ;
14     $x'_b = \operatorname{argmin}_{z \in \{x_b, x\}} f(z, v_z)$ ;  $f_{x'_b} = \min\{f_{x_b}, f_x\}$ ;
15     $\gamma'_b = \bar{\gamma} - f_{x'_b}$ ;  $u' = U(\gamma'_b, \bar{h})$ ;  $x' = x_b + \alpha(u' - x_b)$ ; compute  $f_{x'}$ ;
16    choose  $\bar{x}_b$  in such a way that  $f_{\bar{x}_b} \leq \min\{f_{x'_b}, f_{x'}\}$ ;
17     $\bar{\gamma}_b = \bar{\gamma} - f_{\bar{x}_b}$ ;  $\bar{u} = U(\bar{\gamma}_b, \bar{h})$ ;  $\bar{\eta} = E(\bar{\gamma}_b, \bar{h}) - \mu$ ;  $x_b = \bar{x}_b$ ;  $f_{x_b} = f_{\bar{x}_b}$ ;
18    if  $f_{x_b} \leq f_{\text{target}}$  then
19      stop;
20    else
21      update the parameters  $\alpha$ ,  $h$ ,  $\gamma$ ,  $\eta$  and  $u$  using UPS;
22    end
23  end
24 end

```

The following inequality is needed in Section 3.3 to establish the complexity of Algorithm 4.

Theorem. 3.2.2. *In OSGA, the error factors are related by*

$$\bar{\eta} - (1 - \alpha)\eta \leq \frac{\alpha^2 \|g_x\|_*^2}{2(1 - \alpha)(\eta + \mu)Q_0}.$$

Proof. See Theorem 4.1 in [130]. □

Theorem. 3.2.3. *If f has Lipschitz continuous gradients with the constant L , then in OSGA we have*

$$\bar{\eta} > (1 - \alpha)\eta \Rightarrow (1 - \alpha)(\eta + \mu) < \alpha^2 L.$$

Proof. See Theorem 4.2 in [130]. □

Features of OSGA-V and OSGA. On the basis of the construction of the optimal subgradient framework, OSGA-V and OSGA share the following features:

1. Since only the first-order nonsmooth oracle is claimed, they can solve general nonsmooth convex problems without considering the structure of problems. This is especially important for classes of problems that are expensive or can not be solved by proximal-based methods or Nesterov-type optimal methods;
2. No information regarding global parameters such as Lipschitz constants of function and gradients is needed;
3. OSGA-V and OSGA have a simple structure and low memory requirement;
4. OSGA's subproblem (3.4) is simple nonconvex smooth problem that can be solved in a closed form or by a simple iterative scheme for many interesting domains C (see Chapter 5);
5. OSGA is optimal for Lipschitz continuous nonsmooth problems (the complexity $O(\epsilon^{-2})$) and smooth problems that have Lipschitz continuous gradients (the complexity $O(\epsilon^{-1/2})$) (see Theorem 3.3.3). However, OSGA-V is optimal for Lipschitz continuous nonsmooth problems (see Theorem 3.3.3).

3.3 Convergence analysis

In this section we establish the complexity bounds of OSGA-V for Lipschitz continuous nonsmooth problems and the complexity bound of OSGA for Lipschitz continuous nonsmooth problems and smooth problems with Lipschitz continuous gradients. We also show that if f is strictly or in particular strongly convex, the sequence generated by OSGA-V and OSGA is convergent to $\hat{x}$.

To guarantee the existence of a minimizer for OSGA-V and OSGA, we assume that the following conditions :

(H1) The objective function f is proper and convex;

(H2) The upper level set $N_f(x_0) = \{x \in C \mid f(x) \leq f(x_0)\}$ is bounded, for the starting point x_0 .

Since f is convex, the upper level set $N_f(x_0)$ is closed and convex, and V is a finite-dimensional vector space, (H2) implies that the upper level set $N_f(x_0)$ is convex and compact. It follows from the continuity and properness of the objective function f that it attains its global minimizer on the upper level set $N_f(x_0)$. Therefore, there is at least one minimizer $\hat{x}$, and its corresponding minimum is denoted by $\hat{f}$.

The next two results are necessary to obtain the optimal complexity of OSGA-V and OSGA.

Proposition 3.3.1. *Suppose that the sequence $\{x_k\}$ is generated by OSGA-V or OSGA. Suppose also that the dual norm of the subgradient g_x encountered during the iteration remains bounded by the constant c_0 . Define*

$$c_1 := \frac{c_0^2}{2Q_0}, \quad c_2 := \max \left(\frac{e^\kappa c_1}{(1-\lambda)(1-\alpha_{\max})}, \frac{\eta_0(\eta_0 + \mu)}{\alpha_0} \right), \quad c_3 := \frac{c_2}{2\lambda}.$$

(i) in each iteration we have

$$\eta(\eta + \mu) \leq \alpha c_2.$$

(ii) The algorithm stops after at most

$$K_\mu(\alpha, \eta) := 1 + \kappa^{-1} \log \frac{c_2 \alpha}{\varepsilon(\varepsilon + \mu)} + \frac{c_3}{\varepsilon(\varepsilon + \mu)} + \frac{c_3}{\eta(\eta + \mu)}$$

further iterations.

Proof. From Theorems 3.2.1 and 3.2.2 and Proposition 5.2 of in [130] the results are valid. $\square$

In particular, (i) and (ii) hold when the iterates stay in a bounded region of the interior of C , or when f is Lipschitz continuous in C . Note that any convex function is Lipschitz continuous in any closed and bounded domain inside the interior of its support. Hence if the iterations stay in a bounded region R of the interior of C , $\|g\|$ is bounded by the Lipschitz constant of f in the closure of the region R .

Proposition 3.3.2. *Suppose that the sequence $\{x_k\}$ is generated by OSGA. Suppose also that f has Lipschitz continuous gradients with the constant L , and set*

$$c_4 := \max\left(\frac{\eta_0 + \mu}{\alpha_0^2}, \frac{e^{2\kappa L}}{1 - \alpha_{\max}}\right), \quad c_5 := \frac{4c_4}{\lambda^2}, \quad c_6 := \sqrt{\frac{c_4}{\lambda}}, \quad c_7 := \frac{c_6}{\lambda}.$$

(i) *in each iteration we have*

$$\eta + \mu \leq \alpha^2 c_4.$$

(ii) *The algorithm stops after at most*

$$K_\mu(\alpha, \eta) := 1 + \kappa^{-1} \log\left(\alpha \sqrt{\frac{c_4}{\varepsilon}}\right) + \sqrt{\frac{c_5}{\varepsilon}} - \sqrt{\frac{c_4}{\eta}}$$

if $\mu = 0$, and

$$K_\mu(\alpha, \eta) := 1 + \frac{\log(c_6 \alpha)}{\kappa} + c_7 \log \frac{\eta}{\varepsilon} \sqrt{\frac{c_5}{\varepsilon}} - \sqrt{\frac{c_4}{\eta}}$$

if $\mu > 0$.

Proof. From Theorem 3.2.3 and Proposition 5.3 in [130] the results are valid. $\square$

Theorem. 3.3.3. *Suppose that $f - \mu Q$ is convex, then we have*

(i) (NONSMOOTH COMPLEXITY BOUND) *If the points generated by OSGA-V or OSGA stay in a bounded region of the interior of C , or if f is Lipschitz continuous in C , the total number of iterations needed to reach a point with $f(x) \leq f(\hat{x}) + \varepsilon$ is at most $O((\varepsilon^2 + \mu\varepsilon)^{-1})$. Thus the asymptotic worst case complexity is $O(\varepsilon^{-2})$ when $\mu = 0$ and $O(\varepsilon^{-1})$ when $\mu > 0$.*

(ii) (SMOOTH COMPLEXITY BOUND) *Let the points generated by OSGA. If f has Lipschitz continuous gradients with the constant L , the total number of iterations needed to reach a point with $f(x) \leq f(\hat{x}) + \varepsilon$ is at most $O(\varepsilon^{-1/2})$ if $\mu = 0$ and at most $O(|\log \varepsilon| \sqrt{L/\mu})$ if $\mu > 0$.*

Proof. See Theorem 5.1 of [130]. $\square$

We conclude this section by establishing the convergence of the sequence $\{x_k\}$ generated by OSGA-V or OSGA to $\hat{x}$ if f is strictly or in particular strongly convex.

Proposition 3.3.4. *Suppose that f is strictly convex, then the sequence $\{x_k\}$ is generated by OSGA-V or OSGA is convergent to $\hat{x}$ if $\hat{x} \in \text{int}(C)$.*

Proof. Since f is strictly convex, the minimizer $\hat{x}$ is unique. By $\hat{x} \in \text{int}(C)$, there exists a small $\delta > 0$ such that the neighborhood

$$N(\hat{x}) := \{x \in V \mid \|x - \hat{x}\| < \delta\} \subseteq C,$$

is included in C , which is a convex set. Let x_δ be a minimizer of the problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \partial N(\hat{x}), \end{aligned} \tag{3.27}$$

where $\partial N(\hat{x})$ denotes the boundary of $N(\hat{x})$. Set $\varepsilon_\delta := f(x_\delta) - \hat{f}$ and consider the upper level set

$$N_f(x_\delta) := \{x \in C \mid f(x) \leq f(x_\delta) = \hat{f} + \varepsilon_\delta\}.$$

Now Theorem 3.3.3 implies that the algorithm attains an ε_δ -solution of (5.22) in a finite number κ of iterations. Hence after κ iterations the best point x_b attained by OSGA satisfies $f(x_b) \leq \hat{f} + \varepsilon_\delta$, i.e., $x_b \in N_f(x_\delta)$. We now show that $N_f(x_\delta) \subseteq N(\hat{x})$. To prove this statement by contradiction, we suppose that there exists $x \in N_f(x_\delta) \setminus N(\hat{x})$. Since $x \notin N(\hat{x})$, we have $\|x - \hat{x}\| > \delta$. Therefore, there exists λ_0 such that

$$\|\lambda_0 x + (1 - \lambda_0)\hat{x}\| = \delta.$$

From (3.27), $f(x) \leq f(x_\delta)$, and the strictly convex property of f , we obtain

$$\begin{aligned} f(x_\delta) &\leq f(\lambda_0 x + (1 - \lambda_0)\hat{x}) < \lambda_0 f(x) + (1 - \lambda_0)f(\hat{x}) \\ &\leq \lambda_0 f(x_\delta) + (1 - \lambda_0)f(x_\delta) = f(x_\delta), \end{aligned}$$

which is a contradiction, i.e., $N_f(x_\delta) \subseteq N(\hat{x})$ implying $x \in N(\hat{x})$, which gives the result. $\square$

Proposition 3.3.4 is valid for any strictly convex functions, but if the function is strongly convex, we can get more information.

Proposition 3.3.5. *Suppose that f is strongly convex, then the sequence $\{x_k\}$ is generated by OSGA-V or OSGA is convergent to $\hat{x}$ and we have*

$$\|x - \hat{x}\| \leq \left(\frac{2(t_2 + 1)}{\sigma_{t_1}(1 - t_2)} \varepsilon \right)^{1/2},$$

where $t \in [t_1 \ t_2] \subset]0 \ 1[$.

Proof. Since f is strongly convex, for $x, z \in V$ and $t \in (0, 1)$, we have

$$\begin{aligned} t(1-t)\frac{\sigma}{2}\|x-\hat{x}\|^2 &\leq tf(x) + (1-t)f(\hat{x}) - f(tx + (1-t)\hat{x}) \\ &\leq t|f(x) - f(\hat{x})| + |f(\hat{x}) - f(tx + (1-t)\hat{x})| \\ &\leq (t+1)\varepsilon, \end{aligned}$$

leading to

$$\|x - \hat{x}\| \leq \left(\frac{2(t+1)}{\sigma t(1-t)} \varepsilon \right)^{1/2} \leq \left(\frac{2(t_2+1)}{\sigma t_1(1-t_2)} \varepsilon \right)^{1/2}.$$

for $t \in [t_1, t_2] \subset]0, 1[$, giving the results. □

Part II

Optimal Subgradient Algorithms: Applicability & Developments

Chapter 4

Unconstrained convex optimization

In this chapter we consider the unconstrained minimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in V. \end{array} \quad (4.1)$$

In particular, we first consider objectives with composite form involving several nonlinear terms and linear operators (multi-terms affine composite problems), which cannot be generally solved with proximal gradient algorithms, and then we consider a class of problems involving costly linear operators and cheap nonlinear terms. In both cases we discuss how OSGA can be efficiently applied for solving such problems.

4.1 Multi-terms affine composite problems

In this section we consider a minimization problem (4.1) with an objectives of the form

$$f(x) := \sum_{i=1}^{n_1} f_i(A_i x) + \sum_{j=1}^{n_2} \varphi_j(W_j x), \quad (4.2)$$

where $f_i : U_i \rightarrow \overline{\mathbb{R}}$, for $i = 1, 2, \dots, n_1$, are proper and convex functions, $\varphi_j : V_j \rightarrow \overline{\mathbb{R}}$, for $j = 1, 2, \dots, n_2$, are smooth or nonsmooth convex functions, and $A_i : V \rightarrow U_i$, for $i = 1, 2, \dots, n_1$, and $W_j : V \rightarrow V_j$, for $j = 1, 2, \dots, n_2$, are linear operators. Here, f is called the **multi-term affine composite function**, see also [92]. Since affine terms $A_i x$ and $W_j x$ and functions f_i and φ_j , for $i = 1, \dots, n_1$ and $j = 1, \dots, n_2$, are convex and the domain of f , i.e.,

$$\text{dom } f = \left(\bigcap_{i=1}^{n_1} \text{dom } f_i \circ A_i \right) \cap \left(\bigcap_{j=1}^{n_2} \text{dom } \varphi_j \circ W_j \right),$$

is a convex set, the function f is convex on its domain. We suppose that $\text{ri}(\text{dom } f) \neq \emptyset$. This, the fact that V is a finite-dimensional vector space, and Proposition 2.1.13 lead to

$$\partial f(x) = \sum_{i=1}^{n_1} A_i^* \partial f_i(A_i x) + \sum_{j=1}^{n_2} W_j^* \partial \varphi_j(W_j x). \quad (4.3)$$

Considering (4.2), we are generally interested in solving the composite convex minimization problem

$$\hat{f} := \min_{x \in V} f(x). \quad (4.4)$$

Under considered features of f , the problem (4.4) has a global minimizer denoted by $\hat{x}$. The functions of the form f are frequently appeared in the recent interests of employing hybrid regularizations or mixed penalty functions for solving problems in the fields such as signal and image processing, machine learning, geophysics, economics, and statistics. In the sequel, we will see that many well-studied structured optimization problems are special cases of (4.2).

Example. 4.1.1. (LINEAR INVERSE PROBLEM) *In many applications, e.g., those arising in signal and image processing, machine learning, compressed sensing, geophysics, and statistics, key features cannot be studied by straightforward investigations, but must be indirectly inferred from some observable quantities. Thanks to this characteristic, they are typically referred as **inverse problems**. If a linear relevance between the features of interest and observed data can be prescribed, this leads to linear inverse problems. If $y \in \mathbb{R}^m$ be an indirect observation of an original object $x \in \mathbb{R}^n$ and $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a linear operator, then the linear inverse problem is defined by*

$$y = Ax + v, \quad (4.5)$$

where $v \in \mathbb{R}^m$ represents an additive noise about which little is known apart from which qualitative knowledge.

In practice, the systems (4.5) is typically **underdetermined**, **rank-deficient**, or **ill-conditioned**. The primary difficulty with linear inverse problems is that the inverse object is extremely sensitive to y due to small or zero singular values of A meaning ill-conditioned systems behave like singular cases. Indeed, in the case that A^{-1} for square problems or **pseudo-inverse** $A^\dagger = (A^*A)^{-1}A^*$ for full rank over-determined systems is exist, then analyzing the singular value decomposition has shown that $\tilde{x} = A^{-1}y$ or $\tilde{x} = A^\dagger y$ is inaccurate and meaningless approximation for x , see [127]. Moreover, when the vector δ is not known, one cannot solve (4.5) directly.

From underdetermined or rank-deficient feature of inverse problems, we know that if a solution exists, then there exist infinitely many solutions. Hence some additional information is required to determine a satisfactory solution of (4.5). It

is usually interested to determine a solution by minimizing $\|Ax - y\|_2$, leading to the least-square problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - y\|_2^2, \quad (4.6)$$

where $\|\cdot\|_2$ denotes the ℓ_2 -norm. In view of ill-condition feature of the problem (4.5), the solution of (4.6) is usually improper. Hence Tikhonov in [152] proposed the penalized minimization problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - y\|_2^2 + \frac{\lambda}{2} \|x\|_2^2, \quad (4.7)$$

where λ is a regularization parameter controlling the trade-off between the least square data fitting term and the regularization term. The problem (4.7) is convex and smooth, and selecting a suitable regularization parameter leads to a well-posed problem. In many applications, we seek the sparsest solution of (4.5) among all solutions provided that y is acquired from a highly sparse observation. To this end, the constrained problem

$$\begin{aligned} \min \quad & \|x\|_0 \\ \text{s.t.} \quad & \|Ax - y\| \leq \varepsilon \end{aligned}$$

is used, where $\|x\|_0$ is the number of all nonzero elements of x and ε is a nonnegative constant. It is known that the objective function is nonconvex, however, its convex relaxation is much more preferable. In [46], it was proposed to solve

$$\begin{aligned} \min \quad & \|x\|_1 \\ \text{s.t.} \quad & \|Ax - y\| \leq \varepsilon. \end{aligned}$$

or its unconstrained reformulation

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - y\|_2^2 + \lambda \|x\|_1, \quad (4.8)$$

which is referred as **basis pursuit denoising** or **lasso**. Due to non-differentiability of the ℓ_1 -norm, solving this problem is more difficult than solving (4.7).

Example. 4.1.2. (SCALED ELASTIC NET) Consider the unconstrained problem (4.1) with the objective

$$f(x) = \frac{1}{2} \|Ax - b\|_2^2 + \lambda_1 \|W_1 x\|_1 + \frac{1}{2} \lambda_2 \|W_2 x\|_2^2. \quad (4.9)$$

This objective can be considered in the form (4.2) by setting

$$f(x) = f_1(A_1 x) + \varphi_1(W_1 x) + \varphi_2(W_2 x),$$

where

$$f_1(A_1 x) = \frac{1}{2} \|A_1 x - b\|_2^2, \quad \varphi_1(W_1 x) = \lambda_1 \|W_1 x\|_1, \quad \varphi_2(W_2 x) = \frac{1}{2} \lambda_2 \|W_2 x\|_2^2.$$

This problem has been used in many applications of linear inverse problems.

4.1.1 Convergence analysis

We discuss the convergence of OSGA for the multi-term affine composite functions (4.2). To this end, we assume:

(H1) The objective function f is proper and convex;

(H2) The upper level set $N_f(x_0) = \{x \in V \mid f(x) \leq f(x_0)\}$ is bounded, for the starting point x_0 .

Since f is convex, the upper level set $N_f(x_0)$ is closed and convex, and V is finite-dimensional, (H2) implies that the upper level set $N_f(x_0)$ is convex and compact. From the continuity and properness of the objective f , we obtain that f attains its global minimizer on the upper level set $N_f(x_0)$. Therefore, there is at least one minimizer $\hat{x}$, and its corresponding minimum is denoted by $\hat{f}$.

Notice that the problem considered in this section is a special case of (3.1) for $C = V$, and the complexity bounds of OSGA are not related to the constraint C . Therefore, the complexity bounds of OSGA for the multi-term composite functions remains valid. We establish this result in the next theorem.

Theorem. 4.1.3. *Suppose that f is a convex function of the form (4.1) and the sequence $\{x_k\}$ is generated by OSGA-V or OSGA. Then we have*

(i) (Nonsmooth complexity bound) *If the points stays in a bounded region of V , or f is Lipschitz continuous in V , then the total number of iterations needed is $O(\epsilon^{-2})$. Thus the asymptotic worst case complexity is $O(\epsilon^{-2})$.*

(ii) (Smooth complexity bound) *Let the sequence $\{x_k\}$ is generated by OSGA. If f has Lipschitz continuous gradient, the total number of iterations needed for the algorithm is $O(\epsilon^{-1/2})$.*

In the next two results we show that the sequence $\{x_k\}$ generated by OSGA-V or OSGA is convergent to $\hat{x}$ if the objective f is strictly or strongly convex.

Proposition 4.1.4. *Suppose that f is strictly convex, then the sequence $\{x_k\}$ is generated by OSGA-V or OSGA is convergent to $\hat{x}$.*

Proof. The result is valid by Proposition 3.3.4 for $C = V$. □

Proposition 4.1.4 is valid for any strictly convex functions, but if the function is strongly convex, we can get more information.

Proposition 4.1.5. *Suppose that f is strongly convex, then the sequence $\{x_k\}$ is generated by OSGA is convergent to $\hat{x}$ and we have*

$$\|x - \hat{x}\| \leq \left(\frac{2(t_2 + 1)}{\sigma t_1(1 - t_2)} \epsilon \right)^{1/2},$$

where $t \in [t_1 \ t_2] \subset]0 \ 1[$.

Proof. The result is valid by Proposition 3.3.5 for $C = V$. $\square$

In the remainder of this section we propose some prox-functions and then derive a closed form solution for the subproblem (3.4).

4.1.2 Solving the auxiliary subproblem

A prox-function Q is strongly convex that is analytically known. This means that it takes its unique global minimum which is positive by the definition of Q . Suppose z_0 is the global minimizer of Q . By the definition of center of Q and the first-order optimality condition for (3.3), we have that $g_Q(z_0) = 0$. This fact, along with the strongly convex condition (2.4), implies

$$Q(z) \geq Q_0 + \frac{\sigma}{2} \|z - z_0\|^2, \quad (4.10)$$

which means that $Q(z) > 0$, for all $z \in V$. In addition, we are interested to use separable prox-functions. Taking this fact into account and (4.10), appropriate choices of prox-functions for unconstrained problem (4.2) can be

$$Q_1(z) := Q_0 + \frac{\sigma}{2} \|z - z_0\|_2^2 \quad (4.11)$$

and

$$Q_2(z) := Q_0 + \frac{\sigma}{2} \sum_{i=1}^n w_i (z_i - (z_0)_i)^2, \quad (4.12)$$

where $Q_0 > 0$ and $w_i \geq 1$, for $i = 1, 2, \dots, n$. It is not hard to show that (4.11) and (4.12) are strongly convex functions satisfying (4.10).

Let $\|\cdot\|$ shows the **quadratic norm** on vector space V , i.e.,

$$\|z\|_D := \sqrt{\langle Dz, z \rangle}$$

by means of a **preconditioner** D , where D is symmetric and positive definite. The associated dual norm on V^* is given by

$$\|h\|_{*D} := \|D^{-1}h\|_D = \sqrt{\langle h, D^{-1}h \rangle},$$

where D^{-1} denotes the inverse of D . For given a symmetric and positive definite preconditioner D , we consider the quadratic function

$$Q(z) := Q_0 + \frac{\sigma}{2} \|z - z_0\|_D^2, \quad (4.13)$$

where Q_0 is a positive number and $z_0 \in V$ is the center of Q . In the next result we will show that (4.13) is a prox-function.

We now emphasize that an efficient solving of the subproblem (3.4) is highly related to the selection of prox-functions. While using a suitable prox-function allows a very efficient way of computing u , employing other selections may significantly slow down the process of solving the auxiliary subproblem. We show that using the prox-function (4.13) the subproblem (3.4) can be explicitly solved.

Proposition 4.1.6. *Let $\eta = E(\gamma, h) > 0$ and $u = U(\gamma, h)$. Then we have*

$$\gamma + \langle h, u \rangle = -E(\gamma, h)Q(u), \quad (4.14)$$

$$\langle E(\gamma, h)g_Q(u) + h, z - u \rangle \geq 0 \quad \text{for all } z \in V, \quad (4.15)$$

$$\gamma + \langle h, u \rangle \geq E(\gamma, h)Q(u) \left(\frac{1}{2} \|z - u\|^2 - Q(z) \right) \quad \text{for all } z \in V. \quad (4.16)$$

Proof. The results are valid by Proposition 3.1.2 for $C = V$. $\square$

Proposition 4.1.7. *Suppose Q is determined by (4.13) and $Q_0 > 0$. Then Q is a prox-function with the center z_0 and satisfies (4.10). Moreover, the subproblem (3.4) using this Q is explicitly solved by*

$$u = z_0 - E(\gamma, h)^{-1} \sigma^{-1} D^{-1} h \quad (4.17)$$

with

$$E(\gamma, h) = \frac{-\beta_1 + \sqrt{\beta_1^2 + 4Q_0\beta_2}}{2Q_0} = \frac{2\beta_2}{\beta_1 + \sqrt{\beta_1^2 + 4Q_0\beta_2}}, \quad (4.18)$$

where $\beta_1 = \gamma + \langle h, z_0 \rangle$ and $\beta_2 = (\frac{1}{2}\sigma^{-2} - \sigma^{-1}) \|h\|_*^2$.

Proof. We first show that Q is a prox-function satisfying (4.10). From $g_Q(x) = \sigma D(x - z_0)$, we get

$$\begin{aligned} Q(x) + \langle g_Q(x), z - x \rangle + \frac{\sigma}{2} \|z - x\|_D^2 &= Q(x) + \langle g_Q(x), z - x \rangle + \frac{\sigma}{2} \langle D(z - x), z - x \rangle \\ &= Q_0 + \frac{\sigma}{2} \langle D(x - z_0), x - z_0 \rangle + \langle g_Q(x), z - x \rangle + \frac{\sigma}{2} \langle D(z - x), z - x \rangle \\ &= Q_0 + \frac{\sigma}{2} \langle D(x - z_0), z - z_0 \rangle + \frac{\sigma}{2} \langle D(z - z_0), z - x \rangle \\ &= Q_0 + \frac{\sigma}{2} \langle D(x - z_0), z - z_0 \rangle + \frac{\sigma}{2} \langle D(z - x), z - z_0 \rangle \\ &= Q_0 + \frac{\sigma}{2} \langle D(z - z_0), z - z_0 \rangle \\ &= Q(z). \end{aligned}$$

This clearly means that Q is strongly convex by the convexity parameter σ , i.e., (4.13) is a prox-function. Moreover, z_0 is the center of Q and $g_Q(z_0) = 0$. This fact, together with (2.5) and (3.3), implies that (4.10) holds.

By Proposition 3.1.1, we may assume that $e := E(\gamma, h) > 0$. Since $C = V$ and $g_Q(z) = D(z - z_0)$, we conclude from the proposition that $E(\gamma, h)D(z - z_0) + h = 0$, where $u = U(\gamma, h)$, so that

$$U(\gamma, h) = z_0 - e^{-1}Dh.$$

Inserting this into (4.14), we find

$$e(Q_0 + \frac{1}{2}\| -e^{-1}D^{-1}h\|^2) = eQ(u) = -\gamma - \langle h, z_0 - e^{-1}D^{-1}h \rangle,$$

which simplifies to the quadratic equation

$$Q_0e^2 + (\gamma + \langle h, z_0 \rangle)e - \frac{1}{2}\|h\|_*^2 = \beta_1e^2 + \beta_2e + \beta_3 = 0.$$

Since the left hand side is negative at $e = 0$, there is exactly one positive solution, which therefore is the unique maximizer. It is given by

$$e = \frac{-\beta_2 + \sqrt{\beta_2^2 + 4\beta_1 * \beta_3}}{2\beta_1},$$

giving the result. □

It is obvious that (4.11) and (4.12) are special cases of (4.13), so the solution (4.17) and (4.18) derived for (4.13) can be simply adapted for cases that we use (4.11) or (4.12). Furthermore, notice that the error bound (3.6) is a proportional of $Q(\hat{x})$, which means that an acceptable choosing of x_0 makes the term $Q(\hat{x}) = Q_0 + \frac{1}{2}\|\hat{x} - x_0\|_D^2$ enough small. Hence selecting a suitable starting point x_0 , as close as possible to the optimizer $\hat{x}$, has a positive effect on giving a better complexity bound. This also proposes that a reasonable choice for Q_0 can be $Q_0 \approx \frac{1}{2}\|\hat{x} - x_0\|_D^2$.

We conclude this section by considering cases that the variable domain is a set of all $m \times n$ matrices, $V = \mathbb{R}^{m \times n}$. There are many applications of convex optimization with matrix variables, e.g., nuclear norm minimization (see RECHT et al. [145]) and sparse covariance selection (see D'ASPREMONT et al. [71]). For such cases, computing a function value or a subgradient needs the matrix variables. Therefore, one cannot consider the vector representation of the matrix variables to apply OSGA for solving such minimization problems. This suggests that we need a prox-function that can be used for matrix variables. Hence we introduce a suitable prox-function such that the subproblem (3.4) is solved in a closed form. The details are described in the next result.

Proposition 4.1.8. Suppose $Q_0 > 0$ and Q is determined by

$$Q(Z) := Q_0 + \frac{\sigma}{2} \|Z - Z_0\|_F^2, \quad (4.19)$$

where $\|\cdot\|_F$ is the Frobenius norm. Then Q is a prox-function with the center Z_0 and satisfies (4.10). Moreover, the subproblem (3.4) corresponded to this Q is explicitly solved by

$$U = Z_0 - E(\gamma, H)^{-1} \sigma^{-1} H \quad (4.20)$$

with

$$E(\gamma, H) = \frac{-\xi_1 + \sqrt{\xi_1^2 + 4Q_0\xi_2}}{2Q_0} = \frac{2\xi_2}{\xi_1 + \sqrt{\xi_1^2 + 4Q_0\xi_2}}, \quad (4.21)$$

where $\xi_1 = \gamma + \text{Tr}(H^T Z_0)$ and $\xi_2 = (\frac{1}{2}\sigma^{-2} - \sigma^{-1}) \|H\|_F^2$.

Proof. By the definition of the Frobenius norm, Z_0 is the minimizer of (3.3). The function Q is continuously differentiable and $g_Q(x) = \sigma(X - Z_0)$. Thus, we have

$$\begin{aligned} Q(X) + \langle g_Q(X), Z - X \rangle + \frac{\sigma}{2} \|Z - X\|_F^2 &= Q(X) + \sigma \langle X - Z_0, Z - X \rangle + \frac{\sigma}{2} \|Z - X\|_F^2 \\ &= Q_0 + \frac{\sigma}{2} \langle X - Z_0, X - Z_0 \rangle + \sigma \langle X - Z_0, Z - X \rangle + \frac{\sigma}{2} \|Z - X\|_F^2 \\ &= Q_0 + \frac{\sigma}{2} \langle (X - Z_0), Z - Z_0 \rangle + \frac{\sigma}{2} \langle (Z - Z_0), Z - X \rangle \\ &= Q_0 + \frac{\sigma}{2} \langle (Z - Z_0), Z - Z_0 \rangle \\ &= Q(Z), \end{aligned}$$

implying that Q is strongly convex by the convexity parameter σ , i.e., (4.13) is a prox-function. By using (2.5) and (3.3), the condition (4.10) is clearly held.

Now, we consider the subproblem (3.4) and drive its minimizer. Let us define the function $E_{\gamma, H} : V \rightarrow \mathbb{R}$ using

$$E_{\gamma, H}(Z) := -\frac{\gamma + \langle H, Z \rangle}{Q(Z)},$$

where it is continuously differentiable and attains its supremum at $Z = U$. Therefore, in the point $Z = U$, we obtain

$$E(\gamma, H)Q(U) = -\gamma - \langle H, U \rangle. \quad (4.22)$$

It follows from $g_Q(Z) = \sigma(Z - Z_0)$, (4.22) and the first-order necessary optimality condition that

$$E(\gamma, H)\sigma(U - Z_0) + H = 0$$

or equivalently

$$U = Z_0 - E(\gamma, H)^{-1} \sigma^{-1} H.$$

By setting $e := E(\gamma, H)$ and substituting it into (4.22), we get

$$\begin{aligned} eQ(U) + \gamma + \langle H, U \rangle &= e \left(Q_0 + \frac{1}{2} \|e^{-1} \sigma^{-1} H\|_F^2 \right) + \gamma + \langle H, Z_0 - e^{-1} \sigma^{-1} H \rangle \\ &= e \left(Q_0 + \frac{1}{2} e^{-2} \sigma^{-2} \|H\|_F^2 \right) + \gamma + \text{Tr}(H^T Z_0) + e^{-1} \sigma^{-1} \|H\|_F^2 \\ &= Q_0 e^2 + (\gamma + \text{Tr}(H^T Z_0)) e + \left(\frac{1}{2} \sigma^{-2} - \sigma^{-1} \right) \|H\|_F^2 \\ &= Q_0 e^2 + \xi_1 e + \xi_2 = 0, \end{aligned}$$

where $\xi_1 = \gamma + \text{Tr}(H^T Z_0)$ and $\xi_2 = \left(\frac{1}{2} \sigma^{-2} - \sigma^{-1} \right) \|H\|_F^2$. Solving this quadratic equation and selecting the larger solution lead to

$$e = \frac{-\xi_1 + \sqrt{\xi_1^2 + 4Q_0\xi_2}}{2Q_0} = \frac{2\xi_2}{\xi_1 + \sqrt{\xi_1^2 + 4Q_0\xi_2}},$$

giving the result. □

4.2 Problems involving costly linear operators

In this section we consider the convex optimization problem

$$\begin{aligned} \min \quad & f(x) := \sum_{i=1}^p f_i(x, A_i x) \\ \text{s.t.} \quad & x \in V, \end{aligned} \tag{4.23}$$

where $f_i : U_i \rightarrow \overline{\mathbb{R}}$ is a proper and convex function, and $A_i : V \rightarrow U_i$ is a linear operator, for real finite-dimensional vector spaces U_i . Problems of the form (4.23) appear in many applications such as signal and image processing, machine learning, statistics, data fitting, and inverse problems. We here mention the next example:

Example. 4.2.1. (OVERDETERMINED SYSTEM OF EQUATIONS) *Consider the overdetermined system of equations*

$$y = Ax + v, \tag{4.24}$$

where $x \in \mathbb{R}^n$ is an original object, $A \in \mathbb{R}^{m \times n}$ with $m > n$, $y \in \mathbb{R}^m$ is an observation, and $v \in \mathbb{R}^m$ is an additive noise. The objective is to recover x from the observation

y by solving (4.24). Such problems appear in many applications, see, for example, [21, 22, 48], especially they are of interest for robust fitting of linear models to data. In practice, this problem is typically ill-posed, and v is unknown, so finding the direct solution is impossible or in the possible cases is inaccurate, see [127]. Hence x is usually computed by a minimization problem of the form (4.23) with one of the objective functions of Table 4.1.

Table 4.1: List of minimization problems for solving overdetermined systems of equations

function	name	function	name
$f_1(x, Ax) = \frac{1}{2} \ y - Ax\ _2^2$	L22R	$f_7(x, Ax) = \ y - Ax\ _1$	L1R
$f_2(x, Ax) = \frac{1}{2} \ y - Ax\ _2^2 + \frac{1}{2} \lambda \ x\ _2^2$	L22L22R	$f_8(x, Ax) = \ y - Ax\ _1 + \frac{1}{2} \lambda \ x\ _2^2$	L1L22R
$f_3(x, Ax) = \frac{1}{2} \ y - Ax\ _2^2 + \lambda \ x\ _1$	L22L1R	$f_9(x, Ax) = \ y - Ax\ _1 + \lambda \ x\ _1$	L1L1R
$f_4(x, Ax) = \ y - Ax\ _2$	L2R	$f_{10}(x, Ax) = \ y - Ax\ _\infty$	L1R
$f_5(x, Ax) = \ y - Ax\ _2 + \frac{1}{2} \lambda \ x\ _2^2$	L2L22R	$f_{11}(x, Ax) = \ y - Ax\ _\infty + \frac{1}{2} \lambda \ x\ _2^2$	L1L22R
$f_6(x, Ax) = \ y - Ax\ _2 + \lambda \ x\ _1$	L2L1R	$f_{12}(x, Ax) = \ y - Ax\ _\infty + \lambda \ x\ _1$	L1L1R

In Table 4.1, λ denotes the regularization parameter, and since the p -norm is a convex function, these objective functions are convex and includes the linear mapping A , which is typically a dense matrix.

In many applications, the objective function of optimization problems (4.23) involves expensive linear mappings (equivalently matrix-vector products with dense matrices). To apply a first-order method for minimizing such problems, the first-order oracle (function values and subgradients) should be available, i.e.,

$$f_x = f(x, Ax), \quad g_x \in \partial f(\cdot, Ax)(x) + A^* \partial f(x, A\cdot)(x).$$

Hence, in each call of the first-order oracle, one forward linear operator A and one adjoint operator A^* should be computed requiring $O(n^2)$ operations. This computationally leads to overall expensive function and subgradient evaluations such that the total cost of using a first-order method is dominated by the cost of applying forward and adjoint linear operators. This motivates us to develop an acceleration of OSGA using a multi-dimensional subspace search for solving such problems.

4.2.1 Multi-dimensional subspace search

A multi-dimensional subspace search scheme can be regarded as a generalization of line search techniques, which are one-dimensional search schemes for finding a step-size in a specific direction. Hence, in multi-dimensional subspace search, one searches a vector of step-sizes allowing the best combination of several search directions for optimizing an objective function. Generally, subspace search techniques are a class of descent methods, where they can be used independently or employed as an accelerator inside of iterative schemes to attain a faster convergence. The pioneering work of subspace optimization was proposed in 1969 for smooth problems by MIELE AND CANTRELL [113] and CRAGG & LEVY [69] called memory gradient technique defining a subspace of the form

$$S = \text{span}\{-g_k, d_{k-1}\},$$

where g_k denotes the gradient at x_k and d_{k-1} is the last available direction. Since then many subspace search schemes have been proposed by selecting various search directions, see, for example, [63] and references therein. Generally, in view of selected search directions used for constructing a subspace, two classes of subspace methods are recognized, namely, gradient-type techniques, see [69, 78, 115], and Newton-type schemes, see [110, 155, 162].

The primary idea of multi-dimensional subspace methods is to restrict the next iteration to a low-dimensional subspace by constructing a subproblem with a reduced dimension. Let us fix $M \ll n$, where n is the number of variables. We now suppose that $d_1, d_2, \dots, d_M$ are M directions used to span the subspace

$$S = \text{span}\{d_1, d_2, \dots, d_M\}. \quad (4.25)$$

In this case a direction d belongs to the subspace S if and only if there exist constants $t_1, t_2, \dots, t_M$ such that

$$d = \sum_{i=1}^M t_i d_i = Ut, \quad (4.26)$$

where $U = (d_1, d_2, \dots, d_M)$ is a matrix constructed by the considered directions and $t = (t_1, t_2, \dots, t_M)$ is a M -vector of coefficients. Afterwards, the M -dimensional minimization subproblem

$$\begin{aligned} \min \quad & \sum_{i=1}^p f_i(x + Ut, A_i(x + Ut)) = \sum_{i=1}^p f_i(x + Ut, v_i + V_i t) \\ \text{s.t.} \quad & t \in \mathbb{R}^M \end{aligned} \quad (4.27)$$

is defined to determine the best possible vector of step-sizes t , where $v_i = Ax$ and $V_i = A_i U$. The minimization problem (4.27) shows that the procedure of searching

the best possible direction of the form (4.26) in the subspace (4.25) generalizes the idea of exact line search, see, for example, [134]. If we construct the subspace

$$S = \text{span}\{x_{k-M+1}, x_{k-M+2}, \dots, x_k\}, \quad (4.28)$$

then the subspace minimization is defined by

$$\begin{aligned} \min \quad & \sum_{i=1}^p f_i(Ut, A_i(Ut)) = \sum_{i=1}^p f_i(Ut, V_i t) \\ \text{s.t.} \quad & t \in \mathbb{R}^M. \end{aligned} \quad (4.29)$$

Since $M \ll n$, the minimization subproblems (4.27) and (4.29) are low-dimensional and can be solved efficiently by classical optimization methods. Hence subspace search techniques can be implemented extremely fast and this leads to suitable schemes for large-scale optimization as the number of variables of practical problems growing up. Moreover, using a multi-dimensional subspace search as a inner step of iterative schemes needs low memory requirement, which is considerably cheaper than performing one step of the algorithm in the full dimension.

Motivated by the above-mentioned discussion, the multi-dimensional subspace search scheme can be described as follows:

Algorithm 5: MDSS (multi-dimensional subspace search)

Input: $x_b \in V$, U , $v_i \in U_i$, V_i ($i = 1, \dots, p$);

Output: x_{new} , f_{new} ;

1 **begin**

2 approximately solve the M -dimensional minimization problem (4.27)
or (4.29) to find t^* ;

3 set $x_{\text{new}} = x_b + Ut^*$ for (4.27) or $x_{\text{new}} = Ut^*$ for (4.29);

4 **end**

To implement Algorithm 5 successfully, some factors are crucial: (i) the number of directions M controlling the computational cost of the scheme; (ii) choosing suitable directions to construct the subspaces; (iii) solving the minimization problem (4.27) or (4.29) efficiently. Indeed, for choosing the number of directions M , there is a trade-off between the total computational cost per iteration and the amount of possible decrease in function values. Many common ideas in optimization can be considered as multi-dimensional subspace search techniques, namely conjugate gradient, limited memory quasi-Newton, memory gradient methods and so on, see, for example, [63, 78, 162].

We here use Algorithm 5 as an accelerator of OSGA for solving problems involving costly linear operators. More precisely, we save some previously computed points, construct a subspace of the form (4.25) and apply Algorithm 3 to

find a point $\bar{x}_b$ in Line 18 of Algorithm 4. This possibly give us a better point to select x_b in Algorithm 2. In the next section, we will show how the subspace S is constructed and how the subproblem (4.29) can be solved efficiently by OSGA in a reasonable cost.

4.2.2 Solving the subspace subproblem by OSGA

To construct the subspace S (4.25), we use some of the points appearing in OSGA. Let us fix the iteration counter k . If $k < M$, we will use OSGA without subspace search. If $k \geq M$, we consider the last $2M + 1$ points

$$x_b, x_i, x'_i \text{ for } j = k - M + 1, \dots, k$$

to construct the subspace

$$S := \text{span}\{x_{k-M+1}, x'_{k-M+1}, \dots, x_k, x'_k, x_b\}. \quad (4.30)$$

Then we define

$$U_k := (x_{k-M+1}, x'_{k-M+1}, \dots, x_k, x'_k, x_b) \quad (4.31)$$

and set

$$v_i^b := A_i x_b, \quad v_i^j := A_i x_j, \quad (v_i^j)' := A_i x'_j \text{ for } j = k - M + 1, \dots, k, \quad i = 1, \dots, p,$$

leading to

$$\begin{aligned} A_i(U_k t) &= (A_i U_k) t = (A_i x_{k-M+1}, A_i x'_{k-M+1}, \dots, A_i x_k, A_i x'_k, A_i x_b) t \\ &= (v_{k-M+1}, (v_{k-M+1}^j)' , \dots, v_k, (v_k^j)', v_b) t = V_i^k t, \end{aligned} \quad (4.32)$$

for $i = 1, \dots, p$, where

$$V_i^k := A_i U_k = (v_{k-M+1}, (v_{k-M+1}^j)' , \dots, v_k, (v_k^j)', v_b) \text{ for } i = 1, \dots, p.$$

To call OSGA for solving (4.29), one needs a routine for computing function values and subgradients. Now, using (4.32), the function value of f at t can be computed by

$$f_t = \sum_{i=1}^p f_i(U_k t, A_i(U_k t)) = \sum_{i=1}^p f(U_k t, V_i^k t), \quad (4.33)$$

which is free of matrix-vector multiplications in the full dimension. Meanwhile, from (4.32) and Proposition 2.1.13, we compute a subgradient of function f at $x_b + U_k t$ by

$$\begin{aligned} g_t &\in \sum_{i=1}^p U_k^T \partial f_i(\cdot, V_i^k t)(U_k t) + (A_i U_k)^T (\partial f_i(U_k t, A_i \cdot))(U_k t) \\ &= \sum_{i=1}^p U_k^T \partial f_i(\cdot, V_i^k t) + (V_i^k)^T (\partial f_i(U_k t, A_i \cdot))(U_k t). \end{aligned} \quad (4.34)$$

The matrix-vector multiplications of the forms $U_k t$, $V_i^k t$, $U_k^T y$, and $(V_i^k)^T z$ for $t, y, z \in \mathbb{R}^{2M+1}$ in the subspace S need $(2M+1)n$ operations, while each call of oracle in the full space needs $O(n^2)$ operations. Indeed, if $M \ll n$, say $M = 2$ to 10 , then $(2M+1)n$ is much less than $O(n^2)$ implying that for applying OSGA to solve the subproblem (4.27) no further expensive linear algebra cost are needed as long as the condition $M \ll n$ is satisfied. Hence Algorithm 3 can be applied efficiently to accelerate OSGA without imposing too much computational cost, especially for objectives involving expensive linear operators and cheap nonlinear terms.

Lemma. 4.2.2. *If the subspace S is defined by (4.30), then the solution of the subproblem (4.29) satisfies*

$$f_{\bar{x}_b} \leq \min\{f_{x'_b}, f_{x'}\}.$$

Proof. By setting $t_1 = (0, \dots, 1, 0, 0)^T$ and $t_2 = (0, \dots, 0, 0, 1)^T$ and the definition of U_k , we have

$$U_k t_1 = x, \quad U_k t_2 = x_b,$$

which implies that the subspace S contains x and x_b leading to

$$x'_b = \operatorname{argmin}_{z \in \{x_b, x\}} f(z, v_z) \in S. \quad (4.35)$$

If we set $t_3 = (0, \dots, 0, 1, 0)^T$, then

$$U_k t_3 = x' \in S. \quad (4.36)$$

If t^* denotes the minimizer of the subproblem (4.29), then (4.35) and (4.36) imply

$$\bar{x}_b = U_k t^*, \quad f(\bar{x}_b) \leq \min\{f_{x'_b}, f_{x'}\},$$

giving the result. $\square$

Lemma 4.2.2 implies that Algorithm 5 is a special case of OSGA (Algorithm 4) obtained by specializing the choice in Line 16. Since the multi-dimensional subspace search only gives the possibility to attain a better point $\bar{x}_b$ (Line 16 of OSGA), the complexity results of OSGA given in Theorem 3.3.3 and also the results presented in Proposition 3.3.4 and Proposition 3.3.5 remain valid.

Motivated by our discussion, we now present a variant of OSGA using the multi-dimensional subspace search technique:

Algorithm 6: OSGA-S (OSGA with multi-dimensional subspace search)**Input:** $\delta, \alpha_{\max} \in]0, 1[, 0 < \kappa' \leq \kappa, x_0, \mu \geq 0, f_{\text{target}};$ **Output:** $x_b, f_{x_b};$ **1 begin****2** choose an initial best point x_b ; compute f_{x_b} and g_{x_b} ;**3** **if** $f_{x_b} \leq f_{\text{target}}$ **then****4** | stop;**5** **else****6** | $h = g_{x_b} - \mu g_Q(x_b); \gamma = f_{x_b} - \mu Q(x_b) - \langle h, x_b \rangle;$ **7** | $\gamma_b = \gamma - f_{x_b}; u = U(\gamma_b, h); \eta = E(\gamma_b, h) - \mu;$ **8** **end****9** $\alpha \leftarrow \alpha_{\max}; r = 0; \text{flag} = 0;$ **10** **while** *stopping criteria do not hold* **do****11** | $x = x_b + \alpha(u - x_b); v_i^x = A_i x, i = 1, \dots, p;$ **12** | $r = r + 1, U_{:r} = x; (V_i)_{:r} = v_i^x, i = 1, \dots, p;$ **13** | compute f_x and $g_x; g = g_x - \mu g_Q(x); \bar{h} = h + \alpha(g - h);$ **14** | $\bar{\gamma} = \gamma + \alpha(f_x - \mu Q(x) - \langle g, x \rangle - \gamma);$ **15** | $x'_b = \operatorname{argmin}_{z \in \{x_b, x\}} f(z, v_z); f_{x'_b} = \min\{f_{x_b}, f_x\};$ **16** | $\gamma'_b = \bar{\gamma} - f_{x'_b}; u' = U(\gamma'_b, \bar{h});$ **17** | $x' = x_b + \alpha(u' - x_b); v_{x'}^i = A_i(x'), i = 1, \dots, p; \text{ compute } f_{x'};$ **18** **if** $r \leq 2M$ **then****19** | $r = r + 1, U_{:r} = x; (V_i)_{:r} = v_{x'}^i, i = 1, \dots, p;$ **20** | $U_{:2M+1} = x_b; (V_i)_{:2M+1} = v_{x'_b}^i, i = 1, \dots, p; \text{flag} = 0;$ **21** **else****22** | $r = 1, U_{:r} = x; (V_i)_{:r} = v_{x'}^i, i = 1, \dots, p;$ **23** **end****24** **if** **flag** **then****25** | $f_{\bar{x}_b} = \min\{f_{x'_b}, f_{x'}\}; \bar{x}_b = \operatorname{argmin}\{f_{x'_b}, f_{x'}\}$ **26** **else****27** | Solve the subspace subproblem (4.27);**28** **end****29** | $\bar{\gamma}_b = \bar{\gamma} - f_{\bar{x}_b}; \bar{u} = U(\bar{\gamma}_b, \bar{h}); \bar{\eta} = E(\bar{\gamma}_b, \bar{h}) - \mu;$ **30** | $x_b = \bar{x}_b; f_{x_b} = f_{\bar{x}_b}; v_i^b = A_i x_b, i = 1, \dots, p;$ **31** **if** $f_{x_b} \leq f_{\text{target}}$ **then****32** | stop;**33** **else****34** | update the parameters α, h, γ, η and u using PUS;**35** **end****36** **end****37 end**

Chapter 5

Convex optimization with simple constraints

In this chapter we consider problems of the form (2.8) with simple constraints (bound constraints, simple domains, simple functional constraints) and show that by choosing suitable prox-functions OSGA's subproblem can be solved in a closed form or by a simple scheme.

5.1 Convex optimization with simple domains

In this section we consider the convex constrained optimization problem

$$\begin{array}{ll} \min & f(Ax) \\ \text{s.t.} & x \in C, \end{array} \quad (5.1)$$

where $f : C \rightarrow \overline{\mathbb{R}}$ is proper and convex, $A : V \rightarrow U$ is a linear operator, and $C \subseteq U$ is a simple convex domain. We call problem (5.1) a **simple domain** problem. This problem appears in many applications such as signal and image processing, machine learning, statistics, and inverse problem.

5.1.1 Convex problems with simple domains

In this section we consider a convex problem of the form (5.1) where the orthogonal projection onto the domain C is available either in a closed form or by a simple iterative scheme. To motivate the results of this section, we first give some examples appearing in applications.

Example. 5.1.1. (IMAGE RESTORATION) *The process of reconstructing or estimating a true image from a degraded observation is known as the image restoration, also called deblurring or deconvolution. Image restoration is addressed by*

solving a constraint satisfaction problem of the form

$$Ax = b, \quad x \in C,$$

where C a convex domain C that is commonly a box or the nonnegativity constraint. This is an ill-posed problem, see NEUMAIER [127], and normally handled by the regularized least-squares problem

$$\begin{aligned} \min \quad & \frac{1}{2} \|Ax - b\|_2^2 + \lambda \varphi(x) \\ \text{s.t.} \quad & x \in C \end{aligned} \tag{5.2}$$

or the regularized ℓ_1 problem

$$\begin{aligned} \min \quad & \|Ax - b\|_1 + \lambda \varphi(x) \\ \text{s.t.} \quad & x \in C, \end{aligned} \tag{5.3}$$

where $\varphi : C \rightarrow \mathbb{R}$ is a convex regularization function such as $\|\cdot\|_2^2$, $\|\cdot\|_1$, $\|\cdot\|_{ITV}$, and $\|\cdot\|_{ATV}$. The regularizers $\|\cdot\|_{ITV}$ and $\|\cdot\|_{ATV}$ are respectively called isotropic and anisotropic total variation, see, for example, [54], where they are defined by

$$\begin{aligned} \|x\|_{ITV} = & \sum_i^{m-1} \sum_j^{n-1} \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2} \\ & + \sum_i^{m-1} |x_{i+1,n} - x_{i,n}| + \sum_i^{n-1} |x_{m,j+1} - x_{m,j}| \end{aligned}$$

and

$$\begin{aligned} \|x\|_{ATV} = & \sum_i^{m-1} \sum_j^{n-1} \{|x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}|\} \\ & + \sum_i^{m-1} |x_{i+1,n} - x_{i,n}| + \sum_i^{n-1} |x_{m,j+1} - x_{m,j}|, \end{aligned}$$

for $x \in \mathbb{R}^{m \times n}$.

Example. 5.1.2. (BASIS PURSUIT PROBLEM) Let $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a linear operator with $m < n$ and $y \in \mathbb{R}^m$. The basis pursuit problem is the constrained minimization problem

$$\begin{aligned} \min \quad & \|x\|_1 \\ \text{s.t.} \quad & Ax = y, \end{aligned} \tag{5.4}$$

which determines an ℓ_1 -minimal solution $\hat{x}$ of the undetermined linear system $Ax = y$. This problem appears in many applications such signal and image processing and compressed sensing, see [31, 32, 58, 76, 156, 159, 160] and references therein.

According to the features of objective functions, (5.2) can be solved by Nesterov-type optimal methods, however, (5.3) and (5.4) cannot be solved by Nesterov-type optimal methods. Since OSGA only needs first-order information, it can deal with all of these problems without considering the structure of problems. In the remainder of this section, we establish how OSGA can be used to efficiently solve the

problem (5.1). Since the underlying problem (5.1) is a special case of the problem (3.1) considered in Chapter 3, the complexity of OSGA remains valid for both smooth and nonsmooth problems.

The quadratic function

$$Q(z) := \frac{1}{2} \|z\|_2^2 + Q_0, \quad (5.5)$$

is a prox-function, see e.g. [1]. We now show that the solution of OSGA's subproblem (3.4) can be found either in a closed form or by a simple iterative scheme. In particular, we address some convex domains that a closed form solution for associated OSGA's subproblem (3.4) can be found.

The next result shows that the solution of the auxiliary subproblem (3.4) is given by the orthogonal projection (2.13) of $y := e^{-1}h$ on the domain C followed by solving a one-dimensional nonlinear equation to determine e .

Theorem. 5.1.3. *Let u be a minimizer of (3.4) and also let $e = E_{\gamma,h}(u) > 0$. Then*

$$u = \hat{u}(e) := P_C(y), \quad y := -e^{-1}h,$$

where e is a solution of the univariate equation

$$\varphi(e) = 0$$

with

$$\varphi(e) := e \left(\frac{1}{2} \|\hat{u}(e)\|_2^2 + Q_0 \right) + \gamma + \langle h, \hat{u}(e) \rangle. \quad (5.6)$$

Proof. From Proposition 3.1.2, at the minimizer u , we obtain

$$eQ(u) = -\gamma - \langle h, u \rangle \quad (5.7)$$

and

$$\langle eu + h, z - u \rangle \geq 0 \quad \text{for all } z \in C. \quad (5.8)$$

By setting $z = u$ in this variational inequality, it follows that u is a solution of the minimization problem

$$\inf_{z \in C} \langle eu + h, z - u \rangle.$$

The first-order optimality condition (2.9) for this problem is

$$0 \in eu + h + N_C(u), \quad (5.9)$$

where

$$N_C(u) := \{p \in V \mid \forall y \in C, \langle p, u - y \rangle \geq 0\}$$

denotes the normal cone to C at u . Using $e > 0$ and (2.14), u satisfies

$$u = \operatorname{argmin}_{z \in C} \frac{1}{2} \|ez + h\|_2^2 = \operatorname{argmin}_{z \in C} \frac{1}{2} \|z - y\|_2^2 = P_C(y) = \hat{u}(e),$$

where $y = -e^{-1}h$ giving the result. $\square$

Theorem 5.1.3 gives a way to compute a solution of OSGA's subproblem (3.4) involving a projection on the domain C and solving the one-dimensional nonlinear equation. This equation can be solved exactly for some projection operators (see Table 5.1). However, one can solve this nonlinear equation approximately using zero finding schemes, see, e.g., Chapter 5 of [128]. We apply the results of Theorem 5.1.3 in the next scheme to solve OSGA's subproblem (3.4):

Algorithm 7: OSS (OSGA's subproblem solver)

Input: Q_0 , γ , h . a program for evaluating $\varphi(e)$ defined in (5.60);

Output: u , e ;

1 **begin**

2 solve the nonlinear equation $\varphi(e) = 0$ either in a closed form or approximately by a root finding solver;

3 set $u = \hat{u}(e)$.

4 **end**

To implement Algorithm 7 (OSS), we first need to solve the projection problem (2.13) effectively. Indeed, computing the orthogonal projection is a well-studied topic on convex optimization, and the projection operator is available for many domains C either in a closed form or by a simple iterative scheme. Table 5.1 gives some practically interesting convex domains, associated projection operators, and references for the formulas or iterative schemes. If one solves the equation $\varphi(e) = 0$ approximately, and an initial interval $[a, b]$ is available such that $\varphi(a)\varphi(b) < 0$, then a solution can be computed to ε -accuracy using the bisection scheme in $O(\log_2((b-a)/\varepsilon))$ iterations, see, for example, [128]. However, it is preferable to use a more sophisticated zero finder like the secant bisection scheme (Algorithm 5.2.6, [128]). If an interval $[a, b]$ with sign change is available¹, one can also use MATLAB's `fzero` function combining the bisection scheme, the inverse quadratic interpolation, and the secant method.

In the following we investigate special domains C , where the nonlinear equation $\varphi(e) = 0$ can be solved explicitly, see Table 5.2.

¹Without a sign change, `fzero` is unreliable; it fails on the simple quadratic $x^2 - 0.0001 = 0$ with starting point 0.2.

Table 5.1: List of some available projection operators for $C = \{x \in V \mid c(x)\}$, where $x \succcurlyeq 0$ means that x is positive semidefinite, $\sum_{i=1}^n \lambda_i u_i u_i^T$ is the eigenvalue decomposition of x , $\text{tr}(x)$ is the trace of the matrix x , and $\sigma_s : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^n$ is the function takes matrix x and return a vector of its singular values in nonincreasing order (in our case the singular value decomposition is equivalent to the eigenvalue decomposition)

defining constraint $c(x)$	Projection operator	Ref.
$Ax = b$	$u = y - A^\dagger(Ay - b)$	[138]
$\langle a, x \rangle = b$	$u = y - (\langle a, y \rangle - b) / (\ a\ _2^2) a$	[23]
$\langle a, x \rangle \leq b$	$u = y - (\langle a, y \rangle - b)_+ / (\ a\ _2^2) a$	[23]
$ \langle a, x \rangle \leq b$	$u = \begin{cases} y & \text{if } \langle a, y \rangle \leq b \\ y + (b - \langle a, y \rangle) / (\ a\ _2^2) a & \text{if } \langle a, y \rangle > b \\ y + (-b - \langle a, y \rangle) / (\ a\ _2^2) a & \text{if } \langle a, y \rangle < -b \end{cases}$	[23, 24]
$\underline{b} \leq Ax \leq \bar{b}$	$u = x - \sum_{i=1}^N \lambda_i(x) / (\ A_{i:}\ _2^2) A_{i:},$ $\lambda_i(x) := \begin{cases} 0 & \text{if } \underline{b}_i \leq \langle A_{i:}, x \rangle \leq \bar{b}_i, \\ \langle A_{i:}, x \rangle - \bar{b}_i & \text{if } \langle A_{i:}, x \rangle > \bar{b}_i, \\ \langle A_{i:}, x \rangle - \underline{b}_i & \text{if } \langle A_{i:}, x \rangle < \underline{b}_i. \end{cases}$	[23]
$x \in [\underline{x}, \bar{x}]$	$u = \sup\{\underline{x}, \inf\{y, \bar{x}\}\}$	[23]
$x \geq 0$	$u = (y)_+ := \max(y, 0)$	[138]
$\ x\ _1 \leq \xi$	iterative scheme	[77, 138]
$\ x\ _2 \leq \xi$	$u = \begin{cases} \xi y / \ y\ _2 & \text{if } \ y\ _2 > \xi \\ y & \text{if } \ y\ _2 \leq \xi \end{cases}$	[23]
$\ x\ _\infty \leq \xi$	$u = \sup\{-\xi I, \inf\{y, \xi I\}\}$	[138]
$\{(x, t) \mid \ x\ _2 \leq t\}$	$u = \begin{cases} 0 & \text{if } \ y\ _2 \leq -t \\ (y, t) & \text{if } \ y\ _2 \leq t \\ 1/2(1 + t/\ y\ _2)(y, \ y\ _2) & \text{if } \ y\ _2 \geq t \end{cases}$	[23]
Exponential cone	iterative scheme	[138]
Epigraphs	iterative scheme	[23]
Sublevel sets	iterative scheme	[23]
Simplex	iterative scheme	[138]
$x \succcurlyeq 0, x = \sum_{i=1}^n \lambda_i u_i u_i^T$	$u = \sum_{i=1}^n (\lambda_i)_+ u_i u_i^T$	[138]
$x \succcurlyeq 0, \text{tr}(x) = 1$	iterative scheme	[138]
$x \succcurlyeq 0, \ \sigma_s(x)\ _\infty \leq 1,$ $x = \sum_{i=1}^n \lambda_i u_i u_i^T$	$u = \sum_{i=1}^n \max(\lambda_i, 1) u_i u_i^T$	[138]

Table 5.2: List of domains C where $\varphi(e) = 0$ can be solved explicitly

defining constraint $c(x)$	solution
$Ax = b$	Proposition 5.1.4
$\langle a, x \rangle = b$	Corollary 5.1.5
$\langle a, x \rangle \leq b$	Proposition 5.1.6
$x \geq 0$	Proposition 5.1.7
$\ x\ _2 \leq \xi$	Proposition 5.1.8

Proposition 5.1.4. *If $C = \{x \in V \mid Ax = b\}$ is an affine set, then the subproblem (3.4) is solved by $u = P_C(-e^{-1}h)$, where*

$$P_C(y) = y - A^\dagger(Ay - b). \quad (5.10)$$

and

$$e = \frac{-\beta_2 + \sqrt{\beta_2^2 - 4\beta_1\beta_3}}{2\beta_1}, \quad (5.11)$$

with

$$\beta_1 := \frac{1}{2}\|A^\dagger b\|_2^2 + Q_0, \quad \beta_2 := \langle A^\dagger(Ah), A^\dagger b \rangle + \gamma, \quad \beta_3 := \frac{1}{2}\|A^\dagger(Ah)\|_2^2 + \frac{1}{2}\|h\|_2^2. \quad (5.12)$$

Proof. The projection operator on C is given by (5.10). This and $y = -e^{-1}h$ give

$$P_C(-e^{-1}h) = -e^{-1}(A^\dagger(Ah + eb) - h).$$

This, together with (5.7), yields

$$\begin{aligned}
eQ(u) + \gamma + \langle h, u \rangle &= e \left(\frac{1}{2}(\|P_C(-e^{-1}h)\|_2^2) + Q_0 \right) + \gamma + \langle h, P_C(-e^{-1}h) \rangle \\
&= \frac{1}{2}\|A^\dagger(Ah + eb)\|_2^2 + \frac{1}{2}\|h\|_2^2 - \langle A^\dagger(Ah + eb), h \rangle + Q_0 e^2 \\
&\quad + \gamma e + \langle A^\dagger(Ah + eb) - h, h \rangle \\
&= \left(\frac{1}{2}\|A^\dagger b\|_2^2 + Q_0 \right) e^2 + (\langle A^\dagger(Ah), A^\dagger b \rangle + \gamma) e \\
&\quad + \frac{1}{2}\|A^\dagger(Ah)\|_2^2 + \frac{1}{2}\|h\|_2^2 \\
&= \beta_1 e^2 + \beta_2 e + \beta_3 = 0,
\end{aligned}$$

where β_1 , β_2 , and β_3 are defined in (5.12). Since the subproblem (3.4) is the maximization, the bigger root of this equation is selected, which is given by (5.11). $\square$

Corollary 5.1.5. *If $C = \{x \in V \mid a^T x = b\}$ is a hyperplane, then the subproblem (3.4) is solved by $u = P_C(-e^{-1}h)$, where*

$$P_C(y) = y - \left(\frac{\langle a, y \rangle - b}{\|a\|_2^2} \right) a, \quad (5.13)$$

and e is given by (5.11) with

$$\beta_1 := \frac{b}{2\|a\|_2^2} + Q_0, \quad \beta_2 := \frac{b\langle a, h \rangle}{\|a\|_2^2} + \gamma, \quad \beta_3 := \frac{1}{2} \frac{\langle a, h \rangle^2}{\|a\|_2^2} - \frac{1}{2} \|h\|_2^2. \quad (5.14)$$

Proof. Since the hyperplane $C = \{x \in V \mid a^T x = b\}$ is an affine set, this is a special case of Proposition 5.1.4. $\square$

Proposition 5.1.6. *If $C = \{x \in V \mid \langle a, x \rangle \leq b\}$ is a halfspace, then the subproblem (3.4) is solved by $u = P_C(-e^{-1}h)$, where*

$$P_C(y) = y - \frac{(\langle a, y \rangle - b)_+}{\|a\|_2^2} a \quad (5.15)$$

and e is given by (5.11) with

$$\beta_1 := Q_0, \quad \beta_2 := \gamma, \quad \beta_3 := -\frac{1}{2} \|h\|_2^2, \quad (5.16)$$

say e_1 , and with β_1 , β_2 , and β_3 is given in (5.14), say e_2 . If $\langle a, h \rangle \geq e_1^{-1}b$ and $\langle a, h \rangle \geq e_2^{-1}b$, then $e = e_1$. If $\langle a, h \rangle \leq e_1^{-1}b$ and $\langle a, h \rangle < e_2^{-1}b$, then $e = e_2$. If $\langle a, h \rangle \geq e_1^{-1}b$ and $\langle a, h \rangle < e_2^{-1}b$, then $e = \max\{e_1, e_2\}$.

Proof. The projection operator on C is given by (5.15). This gives

$$P_C(-e^{-1}h) = -e^{-1} \left(h + \frac{(\langle a, h \rangle + eb)_-}{\|a\|_2^2} a \right). \quad (5.17)$$

If $\langle a, h \rangle \geq -eb$, we obtain

$$P_C(-e^{-1}h) = -e^{-1}h,$$

leading to

$$\begin{aligned} eQ(P_C(-e^{-1}h)) + \gamma + \langle h, P_C(-e^{-1}h) \rangle &= \frac{1}{2} e^{-1} \|h\|_2^2 + Q_0 e + \gamma - e^{-1} \|h\|_2^2 \\ &= Q_0 e^2 + \gamma e - \frac{1}{2} \|h\|_2^2 \\ &= \beta_1 e^2 + \beta_2 e + \beta_3 = 0, \end{aligned}$$

where $\beta_1 := Q_0$, $\beta_2 := \gamma$, and $\beta_3 := -\frac{1}{2}\|h\|_2^2$. This identity leads to a solution of the form (5.11), say e_1 . If $\langle a, h \rangle < -eb$, (5.13) is valid and e is computed by (5.11) where β_1 , β_2 , and β_3 is defined in (5.14), say e_2 . After computing e_1 and e_2 , we check whether the inequalities $\langle a, h \rangle \geq -e_1b$ and $\langle a, h \rangle < -e_2b$ are satisfied. Since the subproblem (3.4) has a solution, at least one of the conditions has to be satisfied. If one of them is satisfied, the corresponding e and (5.17) give the solution. If both of them hold, we consider the solution with bigger e . $\square$

Proposition 5.1.7. *If $C = \{x \in \mathbb{R}^n \mid x_i \geq 0 \quad i = 1, \dots, n\}$ is the nonnegative orthant, then the subproblem (3.4) is solved by $u = P_C(-e^{-1}h)$, where*

$$P_C(y) = (y)_+ \quad (5.18)$$

and e is given by (5.11) with

$$\beta_1 := Q_0, \quad \beta_2 := \gamma, \quad \beta_3 := \frac{1}{2}\|(h)_-\|_2^2 - \langle h, (h)_- \rangle. \quad (5.19)$$

Proof. The projection operator on C is given by (5.18) leading to

$$P_C(-e^{-1}h) = -e^{-1}(h)_-.$$

This and (5.7) imply

$$\begin{aligned} eQ(P_C(-e^{-1}h)) + \gamma + \langle h, P_C(-e^{-1}h) \rangle &= \frac{1}{2}e^{-1}\|(h)_-\|_2^2 + Q_0e + \gamma - e^{-1}\langle h, (h)_- \rangle \\ &= Q_0e^2 + \gamma e + \frac{1}{2}\|(h)_-\|_2^2 - \langle h, (h)_- \rangle \\ &= \beta_1e^2 + \beta_2e + \beta_3 = 0, \end{aligned}$$

where β_1 , β_2 , and β_3 are defined in (5.19), giving the result. $\square$

Proposition 5.1.8. *Let $C = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq \xi\}$ be the Euclidean ball. Then*

$$P_C(y) = \begin{cases} \xi y / \|y\|_2 & \|y\|_2 > \xi, \\ y & \|y\|_2 \leq \xi, \end{cases} \quad (5.20)$$

If $\|e^{-1}h\|_2 \leq \xi$ where e is given by (5.11) with

$$\beta_1 := Q_0, \quad \beta_2 := \gamma, \quad \beta_3 := -\frac{1}{2}\|h\|_2^2, \quad (5.21)$$

then $u = -e^{-1}h$; otherwise, the solution of OSGA's subproblem (3.4) is given by

$$u = -\frac{\xi}{\|h\|_2}h, \quad e = -\frac{2(\gamma + \xi\|h\|_2)}{\xi^2 + 2Q_0}.$$

Proof. The projection operator on C is given by (5.20), leading to

$$P_C(-e^{-1}h) = \begin{cases} -\xi h / \|h\|_2 & \|h\|_2 > e\xi, \\ -e^{-1}h & \|h\|_2 \leq e\xi. \end{cases}$$

We first assume that $\|h\|_2 \leq e\xi$ implying $P_C(-e^{-1}h) = -e^{-1}h$. Substituting this into (5.7) yields

$$\begin{aligned} eQ(P_C(-e^{-1}h)) + \gamma + \langle h, P_C(-e^{-1}h) \rangle &= \frac{1}{2}e^{-1}\|h\|_2^2 + Q_0e + \gamma - e^{-1}\|h\|_2^2 \\ &= Q_0e^2 + \gamma e - \frac{1}{2}\|h\|_2^2 \\ &= \beta_1e^2 + \beta_2e + \beta_3 = 0, \end{aligned}$$

where $\beta_1 := Q_0$, $\beta_2 := \gamma$, and $\beta_3 := -\frac{1}{2}\|h\|_2^2$. Hence e is given by (5.11). If this e satisfies $\|h\|_2 \leq e\xi$, then $u = -e^{-1}h$. Otherwise, we assume that $\|h\|_2 > e\xi$. Substituting $P_C(-e^{-1}h) = -\xi h / \|h\|_2$ into (5.7) yields

$$e \left(\frac{1}{2}\xi^2 + Q_0 \right) + \gamma - \xi\|h\|_2 = 0,$$

implying

$$e = -\frac{2(\gamma + \xi\|h\|_2)}{\xi^2 + 2Q_0}$$

and $u = -\xi h / \|h\|_2$. This completes the proof. $\square$

To solve bound-constrained problems with OSGA, we develop an algorithm that can find the global solution of the subproblem (3.4) by solving a sequence of one-dimensional rational optimization problems (see Algorithm 8 in Section 5.2). Notice that the constraint $C := \{x \in V \mid \|x\|_\infty \leq \xi\}$ is a special case of bound-constrained problem with $\underline{x} = -\xi \mathbf{1}$ and $\bar{x} = \xi \mathbf{1}$ where $\mathbf{1}$ is a n -dimensional vector with all elements equal to unity.

5.1.2 Structured problems with a simple functional constraint

In this section we consider the structured convex constrained problem

$$\begin{aligned} \min \quad & f(Ax) \\ \text{s.t.} \quad & \phi(x) \leq \xi, \end{aligned} \tag{5.22}$$

where $\phi : V \rightarrow \overline{\mathbb{R}}$ is a simple smooth or nonsmooth, real-valued, and convex loss function, and ξ is a real constant. We call the problem (5.22) a **functional constraint** problem. While it the special case of (5.1) with

$$C := \{x \in V \mid \phi(x) \leq \xi\},$$

one can solve OSGA's subproblem (3.4) directly by using the KKT optimality conditions, especially when no efficient method for finding the projection on C is known. Indeed, if a nonsmooth problem can be reformulated in the form (5.1) with a smooth f and a nonsmooth ϕ , then OSGA can solve this nonsmooth problem with the complexity of the order $O(\varepsilon^{-1/2})$, which is optimal for smooth problems.

Example. 5.1.9. (LINEAR INVERSE PROBLEM) *Let $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be an ill-conditioned or singular linear operator and $y \in \mathbb{R}^m$ be a vector of observations. The linear inverse problem is the quest of finding $x \in \mathbb{R}^n$ such that*

$$y = Ax + v, \quad (5.23)$$

with unknown but small additive or impulsive noise $v \in \mathbb{R}^m$. The problem is solvable if one knows additional qualitative information about x . This qualitative information is encoded in a constraint on x , under which the Euclidean norm of v is minimized. Constrained optimization problems resulting from two typical qualitative constraints are

$$\begin{aligned} \min \quad & \frac{1}{2} \|y - Ax\|_2^2 \\ \text{s.t.} \quad & \|x\|_2 \leq \xi, \end{aligned} \quad (5.24)$$

$$\begin{aligned} \min \quad & \frac{1}{2} \|y - Ax\|_2^2 \\ \text{s.t.} \quad & \|x\|_{1,2} \leq \xi, \end{aligned} \quad (5.25)$$

in which ξ is a nonnegative real constant. This problem often occurs in applied sciences and engineering, see [104, 151].

In the reminder of this section we assume that the functional constraint satisfies the **Cottle constraint qualification** [19]

(H1) For all $x \in C$, either $\phi(x) < 0$ or $0 \notin \partial\phi(x)$.

The next result gives the optimality conditions for solving the problem (5.1).

Theorem. 5.1.10. *Let (H1) satisfies for the problem (5.22). Then, for a real constant ξ , the solution u of OSGA's subproblem*

$$\begin{aligned} \min \quad & \frac{-\gamma - \langle h, x \rangle}{Q(x)} \\ \text{s.t.} \quad & \phi(x) \leq \xi, \end{aligned}$$

satisfies either

$$u = -e^{-1}h, \quad \mu = 0, \quad \phi(u) < \xi \quad (5.26)$$

or

$$\frac{1}{\mu} \frac{-eu - h}{Q(u)} \in \partial\phi(u), \quad \mu > 0, \quad \phi(u) = \xi, \quad (5.27)$$

where $e := -(\gamma + \langle h, u \rangle)/Q(u)$.

Proof. Let us define the function

$$E_{\gamma,h} : C \rightarrow \mathbb{R}, \quad E_{\gamma,h}(x) := -\frac{\gamma + \langle h, x \rangle}{Q(x)}.$$

Since this function is differentiable, by differentiating both sides of the equality $E_{\gamma,h}(x)Q(x) = -\gamma - \langle h, x \rangle$ with respect to x , we obtain

$$\partial E_{\gamma,h}(x) = \left\{ \frac{-E_{\gamma,h}(x)x - h}{Q(x)} \right\}. \quad (5.28)$$

In view of the KKT optimality conditions for inequality constrained nonsmooth problems, see [19], we have the optimality condition

$$\begin{cases} 0 \in \partial E_{\gamma,h}(u) + \mu \partial \phi(u), \\ \phi(u) \leq \xi, \\ \mu \geq 0, \\ \mu(\phi(u) - \xi) = 0, \end{cases} \quad (5.29)$$

for (5.1). Now, by substituting (5.28) into (5.29), setting $e := -(\gamma + \langle h, u \rangle)/Q(u)$, and distinguishing between $\mu = 0$ and $\mu > 0$, we obtain either (5.26) or (5.27). $\square$

Theorem 5.1.10 gives the optimality conditions for general function ϕ , however, in view of Theorem 5.1.3, it is especially useful when the projection in $C = \{x \mid \phi(x) \leq \xi\}$ is not efficiently available. In the remainder of this section, we derive the solution of OSGA's subproblem (3.4) for some ϕ such as $\|\cdot\|_2$ and $\|\cdot\|_{1,2}$ that appear in many applications. We already solve OSGA's subproblem (3.4) with the constraint $C = \{x \mid \|x\|_2 \leq \xi\}$ in Proposition 5.1.8, but to show how to apply Theorem 5.1.10 we study it in the next result.

Proposition 5.1.11. *Let V be a real finite-dimensional Hilbert space with the induced norm $\phi(\cdot) = \|\cdot\|_2$. Then OSGA's subproblem (3.4) is solved by*

$$u = -e^{-1}h, \quad e = \frac{-\beta_2 + \sqrt{\beta_2^2 - 4\beta_1\beta_3}}{-2\beta_1}, \quad \mu = 0,$$

where

$$\beta_1 := Q_0, \quad \beta_2 := \gamma, \quad \beta_3 := \frac{1}{2}\|h\|_2^2,$$

if $\phi(u) < \xi$; Otherwise it is solved by

$$u = \frac{\xi}{\|h\|_2}h, \quad e = -\frac{2\|h\|_2(\gamma\|h\|_2 + \xi\|h\|_2)}{\xi^2\|h\|_2^2 + 2Q_0\|h\|_2^2}, \quad \mu = \frac{2(\|h\|_2 + e\xi)\|h\|_2^2}{\|h\|_2^2 + 2Q_0\|h\|_2^2}.$$

Proof. Since $\|\cdot\|_2$ is self-dual, Proposition 2.1.17 implies

$$\partial\phi(u) = \begin{cases} \{g \in V^* \mid \|g\|_2 \leq 1\} & \text{if } u = 0, \\ \frac{u}{\|u\|_2} & \text{if } u \neq 0. \end{cases}$$

We now apply Theorem 5.1.10 leading to two cases: (i) (5.26) holds; (ii) (5.27) holds.

Case (i). The condition (5.26) holds. Then we have $u = -e^{-1}h$. By substituting this into the identity $E_{\gamma,h}(u) = e$, we get

$$e = -\frac{\gamma - \|h\|_2^2 e^{-1}}{\frac{1}{2}\|h\|_2^2 e^{-2} + Q_0},$$

implying

$$Q_0 e^2 + \gamma e - \frac{1}{2}\|h\|_2^2 = 0.$$

By using the bigger root of this equation, we have

$$e = \frac{-\beta_2 + \sqrt{\beta_2^2 - 4\beta_1\beta_3}}{-2\beta_1},$$

where $\beta_1 = Q_0$, $\beta_2 = \gamma$, and $\beta_3 = \frac{1}{2}\|h\|_2^2$.

Case (ii). We first suppose that $u = 0$. Then the condition (5.27) implies that $\xi = 0$ and

$$\frac{1-h}{\mu Q_0} \in \{g \in V^* \mid \|g\|_2 \leq 1\},$$

leading to

$$\|h\|_2 \leq \mu Q_0.$$

Therefore, if $\|h\|_2 \leq \mu Q_0$, then $u = 0$. Otherwise, we have

$$\frac{-eu - h}{\frac{1}{2}\|u\|_2^2 + Q_0} = -\mu \frac{u}{\|u\|_2},$$

giving

$$(-eu - h)\|u\|_2 + \mu \left(\frac{1}{2}\|u\|_2^2 + Q_0 \right) u = 0,$$

leading to

$$(-e\|u\|_2 + \frac{1}{2}\mu\|u\|_2^2 + \mu Q_0)u = \|u\|_2 h. \quad (5.30)$$

This implies that there exist λ such that $u = \lambda h$. By substituting this into $\phi(u) = \|u\|_2 = \xi$ we get

$$\lambda = \frac{\xi}{\|h\|_2}.$$

Now, substituting u into (5.30), we obtain

$$\mu = \frac{2(\|h\|_2 + e\xi)\|h\|_2^2}{\|h\|_2^2 + 2Q_0\|h\|_2^2}. \quad (5.31)$$

It follows from $E_{\gamma,h}(u) = e$ that

$$e = -\frac{2\|h\|_2(\gamma\|h\|_2 + \xi\|h\|_2)}{\xi^2\|h\|_2^2 + 2Q_0\|h\|_2^2}.$$

Therefore, for both cases $\|h\|_2 \leq \mu Q_0$ and $\|h\|_2 > \mu Q_0$, the result is valid. $\square$

In 2004, YUAN and LIN in [161] proposed an interesting regularizer called grouped LASSO for the linear regression. Later KIM et al. in [104] proposed a constrained ridge regression model using the constraint

$$\|x\|_{1,2} \leq \xi,$$

where $\|x\|_{1,2}$ is called $\ell_{1,2}$ group norm defined in (2.1). We consider this constraint in the next result.

Proposition 5.1.12. *Let V be a real finite-dimensional vector space with the induced norm $\phi(\cdot) = \|\cdot\|_{1,2}$. Then OSGA's subproblem (3.4) is solved by*

$$u_{g_i} = -e^{-1}h_{g_i} \quad \text{for all } i = 1, \dots, m,$$

and

$$e = \frac{-\beta_2 + \sqrt{\beta_2^2 - 4\beta_1\beta_3}}{-2\beta_1}, \quad \mu = 0,$$

where

$$\beta_1 := Q_0, \quad \beta_2 := \gamma, \quad \beta_3 := \frac{1}{2}\|h\|_2^2 - \sum_{i=1}^m \|h_{g_i}\|_2^2,$$

if $\phi(u) < \xi$; Otherwise it is solved by

$$u_i = \rho_i h_{g_i}, \quad \rho_i = \frac{\|h_{g_i}\|_2 - \mu(\frac{1}{2}\xi^2 + Q_0)}{e\|h_{g_i}\|_2} \quad \text{for all } i = 1, \dots, m,$$

and

$$e = -\frac{\gamma + \langle h, u \rangle}{\frac{1}{2}\xi^2 + Q_0} = -\frac{2(\gamma + \sum_{i=1}^n \tau_i^2 \|h_{g_i}\|_2^2)}{\sum_{i=1}^n \tau_i^2 \|h_{g_i}\|_2^2 + 2Q_0}, \quad \mu = \frac{2(\sum_{i=1}^m \|h_{g_i}\|_2 + e\xi)}{m(\sum_{i=1}^m \tau_i^2 \|h_{g_i}\|_2^2 + 2Q_0)}.$$

Proof. Let $u = (u_{g_1}, \dots, u_{g_m})$. In view of Proposition 2.1.17, we get

$$\partial\phi(u_{g_i}) = \left\{ \frac{u_{g_i}}{\|u_{g_i}\|_2} \right\} \quad \text{for all } i = 1, \dots, m,$$

leading to

$$\partial\phi(u) = \left\{ \left(\frac{u_{g_1}}{\|u_{g_1}\|_2}, \dots, \frac{u_{g_m}}{\|u_{g_m}\|_2} \right) \right\}.$$

We now apply Theorem 5.1.10 leading to two cases: (i) (5.26) holds; (ii) (5.27) holds.

Case (i). The condition (5.26) holds. Then we have $u_{g_i} = -e^{-1}h_{g_i}$ for $i = 1, \dots, m$. By substituting $u = (u_{g_1}, \dots, u_{g_m})$ into the identity $E_{\gamma,h}(u) = e$, we get

$$e = \frac{-\gamma + \sum_{i=1}^m \|h_{g_i}\|_2^2 e^{-1}}{\frac{1}{2}\|h\|_2^2 e^{-2} + Q_0},$$

implying

$$Q_0 e^2 + \gamma e + \frac{1}{2}\|h\|_2^2 - \sum_{i=1}^m \|h_{g_i}\|_2^2 = 0.$$

By using the bigger root of this equation, we get

$$e = \frac{-\beta_2 + \sqrt{\beta_2^2 - 4\beta_1\beta_3}}{-2\beta_1},$$

where $\beta_1 := Q_0$, $\beta_2 := \gamma$, and $\beta_3 := \frac{1}{2}\|h\|_2^2 - \sum_{i=1}^m \|h_{g_i}\|_2^2$.

Case (ii). The condition (5.27) holds. Then we have

$$\frac{-eu_{g_i} - h_{g_i}}{\frac{1}{2}\|u\|_2^2 + Q_0} = -\mu \frac{u_{g_i}}{\|u_{g_i}\|_2} \quad \text{for all } i = 1, \dots, m.$$

Since $\phi(u) = \|u\| = \xi$, we equivalently get

$$\left(\frac{1}{2}\|u\|_2^2 + Q_0 \right) \left(-\frac{e}{\frac{1}{2}\|u\|_2^2 + Q_0} + \frac{\mu}{\|u_{g_i}\|_2} \right) u_{g_i} = h_{g_i}$$

implying $u_{g_i} = \tau_i h_{g_i}$. If $h_{g_i} = 0$, then $u_{g_i} = 0$. Now let $h_{g_i} \neq 0$. Substituting $u_{g_i} = \tau_i h_{g_i}$ into the previous identity, it follows that

$$\left(\frac{1}{2} \sum_{i=1}^m \tau_i^2 \|h_{g_i}\|_2^2 + Q_0 \right) \left(-\frac{e}{\frac{1}{2} \sum_{i=1}^m \tau_i^2 \|h_{g_i}\|_2^2 + Q_0} + \frac{\mu}{\tau_i \|h_{g_i}\|_2} \right) \tau_i h_{g_i} = h_{g_i}.$$

giving

$$-e\tau_i\|h_{g_i}\|_2 + \mu \left(\frac{1}{2} \sum_{i=1}^m \tau_i^2 \|h_{g_i}\|_2^2 + Q_0 \right) = \|h_{g_i}\|_2 \quad \text{for all } i = 1, \dots, m.$$

Applying a summation from both sides, together with $\sum_{i=1}^m \tau_i \|h_{g_i}\|_2 = \xi$, yields

$$-e\xi + m\mu \left(\frac{1}{2} \sum_{i=1}^m \tau_i^2 \|h_{g_i}\|_2^2 + Q_0 \right) = \sum_{i=1}^m \|h_{g_i}\|_2, \quad (5.32)$$

implying

$$\mu = \frac{2(\sum_{i=1}^m \|h_{g_i}\|_2 + e\xi)}{m(\sum_{i=1}^m \tau_i^2 \|h_{g_i}\|_2^2 + 2Q_0)}.$$

By substituting this into (5.32), we have

$$\tau_i = -\frac{1}{me\|h_{g_i}\|_2} \left(m\|h_{g_i}\|_2 - \sum_{i=1}^m \|h_{g_i}\|_2 - e\xi \right)$$

leading to

$$u = (\tau_1 h_{g_1}, \dots, \tau_m h_{g_m}).$$

By substituting this into $E_{\gamma,h}(u) = e$, we get

$$e = -\frac{\gamma + \langle h, u \rangle}{\frac{1}{2}\xi^2 + Q_0} = -\frac{2(\gamma + \sum_{i=1}^n \tau_i^2 \|h_{g_i}\|_2^2)}{\sum_{i=1}^n \tau_i^2 \|h_{g_i}\|_2^2 + 2Q_0},$$

giving the result. $\square$

5.2 Bound-constrained convex optimization

In this section we consider the bound-constrained convex minimization problem

$$\begin{aligned} \min \quad & f(Ax) \\ \text{s.t.} \quad & x \in \mathbf{x}, \end{aligned} \quad (5.33)$$

where $f : \mathbf{x} \rightarrow \overline{\mathbb{R}}$ is a – smooth or nonsmooth – proper convex function, $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a linear operator, and $\mathbf{x} = [\underline{x}, \bar{x}]$ is an axiparallel box in $\mathbb{R}^m$ in which $\underline{x}$ and $\bar{x}$ are the vectors of lower and upper bounds on the components of x , respectively. (Lower bounds are allowed to take the value $-\infty$, and upper bounds the value $+\infty$.)

It is assumed that the set of optimal solutions of (5.33) is nonempty and the first-order information about the objective function (i.e., for any $x \in \mathbf{x}$, the function

value $f(x)$ and some subgradient $g(x)$ at x are available by a first-order oracle.

Bound-constrained optimization in general is an important problem appearing in many fields of science and engineering, where the parameters describing physical quantities are constrained to be in a given range. Furthermore, it plays a prominent role in the development of general constrained optimization methods since many methods reduce the solution of the general problem to the solution of a sequence of bound-constrained problems.

There are lots of bound-constrained algorithms and solvers for smooth and nonsmooth optimization; here, we mention only those related to our study. LIN & MORÉ in [109] and KIM et al. in [103] proposed Newton and quasi-Newton methods for solving bound-constrained optimization. In 1995, BYRD et al. [47] proposed a limited memory algorithm called LBFGS-B for general smooth nonlinear bound-constrained optimization. BRANCH et al. in [45] proposed a trust-region method to solve this problem. NEUMAIER & AZMI [131] solved this problem by a limited memory algorithm. The smooth bound-constrained optimization problem was also solved by BIRGIN et al. in [38] and HAGER & ZHANG in [89, 90] using nonmonotone spectral projected gradient methods, active set strategy and affine scaling scheme, respectively. Some limited memory bundle methods for solving bound-constrained nonsmooth problems were proposed by KARMITSA & MÄKELÄ [100, 99].

In recent years convex optimization has received much attention because it arises in many applications and is suitable for solving problems involving high-dimensional data. The particular case of bound-constrained convex optimization involving a smooth or nonsmooth objective function also appears in a variety of applications, of which we mention the following:

Example. 5.2.1. (BOUND-CONSTRAINED LINEAR INVERSE PROBLEMS) *Given $A, W \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}$, for $m \geq n$, the bound-constrained least-squares problem is given by*

$$\begin{aligned} \min \quad & \frac{1}{2} \|Ax - b\|_2^2 + \lambda \varphi(x) \\ \text{s.t.} \quad & x \in \mathbf{x}, \end{aligned} \tag{5.34}$$

and the bound-constrained ℓ_1 problem is given by

$$\begin{aligned} \min \quad & \|Ax - b\|_1 + \lambda \varphi(x) \\ \text{s.t.} \quad & x \in \mathbf{x}, \end{aligned} \tag{5.35}$$

where $\mathbf{x} = [x, \bar{x}]$ is a box and φ is a smooth or nonsmooth regularizer, often a weighted power of a norm.

The problems (5.34) and (5.35) are commonly arising in the context of control and inverse problems, especially for some imaging problems like denoising,

deblurring and inpainting. MORINI et al. [114] formulated the bound-constrained least-squares problem (5.34) as a nonlinear system of equations and proposed an iterative method based on a reduced Newton's method. Recently, ZHANG & MORINI [163] used alternating direction methods to solve these problems. More recently, CHAN et al. in [56], BOT et al. in [39], and BOT & HENDRICH in [41] proposed alternating direction methods, primal-dual splitting methods, and a Douglas-Rachford primal-dual method, respectively, to solve both (5.34) and (5.35) for some applications.

In the next section we investigate the solution of a bound-constrained version of the subproblem (3.4) and give two iterative schemes, where the first one solves (3.4) exactly and the second one solves it approximately.

5.2.1 Explicit solution of OSGA's rational subproblem

In this section we describe an explicit solution of the bound-constrained subproblem (3.4).

Without loss of generality, we here consider $V = \mathbb{R}^n$. It is not hard to adapt the results to $V = \mathbb{R}^{n \times n}$ and other spaces. The method is related to one used in several earlier researches. In 1980, HELGASON et al. [93] characterized the solution of singly constrained quadratic problem with bound constraints. Later, PARADOLAS & KOVOOR [137] developed an $O(n)$ algorithm for this problem using binary search to solve the associated Kuhn-Tucker system. This problem was also solved by DAI & FLETCHER [70] using a projected gradient method. ZHANG et al. [164] solved the linear support vector machine problem by a cutting plane method employing a similar technique.

In the articles mentioned, the key is showing that the problem can be reduced to a piecewise linear problem in a single dimension. To apply this idea to the present problem, we prove that (3.4) is equivalent to an one-dimensional minimization problem and then develop a procedure to calculate its minimizer. We write

$$x(\lambda) := \sup\{\underline{x}, \inf\{x^0 - \lambda h, \bar{x}\}\} \quad (5.36)$$

for the projection of $x^0 - \lambda h$ to the box $\mathbf{x}$.

Proposition 5.2.2. *For $h \neq 0$, the maximum of the subproblem (3.4) is attained at $\hat{x} := x(\lambda)$, where $\lambda > 0$ or $\lambda = +\infty$ is the inverse of the value of the maximum.*

Proof. The function $E_{\gamma,h} : V \rightarrow \mathbb{R}$ defined by (3.5) is continuously differentiable and from Proposition 3.1.1 we have $e := E_{\gamma,h} > 0$. By differentiating both side of the equation $E_{\gamma,h}(x)Q(x) = -\gamma - \langle h, x \rangle$, we obtain

$$\frac{\partial E_{\gamma,h}}{\partial x} Q(x) = -e(x - x^0) - h.$$

At the maximizer $\hat{x}$, we have $eQ(\hat{x}) = -\gamma - \langle h, \hat{x} \rangle$. Now the first-order optimality conditions imply that for $i = 1, 2, \dots, n$,

$$-e(\hat{x}_i - x_i^0) - h_i \begin{cases} \leq 0 & \text{if } \hat{x}_i = \underline{x}_i, \\ \geq 0 & \text{if } \hat{x}_i = \bar{x}_i, \\ = 0 & \text{if } \underline{x}_i < \hat{x}_i < \bar{x}_i. \end{cases} \quad (5.37)$$

Since $e > 0$, we may define $\lambda := e^{-1}$ and find that, for $i = 1, 2, \dots, n$,

$$\hat{x}_i = \begin{cases} \underline{x}_i & \text{if } \underline{x}_i \geq x_i^0 - \lambda h_i, \\ \bar{x}_i & \text{if } \bar{x}_i \leq x_i^0 - \lambda h_i, \\ x_i^0 - \lambda h_i & \text{if } \underline{x}_i \leq x_i^0 - \lambda h_i \leq \bar{x}_i. \end{cases} \quad (5.38)$$

This implies that $\hat{x} = x(\lambda)$. □

Proposition 5.2.2 gives the key feature of the solution of the subproblem (3.4) implying that it can be considered in the form of (5.36) with only one variable λ . In the remainder of this section, we focus on deriving the optimal λ .

Example. 5.2.3. *Let first consider a very special case that $\mathbf{x}$ is nonnegative orthant, i.e., $\underline{x}_i = 0$ and $\bar{x}_i = +\infty$, for $i = 1, \dots, n$. Nonnegativity constraint is important in many applications, see [20, 80, 81, 101, 102]. In this case we consider the quadratic function*

$$Q(z) := \frac{1}{2} \|z\|_2^2 + Q_0, \quad (5.39)$$

where $Q_0 > 0$. In [1], it is proved that (5.39) is a prox-function. By using this prox-function and (5.36), we obtain

$$x(\lambda) = \sup\{\underline{x}, \inf\{-\lambda h, \bar{x}\}\} = -\lambda(h)_-,$$

where $(z)_- = \min\{0, z\}$. By Proposition 3.1.2, we have

$$\begin{aligned} \frac{1}{\lambda} \left(\frac{1}{2} \|x(\lambda)\|_2^2 + Q_0 \right) + \gamma + \langle h, x(\lambda) \rangle &= \left(\frac{1}{2} \|(h)_-\|_2^2 + \langle h, x(\lambda) \rangle \right) \lambda^2 + \gamma \lambda + Q_0 \\ &= 0. \end{aligned}$$

By substituting $\lambda = 1/e$ into this equation, we get

$$\beta_1 e^2 + \beta_2 \lambda + \beta_3 = 0,$$

where $\beta_1 = Q_0$, $\beta_2 = \gamma$, and $\beta_3 = \frac{1}{2} \|(h)_-\|_2^2 - \langle h, (h)_- \rangle$. Since we search for the maximum e , the solution is the bigger root of this equation, i.e.,

$$e = \frac{-\beta_2 + \sqrt{\beta_2^2 - 4\beta_1\beta_3}}{2\beta_1}.$$

This shows that for the nonnegativity constraint the subproblem (3.4) can be solved in a closed form, however, for a general bound-constrained problem, we need a much more sophisticated scheme to solve (3.4).

To derive the optimal $\lambda \geq 0$ in Proposition 5.2.2, we first determine its permissible range provided by the three conditions considered in (5.38) leading to the interval

$$\lambda \in [\underline{\lambda}_i, \bar{\lambda}_i], \quad (5.40)$$

for each component of x . In particular, if $h_i = 0$, since x^0 is a feasible point, $\hat{x}_i = x_i^0 - \lambda h_i = x_i^0$ satisfies the third condition in (5.38). Thus there is no upper bound for λ , leading to

$$\underline{\lambda}_i = 0, \quad \bar{\lambda}_i = +\infty \quad \text{if } \hat{x}_i = x_i^0, \quad h_i = 0. \quad (5.41)$$

If $h_i \neq 0$, we consider the three cases (i) $\underline{x}_i \geq x_i^0 - \lambda h_i$, (ii) $\bar{x}_i \leq x_i^0 - \lambda h_i$, and (iii) $\underline{x}_i \leq x_i^0 - \lambda h_i \leq \bar{x}_i$ of (5.38). In Case (i), if $h_i < 0$, division by h_i implies that $\lambda \leq -(\underline{x}_i - x_i^0)/h_i \leq 0$, which is not in the acceptable range for λ . In this case, if $h_i > 0$ then $\lambda \geq -(\underline{x}_i - x_i^0)/h_i$ leading to

$$\underline{\lambda}_i = -\frac{\underline{x}_i - x_i^0}{h_i}, \quad \bar{\lambda}_i = +\infty \quad \text{if } \hat{x}_i = \underline{x}_i, \quad h_i > 0. \quad (5.42)$$

In Case (ii), if $h_i < 0$ then $\lambda \geq -(\bar{x}_i - x_i^0)/h_i$ implying

$$\underline{\lambda}_i = -\frac{\bar{x}_i - x_i^0}{h_i}, \quad \bar{\lambda}_i = +\infty \quad \text{if } \hat{x}_i = \bar{x}_i, \quad h_i < 0. \quad (5.43)$$

In Case (ii), if $h_i > 0$ then $\lambda \leq -(\bar{x}_i - x_i^0)/h_i \leq 0$, which is not in the acceptable range of λ . In Case (iii), if $h_i < 0$, division by h_i implies

$$-\frac{\underline{x}_i - x_i^0}{h_i} \leq \lambda \leq -\frac{\bar{x}_i - x_i^0}{h_i}.$$

The lower bound satisfies $-(\underline{x}_i - x_i^0)/h_i \leq 0$, so it is not acceptable, leading to

$$\underline{\lambda}_i = 0, \quad \bar{\lambda}_i = -\frac{\bar{x}_i - x_i^0}{h_i} \quad \text{if } \hat{x}_i = x_i^0 - \lambda h_i \in [\underline{x}_i, \bar{x}_i], \quad h_i < 0. \quad (5.44)$$

In Case (iii), if $h_i > 0$, then

$$-\frac{\bar{x}_i - x_i^0}{h_i} \leq \lambda \leq -\frac{\underline{x}_i - x_i^0}{h_i}.$$

However, the lower bound $-(\bar{x}_i - x_i^0)/h_i \leq 0$ is not acceptable, i.e.,

$$\underline{\lambda}_i = 0, \quad \bar{\lambda}_i = -\frac{\underline{x}_i - x_i^0}{h_i} \quad \text{if } \hat{x}_i = x_i^0 - \lambda h_i \in [\underline{x}_i, \bar{x}_i], \quad h_i > 0. \quad (5.45)$$

As a result, the following proposition is valid.

Proposition 5.2.4. *If $x(\lambda)$ is solution of the problem (3.4), then*

$$\lambda \in [\underline{\lambda}_i, \bar{\lambda}_i] \quad i = 1, \dots, n,$$

where $\underline{\lambda}_i$ and $\bar{\lambda}_i$ are computed by

$$\underline{\lambda}_j = \begin{cases} -\frac{x_i - x_i^0}{h_i} & \text{if } \hat{x}_i = \underline{x}_i, h_i > 0, \\ -\frac{\bar{x}_i - x_i^0}{h_i} & \text{if } \hat{x}_i = \bar{x}_i, h_i < 0, \\ 0 & \text{if } \tilde{x}_i \in [\underline{x}_i, \bar{x}_i], h_i < 0, \\ 0 & \text{if } \tilde{x}_i \in [\underline{x}_i, \bar{x}_i], h_i > 0, \\ 0 & \text{if } h_i = 0, \end{cases} \quad \bar{\lambda}_j = \begin{cases} +\infty & \text{if } \hat{x}_i = \underline{x}_i, h_i > 0, \\ +\infty & \text{if } \hat{x}_i = \bar{x}_i, h_i < 0, \\ -\frac{\bar{x}_i - x_i^0}{h_i} & \text{if } \tilde{x}_i \in [\underline{x}_i, \bar{x}_i], h_i < 0, \\ -\frac{x_i - x_i^0}{h_i} & \text{if } \tilde{x}_i \in [\underline{x}_i, \bar{x}_i], h_i > 0, \\ +\infty & \text{if } h_i = 0, \end{cases} \quad (5.46)$$

in which $\tilde{x}_i = x_i^0 - \lambda h_i$ for $i = 1, \dots, n$.

Proposition 5.2.4 implies that each element of x satisfies in only one of the conditions (5.41)–(5.45). Thus, for each $i = 1, \dots, n$ with $h_i \neq 0$, we have a single breakpoint

$$\tilde{\lambda}_i := \begin{cases} -\frac{\bar{x}_i - x_i^0}{h_i} & \text{if } h_i < 0, \\ -\frac{x_i - x_i^0}{h_i} & \text{if } h_i > 0, \\ +\infty & \text{if } h_i = 0. \end{cases} \quad (5.47)$$

Sorting the n bounds $\tilde{\lambda}_i$, $i = 1, \dots, n$, in increasing order, augmenting the resulting list by 0 and $+\infty$, and deleting possible duplicate points, we obtain a list of $m + 1 \leq n + 2$ different breakpoints, denoted by

$$0 = \lambda_1 < \lambda_2 < \dots < \lambda_m < \lambda_{m+1} = +\infty. \quad (5.48)$$

By construction, $x(\lambda)$ is linear in each interval $[\lambda_k, \lambda_{k+1}]$, for $k = 1, \dots, m$. The next proposition gives an explicit representation for $x(\lambda)$.

Proposition 5.2.5. *The solution $x(\lambda)$ of the auxiliary problem (3.4) defined by (5.36) has the form*

$$x(\lambda) = p^k + \lambda q^k \quad \text{for } \lambda \in [\lambda_k, \lambda_{k+1}] \quad (k = 1, 2, \dots, m), \quad (5.49)$$

where

$$p_i^k = \begin{cases} x_i^0 & \text{if } h_i = 0, \\ x_i^0 & \text{if } \lambda_{k+1} \leq \tilde{\lambda}_i, \\ \underline{x}_i & \text{if } \lambda_k \geq \tilde{\lambda}_i, h_i > 0, \\ \bar{x}_i & \text{if } \lambda_k \geq \tilde{\lambda}_i, h_i < 0, \end{cases} \quad q_i^k = \begin{cases} 0 & \text{if } h_i = 0, \\ -h_i & \text{if } \lambda_{k+1} \leq \tilde{\lambda}_i, \\ 0 & \text{if } \lambda_k \geq \tilde{\lambda}_i, h_i > 0, \\ 0 & \text{if } \lambda_k \geq \tilde{\lambda}_i, h_i < 0. \end{cases} \quad (5.50)$$

Proof. Since $\lambda > 0$, there exists $k \in \{1, \dots, m\}$ such that $\lambda \in [\lambda_k, \lambda_{k+1}]$. Let $i \in \{1, \dots, n\}$. If $h_i = 0$, (5.41) implies $\hat{x}_i = x_i^0$. If $h_i \neq 0$, the way of construction of λ_i for $i = 1, \dots, m$ implies that $\tilde{\lambda}_i \notin (\lambda_k, \lambda_{k+1})$, so two cases are distinguished: (i) $\lambda_{k+1} \leq \tilde{\lambda}_i$; (ii) $\lambda_k \geq \tilde{\lambda}_i$. In Case (i), Proposition 5.2.4 implies that $\tilde{\lambda}_i = \bar{\lambda}_i$, while it is not possible $\tilde{\lambda}_i \neq \underline{\lambda}_i$. Therefore, either (5.44) or (5.45) holds dependent on the sign of h_i , implying $x_i^0 - \lambda h_i \in [x_i, \bar{x}_i]$, so that $p_i^k = x_i^0$ and $q_i^k = -h_i$. In Case (ii), Proposition 5.2.4 implies that $\tilde{\lambda}_i = \underline{\lambda}_i$, while it is not possible $\tilde{\lambda}_i \neq \bar{\lambda}_i$. Therefore, either (5.42) or (5.43) holds. If $h_i < 0$, then (5.43) holds, i.e., $p_i^k = \bar{x}_i$ and $q_i^k = 0$. Otherwise, (5.42) holds, implying $p_i^k = x_i$ and $q_i^k = 0$. This proves the claim. $\square$

Proposition 5.2.5 exhibits the solution $x(\lambda)$ of the auxiliary problem (3.4) as a piecewise linear function of λ . In the next result, we show that solving the problem (3.4) is equivalent to maximizing a one-dimensional piecewise rational function.

Proposition 5.2.6. *The maximal value of the subproblem (3.4) is the maximum of the piecewise rational function $e(\lambda)$ defined by*

$$e(\lambda) := \frac{a_k + b_k \lambda}{c_k + d_k \lambda + s_k \lambda^2} \quad \text{if } \lambda \in [\lambda_k, \lambda_{k+1}] \quad (k = 1, 2, \dots, m), \quad (5.51)$$

where

$$\begin{aligned} a_k &:= -\gamma - \langle h, p^k \rangle, \quad b_k := -\langle h, q^k \rangle, \\ c_k &:= Q_0 + \frac{1}{2} \|p^k - x^0\|^2, \quad d_k := \langle p^k - x^0, q^k \rangle, \quad s_k := \frac{1}{2} \|q^k\|^2. \end{aligned}$$

Moreover, $c_k > 0$, $c_k > 0$ and $4s_k c_k > d_k^2$.

Proof. By Proposition 5.2.2 and 5.2.5, the global minimizer of (3.4) has the form (5.49). For $k = 1, 2, \dots, m$ and $\lambda \in [\lambda_k, \lambda_{k+1}]$, we substitute (5.49) into the function (3.5), and obtain

$$\begin{aligned} E_{\gamma, h}(x(\lambda)) &= -\frac{\gamma + \langle h, x^k(\lambda) \rangle}{Q(x^k(\lambda))} = -\frac{\gamma + \langle h, p^k + q^k \lambda \rangle}{Q(p^k + q^k \lambda)} \\ &= -\frac{\gamma + \langle h, p^k \rangle + \langle h, q^k \rangle \lambda}{Q_0 + \frac{1}{2} \|p^k - x^0\|^2 + \langle p^k - x^0, q^k \rangle \lambda + \frac{1}{2} \|q^k\|^2 \lambda^2} = e(\lambda), \end{aligned} \quad (5.52)$$

as defined in the proposition.

Since $Q_0 > 0$, we have $c_k > 0$ and the denominator of (5.51) is bounded away from zero, implying $4s_k c_k > d_k^2$. It is enough to verify $s_k > 0$. The definition of q_k in (5.50) implies that $h_i \neq 0$ for $i \in I = \{i : \lambda_{k+1} \leq \tilde{\lambda}_i\}$ leading to $q^k \neq 0$ implying $s_k > 0$. $\square$

The next result leads to a systematic way to maximize the one-dimensional rational problem (5.51).

Proposition 5.2.7. *Let a, b, c, d , and s be real constants with $c > 0$, $s > 0$, and $4sc > d^2$. Then*

$$\phi(\lambda) := \frac{a + b\lambda}{c + d\lambda + s\lambda^2} \quad (5.53)$$

defines a function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ that has at least one stationary point. Moreover, the global maximizer of ϕ is determined by the following cases:

(i) *If $b \neq 0$, then $a^2 - b(ad - bc)/s > 0$ and the global maximum*

$$\phi(\hat{\lambda}) = \frac{b}{2s\hat{\lambda} + d} \quad (5.54)$$

is attained at

$$\hat{\lambda} = \frac{-a + \sqrt{a^2 - b(ad - bc)/s}}{b}. \quad (5.55)$$

(ii) *If $b = 0$ and $a > 0$, the global maximum is*

$$\phi(\hat{\lambda}) = \frac{4as}{4cs - d^2}, \quad (5.56)$$

attained at

$$\hat{\lambda} = -\frac{d}{2s}. \quad (5.57)$$

(iii) *If $b = 0$ and $a \leq 0$, the maximum is $\phi(\hat{\lambda}) = 0$, attained at $\hat{\lambda} = +\infty$ for $a < 0$ and at all $\lambda \in \mathbb{R}$ for $a = 0$.*

Proof. The denominator of (5.53) is positive for all $\lambda \in \mathbb{R}^n$ iff the stated condition on the coefficients hold. By the differentiation of ϕ and using the first-order optimality condition, we obtain

$$\phi'(\lambda) = \frac{b(c + d\lambda + s\lambda^2) - (a + b\lambda)(d + 2s\lambda)}{(c + d\lambda + s\lambda^2)^2} = -\frac{bs\lambda^2 + 2as\lambda + ad - bc}{(c + d\lambda + s\lambda^2)^2}.$$

For solving $\phi'(\lambda) = 0$, we consider possible solutions of the quadratic equation $bs\lambda^2 + 2as\lambda + ad - bc = 0$. Using the assumption $4sc > d^2$, we obtain

$$\begin{aligned} (2as)^2 - 4bs(ad - bc) &= (2as)^2 - (4abds - 4b^2cs) \\ &= (2as)^2 - 4abds - b^2(d^2 - 4cs - d^2) \\ &= (2as)^2 - 4abds + (bd)^2 - b^2(d^2 - 4cs) \\ &\geq (2as - bd)^2 - b^2(d^2 - 4cs) \geq 0, \end{aligned}$$

leading to

$$a^2 - \frac{b(ad - bc)}{s} \geq 0,$$

implying that $\phi'(\lambda) = 0$ has at least one solution.

(i) If $b \neq 0$, then

$$a^2 - \frac{b(ad - bc)}{s} = a^2 - \frac{bd}{s}a - \frac{b^2c}{s} = \left(a - \frac{bd}{2s}\right)^2 + \frac{b^2}{4s^2}(4sc - d^2) > 0,$$

implying there exist two solutions. Solving $\phi'(\lambda) = 0$, the stationary points of the function are found to be

$$\lambda = \frac{-a \pm \sqrt{a^2 - b(ad - bc)/s}}{b}. \quad (5.58)$$

Therefore, $a + b\lambda = \pm w$ with

$$w := \sqrt{a^2 - b(ad - bc)/s} > 0,$$

and we have

$$\phi(\lambda) = \frac{\pm w}{c + d\lambda + s\lambda^2}. \quad (5.59)$$

Since the denominator of this fraction is positive and $w \geq 0$, the positive sign in equation (5.58) gives the maximizer, implying that (5.55) is satisfied. Finally, substituting this maximizer into (5.59) gives

$$\begin{aligned} \phi(\hat{\lambda}) &= \frac{w}{c + d\hat{\lambda} + s\hat{\lambda}^2} = \frac{b^2w}{b^2c + bd(w - a) + s(w - a)^2} \\ &= \frac{b^2w}{a^2s - b(ad - bc) + sw^2 + (bd - 2as)w} = \frac{b^2w}{2sw^2 + (bd - 2as)w} \\ &= \frac{b^2w}{w(2s(w - a) + bd)} = \frac{b}{2s\hat{\lambda} + d}, \end{aligned}$$

so that (5.54) holds.

(ii) If $b = 0$, we obtain

$$\phi'(\lambda) = \frac{-a(d + 2s\lambda)}{(c + d\lambda + s\lambda^2)^2}.$$

Hence the condition $\phi'(\lambda) = 0$ implies that $a = 0$ or $d + 2s\lambda = 0$. The latter case implies

$$\hat{\lambda} = -\frac{d}{2s}, \quad \phi(\hat{\lambda}) = \frac{4as}{4cs - d^2},$$

whence $\hat{\lambda}$ is a stationary point of ϕ . If $a > 0$, its maximizer is $\hat{\lambda} = -\frac{d}{2s}$ and (5.56) is satisfied.

(iii) If $b = 0$ and $a < 0$, then

$$\lim_{\lambda \rightarrow -\infty} \phi(\lambda) = \lim_{\lambda \rightarrow +\infty} \phi(\lambda) = 0$$

implies $\phi(\hat{\lambda}) = 0$ at $\hat{\lambda} = \pm\infty$. In case $a = 0$, $\phi(\lambda) = 0$ for all $\lambda \in \mathbb{R}$. $\square$

We summarize the results of Propositions 5.2.2–5.2.7 into the following algorithm for computing the global optimizer x_b and the optimum e_b of (3.4).

Algorithm 8: BCSS (bound-constrained subproblem solver)

Input: $Q_0, x^0, h, \underline{x}, \bar{x}$;

Output: $u_b = U(\gamma, h), e_b = e(x_b)$;

```

1 begin
2   for  $i = 1, 2, \dots, n$  do
3     find  $\hat{\lambda}_i$  by (5.47) using  $\underline{x}$  and  $\bar{x}$ ;
4   end
5   determine the breakpoints  $\lambda_k, k = 1, \dots, m+1$ , by (5.48);
6    $e_b = 0$ ;
7   for  $k = 1, 2, \dots, m$  do
8     compute  $p^k$  and  $q^k$  using (5.50);
9     construct  $e(\lambda)$  using (5.51) for  $[\lambda_k, \lambda_{k+1}]$ ;
10    find the maximizer  $\hat{\lambda}$  of  $e(\lambda)$  using Proposition 5.2.7;
11    if  $\hat{\lambda} \in [\lambda_k, \lambda_{k+1}]$  then
12      compute  $e^k = e(\hat{\lambda})$  using Proposition 5.2.7;
13       $\hat{\lambda}^k = \hat{\lambda}$ ;
14    else
15       $e^k = \max\{e(\lambda_k), e(\lambda_{k+1})\}$ ;
16       $\hat{\lambda}^k = \operatorname{argmax}_{i \in \{k, k+1\}} \{e(\lambda_i)\}$ ;
17    end
18     $E(k) = e^k, LAM(k) = \hat{\lambda}^k$ ;
19  end
20   $j = \operatorname{argmax}\{E(i) \mid i = 1, \dots, m\}$ ;
21   $e_b = E(j), \hat{\lambda} = LAM(j), u_b = x^0 - \hat{\lambda}h$ ;
22 end
```

5.2.2 Inexact solution of OSGA's rational subproblem

In this section we give a scheme to compute an inexact solution for the subproblem (3.4).

We here use the quadratic prox-function (5.39). In view of Proposition 5.2.2 and Theorem 5.1.3, the solution of the subproblem (3.4) is given by $x(\lambda)$ defined in (5.36), where λ can be computed by solving the one-dimensional nonlinear equation

$$\varphi(\lambda) = 0,$$

in which

$$\varphi(\lambda) := \frac{1}{\lambda} \left(\frac{1}{2} \|x(\lambda)\|_2^2 + Q_0 \right) + \gamma + \langle h, x(\lambda) \rangle. \quad (5.60)$$

The solution of OSGA's subproblem can be found by Algorithm 7 (OSS). In Section 5.1, it is shown that in many convex domains the nonlinear equation (5.60) can be solved explicitly, however, for the bound-constrained problems it can be only solved approximately.

As discussed in Section 5.1, the one-dimensional nonlinear equation can be solved by some zero-finder schemes such as the bisection method and the secant bisection scheme described in Chapter 5 of [128]. One can also use the MATLAB's `fzero` function combining the bisection scheme, the inverse quadratic interpolation, and the secant method. In the next section we will use this inexact solution of OSGA's rational subproblem (3.4) for solving large-scale imaging problems.

Chapter 6

Solving nonsmooth convex problems with complexity $O(\varepsilon^{-1/2})$

In this chapter we consider a class of structured nonsmooth convex optimization problems, reformulate the problem in such a way that the nonsmooth terms only appear in a single constraint and the reformulated objective is smooth with Lipschitz continuous gradients, and give a new setup of OSGA that can solve the problem by the complexity $O(\varepsilon^{-1/2})$.

6.1 Structured convex optimization problems

Let us consider the convex constrained problem

$$\begin{aligned} \min \quad & f(Ax, \phi(x)) \\ \text{s.t.} \quad & x \in C, \end{aligned} \tag{6.1}$$

where $f : U \times \mathbb{R} \rightarrow \overline{\mathbb{R}}$ is a proper and convex function that is smooth with Lipschitz continuous gradients with respect to both arguments and monotone increasing with respect to the second argument, $A : V \rightarrow U$ is a linear operator, $C \subseteq V$ is a simple convex domain, and $\phi : V \rightarrow \mathbb{R}$ is a simple nonsmooth, real-valued, and convex loss function. This class of convex problems generalizes the composite problem considered in [123, 124]. As discussed in Chapter 2, OSGA attains the complexity $O(\varepsilon^{-2})$ for this class of problems. Hence we aim to reformulate the problem (6.1) in such a way that OSGA attains the complexity $O(\varepsilon^{-1/2})$. We here reformulate the problem (6.1) in the form

$$\begin{aligned} \min \quad & \widehat{f}(x, \xi) \\ \text{s.t.} \quad & x \in \widehat{C}, \end{aligned} \tag{6.2}$$

where

$$\widehat{f}(x, \xi) := f(Ax, \xi), \quad (6.3)$$

$$\widehat{C} := \{(x, \xi) \in V \times \mathbb{R} \mid x \in C, \phi(x) \leq \xi\}. \quad (6.4)$$

By the assumptions about f , the reformulated function $\widehat{f}$ is smooth and has Lipschitz continuous gradients. OSGA can handle the problems of the form (6.2) with the complexity $O(\varepsilon^{-1/2})$ in the price of adding a functional constraint to the feasible domain C . In the next section we will show that how OSGA can effectively handle (6.2) with the feasible domain $\widehat{C}$.

Problems of the form (6.1) appear in many applications in the fields of signal and image processing, machine learning, statistics, economic, geophysics, and inverse problems. In the remainder of this section we deal with such applications, however, we here mention the following example.

Example. 6.1.1. (COMPOSITE MINIMIZATION) *We consider the unconstrained minimization problem*

$$\begin{aligned} \min \quad & f(Ax) + \phi(x) \\ \text{s.t.} \quad & x \in C, \end{aligned} \quad (6.5)$$

where $f : U \rightarrow \overline{\mathbb{R}}$ is a proper, smooth, and convex function, $A : V \rightarrow U$ is a linear operator, and $\phi : V \rightarrow \mathbb{R}$ is a simple but nonsmooth, real-valued, and convex loss function. In this case we reformulate (6.5) in the form (6.2) by defining the problem

$$\begin{aligned} \min \quad & \widetilde{f}(Ax, \xi) \\ \text{s.t.} \quad & \phi(x) \leq \xi, \end{aligned} \quad (6.6)$$

where $\widetilde{f} : \widetilde{C} \rightarrow \mathbb{R}$, $\widetilde{f}(Ax, \xi) := f(Ax) + \xi$ with the feasible set $\widetilde{C}$ is defined by

$$\widetilde{C} := \{(x, \xi) \in C \times \mathbb{R} \mid \phi(x) \leq \xi\}.$$

Consider the linear inverse problem

$$y = Ax + v, \quad (6.7)$$

where $x \in \mathbb{R}^n$ is the original object, $y \in \mathbb{R}^m$ is an observation, and $v \in \mathbb{R}^m$ is additive or impulsive noise. The objective is to recover x from y by solving (6.7). In practice, this problem is typically underdetermined and ill-conditioned, and v is unknown. Hence x typically is approximated by one of the minimization problems

$$\begin{aligned} \min \quad & \frac{1}{2} \|y - Ax\|_2^2 + \frac{1}{2} \lambda \|x\|_2^2 \\ \text{s.t.} \quad & x \in \mathbb{R}^n, \end{aligned} \quad (6.8)$$

$$\begin{aligned} \min \quad & \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1 \\ \text{s.t.} \quad & x \in \mathbb{R}^n, \end{aligned} \quad (6.9)$$

or

$$\begin{aligned} \min \quad & \frac{1}{2} \|y - Ax\|_2^2 + \frac{1}{2} \lambda_1 \|x\|_2^2 + \lambda_2 \|x\|_1 \\ \text{s.t.} \quad & x \in \mathbb{R}^n. \end{aligned} \quad (6.10)$$

These problems can be reformulated in the form (6.5) by setting

$$f(x, \xi) := \frac{1}{2} \|y - Ax\|_2^2 + \xi, \quad \phi(x) := \frac{1}{2} \lambda \|x\|_2^2, \quad (6.11)$$

$$f(x, \xi) := \frac{1}{2} \|y - Ax\|_2^2 + \xi, \quad \phi(x) := \lambda \|x\|_1, \quad (6.12)$$

or

$$f(x, \xi) := \frac{1}{2} \|y - Ax\|_2^2 + \xi, \quad \phi(x) := \frac{1}{2} \lambda_1 \|x\|_2^2 + \lambda_2 \|x\|_1, \quad (6.13)$$

respectively.

6.1.1 Description of OSGA's new setup

This section devotes to a new setup of the optimal subgradient framework to deal with problems of the form (6.2). To this end, we adapt OSGA's subproblem and by introducing an appropriate prox-function show the new subproblem is equivalent to a proximal-like problem. We generally assume that the domain C is simple enough such that $E(\eta, y)$ and $U(\eta, y)$ can be computed cheaply, in $O(n \log n)$ operations, say.

Lemma. 6.1.2. *Let $Q : V \times \mathbb{R} \rightarrow \mathbb{R}$ be a function defined by*

$$Q(x, x_0) := Q_0 + \frac{1}{2} (\|x\|_2^2 + x_0^2), \quad (6.14)$$

where $Q_0 > 0$. Then Q is strongly convex, and $Q(x, x_0) > 0$.

Proof. Since $g_Q(x) = (x \ x_0)^T$, we obtain

$$\begin{aligned} & Q(z, z_0) + \langle g_Q(z, z_0), (x - z, x_0 - z_0) \rangle + \frac{1}{2} \|(x - z, x_0 - z_0)^T\|_2^2 \\ &= Q_0 + \frac{1}{2} \langle (z, z_0)^T, (z, z_0)^T \rangle + \langle (z, z_0)^T, (x - z, x_0 - z_0)^T \rangle \\ &\quad + \frac{1}{2} \langle (x - z, x_0 - z_0)^T, (x - z, x_0 - z_0)^T \rangle \\ &= Q_0 + \frac{1}{2} \langle (z, z_0)^T, (x, x_0)^T \rangle + \frac{1}{2} \langle (x, x_0)^T, (x - z, x_0 - z_0)^T \rangle \\ &= Q_0 + \frac{1}{2} \langle (x, x_0)^T, (x, x_0)^T \rangle = Q_0 + \frac{1}{2} \|(x, x_0)^T\|_2^2 \\ &= Q(x, x_0). \end{aligned}$$

This means that Q is a strongly convex function with the convexity parameter 1, and since $Q_0 > 0$, we get $Q(x, x_0) > 0$. $\square$

Lemma 6.1.2 shows that the quadratic function Q defined by (6.14) is a prox-function. We now replace the linear relaxation (3.2) by

$$f(x, x_0) \geq \gamma + \langle h, x \rangle + h_0 x_0 \quad \text{for all } x \in \widehat{C}. \quad (6.15)$$

By using this linear relaxation and the prox-function (6.14), the subproblem (3.4) is rewritten in the form

$$\begin{aligned} \sup \quad & E_{\gamma, h, h_0}(x) \\ \text{s.t.} \quad & (x, x_0) \in C \times \mathbb{R}, \phi(x) \leq x_0, \end{aligned} \quad (6.16)$$

where $E_{\gamma, h, h_0} : V \times \mathbb{R} \rightarrow \mathbb{R}$ and

$$E_{\gamma, h, h_0}(x, x_0) := \frac{\gamma + \langle h, x \rangle + h_0 x_0}{Q(x, x_0)}, \quad (6.17)$$

which is a differentiable function. The next result gives a bound on the error $f(x_b) - \widehat{f}$, which is important for the complexity analysis of the the new setup of OSGA.

Proposition 6.1.3. *Let $\gamma_b := \gamma - f(x_b)$, $u := U(\gamma_b, h, h_0)$, and $\eta := E(\gamma_b, h, h_0)$. Then we have*

$$0 \leq f(x_b) - \widehat{f} \leq \eta Q(\widehat{x}, \widehat{x}_0). \quad (6.18)$$

In particular, if x_b is not yet optimal then the choice $u = U(\gamma_b, h, h_0)$ implies $E(\gamma_b, h, h_0) > 0$.

Proof. Using (6.15), (6.16), and (6.17), this follows similar to Proposition 3.1.1. $\square$

Proposition 6.1.4. *Let $e := E(\gamma, h, h_0) > 0$ and $u = U(\gamma, h, h_0)$. Then*

$$\gamma + \langle h, u \rangle + h_0 u_0 = -e Q(u, u_0), \quad (6.19)$$

$$\langle e g_Q(u, u_0) + h, x - u \rangle + (e u_0 + h_0)(x_0 - u_0) \geq 0 \quad \text{for all } (x, x_0) \in C \times \mathbb{R}, \phi(x) \leq x_0. \quad (6.20)$$

Proof. The problem (6.16) and the definition (6.17) imply that the function $\zeta : C \times \mathbb{R} \rightarrow \mathbb{R}$ defined by

$$\zeta(x, x_0) := \gamma + \langle h, x \rangle + h_0 x_0 + e Q(x, x_0)$$

is nonnegative and vanishes for $(x, x_0) = (u, u_0) := U(\gamma, h, h_0)$, i.e., the identity (6.19) holds. Since $\zeta(x, x_0)$ is continuously differentiable with gradient $g_\zeta(x, x_0) = (h + \eta g_Q(x), eu_0 + h_0)^T$, the first order optimality condition holds, i.e.,

$$\langle g_\zeta(x, x_0), x - u \rangle + (eu_0 + h_0)(x_0 - u_0) \geq 0 \quad (6.21)$$

for all $(x, x_0) \in C \times \mathbb{R}$, $\phi(x) \leq x_0$, giving the results. $\square$

The next result gives a systematic way for solving OSGA's subproblem (6.16) for problems of the form (6.2).

Theorem. 6.1.5. *Let $(u, u_0) \in V \times \mathbb{R}$ be a minimizer of (6.16) and $e = E_{\gamma, h, h_0}(u, u_0)$. Then*

$$u := u(e, \lambda), \quad u_0 := \phi(u),$$

where $y := -e^{-1}h$, $\lambda := u_0 + e^{-1}h_0$, and

$$\hat{u} := u(e, \lambda) := \operatorname{argmin}_{x \in C} \frac{1}{2} \|x - y\|_2^2 + \lambda \phi(x). \quad (6.22)$$

Furthermore, e and λ can be computed by solving the two-dimensional system of equations

$$\begin{cases} \phi(\hat{u}) + e^{-1}h_0 - \lambda = 0, \\ e \left(\frac{1}{2} (\|\hat{u}\|_2^2 + \phi(\hat{u})^2) + Q_0 \right) + \gamma + \langle h, \hat{u} \rangle + h_0 \phi(\hat{u}) = 0. \end{cases} \quad (6.23)$$

Proof. From Proposition 6.1.4, at the minimizer (u, u_0) , we obtain

$$e \left(\frac{1}{2} (\|u\|_2^2 + u_0^2) + Q_0 \right) = -\gamma - \langle h, u \rangle - h_0 u_0 \quad (6.24)$$

and

$$\langle eu + h, x - u \rangle + (eu_0 + h_0)(x_0 - u_0) \geq 0 \quad \forall (x, x_0) \in C \times \mathbb{R}, \phi(x) \leq x_0. \quad (6.25)$$

We conclude the proof in the next two steps:

Step 1. We first show that this inequality is equivalent to the following two inequalities

$$\begin{cases} eu_0 + h_0 \geq 0, \\ \langle eg_Q(u, u_0) + h, x - u \rangle + (eu_0 + h_0)(\phi(x) - u_0) \geq 0 \quad \forall (x, x_0) \in C \times \mathbb{R}. \end{cases} \quad (6.26)$$

Assuming that these two inequalities hold, we prove (6.25). From $\phi(x) \leq x_0$ and $eu_0 + h_0 \geq 0$, we obtain

$$\begin{aligned} \langle eg_Q(u, u_0) + h, x - u \rangle + (eu_0 + h_0)(x_0 - u_0) \\ \geq \langle eg_Q(u, u_0) + h, x - u \rangle + (eu_0 + h_0)(\phi(x) - u_0) \geq 0. \end{aligned}$$

We now assume (6.25) and prove (6.26). The inequality $eu_0 + h_0 \geq 0$ holds; Otherwise, by selecting x_0 big enough, we get

$$\langle eg_Q(u, u_0) + h, x - u \rangle + (eu_0 + h_0)(x_0 - u_0) < 0,$$

which is a contradiction with (6.25). Since $\phi(x) \leq x_0$, the second inequality in (6.26) holds.

Step 2. By setting $x = u$ and $u_0 = \phi(u)$, we see that u is a solution of the minimization problem

$$\inf_{x \in C} \langle eg_Q(u, u_0) + h, x - u \rangle + (eu_0 + h_0)(\phi(x) - u_0).$$

The first-order optimality condition (2.9) for this problem leads to

$$0 \in u + e^{-1}h + (u_0 + e^{-1}h_0) \partial \phi(u) + N_C(u). \quad (6.27)$$

By writing the first-order optimality condition (2.11) for the problem

$$\begin{aligned} \min \quad & \frac{1}{2} \|x - y\|_2^2 + \lambda \phi(x) \\ \text{s.t.} \quad & x \in C, \end{aligned}$$

we get

$$0 \in \hat{u} - y + \lambda \partial \phi(\hat{u}) + N_C(\hat{u}). \quad (6.28)$$

By comparing (6.27) and (6.28) and setting $y = -e^{-1}h$, $\lambda = u_0 + e^{-1}h_0$, we conclude that both problems have the same minimizer $u = \hat{u}$. Since $u_0 = \phi(\hat{u})$, we obtain

$$\lambda = u_0 + e^{-1}h_0 = \phi(\hat{u}) + e^{-1}h_0.$$

Using this and substituting $u_0 = \phi(\hat{u})$ in (6.24), e and λ are found by solving the system of nonlinear equations (6.23). This completes the proof. $\square$

In Theorem 6.1.5, if $C = V$, the problem (6.22) is reduced to the classical proximity operator $\hat{u} = \text{prox}_{\lambda \phi}(y)$ defined in (2.10). Hence the problem (6.22) is called *proximal-like*. Therefore, the word “simple” in the definition of C means that the problem (6.22) can be solved efficiently either in a closed form or by an inexpensive iterative scheme. To have a clear view of Theorem 6.1.5, we give the following example.

Example. 6.1.6. Consider the ℓ_1 -regularized least squares problem (6.9). Then the problem can be reformulated as

$$\begin{aligned} \min \quad & \frac{1}{2} \|y - Ax\|_2^2 + \xi \\ \text{s.t.} \quad & \|x\|_1 \leq \xi. \end{aligned}$$

Since $\phi = \|\cdot\|_1$, the solution of (6.22) is $\hat{u} = \text{sign}(y_i)(|y_i| - \lambda, 0)_+$ with $y = -e^{-1}h$ (see Table 6.1). Substituting this into (6.23) gives

$$\begin{aligned} \sum_{i=1}^n (|y_i| - \lambda)_+ + e^{-1}h_0 - \lambda &= 0, \\ e \left(\frac{1}{2} \left(\sum_{i=1}^n (|y_i| - \lambda)_+^2 + \left(\sum_{i=1}^n (|y_i| - \lambda)_+ \right)^2 \right) + Q_0 \right) &+ \gamma + \sum_{i=1}^n (h_i + h_0)(|y_i| - \lambda)_+ = 0. \end{aligned}$$

This is a two-dimensional system of nonsmooth equations that can be reformulated as a nonlinear least squares problem, see, for example, [136].

Theorem 6.1.5 leads to the two-dimensional nonlinear system

$$F(e, \lambda) := (f_1(e, \lambda), f_2(e, \lambda))^T = 0, \quad (6.29)$$

where

$$\begin{aligned} f_1(e, \lambda) &:= \phi(\hat{u}) + e^{-1}h_0 - \lambda, \\ f_2(e, \lambda) &:= e \left(\frac{1}{2} (\|\hat{u}\|_2^2 + \phi(\hat{u})^2) + Q_0 \right) + \gamma + \langle h, \hat{u} \rangle + h_0 \phi(\hat{u}), \end{aligned}$$

in which $\hat{u} = u(e, \lambda)$ and $e, \lambda > 0$. For an instance, in Example 6.1.6, we have

$$\begin{aligned} f_1(e, \lambda) &= \sum_{i=1}^n (|y_i| - \lambda)_+ + e^{-1}h_0 - \lambda, \\ f_2(e, \lambda) &= e \left(\frac{1}{2} \left(\sum_{i=1}^n (|y_i| - \lambda)_+^2 + \left(\sum_{i=1}^n (|y_i| - \lambda)_+ \right)^2 \right) + Q_0 \right) + \gamma \\ &\quad + \sum_{i=1}^n (h_i + h_0)(|y_i| - \lambda)_+. \end{aligned}$$

The system of nonsmooth equations (6.29) can be handled by the bound-constrained least-squares problem

$$\begin{aligned} \min \quad & \frac{1}{2} \|F(e, \lambda)\|_2^2 \\ \text{s.t.} \quad & e, \lambda > 0 \end{aligned} \quad (6.30)$$

if $f_1(e, \lambda)$ and $f_2(e, \lambda)$ are smooth and by replacing the vector (e, λ) with $(|e|, |\lambda|)$ and solving

$$\begin{aligned} \min \quad & \frac{1}{2} \|F(|e|, |\lambda|)\|_2^2 \\ \text{s.t.} \quad & e, \lambda \in \mathbb{R} \end{aligned} \tag{6.31}$$

if $f_1(e, \lambda)$ and $f_2(e, \lambda)$ are nonsmooth. The problem (6.30) can be handled by various bound-constrained nonlinear optimization schemes such as Newton and quasi-Newton methods [47, 109], Levenberg-Marquardt methods [98], and trust-region methods [65, 85]. The problems (6.29) and (6.31), such as Example 6.1.6, can be solved by the semismooth Newton method or the smoothing Newton method [142], the quasi-Newton methods [150, 108], the secant method [140], and trust-region methods [5, 6, 141].

In view of Theorem 6.1.5, we can propose a systematic way for solving OSGA's subproblem (6.16), which is summarized in next scheme.

Algorithm 9: OSSP (OSGA's subproblem solver for (6.22))

Input: Q_0, γ, h ;

Output: u, e ;

1 **begin**

2 solve the system of nonlinear equation (6.29) approximately by a nonlinear solver to find e and λ ;

3 set $u = \hat{u}(e, \lambda)$.

4 **end**

To implement Algorithm 9 (OSSP), we need a reliable nonlinear solver to solve the system of nonlinear equation (6.29) and a routine giving the solution of the proximal-like problem (6.22) effectively. In Section 6.2 we investigate solving the proximal-like problem (6.22) for some practically important loss function ϕ .

6.1.2 Convergence analysis

We here suppose that the assumptions (H1) and (H2) described in Section 3.3 are satisfied. Therefore, the convexity of f implies that the upper level set $N_f(x_0)$ is closed and convex. Since V is a finite-dimensional vector space, (H2) implies that the upper level set $N_f(x_0)$ is convex and compact. It follows from the continuity and properness of the objective function f that it attains its global minimizer on the upper level set $N_f(x_0)$. Therefore, there is at least one minimizer $\hat{x}$, and its corresponding minimum is denoted by $\hat{f}$.

Since the underlying problem (6.2) is a special case of the problem (3.1) considered in Chapter 3, the complexity results for OSGA remains valid.

Theorem. 6.1.7. *Suppose that $f - \mu Q$ is convex and $\mu \geq 0$. Then we have*

(i) (NONSMOOTH COMPLEXITY BOUND) *If the points generated by OSGA stay in a bounded region of the interior of C , or if f is Lipschitz continuous in C , the total number of iterations needed to reach a point with $f(x) \leq f(\hat{x}) + \varepsilon$ is at most $O((\varepsilon^2 + \mu\varepsilon)^{-1})$. Thus the asymptotic worst case complexity is $O(\varepsilon^{-2})$ when $\mu = 0$ and $O(\varepsilon^{-1})$ when $\mu > 0$.*

(ii) (SMOOTH COMPLEXITY BOUND) *If f has Lipschitz continuous gradients with Lipschitz constant L , the total number of iterations needed by OSGA to reach a point with $f(x) \leq f(\hat{x}) + \varepsilon$ is at most $O(\varepsilon^{-1/2})$ if $\mu = 0$, and at most $O(|\log \varepsilon| \sqrt{L/\mu})$ if $\mu > 0$.*

Indeed, if a nonsmooth problem can be reformulated as (6.2) with a nonsmooth loss function ϕ , then OSGA can solve the reformulated problem with the complexity $O(\varepsilon^{-1/2})$ for an arbitrary accuracy parameter ε . The next result shows that the sequence $\{x_k\}$ generated by OSGA is convergent to $\hat{x}$ if the objective f is strictly convex and $\hat{x} \in \text{int}(C)$, where $\text{int}(C)$ denotes the interior of C .

Proposition 6.1.8. *Suppose that f is strictly convex, then the sequence $\{x_k\}$ is generated by OSGA is convergent to $\hat{x}$ if $\hat{x} \in \text{int } C$.*

Proposition 6.1.8 is valid for any strictly convex functions, but if the function is strongly convex, we can get more information.

Proposition 6.1.9. *Suppose that f is strongly convex, then the sequence $\{x_k\}$ is generated by OSGA is convergent to $\hat{x}$ and we have*

$$\|x - \hat{x}\| \leq \left(\frac{2(t_2 + 1)}{\sigma t_1(1 - t_2)} \varepsilon \right)^{1/2},$$

where $t \in [t_1 \ t_2] \subset]0 \ 1[$.

6.2 Solving proximal-like subproblem

In this section we show that the proximal-like problem (6.22) can be solved in closed forms for many special cases appearing in applications. To this end, we first consider unconstrained problems ($C = V$) and study some problems with simple constrained domains ($C \neq V$). We summarize the proximal-like operators in Table 6.1.

Table 6.1: List of available proximal-like operators for several ϕ and C ($D := \text{diag}(d)$, $d \in \mathbb{R}_+$, $Q^T Q = I$)

$\phi(x)$	C	proximity operator $u = \text{prox}_\phi(y)$	Reference
$\lambda \ Dx\ _1$	V	$u_i = \text{sign}(y_i)(y_i - \lambda d_i)_+$	Prop. 6.2.1
$\lambda \ Qx\ _1$	V	$u = Q^T(\text{sign}(y) \cdot (y - \lambda)_+)$	Prop. 11 [64]
$\lambda \ Dx\ _2$	V	$u_i = \begin{cases} 0 & \text{if } \ D^{-1}y\ _2 \leq \lambda, \\ (\tau y_i)/(\tau + \lambda d_i^2), & \text{if } \ D^{-1}y\ _2 > \lambda. \end{cases}$ τ is the solution of $\sum_{i=1}^n (d_i^2 y_i^2)/(\tau + \lambda d_i^2)^2 - 1 = 0$.	Prop. 6.2.2
$\lambda \ x\ _2$	V	$u = (1 - \lambda/\ y\ _2)_+ y$	[138]
$\frac{1}{2} \lambda \ x\ _2^2$	V	$u = 1/(1 + \lambda)y$	[138]
$\frac{1}{2} \lambda_1 \ x\ _2^2 + \lambda_2 \ x\ _1$	V	$u = 1/(1 + \lambda_1) \text{prox}_{\lambda_2 \ \cdot\ _1}(y)$	[138]
$\lambda \ x\ _\infty$	V	$u_i = \begin{cases} 0 & \text{if } \ y\ _1 \leq \lambda, \\ \text{sign}(y_i)u_\infty & \text{if } \ y\ _1 > \lambda, i \in I, \\ y_i & \text{if } \ y\ _1 > \lambda, i \notin I. \end{cases}$ $I = \{l_1, \dots, l_{\hat{k}}\}$, $u_\infty = \frac{1}{\hat{k}}(\sum_{i \in I} y_i - \lambda)$ $\hat{k}$ is the smallest $k \in 1, \dots, n-1$ such that $\frac{1}{k}(\sum_{i \in I} v_i - \lambda) \geq v_{k+1}$, $v_i = y_i $ $l_1, \dots, l_n$ a permutation of $1, \dots, n$ such that $v_1 \geq v_2 \geq \dots \geq v_n$ and otherwise $\hat{k} = n$	Prop. 6.2.3
$\lambda \ x\ _{1,2}$	V	$u_{gi} = (1 - \lambda/\ y_{gi}\ _2)_+ y_{gi}$	P. 6.2.4

Table 6.1: List of available proximal-like operators for several ϕ and C (continued)

$\lambda \ x\ _{1,\infty}$	V	$u_{g_i} = \begin{cases} 0_{g_i} & \text{if } \ y_{g_i}\ _1 \leq \lambda, \\ \text{sign}(y_{g_i}) u_{g_i}^i & \text{if } \ y_{g_i}\ _1 > \lambda, \ i \in I_{g_i}, \\ y_{g_i} & \text{if } \ y_{g_i}\ _1 > \lambda, \ i \notin I_{g_i}. \end{cases}$ $I = \{l_{g_i}^1, \dots, \widehat{l}_{g_i}^k\}$, $u_{g_i}^i = \frac{1}{k_i} \left(\sum_{j \in I_{g_i}} y_{g_i}^j - \lambda \right) \widehat{k}_i$ is the smallest $k \in 1, \dots, n_i - 1$ such that $\frac{1}{k} \left(\sum_{j \in I_{g_i}} v_{g_i}^j - \lambda \right) \geq v_{g_i}^{k+1}$ $v_{g_i}^j = y_{g_i}^j l_{g_i}^1, \dots, l_{g_i}^n$ a permutation of $1, \dots, n$ such that $v_{g_i}^1 \geq v_{g_i}^2 \geq \dots \geq v_{g_i}^{n_i}$ and otherwise $\widehat{k}_i = n_i$	Prop. 6.2.5
$\lambda \ Dx\ _1$	$x \geq 0$	$u_i = \begin{cases} y_i - \lambda d_i & \text{if } J \neq \emptyset, \ y_i > \lambda d_i , \\ y_i + \lambda d_i & \text{if } J \neq \emptyset, \ y_i < -\lambda d_i , \\ 0 & \text{otherwise.} \end{cases}$ $J := \{j \in \{1, \dots, n\} \mid y_j > \lambda d_j \}$	Prop. 6.2.6
$\frac{1}{2} \lambda \ x\ _2^2$	$x \geq 0$	$u_i = \begin{cases} \underline{x}_i & \text{if } y_i \leq 0, \\ y_i / (1 + \lambda) & \text{if } y_i > 0, \end{cases}$	Prop. 6.2.7
$\frac{1}{2} \lambda_1 \ x\ _2^2 + \lambda_2 \ Dx\ _1$	$x \geq 0$	$u_i = \begin{cases} 1/(1 + \lambda)(y_i - \lambda d_i) & \text{if } J \neq \emptyset, \ y_i > \lambda d_i , \\ 1/(1 + \lambda)(y_i + \lambda d_i) & \text{if } J \neq \emptyset, \ y_i < -\lambda d_i , \\ 0 & \text{otherwise.} \end{cases}$ $J := \{j \in \{1, \dots, n\} \mid y_j > \lambda d_j \}$	Prop. 6.2.8

Table 6.1: List of available proximal-like operators for several ϕ and C (continued)

$\lambda \ Dx\ _1$	$[\underline{x}, \bar{x}]$	$u_i = \begin{cases} \underline{x}_i & \text{if } \omega > 0, \underline{x}_i - y_i + \lambda d_i \operatorname{sign}(\underline{x}_i) \geq 0, \\ \bar{x}_i & \text{if } \omega > 0, \bar{x}_i - y_i + \lambda d_i \operatorname{sign}(\bar{x}_i) \leq 0, \\ y_i - \lambda d_i & \text{if } \omega > 0, y_i > \lambda d_i , \\ y_i + \lambda d_i & \text{if } \omega > 0, y_i < -\lambda d_i , \\ 0 & \text{otherwise.} \end{cases}$ $\omega = \sum_{y_i + \lambda d_i < 0} (y_i + \lambda d_i) \underline{x} + \sum_{y_i + \lambda d_i > 0} (y_i + \lambda d_i) \bar{x}$	Prop. 6.2.6
$\frac{1}{2} \lambda \ x\ _2^2$	$[\underline{x}, \bar{x}]$	$u_i = \begin{cases} \underline{x}_i & \text{if } (1 + \lambda) \underline{x}_i \geq y_i, \\ \bar{x}_i & \text{if } (1 + \lambda) \bar{x}_i \leq y_i, \\ y_i / (1 + \lambda) & \text{if } \underline{x}_i < y_i / (1 + \lambda) < \bar{x}_i, \end{cases}$	Prop. 6.2.7
$\frac{1}{2} \lambda_1 \ x\ _2^2 + \lambda_2 \ Dx\ _1$	$[\underline{x}, \bar{x}]$	$u_i = \begin{cases} \underline{x}_i & \text{if } \omega > 0, \underline{x}_i - y_i + \lambda d_i \operatorname{sign}(\underline{x}_i) \geq 0, \\ \bar{x}_i & \text{if } \omega > 0, \bar{x}_i - y_i + \lambda d_i \operatorname{sign}(\bar{x}_i) \leq 0, \\ 1/(1 + \lambda)(y_i - \lambda d_i) & \text{if } \omega > 0, y_i > \lambda d_i , \\ 1/(1 + \lambda)(y_i + \lambda d_i) & \text{if } \omega > 0, y_i < -\lambda d_i , \\ 0 & \text{otherwise.} \end{cases}$ $\omega = \sum_{y_i + \lambda d_i < 0} (y_i + \lambda d_i) \underline{x} + \sum_{y_i + \lambda d_i > 0} (y_i + \lambda d_i) \bar{x}$	Prop. 6.2.8

6.2.1 Unconstrained examples ($C = V$)

We here consider several interesting unconstrained proximal-like problems appearing in applications and explain how the associated OSGA subproblem (6.22) can be solved.

In recent years the interest of applying regularizations with weighted norms is increased by emerging many applications, see, for example, [72, 143]. Let d be a vector in $\mathbb{R}^n$ such that $d_i \neq 0$ for $i = 1, \dots, n$. Then we define the weight matrix $D := \text{diag}(d)$, which is a diagonal matrix with $D_{i,i} = d_i$ for $i = 1, \dots, n$. It is clear that D is an invertible matrix. The next result shows how to compute a solution of the problem (6.22) for special cases of ϕ arising frequently in applications.

Proposition 6.2.1. *Let $D := \text{diag}(d)$, where $d \in \mathbb{R}^n$ with $d_i \neq 0$, for $i = 1, \dots, n$. If $\phi(x) = \|Dx\|_1$, then the proximity operator (6.22) is given by*

$$\left(\text{prox}_{\lambda\phi}(y) \right)_i = \text{sign}(y_i)(|y_i| - \lambda|d_i|)_+, \quad (6.32)$$

for $i = 1, \dots, n$.

Proof. The optimality condition (2.12) implies that $u = \text{prox}_{\lambda\phi}(y)$ if and only if

$$0 \in u - y + \lambda \partial \|Du\|_1. \quad (6.33)$$

We consider two cases: (i) $\|D^{-1}y\|_\infty \leq \lambda$; (ii) $\|D^{-1}y\|_\infty > \lambda$.

Case (i). Let $\|D^{-1}y\|_\infty \leq \lambda$. Then we show $u = 0$ satisfies (6.33). If $u = 0$, Proposition 2.1.17 implies $\partial\phi(0) = \{g \in V^* \mid \|D^{-1}g\|_\infty \leq 1\}$. By substituting this into (6.33), we get that $u = 0$ is satisfied (6.33) if $y \in \{g \in V^* \mid \|D^{-1}g\|_\infty \leq \lambda\}$ leading to $\text{prox}_{\lambda\phi}(y) = 0$. Since the right hand side of (6.32) is also zero, (6.32) holds.

Case (ii). Let $\|D^{-1}y\|_\infty > \lambda$. Then Case (i) implies $u \neq 0$. Since $\|\cdot\|_* = \|\cdot\|_\infty$ and D is invertible, Proposition 2.1.17 implies that

$$\partial\phi(u) = \{g \in V^* \mid \|D^{-1}g\|_\infty = 1, \langle g, u \rangle = \|Du\|_1\}$$

leading to

$$\sum_{i=1}^n (g_i u_i - |d_i| |u_i|) = 0.$$

By induction on nonzero elements of u , we get $g_i u_i = |d_i| |u_i|$ for $i = 1, \dots, n$. This implies $g_i = |d_i| \text{sign}(u_i)$. The optimality condition (2.12) implies

$$0 \in u_i - y_i + \lambda |d_i| \partial |u_i|$$

for $i = 1, \dots, n$. If $u_i > 0$, then $u_i = -\lambda|d_i| + y_i > 0$. Hence if $\lambda|d_i| < y_i$, we set $u_i = -\lambda|d_i| + y_i$. If $u_i < 0$, then $u_i = \lambda|d_i| + y_i < 0$. Hence if $\lambda|d_i| > y_i$, we set $u_i = \lambda|d_i| + y_i$. Otherwise, we have $u_i = 0$. Therefore, we obtain

$$\left(\text{prox}_{\lambda\phi}(y)\right)_i = \begin{cases} y_i - \lambda|d_i| & \text{if } y_i > \lambda|d_i|, \\ y_i + \lambda|d_i| & \text{if } y_i < -\lambda|d_i|, \\ 0 & \text{otherwise,} \end{cases} \quad (6.34)$$

giving the result. $\square$

Proposition 6.2.2. *Let $D := \text{diag}(d)$, where $d \in \mathbb{R}^n$ and $d_i \neq 0$, for $i = 1, \dots, n$. If $\phi(x) = \|Dx\|_2$, then the proximity operator (6.22) is given by $\text{prox}_{\lambda\phi}(y) = 0$ if $\|D^{-1}y\|_2 \leq \lambda$ and otherwise*

$$\left(\text{prox}_{\lambda\phi}(y)\right)_i = \frac{\tau y_i}{\tau + \lambda d_i^2},$$

for $i = 1, \dots, n$, where τ is given by solving the one-dimensional nonlinear equation

$$\sum_{i=1}^n \frac{d_i^2 y_i^2}{(\tau + \lambda d_i^2)^2} - 1 = 0,$$

which has a unique solution.

Proof. The optimality condition (2.12) shows that $u = \text{prox}_{\lambda\phi}(y)$ if and only if

$$0 \in u - y + \lambda \partial \|D^{-1}u\|_2. \quad (6.35)$$

We consider two cases: (i) $\|D^{-1}y\|_2 \leq \lambda$; (ii) $\|D^{-1}y\|_2 > \lambda$.

Case (i). Let $\|D^{-1}y\|_2 \leq \lambda$. Then we show $u = 0$ satisfies (6.35). If $u = 0$, Proposition 2.1.17 implies $\partial\phi(0) = \{g \in V^* \mid \|D^{-1}g\|_2 \leq 1\}$. By using this and (6.35), we get that $u = 0$ is satisfied (6.35) if $y \in \{g \in V^* \mid \|D^{-1}g\|_2 \leq \lambda\}$ leading to $\text{prox}_{\lambda\phi}(y) = 0$.

Case (ii). Let $\|D^{-1}y\|_2 > \lambda$. Then Case (i) implies $u \neq 0$. Proposition 2.1.17 implies $\partial\phi(u) = D^T Du / \|Du\|_2$, and the optimality conditions (2.12) yields

$$u - y + \lambda D^T \frac{Du}{\|Du\|_2} = 0.$$

By using this and setting $\tau = \|Du\|_2$, we get

$$\left(1 + \frac{\lambda d_i^2}{\tau}\right) u_i - y_i = 0,$$

leading to

$$u_i = \frac{\tau y_i}{\tau + \lambda d_i^2},$$

for $i = 1, \dots, n$. Substituting this into $\tau = \|Du\|_2$ implies

$$\sum_{i=1}^n \frac{d_i^2 y_i^2}{(\tau + \lambda d_i^2)^2} = 1.$$

We define the function $\psi :]0, +\infty[\rightarrow \mathbb{R}$ by

$$\psi(\tau) := \sum_{i=1}^n \frac{d_i^2 y_i^2}{(\tau + \lambda d_i^2)^2} - 1,$$

where it is clear that ψ is decreasing

$$\lim_{\tau \rightarrow 0} \psi(\tau) = \frac{1}{\lambda^2} \sum_{i=1}^n \frac{y_i^2}{d_i^2} - 1 = \frac{1}{\lambda^2} (\|D^{-1}y\|_2^2 - \lambda^2), \quad \lim_{\tau \rightarrow +\infty} \psi(\tau) = -1.$$

By $\|D^{-1}y\|_2 > \lambda$ and the mean value theorem, we get that there exists $\hat{\tau} \in]0, +\infty[$ such that $\psi(\hat{\tau}) = 0$, giving the results. $\square$

We here emphasize that if $D = I$ (I denotes the identity matrix) then the proximity operator for $\phi(\cdot) = \|\cdot\|_2$ is given by

$$\text{prox}_{\lambda\phi}(y) = (1 - \lambda/\|y\|_2)_+ y,$$

see, for example, [138]. If one solves the equation $\psi(\tau) = 0$ approximately, and an initial interval $[a, b]$ is available such that $\psi(a)\psi(b) < 0$, then a solution can be computed to ε -accuracy using the bisection scheme in $O(\log_2((b-a)/\varepsilon))$ iterations, see, for example, [128]. However, it is preferable to use a more sophisticated zero finder like the secant bisection scheme (Algorithm 5.2.6, [128]). If an interval $[a, b]$ with sign change is available one can also use MATLAB's `fzero` function combining the bisection scheme, the inverse quadratic interpolation, and the secant method.

Proposition 6.2.3. *Let $\phi(\cdot) = \|\cdot\|_\infty$. Then the proximity operator (6.22) is given by $\text{prox}_{\lambda\phi}(y) = 0$ if $\|y\|_1 \leq \lambda$ and otherwise*

$$(\text{prox}_{\lambda\phi}(y))_i = \begin{cases} 0 & \text{if } \|y\|_1 \leq \lambda, \\ \text{sign}(y_i)u_\infty & \text{if } \|y\|_1 > \lambda, i \in I, \\ y_i & \text{if } \|y\|_1 > \lambda, i \notin I, \end{cases} \quad (6.36)$$

for $i = 1, \dots, n$, where

$$u_\infty := \frac{1}{\widehat{k}} \left(\sum_{i \in I} |y_i| - \lambda \right) \quad (6.37)$$

with

$$I := \{l_1, \dots, l_{\widehat{k}}\} \quad (6.38)$$

in which $\widehat{k}$ is the smallest $k \in \{1, \dots, n-1\}$ such that

$$\frac{1}{\widehat{k}} \left(\sum_{i=1}^{\widehat{k}} v_i - \lambda \right) \geq v_{\widehat{k}+1} \quad (6.39)$$

where $v_i := |y_{l_i}|$ and $l_1, \dots, l_n$ is a permutation of $1, \dots, n$ such that $v_1 \geq v_2 \geq \dots \geq v_n$. If (6.39) is not satisfied for $k \in \{1, \dots, n-1\}$, then $\widehat{k} = n$.

Proof. The optimality condition (2.12) shows that $u = \text{prox}_{\lambda\phi}(y)$ if and only if

$$0 \in u - y + \lambda \partial \|u\|_\infty. \quad (6.40)$$

We consider two cases: (i) $\|y\|_1 \leq \lambda$; (ii) $\|y\|_1 > \lambda$.

Case (i). Let $\|y\|_1 \leq \lambda$. Then we show $u = 0$ satisfies (6.40). If $u = 0$, the subdifferential of ϕ derived in Example 2.1.18 is $\partial\phi(0) = \{g \in V^* \mid \|g\|_1 \leq 1\}$. By substituting this into (6.40), we get that $u = 0$ is satisfied (6.40) if $y \in \{g \in V^* \mid \|g\|_1 \leq 1\}$ leading to $\text{prox}_{\lambda\phi}(y) = 0$. Since the right hand side of (6.32) is also zero, (6.32) holds.

Case (ii). Let $\|y\|_1 > \lambda$. From Case (i), we have $u \neq 0$. We show that, for $i = 1, \dots, n$,

$$u_i = \begin{cases} \text{sign}(y_i)u_\infty & \text{if } i \in I, \\ y_i & \text{otherwise,} \end{cases} \quad (6.41)$$

with I defined in (6.38), satisfies (6.40). Hence, using the subdifferential of ϕ derived in Example 2.1.18, there exist coefficients β_j , for $j \in I$, such that

$$u - y + \lambda \sum_{j \in I} \beta_j \text{sign}(u_j) e_j = 0, \quad (6.42)$$

where

$$\beta_j \geq 0 \quad j \in I, \quad \sum_{j \in I} \beta_j = 1. \quad (6.43)$$

Let u be the vector defined in (6.41). Let us define

$$\beta_j := \frac{|y_j| - u_\infty}{\lambda}, \quad (6.44)$$

for $j \in I = \{l_1, \dots, l_{\hat{k}}\}$ with u_∞ defined in (6.37). We show that the choice (6.44) satisfies (6.42) and (6.43). We first show $u_\infty > 0$. It follows from (6.37) and (6.39) if $\hat{k} < n$ and from $\|y\|_1 > \lambda$ if $\hat{k} = n$. By (6.41) and (6.42), we have

$$\begin{aligned} u_i - y_i + \lambda \beta_i \operatorname{sign}(u_i) &= \operatorname{sign}(y_i) u_\infty - y_i + (|y_i| - u_\infty) \operatorname{sign}(\operatorname{sign}(y_i) u_\infty) \\ &= \operatorname{sign}(y_i) u_\infty - y_i + (|y_i| - u_\infty) \operatorname{sign}(y_i) = 0, \end{aligned}$$

for $i \in I$. For $i \notin I$, we have $u_i - y_i = 0$. Hence (6.42) is satisfied componentwise. It remains to show that (6.43) holds. From (6.39), we have that $|y_i| \geq u_\infty$, for $i \in I$. This and (6.44) imply $\beta_i \geq 0$ for $i \in I$. From (6.37), we obtain

$$\sum_{i=1}^{\hat{k}} \beta_i = \frac{1}{\lambda} \sum_{i=1}^{\hat{k}} |y_i| - \frac{\hat{k}}{\lambda} u_\infty = \frac{1}{\lambda} \sum_{i=1}^{\hat{k}} |y_i| - \frac{1}{\lambda} \left(\sum_{i=1}^{\hat{k}} |y_i| - \lambda \right) = 1,$$

giving the results. $\square$

Grouped variables typically appear in high-dimensional statistical learning problems. For example, in data mining applications, categorical features are encoded by a set of dummy variables forming a group. Another interesting example is learning sparse additive models in statistical inference, where each component function can be represented using basis expansions and thus can be treated as a group. For such problems, see [111] and references therein, it is more natural to select groups of variables instead of individual ones when a sparse model is preferred.

In the following two results we show how the proximity operator $\operatorname{prox}_{\lambda\phi}(\cdot)$ can be computed for the mixed-norms $\phi(\cdot) = \|\cdot\|_{1,2}$ and $\phi(\cdot) = \|\cdot\|_{1,\infty}$, which are especially important in the context of sparse optimization and sparse recovery with grouped variables.

Proposition 6.2.4. *Let $\phi(\cdot) = \|\cdot\|_{1,2}$. Then the proximity operator (6.22) is given by*

$$(\operatorname{prox}_{\lambda\phi}(y))_{g_i} = \left(1 - \frac{\lambda}{\|y_{g_i}\|_2}\right)_+ y_{g_i}. \quad (6.45)$$

for $i = 1, \dots, m$.

Proof. Since $u = (u_{g_1}, \dots, u_{g_m}) \in \mathbb{R}^{n_1} \times \dots \times \mathbb{R}^{n_m}$ and ϕ is separable with respect to the grouped variables, we fix the index $i \in \{1, \dots, m\}$. The optimality condition (2.12) shows that $u_{g_i} = \operatorname{prox}_{\lambda\phi}(y_{g_i})$ if and only if

$$0 \in u_{g_i} - y_{g_i} + \lambda \partial \|u_{g_i}\|_2, \quad (6.46)$$

for $i = 1, \dots, m$. We now consider two cases: (i) $\|y_{g_i}\|_2 \leq \lambda$; (ii) $\|y_{g_i}\|_2 > \lambda$.

Case (i). Let $\|y_{g_i}\|_2 \leq \lambda$. Then we show $u_{g_i} = 0$ satisfies (6.46). If $u_{g_i} = 0$, Proposition 2.1.17 implies $\partial\phi(0_{g_i}) = \{g \in \mathbb{R}^{n_i} \mid \|g_{g_i}\|_2 \leq 1\}$. By substituting this into (6.46), we get that $u_{g_i} = 0$ is satisfied in (6.46) if $y_{g_i} \in \{g \in \mathbb{R}^{n_i} \mid \|g_{g_i}\|_2 \leq \lambda\}$ leading to $\text{prox}_{\lambda\phi}(y_{g_i}) = 0_{g_i}$. Since the right hand side of (6.45) is also zero, (6.45) holds.

Case (ii). Let $\|y_{g_i}\|_2 > \lambda$. Then Case (i) implies that $u_{g_i} \neq 0$. From Proposition 2.1.17, we obtain

$$\partial\phi(u_{g_i}) = \left\{ \frac{u_{g_i}}{\|u_{g_i}\|_2} \right\}, \quad (6.47)$$

where $i = 1, \dots, m$ and $\|y_{g_i}\|_2 > \lambda$. Then (6.46) and (6.47) imply

$$u_{g_i} - y_{g_i} + \lambda \frac{u_{g_i}}{\|u_{g_i}\|_2} = 0,$$

leading to

$$\left(1 + \frac{\lambda}{\|u_{g_i}\|_2}\right) u_{g_i} = y_{g_i}$$

implying $u_{g_i} = \mu_i y_{g_i}$. By substituting this into the previous identity and solving with respect to μ_i , we get

$$u_{g_i} = \left(1 - \frac{\lambda}{\|y_{g_i}\|_2}\right)_+ y_{g_i},$$

implying the result is valid. $\square$

Proposition 6.2.5. Let $\phi(\cdot) = \|\cdot\|_{1,\infty}$. Then $\text{prox}_{\lambda\phi}(y_{g_i}) = 0_{g_i}$ if $\|y_{g_i}\|_1 \leq \lambda$, for $i = 1, \dots, m$ and otherwise

$$(\text{prox}_{\lambda\phi}(y_{g_i}))_{g_i}^j = \begin{cases} 0 & \text{if } \|y_{g_i}\|_1 \leq \lambda, \\ \text{sign}(y_{g_i}^j) u_\infty^i & \text{if } \|y_{g_i}\|_1 > \lambda, j \in I_{g_i}, \\ y_{g_i}^j & \text{if } \|y_{g_i}\|_1 > \lambda, j \notin I_{g_i}, \end{cases} \quad (6.48)$$

for $i = 1, \dots, m$, where

$$u_\infty^i := \frac{1}{\widehat{k}_i} \left(\sum_{j \in I_{g_i}} |y_{g_i}^j| - \lambda \right) \quad (6.49)$$

with

$$I_{g_i} := \{l_{g_i}^1, \dots, l_{g_i}^{\widehat{k}_i}\} \quad (6.50)$$

in which $\widehat{k}_i$ is the smallest $k \in \{1, \dots, n_i - 1\}$ such that

$$\frac{1}{\widehat{k}_i} \left(\sum_{j=1}^{\widehat{k}_i} v_{g_i}^j - \lambda \right) \geq v_{g_i}^{\widehat{k}_i+1} \quad (6.51)$$

where $v_{g_i}^j := |y_{l_{g_i}^j}|$ and $l_{g_i}^1, \dots, l_{g_i}^{n_i}$ is a permutation of $1, \dots, n_i$ such that $v_{g_i}^1 \geq v_{g_i}^2 \geq \dots \geq v_{g_i}^{n_i}$. If (6.39) is not satisfied for $k \in \{1, \dots, n_i - 1\}$, then $\widehat{k}_i = n_i$, for $i = 1, \dots, m$.

Proof. Since $u = (u_{g_1}, \dots, u_{g_m}) \in \mathbb{R}^{n_1} \times \dots \times \mathbb{R}^{n_m}$ and ϕ is separable with respect to the grouped variables, we fix the index $i \in \{1, \dots, m\}$. The optimality condition (2.12) shows that $u_{g_i} = \text{prox}_{\lambda\phi}(y_{g_i})$ if and only if

$$0 \in u_{g_i} - y_{g_i} + \lambda \partial \|u_{g_i}\|_\infty. \quad (6.52)$$

We now consider two cases: (i) $\|y_{g_i}\|_1 \leq \lambda$; (ii) $\|y_{g_i}\|_1 > \lambda$.

Case (i). Let $\|y_{g_i}\|_1 \leq \lambda$. Then we show $u_{g_i} = 0$ satisfies (6.52). If $u_{g_i} = 0$, the subdifferential of ϕ derived in Example 2.1.18 is $\partial\phi(0_{g_i}) = \{g \in \mathbb{R}^{n_i} \mid \|g\|_1 \leq 1\}$. By substituting this into (6.52), we get that $u_{g_i} = 0$ is satisfied (6.52) if $y_{g_i} \in \{g \in \mathbb{R}^{n_i} \mid \|g\|_1 \leq 1\}$ leading to $\text{prox}_{\lambda\phi}(y_{g_i}) = 0_{g_i}$.

Case (ii). Let $\|y_{g_i}\|_1 > \lambda$. From Case (i), we have $u_{g_i} \neq 0$. We show that

$$u_{g_i}^j = \begin{cases} \text{sign}(y_{g_i}^j) u_\infty^i & \text{if } i \in I_{g_i}, \\ y_{g_i}^j & \text{otherwise,} \end{cases} \quad (6.53)$$

with I_{g_i} defined in (6.50), satisfies (6.52). Hence, using the subdifferential of ϕ derived in Example 2.1.18, there exist coefficients $\beta_{g_i}^j$, for $j \in I_{g_i}$, such that

$$u_{g_i} - y_{g_i} + \lambda \sum_{j \in I} \beta_{g_i}^j \text{sign}(u_{g_i}^j) e_j = 0, \quad (6.54)$$

where

$$\beta_{g_i}^j \geq 0 \quad j \in I_{g_i}, \quad \sum_{j \in I_{g_i}} \beta_{g_i}^j = 1. \quad (6.55)$$

Let u_{g_i} be the vector defined in (6.53). Let us define

$$\beta_{g_i}^j := \frac{|y_{g_i}^j| - u_\infty^i}{\lambda}, \quad (6.56)$$

for $j \in I_{g_i} = \{l_1^i, \dots, l_{\widehat{k}_i}^i\}$ with u_∞^i defined in (6.49). We show that the choice (6.56) satisfies (6.54). We first show $u_\infty^i > 0$. It follows from (6.49) and (6.51) if $\widehat{k}_i < n$ and from $\|y_{g_i}\|_1 > \lambda$ if $\widehat{k}_i = n$. By (6.53) and (6.54), we have

$$\begin{aligned} u_{g_i}^j - y_{g_i}^j + \lambda \beta_{g_i}^j \operatorname{sign}(u_{g_i}^j) &= \operatorname{sign}(y_{g_i}^j) u_\infty^i - y_{g_i}^j + (|y_{g_i}^j| - u_\infty^i) \operatorname{sign}(\operatorname{sign}(y_{g_i}^j) u_\infty^i) \\ &= \operatorname{sign}(y_{g_i}^j) u_\infty^i - y_{g_i}^j + (|y_{g_i}^j| - u_\infty^i) \operatorname{sign}(y_{g_i}^j) = 0, \end{aligned}$$

for $j \in I_{g_i}$. For $j \notin I_{g_i}^j$, we have $u_{g_i}^j - y_{g_i}^j = 0$. Hence (6.54) is satisfied componentwise. It remains to show that (6.55) holds. From (6.51), we have that $|y_{g_i}^j| \geq u_\infty^i$, for $j \in I_{g_i}$. This and (6.56) imply $\beta_{g_i}^j \geq 0$ for $j \in I_{g_i}$. From (6.49), we obtain

$$\sum_{j=1}^{\widehat{k}_i} \beta_{g_i}^j = \frac{1}{\lambda} \sum_{j=1}^{\widehat{k}_i} |y_{g_i}^j| - \frac{\widehat{k}_i}{\lambda} u_\infty^i = \frac{1}{\lambda} \sum_{j=1}^{\widehat{k}_i} |y_{g_i}^j| - \frac{1}{\lambda} \left(\sum_{j=1}^{\widehat{k}_i} |y_{g_i}^j| - \lambda \right) = 1,$$

giving the results. $\square$

6.2.2 Constrained examples ($C \neq V$)

In this section we consider the subproblem (6.22) and show how it can be solved for some ϕ and C . More precisely, we solve the minimization problem

$$\begin{aligned} \min \quad & \frac{1}{2} \|x - y\|_2^2 + \lambda \varphi(x) \\ \text{s.t.} \quad & x \in C, \end{aligned}$$

where $\varphi(x)$ is a simple convex function and C is a simple domain. We consider a few examples of this form.

Proposition 6.2.6. *Let $\varphi(x) = \|Dx\|_1$ and $C = [\underline{x}, \bar{x}]$. Then the global minimizer of the subproblem (6.22) is given by*

$$\left(\operatorname{prox}_{\lambda \phi}^C(y) \right)_i = \begin{cases} \underline{x}_i & \text{if } \omega(y, \lambda) > 0, \underline{x}_i - y_i + \lambda |d_i| \operatorname{sign}(\underline{x}_i) \geq 0, \\ \bar{x}_i & \text{if } \omega(y, \lambda) > 0, \bar{x}_i - y_i + \lambda |d_i| \operatorname{sign}(\bar{x}_i) \leq 0, \\ y_i - \lambda |d_i| & \text{if } \omega(y, \lambda) > 0, y_i > \lambda |d_i|, \\ y_i + \lambda |d_i| & \text{if } \omega(y, \lambda) > 0, y_i < -\lambda |d_i|, \\ 0 & \text{otherwise.} \end{cases} \quad (6.57)$$

for $i = 1, \dots, n$, where

$$\omega(y, \lambda) := \sum_{y_i + \lambda |d_i| < 0} (y_i + \lambda |d_i|) \underline{x} + \sum_{y_i + \lambda |d_i| > 0} (y_i + \lambda |d_i|) \bar{x}. \quad (6.58)$$

Proof. The optimality condition (2.11) shows that $u = \text{prox}_{\lambda\phi}^C(y)$ if and only if

$$0 \in u - y + \lambda \partial \|Du\|_1 + N_C(u), \quad (6.59)$$

where $N_C(u)$ is the normal cone of C at u defined in (2.2). Let us define $\omega(y, \lambda)$ by (6.58). We now consider two cases: (i) $\omega(y, \lambda) \leq 0$; (ii) $\omega(y, \lambda) > 0$.

Case (i). Let $\omega(y, \lambda) \leq 0$. Then we show that $u = 0$ satisfies (6.59). If $u = 0$, Proposition 2.1.17 implies $\partial\phi(0) = \{g \in V^* \mid \|D^{-1}g\|_\infty \leq 1\}$. This and (6.59) lead to

$$y - \lambda \partial\phi(0) \in N_C(0), \quad (6.60)$$

where

$$N_C(0) = \{p \in V \mid \forall z \in [\underline{x}, \bar{x}], \langle p, z \rangle \leq 0\} = \left\{ p \in V \mid \sum_{p_i < 0} p_i \underline{x} + \sum_{p_i > 0} p_i \bar{x} \leq 0 \right\}.$$

This and (6.60) lead to

$$\begin{aligned} \max \quad & \sum_{p_i < 0} p_i \underline{x} + \sum_{p_i > 0} p_i \bar{x} \\ \text{s.t.} \quad & p \in \{y - g \mid \|D^{-1}g\|_\infty \leq \lambda, g \in V^*\}. \end{aligned}$$

It is clear that $p = y - \lambda |D\mathbf{1}|$ is the solution of this problem, where $\mathbf{1}$ is the vector of all ones. Then if $\omega(y, \lambda) \leq 0$, then $u = 0$ is satisfied (6.59) leading to $\text{prox}_{\lambda\phi}(y) = 0$.

Case (ii). Let $\omega(y, \lambda) > \lambda$. Then Case (i) implies $u \neq 0$. Proposition 2.1.17 yields

$$\partial\phi(u) = \{g \in V^* \mid \|D^{-1}g\|_\infty = 1, \langle g, u \rangle = \|Du\|_1\}$$

leading to

$$\sum_{i=1}^n (g_i u_i - |d_i u_i|) = 0.$$

By induction on nonzero elements of u , we get $g_i u_i = |d_i u_i|$, for $i = 1, \dots, n$. This implies that $g_i = |d_i| \text{sign}(u_i)$ if $u_i \neq 0$. This and the definition of $N_C(u)$ imply

$$u_i - y_i + \lambda (\partial \|Du\|_1)_i \begin{cases} \geq 0 & \text{if } u_i = \underline{x}_i, \\ \leq 0 & \text{if } u_i = \bar{x}_i, \\ = 0 & \text{if } \underline{x}_i < u_i < \bar{x}_i, \end{cases}$$

for $i = 1, \dots, n$, and equivalently for $u \neq 0$, we get

$$u_i - y_i + \lambda |d_i| \operatorname{sign}(u_i) \begin{cases} \geq 0 & \text{if } u_i = \underline{x}_i, \\ \leq 0 & \text{if } u_i = \bar{x}_i, \\ = 0 & \text{if } \underline{x}_i < u_i < \bar{x}_i, \end{cases} \quad (6.61)$$

for $i = 1, \dots, n$.

If $u_i = \underline{x}_i$, substituting $u_i = \underline{x}_i$ in (6.61) implies $\underline{x}_i - y_i + \lambda |d_i| \operatorname{sign}(\underline{x}_i) \geq 0$. If $u_i = \bar{x}_i$, substituting $u_i = \bar{x}_i$ in (6.61) implies $\bar{x}_i - y_i + \lambda |d_i| \operatorname{sign}(\bar{x}_i) \leq 0$. If $\underline{x}_i < u_i < \bar{x}_i$, there are three possibilities: (a) $u_i > 0$; (b) $u_i < 0$; (c) $u_i = 0$. In Case (a), the fact $\operatorname{sign}(u_i) = 1$ and (6.61) imply $u_i = y_i - \lambda |d_i| > 0$. In Case (b), the condition $\operatorname{sign}(u_i) = -1$ and (6.61) imply $u_i = y_i + \lambda |d_i| < 0$. In Case (c), we get $u_i = 0$. This completes the proof. $\square$

Proposition 6.2.7. Let $\varphi(x) = \frac{1}{2} \|x\|_2^2$ and $C = [\underline{x}, \bar{x}]$. Then the global minimizer of the subproblem (6.22) is given by

$$\left(\operatorname{prox}_{\lambda \varphi}^C(y) \right)_i = \begin{cases} \underline{x}_i & \text{if } (1 + \lambda) \underline{x}_i \geq y_i, \\ \bar{x}_i & \text{if } (1 + \lambda) \bar{x}_i \leq y_i, \\ y_i / (1 + \lambda) & \text{if } \underline{x}_i < y_i / (1 + \lambda) < \bar{x}_i, \end{cases} \quad (6.62)$$

for $i = 1, \dots, n$.

Proof. The function $\varphi(x) = \frac{1}{2} \|x\|_2^2$ is differentiable, i.e.,

$$\partial \varphi(x) = \{x\}.$$

This and the definition of $N_C(u)$ imply

$$u_i - y_i + \lambda u_i \begin{cases} \geq 0 & \text{if } u_i = \underline{x}_i, \\ \leq 0 & \text{if } u_i = \bar{x}_i, \\ = 0 & \text{if } \underline{x}_i < u_i < \bar{x}_i, \end{cases} \quad (6.63)$$

for $i = 1, \dots, n$. If $u_i = \underline{x}_i$, substituting $u_i = \underline{x}_i$ in (6.63) implies $(1 + \lambda) \underline{x}_i \geq y_i$. If $u_i = \bar{x}_i$, substituting $u_i = \bar{x}_i$ in (6.63) implies $(1 + \lambda) \bar{x}_i \leq y_i$. If $\underline{x}_i < u_i < \bar{x}_i$, then $u_i = y_i / (1 + \lambda)$. This gives the result. $\square$

Proposition 6.2.8. *Let $\varphi(x) = \frac{1}{2}\lambda_1\|x\|_2^2 + \lambda_2\|Dx\|_1$ and $C = [\underline{x}, \bar{x}]$. Then the global minimizer of the subproblem (6.22) is determined by*

$$\left(\text{prox}_{\lambda\varphi}^C(y)\right)_i = \begin{cases} \underline{x}_i & \text{if } \omega(y, \lambda) > 0, \\ & (1 + \lambda_1)\underline{x}_i - y_i + \lambda_2|d_i| \text{ sign}(\underline{x}_i) \geq 0, \\ \bar{x}_i & \text{if } \omega(y, \lambda) > 0, \\ & (1 + \lambda_1)\bar{x}_i - y_i + \lambda_2|d_i| \text{ sign}(\bar{x}_i) \leq 0, \\ 1/(1 + \lambda_1)(y_i - \lambda_2|d_i|) & \text{if } \omega(y, \lambda) > 0, y_i > \lambda_2|d_i|, \\ 1/(1 + \lambda_1)(y_i + \lambda_2|d_i|) & \text{if } \omega(y, \lambda) > 0, y_i < -\lambda_2|d_i|, \\ 0 & \text{otherwise,} \end{cases} \quad (6.64)$$

for $i = 1, \dots, n$, where

$$\omega(y, \lambda_2) := \sum_{y_i + \lambda_2|d_i| < 0} (y_i + \lambda_2|d_i|)\underline{x} + \sum_{y_i + \lambda_2|d_i| > 0} (y_i + \lambda_2|d_i|)\bar{x}. \quad (6.65)$$

Proof. Since V is finite-dimensional and $\text{ri}(\text{dom}(\frac{1}{2}\lambda_1\|\cdot\|_2^2)) \cap \text{ri}(\text{dom}\lambda_2\|D\cdot\|_1) \neq \emptyset$, we get

$$\partial\left(\frac{1}{2}\lambda_1\|x\|_2^2 + \lambda_2\|Dx\|_1\right) = \lambda_1\partial\left(\frac{1}{2}\|x\|_2^2\right) + \lambda_2\partial(\|Dx\|_1). \quad (6.66)$$

The optimality condition (2.11) shows that $u = \text{prox}_{\lambda\varphi}^C(y)$ if and only if

$$0 \in u - y + \lambda_1 u + \lambda_2 \partial\|Du\|_1 + N_C(u), \quad (6.67)$$

where $N_C(u)$ is the normal cone of C at u defined in (2.2). Let us consider $\omega(y, \lambda)$ defined by (6.58). We now consider two cases: (i) $\omega(y, \lambda) \leq 0$; (ii) $\omega(y, \lambda) > 0$.

Case (i). Let $\omega(y, \lambda) \leq 0$. Then we show that $u = 0$ satisfies (6.67). If $u = 0$, Proposition 2.1.17 implies $\partial\phi(0) = \{g \in V^* \mid \|D^{-1}g\|_\infty \leq 1\}$. This and (6.67) lead to

$$y - \lambda_2 \partial\phi(0) \in N_C(0), \quad (6.68)$$

where

$$N_C(0) = \{p \in V \mid \forall z \in [\underline{x}, \bar{x}], \langle p, z \rangle \leq 0\} = \left\{p \in V \mid \sum_{p_i < 0} p_i \underline{x} + \sum_{p_i > 0} p_i \bar{x} \leq 0\right\}.$$

This and (6.68) lead to

$$\begin{aligned} \max \quad & \sum_{p_i < 0} p_i \underline{x} + \sum_{p_i > 0} p_i \bar{x} \\ \text{s.t.} \quad & p \in \{y - g \mid \|D^{-1}g\|_\infty \leq \lambda_2, g \in V^*\}. \end{aligned}$$

It is clear that $p = y - \lambda_2 |D\mathbf{1}|$ is the solution of this problem, where $\mathbf{1}$ is the vector of all ones. Then if $\omega(y, \lambda_2) \leq 0$, then $u = 0$ is satisfied (6.67) leading to $\text{prox}_{\lambda\phi}(y) = 0$.

Case (ii). Let $\omega(y, \lambda_2) > 0$. Then Case (i) implies $u \neq 0$. From (6.66) and the definition of $N_C(u)$, we obtain

$$u_i - y_i + \lambda_1 u_i + \lambda_2 \partial |d_i u_i| \begin{cases} \geq 0 & \text{if } u_i = \underline{x}_i, \\ \leq 0 & \text{if } u_i = \bar{x}_i, \\ = 0 & \text{if } \underline{x}_i < u_i < \bar{x}_i, \end{cases}$$

for $i = 1, \dots, n$. This leads to

$$(1 + \lambda_1)u_i - y_i + \lambda_2 |d_i| \text{sign}(u_i) \begin{cases} \geq 0 & \text{if } u_i = \underline{x}_i, \\ \leq 0 & \text{if } u_i = \bar{x}_i, \\ = 0 & \text{if } \underline{x}_i < u_i < \bar{x}_i, \end{cases} \quad (6.69)$$

for $i = 1, \dots, n$.

If $u_i = \underline{x}_i$, substituting $u_i = \underline{x}_i$ in (6.61) implies $(1 + \lambda_1)\underline{x}_i - y_i + \lambda_2 |d_i| \text{sign}(\underline{x}_i) \geq 0$. If $u_i = \bar{x}_i$, substituting $u_i = \bar{x}_i$ in (6.61) implies $(1 + \lambda_1)\bar{x}_i - y_i + \lambda_2 |d_i| \text{sign}(\bar{x}_i) \leq 0$. If $\bar{x}_i < u_i < \bar{x}_i$, there are three possibilities: (i) $u_i > 0$; (ii) $u_i < 0$; (iii) $u_i = 0$. In Case (i), the fact $\text{sign}(u_i) = 1$ and (6.61) imply $u_i = 1/(1 + \lambda_1)(y_i - \lambda_2 |d_i|) > 0$. In Case (ii), the condition $\text{sign}(u_i) = -1$ and (6.61) imply $u_i = 1/(1 + \lambda_1)(y_i + \lambda_2 |d_i|) < 0$. In Case (iii), we get $u_i = 0$ giving the result. $\square$

Let $x \geq 0$ be nonnegativity constraints. These constraints are important in many applications, especially if x describes physical quantities, see, for example, [81, 101, 102]. Since nonnegativity constraints can be regarded as especial case of bound-constrained domain, Propositions 6.2.6, 6.2.7, and 6.2.8 can be used to derive the results for nonnegativity constraints (see Table 6.1). We here emphasize that the proximal-like examples considered in this section can be used in proximal algorithms such as those reported in PARIKH & BOYD [138] and in Nesterov-type optimal methods as well (see Section 2.5).

Part III

**Numerical Results, Applications,
and Future Work**

Chapter 7

OSGA software package

OSGA is a MATLAB package designed to solve large-scale convex optimization problems with simple constraints using optimal subgradient algorithm OSGA discussed Chapter 3. It was developed for practical unconstrained, bound-constrained, simply convex constrained, and simple functional constrained convex optimization (see AHOOKHOSH [1] and AHOOKHOSH & NEUMAIER [9, 12, 10, 11, 13]). The algorithm requires first-order information and can solve problems involving high-dimensional data due to its low memory requirement. OSGA is available from the URL:

<http://homepage.univie.ac.at/masoud.ahookhosh/>

In order to use OSGA, simply unpack the compressed file and add the base directory to your MATLAB path. Once it is installed, some drivers are available and ready to run. You also can solve your own problem by OSGA, where it will be discussed in the rest of this section.

This version of OSGA solves minimization problems of the form (2.8) with the multi-term affine composite objective (4.2). We suppose that $\text{ri}(\text{dom } f) \neq \emptyset$. Under considered properties of f , the problem has a global optimizer denoted by $\hat{x}$ and $\hat{f}$ is its global optimum. On the one hand, practical problems are typically involving high-dimensional data, i.e., the computation of function values and subgradients is expensive. On the other hand, in many applications the most computational cost of computing function values and subgradients relates to applying forward and adjoint linear operators, originated from an existence of affine terms in (4.2). This suggests that we should apply linear operators and their adjoints as fewer as possible in the implementations. Considering the structure of (4.2), we see that affine terms $A_i x$, for $i = 1, 2, \dots, n_1$, and $W_j x$, for $j = 1, 2, \dots, n_2$, are appearing in both function values and subgradients. By setting $v_x^i := A_i x$, for $i = 1, 2, \dots, n_1$, and $w_x^j := W_j x$, for $j = 1, 2, \dots, n_2$ and using (4.3), we provide the

first-order oracle in the following:

Algorithm 10: NFO-FG (nonsmooth first-order oracle)

Input: A_i for $i = 1, \dots, n_1$, W_j for $j = 1, \dots, n_2$, x ;

Output: f_x ; g_x ;

1 **begin**

2 $v_x^i \leftarrow Ax$ for $i = 1, \dots, n_1$;

3 $w_x^j \leftarrow Wx$ for $j = 1, \dots, n_2$;

4 $f_x \leftarrow \sum_{i=1}^{n_1} f_i(v_x^i) + \sum_{j=1}^{n_2} \phi_j(w_x^j)$;

5 $g_x \leftarrow \sum_{i=1}^{n_1} A_i^* \partial f_i(v_x^i) + \sum_{j=1}^{n_2} W_j^* \partial \phi_j(w_x^j)$;

6 **end**

Each call of the oracle $\mathcal{O}(x) = (f_x, g_x)$ requires $n_1 + n_2$ calls of forward and adjoint linear operators. By using this scheme, one can avoid double applying of expensive linear operators in the computation of function values and subgradients. In cases that algorithms need only a function value or a subgradient, we consider two special cases of NFO-FG that just return a function value or a subgradient, which are denoted by NFO-F and NFO-G, respectively. We also emphasize that if total computational cost of the oracle is dominated by applying linear mappings, the complexity of an algorithm can be measured by counting the number of forward and adjoint linear operators used to achieve an ε -solution.

7.1 Inputs and outputs

OSGA has a simple interface with 5 input arguments and 1, 2 or 3 output arguments described in the following.

Input arguments. OSGA requires the following input arguments:

1. **func:** a function handle returns a function value and a subgradient as described in Algorithm 10;
2. **prox:** a function handle for a prox-function;
3. **subprob:** a function handle for OSGA's subproblem solver;
4. **x_0 :** an initial point;
5. **options:** a structure involving the parameters needed for OSGA.

The structure options can be specified by users. Otherwise, OSGA uses its default parameters¹. To see the default value of the parameters, users are invited to take a look at the function `OSGA_Init.m`.

Output arguments. The outputs of OSGA are as follows:

1. `x`: the best approximation of the optimizer obtained by OSGA;
2. `f`: the best approximation of the optimum obtained by OSGA;
3. `out`: the structure involving more output information.

OSGA offers various output options for the user using the structure `out` (see [2] for a full description).

7.2 Forward and adjoint operations

Considering the problem (2.8), OSGA requires that the cell array `A` containing matrices or linear operators is defined by

$$A = \{\{A_1, \dots, A_{n_1}, A_1^*, \dots, A_{n_1}^*\}, \{W_1, \dots, W_{n_2}, W_1^*, \dots, W_{n_2}^*\}\},$$

where $A_1, \dots, A_{n_1}$ and $A_1^*, \dots, A_{n_1}^*$ denote forward and adjoint linear operators (matrices) of smooth functions, respectively, and $W_1, \dots, W_{n_2}$ and $W_1^*, \dots, W_{n_2}^*$ stand for forward and adjoint linear operators (matrices) of nonsmooth functions, respectively. For example, for the LASSO problem (4.8), `A` is constructed by

$$A = \{\{A_1, A_1^*\}, \{\}\},$$

Let us consider the scaled elastic net problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|A_1 x - b\|_2^2 + \frac{\lambda_2}{2} \|A_2 x\|_2^2 + \lambda_1 \|W_1 x\|_1. \quad (7.1)$$

Then the cell array `A` is constructed by

$$A = \{\{A_1, A_2, A_1^*, A_2^*\}, \{W_1, W_1^*\}\}$$

because first and second function in (7.1) are smooth and the third one is nonsmooth.

In the package forward and adjoint linear operations are managed by the function `LOPMVM` as follows

```
Ax = feval(LOPMVM(A), x, 1);
Atx = feval(LOPMVM(A), x, 2);
```

where `Ax` stands for the forward operation and `Atx` stands for the adjoint operation.

¹To see more information regarding the parameters that can be specified in options, see the user guide [2]

7.3 Stopping criteria

The package offers 8 commonly used stopping criteria to terminate OSGA, which can be specified by setting the parameter `Stopping_Crit` equal to 1 to 8. In particular, each number means that the algorithm stops

1. if the maximum number of iterations is reached;
2. if the maximum number of function evaluations is reached;
3. if the maximum number of subgradient evaluations is reached;
4. if the maximum number of linear operators is reached;
5. if the maximum running time is reached;
6. if $\eta \leq \varepsilon$;
7. if $\|x_{x_b} - x_{old}\| / \max(1, \|x_{x_b}\|) \leq \varepsilon$;
8. if $f(x_b) \leq f_{target}$;

where x_b is the current best founded approximation of the optimizer and x_{old} is the former best founded approximate of the optimizer.

7.4 Building your own problem

To solve your own problem of the form (2.8), one only needs to write a routine to calculate function values and subgradients of the objective function. Based on construction of the cell array `A`, we require to divide the objective function to the following four parts:

1. those parts that are smooth and involve linear operators;
2. those parts that are nonsmooth and involve linear operators;
3. the part that is smooth and involves no linear operators;
4. the part that is nonsmooth and involves no linear operators.

Therefore, the output arguments of the routine are

1. `fx` : a number returning the objective function value at `x`;

2. $gx1$: a cell array returning the subgradients of those parts involving linear operators defined by

$$gx1 = \{\{g_1^1, \dots, g_{n_1}^1\}, \{g_1^2, \dots, g_{n_2}^2\}\},$$

where $g_1^1, \dots, g_{n_1}^1$ are gradients of smooth parts, and $g_1^2, \dots, g_{n_2}^2$ are subgradients of nonsmooth parts;

3. $gx2$: a cell array returning the subgradients of those parts involving no linear operators defined by

$$gx1 = \{g^3, g^4\},$$

where g^3 is the gradient of the smooth part, and g^4 is a subgradient of the nonsmooth part.

Finally, the subgradient of whole function is constructed using the m-file SubGradEval.m by setting

$$gx = \sum_{i=1}^{n_1} A_i^* g_i^1 + \sum_{i=1}^{n_2} W_i^* g_i^2 + g^3 + g^4.$$

7.5 How to call OSGA

To illustrate the suage of the software, let us show how to apply OSGA for solving lasso (4.8). We construct the data randomly, determine a stopping criterion, and apply OSGA to solve the problem.

```
A=rand(m,n);
b=rand(m,1);
x0=rand(n,1);
opt.b=b;
opt.lambda=1;
func=@(varargin)L22L1R(opt,varargin{:});
prox=@(varargin)Prox_Quad(varargin{:});
subprob=@(varargin)Subuncon(varargin{:});
options.A={{A,A'},{}};
options.MaxNumIter=100;
options.Stopping_Crit=1;
[x,f,out]=OSGA(func,prox,subprob,x0,options);
```

Here L22L1R is the routine returning a function value and a subgradient, ProxQuad is the quadratic prox function (4.13), and Subuncon is the subroutine for solving the subproblem (3.4). To see more information, one can check the driver L1_driver.m. One can also find several examples in the folder “Drivers” o the software package.

Chapter 8

Unconstrained convex optimization

In this chapter we first consider multi-term affine composite problems and then consider convex problems involving costly linear operators and report some numerical results and comparison with such problems.

8.1 Multi-term affine composite problems

In this section we first consider several imaging problems (denoising, inpainting, and deblurring) and some sparse optimization problems and solve them by OSGA and several state-of-the-art solvers

8.1.1 Image restoration

Image reconstruction, also called image restoration, is one of the classical linear inverse problems dating back to the 1960s, see, for example, [15, 34]. The goal is to reconstruct images from some observations. Let $y \in U$ be a noisy indirect observation of an image $x \in V$ with an unknown noise $\delta \in U$, and let $A : V \rightarrow U$ be a linear operator. To recover the unknown vector x , we use the linear inverse model (4.5). According to the ill-conditioned or singular nature of the problem, some regularizations are needed. It is typical to use the models (4.7), (4.8), or (4.2) to derive a solution for the corresponding inverse problem, see also [101, 102, 127]. The strategy employed for solving such penalized optimization problems depends on features of objectives and regularization terms such as differentiability or non-differentiability and convexity or nonconvexity. The quality of the reconstructed image is highly dependent on these features. A typical model for image reconstruction involves a smooth objective penalized by some smooth or nonsmooth regularization penalties, which are selected based on expected features of the recovered image.

Suppose that a known image $x \in V$ is represented by $x = W\tau$ in some basis domain, e.g., wavelet, curvelet, or time-frequency domains, where the operator $W : U \rightarrow V$ can be orthogonal or any dictionary. There exist two main strategies for restoring the image x from an indirect observation y , namely **synthesis** and **analysis**. To recover an image, the synthesis approach reconstructs the image by solving the minimization problem

$$\min_{\tau \in U} \frac{1}{2} \|AW\tau - y\|_2^2 + \lambda \varphi(\tau) \quad (8.1)$$

or

$$\min_{\tau \in U} \|AW\tau - y\|_1 + \lambda \varphi(\tau), \quad (8.2)$$

where φ is a convex regularizer. If τ^* is an optimizer of (8.1), then an approximation of the clean image x will be reconstructed using $\hat{x} = W\tau^*$. Alternatively, the analysis strategy recovers the image by solving

$$\min_{x \in V} \frac{1}{2} \|Ax - y\|_2^2 + \lambda \varphi(x) \quad (8.3)$$

or

$$\min_{x \in V} \|Ax - y\|_1 + \lambda \varphi(x). \quad (8.4)$$

The problems (8.1) and (8.3) are strongly related. More specifically, these two approaches are equivalent in some assumptions, while variant in others, for more details see [79]. Although various regularizers like the ℓ_p -norm are popular in image restoration, it is arguable that the best known and frequently employed regularizer in the analysis approach is **total variation** (TV). In the remainder of this section, we report our numerical experiments based on the analysis strategy with TV.

The pioneering work on total variation (TV) for image denoising and restoration was proposed by RUDIN, OSHER, and FATEMI in [135]. Total variation regularizations are able to restore discontinuities of images and recover the edges, and so they have been widely used in applications. TV is originally defined in an infinite-dimensional Hilbert space, however, for the digital image processing a discrete version in a finite-dimensional space like $\mathbb{R}^n$ of pixel values on a two-dimensional lattice has been used. Two standard choices of discrete TV-based regularizers, namely **isotropic total variation** (ITV) and **anisotropic total variation** (ATV), are popular in signal and image processing, defined by

$$\begin{aligned} \|x\|_{ITV} &:= \sum_i^{m-1} \sum_j^{n-1} \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2} \\ &+ \sum_i^{m-1} |x_{i+1,n} - x_{i,n}| + \sum_i^{n-1} |x_{m,j+1} - x_{m,j}|, \end{aligned} \quad (8.5)$$

and

$$\begin{aligned} \|x\|_{ATV} &:= \sum_i^{m-1} \sum_j^{n-1} \{|x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}|\} \\ &+ \sum_i^{m-1} |x_{i+1,n} - x_{i,n}| + \sum_i^{n-1} |x_{m,j+1} - x_{m,j}|, \end{aligned} \quad (8.6)$$

for $x \in \mathbb{R}^{m \times n}$, respectively. In practice, the regularizer ITV is much more popular than ATV. The model (8.3) with $\phi = \|\cdot\|_{ITV}$ is denoted by L22ITVR and the model (8.4) with $\phi = \|\cdot\|_{ITV}$ is denoted by L1ITVR. In the remainder of this section we consider some prominent classes of imaging problems such as denoising, inpainting, and deblurring, and then we conduct some numerical experiments on a set of images.

8.1.1.1 Denoising with total variation

Image denoising is one of the most fundamental image restoration problems aiming to remove noise from images, in which the noise comes from some resources such as sensor imperfection, poor illumination, and communication errors. While this task itself is a prominent imaging problem, it is also emerged in other imaging problems such as deblurring with the proximal algorithms. The main objective is to denoise images while the edges are preserved, where, among all of the regularizers, TV can preserve discontinuities (edges).

In particular, if A is the identity operator in (8.3), it is called a *denoising* problem and is very popular with total variation regularizers, i.e.,

$$x_{y,\lambda} = \operatorname{argmin}_{x \in \mathbb{R}^n} \frac{1}{2} \|x - y\|_2^2 + \lambda \phi(x), \quad (8.7)$$

where $\phi = \|\cdot\|_{ITV}$ or $\phi = \|\cdot\|_{ATV}$. It is clear that both $\|\cdot\|_{ITV}$ and $\|\cdot\|_{ATV}$ are nonsmooth semi-norms and the proximal problem (8.7) cannot be solved explicitly for a given y . Rather than solving the problem (8.7) explicitly, it has been approximately solved by iterative schemes, cf. [53]. Note that $\|\cdot\|_{ITV}$ and $\|\cdot\|_{ATV}$ are convex, and their subdifferentials are available, so OSGA can handle (8.7) directly.

We consider the 1024×1024 pirate image and its noisy version y , where y is constructed by

$$\text{SNR}=15, \quad y = \text{awgn}(x_0, \text{SNR}, 'measured').$$

We restore the image using the model L22ITVR, where $x_{y,\lambda}$ is the denoised image associated to a regularization parameter λ . We run IST, TwIST [37], FISTA [28, 27], and OSGA to recover the image. For IST, TwIST, and FISTA, we use codes provided by the authors with only adding more stopping criteria to their codes. We set $\lambda = 5 \times 10^{-2}$ for all algorithms. As IST, TwIST, and FISTA need to solve their

subproblems iteratively, we consider three different cases by limiting the number of Chambolle's iterations to $chit = 5, 10$, and 20 . Let us consider the 1024×1024 Pirate image, where the noisy image is generated by the MATLAB internal function `awgn` with $SNR = 15$ (see Figure 8.2). The algorithms are stopped after 50 iterations, and the results are summarized in Table 8.1 and Figure 8.1.

Table 8.1: A comparison among IST, TwIST, FISTA, and OSGA for denoising the 1024×1024 Pirate image with L22ITVR, where IST, TwIST, and FISTA use several numbers of Chambolle's iterations.

Efficiency measure	Chambolle iterations	IST	TwIST	FISTA	OSGA
Function value	5	3791.56	3791.56	3791.56	3651.67
PSNR		30.05	30.05	30.05	30.09
Time		20.32	38.55	19.67	8.52
Function value	10	3750.28	3729.10	3750.28	3651.67
PSNR		30.16	30.06	30.16	30.09
Time		37.13	100.13	35.60	4.43
Function value	20	3741.17	3735.04	3741.17	3651.67
PSNR		30.08	30.00	30.08	30.09
Time		68.54	212.98	65.19	9.38

In Figure 8.1, we compare the results of implementation in several measures. Let $\hat{x}$ and $\hat{f}$ be an optimizer and the corresponding optimum of the problems (8.3) or (8.4), and let x and f be the current iteration point and function value of the algorithms respectively. The performances are measured by the relative error of iteration points

$$\delta_1 := \frac{\|x - \hat{x}\|_2}{\|\hat{x}\|_2} \quad \text{and} \quad (8.8)$$

and the relative error of function values

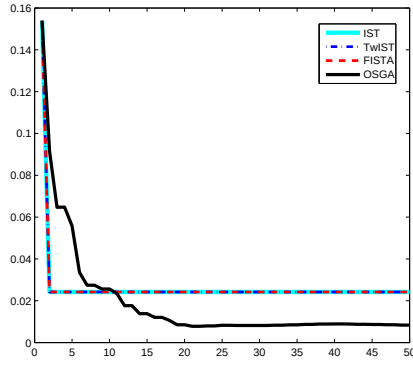
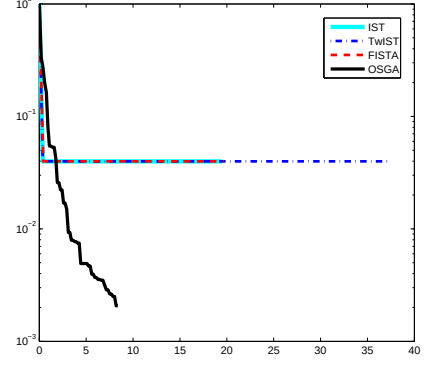
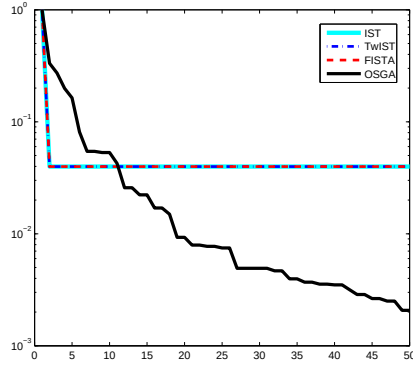
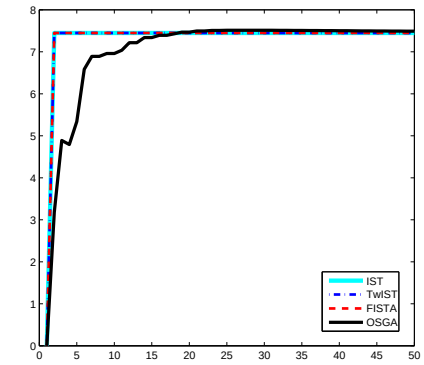
$$\delta_2 := \frac{f - \hat{f}}{f_0 - \hat{f}}, \quad (8.9)$$

and also so-called signal-to-noise improvement (ISNR) defined by

$$\text{ISNR} := 20 \log_{10} \left(\frac{\|y - x_0\|_F}{\|x - x_0\|_F} \right) \quad \text{and} \quad (8.10)$$

and peak signal-to-noise ratio (PSNR) defined by

$$\text{PSNR} := 20 \log_{10} \left(\frac{\sqrt{mn}}{\|x - x_0\|_F} \right), \quad (8.11)$$

(a) δ_1 versus iterations(b) δ_2 versus time(c) δ_2 versus iterations

(d) ISNR versus iterations

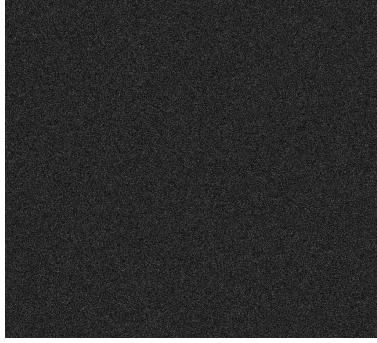
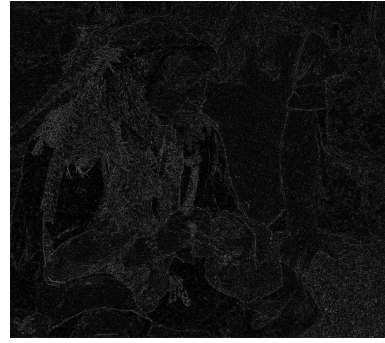
(e) Residual $|y - x_0|$ (f) Residual $|\hat{x} - x_0|$ for OSGA

Figure 8.1: Denoising by IST, TwiST, FISTA, and OSGA for the 1024×1024 Pirate image with $\lambda = 5 \times 10^{-2}$, where they were stopped after 50 iterations. Subfigure (a) displays the relative error δ_1 of points versus iterations, Subfigure (b) shows the relative error δ_2 of function values versus time, Subfigure (c) displays the relative error δ_2 of function values versus iterations, Subfigure (d) depicts ISNR versus iterations, Subfigure (e) displays the residual $|y - x_0|$, and Subfigure (f) shows the residual $|\hat{x} - x_0|$.



(a) Clean image



(b) Noisy image

(c) IST ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 20.32$)(d) TwIST ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 38.55$)(e) FISTA ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 19.67$)(f) OSGA ($f = 3651.67$, $\text{PSNR} = 30.09$, $\text{time} = 5.52$)

Figure 8.2: A comparison among IST, TwIST, FISTA, and OSGA for denoising the 1024×1024 Pirate image, where they were stopped after 50 iterations. We used the regularization parameter $\lambda = 5 \times 10^{-2}$. Subfigures (a) and (b) show the clean and noisy images, respectively. Subfigures (c), (d), (e), and (f) illustrate the recovered images.

where the pixel values are in $[0, 1]$. Generally, ISNR and PSNR measure the quality of the restored image x relative to the blurred or noisy observation y . From Figure 8.1, it is observed that OSGA outperforms IST, TwIST, and FISTA with respect to all considered measures. Figure 8.1 shows that the denoised image looks good for all considered algorithms in which the best function value and the second best PSNR are attained by OSGA, where much less running time is needed. However, it can be seen that by increasing the number of Chambolle's iterations, the running time of IST, TwIST, and FISTA is dramatically increased, but better function values or PSNR (compared with OSGA) are not attained.

8.1.1.2 Inpainting with total variation

Image inpainting is a set of techniques to fill lost or damaged parts of an image such that these modifications are undetectable for viewers who are not aware of the original image. The basic idea is originated from reconstructing of degraded or damaged paintings conventionally carried out by expert artists. This is with no doubt very time consuming and risky. Hence the goal is to produce an image in which the inpainted region is unified into the observation and looks normal for ordinary observers. The pioneering work on digital image inpainting was proposed by BERTALMI et al. in [33] based on nonlinear PDE. Since then lots of applications of inpainting, e.g., image interpolation, photo restoration, zooming, super-resolution images, image compression, and image transmission, have been studied.

In 2006, CHAN in [57] proposed an efficient method to recover piecewise constant or smooth images by combining TV regularizers and wavelet representations. Motivated by this work, we consider inpainting by L22ITVR to reconstruct images with missing data. We consider the 512×512 Head CT image with 40% missing sample (see Figure 8.4). The data is constructed by setting

$$[m, n] = \text{size}(x_0), \quad A = \text{rand}(m, n) > 0.4, \\ y = x_0 .* A + \text{sigma} * \text{rand}(m, n),$$

where

$$\text{BSNR} = 40, \quad V = \text{var}(x_0), \quad \text{sigma} = \sqrt{V / 10^{(\text{BSNR} / 10)}}.$$

Afterwards, TwIST, SpaRSA [158], FISTA, and OSGA are employed to recover the image. We set the regularization parameter $\lambda = 9 \times 10^{-2}$. We draw your attention to the different running time of these algorithms for denoising (see Table 8.1), and so we stop the algorithms after 20 seconds of the running time. To have a fair comparison, three versions of each TwIST, SpaRSA, and FISTA regarding 5, 10, and 20 iterations of Chambolle's algorithm for solving the proximal subproblems

are considered. The results of these implementations are illustrated in Table 8.2 and Figure 8.3.

Table 8.2: A comparison among TwIST, SpaRSA, FISTA, and OSGA for inpainting the 512×512 Head CT image with L2ITVR, where TwIST, SpaRSA, and FISTA use Chambolle's algorithm with several numbers of iterations.

Efficiency measure	Chambolle iter.	TwIST	TwIST	FISTA	OSGA
Function value	5	189601.72	189419.55	189015.72	186463.15
PSNR		29.94	29.96	30.44	33.01
Function value	10	191105.87	188561.73	196281.17	185356.92
PSNR		29.43	30.16	30.15	32.98
Function value	20	214236.01	238919.49	281575.93	185745.69
PSNR		27.26	25.57	25.77	32.91

Figure 8.3 displays results for the relative errors δ_1 and δ_2 (8.9) and also ISNR, where the results show that OSGA is superior to TwIST, SpaRSA, and FISTA regarding all considered efficiency measures. Since the data is not scaled, PSNR is computed by

$$\text{PSNR} = 20 \log_{10} \left(\frac{255\sqrt{mn}}{\|x - x_0\|_F} \right), \quad (8.12)$$

where x_0 is the clean image. This definition proposes that a bigger PSNR value means a smaller $\|x - x_0\|_2$, suggesting a better quality for the restored image. Conversely, a smaller PSNR value indicates a bigger $\|x - x_0\|_2$ leading to the worse image quality. The results suggest that OSGA achieves the best function value 186463.15 and the best PSNR 33.01 among the others.

8.1.1.3 Deblurring with total variation

Image blur is one of the most common problem which happens in the photography and can often ruin photographs. In digital photography, the motion blur is caused by camera shakes, which is difficult to avoid in many situations. Deblurring has been a important problem in the fields of digital imaging, which is an inverse problem of the form (4.5). This is an inherently ill-conditioned problem leading to the optimization problem of the form (4.2). In general, deblurring is much harder than denoising because most of algorithms need to solve a subproblem of the form (8.7) in each iteration.

We here study deblurring by using the models L2ITVR and L1ITVR, where they used for problems involving additive and impulsive noise, respectively, cf.

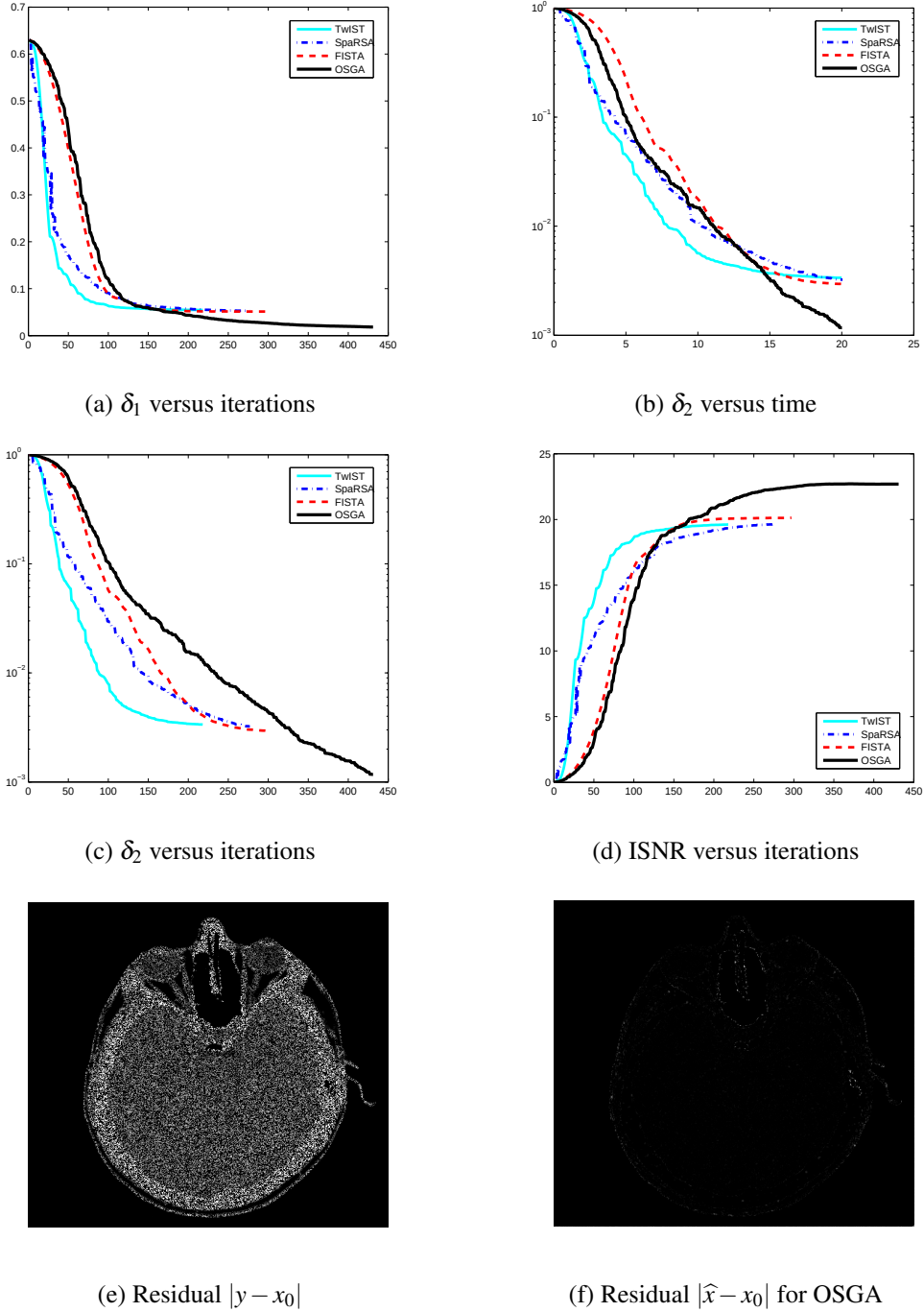


Figure 8.3: Inpainting by TwIST, SpaRSA, FISTA, and OSGA for the 512×512 Head CT, where they were stopped after 20 seconds with $\lambda = 9 \times 10^{-2}$. Subfigure (a) displays the relative error δ_1 of points versus iterations, Subfigure (b) shows the relative error δ_2 of function values versus time, Subfigure (c) displays the relative error δ_2 of function values versus iterations, Subfigure (d) depicts ISNR versus iterations, Subfigure (e) displays the residual $|y - x_0|$, and Subfigure (f) shows the residual $|\hat{x} - x_0|$ for OSGA.,.



(a) Clean image



(b) Noisy image

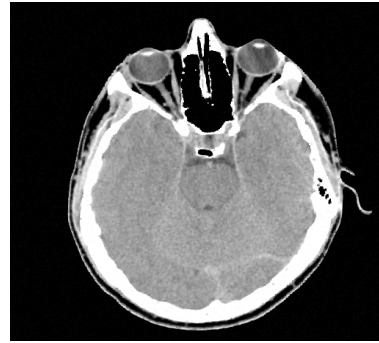
(c) IST ($f = 3791.56$, $\text{PSNR} = 30.05$, time = 20.32)(d) TwIST ($f = 3791.56$, $\text{PSNR} = 30.05$, time = 38.55)(e) FISTA ($f = 3791.56$, $\text{PSNR} = 30.05$, time = 19.67)(f) OSGA ($f = 3651.67$, $\text{PSNR} = 30.09$, time = 5.52)

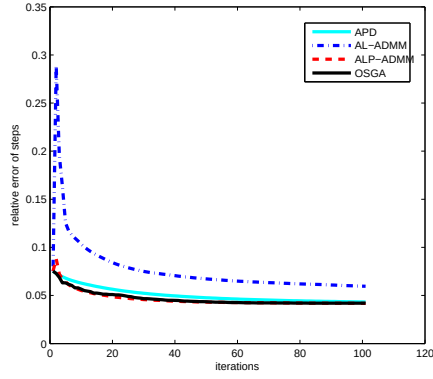
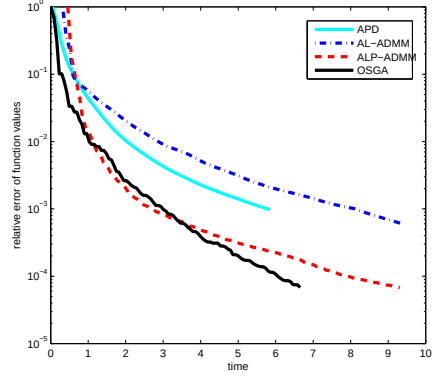
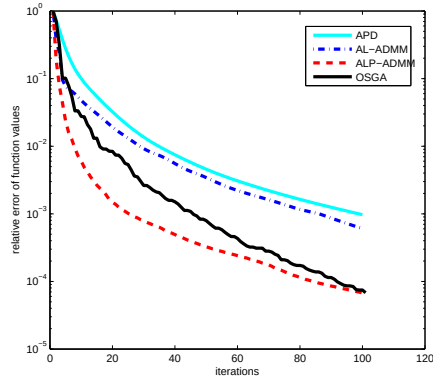
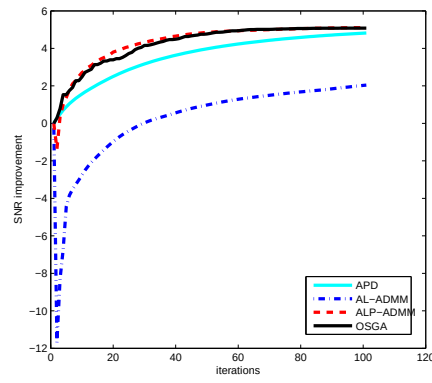
Figure 8.4: A comparison among TwIST, SpaRSA, FISTA, and OSGA for inpainting the 512×512 Head CT, where they were stopped after 20 seconds of the running time. We used the regularization parameter $\lambda = 9 \times 10^{-2}$. Subfigures (a) and (b) show the clean and noisy images, respectively. Subfigures (c), (d), (e), and (f) illustrate the recovered images. 115

[59, 133, 132]. We first consider deblurring of the 512×512 blurred/noisy Elaine image (see Figure 8.6) and reconstruct the image using the L22ITVR model by APD (an accelerated primal-dual method proposed by CHEN et al. [61]), AL-ADMM and ALP-ADMM (two accelerated ADMM methods proposed by CHEN et al. [60]), and OSGA. The blurred/noisy image y is generated by a 9×9 uniform blur and by adding a Gaussian noise with $\text{SNR} = 40$ dB. We stop the algorithms after 100 iterations. The results are summarized in Table 8.3 and Figures 8.5 and 8.6. We then consider deblurring of the 512×512 blurred/noisy Mandril image (Figure 8.8) and restore the image using the L1ITVR model by DRPD1, DRPD2 (two Douglas-Rochford primal-dual methods proposed by BOT & HENDRICH [42]), ADMM (an alternating direction methods proposed by CHAN et al. in [56]), and OSGA. In this case the blurred/noisy image is generated by a 7×7 Gaussian kernel with standard deviation 5 and salt-and-peper impulsive noise with the level 40%. We stop the algorithms after 100 iterations. The results are summarized in Table 8.4 and Figures 8.7 and 8.8.

In Tables 8.3 and 8.4, we report the results of the reconstruction with three regularization parameters that involves the best function value, PSNR and the running time of the solvers. The results show the competitive behavior of the algorithms, however, OSGA obtains the best results in the most cases. In Figure 8.5, one can see that OSGA and ALP-ADMM are comparable in the sense of all efficiency measures. ADP comparable with OSGA and ALP-ADMM regarding the relative error δ_1 and ISNR, but not in the sense of the relative error of function values δ_2 . The worst results obtained by AL-ADMM for all efficiency measures. In Figure 8.7, we see that DRPD1, DRPD2, and OSGA are nearly competitive and superior to ADMM regarding all efficiency measures. However, OSGA attains the best results in the most cases.

We finally consider 72 popular test images to do more comparisons, where blurred/noisy images are generated by a 9×9 uniform blur and adding a Gaussian noise with $\text{SNR} = 40$ dB. The images recovered by TwIST, SpaRSA, FISTA, and OSGA, where the algorithm are stopped after 100 iterations. The results of reconstructions are summarized in Table 8.5, where we report the best function value (F), PSNR and the running time (T) suggesting OSGA outperforms the others.

To have a fair comparison among the considered algorithms, we use the performance profiles of DOLAN & MORE in [75], where the measures of performance are the number of best function evaluations (N_F), and the number of best PSNR (N_{PSNR}). The performance of each code is measured by considering the ratio of its computational outcome versus the best numerical outcome of all codes. This profile offers a tool for comparing the performance of iterative schemes in a statistical structure. Let S be a set of all algorithms and P be a set of test problems. For each problem p and solver s , $t_{p,s}$ is the computational outcome regarding to

(a) δ_1 versus iterations(b) δ_2 versus time(c) δ_2 versus iterations

(d) ISNR versus iterations

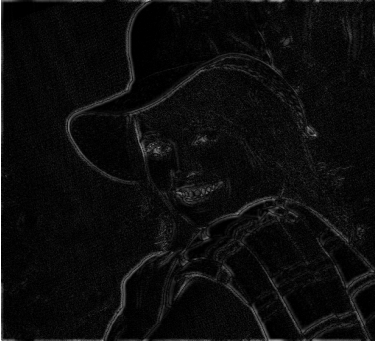
(e) Residual $|y - x_0|$ (f) Residual $|\hat{x} - x_0|$ for OSGA

Figure 8.5: Deblurring with APD, AL-ADMM, ALP-ADMM, and OSGA for the 512×512 Elaine image, where they were stopped after 100 iterations with $\lambda = 1 \times 10^{-2}$. Subfigure (a) displays the relative error δ_1 , Subfigure (b) shows the relative error δ_2 of function values versus time, Subfigure (c) displays the relative error δ_2 of function values versus iterations, Subfigure (d) depicts ISNR versus iterations, Subfigure (e) displays the residual $|y - x_0|$, Subfigure (f) shows the residual $|\hat{x} - x_0|$ for OSGA.



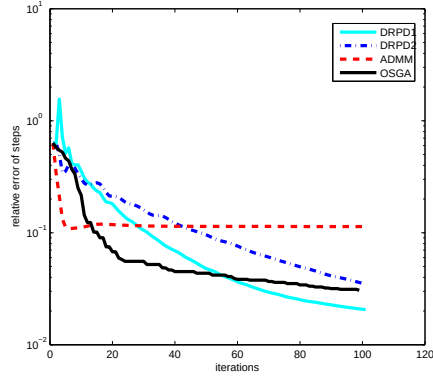
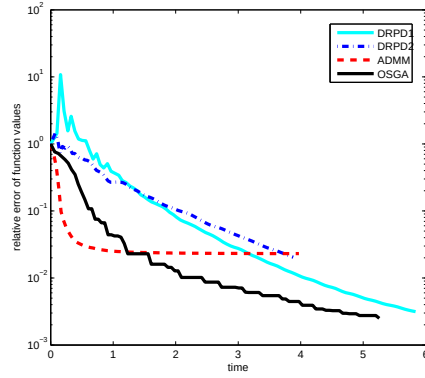
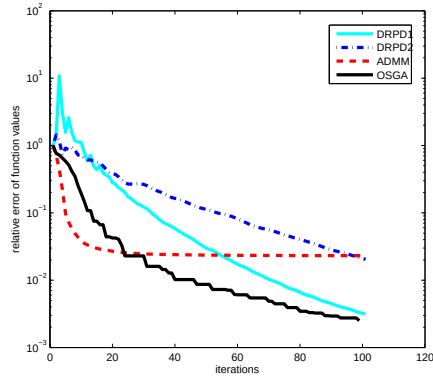
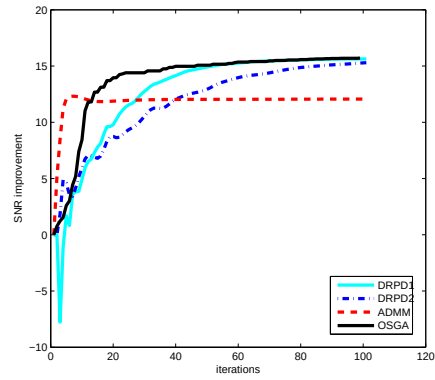
(a) Clean image



(b) Noisy image

(c) IST ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 20.32$)(d) TwIST ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 38.55$)(e) FISTA ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 19.67$)(f) OSGA ($f = 3651.67$, $\text{PSNR} = 30.09$, $\text{time} = 5.52$)

Figure 8.6: Deblurring APD, AL-ADMM, ALP-ADMM, and OSGA for the 512×512 Elaine image, where they were stopped after 100 iterations with $\lambda = 1 \times 10^{-2}$, and all algorithms. Subfigures (a) and (b) show the clean and noisy images, respectively. Subfigures (c), (d), (e), and (f) illustrate the recovered images.

(a) δ_1 versus iterations(b) δ_2 versus time(c) δ_2 versus iterations

(d) ISNR versus iterations

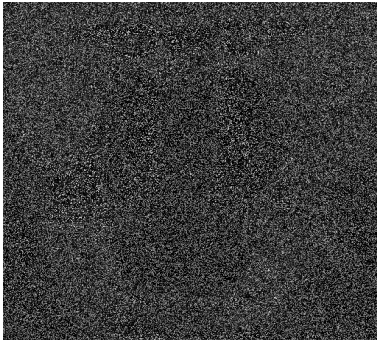
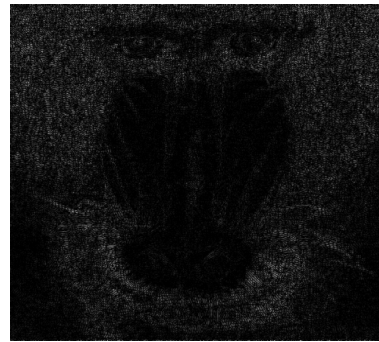
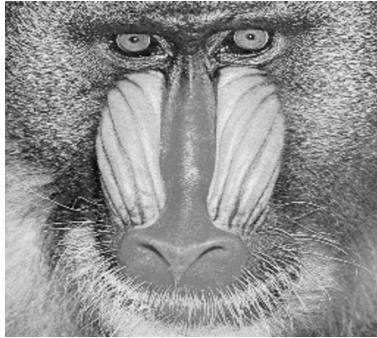
(e) Residual $|y - x_0|$ (f) Residual $|\hat{x} - x_0|$ for OSGA

Figure 8.7: Deblurring with DRPD1, DRPD2, ADMM, and OSGA for the 512×512 Mandril image, where they were stopped after 100 iterations with $\lambda = 10^{-2}$. Subfigure (a) displays the relative error δ_1 , Subfigure (b) shows the relative error δ_2 of function values versus time, Subfigure (c) displays the relative error δ_2 of function values versus iterations, Subfigure (d) depicts ISNR versus iterations, Subfigure (e) displays the residual $|y - x_0|$, and Subfigure (f) shows the residual $|\hat{x} - x_0|$ for OSGA.



(a) Clean image



(b) Noisy image

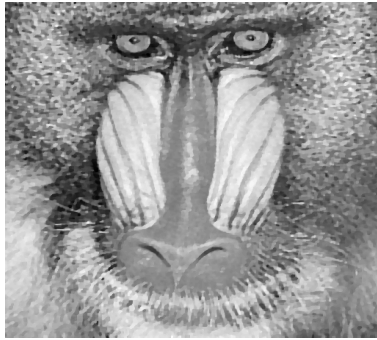
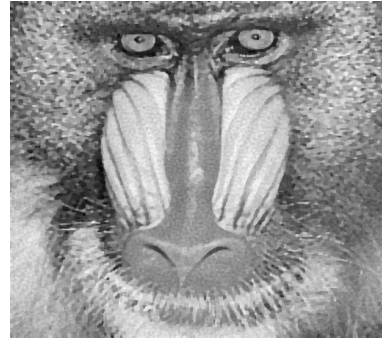
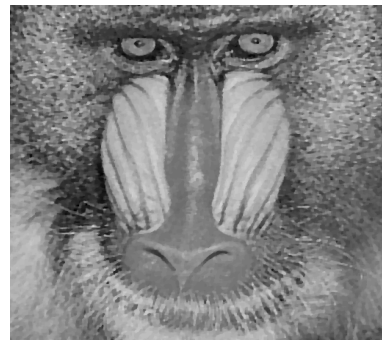
(c) IST ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 20.32$)(d) TwIST ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 38.55$)(e) FISTA ($f = 3791.56$, $\text{PSNR} = 30.05$, $\text{time} = 19.67$)(f) OSGA ($f = 3651.67$, $\text{PSNR} = 30.09$, $\text{time} = 5.52$)

Figure 8.8: Deblurring with DRPD1, DRPD2, ADMM, and OSGA for the 512×512 Mandril image, where they were stopped after 100 iterations with $\lambda = 10^{-2}$, and all algorithms . Subfigures (a) and (b) show the clean and noisy images, respectively. Subfigures (c), (d), (e), and (f) illustrate the recovered images.

Table 8.3: A comparison among APD, AL-ADMM, ALP-ADMM, and OSGA for deblurring the 512×512 Elaine image with L22ITVR for several regularization parameters.

Efficiency measure	Reg. par.	APD	AL-ADMM	ALP-ADMM	OSGA
Function value	$\lambda = 4 \times 10^{-2}$	70603.53	599796.83	65266.97	65231.30
PSNR		31.49	28.47	32.30	32.30
Time		5.85	9.56	9.64	6.84
Function value	$\lambda = 1 \times 10^{-2}$	33471.17	149781.26	32723.75	32546.81
PSNR		32.25	29.47	32.54	32.51
Time		5.91	5.45	9.37	6.73
Function value	$\lambda = 7 \times 10^{-3}$	29058.77	95691.36	28742.07	28474.97
PSNR		32.31	29.80	32.33	32.36
Time		5.38	9.22	9.38	9.69

the performance index, which is used in defining the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}. \quad (8.13)$$

If an algorithm s is failed to solve a problem p , the procedure sets $r_{p,s} = r_{\text{failed}}$, where r_{failed} should be strictly larger than any performance ratio (8.13). For any factor τ , the overall performance of an algorithm s is given by

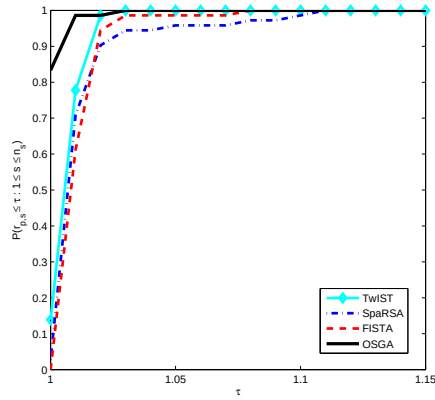
$$\rho_s(\tau) = \frac{1}{n_p} \text{size}\{p \in P : r_{p,s} \leq \tau\}.$$

In fact $\rho_s(\tau)$ is the probability that a performance ratio $r_{p,s}$ of the algorithm $s \in S$ is within a factor $\tau \in \mathbb{R}^n$ of the best possible ratio. The function $\rho_s(\tau)$ is a distribution function for the performance ratio. In particular, $\rho_s(1)$ gives the probability that the algorithm s wins over all other considered algorithms, and $\lim_{\tau \rightarrow r_{\text{failed}}} \rho_s(\tau)$ gives the probability of that the algorithm s solve all considered problems. Hence the performance profile can be considered as a measure of efficiency for comparing iterative schemes.

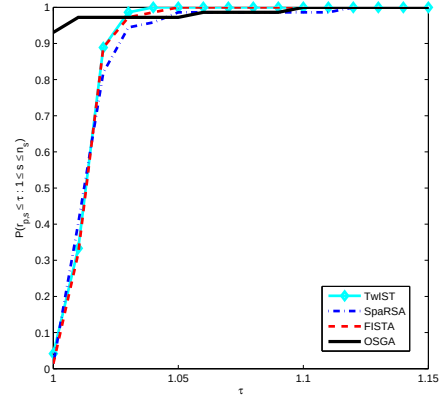
The results of Table 8.5 are illustrated in Figure 8.9, where the comparisons are based on the performance profiles for the best function values and the best PSNR. In Figure 8.9, the x-axis shows the number τ while the y-axis inhibits $P(r_{p,s} \leq \tau : 1 \leq s \leq n_s)$. From Figure 8.9, it can be shown that OSGA wins with 84% and 93% score for function values and PSNR, respectively. It also indicates that all algorithms recover images successfully, however, the quality of the recovered images are different for the considered algorithms.

Table 8.4: A comparison among DRPD1, DRPD2, ADMM, and OSGA for deblurring the 512×512 Mandril image with L1ITVR for several regularization parameters.

Efficiency measure	Reg. par.	DRPD1	DRPD2	ADMM	OSGA
Function value	$\lambda = 4 \times 10^{-1}$	55511.83	56930.71	58761.89	655271.36
PSNR		23.09	22.53	21.54	23.19
Time		6.56	4.34	4.17	5.69
Function value	$\lambda = 1 \times 10^{-1}$	53527.63	53773.75	58757.44	53533.96
PSNR		25.10	24.76	21.52	25.18
Time		5.84	3.88	4.04	5.30
Function value	$\lambda = 7 \times 10^{-2}$	53250.56	53370.86	58758.89	53322.27
PSNR		25.70	25.38	21.53	25.77
Time		5.58	3.57	3.54	4.85



(a) performance profile for function values



(b) performance profile for PSNR

Figure 8.9: Deblurring of 72 test images with isotropic total variation using TwIST, SpaRSA FISTA, OSGA in 100 iterations.

Table 4.5. Deblurring using the isotropic TV for efficiency measures F, T, and PSNR

Image name	Dimension	TwIST			SpaRSA			FISTA			OSGA		
		F	PSNR	T	F	PSNR	T	F	PSNR	T	F	PSNR	T
Aerial	256 × 256	53413.79	25.00	2.74	53071.70	25.10	1.88	53540.25	25.10	1.92	50306.67	25.64	1.54
Airplane	256 × 256	16532.72	33.78	3.10	16571.13	33.73	1.84	16556.29	33.72	1.83	14010.82	34.43	1.48
Cameraman	256 × 256	37080.67	27.38	3.14	37284.69	27.06	1.87	37099.38	27.44	1.84	35634.80	27.74	1.53
Chemical plant	256 × 256	37862.89	26.76	2.48	37991.77	26.89	1.76	37893.42	26.79	1.94	35490.22	27.21	1.57
Clock	256 × 256	31558.86	31.13	2.54	31976.01	31.09	1.75	31583.19	31.22	1.86	30434.16	31.65	1.51
Fingerprint1	256 × 256	54297.30	20.98	3.16	54011.03	21.08	1.74	54387.92	20.98	1.77	52756.10	21.22	1.52
Lena	256 × 256	30761.92	28.36	3.00	30680.49	28.32	1.68	30787.62	28.40	1.91	28885.18	28.77	1.50
Moon surface	256 × 256	16634.15	30.23	3.80	16539.92	30.29	2.00	16707.90	30.27	2.15	15649.84	30.54	1.78
Pattern 1	256 × 256	23117.29	18.08	2.24	231927.89	17.99	1.96	231888.76	18.09	2.04	230312.01	18.08	1.83
Pattern 2	256 × 256	170.76	48.88	4.27	183.58	48.91	2.04	186.14	48.90	2.37	179.46	48.88	1.74
Star	256 × 256	23529.93	24.31	2.47	23755.74	24.25	1.90	23560.15	24.37	2.13	23123.93	24.48	1.82
Aerial	512 × 512	155822.07	25.94	14.00	155721.55	25.89	9.63	156011.24	26.01	10.78	148752.16	26.51	7.38
Airfield	512 × 512	169670.32	26.06	12.93	169736.38	25.94	9.47	169745.72	26.14	10.35	161605.42	26.64	7.34
Airplane 1	512 × 512	105964.34	32.00	18.37	108332.51	32.10	9.81	106086.67	32.08	10.76	99262.86	32.69	7.32
Airplane 2	512 × 512	29312.36	35.91	12.09	29615.58	35.16	10.37	29326.77	35.91	10.66	28148.52	35.86	7.45
APC	512 × 512	45314.17	33.18	15.51	45210.76	33.20	10.14	45434.17	33.21	10.66	43189.14	33.47	7.41
Barbara	512 × 512	114574.88	25.46	19.06	115650.15	25.45	9.73	114877.79	25.43	10.31	109861.71	25.60	7.45
Blobs	512 × 512	191935.15	20.63	12.95	192076.22	20.61	9.74	192064.04	20.63	10.85	186085.58	20.85	7.48
Blonde woman	512 × 512	96415.14	29.64	19.90	95926.96	29.69	10.15	96647.01	29.66	10.36	89192.17	30.01	7.35
Boat	512 × 512	118502.34	29.88	18.76	118296.01	29.88	9.85	118683.01	29.91	10.40	114310.67	30.33	7.33
Cameraman	512 × 512	111597.14	32.97	18.00	111625.87	33.02	10.21	111665.27	33.00	10.29	107864.88	33.54	7.40
Car & APCs 1	512 × 512	75485.90	32.67	19.37	93730.90	32.80	9.89	75818.64	32.61	10.81	69079.84	32.97	7.37
Car & APCs 2	512 × 512	69199.09	32.52	19.52	68666.22	32.66	9.97	69628.04	32.43	10.24	65947.88	32.82	7.28
Clown	512 × 512	137174.69	31.71	16.96	137012.22	31.74	10.08	137270.47	31.76	10.44	132934.32	32.13	7.44
Crowd 1	512 × 512	127146.90	30.56	13.68	126737.37	30.60	9.83	127377.69	30.63	10.55	121181.67	31.33	7.29
Crowd 2	512 × 512	213755.44	28.15	14.24	211781.24	28.58	9.57	214364.33	28.14	10.57	204709.94	29.01	7.39
Darkhair woman	512 × 512	85750.82	37.30	19.62	85313.93	37.59	9.67	86072.42	36.95	10.34	81041.55	37.70	7.47
Dollar	512 × 512	249192.85	21.49	12.74	250157.44	21.36	9.62	249360.99	21.50	10.30	242033.17	21.69	7.35
Elaine	512 × 512	80496.96	32.01	14.88	79759.33	32.13	9.79	80842.38	31.93	10.23	75342.08	32.22	7.42
Fingerprint	512 × 512	256083.70	26.10	17.93	255341.46	26.19	9.49	256869.36	25.86	10.35	246958.84	26.48	7.48
Flintstones	512 × 512	244801.40	25.62	13.47	246006.66	25.36	9.69	245079.39	25.65	11.45	236039.78	26.44	7.42
Girlface	512 × 512	82824.71	33.00	18.12	83004.44	33.07	9.55	83085.95	32.90	10.24	79696.26	33.45	7.41
Goldhill	512 × 512	109881.34	30.69	18.80	109584.32	30.78	9.82	110089.95	30.71	10.41	103156.32	31.05	7.40
Head CT	512 × 512	178747.61	30.90	13.08	179909.50	30.09	10.12	178905.76	30.90	10.77	176773.80	31.48	7.47
Houses	512 × 512	250247.06	25.77	13.61	250488.78	25.63	9.87	250381.11	25.83	10.52	244706.72	25.92	7.59
Kiel	512 × 512	156156.28	25.82	13.10	156220.46	25.72	9.86	156234.66	25.90	10.43	151066.37	26.05	7.31

Table 4.5. Deblurring using the isotropic TV for efficiency measures F, T, and PSNR (continued)

Lake	512 × 512	154211.63	28.90	18.58	155519.67	28.87	9.77	154391.59	28.95	10.37	147034.65	29.44	7.37
Lena	512 × 512	91477.32	31.76	17.96	91841.72	31.80	9.40	91619.94	31.76	9.92	86890.56	32.26	6.99
Lena & numbers	512 × 512	258169.51	20.82	11.76	260306.99	20.50	9.08	258380.34	20.87	10.29	248372.60	20.90	6.98
Lifingbody	512 × 512	43689.69	37.42	17.87	44045.13	37.28	9.67	43736.07	37.47	9.86	42027.60	37.74	7.15
Lighthouse	512 × 512	169472.66	27.50	16.88	169541.23	27.44	9.55	169562.61	27.56	9.68	163735.37	27.71	7.20
Livingroom	512 × 512	126923.58	29.84	16.82	126599.80	29.85	9.61	127124.15	29.90	10.13	121836.73	30.21	7.21
Mandrill	512 × 512	149477.91	25.89	15.51	150744.03	26.02	9.68	149891.20	25.88	9.72	142733.97	26.30	7.32
MRI spine	512 × 512	85227.94	35.57	18.10	85178.89	35.55	9.43	85430.62	35.49	10.24	84065.50	36.02	7.08
Peppers	512 × 512	101406.85	32.91	18.33	108648.01	33.09	9.37	101658.96	32.89	9.95	95402.78	33.42	7.38
Pirate	512 × 512	108272.00	29.83	13.44	110131.28	29.89	10.18	108462.75	29.89	10.05	103641.24	30.32	7.24
Smiling woman	512 × 512	110529.94	33.10	19.48	110368.95	33.13	9.86	110719.24	33.15	10.24	107609.43	33.61	7.21
Squares	512 × 512	154044.78	50.57	13.20	155031.10	48.74	9.29	154088.35	50.29	10.42	154150.72	47.28	7.14
Tank 1	512 × 512	63514.51	31.45	14.94	63242.30	31.48	9.48	63699.75	31.44	10.03	61017.54	31.71	7.08
Tank 2	512 × 512	85811.79	29.38	19.05	88841.30	29.50	9.38	86380.50	29.37	9.74	81967.52	29.70	7.27
Tank 3	512 × 512	93860.02	29.65	18.55	93157.67	29.80	9.35	94117.77	29.66	10.13	89911.98	29.95	7.18
Truck	512 × 512	73350.18	31.77	18.41	72892.65	31.91	9.24	73680.47	31.76	10.30	69367.80	32.17	7.15
Truck & APCs 1	512 × 512	106688.21	28.70	15.70	106115.20	28.78	9.37	107107.08	28.70	10.65	101953.56	28.99	7.09
Truck & APCs 2	512 × 512	105130.10	28.74	19.35	104575.63	28.81	9.60	105625.01	28.71	10.65	100309.61	29.04	7.15
Washington DC 1	512 × 512	255763.94	20.10	16.79	255184.94	20.05	9.40	256031.63	20.09	10.20	245960.99	20.40	7.06
Washington DC 2	512 × 512	257130.06	20.06	17.75	258341.54	20.06	9.36	257380.51	20.07	10.02	247358.58	20.38	7.02
Washington DC 3	512 × 512	261625.21	20.02	18.83	260465.27	20.05	9.60	261838.35	20.04	10.03	251496.47	20.35	7.18
Washington DC 4	512 × 512	222447.39	20.81	18.04	221281.76	20.84	9.25	222626.17	20.82	10.01	214024.43	21.07	7.12
Zelda	512 × 512	71208.74	35.27	19.85	70564.71	35.48	9.45	71622.59	35.09	9.77	66543.63	35.76	7.22
Dark blobs 1	600 × 600	200775.29	23.93	19.97	201055.66	23.91	14.06	200897.01	23.95	15.01	194222.26	24.09	11.07
Dark blobs 2	600 × 600	197989.51	23.85	20.03	198142.62	23.83	14.85	198111.05	23.86	15.68	191100.22	24.01	10.80
House	600 × 600	129609.25	33.80	23.94	129515.79	33.84	15.20	129767.13	33.84	15.30	123903.08	33.93	10.74
Ordered matches	600 × 600	208071.15	29.23	26.76	207434.95	29.31	14.40	208327.89	29.18	15.45	202771.66	29.38	10.82
Random matches	600 × 600	218890.81	28.00	27.03	543099.79	28.09	14.41	219474.50	27.90	15.42	212901.36	28.29	10.83
Rice	600 × 600	92126.13	40.14	28.10	93409.46	40.81	14.70	92749.29	39.56	15.14	87191.04	41.13	10.91
Shepp-logan phantom	600 × 600	83287.13	35.25	17.61	85801.08	31.65	14.78	83620.64	34.39	15.49	84941.71	32.35	10.87
Airport	1024 × 1024	423716.48	28.26	71.13	422343.15	28.26	52.60	424138.62	28.33	54.02	408879.42	28.68	31.84
Pentagon	1024 × 1024	324888.21	28.48	97.96	323296.19	28.54	51.57	325971.63	28.52	54.37	313935.78	28.83	32.29
Pirate	1024 × 1024	461898.16	30.22	104.04	460328.32	30.27	51.60	462700.56	30.24	54.65	445895.79	30.70	31.89
Rose	1024 × 1024	289211.25	37.12	95.60	289158.13	37.16	50.85	289587.98	36.99	51.83	286811.37	37.31	29.97
Testpat	1024 × 1024	922177.51	21.88	60.62	928592.01	21.68	48.79	923710.50	21.88	51.84	91885.14	21.86	30.24
Washington DC	1024 × 1024	816605.53	23.60	97.23	812400.30	23.67	48.84	817541.98	23.61	52.24	785992.35	23.97	30.61
Average		159829.01	29.37	21.09	164820.79	29.30	12.38	160109.55	29.34	13.10	154452.35	29.64	8.75

8.1.2 A comparison among first-order methods

We here give a comparison among some first-order algorithms, where they use the nonsmooth first-order oracle or employing the smooth first-order oracle and a proximity operator. In our comparison we particularly consider PGA (proximal gradient algorithm [138]), NSDSG (nonsummable diminishing subgradient algorithm [44]), FISTA (Beck and Teboulle's fast proximal gradient algorithm [28]), NESCO (Nesterov's composite optimal algorithm [123]), NESUN (Nesterov's universal gradient algorithm [124]), NES83 (Nesterov's 1983 optimal algorithm [118]), NESCS (Nesterov's constant step optimal algorithm [119]), and NES05 (Nesterov's 2005 optimal algorithm [121]). The codes of these algorithms are written in MATLAB, where we use the parameters proposed in associated papers.

The algorithms NES83, NESCS, and NES05 were originally proposed to solve smooth convex problems, where they use the smooth first-order oracle and attain the optimal complexity $O(\varepsilon^{-1/2})$. Although, obtaining the optimal complexity bound of smooth problems is computationally very interesting, the considered problem (4.2) is commonly nonsmooth. Recently, LEWIS and OVERTON in [107] investigated the behavior of the BFGS method for nonsmooth nonconvex problems, where their results are interesting. In addition, NESTEROV in [123] showed that the subgradient of composite functions preserve the most important features of the gradient of smooth convex functions. These facts motivate the conquest to conduct some computational investigation on the behavior of optimal smooth first-order methods by passing the nonsmooth first-order oracle, which provides a function value and a subgradient. In particular, we consider Nesterov's 1983 algorithm, see [118], and adapt it to solve (4.2) by simply passing a subgradient of the function as described in Algorithm 11.

In a similar way considered for NES83, NESCS and NES05 are adapted from NESTEROV's constant step [119] and NESTEROV's 2005 algorithms [121], respectively. The adapted versions of NES83, NESCS, and NES05 are able to solve nonsmooth problems as well, however, there is no theory to support their convergence at the moment.

Among the algorithms considered, PGA, NSDSG, FISTA, NESCO, and NESUN are originally proposed for solving nonsmooth problems, where NSDSG only needs the nonsmooth first-order oracle and the others need the smooth first-order oracle and a proximal mapping. NSDSG is optimal for nonsmooth Lipschitz continuous problems with the complexity $O(\varepsilon^{-2})$, PGA is not optimal attaining the complexity $O(\varepsilon^{-1})$, and FISTA, NESCO, and NESUN are optimal for composite problems with the complexity $O(\varepsilon^{-1/2})$. These complexity bounds propose that NSDSG and PGA are not theoretically comparable with the others. However, we still consider them in our comparisons to show the impact of the complexity bounds for the first-order methods, especially for solving problems in applications

Algorithm 11: NES83 (Nesterov's 1983 scheme for composite functions)**Input:** select z and y_0 such that $z \neq y_0$ and $g_{y_0} \neq g_z$, $\varepsilon > 0$;**Output:** x_k, f_{x_k} ;**1 begin****2** $a_0 = 0$; $x_{-1} = y_0$; compute g_{y_0} and g_z using NFO-G;**3** $\alpha_{-1} = \|y_0 - z\| / \|g_{y_0} - g_z\|$;**4** **while** *stopping criteria do not hold* **do****5** $\hat{\alpha}_k = \alpha_{k-1}$; $\hat{x}_k = y_k - \hat{\alpha}_k g_{y_k}$; compute $f_{\hat{x}_k}$ using NFO-F;**6** **while** $f_{\hat{x}_k} < f_{y_k} - \frac{1}{2} \hat{\alpha}_k \|g_{y_k}\|^2$ **do****7** $\hat{\alpha}_k = \rho \hat{\alpha}_k$; $\hat{x}_k = y_k - \hat{\alpha}_k g_{y_k}$; compute $f_{\hat{x}_k}$ using NFO-F;**8** **end****9** $x_k = \hat{x}_k$; $f_{x_k} = f_{\hat{x}_k}$; $\alpha_k = \hat{\alpha}_k$; $a_{k+1} = \left(1 + \sqrt{4a_k^2 + 1}\right) / 2$;**10** $y_{k+1} = x_k + (a_k - 1)(x_k - x_{k-1}) / a_{k+1}$; compute $g_{y_{k+1}}$ using NFO-G;**11** **end****12 end**

involving high-dimensional data.

We compare OSGA with the algorithms only using the nonsmooth first-order oracle (NSDSG, NES83, NESCS, NES05) and with those using the smooth first-order oracle and a proximal mapping (PGA, NSDSG, FISTA, NESCO, NESUN) in separate comparisons. Most of these algorithms need to know about Lipschitz constants to determine a step-size. While NESCS, NES05, PGA and FISTA use the constant L to determine a step-size, NESCO and NESUN get a lower approximation of L and adapt it by a backtracking line search, for more details see [123] and [124]. In our implementations, similar to BECKER et al. [30], NESCO and NESUN use the initial estimate

$$L_0 = \frac{\|g(x_0) - g(z_0)\|_*}{\|x_0 - z_0\|}$$

of the Lipschitz constant of gradients, where x_0 and z_0 are two arbitrary points satisfying $x_0 \neq z_0$ and $g(x_0) \neq g(z_0)$. In [118], NESTEROV used a similar term to determine an initial step-size for NES83. For NSDSG, the step-size is computed by $\alpha_0(k)^{-1/2}$, where k is the iteration counter and α_0 is a positive constant that should be specified as big as possible such that the algorithm is not divergent, cf. [44]. All the other parameters of the algorithms are set to those reported by the authors.

We here consider solving an underdetermined system $Ax = y$, where A is a $m \times n$ random matrix and y is a random m -vector. This problem is frequently appeared in many applications, where the goal is to determine x by some optimiza-

tion models. Considering the ill-conditioned feature of this problem, the most popular optimization models are (4.7), (4.8), and (4.9), where (4.7) is smooth and (4.8) and (4.9) are nonsmooth. We are particularly interested in solving the multi-term problem (4.9) with both dense and sparse data. Since proximal-based algorithm cannot solve (4.9) for general W , for a comparison of OSGA with PGA, FISTA, NESCO, and NESUN, we set $W = I$, where I is the identity operator. For $m = 2000$ and $n = 5000$, the data is constructed by

$$A = \text{rand}(m, n), \quad y = \text{rand}(1, m), \quad x_0 = \text{rand}(1, n),$$

for the dense problem, and

$$A = \text{sprand}(m, n), \quad y = \text{sprand}(1, m), \quad x_0 = \text{sprand}(1, n),$$

for the sparse problem. The results for three regularization parameters are reported in Table 8.6 and Figure 8.10. For a comparison of OSGA with NSDSG, NES83, NESCS, and NES05, we also set

$$W_1 = \text{rand}(m, n), \quad W_2 = \text{rand}(m, n),$$

for the dense problem, and

$$W_1 = \text{sprand}(p, n), \quad W_2 = \text{sprand}(p, n),$$

for the sparse problem, where $p = 1000$. The results for three regularization parameters are reported in Table 8.7 and Figure 8.11. In both cases we stop the algorithms after 20 seconds of the running time. In our comparison we define

$$\hat{L} := \max_{1 \leq i \leq n} \|a_i\|^2,$$

where a_i , for $i = 1, 2, \dots, n$, is the i -th column of A . In the implementation, NESCS, NES05, PGA, and FISTA use $L = 10^4 \hat{L}$ and $L = 10^2 \hat{L}$ for dense and sparse problems, respectively. Moreover, NSDSG employs $\alpha_0 = 10^{-7}$ and $\alpha_0 = 10^{-4}$ for the dense and sparse problems, respectively.

The results of Tables 8.6 and 8.7 show that OSGA get the best function values for both sparse and dense data and also for all considered regularization parameters. In Figure 8.11, it can be seen that NESCO and NESUN are comparable with OSGA, however, OSGA attains the best results. It is notable that FISTA, NESCO, NESUN, and OSGA get much better results in comparison to the PGA, which supports the theoretical results about the complexity of these methods. The results of Figure 8.11 show that the adapted algorithms NES83, NESCS, and NES05 surprisingly behave comparable with OSGA and their results are much better than NSDSG. It is notable that among the adapted algorithms NES83, NESCS, and NES05, NES83 perform better than the others. However, OSGA is superior to the algorithms regarding function values.

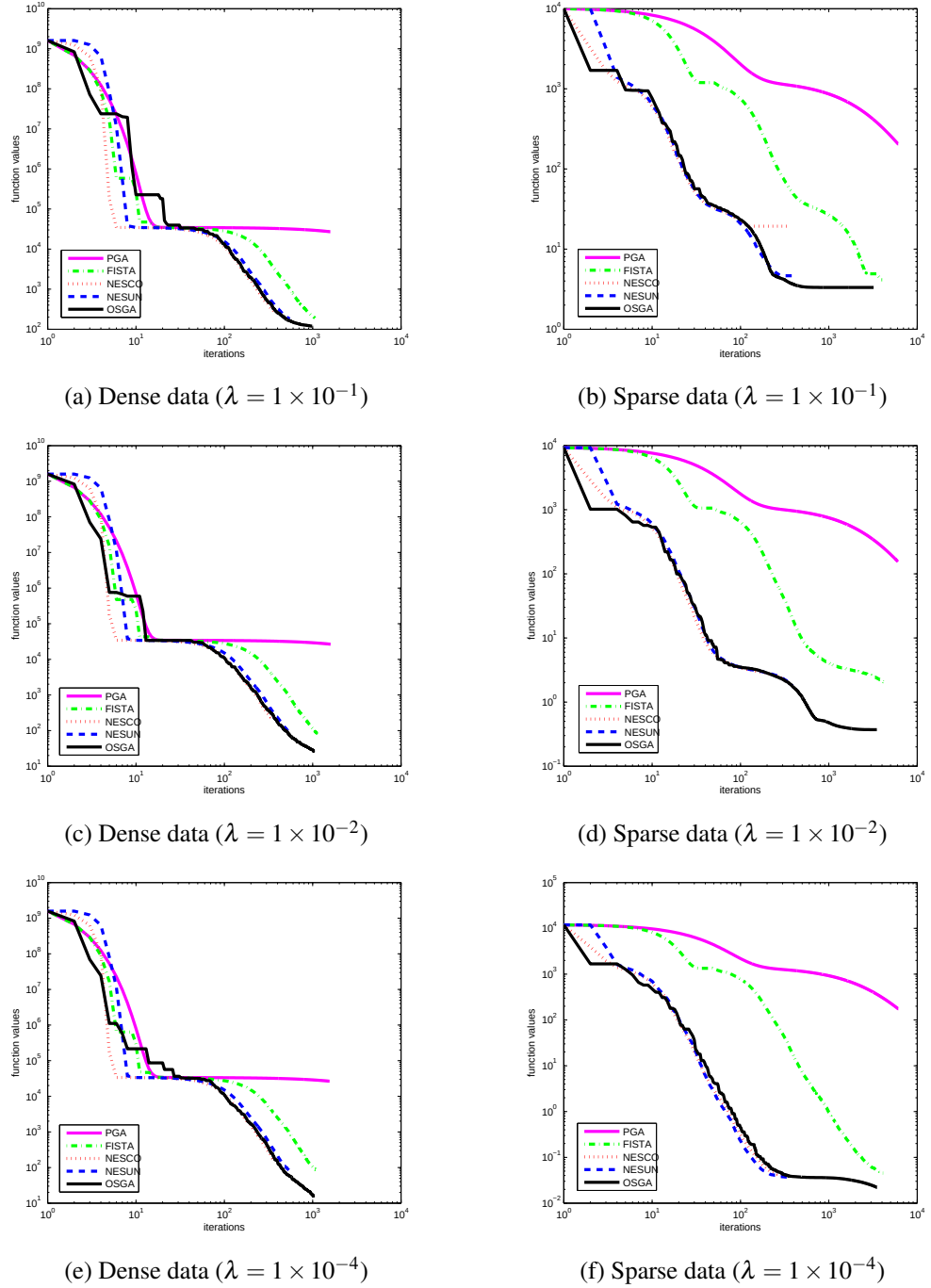


Figure 8.10: A comparison among PGA, FISTA, NESCO, NESUN, and OSGA for solving the Elastic Net problem, where the algorithms stopped after 20 seconds: Subfigures (a) and (b) display function value versus iteration for $\lambda = 1 \times 10^{-1}$; Subfigures (c) and (d) display function value versus iteration for $\lambda = 1 \times 10^{-2}$; Subfigures (e) and (f) display function value versus iteration for $\lambda = 1 \times 10^{-4}$.

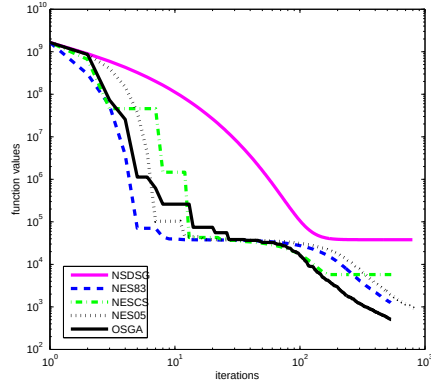
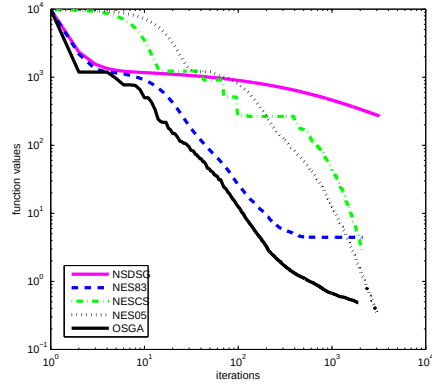
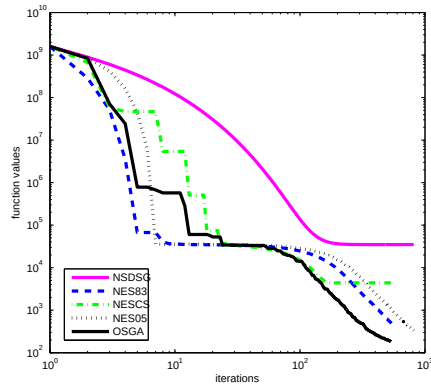
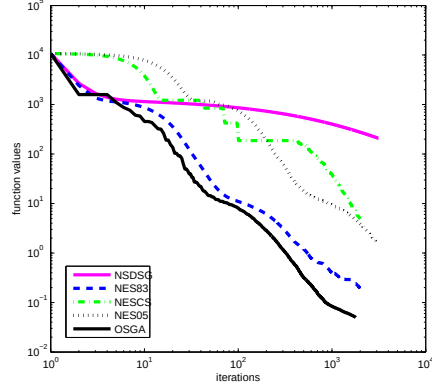
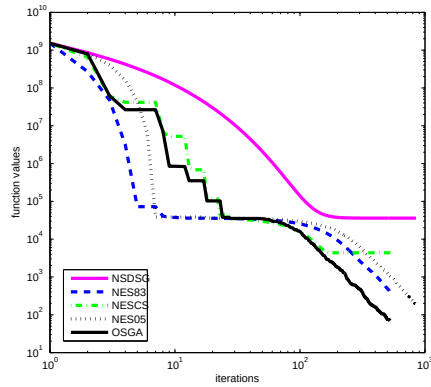
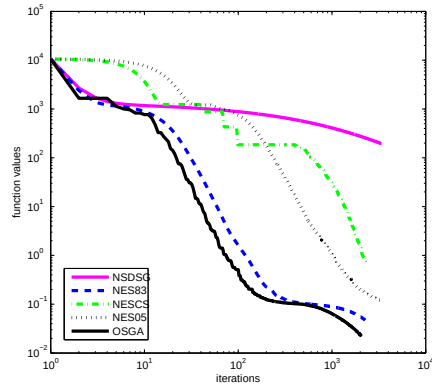
(a) Dense data ($\lambda = 1 \times 10^{-1}$)(b) Sparse data ($\lambda = 1 \times 10^{-1}$)(c) Dense data ($\lambda = 1 \times 10^{-2}$)(d) Sparse data ($\lambda = 1 \times 10^{-2}$)(e) Dense data ($\lambda = 1 \times 10^{-4}$)(f) Sparse data ($\lambda = 1 \times 10^{-4}$)

Figure 8.11: A comparison among NSDSG, NES83, NESCS, NES05, and OSGA for solving the scaled Elastic Net problem, where the algorithms stopped after 20 seconds: Subfigures (a) and (b) display function value versus iteration for $\lambda = 1 \times 10^{-1}$; Subfigures (c) and (d) display function value versus iteration for $\lambda = 1 \times 10^{-2}$; Subfigures (e) and (f) display function value versus iteration for $\lambda = 1 \times 10^{-4}$.

Table 8.6: Function values of PGA, FISTA, NESCO, NESUN, and OSGA for solving scaled Elastic Net problem with several regularization parameters.

Data type	Regularization parameter	PGA	FISTA	NESCO	NESUN	OSGA
Dense	$\lambda = 1 \times 10^{-1}$	27116.97	191.73	275.83	185.09	119.15
	$\lambda = 1 \times 10^{-2}$	26609.03	84.45	195.51	94.92	26.21
	$\lambda = 1 \times 10^{-4}$	26539.66	86.82	170.51	83.02	15.52
Sparse	$\lambda = 1 \times 10^{-1}$	204.63	4.13	19.31	4.64	3.31
	$\lambda = 1 \times 10^{-2}$	154.74	2.06	2.02	1.78	0.37
	$\lambda = 1 \times 10^{-4}$	174.65	0.04	0.03	0.03	0.02

Table 8.7: Function values of NSDSG, NES83, NESCS, NES05, and OSGA for solving Elastic Net problem with several regularization parameters.

Data type	Regularization parameter	NSDSG	NES83	NESCS	NES05	OSGA
Dense	$\lambda = 1 \times 10^{-1}$	37934.77	1259.59	5757.03	915.40	5.3.07
	$\lambda = 1 \times 10^{-2}$	34873.95	457.42	4397.17	335.98	182.26
	$\lambda = 1 \times 10^{-4}$	35995.24	357.86	4356.74	197.82	69.97
Sparse	$\lambda = 1 \times 10^{-1}$	268.67	4.43	2.61	0.33	0.48
	$\lambda = 1 \times 10^{-2}$	207.86	0.19	4.88	1.68	0.05
	$\lambda = 1 \times 10^{-4}$	149.95	0.47	0.66	0.12	0.02

8.1.3 Compressed sensing and sparse optimization

Over the past few decades, finding sparse solutions for many problems by using the structured models has become popular in various areas of applied mathematics, e.g., signal and image processing, geophysics, economics, statistics, machine learning, and data mining. In the most cases, the problem involves high-dimensional data with a small number of available measurements, where the core of these problems involves an optimization problem. Thanks to the sparsity of solutions and the structure of problems, these optimization problems can be solved in reasonable time even for the extremely high-dimensional data sets. Basis pursuit, lasso, wavelet-based deconvolution, and compressed sensing are some examples, where the last one receives lots of attentions during the recent years, cf. [49, 76].

Among the fields involving sparse optimization, *compressed sensing*, also called *compressive sensing* and *compressive sampling*, is a novel sensing-sampling framework for acquiring and recovering objects like a sparse image or signal in the most efficient way possible employing an incoherent projecting basis. Con-

ventional processes in the image/signal acquisition from frequency data follow the Nyquist-Shannon density sampling theorem declaring that the number of samples required for reconstructing an image matches the number of pixels in the image and for recovering a signal without error is devoted by its bandwidth. On the other hand, it is known that most of the data we acquire can be thrown away with almost no perceptual loss. Hence the question is: why all the data should be acquired while most of them will be thrown away? Also, can we only recover parts of the data that will be useful in the final reconstruction? In response to these questions, a novel sensing-sampling approach was introduced which goes against common wisdom in data acquisition, namely compressed sensing, see, for example, [51, 50, 76] and references therein.

It is known that compressed sensing supposes that the considered object, image or signal, has a sparse representation in some bases or dictionaries, and the considered representation dictionary is incoherent with the sensing basis, cf. [49, 76]. The idea behind this framework is centred on how many measurements are necessary to reconstruct an image-signal and the related nonlinear reconstruction techniques needed to recover this image-signal. It means that the object is recovered reasonably well from highly undersampled data, in spite of violating the traditional Nyquist-Shannon sampling constraints. In the other words, once an object is known to be sparse in a specific dictionary, one can find a set of measurements or sensing bases supposed to be incoherent with the dictionary. Afterwards, an underdetermined system of linear equations of the form (4.5) is emerged, which is typically ill-conditioned. Some theoretical results in [51, 50, 76] suggest that the minimum number of measurements required to reconstruct the object provides a specific pair of measurement matrices and optimization problems. Therefore, the main challenge is how to reformulate this inverse problem as an appropriate minimization problem involving regularization terms and solve it by a suitable optimization technique.

We now consider the inverse problem (4.5) with $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, and $y \in \mathbb{R}^m$, where the observation is deficient, $m < n$, as a common interest in compressed sensing. Hence the object x cannot be recovered from the observation y directly, even in noiseless system $Ax = y$, unless some additional assumptions like sparsity of x is presumed. We here consider a sparse signal $x \in \mathbb{R}^n$ to be recovered from an observation $y \in \mathbb{R}^m$, where A is obtained by first filling it with independent samples of the standard Gaussian distribution and then orthonormalizing the rows, see [84]. In our experiment, we consider $m = 5000$ and $n = 10000$, where the original signal x consist of 300 randomly placed ± 1 spikes. The data is constructed by

```
n_spikes=floor(0.03*n); x=zeros(n,1);
q=randperm(n); A=randn(k,n); A=orth(A')';
x(q(1:n_spikes)) = sign(randn(n_spikes,1));
```

$y = A * x + 10^{(-3)} * \text{randn}(k, 1);$

We reconstruct x by using the model (4.8), where the regularization parameter is set to $\lambda = \|Ay\|_{\infty} \times 10^{-1}$ or $\lambda = \|Ay\|_{\infty} \times 10^{-3}$. The algorithms are stopped after 20 seconds of the running time, and the results are illustrated in Figure 8.12.

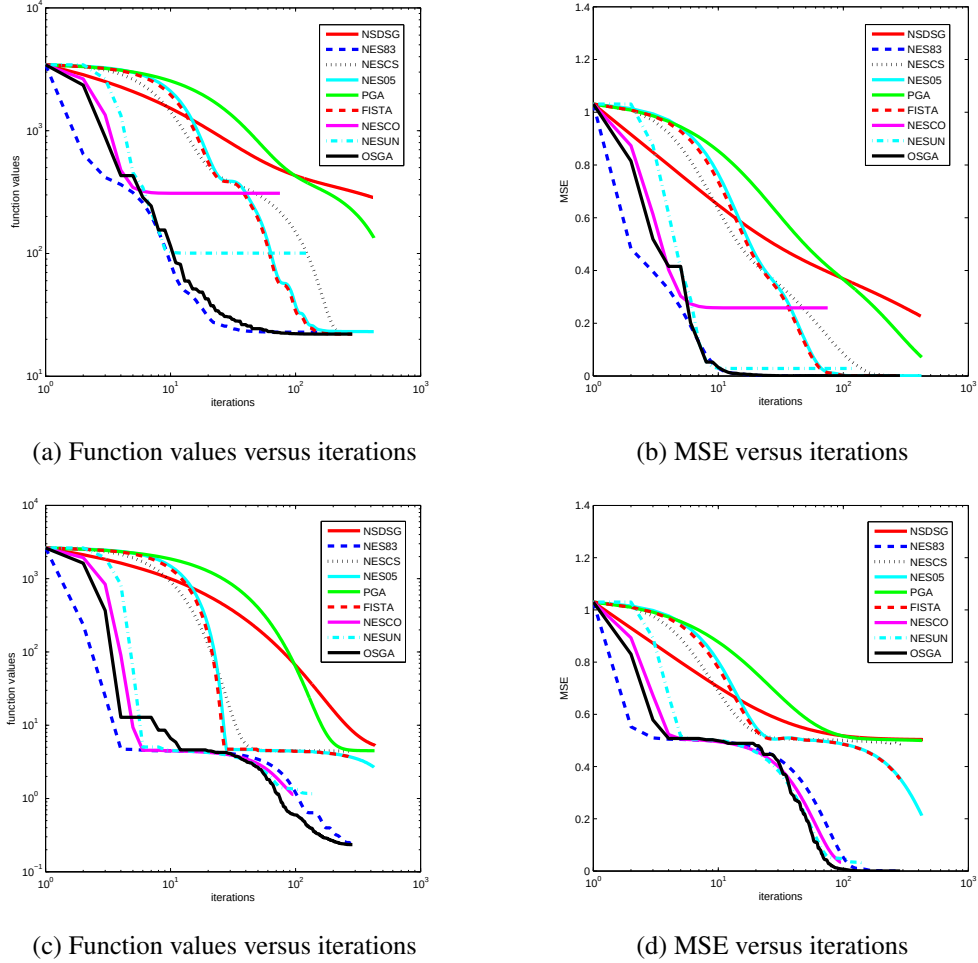


Figure 8.12: Recovering a noisy sparse signal using the $l_2^2 - l_1$ problem, where the algorithms were stopped after 20 seconds. Subfigures (a) and (b) illustrate the results for $\lambda = \|Ay\|_{\infty} \times 10^{-1}$, while subfigures (c) and (d) display the results $\lambda = \|Ay\|_{\infty} \times 10^{-3}$.

Figure 8.12 illustrates the results of the sparse signal recovery by NSDSG, NES83, NESCS, NES05, PGA, FISTA, NESCO, NESUN, and OSGA, where the results are compared in the sense of function values and MSE, where MSE is

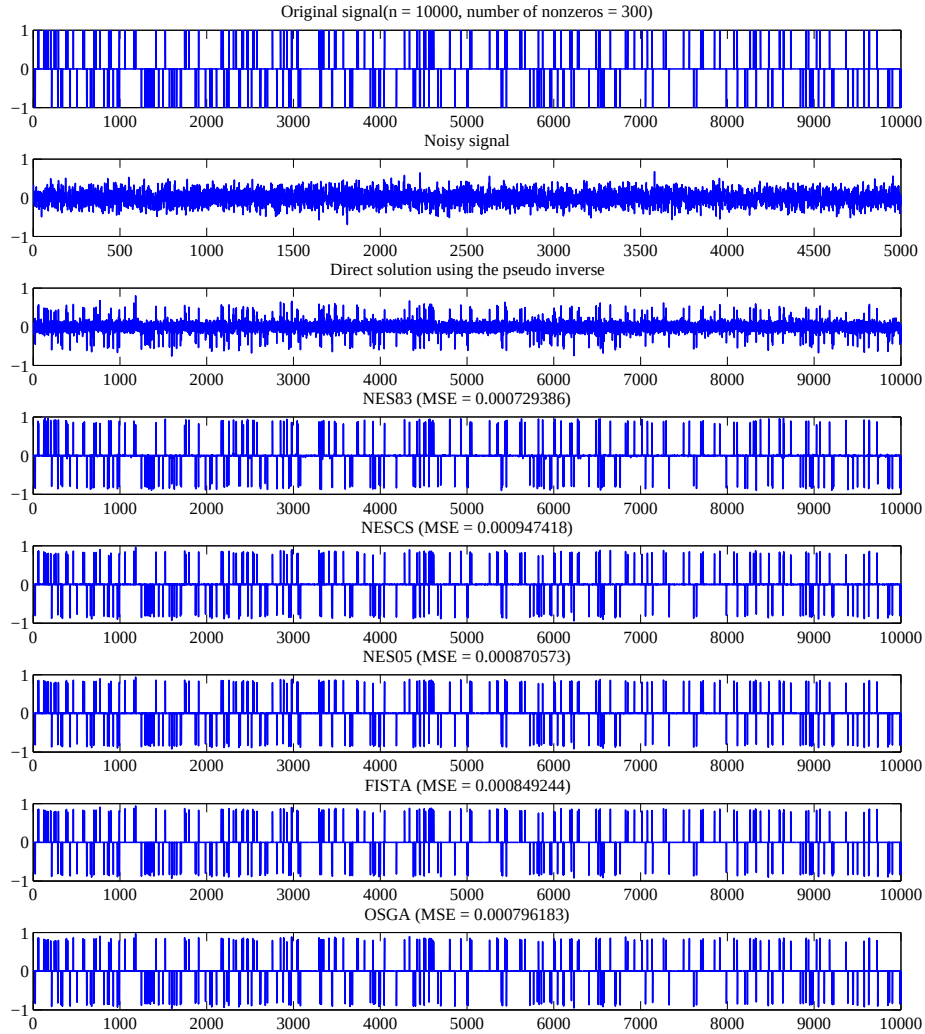


Figure 8.13: Recovering a noisy sparse signal using the $l_2^2 - l_1$ problem by NES83, NESCS, NES05, FISTA and OSGA for $\lambda_{\bar{a}} \|Ay\|_\infty \times 10^{-1}$. The algorithms were stopped after 20 seconds, and the quality of the signal is measured by MSE.

a usual measure for comparing the quality of a recovered signal defined by

$$\text{MSE} := \frac{1}{n} \|x - x_0\|_2^2, \quad (8.14)$$

in which x_0 is the original signal. The results of Figure 8.12 show that NSDSG, PGA, NESCO, and NESUN are unsuccessful to recover the signal accurately, where NESCO and NESUN could not leave the backtracking line searches in a reasonable number of inner iterations that dramatically decreases the step-sizes. In Figure 8.12, Subfigures (a) and (b) show that NSE83, NESCS, NES05, FISTA, and OSGA can restore the signal accurately, however, NES83 and OSGA are superior to the others regarding both function values and MSE. Comparing the results of Subfigures (c) and (d) with Subfigures (a) and (b) shows that the algorithms are sensitive to the regularization parameter λ , where NESCO and NESUN get better results for smaller regularization parameters. The results also demonstrate that NESCS, NES05, and FISTA are more sensitive than NES83 and OSGA to the regularization parameter. Indeed, they could not recover the signal accurately when the regularization parameter is small.

To show the results of our experiment visually, we illustrate the recovered signal of the experiment reported in Subfigures (a) and (b) of Figure 8.12 in Figure 8.13. Since NSDSG, PGA, NESCO, and NESUN could not recover the signal accurately, we do not consider them in Figure 8.13. We also add the direct solution $x = A^T(AA^T)^{-1}y$ to our comparison indicating that this solution is very inaccurate. The results show that NSE83, NESCS, NES05, FISTA, and OSGA recover the signal visually acceptable. Summarizing the results of Figures 8.12 and 8.13, it can be seen that NES83 and OSGA only need about 15 iterations to achieve an acceptable result, while NESCS, NES05, and FISTA require about 100 iterations to reach the same accuracy.

8.2 Problems involving costly linear operators

In this section we consider convex problems involving costly linear operators and apply OSGA, OSGA-S, and some subgradient schemes to solve them.

We consider an overdetermined system of equations of the form (4.24) and solve all 12 corresponding minimization problems presented in Example 4.2.1, where SGA-1, SGA-2 (two subgradient schemes, see [44]), OSGA, and OSGA-S. The problems are generated by

$$A = \text{rand}(m, n) - 0.5, \quad y = \text{rand}(m, 1) - 0.5, \quad x_0 = \text{rand}(n, 1) - 0.5,$$

where $m = 50000$ and $n = 5000$. SGA-1 and SGA-2 use the step-sizes

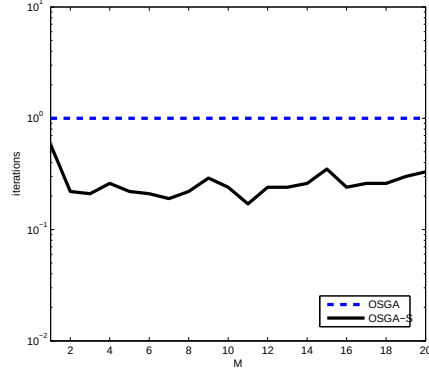
$$\alpha_k = \frac{\alpha_0}{\sqrt{k} \|g_k\|}, \quad \alpha_k = \frac{\alpha_0}{\sqrt{k}},$$

respectively, where $\alpha_0 = 8 \times 10^{-1}$ for SGA-1 and $\alpha_0 = 10^{-4}$ for SGA-2 applied to the problems L22R, L22L22R, L22L1R, L1R, L1L22R, and L1L1R and $\alpha_0 = 2 \times 10^{-2}$ for SGA-2 applied with the problems L2R, L2L22R, L2L1R, L1R, L1L22R, and L1L1R.

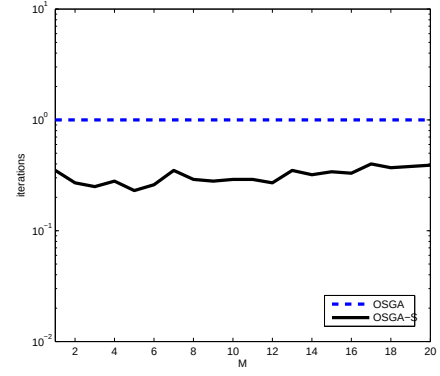
We first conduct an experiment on the parameter M to find an optimal range for this parameter. To this end, we consider the problems of Table 4.1, solve the problem by OSGA in 100 iterations, save the best function value in each case, and run OSGA-S with $M = 1, 2, \dots, 20$ to achieve f_s . The results of our experiment are summarized in Table 8.8, Figures 8.14, 8.15, 8.16, and 8.17. In Table 8.8, the best parameter M_{best} for each problem regarding the best number of iterations and the best running time, along with the results for M_{best} , $M = 2$ and $M = 20$, is reported. Figures 8.14, 8.15, 8.16, and 8.17 illustrate comparisons between OSGA and OSGA-S for $M = 1, 2, \dots, 20$, where Figures 8.14 and 8.15 show the results for the number of iterations and Figures 8.16 and 8.17 display the results for the running time. From the results of Table 8.8 and Figures 8.14, 8.15, 8.16, and 8.17, it can be observed that the best range of the parameter M is the interval $[1 \ 5]$. In addition, we can see that OSGA behaves better than OSGA-S for L1R, L1L22R, and L1L1R, however, OSGA-S outperforms OSGA for the others.

Table 8.8: Summary of numerical results for all 12 problems of Example 4.2.1, where M_{best} denotes the best value of the parameter $M \in \{1, 2, \dots, 20\}$, $N(M)$ the number of iterations needed to achieve the function value less or equal than f_s , and $T(M)$ the corresponding running time.

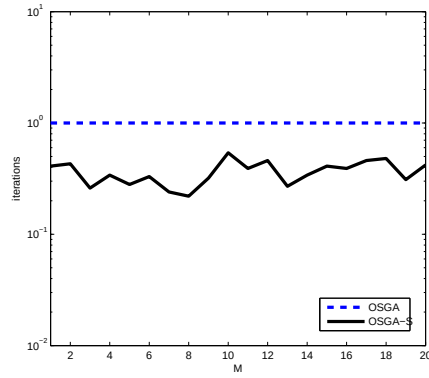
Problem name	Iteration				Time			
	M_{best}	$N(M_{best})$	$N(2)$	$N(20)$	M_{best}	$T(M_{best})$	$T(2)$	$T(20)$
L22R	11	17	22	33	7	8.31	11.32	25.64
L22L22R	8	22	43	42	8	10.18	16.88	27.82
L22L1R	2	14	14	20	4	6.92	7.41	10.22
L2R	5	23	27	39	5	9.76	11.22	23.74
L2L22R	1	18	19	39	1	6.94	7.62	18.61
L2L1R	1	27	36	31	1	10.56	14.72	73.12
L1R	1	93	94	103	1	36.90	40.80	73.66
L1L22R	3	95	100	104	3	40.22	41.77	82.40
L1L1R	19	59	85	114	2	35.46	35.46	65.66
L1R	1	2	2	93	1	0.67	0.69	3.57
L1L22R	10	32	64	88	10	16.60	26.33	61.95
L1L1R	1	3	8	92	1	1.12	3.19	67.75



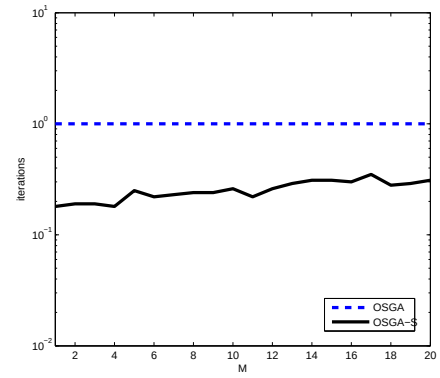
(a) L22R



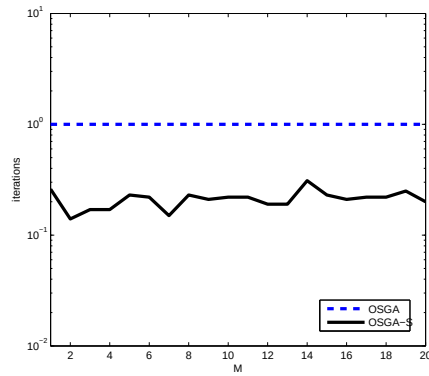
(b) L2R



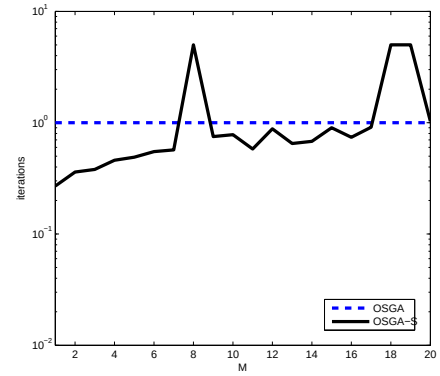
(c) L22L22R



(d) L2L22R



(e) L22L1R



(f) L2L1R

Figure 8.14: A comparison between OSGA and OSGA-S for the total number of iterations ($N(M)$) needed to solve the overdetermined systems of equations. Displayed is $N(M)/N_{OSGA}$ as a function of M .

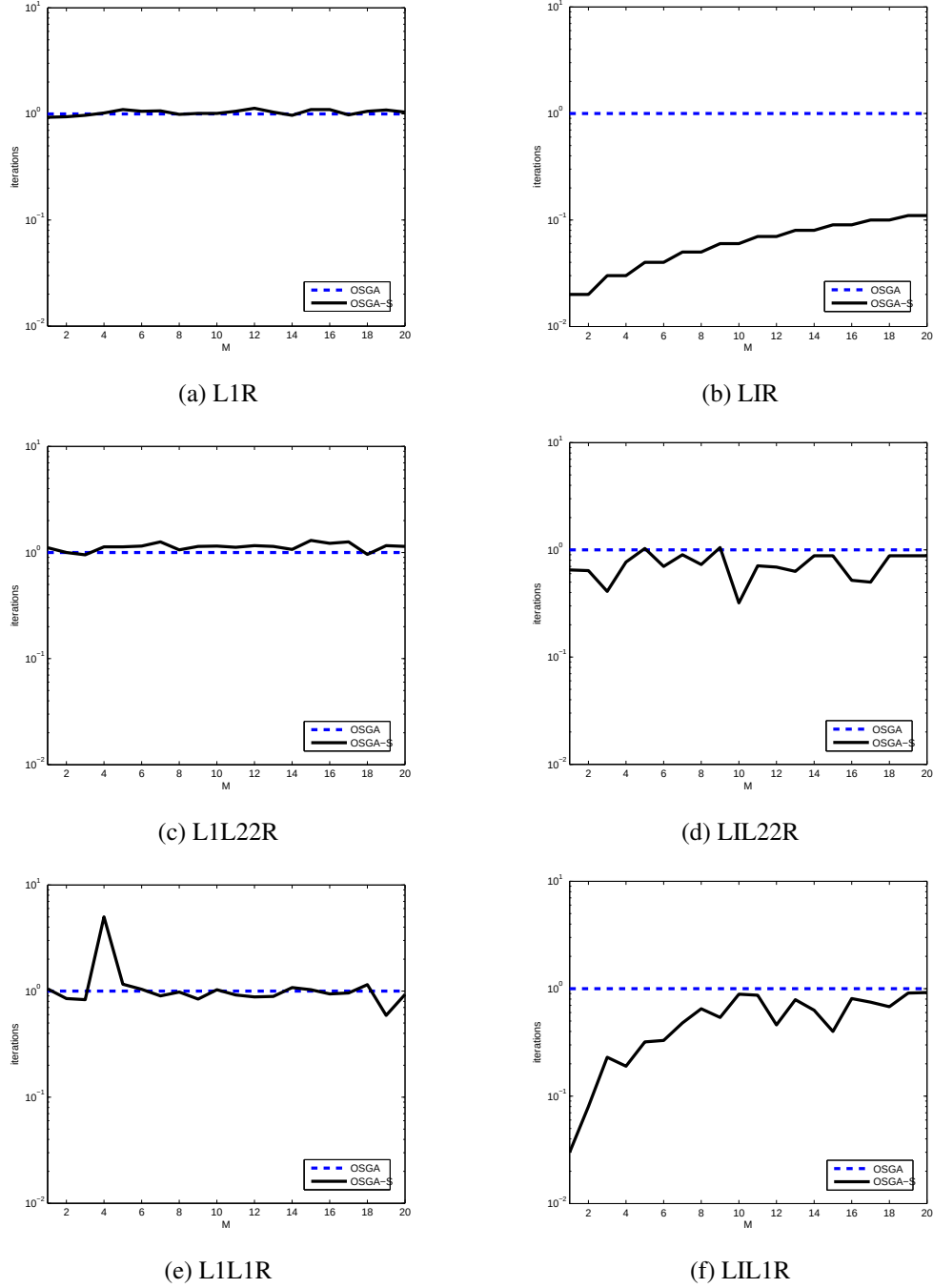


Figure 8.15: A comparison between OSGA and OSGA-S for the total number of iterations ($N(M)$) needed to solve the overdetermined systems of equations. Displayed is $N(M)/N_{OSGA}$ as a function of M .

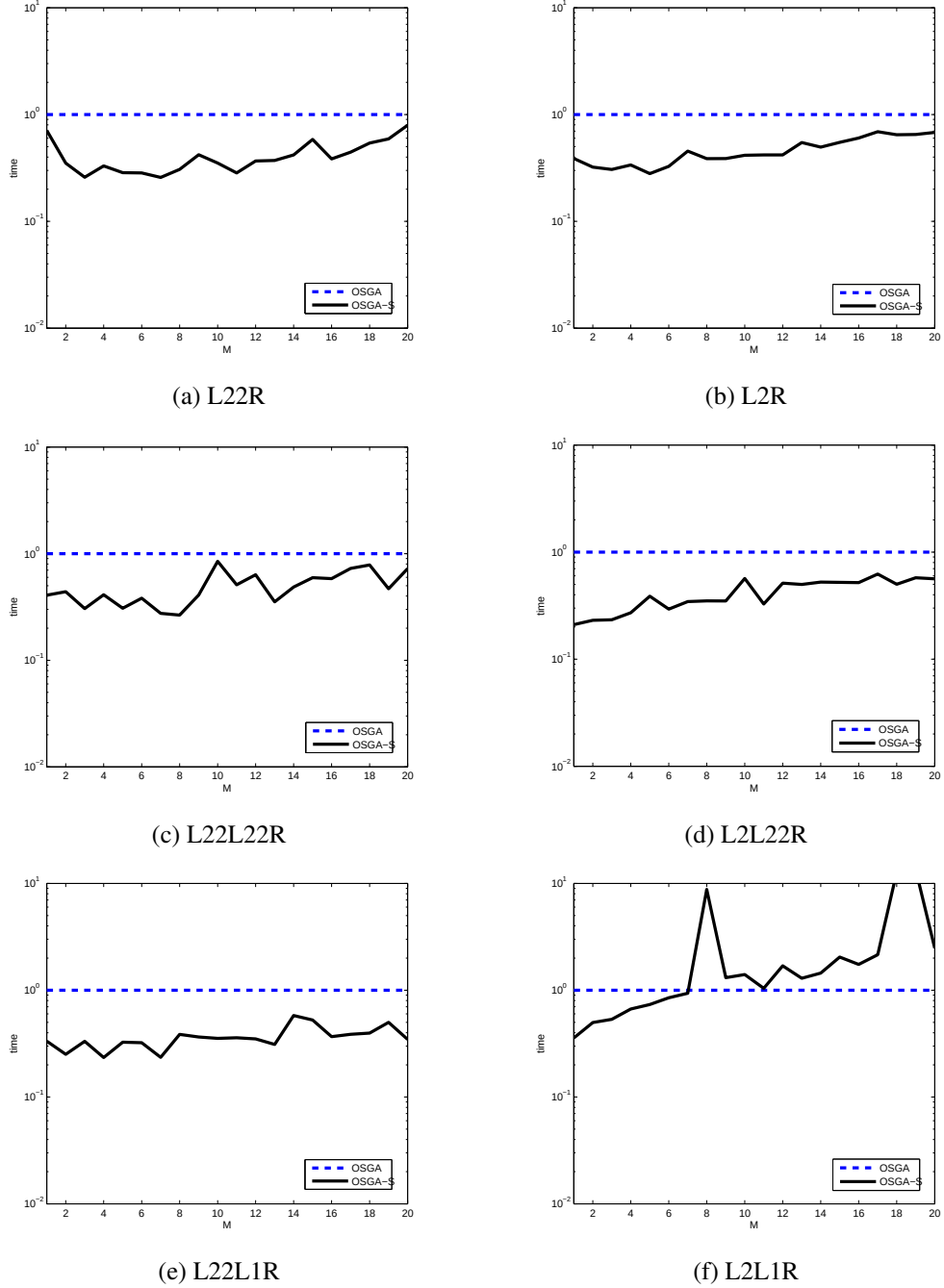
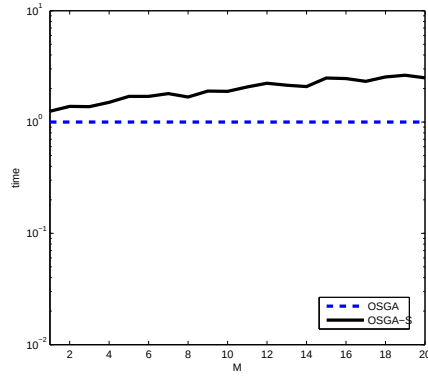
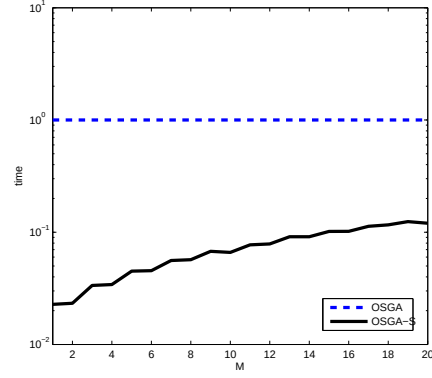


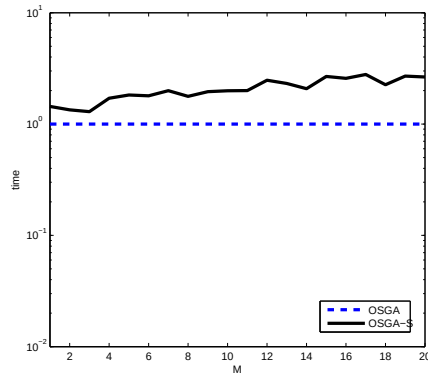
Figure 8.16: A comparison between OSGA and OSGA-S for the total number of iterations ($T(M)$) needed to solve the overdetermined systems of equations. Displayed is $T(M)/T_{OSGA}$ as a function of M .



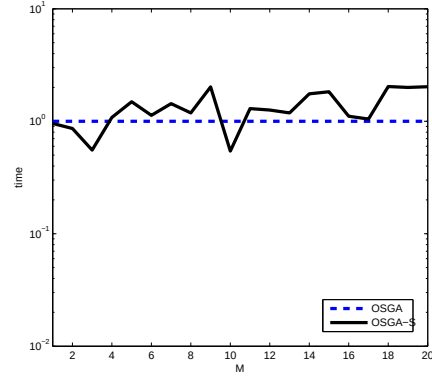
(a) L1R



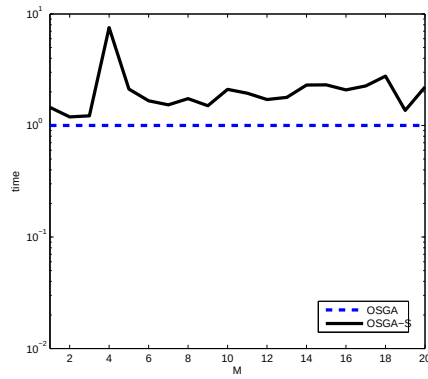
(b) LIR



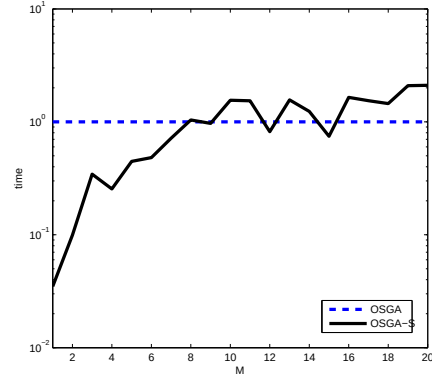
(c) L1L22R



(d) L1L22R



(e) L1L1R



(f) L1L1R

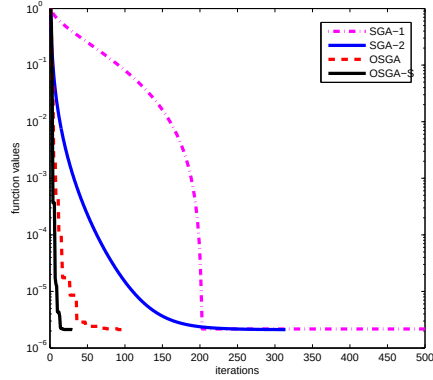
Figure 8.17: A comparison between OSGA and OSGA-S for the total number of iterations ($T(M)$) needed to solve the overdetermined systems of equations. Displayed is $T(M)/T_{OSGA}$ as a function of M .

We now solve the problems reported in Table 4.1 by SGA-1, SGA-2, OSGA, and OSGA-S, where we first solve these problems by OSGA in 100 iterations, save the best function value f_s and stop the others where they attain a function value less or equal than f_s or the number of iterations reaches to the maximum number of iterations, which is 500 here. We set with $M = 2$ for OSGA-S. The results of implementation are summarized in Table 8.9 and Figures 8.18 and 8.19.

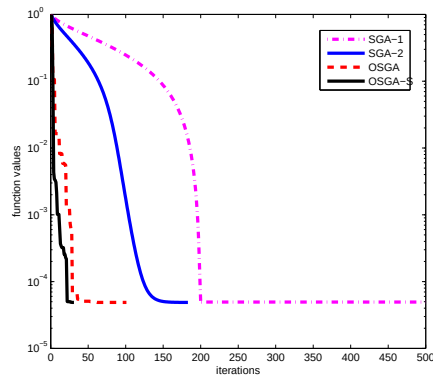
Table 8.9: Numerical results of the algorithms considered. In each problems, the best time is displayed as bold.

Problem name	SGA-1		SGA-2		OSGA		OSGA-S	
	N_i	T	N_i	T	N_i	T	N_i	T
L22R	500	151.04	314	89.90	100	43.73	29	15.90
L22L22R	500	131.81	398	115.42	100	35.65	39	20.76
L22L1R	500	128.36	246	64.97	100	35.04	13	6.77
L2R	500	138.10	183	57.17	100	40.24	30	19.73
L2L22R	500	156.11	139	43.22	100	47.17	18	11.57
L2L1R	500	135.02	500	139.22	100	36.87	42	25.67
L1R	500	137.13	364	114.28	100	36.87	100	62.24
L1L22R	500	133.16	252	58.60	100	46.91	64	38.51
L1L1R	500	150.47	244	74.00	100	33.27	64	33.62
L1R	72	15.10	120	27.36	100	34.14	3	1.16
L1L22R	500	145.78	500	125.46	100	45.82	23	14.11
L1L1R	485	98.19	161	34.56	100	36.26	45	27.74

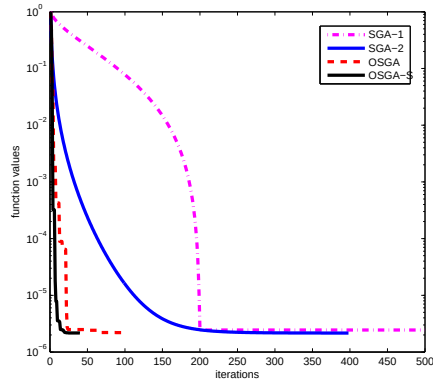
In Table 8.9, N_i and T denote the number of iterations and the running time, respectively. The results of Table 8.9 show that OSGA and OSGA-S outperform SGA-1 and SGA-2 significantly regarding both the number of iterations and the running time. However, OSGA-S needs fewer iterations and less running time than OSGA. Figures 8.18 and 8.19 illustrate the relative error of function values versus iterations (8.9), where f_0 , f_k , and f_* denote the function values at a starting point x_0 , the current point x_k , and the minimizer $\hat{x}$, respectively. The results of Figures 8.18 and 8.19 demonstrate that the best results are obtained by OSGA-S.



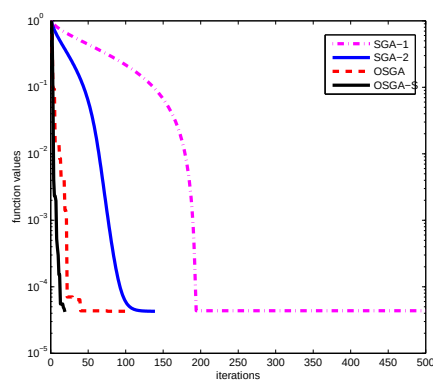
(a) L22R



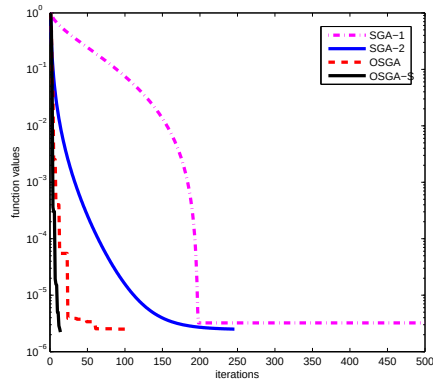
(b) L2R



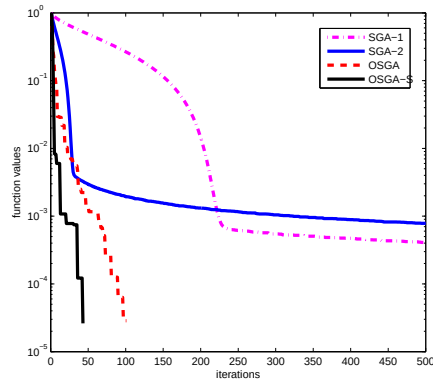
(c) L22L22R



(d) L2L22R



(e) L22L1R



(f) L2L1R

Figure 8.18: A comparison among SGA-1, SGA-2, OSGA, and OSGA-S for solving overdetermined systems of equations using the minimization problems presented in Example 4.2.1, where we give the relative error of function values versus iterations δ_2 .

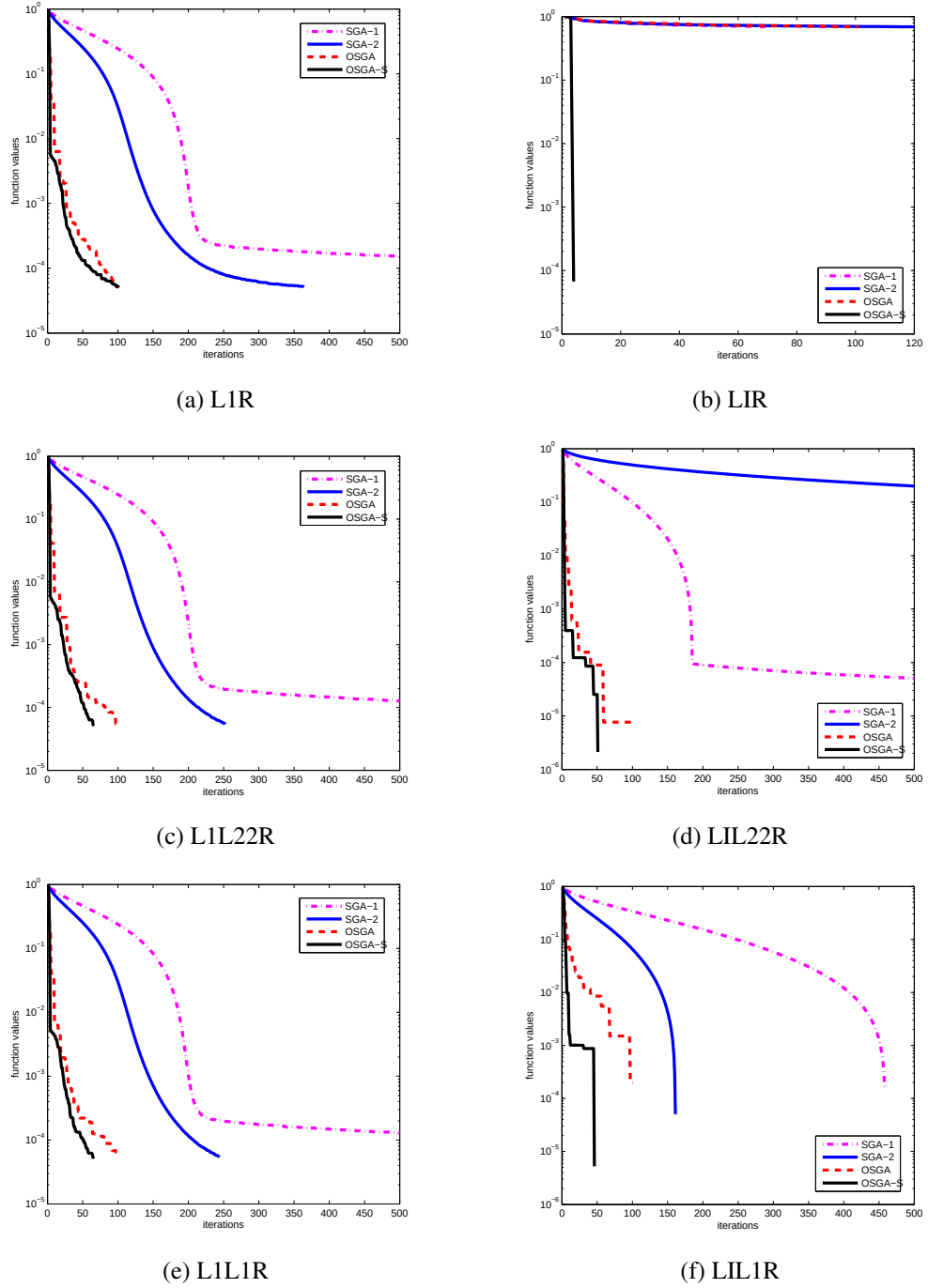


Figure 8.19: A comparison among SGA-1, SGA-2, OSGA, and OSGA-S for solving overdetermined systems of equations using the minimization problems presented in Example 4.2.1, where we give the relative error of function values versus iterations δ_2 .

Chapter 9

Convex optimization with simple constraints

In this chapter we consider practical problems of the form (2.8) with simple constraints (bound constraints, simple domains, simple functional constraints) and apply OSGA and some state-of-the-art schemes to such applications.

9.1 Convex optimization with simple constraints

This section devotes to considering some convex problems from applications with simple convex domains (orthogonal projection is assumed to be available) and applying OSGA and some state-of-the-art schemes to solve them.

9.1.1 Ridge regression

In this section we consider a ℓ_2 -constrained least squares of the form (5.24) (so-called ridge regression, see [95]) and report some numerical results.

The problem is generated by

$$[A, z, x] = \text{i_laplace}(n), \quad y = z + 0.1 * \text{rand},$$

where $n = 5000$ is the problem dimension and `i_laplace.m` is an ill-posed test problem generator using the inverse Laplace transformation from Regularization Tools MATLAB package, which is available in

<http://www.imm.dtu.dk/~pcha/Regutools/>.

Since (5.24) is smooth and the projection on $C = \{x \in \mathbb{R}^n \mid \|x\| \leq \xi\}$ is available (see Table 5.1), we employ gradient projection algorithm (PGA), the spectral gradient projection [38] with the GRIPPO et al. nonmonotone term [88] (SPG-G), the

spectral gradient projection with the AMINI et al. nonmonotone term [14] (SPG-A), and OSGA (see Proposition 5.1.11) to solve this minimization problem. The parameters of SPG-G and SPG-A are the same as those reported in the associated papers, but SPG-A uses

$$\eta_k = \begin{cases} \eta_0/2 & \text{if } k = 1, \\ (\eta_{k-1} + \eta_{k-2})/2 & \text{if } k \geq 2. \end{cases}$$

The algorithms are stopped after 500 iterations.

Table 9.1: Result summary for the ridge regression

	ξ	PGA	SPG-G	SPG-A	OSGA
f_b	10	101.70e-3	7.60e-3	6.41e-3	3.60e-3
Time(s)		77.78	30.08	31.20	22.09
f_b	15	48.23e-3	1.70e-3	1.31e-3	1.52e-3
Time(s)		66.54	25.00	24.24	21.55
f_b	20	23.08e-2	2.01e-2	1.74e-2	8.60e-3
Time(s)		64.60	28.47	27.11	21.40
f_b	25	23.00e-2	2.22e-2	1.24e-2	8.96e-3
Time(s)		62.55	30.20	31.18	26.50

In Table 9.1 we consider $\xi = 10, 15, 20, 25$ and report the best attained function values and the running time. The results imply that OSGA attains the best running time and except for $\xi = 15$ gives the best function values. To see the results of implementation in more details, we demonstrate the relative error δ_2 (8.9) of function values in Figure 9.1, where $\hat{f}$ denotes the minimum and f_0 shows the function value on an initial point x_0 .

9.1.2 Image deblurring with nonnegativity constraint

As discussed in Chapter 5, inverse problems are appearing in many fields of applied sciences and Engineering. This is particularly happen when researchers use digital images to record and analyze results from experiments in many fields such as astronomy, medical sciences, biology, geophysics, and physics. In these cases, observing blurred and noisy images is a common phenomenon happening frequently because of environmental effects and imperfections in the imaging system.

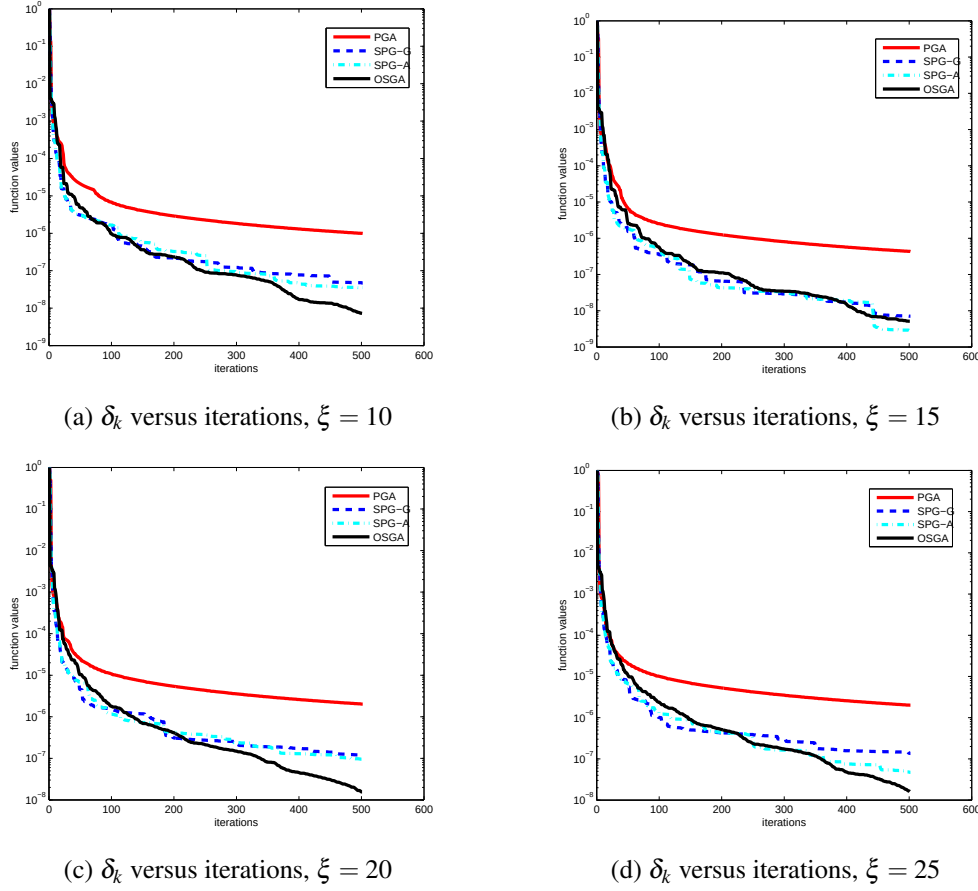


Figure 9.1: A comparison among PGA, SPG-G, SPG-A, and OSGA for solving ridge regression based on the relative error of function values δ_2 . The algorithms were stopped after 500 iterations.

In many applications, the variable x describes physical quantities, which is meaningful if each component of x is restricted to be nonnegative. This constraint is referred as the nonnegativity constraint; it is especially useful for restoring blurred and noisy images, see [20, 101, 102, 154].

We restore the 256×256 blurred and noisy MR-brain image using the model (5.2) equipped with the isotropic total variation regularizer. The true image is available in

<http://graphics.stanford.edu/data/voldata/>.

The blurred/noisy image y is generated by a 9×9 uniform blur and adding a Gaussian noise with zero mean and standard deviation set to 10^{-3} . For restoring the image, we use OSGA (see Proposition 5.1.7), MFISTA (a monotone version of

FISTA proposed by BECK & TEBoulLE in [27]), ADMM (an alternating direction method proposed by CHAN et al. in [56]), and PSGA (a projected subgradient scheme with nonsummable diminishing step size), see [44]. The original codes of MFISTA and ADMM provided by the authors are used. Since the methods are sensitive to the regularization parameter λ , three different regularization parameters are used. The algorithms are stopped after 100 iterations. The comparison concerning the quality of the recovered image is made via the so-called peak signal-to-noise ratio (PSNR) (8.11) and the improvement in signal-to-noise ratio (ISNR) (8.10), where $\|\cdot\|_F$ is the Frobenius norm, x_t denotes the $m \times n$ true image, y is the observed image, and pixel values are in $[0, 1]$. The results of implementation are summarized in Table 9.2 and Figures 9.2 and 9.3.

Table 9.2: Result summary for L22ITV

	λ	PSGA	MFISTA	ADMM	OSGA
PSNR		32.59	32.67	32.66	32.73
f_b	5×10^{-4}	0.3528	0.3079	0.3080	0.3149
Time(s)		1.14	7.61	1.11	1.82
PSNR		33.23	33.96	33.95	33.97
f_b	1×10^{-4}	0.1184	0.0960	0.0958	0.0980
Time(s)		1.14	7.34	1.04	1.71
PSNR		33.24	34.45	34.49	34.46
f_b	5×10^{-5}	0.1174	0.0653	0.0651	0.0669
Time(s)		1.15	6.51	1.06	1.67

In Table 9.2 we report PSNR, the best available approximation f_b of the minimum, and the running time in seconds for three different regularization parameters. The results reported in Figure 9.2 regarding function values and ISNR show that the algorithms considered are sensitive to the parameter λ , however, the best results obtained for $\lambda = 10^{-4}$. More specifically, the results about function values in subfigures (a), (c), and (e) demonstrate that OSGA outperforms PSGA, which means it performs much better than the lower complexity bound $O(\varepsilon^{-2})$, however, it cannot perform similar to MFISTA attaining the complexity $O(\varepsilon^{-1/2})$. Subfigures (b), (d), and (f) show that OSGA is comparable with MFISTA and ADMM and even better than them in the sense of ISNR. The deblurred images by the algorithms considered are illustrated in Figure 9.3 for $\lambda = 10^{-4}$.

We also consider the restoration of the 641×641 blurred/noisy Dione image using (5.3). The true image is available in

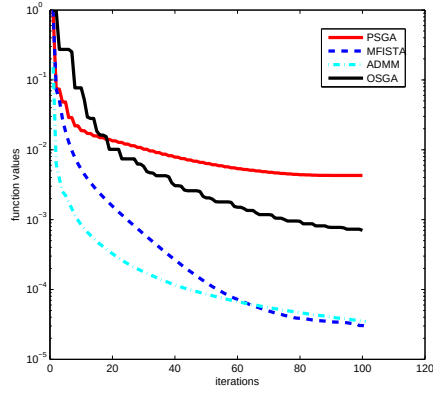
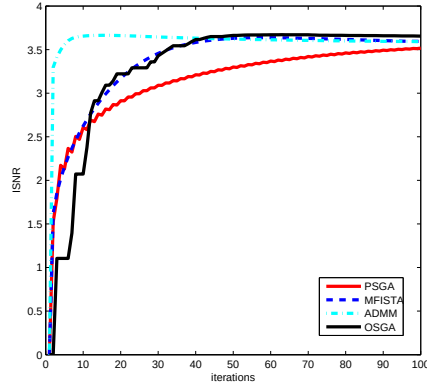
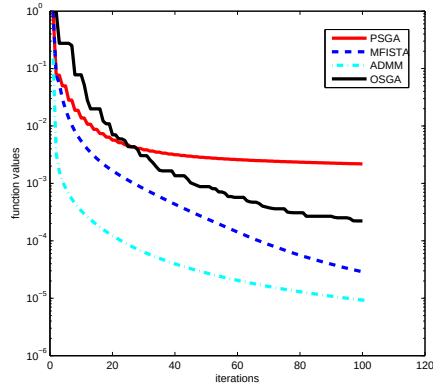
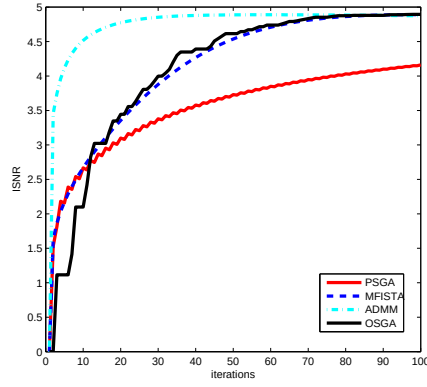
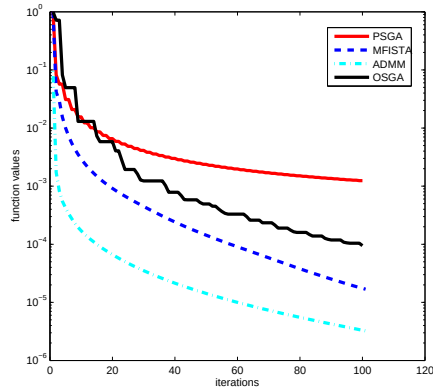
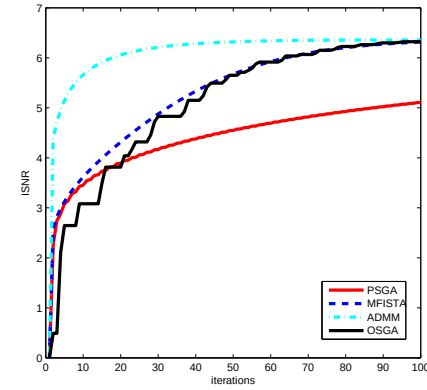
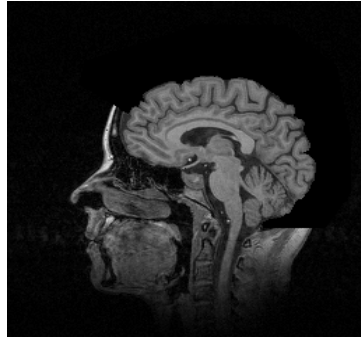
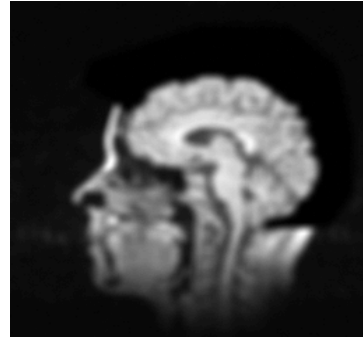
(a) δ_k versus iterations, $\lambda = 5 \times 10^{-4}$ (b) ISNR versus iterations, $\lambda = 5 \times 10^{-4}$ (c) δ_k versus iterations, $\lambda = 1 \times 10^{-4}$ (d) ISNR versus iterations, $\lambda = 1 \times 10^{-4}$ (e) δ_k versus iterations, $\lambda = 5 \times 10^{-5}$ (f) ISNR versus iterations, $\lambda = 5 \times 10^{-5}$

Figure 9.2: A comparison among PSGA, MFISTA, ADMM, and OSGA for de-blurring the 256×256 MR-brain image with the 9×9 uniform blur and the Gaussian noise with deviation 10^{-3} . The algorithms were stopped after 100 iterations. Subfigures (a), (c), and (e) display the relative error of function values δ_2 versus iterations, and Subfigures (b), (d), and (f) display ISNR versus iterations.



(a) Original image



(b) Blurred/noisy image

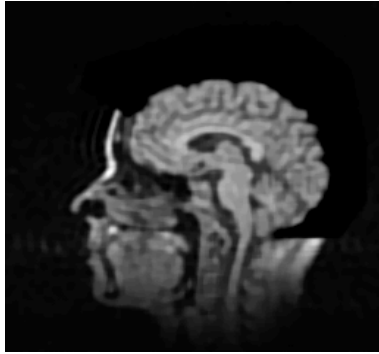
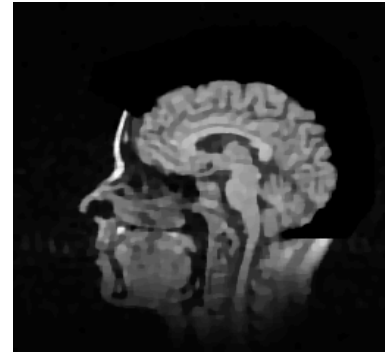
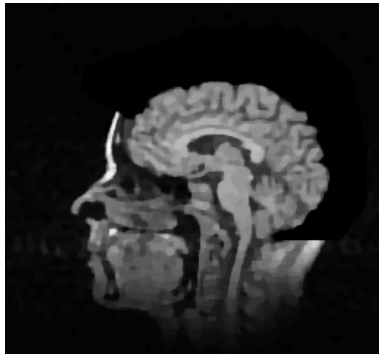
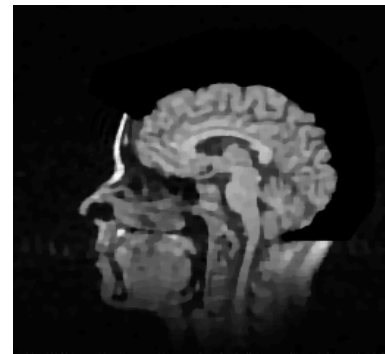
(c) PSGA: $f = 0.1174$, PSNR = 33.24, $T = 1.15$ (d) MFISTA: $f = 0.0653$, PSNR = 34.45, $T = 6.51$ (e) ADMM: $f = 0.0651$, PSNR = 34.49, $T = 1.06$ (f) OSGA: $f = 0.0669$, PSNR = 34.46, $T = 1.97$

Figure 9.3: Deblurring of the 256×256 MR-brain image with the 9×9 uniform blur and the Gaussian noise with deviation 10^{-3} by PSGA, MFISTA, ADMM, and OSGA with the regularization parameter $\lambda = 10^{-4}$. The algorithms were stopped after 100 iterations.

<http://photojournal.jpl.nasa.gov/Help/ImageGallery.html>.

The blurred/noisy image is constructed from the 7×7 Gaussian kernel with standard deviation 5 and salt-and-pepper impulsive noise with the level 50%. To recover the image, we use DRPD-1, DRPD-2 (Douglas-Rachford primal-dual schemes proposed by BOT&HENDRICH in [41]), ADMM, and OSGA. The algorithms are stopped after 100 iterations, and three different regularization parameters are considered. The results of implementation are reported in Table 9.3 and Figures 9.4 and 9.5.

The results of Table 9.3 shows that OSGA outperforms the others in the sense of PSNR. Figure 9.4 indicates that OSGA attains the best function values for $\lambda = 10^{-1}$ and $\lambda = 5 \times 10^{-2}$, however, ADMM get the best function value for $\lambda = 5 \times 10^{-1}$. It also implies that OSGA are comparable or even better than the others regarding ISNR. The resulted images for $\lambda = 10^{-1}$ are illustrated in Figure 9.5, demonstrating that the algorithms can restore the image by acceptable qualities while OSGA obtains the best function value and PSNR.

Table 9.3: Results summary for L1ITV

	λ	DRPD-1	DRPD-2	ADMM	OSGA
PSNR	5×10^{-1}	37.43	36.66	37.42	37.50
f_b		1.0352e+5	1.0365e+5	1.0293e+5	1.0326e+5
Time		10.86	6.83	8.57	9.01
PSNR	1×10^{-1}	38.70	38.11	38.35	38.73
f_b		1.0324e+5	1.0294e+5	1.0281e+5	1.0281e+5
Time		10.43	6.68	8.46	8.32
PSNR	5×10^{-2}	37.09	36.77	30.06	37.06
f_b		1.0336e+5	1.0321e+5	1.0312e+5	1.0299e+5
Time		10.26	6.27	8.25	9.23

9.2 Bound-constrained convex optimization problems

In this section we consider some applications with bound-constrained domains and apply OSGA and several state-of-the-art schemes to solve the problem. The results involve giving numerical results and comparisons.

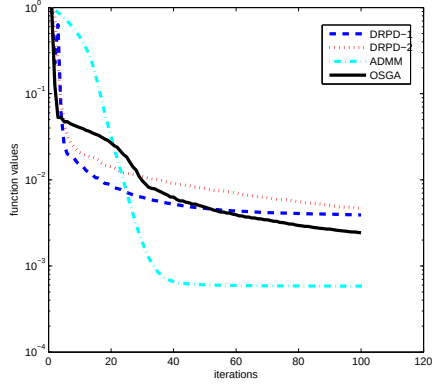
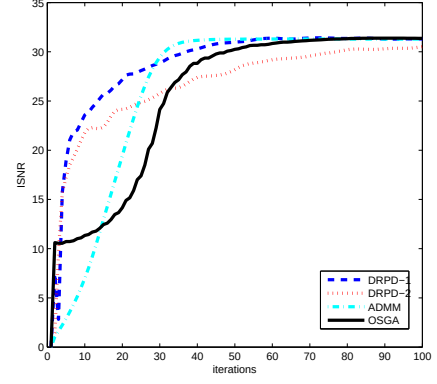
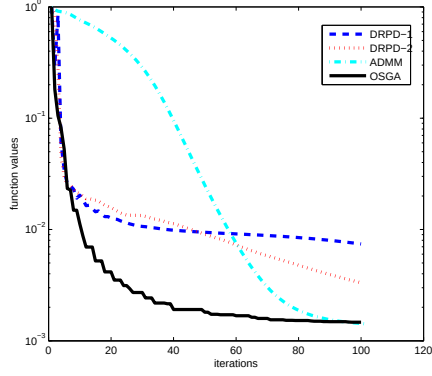
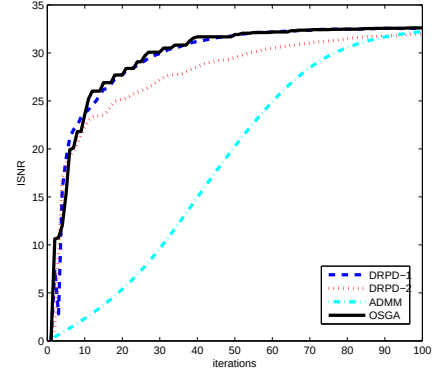
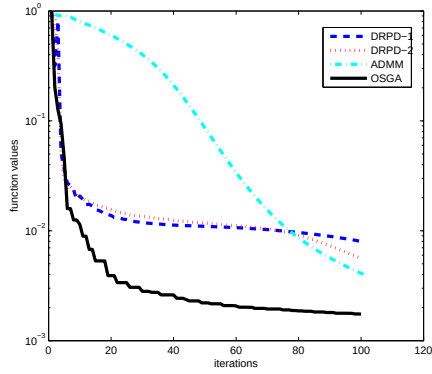
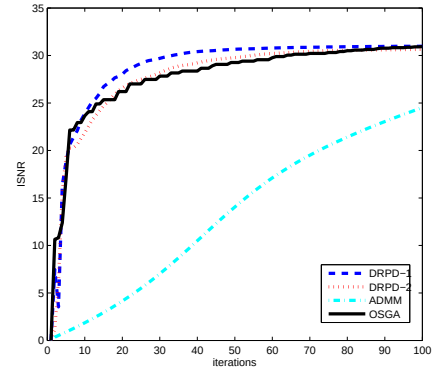
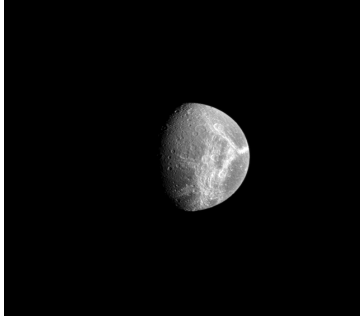
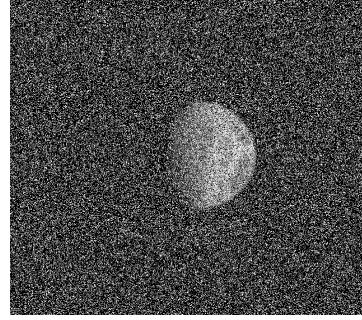
(a) δ_k versus iterations, $\lambda = 5 \times 10^{-1}$ (b) ISNR versus iterations, $\lambda = 5 \times 10^{-1}$ (c) δ_k versus iterations, $\lambda = 1 \times 10^{-1}$ (d) ISNR versus iterations, $\lambda = 1 \times 10^{-1}$ (e) δ_k versus iterations, $\lambda = 5 \times 10^{-2}$ (f) ISNR versus iterations, $\lambda = 5 \times 10^{-2}$

Figure 9.4: A comparison among DRPD-1, DRPD-2, ADMM, and OSGA for deblurring the 641×641 Dione image with the various regularization parameter λ . The blurred/noisy image was constructed by the 7×7 Gaussian kernel with standard deviation 5 and salt-and-pepper impulsive noise with the level 50%. The algorithms were stopped after 100 iterations. Subfigures (a), (c), and (e) display the relative error of function values δ_2 versus iterations, and (b), (d), and (f) demonstrate ISNR versus iterations. 150



(a) Original image



(b) Blurred/noisy image

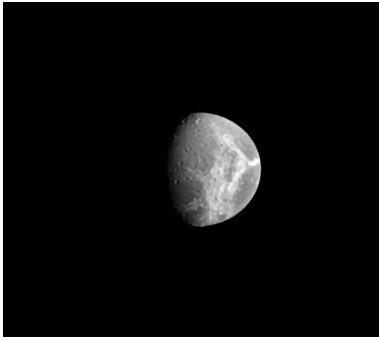
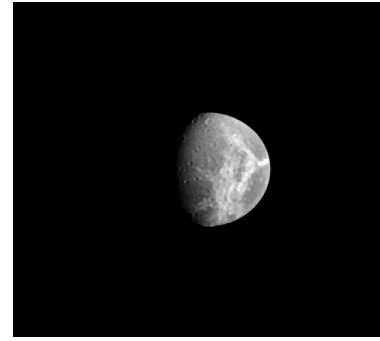
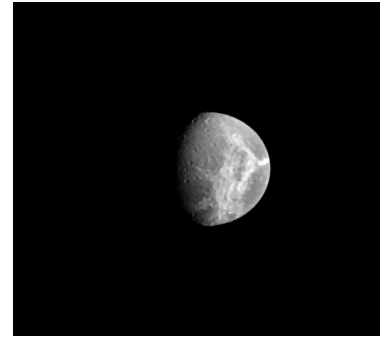
(c) DRPD-1: $f = 1.0324e + 5$, PSNR = 38.70, T = 10.43(d) DRPD-2: $f = 1.0294e + 5$, PSNR = 38.11, T = 6.68(e) ADMM: $f = 1.0281e + 5$, PSNR = 38.35, T = 8.46(f) OSGA: $f = 1.0281e + 5$, PSNR = 38.73, T = 8.32

Figure 9.5: Deblurring of the 641×641 Dione image using DRPD-1, DRPD-2, ADMM and OSGA with the parameter $\lambda = 10^{-1}$. The algorithms were stopped after 100 iterations. The blurred/noisy image was constructed by the 7×7 Gaussian kernel with standard deviation 5 and salt-and-pepper impulsive noise with the level 50%.

9.2.1 Experiment with artificial data

In this section we deal with solving the problem (5.33) with the objective functions

$$\begin{aligned}
 f(x) &= \frac{1}{2}\|Ax - b\|_2^2 + \frac{1}{2}\|x\|_2^2 & (L22L22R), \\
 f(x) &= \frac{1}{2}\|Ax - b\|_2^2 + \|x\|_1 & (L22L1R), \\
 f(x) &= \|Ax - b\|_1 + \frac{1}{2}\|x\|_2^2 & (L1L22R), \\
 f(x) &= \|Ax - b\|_1 + \|x\|_1 & (L1L1R).
 \end{aligned} \tag{9.1}$$

The problem is generated by

$$[A, z, x] = \text{i_laplace}(n), \quad b = z + 0.1 * \text{rand},$$

where n is the problem dimension and `i_laplace.m` is a code generating an ill-posed test problem using the inverse Laplace transformation from the Regularization Tools MATLAB package, which is available at

$$\text{http://www.imm.dtu.dk/~pcha/Regutools/}.$$

The upper and lower bounds on variables are generated by

$$\underline{x} = 0.05 * \text{ones}(n), \quad \bar{x} = 0.95 * \text{ones}(n),$$

respectively. Since among the problems (9.1) only L22L22R is differentiable, we need some nonsmooth algorithms to be compared with OSGA. In our experiment we consider two versions of OSGA, i.e., a version uses BCSS for solving the subproblem (3.4) (OSGA-1) and a version uses the inexact solution described in section 3.2 for solving the subproblem (3.4) (OSGA-2), compared with PSGA-1 (a projected subgradient algorithm with nonsummable diminishing step-size), and PSGA-2 (a projected subgradient algorithm with nonsummable diminishing steplength), see Section 2.4 of Chapter 2.

We solve all of the above-mentioned problems with the dimensions $n = 2000$ and $n = 5000$. The results for L22L22R and L22L1R are illustrated in Table 9.4 and Figure 9.6, and the results for L1L22R and L1L1R are summarized in Table 9.5 and Figure 9.7. More precisely, Figures 9.6 and 9.7 show the relative error δ_2 (8.9) of function vales versus iterations, where $\hat{f}$ denotes the minimum and f_0 shows the function value on an initial point x_0 . In our experiments, PSGA-1 and PSGA-2 exploit the step-sizes $\alpha := 5/\sqrt{k}\|g_k\|$ and $\alpha := 0.1/\sqrt{k}$, respectively, in which k is the iteration counter and g_k is a subgradient of f at x_k . The algorithms are stopped after 100 iterations.

Table 9.4: Result summary for L22L22R and L22L1R

	problem	dimension	PSGA-1	PSGA-2	OSGA-1	OSGA-2
f_b	L22L22R	$n = 2000$	81.8427	77.1302	77.1285	77.1285
Time			0.74	0.75	4.15	3.08
f_b	L22L22R	$n = 5000$	4.7561e+2	4.2646e+2	4.2640e+2	4.2645e+2
Time			3.67	3.58	14.09	7.57
f_b	L22L1R	$n = 2000$	1.8922e+2	1.8827e+2	1.8682e+2	1.2367e+2
Time			0.67	0.61	3.91	1.21
f_b	L22L1R	$n = 5000$	7.0679e+2	6.8084e+2	6.7887e+2	6.8064e+2
Time			3.72	3.42	14.20	7.61

Table 9.5: Result summary for L1L22R and L1L1R

	problem	dimension	PSGA-1	PSGA-2	OSGA-1	OSGA-2
f_b	L1L22R	$n = 2000$	1.8981e+2	1.9420e+2	1.8671e+2	1.8676e+2
Time			0.69	0.75	4.04	2.73
f_b	L1L22R	$n = 5000$	1.9256e+2	3.4612e+2	1.6971e+2	1.6995e+2
Time			3.69	3.57	14.06	6.83
f_b	L1L1R	$n = 2000$	2.6713e+2	2.6936e+2	2.6536e+2	2.6703e+2
Time			0.76	0.75	4.27	3.02
f_b	L1L1R	$n = 5000$	6.9728e+2	7.4536e+2	6.9411e+2	6.9687e+2
Time			3.68	3.57	14.46	7.46

In Figure 9.6, Subfigures (a) and (b) show that OSOA-1 and OSOA-2 substantially outperform PSGA-1 and PSGA-2 with respect to the relative error of function values δ_2 (8.9), however, they need more running time. In this case OSOA-1 and OSOA-2 are competitive, while OSOA-1 performs better. In Figure 9.6, Subfigure (c) shows that OSOA-2 produces the best results and the others are competitive. Subfigure (d) of Figure 9.6 demonstrates that OSOA-1 attains the best results and SPGA-2 and OSOA-2 are competitive but much better than SPGA-1. In Figure 9.7, Subfigures (a) and (b) show that OSOA-1 and OSOA-2 are comparable but much better than PSGA-1 and PSGA-2. Subfigures (c) and (s) show that the best results produced by OSOA-1 and OSOA-2, respectively.

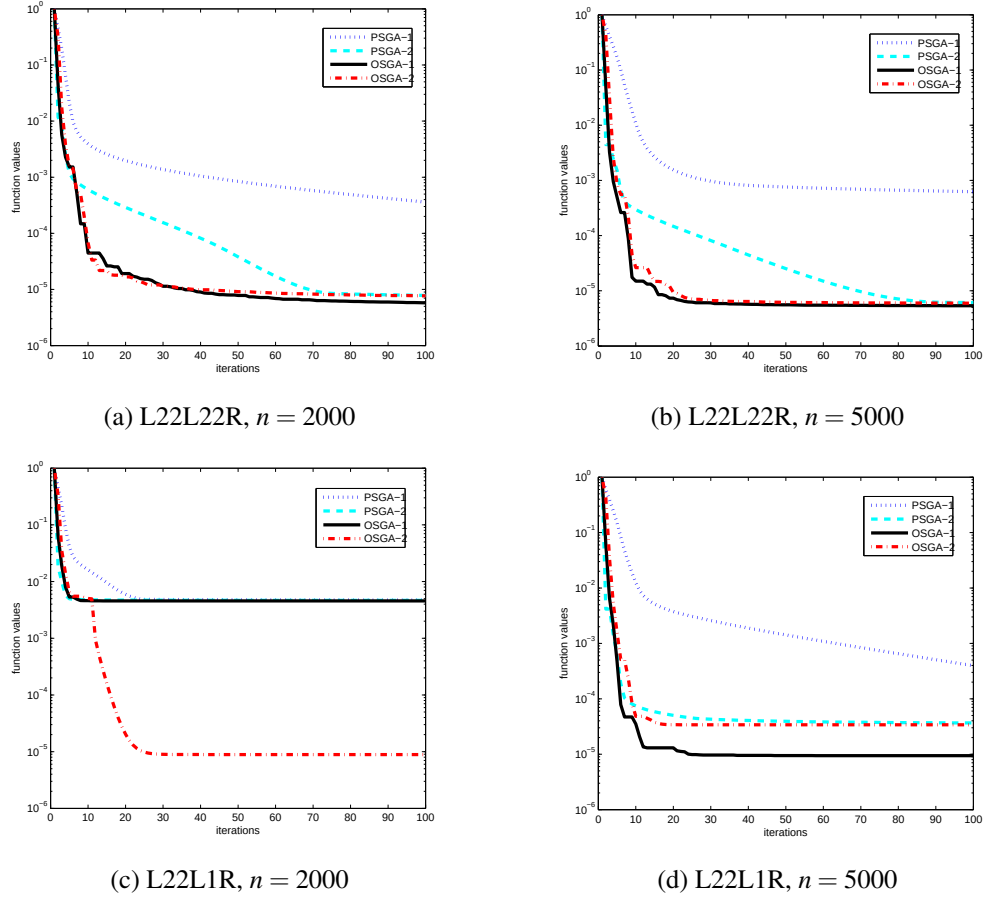


Figure 9.6: A comparison among PSGA-1, PSGA-2, OSGA-1, and OSGA-2 for solving L22L22R and L22L1R, where the algorithms were stopped after 100 iterations.

9.2.2 Image deblurring/denoising

As discussed in Section 8.1.1, image deblurring/denoising is one of the fundamental tasks in the context of digital imaging processing, aiming at recovering an image from a blurred/noisy observation. The problem is typically modeled as linear inverse problem

$$y = Ax + \omega, \quad x \in V, \quad (9.2)$$

where V is a finite-dimensional vector space, A is a blurring linear operator, x is a clean image, y is an observation, and ω is either Gaussian or impulsive noise.

The system of equation (9.2) is usually underdetermined and ill-conditioned, and ω is not commonly available, so it is not possible to solve it directly, see [127]. Hence the solution is generally approximated by an optimization problem

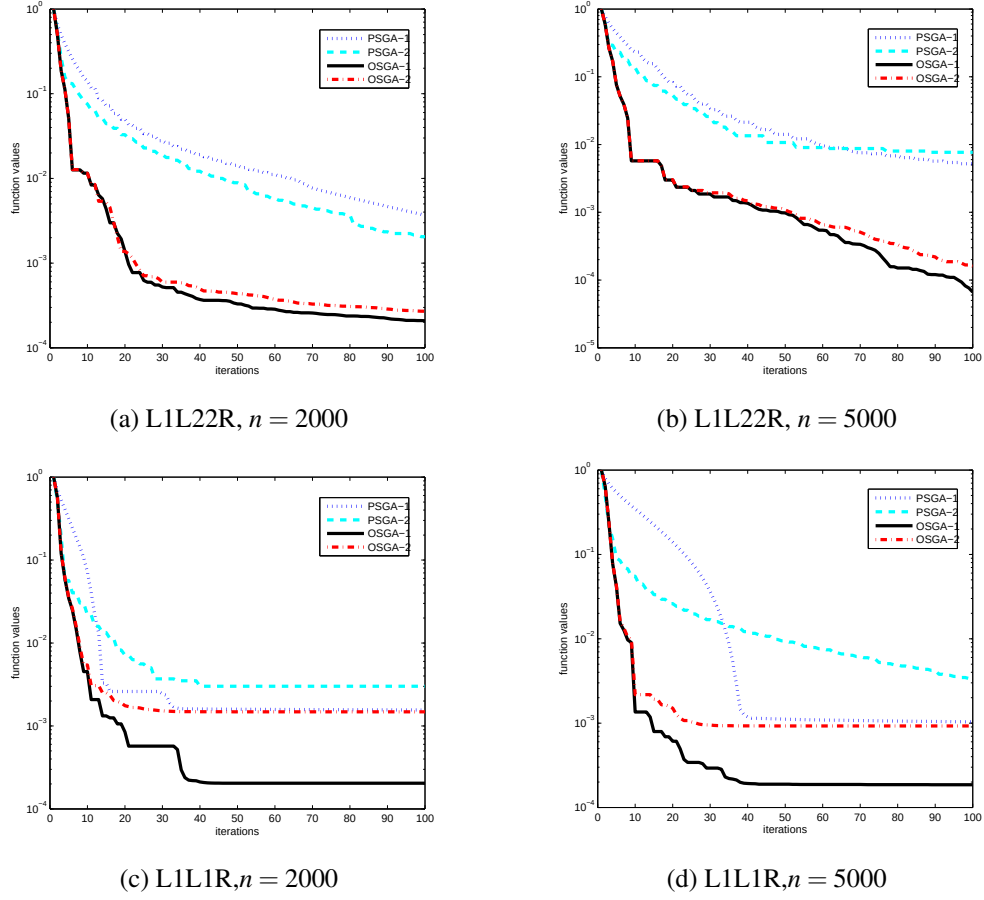


Figure 9.7: A comparison among PSGA-1, PSGA-2, OSGA-1, and OSGA-2 for solving L1L22R and L1L1R, where the algorithms were stopped after 100 iterations.

of the form

$$\begin{aligned} \min \quad & \frac{1}{2} \|Ax - b\|_2^2 + \lambda \varphi(x) \\ \text{s.t.} \quad & x \in V, \end{aligned} \quad (9.3)$$

or

$$\begin{aligned} \min \quad & \|Ax - b\|_1 + \lambda \varphi(x) \\ \text{s.t.} \quad & x \in V, \end{aligned} \quad (9.4)$$

where φ is a smooth or nonsmooth regularizer such as $\varphi(x) = \frac{1}{2} \|x\|_2^2$, $\varphi(x) = \|x\|_1$, $\varphi(x) = \|x\|_{ITV}$, and $\varphi(x) = \|x\|_{ATV}$. Among the various regularizers, the total variation is much more popular due to its strong edge preserving feature. Two

important types of the total variation, namely isotropic and anisotropic, see [54], are defined for $x \in \mathbb{R}^{m \times n}$ in (8.5) and (8.6), respectively.

The common drawback of the unconstrained problem (9.3) is that it usually gives a solution outside of the dynamic range of the image, which is either $[0, 1]$ or $[0, 255]$ for 8-bit gray-scale images. Hence one has to project the unconstrained solution to the dynamic range of the image. However, the quality of the projected images is not always acceptable. As a result, it is worth to solve a bound-constrained problem of the form (5.34) in place of the unconstrained problem (9.3), where the bounds are defined by the dynamic range of the images, see [27, 56, 157].

The comparison concerning the quality of the recovered image is made via the so-called peak signal-to-noise ratio (PSNR) (8.11) and the improvement in signal-to-noise ratio (ISNR) (8.10) where pixel values are in $[0, 1]$.

9.2.2.1 Experiment with ℓ_2^2 isotropic total variation

We here consider the image restoration from a blurred/noisy observation using the model (5.34) equipped with the isotropic total variation regularizer. We employ OSGA, MFISTA (a monotone version of FISTA proposed by BECK & TEOULLE in [28]), ADMM (an alternating direction method proposed by CHAN et al. in [56]), and a projected subgradient algorithms PSGA (with nonsummable diminishing step-size, see [44]). In our implementation we use the original code of MFISTA and ADMM provided by the authors, with minor adaptations to solve the problem form (5.34) and to stop in a fixed number of iterations.

We restore the 512×512 blurred/noisy Barbara image. Let y be a blurred/noisy version of this image generated by a 9×9 uniform blur and adding a Gaussian noise with zero mean and standard deviation set to $10^{-3/2}$. Our implementation shows that the algorithms are sensitive to the regularization parameter λ . Hence we consider three different regularization parameters $\lambda = 1 \times 10^{-2}$, $\lambda = 7 \times 10^{-3}$, and $\lambda = 4 \times 10^{-3}$. All algorithms are stopped after 50 iterations. The results of our implementation are summarized in Table 9.6 and Figures 9.8 and 9.9.

The results of Table 9.6 and Figure 9.8 show that function values, PSNR, and ISNR produced by the algorithms are sensitive to the regularization parameter λ . However, the function values are less sensitive. Subfigures (a), (c), and (e) show that OSGA gives the best performance in terms of function values. According to Subfigures (b), (d), and (f), the best ISNR is attained for $\lambda = 4 \times 10^{-3}$, and the algorithms are comparable with each other, but OSGA outperforms the others slightly. Figure 9.9 illustrates the resulting deblurred images for $\lambda = 4 \times 10^{-3}$.

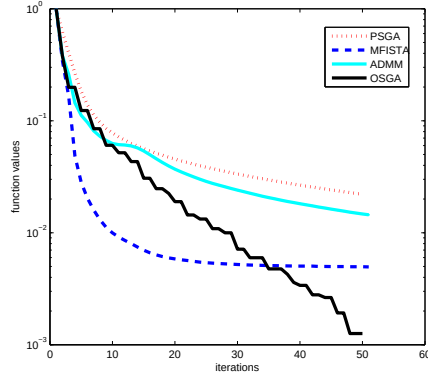
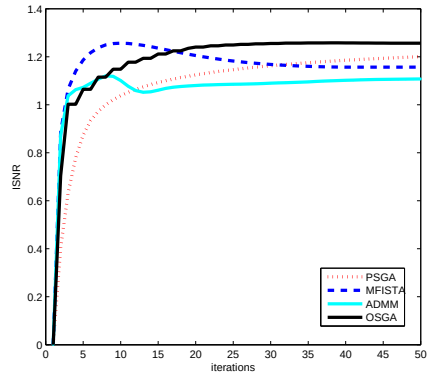
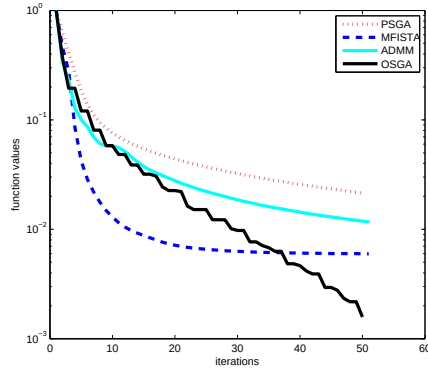
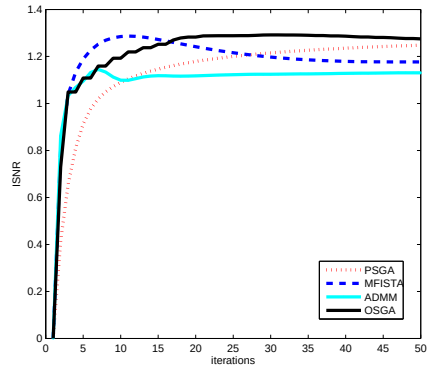
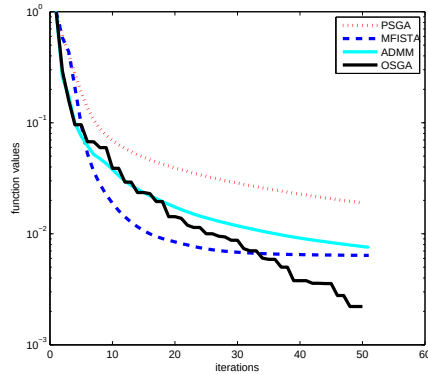
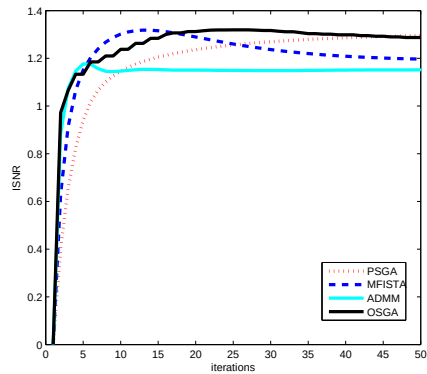
(a) δ_k versus iterations, $\lambda = 1 \times 10^{-3}$ (b) ISNR versus iterations, $\lambda = 1 \times 10^{-3}$ (c) δ_k versus iterations, $\lambda = 6 \times 10^{-4}$ (d) ISNR versus iterations, $\lambda = 6 \times 10^{-4}$ (e) δ_k versus iterations, $\lambda = 4 \times 10^{-4}$ (f) ISNR versus iterations, $\lambda = 4 \times 10^{-4}$

Figure 9.8: A comparison among PSGA, MFISTA, ADMM, and OSGA for deblurring the 512×512 Barbara image with the 9×9 uniform blur and the Gaussian noise with deviation $10^{-3/2}$. The algorithms were stopped after 50 iterations. Subfigures (a), (c) and (e) display the relative error δ_2 of function values versus iterations, and (b), (d), and (f) show ISNR versus iterations.



(a) Original image



(b) Blurred/noisy image

(c) PSGA: $f = 1.4402e + 2$, PSNR = 23.77, T = 2.09(d) MFISTA: $f = 1.4321e + 2$, PSNR = 23.67, T = 31.93(e) ADMM: $f = 1.4329e + 2$, PSNR = 23.63, T = 2.06(f) OSGA: $f = 1.4294e + 2$, PSNR = 23.77, T = 5.49

Figure 9.9: Deblurring of the 512×512 Barbara image with the 9×9 uniform blur and the Gaussian noise with deviation $10^{-3/2}$ by PSGA, MFISTA, ADMM, and OSGA with the regularization parameter $\lambda = 4 \times 10^{-3}$. The algorithms were stopped after 50 iterations.

Table 9.6: Result summary for the ℓ_2^2 isotropic total variation

	λ	PSGA	MFISTA	ADMM	OSGA
PSNR	1×10^{-2}	23.69	23.64	23.59	23.74
f_b		1.6804e+2	1.6580e+2	1.6705e+2	1.6531e+2
Time		2.15	30.55	2.05	5.60
PSNR	7×10^{-3}	23.73	23.66	23.67	23.76
f_b		1.5694e+2	1.5543e+2	1.5599e+2	1.5500e+2
Time		2.09	31.31	2.12	5.39
PSNR	4×10^{-3}	23.77	23.67	23.63	23.77
f_b		1.4402e+2	1.4321e+2	1.4329e+2	1.4294e+2
Time		2.09	31.93	2.06	5.49

9.2.2.2 Experiment with ℓ_1 isotropic total variation

In this section we study the image restoration from a blurred/noisy observation using the model (5.35) equipped with the isotropic total variation regularizer. The optimization problem is solved by DRPD1, DRPD2 (Douglas-Rachford primal-dual algorithms proposed by BOT & HENDRICH in [41]), ADMM, and OSGA.

Here we consider recovering the 256×256 blurred/noisy Fingerprint image from a blurred/noisy image constructed by a 7×7 Gaussian kernel with standard deviation 5 and salt-and-pepper impulsive noise with the level 40%. The algorithms are stopped after 50 iterations. Three different regularization parameters $\lambda = 3 \times 10^{-1}$, $\lambda = 1 \times 10^{-1}$, and $\lambda = 8 \times 10^{-2}$ are considered. The results are presented in Table 9.7 and Figures 9.10 and 9.11.

Table 9.7: Results for the ℓ_1 isotropic total variation

	λ	DRPD-1	DRPD-2	ADMM	OSGA
PSNR	3×10^{-1}	17.95	17.61	18.47	18.67
f_b		1.4581e+4	1.4867e+4	1.4476e+4	1.4564e+4
Time		1.04	0.67	0.81	6.11
PSNR	1×10^{-1}	21.87	21.23	20.10	22.05
f_b		1.3571e+4	1.3635e+4	1.3799e+4	1.3612e+4
Time		1.28	0.83	0.76	6.01
PSNR	8×10^{-2}	22.26	21.56	18.67	22.46
f_b		1.3519e+4	1.3573e+4	1.3879e+4	1.3582e+4
Time		1.00	0.65	0.79	6.11

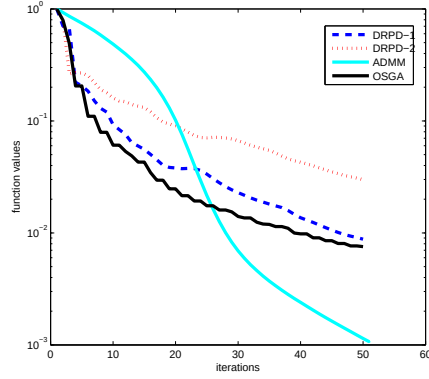
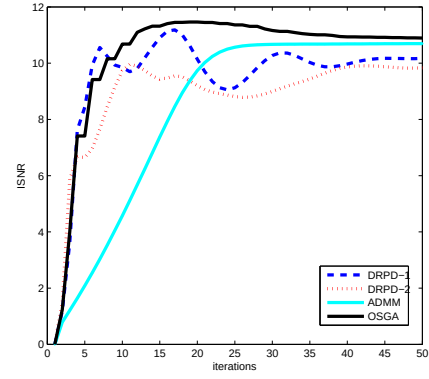
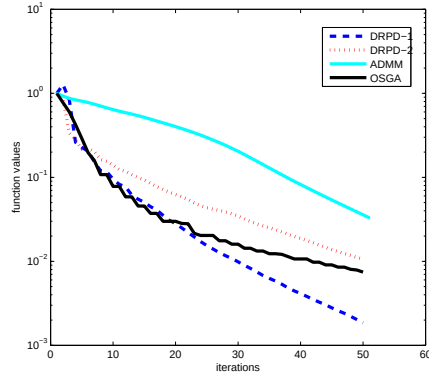
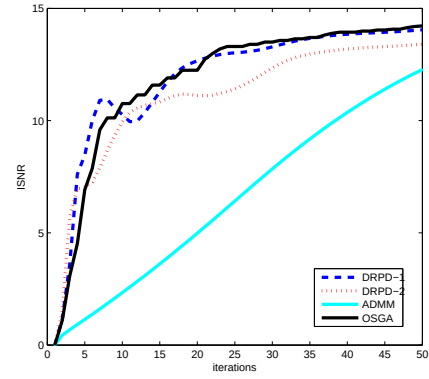
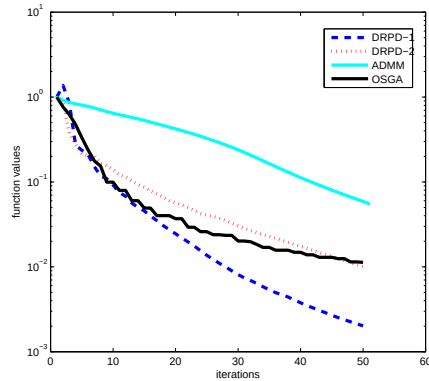
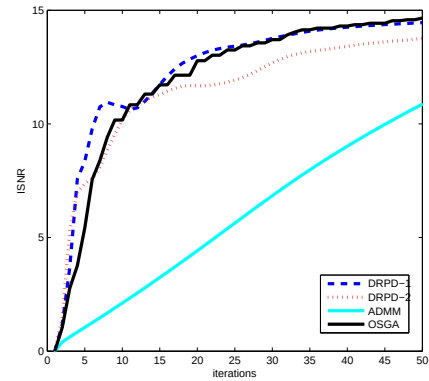
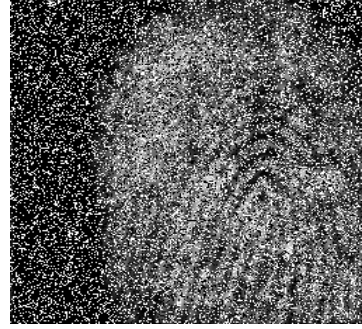
(a) δ_k versus iterations, $\lambda = 2 \times 10^{-1}$ (b) ISNR versus iterations, $\lambda = 2 \times 10^{-1}$ (c) δ_k versus iterations, $\lambda = 8 \times 10^{-2}$ (d) ISNR versus iterations, $\lambda = 8 \times 10^{-2}$ (e) δ_k versus iterations, $\lambda = 6 \times 10^{-2}$ (f) ISNR versus iterations, $\lambda = 6 \times 10^{-2}$

Figure 9.10: A comparison among DRPD1, DRPD2, ADMM, and OSGA for deblurring the 256×256 Fingerprint image with the various regularization parameter λ . The blurred/noisy image was constructed by the 7×7 Gaussian kernel with standard deviation 5 and salt-and-pepper impulsive noise with the level 50%. The algorithms were stopped after 50 iterations. Subfigures (a), (c), and (e) display the relative error δ_2 of function values versus iterations, and (b), (d), and (f) show ISNR versus iterations.



(a) Original image



(b) Blurred/noisy image

(c) DRPD-1: $f = 1.3571e + 4$, PSNR = 21.87, T = 1.28(d) DRPD-2: $f = 1.3635e + 4$, PSNR = 21.23, T = 0.83(e) ADMM: $f = 1.3799e + 4$, PSNR = 20.10, T = 0.76(f) OSGA: $f = 1.3612e + 4$, PSNR = 22.04, T = 6.01

Figure 9.11: Deblurring of the 256×256 Fingerprint image using DRPD1, DRPD2, ADMM, and OSGA with the regularization parameter $\lambda = 7 \times 10^{-3}$. The algorithms were stopped after 50 iterations. The blurred/noisy image was constructed by the 7×7 Gaussian kernel with standard deviation 5 and salt-and-pepper impulsive noise with the level 50%.

The results of Figure 9.10 demonstrate that the algorithms are sensitive to the regularization parameter. For example, ADMM get the best function value and an acceptable ISNR compared with the others for $\lambda = 3 \times 10^{-1}$, however, for $\lambda = 1 \times 10^{-1}$ and $\lambda = 8 \times 10^{-2}$, it behaves worse than the others. In the sense of ISNR, DRPD1, DRPD2, and OSGA are competitive, but OSGA get the better results. The resulting images for $\lambda = 8 \times 10^{-2}$ are illustrated in Figure 9.11, demonstrating that ADMM could not recover the image properly, however, DRPD1, DRPD2, and OSGA reconstruct acceptable approximations, where OSGA attains the best PSNR.

Chapter 10

Solving nonsmooth convex problems with complexity $O(\varepsilon^{-1/2})$

In this chapter we consider structured nonsmooth problems of the form

$$\begin{aligned} \min \quad & f(Ax, \phi(x)) \\ \text{s.t.} \quad & x \in C, \end{aligned}$$

where $f : U \times \mathbb{R} \rightarrow \overline{\mathbb{R}}$ is a proper and convex function that is smooth with Lipschitz continuous gradients with respect to both arguments and monotone increasing with respect to the second argument, $A : V \rightarrow U$ is a linear operator, $C \subseteq V$ is a simple convex domain, and $\phi : V \rightarrow \mathbb{R}$ is a simple nonsmooth, real-valued, and convex loss function. Then we apply the new setup of the optimal subgradient framework (OSGA-O) described in Chapter 6 to the reformulated problem

$$\begin{aligned} \min \quad & \widehat{f}(x, \xi) \\ \text{s.t.} \quad & x \in \widehat{C}, \end{aligned} \tag{10.1}$$

where

$$\begin{aligned} \widehat{f}(x, \xi) &:= f(Ax, \xi), \\ \widehat{C} &:= \{(x, \xi) \in V \times \mathbb{R} \mid x \in C, \phi(x) \leq \xi\}, \end{aligned}$$

and we give some comparisons with OSGA and some state-of-the-art solvers.

We consider solving an underdetermined system $Ax = y$, where A is a $m \times n$ matrix ($m \leq n$) and y is a m -vector. Underdetermined system of linear equations is frequently appeared in many applications of linear inverse problem such as those in the fields signal and image processing, geophysics, economics, machine learning, and statistics. The objective is to recover x from the observed vector y by some optimization models. Due to the ill-conditioned feature of the problem,

the most popular optimization models are (6.8), (6.9), and (6.10), where (6.8) is smooth and (6.9) and (6.10) are nonsmooth. In Section 10.1 we report numerical results with the ℓ_1 minimization (6.9) and in Section 10.2 give results regarding the Elastic Net minimization problem (6.10).

10.1 ℓ_1 minimization

We here consider the ℓ_1 minimization problem

$$\begin{aligned} \min \quad & \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1 \\ \text{s.t.} \quad & x \in \mathbb{R}^n, \end{aligned}$$

reformulate it as a minimization problem of the form (10.1) with

$$f(x, \xi) := \frac{1}{2} \|y - Ax\|_2^2 + \xi, \quad \phi(x) := \lambda \|x\|_1,$$

and apply OSGA-O to solve it. We give some numerical results and a comparison with OSGA and some state-of-the-art solvers.

Let us set $m = 5000$ and $n = 10000$. The data for problem (4.8) is constructed randomly by

$$A = \text{rand}(m, n), \quad y = \text{rand}(1, m), \quad x_0 = \text{rand}(1, n).$$

In our comparison with OSGA, and OSGA-O, we consider two set of solvers: (i) PGA, FISTA, NESCO, and NESUN described in Section 8.1.2 that can directly applied to the nonsmooth problem; (ii) NSDSG, NES83, NESCS, and NES05 in which the nonsmooth first-order oracle required, where NES83, NESCS, and NES05 are adapted to take a subgradient in the place of the gradient (see Section 8.1.2 for more details). In both cases we stop the algorithms after 30 seconds of the running time. We set

$$\widehat{L} := \max_{1 \leq i \leq n} \|a_i\|^2,$$

where a_i , for $i = 1, 2, \dots, n$, is the i -th column of A . In the implementation, NESCS, NES05, PGA, and FISTA use $L = 10^4 \widehat{L}$, and NSDSG employs $\alpha_0 = 10^{-7}$. We report results for several regularization parameters described in Tables 10.1 and 10.2. In Figures 10.1 and 10.2, we illustrate function values versus iterations.

The results of Tables 10.1 and 10.2 show that OSGA is comparable with the others, however, OSGA-O obtain the much better function values for the ℓ_1 minimization problem. From Figures 10.1 and 10.2, it can be observed that the worst results obtained by NSDSG and PGA; FISTA, NESCO, NESUN, NES83, NESCS, NES05 and OSGA are comparable to some extent; OSGA-O superiors to the other methods significantly.

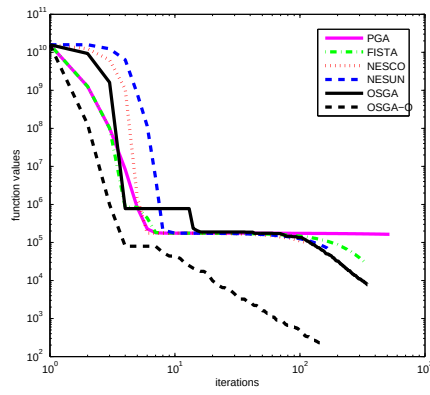
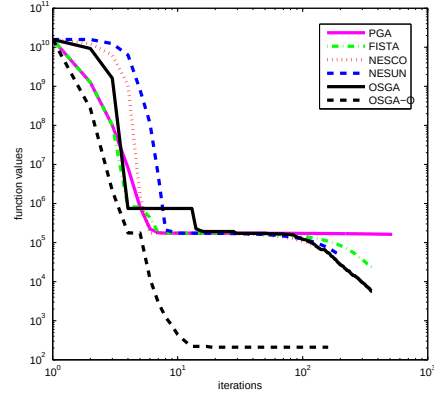
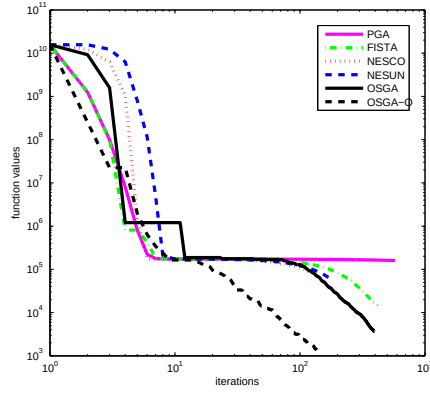
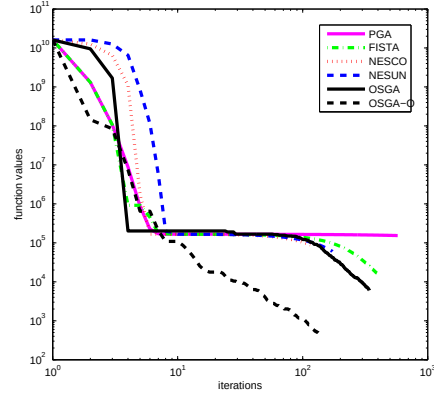
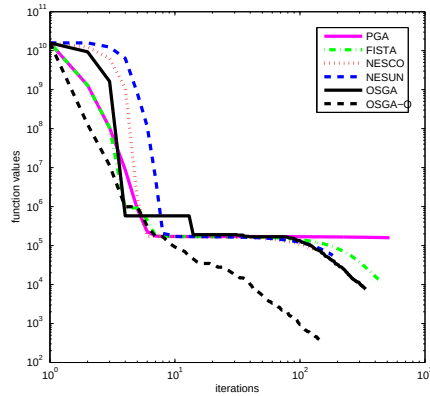
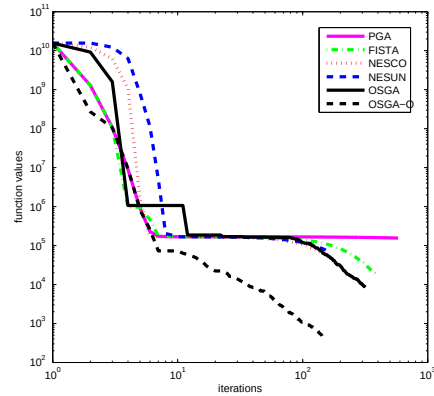
(a) $\lambda = 1$ (b) $\lambda = 10^{-1}$ (c) $\lambda = 10^{-2}$ (d) $\lambda = 10^{-3}$ (e) $\lambda = 10^{-4}$ (f) $\lambda = 10^{-5}$

Figure 10.1: A comparison among PGA, FISTA, NESCO, NESUN, OSGA, and OSGA-O for solving ℓ_1 minimization problem, where the algorithms stopped after 30 seconds. The problem was solved for 6 different regularization parameters: (a) $\lambda = 1$; (b) $\lambda = 10^{-1}$; (c) $\lambda = 10^{-2}$; (d) $\lambda = 10^{-3}$; (e) $\lambda = 10^{-4}$; (f) $\lambda = 10^{-5}$.

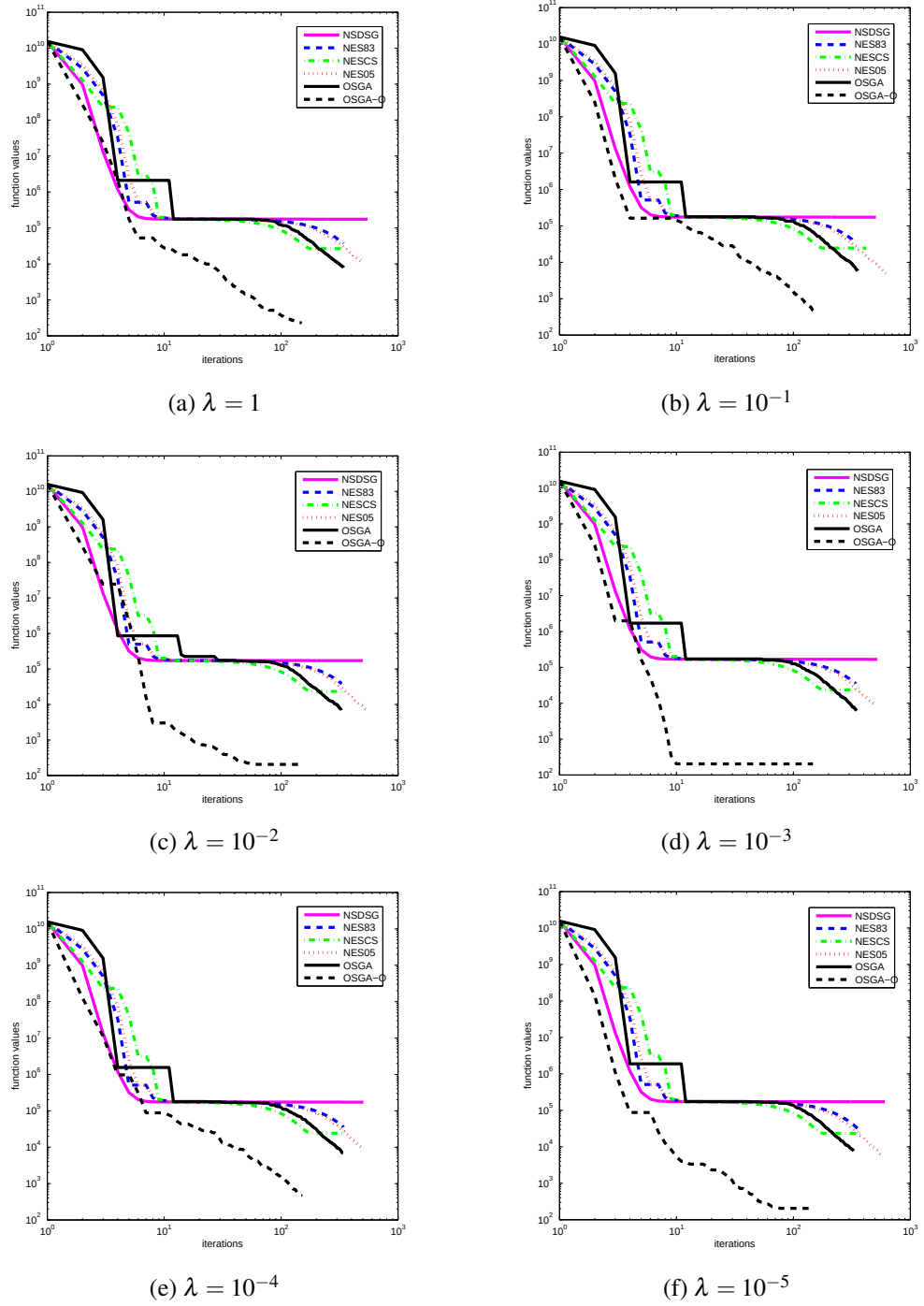


Figure 10.2: A comparison among NSDSG, NES83, NESCS, NES05, OSGA, and OSGA-O for solving ℓ_1 minimization problem, where the algorithms stopped after 30 seconds. The problem was solved for 6 different regularization parameters: (a) $\lambda = 1$; (b) $\lambda = 10^{-1}$; (c) $\lambda = 10^{-2}$; (d) $\lambda = 10^{-3}$; (e) $\lambda = 10^{-4}$; (f) $\lambda = 10^{-5}$.

Table 10.1: Function values of PGA, FISTA, NESCO, NESUN, OSGA, and OSGA-O for solving the ℓ_1 minimization problem with several regularization parameters.

Reg. parameter	PGA	FISTA	NESCO	NESUN	OSGA	OSGA-O
$\lambda = 1$	163813.29	29254.36	95648.33	65551.18	7705.11	224.05
$\lambda = 10^{-1}$	161882.90	28826.02	73503.70	52081.40	5654.31	209.12
$\lambda = 10^{-2}$	160173.56	14223.04	69294.26	62919.36	3668.93	1134.07
$\lambda = 10^{-3}$	153709.75	16835.48	88402.78	60112.50	6065.37	418.97
$\lambda = 10^{-4}$	158812.94	12630.09	74889.01	55774.92	76092.84	364.55
$\lambda = 10^{-5}$	155573.77	19060.71	60964.63	64549.46	8454.57	418.63

Table 10.2: Function values of NSDSG, NES83, NESCS, NES05, OSGA, and OSGA-O for solving the ℓ_1 minimization problem with several regularization parameters.

Reg. parameter	NSDSG	NES83	NESCS	NES05	OSGA	OSGA-O
$\lambda = 1$	174672.58	37465.49	26510.61	10918.82	789485	224.47
$\lambda = 10^{-1}$	173411.33	34340.16	42601.60	4893.27	583.5.84	444.79
$\lambda = 10^{-2}$	170935.10	38405.66	23236.88	6730.46	6881.61	204.62
$\lambda = 10^{-3}$	170272.21	35226.85	23708.27	9457.75	6295.59	203.87
$\lambda = 10^{-4}$	172775.42	35738.34	24127.66	87641.85	69687.72	452.96
$\lambda = 10^{-5}$	171480.55	25210.72	23291.35	6074.28	6750.34	204.52

10.2 Elastic Net minimization

We here consider the Elastic Net minimization problem

$$\begin{aligned} \min \quad & \frac{1}{2} \|y - Ax\|_2^2 + \frac{1}{2} \lambda_1 \|x\|_2^2 + \lambda_2 \|x\|_1 \\ \text{s.t.} \quad & x \in \mathbb{R}^n, . \end{aligned}$$

reformulate it as a minimization problem of the form (10.1) with

$$f(x, \xi) := \frac{1}{2} \|y - Ax\|_2^2 + \xi, \quad \phi(x) := \frac{1}{2} \lambda_1 \|x\|_2^2 + \lambda_2 \|x\|_1,$$

and apply OSGA-O to solve it. We give some numerical results and comparison with OSGA and some state-of-the-art solvers.

We construct the problem and consider solvers for a comparison the same as Section 10.1 by setting $m = 5000$ and $n = 10000$ and

$$A = \text{rand}(m, n), \quad y = \text{rand}(1, m), \quad x_0 = \text{rand}(1, n).$$

The results of our implementation are summarized in Tables 10.3 and 10.4 and Figures 10.3 and 10.4.

Table 10.3: Function values of PGA, FISTA, NESCO, NESUN, OSGA, and OSGA-O for solving Elastic Net problem with several regularization parameters.

Reg. parameter	PGA	FISTA	NESCO	NESUN	OSGA	OSGA-O
$\lambda = 1$	163170.01	23221.75	81268.32	61438.91	7242.91	209.89
$\lambda = 10^{-1}$	155821.50	18934.35	83111.27	57095.06	5439.81	213.44
$\lambda = 10^{-2}$	160336.99	23076.71	80901.20	50686.03	4891.32	1866.06
$\lambda = 10^{-3}$	159181.19	26166.87	92879.45	56724.63	8078.09	205.84
$\lambda = 10^{-4}$	163193.78	27845.31	855474.09	62852.97	7759.84	208.86
$\lambda = 10^{-5}$	163771.85	26335.87	90349.26	67991.56	7908.11	210.66

Table 10.4: Function values of NSDSG, NES83, NESCS, NES05, OSGA, and OSGA-O for solving Elastic Net problem with several regularization parameters.

Reg. parameter	NSDSG	NES83	NESCS	NES05	OSGA	OSGA-O
$\lambda = 1$	175349.77	33979.90	24608.58	69894.52	9476.02	207.28
$\lambda = 10^{-1}$	171367.80	28446.35	24270.92	5675.27	7624.94	201.33
$\lambda = 10^{-2}$	174310.41	22264.53	24207.23	9436.33	6944.39	2608.05
$\lambda = 10^{-3}$	171458.76	31253.67	24127.23	9476.60	8160.70	209.95
$\lambda = 10^{-4}$	164912.11	21584.42	23423.70	9174.28	4433.32	445.25
$\lambda = 10^{-5}$	180334.49	31765.64	24834.49	7656.03	6871.54	204.15

The results of Tables 10.3 and 10.4 show that the best function values are obtained by OSGA-O. In Figures 10.3 and 10.4, we can see that the worst results obtained by NSDSG and PGA; FISTA, NESCO, NESUN, NES83, NESCS, NES05 and OSGA behave competitively; OSGA-O outperforms the other methods considered.

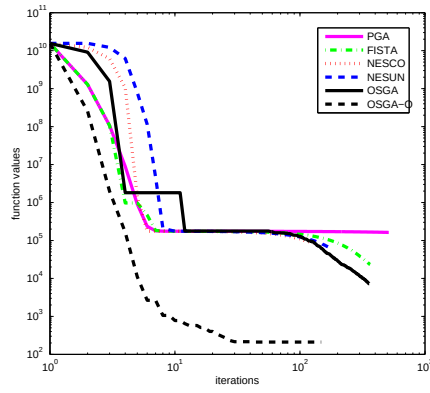
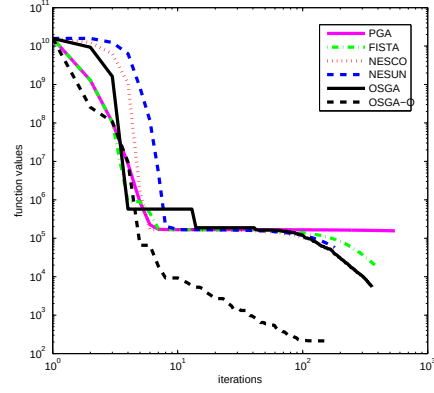
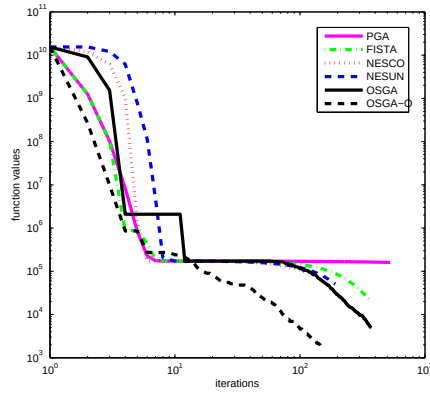
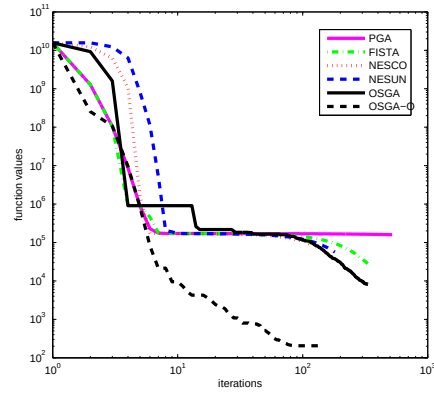
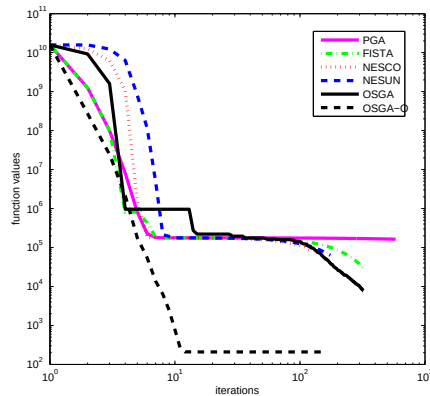
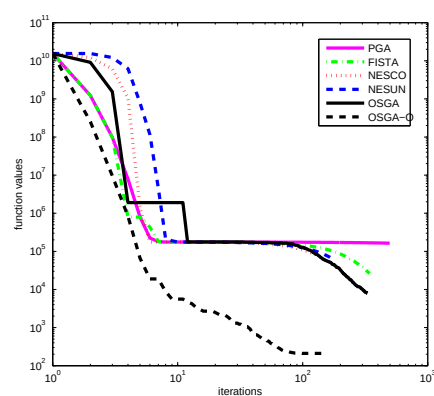
(a) $\lambda = 1$ (b) $\lambda = 10^{-1}$ (c) $\lambda = 10^{-2}$ (d) $\lambda = 10^{-3}$ (e) $\lambda = 10^{-4}$ (f) $\lambda = 10^{-5}$

Figure 10.3: A comparison among PGA, FISTA, NESCO, NESUN, OSGA, and OSGA-O for solving Elastic Net problem, where the algorithms stopped after 30 seconds. The problem was solved for 6 different regularization parameters: (a) $\lambda = 1$; (b) $\lambda = 10^{-1}$; (c) $\lambda = 10^{-2}$; (d) $\lambda = 10^{-3}$; (e) $\lambda = 10^{-4}$; (f) $\lambda = 10^{-5}$.

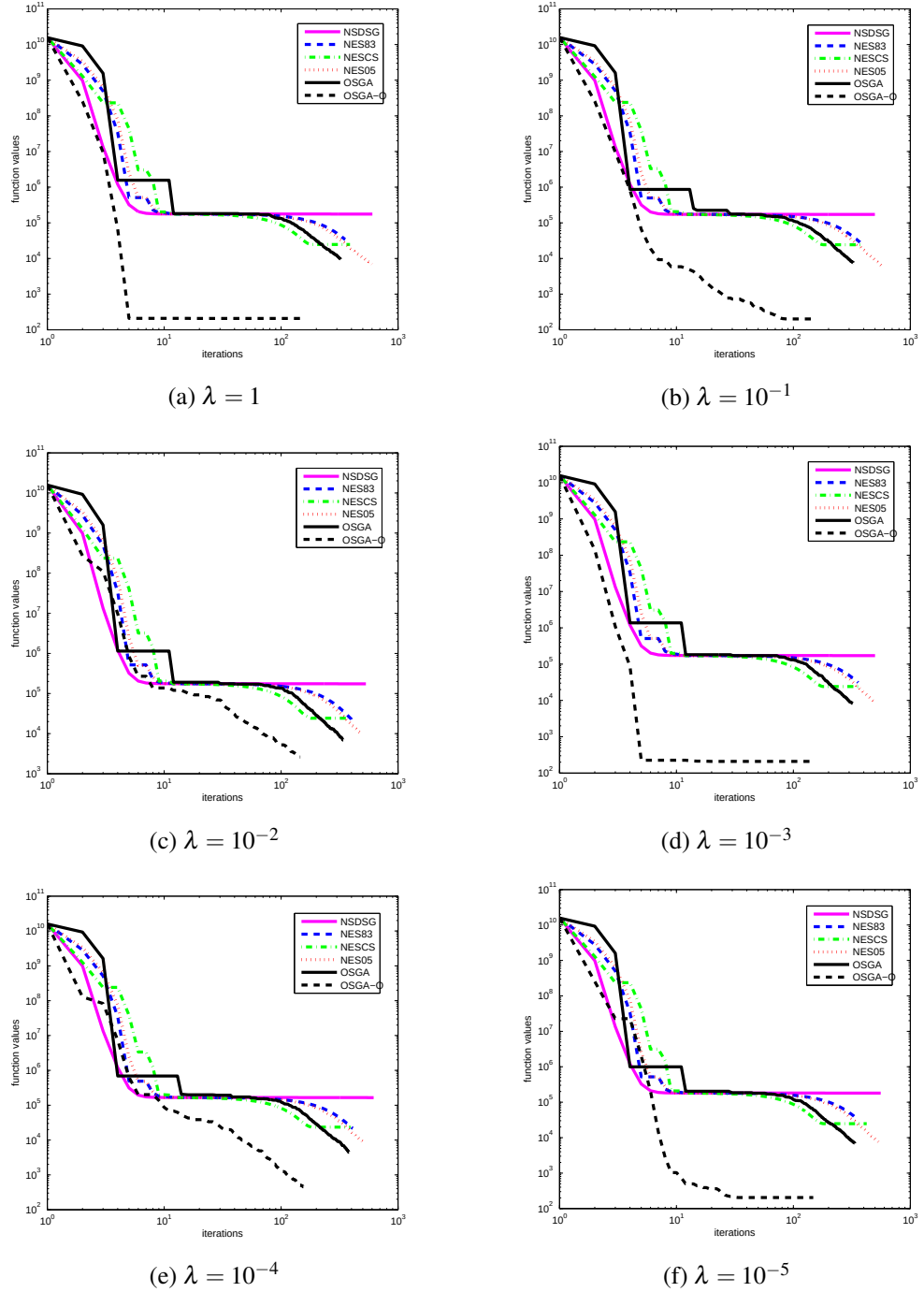


Figure 10.4: A comparison among NSDSG, NES83, NESCS, NES05, OSGA, and OSGA-O for solving Elastic Net problem, where the algorithms stopped after 30 seconds. The problem was solved for 6 different regularization parameters: (a) $\lambda = 1$; (b) $\lambda = 10^{-1}$; (c) $\lambda = 10^{-2}$; (d) $\lambda = 10^{-3}$; (e) $\lambda = 10^{-4}$; (f) $\lambda = 10^{-5}$.

Chapter 11

Summary and future work

In this chapter I first summarize the main contributions and results obtained in the thesis and then discuss possible extensions and some directions for future research.

11.1 Extended summary

In this thesis I have discussed several iterative schemes designed to solve convex optimization problems involving high-dimensional or big data. The main contributions of this thesis are summarized as follows:

1. We developed two optimal subgradient algorithms, namely single and double projection OSGA, that can be applied to general convex problems without considering their structure. The complexity analysis of these schemes showed that the number of applying the nonsmooth first-order oracle to reach an ε -solution is optimal and free from the problem dimension (see Chapter 3).
2. The auxiliary problem defined in the optimal subgradient framework is smooth but nonconvex in general (converse to Nesterov-type optimal methods for composite minimization requiring to solve nonsmooth but convex auxiliary problems). For unconstrained problems, the OSGA subproblem was solved in a closed form by choosing a suitable quadratic prox-function. It was also shown how one can efficiently apply OSGA to multi-term affine composite problems involving high-dimensional data (see Section 4.1, and Section 8.1 for numerical results).
3. An accelerated version of OSGA for solving convex problems involving costly linear operators and cheap nonlinear terms was developed by using

a multi-dimensional subspace search technique. Since the subspace search produces a low-dimensional problem, it is inexpensive and in the most cases produce a much better point compared to the current point (see Section 4.2, and Section 8.2 for numerical results).

4. It was shown that the OSGA subproblem can be solved globally for bound-constrained problems by translating it into a one-dimensional piece-wise rational problem and giving a simple iterative scheme to handle it (see Section 5.2, and Section 9.2 for numerical results).
5. Finding the solution of the OSGA subproblem was investigated for convex problems in simple domains (the orthogonal projection should be available cheaply). In this case it was shown that a closed form solution of the OSGA subproblem exists for some simple domains. In addition, it was investigated that OSGA can handle convex problems with simple functional constraints (see Section 5.1, and Section 9.1 for numerical results).
6. If the nonsmoothness of the objective is manifested in an appropriately structured form, a novel optimal subgradient method was presented that can attain the complexity $O(\epsilon^{-1/2})$, the optimal complexity as for smooth problems with Lipschitz continuous gradients. In this case, using an appropriate prox-function, it was proved that solving the OSGA subproblem is equivalent to solving a proximal-like problem that also appears in Nesterov-type optimal methods for solving composite minimization. In addition, the proximal-like problem was solved for many interesting cases appeared in applications, which can be also used in proximal-based methods (see Chapter 6, and Chapter 10 for numerical results).
7. The thesis is accompanied by a MATLAB software package release, called OSGA, which is publicly available. The OSGA software package covers several classes of problems such as unconstrained, bound-constrained, simply constrained, and simply functional constrained problems (see Chapter 7 and the user guide [2]).
8. We finally considered many applications in signal and image processing, compressed sensing, sparse recovery, and statistics and conducted extensive numerical experiments and comparisons with state-of-the-art solvers. The numerical results show that OSGA has good behavior for solving real-life applications and outperforms most of state-of-the-art solvers considered (see Chapters 7, 8, 9, and 10).

11.2 Directions for future research

This thesis consists some iterative schemes for solving convex optimization problems. We conclude this chapter by briefly outlining some questions remained unanswered and some of the possible future directions of this research.

1. We have seen that, for some simple domains C , it is possible to use some suitable prox-functions such that the OSGA subproblem can be solved efficiently in a closed form or by simple iterative schemes. Which kind of simple domains and which prox-functions can be considered such that the OSGA subproblem can be solved effectively?
2. Is it possible to give universal versions of OSGA-V and OSGA that are optimal for Lipschitz continuous nonsmooth problems, weakly smooth problems, and smooth problems with Lipschitz continuous gradients at the same time?
3. In the thesis we have supposed that the exact first-order information of an objective is available. It would be interesting to verify the convergence analysis of OSGA-V and OSGA with inexact oracles.
4. It might be helpful to study whether the approximate solution of the subproblem affect the quality of iterations before the function values match an optimum within a reasonable accuracy.
5. It would be interesting to study a cutting-plane version and a bundle version of OSGA-V and OSGA.
6. We have seen that problems with a simple convex functional constraint are tractable. Is it possible to extend techniques to handle problems involving general convex functional constraints?
7. No numerical results for OSGA-V were reported. It would be interesting to compare the performance of OSGA-V with OSGA and some state-of-the-art solvers.
8. There are many aspects of development for the OSGA software package such as considering many more prox-functions, simple domains, and proximal-like problems. It is also possible to write OSGA in faster programming languages such as C, C++, and Fortran. It would be interesting to possibly use parallel computing techniques to accelerate OSGA.
9. There exist many more interesting applications in fields of applied sciences and engineering such as problems with matrix variables like nuclear norm

minimization and sparse covariance selection that can be considered to evaluate the performance of OSGA and OSGA-V. Which kind of applications can be handled efficiently by OSGA-V and OSGA?

Bibliography

- [1] M. Ahookhosh. Optimal subgradient algorithms with application to large-scale linear inverse problems. *Submitted*, 2014. <http://arxiv.org/abs/1402.7291>.
- [2] M. Ahookhosh. User's manual for OSGA (optimal subgradient algorithm). 2014. http://homepage.univie.ac.at/masoud.ahookhosh/uploads/User's_manual_for_OSGA.pdf.
- [3] M. Ahookhosh and K. Amini. An efficient nonmonotone trust-region method for unconstrained optimization. *Numerical Algorithms*, 59 (4):523–540, 2012.
- [4] M. Ahookhosh, K. Amini, and S. Bahrami. A class of nonmonotone armijo-type line search method for unconstrained optimization. *Optimization*, 61(4):387–404, 2012.
- [5] M. Ahookhosh, K. Amini, and M. Kimiaei. A globally convergent trust-region method for large-scale symmetric nonlinear systems. *Numerical Functional Analysis and Optimization*, 36:830–855, 2015.
- [6] M. Ahookhosh, H. Esmaili, and M. Kimiaei. An effective trust-region-based approach for symmetric nonlinear systems. *International Journal of Computer Mathematics*, 90 (3):671–690, 2013.
- [7] M. Ahookhosh and S. Ghaderi. On efficiency of nonmonotone armijo-type line searches. *Submitted*, 2014. <http://arxiv.org/abs/1408.2675>.
- [8] M. Ahookhosh and S. Ghaderi. Two globally convergent nonmonotone trust-region methods for unconstrained optimization. *Journal of Applied Mathematics and Computing*, 2015. DOI 10.1007/s12190-015-0883-9.
- [9] M. Ahookhosh and A. Neumaier. High-dimensional convex optimization via optimal affine subgradient algorithms. In *ROKS Workshop*, pages 83–84, 2013.

- [10] M. Ahookhosh and A. Neumaier. An optimal subgradient algorithm for large-scale bound-constrained convex optimization. *Submitted*, 2015. <http://arxiv.org/abs/1501.01497>.
- [11] M. Ahookhosh and A. Neumaier. An optimal subgradient algorithm for large-scale convex optimization in simple domains. *Submitted*, 2015. <http://arxiv.org/abs/1501.01451>.
- [12] M. Ahookhosh and A. Neumaier. An optimal subgradient algorithm with subspace search for costly convex optimization problems. *Submitted*, 2015. http://www.optimization-online.org/DB_HTML/2015/04/4852.html.
- [13] M. Ahookhosh and A. Neumaier. Solving nonsmooth convex optimization with complexity $o(\varepsilon^{-1/2})$. *Submitted*, 2015. http://www.optimization-online.org/DB_HTML/2015/05/4900.html.
- [14] K. Amini, M. Ahookhosh, and H. Nosratipour. An inexact line search approach using modified nonmonotone strategy for unconstrained optimization. *Numerical Algorithms*, 66:49–78, 2014.
- [15] H. Andrews and B. Hunt. *Digital Image Restoration*. NJ: Prentice-Hall, Englewood Cliffs, 1977.
- [16] A. Auslender and M. Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM Journal on Optimization*, 16:697–725, 2006.
- [17] M. Baes. Estimate sequence methods: extensions and approximations. *IFOR Internal report, ETH, Zurich, Switzerland*, 2009.
- [18] M. Baes and Michael Bürgisser. An acceleration procedure for optimal first-order methods. *Optimization Methods & Software*, 9(3):610–628, 2014.
- [19] A. Bagirov, N. Karimtsa, and M.M. Mäkelä. *Introduction to Nonsmooth Optimization: theory, practice and software*. Springer International, Publishing, 2014.
- [20] J. Bardsley and C.R. Vogel. A nonnegatively constrained convex programming method for image reconstruction. *SIAM Journal on Scientific Computing*, 25:1326–1343, 2003.

- [21] R.H. Bartels, A.R. Conn, and Y. Li. Primal methods are better than dual methods for solving overdetermined linear systems in the ℓ_∞ sense? *SIAM Journal on Numerical Analysis*, 26(3):693–726, 1989.
- [22] R.H. Bartels, A.R. Conn, and J.W. Sinclair. Minimization techniques for piecewise differentiable functions: The l_1 solution to an overdetermined linear system. *SIAM Journal on Numerical Analysis*, 15(2):224–241, 1978.
- [23] H.H. Bauschke. *Projection algorithms and monotone operators*. PhD thesis, Simon Fraser University, 1996. https://people.ok.ubc.ca/bauschke/Research/bauschke_thesis.pdf.
- [24] H.H. Bauschke and P.L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. CMS Books Math., Springer-Verlag, New York, 2011.
- [25] A. Beck, A. Ben-Tal, N. Guttman-Beck, and L. Tetruashvili. The CoMirror algorithm for solving nonsmooth constrained convex problems. *Operations Research Letters*, 38(6):493–498, 2010.
- [26] A. Beck and M. Teboulle. Mirror descent and nonlinear projected sub-gradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.
- [27] A. Beck and M. Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009.
- [28] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.
- [29] A. Beck and M. Teboulle. Smoothing and first order methods: a unified framework. *SIAM Journal on Optimization*, 22:557–580, 2012.
- [30] S.R. Becker, E.J. Candès, and M.C. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3:165–218, 2011.
- [31] E.V.D. Berg and M. P. Friedlander. Sparse optimization with least-squares constraints. *SIAM Journal on Optimization*, 21:1201–1229, 2011.
- [32] E.V.D. Berg and M.P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

- [33] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. *in Proceedings of SIGGRAPH*, pages 417–424, 2000.
- [34] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. U.K.: IOP, Bristol, 1998.
- [35] D.P. Bertsekas. A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization*, 7(4):913–926, 1997.
- [36] D.P. Bertsekas. *Nonlinear Programming*. 2nd ed., Athena Scientific, Belmont, MA, 1999.
- [37] J. Bioucas-Dias and M. Figueiredo. A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16(12):2992–3004, 2007.
- [38] E.G. Birgin, J.M. Martinez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10:1196–1211, 2000.
- [39] R.I. Boş, E.R. Csetnek, and C. Hendrich. A primal-dual splitting algorithm for finding zeros of sums of maximally monotone operators. *SIAM Journal on Optimization*, 23:2011–2036, 2013.
- [40] R.I. Boş and C. Hendrich. A double smoothing technique for solving unconstrained nondifferentiable convex optimization problems. *Computational Optimization and Applications*, 54(2):239–262, 2013.
- [41] R.I. Boş and C. Hendrich. A Douglas-Rachford type primal-dual method for solving inclusions with mixtures of composite and parallel-sum type monotone operators. *SIAM Journal on Optimization*, 23(4):2541–2565, 2013.
- [42] R.I. Boş and C. Hendrich. On the acceleration of the double smoothing technique for unconstrained convex optimization problems. *Optimization*, 64(2):265–288, 2015.
- [43] S. Boyd and L. Vandenberg. *Convex Optimization (12th printing)*. Cambridge University Press, New York, 2012.
- [44] S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. *Stanford University*, 2003. http://www.stanford.edu/class/ee392o/subgrad_method.pdf.

- [45] M.A. Branch, T.F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21:1–23, 1999.
- [46] A.M. Bruckstein, D.L. Donoho, and M Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.
- [47] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16:1190–1208, 1995.
- [48] J.A. Cadzow. Minimum ℓ_1 , ℓ_2 , and ℓ_∞ norm approximate solutions to an overdetermined system of linear equations. *Digital Signal Processing*, 12:524–560, 2002.
- [49] E. Candés. Compressive sampling. in *Proceedings of International Congress of Mathematics, Vol. 3, Madrid, Spain*, pages 1433–1452, 2006.
- [50] E. Candés, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions of Information Theory*, 52(2):489–509, 2006.
- [51] E. Candés and T. Tao. Near-optimal signal recovery from random projections: universal encoding strategies? *IEEE Transactions of Information Theory*, 52(12):5406–5425, 2006.
- [52] V. Cevher, S. Becker, and M. Schmidt. Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Processing Magazine*, 31(5):32–43, 2014.
- [53] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1-2):89–97, 2004.
- [54] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. In *Theoretical Foundations and Numerical Methods for Sparse Recovery*, volume 9, pages 263–340. De Gruyter, 2010.
- [55] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

- [56] R.H. Chan, M. Tao, and X. Yuan. Constrained total variation deblurring models and fast algorithms based on alternating direction method of multipliers. *SIAM Journal on Imaging Science*, 6(1):680–697, 2013.
- [57] T.F. Chan, J. Shen, and H.M. Zhou. Total variation wavelet inpainting. *Journal of Mathematical Imaging and Vision*, 25:107–125, 2006.
- [58] S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1999.
- [59] Y. Chen, W.W. Hager, M. Yashtini, X. Ye, and H. Zhang. Bregman operator splitting with variable stepsize for total variation image reconstruction. *Computational Optimization and Applications*, 54(2):317–342, 2013.
- [60] Y. Chen, G. Lan, and Y. Ouyang. An accelerated linearized alternating direction method of multipliers. *Submitted*, 2014. <http://arxiv.org/pdf/1401.6607v3.pdf>.
- [61] Y. Chen, G. Lan, and Y. Ouyang. Optimal primal-dual methods for a class of saddle point problems. *SIAM Journal on Optimization*, 24(4):1779–1814, 2014.
- [62] Y. Chen, G. Lan, Y. Ouyang, and W. Zhang. Fast bundle-level type methods for unconstrained and ball-constrained convex optimization. *Submitted*, 2014. <http://arxiv.org/pdf/1412.2128v1.pdf>.
- [63] E. Chouzenoux, J. Idier, and S. Moussaoui. A majorize-minimize strategy for subspace optimization applied to image restoration. *IEEE Transactions on Image Processing*, 20(18):1517–1528, 2011.
- [64] P. Combettes and J.C. Pesquet. Proximal splitting methods in signal processing. In H.H. Bauschke and R. Burachik, P.L. Combettes, V. Elser, D.R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer, 2011.
- [65] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25:433–460, 1988.
- [66] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, 2000.
- [67] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. SIAM, Philadelphia, 2009.

- [68] B. Cox, A. Juditsky, and A. Nemirovski. Dual subgradient algorithms for large-scale nonsmooth learning problems. *Mathematical Programming*, 148(1-2, Ser. B):143–180, 2014.
- [69] E.E. Cragg and A.V. Levy. Study on a supermemory gradient method for the minimization of functions. *Journal of Optimization Theory and Applications*, 4(3):191–205, 1969.
- [70] Y.H. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming*, 106:403–421, 2006.
- [71] A. D'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30:56–66, 2008.
- [72] I. Daubechies, R. DeVore, M. Fornasier, and C.S. Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010.
- [73] O. Devolder, F. Glineur, and Y. Nesterov. Double smoothing technique for large-scale linearly constrained convex optimization. *SIAM Journal on Optimization*, 22(2):702–727, 2012.
- [74] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146:37–75, 2013.
- [75] E. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.
- [76] D.L. Donoho. Compressed sensing. *IEEE Transactions of Information Theory*, 52(4):1289–1306, 2006.
- [77] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l_1 -ball for learning in high dimensions. in *Proc. Int. Conf. Mach. Learn. (ICML), Helsinki, Finland*, 2008.
- [78] M. Elad, B. Matalon, and M. Zibulevsky. Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization. *Applied and Computational Harmonic Analysis*, 23(3):346–367, 2006.
- [79] M. Elad, P. Milanfar, and R. Rubinstein. Analysis versus synthesis in signal priors. *Inverse Problems*, 23:947–968, 2007.

- [80] T. Elfving, P.C. Hansen, and T. Nikazad. Semiconvergence and relaxation parameters for projected sirt algorithms. *SIAM Journal on Scientific Computing*, 34(4):A2000–A2017, 2012.
- [81] E. Esser, Y. Lou, and J. Xin. A method for finding structured sparse solutions to nonnegative least squares problems with applications. *SIAM Journal on Imaging Science*, 6(4):2010–2046, 2013.
- [82] J. Fan, F. Han, and H. Liu. Challenges of big data analysis. *National Science Review*, 1:293–314, 2014.
- [83] J. Fan, J. Lv, and L. Qi. Sparse high-dimensional models in economics. *Annual Review of Economics*, 3:291–317, 2011.
- [84] 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(4):586–597, 2007.
- [85] A. Friedlander, J.M. Martínez, and S.A. Santos. A new trust region algorithm for bound constrained minimization. *Applied Mathematics and Optimization*, 30:235–266, 1994.
- [86] C.C. Gonzaga and E. W. Karas. Fine tuning Nesterov’s steepest descent algorithm for differentiable convex programming. *Mathematical Programming*, 138:141–166, 2013.
- [87] C.C. Gonzaga, E. W. Karas, and D.R. Rossetto. An optimal algorithm for constrained differentiable convex optimization. *SIAM Journal on Optimization*, 23(4):1939–1955, 2013.
- [88] L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for Newton’s method. *SIAM Journal on Numerical Analysis*, 23:707–716, 1986.
- [89] W.W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on Optimization*, 16:170–192, 2005.
- [90] W.W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*, 2:35–58, 2006.
- [91] A. Hantoute, M.A. López, and C. Zălinescu. Subdifferential calculus rules in convex analysis: A unifying approach via pointwise supremum functions. *SIAM Journal on Optimization*, 19:863–882, 2008.

- [92] N. He, A. Juditsky, and A. Nemirovski. Mirror prox algorithm for multi-term composite minimization and semi-separable problems. *Computational Optimization and Applications*, pages DOI 10.1007/s10589-014-9723-3, 2015.
- [93] R. Helgason, J. Kennington, and H. Lall. A polynomially bound algorithms for a singly constrained quadratic program. *Mathematical Programming*, 18:338–343, 1980.
- [94] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I & II*. Springer-Verlag, New York, 1993.
- [95] A.E. Hoerl and R.W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.
- [96] A.D. Ioffe and V.H. Tikhomirov. *Theory of Extremal Problems*. in Stud. Math. Appl. 6, North-Holland, Amsterdam, 1979.
- [97] A. Juditsky and Y. Nesterov. Deterministic and stochastic primal-dual sub-gradient algorithms for uniformly convex minimization. *Stochastic Systems*, 4(1):44–80, 2014.
- [98] C. Kanzow, N. Yamashita, and M. Fukushima. Levenberg-Marquardt methods with strong local convergence properties for solving equations with convex constraints. *Journal of Computational and Applied Mathematics*, 172:375–397, 2004.
- [99] N. Karmita and M.M. Mäkelä. Adaptive limited memory bundle method for bound constrained large-scale nonsmooth optimization. *Optimization*, 59(6):945–962, 2010.
- [100] N. Karmita and M.M. Mäkelä. Limited memory bundle method for large bound constrained nonsmooth optimization: convergence analysis. *Optimization Methods and Software*, 25(6):895–916, 2010.
- [101] L. Kaufman and A. Neumaier. Pet regularization by envelope guided conjugate gradients. *IEEE Transactions on Medical Imaging*, 15:385–389, 1996.
- [102] L. Kaufman and A. Neumaier. Regularization of ill-posed problems by envelope guided conjugate gradients. *Journal of Computational and Graphical Statistics*, 6(4):451–463, 1997.
- [103] D. Kim, S. Sra, and I.S. Dhillon. Tackling box-constrained optimization via a new projected quasi-Newton approach. *SIAM Journal on Scientific Computing*, 32:3548–3563, 2010.

- [104] Y. Kim, J. Kim, and Y. Kim. Blockwise sparse regression. *Statistica Sinica*, 16(2):375–390, 2006.
- [105] G. Lan. Bundle-level type methods uniformly optimal for smooth and non-smooth convex optimization. *Mathematical Programming*, 149:1–45, 2015.
- [106] G. Lan, Z. Lu, and R.D.C. Monteiro. Primal-dual first-order methods with $o(1/\epsilon)$ iteration-complexity for cone programming. *Mathematical Programming*, 126:1–29, 2011.
- [107] A.S. Lewis and M.L. Overton. Nonsmooth optimization via quasi-Newton methods. *Mathematical Programming*, 141:135–163, 2013.
- [108] D.H. Li, N. Yamashita, and M. Fukushima. Nonsmooth equation based bfgs method for solving kkt systems in mathematical programming. *Journal of Optimization Theory and Applications*, 109(1):123–167, 2001.
- [109] C.J. Lin and J.J. Moré. Newton’s method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9:1100–1127, 1999.
- [110] D.C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.
- [111] H. Liu, J. Zhang, X. Jiang, and J. Liu. The group Dantzig selector. *Journal of Machine Learning Research-Proceedings Track*, 9:461–468, 2010.
- [112] M.M. Mäkelä and P. Neittaanmäki. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific, Singapore, 1992.
- [113] A. Miele and J.W. Cantrell. Study on a memory gradient method for the minimization of functions. *Journal of Optimization Theory and Applications*, 3(6):459–470, 1969.
- [114] S. Morini, M. Porcelli, and R.H. Chan. A reduced Newton method for constrained linear least squares problems. *Journal of Computational and Applied Mathematics*, 233:2200–2212, 2010.
- [115] G. Narkiss and M. Zibulevsky. Sequential subspace optimization method for large-scale unconstrained problems. *Technical report CCIT 559, EE Dept., Technion, Haifa, Israel*, 2005.

- [116] A. Nedić and D.P. Bertsekas. Incremental subgradient methods for non-differentiable optimization. *JSIAM Journal on Optimization*, 12:109–138, 2001.
- [117] A.S. Nemirovsky and D.B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley, New York, 1983.
- [118] Y. Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Doklady AN SSSR (In Russian)*, 269:543–547, 1983.
- [119] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Dordrecht, 2004.
- [120] Y. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, 16:235–249, 2005.
- [121] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.
- [122] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120:221–259, 2006.
- [123] Y. Nesterov. Gradient methods for minimizing composite objective function. *Mathematical Programming*, 140:125–161, 2013.
- [124] Y. Nesterov. Universal gradient methods for convex optimization problems. *Mathematical Programming*, pages DOI 10.1007/s10107-014-0790-0, 2015.
- [125] Y. Nesterov and A. Nemirovski. On first-order algorithms for ℓ_1 /nuclear norm minimization. *Acta Numerica*, 22:509–575, 2013.
- [126] Y. Nesterov and A. Nemirovsky. *Interior-Point Polynomial Methods in Convex Programming*. volume 13 of Studies in Applied Mathematics, SIAM, Philadelphia, 1994.
- [127] A. Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40(3):636–666, 1998.
- [128] A. Neumaier. *Introduction to Numerical Analysis*. Cambridge University Press, Cambridge, 2001.
- [129] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.

- [130] A. Neumaier. OSGA: a fast subgradient algorithm with optimal complexity. *Mathematical Programming*, pages DOI 10.1007/s10107-015-0911-4, 2015.
- [131] A. Neumaier and B. Azmi. LMBOPT: a limited memory method for bound-constrained optimization. *Manuscript, University of Vienna*, 2015.
- [132] M. Nikolova. Minimizers of cost-functions involving nonsmooth data-fidelity terms. *SIAM Journal on Numerical Analysis*, 40(3):965–994, 2002.
- [133] M. Nikolova. A variational approach to remove outliers and impulse noise. *Journal of Mathematical Imaging and Vision*, 20:99–120, 2004.
- [134] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, New York, 2006.
- [135] S. Osher, L. Rudin, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica*, 60:259–268, 1992.
- [136] J.S. Pang and L. Qi. Nonsmooth equations: motivation and algorithms. *SIAM Journal on Optimization*, 3:443–465, 1993.
- [137] P.M. Pardalos and N. Kuvorov. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46:321–328, 1990.
- [138] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.
- [139] B. Polyak. *Introduction to Optimization*. Optimization Software, Inc., Publications Division, New York, 1987.
- [140] F.A. Potra, L. Qi, and D. Sun. Secant methods for semismooth equations. *Numerische Mathematik*, 80(2):305–324, 1998.
- [141] L. Qi. Trust region algorithms for solving nonsmooth equations. *SIAM Journal on Optimization*, 5:219–230, 1995.
- [142] L. Qi and D. Sun. A survey of some nonsmooth equations and smoothing Newton methods. *Progress in Optimization*, 30:121–146, 1999.
- [143] H. Rauhut and R. Ward. Interpolation via weighted ℓ_1 -minimization. *Applied and Computational Harmonic Analysis*, 2015.

- [144] M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7:26–33, 1997.
- [145] B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.
- [146] L.M. Rios and N.V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56:1247–1293, 2013.
- [147] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, N.J., 1970.
- [148] N.Z. Shor. *Minimization Methods for Non-Differentiable Functions*. Springer Series in Computational Mathematics, Springer, New York, 1985.
- [149] M.V. Solodov. Incremental gradient algorithms with step sizes bounded away from zero. *Computational Optimization and Applications*, 11:23–35, 1998.
- [150] D. Sun and J. Han. Newton and quasi-Newton methods for a class of non-smooth equations and related problems. *SIAM Journal on Optimization*, 7(2):463–480, 1997.
- [151] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.
- [152] A.N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics Doklady*, 4:1035–1038, 1963.
- [153] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *Technical report, Mathematics Department, University of Washington*, 2008. <http://pages.cs.wisc.edu/~brecht/cs726docs/Tseng.APG.pdf>.
- [154] C.R. Vogel. *Computational Methods for Inverse Problems*. Frontiers Appl. Math. 23, SIAM, Philadelphia, 2002.
- [155] Z. Wang and Y. Yuan. A subspace implementation of quasi-Newton trust region methods for unconstrained optimization. *Numerische Mathematik*, 104:241–269, 2006.

- [156] 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.
- [157] H. Woo and S. Yun. Proximal linearized alternating direction method for multiplicative denoising. *SIAM Journal on Scientific Computing*, 35:336–358, 2013.
- [158] 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.
- [159] W. Yin. Analysis and generalizations of the linearized bregman method. *SIAM Journal on Imaging Sciences*, 3(4):856–877, 2010.
- [160] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for ℓ_1 minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1:143–168, 2008.
- [161] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, 68:49–67, 2006.
- [162] Y. Yuan. Subspace techniques for nonlinear optimization. In R. Jeltsch, D.Q. Li, and I. H. Sloan, editors, *Some Topics in Industrial and Applied Mathematics*, pages 206–218. (Series in Contemporary Applied Mathematics CAM 8) Higher Education Press, 2007.
- [163] J. Zhang and B. Morini. Solving regularized linear least-squares problems by the alternating direction method with applications to image restoration. *Electronic Transactions on Numerical Analysis*, 40:356–372, 2013.
- [164] X. Zhang, A. Saha, and S.V.N. Vishwanathan. Lower bounds on rate of convergence of cutting plane methods. *Manuscript, Department of Computing Sciences, University of Alberta*, 2013.
- [165] C. Zălinescu. *Convex Analysis in General Vector Spaces*. World Scientific, River Edge, NJ, 2002.

EDUCATION

- 2012–2015 **PhD in Computational Optimization**, Faculty of Mathematics, University of Vienna,, Vienna, Austria.
- Thesis title: *High-dimensional nonsmooth convex optimization via optimal subgradient methods*
 - Supervisor: Prof. Arnold Neumaier
- 2007–2009 **Master in Applied Mathematics (Optimization)**, Department of Mathematics, Razi University, Kermanshah, Iran.
- Thesis title: *On nonmonotone trust-region methods with adaptive radius*
 - Supervisor: Prof. Keyvan Amini
- 2000–2005 **Bachelor in Applied Mathematics**, Department of Mathematics, Razi University, Kermanshah, Iran.

RESEARCH INTERESTS

- Nonsmooth Analysis and Optimization
- Convex Analysis and Optimization
- Large-Scale Nonlinear Optimization and Systems
- Software Development for Optimization
- Computational Mathematics

AWARDS AND GRANTS

- 2014 **SIAM Student Travel Award**, SIAM conference on Optimization (OP14), San Diego, California, USA.
- 2012–2015 **Initiativkolleg Computational Optimization**, Department of Mathematics, University of Vienna.

JOURNAL REFEREE

- Computational Optimization and Applications
- Optimization Theory and Applications
- Optimization Methods and Software
- Optimization
- Numerical Algorithms
- Applied Mathematics and Computing
- Applied Mathematics and Computation
- Bulletin of the Iranian Mathematical Society
- Signal, Image and Video Processing
- International Journal of Computer Mathematics
- Applied Mathematics

SCIENTIFIC MEMBERSHIP & AFFILIATION

- Member of Mathematical Optimization Society (MOS)
- Member of Society for Industrial and Applied Mathematics (SIAM)
- Member of Iranian Operations Research Society (IORS)
- Member of Iranian Mathematical Society (IMS)

PUBLICATIONS (PUBLISHED)

- [1] **M. Ahookhosh**, K. Amini, M. Kimiaei, *A globally convergent trust-region method for large-scale symmetric nonlinear systems*, Numerical Functional Analysis and Optimization, 36 (2015), 830–855.
- [2] **M. Ahookhosh**, S. Ghaderi, *Two globally convergent nonmonotone trust-region methods for unconstrained optimization*, Applied Mathematics and Computing, DOI 10.1007/s12190-015-0883-9, (2015).
- [3] K. Amini, **M. Ahookhosh**, H. Nosratipour, *An inexact line search approach using modified nonmonotone strategy for unconstrained optimization*, Numerical Algorithms, 66 (2014), 49–78.
- [4] **M. Ahookhosh**, K. Amini, M. Kimiaei, M.R. Peyghami, *A limited memory trust-region method with adaptive radius for large-scale unconstrained optimization*, Bulletin of the Iranian Mathematical Society, (2015).
- [5] K. Amini, **M. Ahookhosh**, *A hybrid of adjustable trust-region and nonmonotone algorithms for unconstrained optimization*, Applied Mathematical Modelling, 38 (2014), 2601–2612.
- [6] **M. Ahookhosh**, H. Esmaeili, M. Kimiaei, *An effective trust-region-based approach for symmetric nonlinear systems*, International Journal of Computer Mathematics, 90 (3) (2013), 671–690.
- [7] **M. Ahookhosh**, K. Amini, S. Bahrami, *Two derivative-free projection approaches for systems of large-scale nonlinear monotone equations*, Numerical Algorithms, 64 (2013), 21–42.
- [8] **M. Ahookhosh**, K. Amini, S. Bahrami, *A class of nonmonotone Armijo-type line search method for unconstrained optimization*, Optimization, 61(4) (2012), 387–404.
- [9] **M. Ahookhosh**, K. Amini, *An efficient nonmonotone trust-region method for unconstrained optimization*, Numerical Algorithms, 59(4) (2012), 523–540.
- [10] **M. Ahookhosh**, K. Amini, M.R. Peyghami, *A nonmonotone trust-region line search method for large-scale unconstrained optimization*, Applied Mathematical Modelling, 36 (2012), 478–487.
- [11] K. Amini, **M. Ahookhosh**, *Combination adaptive trust region method with nonmonotone strategy for unconstrained optimization*, Asia-Pacific Journal of Operational Research, 28(5) (2011), 585–600.
- [12] **M. Ahookhosh**, K. Amini, *A nonmonotone trust region method with adaptive radius for unconstrained optimization problems*, Computers and Mathematics with Applications, 60 (2010), 411–422.

PUBLICATIONS (SUBMITTED OR REVISED)

- [1] **M. Ahookhosh**, A. Neumaier, *Solving nonsmooth convex optimization with complexity $O(\varepsilon^{-1/2})$* , Submitted, (2015), http://www.optimization-online.org/DB_HTML/2015/05/4900.html.
- [2] **M. Ahookhosh**, A. Neumaier, *An optimal subgradient method with subspace search for costly convex optimization problems*, Submitted, (2015), http://www.optimization-online.org/DB_HTML/2015/04/4852.html.
- [3] **M. Ahookhosh**, A. Neumaier, *An optimal subgradient algorithms for large-scale convex optimization in simple domains*, Submitted, (2015), <http://arxiv.org/abs/1501.01451>.
- [4] **M. Ahookhosh**, A. Neumaier, *An optimal subgradient algorithms for large-scale bound-constrained convex optimization*, Submitted, (2015), <http://arxiv.org/abs/1501.01497>.
- [5] **M. Ahookhosh**, *Optimal subgradient algorithms with application to large-scale linear inverse problems*, Submitted, (2014), <http://arxiv.org/abs/1402.7291>.

- [6] **M. Ahookhosh**, S Ghaderi, *On efficiency of nonmonotone Armijo-type line searches*, Submitted, (2014), <http://arxiv.org/abs/1408.2675>.
- [7] A. Kamandi, K. Amini, **M. Ahookhosh**, *An improved adaptive trust-region framework*, Submitted, (2012).

PRESENTATIONS

- [1] **M. Ahookhosh**, A. Neumaier, *Optimal subgradient algorithms for large-scale structured convex optimization*, SIAM Conference on Optimization (OP14), San Diego, California, May 19–22 (2014).
- [2] **M. Ahookhosh**, A. Neumaier, *Optimal subgradient-based algorithms for large-scale convex optimization*, ICCOPT, Caparica, Lisbon, Portugal, (2013).
- [3] **M. Ahookhosh**, A. Neumaier, *High-dimensional convex optimization via optimal affine subgradient algorithms*, ROKS, Leuven, Belgium, July 8-10, (2013).
- [4] **M. Ahookhosh**, K. Amini, H. Nosratipour, *An improved nonmonotone technique for both line search and trust-region frameworks*, 21th International Symposium of Mathematical Programming (ISMP), Berlin, Germany, September, (2012).
- [5] **M. Ahookhosh**, K. Amini, H. Nosratipour, *A new nonmonotone line search method and its application for unconstrained optimization*, 4rd International Conference of Operations Research Society (OR), Guilan, Iran, May, (2011).
- [6] K. Amini, **M. Ahookhosh**, *A new nonmonotone trust-region method for unconstrained nonlinear programming*, 4rd International Conference of Operations Research Society (OR), Guilan, Iran, May, (2011).
- [7] **M. Ahookhosh**, K. Amini, S. Bahrami, *A modified class of nonmonotone Armijo-type line search method for unconstrained optimization*, 3rd International Conference of Operations Research Society (OR), Tehran, Iran, May, (2010).
- [8] K. Amini, **M. Ahookhosh**, *A nonmonotone trust region method with adaptive radius for unconstrained optimization problems*, 3rd International Conference of Operations Research Society (OR), Tehran, Iran, May, (2010).
- [9] K. Amini, **M. Ahookhosh**, *On nonmonotone trust region methods*, 2th Workshop on Optimization and its Applications, K. N. Toosi University, Tehran, Iran, May, (2010).
- [10] **M. Ahookhosh**, K. Amini, *Some adaptive trust region methods for unconstrained optimization*, 39th Annual Iranian Mathematics Conference, Kerman, Iran, August, (2008).

LANGUAGE SKILLS

- | | |
|----------------------------------|-----------------------|
| ◦ Kurdish (Native-Mother tongue) | ◦ Persian (Native) |
| ◦ English (Advanced knowledge) | ◦ German (Elementary) |

TECHNICAL SKILLS

Programming C, C++, Matlab, Maple, Mathematica

