Algorithms for Sparse and Low-Rank Optimization: Convergence, Complexity and Applications

Shiqian Ma

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2011

ABSTRACT

Algorithms for Sparse and Low-Rank Optimization:

Convergence, Complexity and Applications

Shiqian Ma

Solving optimization problems with sparse or low-rank optimal solutions has been an important topic since the recent emergence of compressed sensing and its matrix extensions such as the matrix rank minimization and robust principal component analysis problems. Compressed sensing enables one to recover a signal or image with fewer observations than the "length" of the signal or image, and thus provides potential breakthroughs in applications where data acquisition is costly. However, the potential impact of compressed sensing cannot be realized without efficient optimization algorithms that can handle extremely large-scale and dense data from real applications. Although the convex relaxations of these problems can be reformulated as either linear programming, second-order cone programming or semidefinite programming problems, the standard methods for solving these relaxations are not applicable because the problems are usually of huge size and contain dense data. In this dissertation, we give efficient algorithms for solving these "sparse" optimization problems and analyze the convergence and iteration complexity properties of these algorithms.

Chapter 2 presents algorithms for solving the linearly constrained matrix rank minimization problem. The tightest convex relaxation of this problem is the linearly constrained nuclear norm minimization. Although the latter can be cast and solved as a semidefinite

programming problem, such an approach is computationally expensive when the matrices are large. In Chapter 2, we propose fixed-point and Bregman iterative algorithms for solving the nuclear norm minimization problem and prove convergence of the first of these algorithms. By using a homotopy approach together with an approximate singular value decomposition procedure, we get a very fast, robust and powerful algorithm, which we call FPCA (Fixed Point Continuation with Approximate SVD), that can solve very large matrix rank minimization problems. Our numerical results on randomly generated and real matrix completion problems demonstrate that this algorithm is much faster and provides much better recoverability than semidefinite programming solvers such as SDPT3. For example, our algorithm can recover 1000×1000 matrices of rank 50 with a relative error of 10^{-5} in about 3 minutes by sampling only 20 percent of the elements. We know of no other method that achieves as good recoverability. Numerical experiments on online recommendation, DNA microarray data set and image inpainting problems demonstrate the effectiveness of our algorithms.

In Chapter 3, we study the convergence/recoverability properties of the fixed-point continuation algorithm and its variants for matrix rank minimization. Heuristics for determining the rank of the matrix when its true rank is not known are also proposed. Some of these algorithms are closely related to greedy algorithms in compressed sensing. Numerical results for these algorithms for solving linearly constrained matrix rank minimization problems are reported.

Chapters 4 and 5 considers alternating direction type methods for solving composite convex optimization problems. We present in Chapter 4 alternating linearization algorithms

that are based on an alternating direction augmented Lagrangian approach for minimizing the sum of two convex functions. Our basic methods require at most $O(1/\varepsilon)$ iterations to obtain an ε-optimal solution, while our accelerated (i.e., fast) versions require at most $O(1/\sqrt{\varepsilon})$ iterations, with little change in the computational effort required at each iteration. For more general problem, i.e., minimizing the sum of K convex functions, we propose multiple-splitting algorithms for solving them. We propose both basic and accelerated algorithms with $O(1/\varepsilon)$ and $O(1/\sqrt{\varepsilon})$ iteration complexity bounds for obtaining an ε-optimal solution. To the best of our knowledge, the complexity results presented in these two chapters are the first ones of this type that have been given for splitting and alternating direction type methods. Numerical results on various applications in sparse and low-rank optimization, including compressed sensing, matrix completion, image deblurring, robust principal component analysis, are reported to demonstrate the efficiency of our methods.

Contents

List of Figures

List of Tables

Acknowledgements

I feel very fortunate to have had the opportunity to work with my advisor, Donald Goldfarb. Don is an inspiring and creative mentor. His consistent support, encouragement and great sense of humor have made working with him a very joyful experience. Without his help, I could not have finished my dissertation.

I would like to express my sincere appreciation to the members of my dissertation committee, Garud Iyengar, Cliff Stein, Katya Scheinberg and Jonathan Eckstein, for their careful reading and valuable suggestions and comments on my dissertation. I learned a lot from the convex optimization, network flow and machine learning courses taught by Garud, Cliff and Katya. Their lectures really broadened my research scope. I would also like to thank Jonathan for many detailed and insightful comments to my papers and presentations.

I am very grateful to my collaborators I have had the great opportunity to work with in the past five years. Specifically, I want to thank Amit Chakraborty and Wotao Yin for introducing me to the very exiting research area of medical imaging. The joint projects with Katya Scheinberg on machine learning significantly extended my research areas. I would also like to thank Zaiwen Wen, Lifeng Chen, Zhiwei Qin, Bo Huang, Necdet Serhat Aybat, Wei Liu and Jun Wang. The numerous fruitful discussions that we had have stimulated a lot of interesting and important joint papers. It was a privilege to have worked with such a group of talented people.

I would further like to thank people who have helped me during my career search. I especially want to thank Wotao Yin, Yinyu Ye, Yin Zhang, Garud Iyengar and Cliff Stein

for their tremendous unconditional support and help. Special thanks go to the Department faculty and stuff members for providing an enjoyable working environment.

I have made many very good friends during the past five years at Columbia University. I am very grateful to my friends Lifeng Chen, Zaiwen Wen, Zongjian Liu, Ning Cai, Xianhua Peng, Ming Hu, Thiam Hui Lee, Necdet Serhat Aybat, Yori Zwols, Yu Hang Kan, Guodong Pang, Min Wang, Yiping Du, Soonmin Ko, Ruxian Wang, Ranting Yao, Ana Zenteno Langle, Yixi Shi, Pingkai Liu, Shuheng Zheng, Zhiwei Qin, Haowen Zhong, Xingbo Xu, Yina Lu, Xinyun Chen, Jing Dong, Juan Li, Zhen Qiu, Xin Li, Chen Chen, Yupeng Chen, Chun Wang, Yan Liu, Bo Huang, Ningyuan Chen, Yin Lu, for the fruitful discussions, for sharing ideas, and for every joyful and memorable moment we had. Especially, I owe my deepest gratitude to Jing, Haowen, Chun, Xingbo, Yina, Juan, Yupeng, Chen, Zhiwei, Lifeng, Ranting, Zongjian, for accompanying with me and taking very good care of me when I was once in need of helps in the Summer of 2010.

I am indebted to my parents Yinci Ma and Qiaoxia Wang. Their unconditional love and support throughout my life made it possible for me to come to the United States to pursue my career goal. My brother Shihao Ma provided happiness and joy to the family. I thank him for staying close to and taking care of my parents. I wish happiness will always be with my parents and my brother.

Shiqian Ma

June 21, 2011

To my parents and my brother

Chapter 1

Introduction

1.1 Background and Motivation

This dissertation is devoted to algorithms for solving problems with sparse or low-rank optimal solutions. Research in this area was mainly ignited by the recent emergence of compressed sensing (CS) and its matrix extensions such as the matrix completion and robust principal component analysis (PCA) problems. CS enables one to recover a signal or image with fewer observations than the "length" of the signal or image, and thus provides potential breakthroughs in applications where data acquisition is costly. For example, in magnetic resonance imaging (MRI) and computerized tomography (CT) problems, one tries to reduce the data acquisition time since patients suffer from keeping still or from exposure to ionizing radiation. However, the potential impact of CS cannot be realized without efficient optimization algorithms that can handle extremely large-scale and dense data from CS applications. Although the convex relaxations of these problems can be re-

formulated as either linear programming, second-order cone programming or semidefinite programming problems, the standard methods for solving these reformulations are not applicable because the problems are usually of huge size and contain dense data. In this dissertation, we will give efficient algorithms for solving these "sparse" optimization problems and analyze the convergence and iteration complexity properties of these algorithms.

1.2 Preliminaries

1.2.1 Basic Notation

We denote the set of real numbers by $\mathbb{R}$ and the n-dimensional Euclidean space by $\mathbb{R}^n$. The superscript "$\top$" denotes the transpose operation. The inner product of vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ is denoted by $\langle x, y \rangle = x^\top y = \sum_{j=1}^n x_j y_j$. The Euclidean norm of $x \in \mathbb{R}^n$ is denoted by $\|x\|_2 = (x^\top x)^{1/2}$. We use $\|x\|_0$, the so-called ℓ_0 norm of $x \in \mathbb{R}^n$, to denote the number of nonzeros of x. We use $\|x\|_1$ to denote the ℓ_1 norm of x, i.e., $\|x\|_1 = \sum_{j=1}^n |x_j|$. $\|x\|_\infty$ denotes the infinity norm, i.e., the largest component of x in magnitude, i.e., $\|x\|_\infty = \max_j |x_j|$.

We use $\mathbb{R}^{m\times n}$ to denote the Euclidean space of the set of $m \times n$ matrices. $\mathrm{rank}(X)$ denotes the rank of the matrix $X \in \mathbb{R}^{m\times n}$, i.e., the number of nonzero singular values of X. If the rank of matrix X is r and its singular value decomposition (SVD) is given by $X = \sum_{j=1}^r u_j \sigma_j v_j^\top$ with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, we denote by $\|X\|_*$ the nuclear norm of X, which is defined as the sum of singular values of X, i.e., $\|X\|_* = \sum_{j=1}^r \sigma_j$. $\|X\|$ denotes the spectral norm of X, which is equal to the largest singular value of $X \in \mathbb{R}^{m\times n}$,

i.e., $\|X\| = \sigma_1$. The Frobenius norm of $X \in \mathbb{R}^{m\times n}$ is defined as $\|X\|_F = \sqrt{\sum_{i,j} X_{ij}^2}$. We use $\mathrm{vec}(X)$ to denote the vector obtained by stacking the columns of $X \in \mathbb{R}^{m\times n}$ as a long vector. We use $\|X\|_1$ and $\|X\|_\infty$ to denote the norms corresponding to the vector form of X, i.e., $\|X\|_1 := \|\mathrm{vec}(X)\|_1$ and $\|X\|_\infty := \|\mathrm{vec}(X)\|_\infty$.

$\mathrm{Diag}(s)$ denotes the diagonal matrix whose diagonal elements are the elements of the vector s. $\mathrm{sgn}(t)$ is the signum function of $t \in \mathbb{R}$, i.e.,

$$\mathrm{sgn}(t) := \begin{cases} +1 & \text{if } t > 0, \\ 0 & \text{if } t = 0, \\ -1 & \text{if } t < 0, \end{cases}$$

while the signum multifunction of $t \in \mathbb{R}$ is

$$\mathrm{SGN}(t) := \partial|t| = \begin{cases} \{+1\} & \text{if } t > 0, \\ [-1,1] & \text{if } t = 0, \\ \{-1\} & \text{if } t < 0. \end{cases}$$

We use $a \odot b$ to denote the elementwise multiplication of two vectors a and b. We use $X(k:l)$ to denote the submatrix of X consisting of the k-th to l-th columns of X.

Henceforth, we will write the linear map $\mathcal{A}(X)$ as $\mathcal{A}X$ as this should not cause any confusion. For example, $\mathcal{A}^*\mathcal{A}X := \mathcal{A}^*(\mathcal{A}(X))$.

1.2.2 Sparse and Low-Rank Optimization

A fundamental problem in signal processing is to recover a sparse signal from a few measurements. This problem can be formulated as

$$\min_{x\in\mathbb{R}^n} \quad \|x\|_0 \quad \text{s.t.} \quad Ax = b, \tag{1.2.1}$$

where x is the sparse signal one wants to recover, $A \in \mathbb{R}^{m\times n}$ is the sensing matrix, $b \in \mathbb{R}^m$ is the measurement and $\|x\|_0$ counts the number of nonzeros of x. Problem (1.2.1) can be interpreted as seeking the sparsest solution of a linear system. In some applications, the signal itself is not sparse but is sparse under some transform $W \in \mathbb{R}^{n\times n}$. Then the problem can be formulated as

$$\min_{x\in\mathbb{R}^n} \quad \|Wx\|_0 \quad \text{s.t.} \quad Ax = b, \tag{1.2.2}$$

i.e., the vector of coefficients of x under transform W is expected to be sparse. Note that (1.2.2) can be reformulated in the form of (1.2.1) when W is invertible

$$\min_{z} \quad \|z\|_0 \quad \text{s.t.} \quad AW^{-1}z = b.$$

It is known that (1.2.1) is usually NP-hard [73] and thus numerically intractable. Compressed sensing connects the NP-hard problem (1.2.1) to its convex relaxation, the ℓ_1 min-

imization problem:

$$\min_{x\in\mathbb{R}^n} \quad \|x\|_1 \quad \text{s.t.} \quad Ax = b, \tag{1.2.3}$$

i.e., $\|x\|_0$ in the objective function is replaced by its convex envelope $\|x\|_1$ on the unit ball $\{x \in \mathbb{R}^n | \|x\|_\infty \leq 1\}$ (see, e.g., [54]). Compressed sensing theory guarantees that under certain conditions, the optimal solution of the NP-hard problem (1.2.1) is given by the optimal solution of the convex problem (1.2.3) with high probability (see [18, 26]). So now the question is how to solve the convex problem (1.2.3) efficiently. Although (1.2.3) can be reformulated as a linear programming problem and thus solved by interior point methods (IPMs), such methods are not practical since the problems arising from compressed sensing are usually of large scale. Sometimes the measurement b is contaminated by noise, then the constraint $Ax = b$ must be relaxed, resulting in either the problem

$$\min_{x\in\mathbb{R}^n} \quad \|x\|_1 \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \theta$$

or its Lagrangian version

$$\min \quad \rho\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{1.2.4}$$

where θ and ρ are parameters. Many algorithms for solving the cardinality minimization problem (1.2.1) and the ℓ_1 norm minimization problem (1.2.3) have been proposed. These include greedy algorithms [7, 8, 25, 27, 28, 74, 90, 94] for (1.2.1) and convex optimization

algorithms [17, 38, 50, 57, 101, 106] for (1.2.3) and its variant (1.2.4). See [24] for more information on the theory and algorithms for compressed sensing.

The matrix rank minimization problem is a matrix extension of the ℓ_0 norm minimization problem (1.2.1). The matrix rank minimization problem can be written as

$$\min_{X \in \mathbb{R}^{m \times n}} \quad \text{rank}(X) \quad \text{s.t.} \quad \mathcal{A}(X) = b, \tag{1.2.5}$$

where $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^p$ is a linear map. This model has many applications such as determining a low-order controller for a plant [42] and a minimum order linear system realization [36], and solving low-dimensional Euclidean embedding problems [65]. The matrix completion problem

$$\min_{X \in \mathbb{R}^{m \times n}} \quad \text{rank}(X) \quad \text{s.t.} \quad X_{ij} = M_{ij}, \quad \forall (i,j) \in \Omega, \tag{1.2.6}$$

is a special case of (1.2.5), where $M \in \mathbb{R}^{m \times n}$ and Ω is a subset of index pairs (i, j). The so called collaborative filtering problem [82, 87] can be cast as a matrix completion problem. Suppose users in an online survey provide ratings of some movies. This yields a matrix M with users as rows and movies as columns whose (i, j)-th entry M_{ij} is the rating given by the i-th user to the j-th movie. Since most users rate only a small portion of the movies, we typically only know a small subset $\{M_{ij} | (i, j) \in \Omega\}$ of the entries. Based on the known ratings of a user, we want to predict the user's ratings of the movies that the user did not rate; i.e., we want to fill in the missing entries of the matrix. It is commonly believed that

only a few factors contribute to an individual's tastes or preferences for movies. Thus the rating matrix M is likely to be of *numerical* low rank in the sense that relatively few of the top singular values account for most of the sum of all of the singular values. Finding such a low-rank matrix M corresponds to solving the matrix completion problem (1.2.6).

The matrix rank minimization (1.2.5) is NP-hard in general due to the combinational nature of the function $\text{rank}(\cdot)$. Similar to the cardinality function $\|x\|_0$, we can replace $\text{rank}(X)$ by its convex envelope to get a convex and more computationally tractable approximation to (1.2.5). It turns out that the convex envelope of $\text{rank}(X)$ on the set $\{X \in \mathbb{R}^{m\times n} : \|X\| \leq 1\}$ is the nuclear norm $\|X\|_*$ [35], i.e., the nuclear norm is the best convex approximation of the rank function over the unit ball of matrices with spectral norm less than one. Using the nuclear norm as an approximation to $\text{rank}(X)$ in (1.2.5) yields the nuclear norm minimization problem

$$\min_{X\in\mathbb{R}^{m\times n}} \|X\|_* \quad \text{s.t.} \quad \mathcal{A}(X) = b. \tag{1.2.7}$$

As in the compressed sensing problem, if b is contaminated by noise, the constraint $\mathcal{A}(X) = b$ must be relaxed, resulting in either the problem

$$\min \quad \|X\|_* \quad \text{s.t.} \quad \|\mathcal{A}(X) - b\|_2 \leq \theta$$

or its Lagrangian version

$$\min \quad \rho\|X\|_* + \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2, \tag{1.2.8}$$

where θ and ρ are parameters.

It was proved recently by Recht et al. [81] that under certain conditions, the optimal solution of (1.2.5) is given by the solution of (1.2.7). Thus, to solve the NP-hard problem (1.2.5), we only need to solve the convex problem (1.2.7).

For the matrix completion problem (1.2.6), the corresponding nuclear norm minimization problem is

$$\min \quad \|X\|_* \quad \text{s.t.} \quad X_{ij} = M_{ij}, (i,j) \in \Omega. \tag{1.2.9}$$

It was proved recently by Candès and Recht [16] and Candès and Tao [19] that, under certain conditions, (1.2.6) is equivalent to (1.2.9) in the sense that they have the same optimal solutions.

1.2.3 Iteration Complexity and Alternating Direction Methods

Chapters 4 and 5 focus on alternating direction type methods for solving the composite optimization problem

$$\min \quad F(x) \equiv \sum_{j=1}^{K} f_j(x), \tag{1.2.10}$$

where $f_j, j = 1, \cdots, K$ are convex functions such that the following problems are easy to solve for any $\tau > 0$ and $z \in \mathbb{R}^n$ relative to minimizing $F(x)$:

$$\min_x \quad \tau f_j(x) + \frac{1}{2}\|x - z\|^2. \tag{1.2.11}$$

In particular, we are specially interested in cases where solving (1.2.11) takes roughly the same effort as computing the gradient (or a subgradient) of $f_j(x)$. Problems of this type arise in many applications of practical interest, including the followings that arise from sparse and low-rank optimization.

Example 1. ℓ_1 minimization in compressed sensing. The unconstrained compressed sensing problem

$$\min \quad \frac{1}{2}\|Ax - b\|_2^2 + \rho\|x\|_1,$$

is of the form of (1.2.10) with $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ and $g(x) := \rho\|x\|_1$. In this case, the two problems (1.2.11) with $f_j = f$ and $f_j = g$ are easy to solve. Specifically, (1.2.11) with $f_j = f$ reduces to solving a linear system and with $f_j = g$ reduces to a vector shrinkage operation which requires $O(n)$ operations (see e.g., [50]).

Example 2. Nuclear norm minimization. The unconstrained nuclear norm minimization problem

$$\min \quad \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2 + \rho\|X\|_*,$$

is of the form of (1.2.10) with $f(X) = \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2$ and $g(X) = \rho\|X\|_*$. In this case, the problem (1.2.11) with $f_j = f$ reduces to solving a linear system. Problem (1.2.11) with $f_j = g$ has a closed-form solution that is given by matrix shrinkage operation (see e.g., [68]).

Example 3. Robust principal component analysis (RPCA). The RPCA problem seeks to recover a low-rank matrix X from a corrupted matrix M. This problem has many applications in computer vision, image processing and web data ranking (see e.g., [15]), and can be formulated as

$$\min_{X,Y\in\mathbb{R}^{m\times n}} \quad \|X\|_* + \rho\|Y\|_1 \quad \text{s.t.} \quad X + Y = M, \tag{1.2.12}$$

where $\rho > 0$ and $M \in \mathbb{R}^{m\times n}$. Note that (1.2.12) can be rewritten as

$$\min_{X\in\mathbb{R}^{m\times n}} \quad \|X\|_* + \rho\|M - X\|_1,$$

which is of the form of (1.2.10). Moreover, the two problems (1.2.11) with $f_j = \|X\|_*$ and $f_j = \rho\|M - X\|_1$ corresponding to (1.2.12) have closed-form solutions given respectively by a matrix shrinkage operation and a vector shrinkage operation.

Example 4. Sparse inverse covariance selection (SICS). Gaussian graphical models are of great interest in statistical learning. Because conditional independence between different nodes correspond to zero entries in the inverse covariance matrix of the Gaussian distribution, one can learn the structure of the graph by estimating a sparse inverse covari-

ance matrix from sample data by solving the following maximum likelihood problem with an ℓ_1-regularization term, (see e.g., [3, 39, 102, 107]).

$$\max \quad \log\det(X) - \langle \Sigma, X\rangle - \rho\|X\|_1,$$

or equivalently,

$$\min \quad -\log\det(X) + \langle \Sigma, X\rangle + \rho\|X\|_1, \tag{1.2.13}$$

where $\rho > 0$ and $\Sigma \in S_+^n$ (the set of symmetric positive semidefinite matrices) is the sample covariance matrix. Note that by defining $f(X) := -\log\det(X) + \langle \Sigma, X\rangle$ and $g(X) := \rho\|X\|_1$, (1.2.13) is of the form of (1.2.10). Moreover, it can be proved that the problem (1.2.11) with $f_j = f$ has a closed-form solution, which is given by a spectral decomposition, while the solution of problem (1.2.11) with $f_j = g$ corresponds to a vector shrinkage operation.

Example 5. Compressed sensing based MRI. One version of the compressed sensing magnetic resonance imaging (MRI) problem can be cast as minimizing the sum of three convex functions (see [69]):

$$\min_{x\in\mathbb{R}^n} \alpha \mathrm{TV}(x) + \beta\|Wx\|_1 + \frac{1}{2}\|Ax - b\|^2, \tag{1.2.14}$$

where $TV(x)$ is the total variation function, W is a wavelet transform, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $\alpha > 0, \beta > 0$ are weighting parameters. The three problems (1.2.11) corresponding to (1.2.14) are also easy to solve.

For the composite convex optimization problems discussed above, since the problems (1.2.11) are easy to solve, alternating direction methods can be applied to solve them efficiently. Despite of the efficiency of alternating direction methods, iteration complexity bounds for these methods were previously unknown. In Chapters 4 and 5 we propose several alternating direction type methods for solving the composite convex optimization problem (1.2.10). We prove that the iteration complexity of the basic version of these algorithms is $O(1/\varepsilon)$ to obtain an ε-optimal solution. The iteration complexity of our accelerated algorithms is improved to $O(1/\sqrt{\varepsilon})$ for an ε-optimal solution, while the work required at each iteration is almost unchanged compared to the basic algorithms. To the best our knowledge, these iteration complexity results are the first ones of this type that have been given for alternating direction type methods.

Chapter 2

Fixed Point and Bregman Iterative Methods for Matrix Rank Minimization

2.1 Introduction

Let us first consider the relationship between matrix rank minimization problem

$$\min \quad \text{rank}(X) \quad \text{s.t.} \quad \mathcal{A}(X) = b, \tag{2.1.1}$$

and its convex relaxation, the nuclear norm minimization problem

$$\min \quad \|X\|_* \quad \text{s.t.} \quad \mathcal{A}(X) = b, \tag{2.1.2}$$

and note that we can write the linear equations $\mathcal{A}(X) = b$ in (2.1.1) and (2.1.2) as

$$A \cdot \mathrm{vec}(X) = b, \tag{2.1.3}$$

where $A \in \mathbb{R}^{p \times mn}$ is the matrix corresponding to the linear map $\mathcal{A}$. An important question is: when will an optimal solution to the nuclear norm minimization problem (2.1.2) give an optimal solution to matrix rank minimization problem (2.1.1). In response to this question, Recht et al. [81] proved that if the entries of A are suitably random, e.g., i.i.d. Gaussian, then with very high probability, most $m \times n$ matrices of rank r can be recovered by solving the nuclear norm minimization (2.1.2) or equivalently, (2.1.3), whenever $p \geq Cr(m+n)\log(mn)$, where C is a positive constant.

For the matrix completion problem

$$\min \quad \mathrm{rank}(X) \quad \text{s.t.} \quad X_{ij} = M_{ij}, (i,j) \in \Omega, \tag{2.1.4}$$

and its convex relaxation

$$\min \quad \|X\|_* \quad \text{s.t.} \quad X_{ij} = M_{ij}, (i,j) \in \Omega, \tag{2.1.5}$$

Candès et al. [16] proved the following result.

Theorem 2.1.1 (Theorem 1.1 in [16]). *Let M be an $n_1 \times n_2$ matrix of rank r with SVD*

$$M = \sum_{k=1}^{r} \sigma_k u_k v_k^\top,$$

where the family $\{u_k\}_{1\le k\le r}$ is selected uniformly at random among all families of r orthonormal vectors, and similarly for the family $\{v_k\}_{1\le k\le r}$. Let $n = \max(n_1, n_2)$. Suppose we observe m entries of M with locations sampled uniformly at random. Then there are constants C and c such that if

$$m \ge C n^{5/4} r \log n,$$

the minimizer to the problem (2.1.5) *is unique and equal to M with probability at least $1 - cn^{-3}$. In addition, if $r \le n^{1/5}$, then the recovery is exact with probability at least $1 - cn^{-3}$ provided that*

$$m \ge C n^{6/5} r \log n.$$

This theorem states that a surprisingly small number of entries are sufficient to complete a low-rank matrix with high probability. Recently, this result was strengthened by Candès and Tao in [19], where it is proved that under certain incoherence conditions, the number of samples m that are required is only $O(nr\text{polylog}(n))$. A similar result was shown by Keshavan et al. in [56].

The dual problem corresponding to the nuclear norm minimization problem (2.1.2) is

$$\max \quad b^\top z \quad \text{s.t.} \quad \|\mathcal{A}^*(z)\| \le 1, \tag{2.1.6}$$

where $\mathcal{A}^*$ is the adjoint operator of $\mathcal{A}$. Both (2.1.2) and (2.1.6) can be rewritten as equivalent semidefinite programming (SDP) problems. The SDP formulation of (2.1.2) is:

$$\begin{aligned} \min_{X,W_1,W_2} \quad & \tfrac{1}{2}(\mathrm{Tr}(W_1)+\mathrm{Tr}(W_2)) \\ \text{s.t.} \quad & \begin{bmatrix} W_1 & X \\ X^\top & W_2 \end{bmatrix} \succeq 0 \\ & \mathcal{A}(X) = b, \end{aligned} \tag{2.1.7}$$

where $\mathrm{Tr}(X)$ denotes the trace of the square matrix X. The SDP formulation of (2.1.6) is:

$$\begin{aligned} \max_{z} \quad & b^\top z \\ \text{s.t.} \quad & \begin{bmatrix} I_m & \mathcal{A}^*(z) \\ \mathcal{A}^*(z)^\top & I_n \end{bmatrix} \succeq 0. \end{aligned} \tag{2.1.8}$$

Thus to solve (2.1.2) and (2.1.6), we can use SDP solvers such as SeDuMi [89] and SDPT3 [100] to solve (2.1.7) and (2.1.8). Note that the number of variables in (2.1.7) is $\frac{1}{2}(m+n)(m+n+1)$. SDP solvers cannot usually solve a problem when m and n are both much larger than 100.

Recently, Liu and Vandenberghe [67] proposed an interior-point method for another nuclear norm approximation problem

$$\min \|\mathcal{A}(x)-B\|_*, \tag{2.1.9}$$

where $B \in \mathbb{R}^{m\times n}$ and

$$\mathcal{A}(x) = x_1A_1 + x_2A_2 + \cdots + x_pA_p$$

is a linear mapping from $\mathbb{R}^p$ to $\mathbb{R}^{m\times n}$. The equivalent SDP formulation of (2.1.9) is

$$\begin{array}{ll} \min\limits_{x,W_1,W_2} & \frac{1}{2}(\mathrm{Tr}(W_1)+\mathrm{Tr}(W_2)) \\ \text{s.t.} & \begin{bmatrix} W_1 & (\mathcal{A}(x)-B)^\top \\ \mathcal{A}(x)-B & W_2 \end{bmatrix} \succeq 0. \end{array} \tag{2.1.10}$$

Liu and Vandenberghe [67] proposed a customized method for computing the scaling direction in an interior point method for solving the SDP (2.1.10). The complexity of each iteration in their method was reduced from $O(p^6)$ to $O(p^4)$ when $m = O(p)$ and $n = O(p)$; thus they were able to solve problems up to dimension $m = n = 350$.

Another algorithm for solving (2.1.2) is due to Burer and Monteiro [12, 13], (see also Rennie and Srebro [82,87]). This algorithm uses the low-rank factorization $X = LR^\top$ of the matrix $X \in \mathbb{R}^{m\times n}$, where $L \in \mathbb{R}^{m\times r}, R \in \mathbb{R}^{n\times r}, r \le \min\{m,n\}$, and solves the optimization problem

$$\begin{array}{ll} \min\limits_{L,R} & \frac{1}{2}(\|L\|_F^2 + \|R\|_F^2) \\ \text{s.t.} & \mathcal{A}(LR^\top) = b. \end{array} \tag{2.1.11}$$

It is known that as long as r is chosen to be sufficiently larger than the rank of the optimal solution matrix of the nuclear norm problem (2.1.2), this low-rank factorization problem is equivalent to the nuclear norm problem (2.1.2) (see e.g., [81]). The advantage

of this low-rank factorization formulation is that both the objective function and the constraints are differentiable. Thus gradient-based optimization algorithms such as conjugate gradient algorithms and augmented Lagrangian algorithms can be used to solve this problem. However, the constraints in this problem are nonconvex, so one can only be assured of obtaining a local minimizer. Also, how to choose r is still an open question.

One very interesting algorithm is the so called singular value thresholding algorithm (SVT) [14] which appeared almost simultaneously with our work. SVT is inspired by the linearized Bregman algorithms for compressed sensing and ℓ_1-regularized problems. In [14] it is shown that SVT is efficient for large matrix completion problems. However, SVT only works well for very low rank matrix completion problems. For problems where the matrices are not of very low rank, SVT is slow and not robust and often fails.

Our algorithms have some similarity with the SVT algorithm in that they make use of *matrix shrinkage* (see Section 2). However, other than that, they are very different. All of our methods are based on a fixed-point continuation (FPC) algorithm which uses an operator splitting technique for solving the unconstrained variant of (2.1.2),

$$\min \mu\|X\|_* + \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2. \tag{2.1.12}$$

By adopting a Monte Carlo approximate SVD in the FPC algorithm, we get an algorithm, which we call FPCA (Fixed-Point Continuation with Approximate SVD), that usually gets the optimal solution to (2.1.1) even if the condition of Theorem 2.1.1, or those for the affine constrained case, are violated. Moreover, our algorithm is much faster than state-of-

the-art SDP solvers such as SDPT3 applied to (2.1.7). Also, FPCA can recover matrices of moderate rank that cannot be recovered by SDPT3, SVT, etc. with the same number of samples. For example, FPCA can recover matrices of size 1000×1000 and rank 50 with a relative error of 10^{-5} in about 3 minutes by sampling only 20 percent of the matrix elements. We know of no other method that has as good a recoverability property.

Outline. The rest of this chapter is organized as follows. In Section 2.2 we propose the fixed-point iterative algorithm for nuclear norm minimization problems. In Section 2.3 we analyze the convergence property of the fixed-point iterative algorithm. In Section 2.4 we discuss a continuation technique for accelerating the convergence of our algorithm. In Section 2.5 we propose a Bregman iterative algorithm for nuclear norm minimization extending the approach in [106] for compressed sensing to the rank minimization problem. In Section 2.6 we incorporate a Monte-Carlo approximate SVD procedure into our fixed-point continuation algorithm to speed it up and improve its ability to recover low-rank matrices. Numerical results for both synthesized matrices and real problems are given in Section 2.7. We give conclusions in Section 2.8.

2.2 Fixed-point iterative algorithm

Our fixed-point iterative algorithm for solving (2.1.12) is the following simple two-line algorithm:

$$\begin{cases} Y^k = X^k - \tau g(X^k) \\ X^{k+1} = S_{\tau\mu}(Y^k), \end{cases} \tag{2.2.1}$$

where $S_\nu(\cdot)$ is the matrix shrinkage operator which will be defined later.

Our algorithm (2.2.1) is inspired by the fixed-point iterative algorithm proposed in [49] for the ℓ_1-regularized problem (1.2.4). The idea behind this algorithm is an operator splitting technique. Note that x^* is an optimal solution to (1.2.4) if and only if

$$\mathbf{0} \in \mu \mathrm{SGN}(x^*) + g^*, \tag{2.2.2}$$

where $g^* = A^\top(Ax^* - b)$. For any $\tau > 0$, (2.2.2) is equivalent to

$$\mathbf{0} \in \tau\mu \mathrm{SGN}(x^*) + \tau g(x^*). \tag{2.2.3}$$

Note that the operator $T(\cdot) := \tau\mu\mathrm{SGN}(\cdot) + \tau g(\cdot)$ on the right hand side of (2.2.3) can be split into two parts: $T(\cdot) = T_1(\cdot) - T_2(\cdot)$, where $T_1(\cdot) = \tau\mu\mathrm{SGN}(\cdot) + I(\cdot)$ and $T_2(\cdot) = I(\cdot) - \tau g(\cdot)$.

Letting $y = T_2(x^*) = x^* - \tau A^\top(Ax^* - b)$, (2.2.3) is equivalent to

$$\mathbf{0} \in T_1(x^*) - y = \tau\mu \mathrm{SGN}(x^*) + x^* - y. \tag{2.2.4}$$

Note that (2.2.4) is actually the optimality conditions for the following convex problem

$$\min_{x^*} \quad \tau\mu\|x^*\|_1 + \frac{1}{2}\|x^* - y\|_2^2. \tag{2.2.5}$$

This problem has a closed form optimal solution given by the so called vector shrinkage operator:

$$x^* = \tilde{s}_\nu(y),$$

where $\nu = \tau\mu$, and the vector shrinkage operator $\tilde{s}_\nu(\cdot)$ is given by

$$\tilde{s}_\nu(\cdot) = \text{sgn}(\cdot) \odot \max\{|\cdot| - \nu, 0\}. \tag{2.2.6}$$

Thus, the fixed-point iterative algorithm is given by

$$x^{k+1} = \tilde{s}_{\tau\mu}(x^k - \tau g^k). \tag{2.2.7}$$

Hale et al. [49] proved global and finite convergence of this algorithm to the optimal solution of the ℓ_1-regularized problem (1.2.4).

Motivated by this work, we develop a fixed-point iterative algorithm for (2.1.12). Since the objective function in (2.1.12) is convex, X^* is the optimal solution to (2.1.12) if and

only if

$$\mathbf{0} \in \mu\partial\|X^*\|_* + g(X^*), \tag{2.2.8}$$

where $g(X^*) = \mathcal{A}^*(\mathcal{A}(X^*) - b)$. Note that if the Singular Value Decomposition (SVD) of X is $X = U\Sigma V^\top$, where $U \in \mathbb{R}^{m\times r}, \Sigma = \mathrm{Diag}(\sigma) \in \mathbb{R}^{r\times r}, V \in \mathbb{R}^{n\times r}$, then (see e.g., [2, 9])

$$\partial\|X\|_* = \{UV^\top + W : U^\top W = 0, WV = 0, \|W\|_2 \le 1\}.$$

Hence, we get the following optimality conditions for (2.1.12):

Theorem 2.2.1. *The matrix $X \in \mathbb{R}^{m\times n}$ with singular value decomposition $X = U\Sigma V^\top$, $U \in \mathbb{R}^{m\times r}, \Sigma = Diag(\sigma) \in \mathbb{R}^{r\times r}, V \in \mathbb{R}^{n\times r}$, is optimal for the problem* (2.1.12) *if and only if there exists a matrix $W \in \mathbb{R}^{m\times n}$ such that*

$$\mu(UV^\top + W) + g(X) = 0, \tag{2.2.9a}$$

$$U^\top W = 0, WV = 0, \|W\|_2 \le 1. \tag{2.2.9b}$$

Now based on the optimality conditions (2.2.8), we can develop a fixed-point iterative scheme for solving (2.1.12) by adopting the operator splitting technique described at the beginning of this section. Note that (2.2.8) is equivalent to

$$\mathbf{0} \in \tau\mu\partial\|X^*\|_* + X^* - (X^* - \tau g(X^*)) \tag{2.2.10}$$

for any $\tau > 0$. If we let

$$Y^* = X^* - \tau g(X^*),$$

then (2.2.10) is reduced to

$$\mathbf{0} \in \tau\mu\partial\|X^*\|_* + X^* - Y^*, \tag{2.2.11}$$

i.e., X^* is the optimal solution to

$$\min_{X\in\mathbb{R}^{m\times n}} \tau\mu\|X\|_* + \frac{1}{2}\|X - Y^*\|_F^2 \tag{2.2.12}$$

In the following we will prove that the matrix shrinkage operator applied to Y^* gives the optimal solution to (2.2.12). First, we need the following definitions.

Definition 2.2.2 (**Nonnegative Vector Shrinkage Operator).** *Assume $x \in \mathbb{R}^n_+$. For any $\nu > 0$, the nonnegative vector shrinkage operator $s_\nu(\cdot)$ is defined as*

$$s_\nu(x) := \bar{x}, \text{ with } \bar{x}_i = \begin{cases} x_i - \nu, & \text{if } x_i - \nu > 0 \\ 0, & \text{o.w.} \end{cases}$$

Definition 2.2.3 (**Matrix Shrinkage Operator).** *Assume $X \in \mathbb{R}^{m\times n}$ and the SVD of X is given by $X = UDiag(\sigma)V^\top$, $U \in \mathbb{R}^{m\times r}, \sigma \in \mathbb{R}^r_+, V \in \mathbb{R}^{n\times r}$. For any $\nu > 0$, the matrix*

shrinkage operator $S_\nu(\cdot)$ *is defined as*

$$S_\nu(X) := UDiag(\bar{\sigma})V^\top, \qquad with\ \bar{\sigma} = s_\nu(\sigma).$$

Theorem 2.2.4. *Given a matrix* $Y \in \mathbb{R}^{m\times n}$ *with* $rank(Y) = t$*, let its Singular Value Decomposition (SVD) be* $Y = U_Y Diag(\gamma)V_Y^\top$*, where* $U_Y \in \mathbb{R}^{m\times t}, \gamma \in \mathbb{R}^t_+, V_Y \in \mathbb{R}^{n\times t}$*. Then for any scalar* $\nu > 0$*,*

$$X := S_\nu(Y) = U_Y Diag(s_\nu(\gamma))V_Y^\top \tag{2.2.13}$$

is an optimal solution of the problem

$$\min_{X\in\mathbb{R}^{m\times n}} f(X) := \nu\|X\|_* + \frac{1}{2}\|X-Y\|_F^2. \tag{2.2.14}$$

Proof. Without loss of generality, we assume $m \le n$. Suppose that the solution $X \in \mathbb{R}^{m\times n}$ to problem (2.2.14) has the SVD $X = U\mathrm{Diag}(\sigma)V^\top$, where $U \in \mathbb{R}^{m\times r}, \sigma \in \mathbb{R}^r_+, V \in \mathbb{R}^{n\times r}$. Hence, X must satisfy the optimality conditions for (2.2.14) which are

$$\mathbf{0} \in \nu\partial\|X\|_* + X - Y;$$

i.e., there exists a matrix

$$W = \bar{U}\begin{bmatrix}\mathrm{Diag}(\bar{\sigma}) & 0\end{bmatrix}\bar{V}^\top,$$

where $\bar{U} \in \mathbb{R}^{m\times(m-r)}, \bar{V} \in \mathbb{R}^{n\times(n-r)}, \bar{\sigma} \in \mathbb{R}^{m-r}_+$, $\|\bar{\sigma}\|_\infty \le 1$ and both $\hat{U} = [U, \bar{U}]$ and $\hat{V} =$

$[V, \bar{V}]$ are orthogonal matrices, such that

$$\mathbf{0} = \nu(UV^\top + W) + X - Y. \tag{2.2.15}$$

Hence,

$$\hat{U} \begin{bmatrix} \nu I + \mathrm{Diag}(\sigma) & 0 & 0 \\ 0 & \nu \mathrm{Diag}(\bar{\sigma}) & 0 \end{bmatrix} \hat{V}^\top - U_Y \mathrm{Diag}(\gamma) V_Y^\top = \mathbf{0}. \tag{2.2.16}$$

To verify that (2.2.13) satisfies (2.2.16), consider the following two cases:

Case 1: $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_t > \nu$. In this case, choosing X as above, with $r = t, U = U_Y, V = V_Y$ and $\sigma = s_\nu(\gamma) = \gamma - \nu e$, where e is a vector of r ones, and choosing $\bar{\sigma} = 0$ (i.e., $W = \mathbf{0}$) satisfies (2.2.16).

Case 2: $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_k > \nu \geq \gamma_{k+1} \geq \ldots \geq \gamma_t$. In this case, by choosing $r = k, \hat{U}(1:t) = U_Y, \hat{V}(1:t) = V_Y, \sigma = s_\nu((\gamma_1, \ldots, \gamma_k))$ and $\bar{\sigma}_1 = \gamma_{k+1}/\nu, \ldots, \bar{\sigma}_{t-k} = \gamma_t/\nu, \bar{\sigma}_{t-k+1} = \ldots = \bar{\sigma}_{m-r} = 0$, X and W satisfy (2.2.16).

Note that in both cases, X can be written as the form in (2.2.13) based on the way we construct X. □

Based on the above we obtain the fixed-point iterative scheme (2.2.1) stated at the beginning of this section for solving problem (2.1.12).

Moreover, from the discussion following Theorem 2.2.1 we have

Corollary 2.2.5. *X^* is an optimal solution to problem* (2.1.12) *if and only if $X^* = S_{\tau\mu}(h(X^*))$, where $h(\cdot) = I(\cdot) - \tau g(\cdot)$.*

2.3 Convergence results

In this section, we analyze the convergence properties of the fixed-point iterative scheme (2.2.1). Before we prove the main convergence result, we need some lemmas.

Lemma 2.3.1. *The shrinkage operator S_ν is non-expansive, i.e., for any Y_1 and $Y_2 \in \mathbb{R}^{m\times n}$,*

$$\|S_\nu(Y_1) - S_\nu(Y_2)\|_F \leq \|Y_1 - Y_2\|_F. \tag{2.3.1}$$

Moreover,

$$\|Y_1 - Y_2\|_F = \|S_\nu(Y_1) - S_\nu(Y_2)\|_F \Longleftrightarrow Y_1 - Y_2 = S_\nu(Y_1) - S_\nu(Y_2). \tag{2.3.2}$$

Proof. Without loss of generality, we assume $m \leq n$. Assume SVDs of Y_1 and Y_2 are $Y_1 = U_1\Sigma V_1^\top$ and $Y_2 = U_2\Gamma V_2^\top$, respectively, where

$$\Sigma = \begin{pmatrix} \mathrm{Diag}(\sigma) & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n}, \Gamma = \begin{pmatrix} \mathrm{Diag}(\gamma) & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n},$$

$\sigma = (\sigma_1, \ldots, \sigma_s), \sigma_1 \geq \ldots \geq \sigma_s > 0$ and $\gamma = (\gamma_1, \ldots, \gamma_t), \gamma_1 \geq \ldots \geq \gamma_t > 0$. Note that here U_1, V_1, U_2 and V_2 are (full) orthogonal matrices; $\Sigma, \Gamma \in \mathbb{R}^{m\times n}$. Suppose that $\sigma_1 \geq \ldots \geq \sigma_k \geq \nu > \sigma_{k+1} \geq \ldots \geq \sigma_s$ and $\gamma_1 \geq \ldots \geq \gamma_l \geq \nu > \gamma_{l+1} \geq \ldots \geq \gamma_t$, then

$$\bar{Y}_1 := S_\nu(Y_1) = U_1\bar{\Sigma}V_1^\top, \bar{Y}_2 := S_\nu(Y_2) = U_2\bar{\Gamma}V_2^\top,$$

where

$$\bar{\Sigma} = \begin{pmatrix} \mathrm{Diag}(\bar{\sigma}) & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n}, \bar{\Gamma} = \begin{pmatrix} \mathrm{Diag}(\bar{\gamma}) & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n},$$

$\bar{\sigma} = (\sigma_1 - \nu, \ldots, \sigma_k - \nu)$ and $\bar{\gamma} = (\gamma_1 - \nu, \ldots, \gamma_l - \nu)$. Thus,

$$\begin{aligned} & \|Y_1 - Y_2\|_F^2 - \|\bar{Y}_1 - \bar{Y}_2\|_F^2 \\ = \; & \mathrm{Tr}((Y_1 - Y_2)^\top (Y_1 - Y_2)) - \mathrm{Tr}((\bar{Y}_1 - \bar{Y}_2)^\top (\bar{Y}_1 - \bar{Y}_2)) \\ = \; & \mathrm{Tr}(Y_1^\top Y_1 - \bar{Y}_1^\top \bar{Y}_1 + Y_2^\top Y_2 - \bar{Y}_2^\top \bar{Y}_2) - 2\mathrm{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2) \\ = \; & \sum_{i=1}^{s} \sigma_i^2 - \sum_{i=1}^{k} (\sigma_i - \nu)^2 + \sum_{i=1}^{t} \gamma_i^2 - \sum_{i=1}^{l} (\gamma_i - \nu)^2 - 2\mathrm{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2). \end{aligned}$$

We note that

$$\begin{aligned} & \mathrm{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2) \\ = \; & \mathrm{Tr}((Y_1 - \bar{Y}_1)^\top (Y_2 - \bar{Y}_2) + (Y_1 - \bar{Y}_1)^\top \bar{Y}_2 + \bar{Y}_1{}^\top (Y_2 - \bar{Y}_2)) \\ = \; & \mathrm{Tr}(V_1(\Sigma - \bar{\Sigma})^\top U_1^\top U_2 (\Gamma - \bar{\Gamma}) V_2^\top + V_1(\Sigma - \bar{\Sigma})^\top U_1^\top U_2 \bar{\Gamma} V_2^\top + V_1 \bar{\Sigma}^\top U_1^\top U_2 (\Gamma - \bar{\Gamma}) V_2^\top \\ = \; & \mathrm{Tr}((\Sigma - \bar{\Sigma})^\top U (\Gamma - \bar{\Gamma}) V^\top + (\Sigma - \bar{\Sigma})^\top U \bar{\Gamma} V^\top + \bar{\Sigma}^\top U (\Gamma - \bar{\Gamma}) V^\top), \end{aligned}$$

where $U = U_1^\top U_2, V = V_1^\top V_2$ are clearly orthogonal matrices. Now let us derive an upper bound for $\mathrm{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2)$. It is known that an orthogonal matrix U is a maximizing matrix for the problem

$$\max\{\mathrm{Tr}(AU) : U \text{ is orthogonal}\}$$

if and only if AU is positive semidefinite matrix (see 7.4.9 in [55]). It is also known that

when AB is positive semidefinite,

$$\mathrm{Tr}(AB) = \sum_i \sigma_i(AB) \leq \sum_i \sigma_i(A)\sigma_i(B). \tag{2.3.3}$$

Thus, $\mathrm{Tr}((\Sigma-\bar{\Sigma})^\top U(\Gamma-\bar{\Gamma})V^\top)$, $\mathrm{Tr}((\Sigma-\bar{\Sigma})^\top U\bar{\Gamma}V^\top)$ and $\mathrm{Tr}(\bar{\Sigma}U(\Gamma-\bar{\Gamma})V^\top)$ achieve their maximum, if and only if $(\Sigma-\bar{\Sigma})^\top U(\Gamma-\bar{\Gamma})V^\top$, $(\Sigma-\bar{\Sigma})^\top U\bar{\Gamma}V^\top$ and $\bar{\Sigma}U(\Gamma-\bar{\Gamma})V^\top$ are all positive semidefinite. Applying (2.3.3) to these three terms, we get $\mathrm{Tr}((\Sigma-\bar{\Sigma})^\top U(\Gamma-\bar{\Gamma})V^\top) \leq \sum_i \sigma_i(\Sigma-\bar{\Sigma})\sigma_i(\Gamma-\bar{\Gamma})$, $\mathrm{Tr}((\Sigma-\bar{\Sigma})^\top U\bar{\Gamma}V^\top) \leq \sum_i \sigma_i(\Sigma-\bar{\Sigma})\sigma_i(\bar{\Gamma})$ and $\mathrm{Tr}(\bar{\Sigma}U(\Gamma-\bar{\Gamma})V^\top) \leq \sum_i \sigma_i(\bar{\Sigma})\sigma_i(\Gamma-\bar{\Gamma})$. Thus, without loss of generality, assuming $k \leq l \leq s \leq t$, we have,

$$\begin{aligned}
& \|Y_1-Y_2\|_F^2 - \|S_\nu(Y_1)-S_\nu(Y_2)\|_F^2 \\
\geq\ & \sum_{i=1}^{s}\sigma_i^2 - \sum_{i=1}^{k}(\sigma_i-\nu)^2 + \sum_{i=1}^{t}\gamma_i^2 - \sum_{i=1}^{l}(\gamma_i-\nu)^2 \\
& -2(\textstyle\sum_{i=1}^{l}\sigma_i\nu + \sum_{i=l+1}^{s}\sigma_i\gamma_i + \sum_{i=1}^{k}(\gamma_i-\nu)\nu + \sum_{i=k+1}^{l}\sigma_i(\gamma_i-\nu)) \\
=\ & \sum_{i=k+1}^{l}(2\gamma_i\nu-\nu^2+\sigma_i^2-2\sigma_i\gamma_i) + (\sum_{i=l+1}^{s}\sigma_i^2 + \sum_{i=l+1}^{t}\gamma_i^2 - \sum_{i=l+1}^{s}2\sigma_i\gamma_i).
\end{aligned}$$

Now

$$\sum_{i=l+1}^{s}\sigma_i^2 + \sum_{i=l+1}^{t}\gamma_i^2 - \sum_{i=l+1}^{s}2\sigma_i\gamma_i \geq 0$$

since $t \geq s$ and $\sigma_i^2+\gamma_i^2-2\sigma_i\gamma_i \geq 0$. Also, since the function $f(x) := 2\gamma_i x - x^2$ is monotonically increasing in $(-\infty,\gamma_i]$ and $\sigma_i < \nu \leq \gamma_i, i = k+1,\ldots,l$,

$$2\gamma_i\nu-\nu^2+\sigma_i^2-2\sigma_i\gamma_i > 0, i = k+1,\ldots,l.$$

Thus we get

$$D(Y_1,Y_2) := \|Y_1 - Y_2\|_F^2 - \|S_\nu(Y_1) - S_\nu(Y_2)\|_F^2 \geq 0;$$

i.e., (2.3.1) holds.

Also, $D(Y_1,Y_2)$ achieves its minimum value if and only if $\mathrm{Tr}((\Sigma-\bar{\Sigma})^\top U(\Gamma-\bar{\Gamma})V^\top)$, $\mathrm{Tr}((\Sigma-\bar{\Sigma})^\top U\bar{\Gamma}V^\top)$ and $\mathrm{Tr}(\bar{\Sigma}U(\Gamma-\bar{\Gamma})V^\top)$ achieve their maximum values simultaneously.

Furthermore, if equality in (2.3.1) holds, i.e., $D(Y_1,Y_2)$ achieves its minimum, and its minimum is zero, then $k=l$, $s=t$, and $\sigma_i=\gamma_i, i=k+1,\ldots,s$, which further implies $\Sigma-\bar{\Sigma}=\Gamma-\bar{\Gamma}$ and $\mathrm{Tr}((\Sigma-\bar{\Sigma})^\top U(\Gamma-\bar{\Gamma})V^\top)$ achieves its maximum. By applying the result 7.4.13 in [55], we get

$$\Sigma-\bar{\Sigma} = U(\Gamma-\bar{\Gamma})V^\top,$$

which further implies that

$$Y_1 - Y_2 = S_\nu(Y_1) - S_\nu(Y_2). \tag{2.3.4}$$

To conclude, clearly $\|S_\nu(Y_1) - S_\nu(Y_2)\|_F = \|Y_1 - Y_2\|_F$ if (2.3.4) holds. □

The following two lemmas and theorem and their proofs are analogous to results and their proofs in Hale et al. [49].

Lemma 2.3.2. *Let $\mathcal{A}X = Avec(X)$ and assume that $\tau \in (0, 2/\lambda_{max}(A^\top A))$. Then the operator $h(\cdot) = I(\cdot) - \tau g(\cdot)$ is non-expansive, i.e., $\|h(X) - h(X')\|_F \leq \|X - X'\|_F$. Moreover, $h(X) - h(X') = X - X'$ if and only if $\|h(X) - h(X')\|_F = \|X - X'\|_F$.*

Proof. First, we note that since $\tau \in (0, 2/\lambda_{max}(A^\top A))$, $-1 < \lambda_i(I - \tau A^\top A) \leq 1, \forall i$, where $\lambda_i(I - \tau A^\top A)$ is the i-th eigenvalue of $I - \tau A^\top A$. Hence,

$$\begin{aligned}\|h(X) - h(X')\|_F &= \|(I - \tau A^\top A)(\mathrm{vec}(X) - \mathrm{vec}(X'))\|_2 \leq \|I - \tau A^\top A\|_2 \|\mathrm{vec}(X) - \mathrm{vec}(X')\|_2 \\ &\leq \|\mathrm{vec}(X) - \mathrm{vec}(X')\|_2 = \|X - X'\|_F.\end{aligned}$$

Moreover, $\|h(X) - h(X')\|_F = \|X - X'\|_F$ if and only if the inequalities above are equalities, which happens if and only if

$$(I - \tau A^\top A)(\mathrm{vec}(X) - \mathrm{vec}(X')) = \mathrm{vec}(X) - \mathrm{vec}(X'),$$

i.e., if and only if $h(X) - h(X') = X - X'$. □

Lemma 2.3.3. *Let X^* be an optimal solution to problem* (2.1.12)*, $\tau \in (0, 2/\lambda_{max}(A^\top A))$ and $\nu = \tau\mu$. Then X is also an optimal solution to problem* (2.1.12) *if and only if*

$$\|S_\nu(h(X)) - S_\nu(h(X^*))\|_F \equiv \|S_\nu(h(X)) - X^*\|_F = \|X - X^*\|_F. \tag{2.3.5}$$

Proof. The "only if" part is an immediate consequence of Corollary 2.2.5. For the "if" part, from Lemmas 2.3.1 and 2.3.2,

$$\|X - X^*\|_F = \|S_\nu(h(X)) - S_\nu(h(X^*))\|_F \leq \|h(X) - h(X^*)\|_F \leq \|X - X^*\|_F.$$

Hence, both inequalities hold with equality. Therefore, first using Lemma 2.3.1 and then

Lemma 2.3.2 we obtain

$$S_\nu(h(X)) - S_\nu(h(X^*)) = h(X) - h(X^*) = X - X^*,$$

which implies $S_\nu(h(X)) = X$ since $S_\nu(h(X^*)) = X^*$. It then follows from Corollary 2.2.5 that X is an optimal solution to problem (2.1.12). □

We now claim that the fixed-point iterations (2.2.1) converge to an optimal solution of problem (2.1.12). Similar convergence results for the problem of finding a zero of the sum of two maximal monotone operators can be found in [31, 40, 97].

Theorem 2.3.4. *The sequence $\{X^k\}$ generated by the fixed-point iterations with $\tau \in (0, 2/\lambda_{max}(A^\top A))$ converges to some $X^* \in \mathcal{X}^*$, where $\mathcal{X}^*$ is the set of optimal solutions of problem* (2.1.12)*.*

Proof. Since both $S_\nu(\cdot)$ and $h(\cdot)$ are non-expansive, $S_\nu(h(\cdot))$ is also non-expansive. Therefore, $\{X^k\}$ lies in a compact set and must have a limit point, say $\bar{X} = \lim_{j\to\infty} X^{k_j}$. Also, for any $X^* \in \mathcal{X}^*$,

$$\|X^{k+1} - X^*\|_F = \|S_\nu(h(X^k)) - S_\nu(h(X^*))\|_F \le \|h(X^k) - h(X^*)\|_F \le \|X^k - X^*\|_F,$$

which means that the sequence $\{\|X^k - X^*\|_F\}$ is monotonically non-increasing. Therefore,

$$\lim_{k\to\infty} \|X^k - X^*\|_F = \|\bar{X} - X^*\|_F, \tag{2.3.6}$$

where $\bar{X}$ can be any limit point of $\{X^k\}$. By the continuity of $S_\nu(h(\cdot))$, the image of $\bar{X}$,

$$S_\nu(h(\bar{X})) = \lim_{j\to\infty} S_\nu(h(X^{k_j})) = \lim_{j\to\infty} X^{k_j+1},$$

is also a limit point of $\{X^k\}$. Therefore, we have

$$\|S_\nu(h(\bar{X})) - S_\nu(h(X^*))\|_F = \|S_\nu(h(\bar{X})) - X^*\|_F = \|\bar{X} - X^*\|_F,$$

which allows us to apply Lemma 2.3.3 to get that $\bar{X}$ is an optimal solution to problem (2.1.12).

Finally, by setting $X^* = \bar{X} \in \mathcal{X}^*$ in (2.3.6), we get that

$$\lim_{k\to\infty} \|X^k - \bar{X}\|_F = \lim_{j\to\infty} \|X^{k_j} - \bar{X}\|_F = 0,$$

i.e., $\{X^k\}$ converges to its unique limit point $\bar{X}$. □

2.4 Fixed-point continuation

In this section, we discuss a continuation technique (i.e., homotopy approach) for accelerating the convergence of the fixed-point iterative algorithm (2.2.1).

2.4.1 Continuation

Inspired by the work of Hale et al. [49], we first describe a continuation technique to accelerate the convergence of the fixed-point iteration (2.2.1). Our fixed-point continuation (FPC) iterative scheme for solving (2.1.12) is outlined below. The parameter η_μ determines the rate of reduction of the consecutive μ_k, i.e.,

$$\mu_{k+1} = \max\{\mu_k \eta_\mu, \bar{\mu}\}, \qquad k = 1, \ldots, L-1$$

Algorithm 2.1 Fixed-point Continuation (FPC)

Initialize: Given X_0, $\bar{\mu} > 0$. Select $\mu_1 > \mu_2 > \cdots > \mu_L = \bar{\mu} > 0$. Set $X = X_0$.
for $\mu = \mu_1, \mu_2, \ldots, \mu_L$ **do**
  while NOT converged **do**
    select $\tau > 0$
    compute $Y = X - \tau \mathcal{A}^*(\mathcal{A}(X) - b)$, and SVD of Y, $Y = U\mathrm{Diag}(\sigma)V^\top$
    compute $X = U\mathrm{Diag}(s_{\tau\mu}(\sigma))V^\top$
  end while
end for

2.4.2 Stopping criteria for inner iterations

Note that in the fixed-point continuation algorithm, in the k-th inner iteration we solve problem (2.1.12) for a fixed $\mu = \mu_k$. There are several ways to determine when to stop this inner iteration, decrease μ and go to the next inner iteration. The optimality conditions for (2.1.12) is given by (2.2.9a) and (2.2.9b). Thus we can use the following condition as a stopping criterion:

$$\|U_k V_k^\top + g^k/\mu\|_2 - 1 < gtol, \tag{2.4.1}$$

where $gtol$ is a small positive parameter. However, the expense of computing the largest singular value of a large matrix greatly decreases the speed of the algorithm. Hence, we do not use this criterion as a stopping rule for large matrices. Instead, we use the criterion

$$\frac{\|X^{k+1} - X^k\|_F}{\max\{1, \|X^k\|_F\}} < xtol, \tag{2.4.2}$$

where $xtol$ is a small positive number, since when X^k gets close to an optimal solution X^*, the distance between X^k and X^{k+1} should become very small.

2.4.3 Debiasing

Debiasing is another technique that can improve the performance of FPC. Debiasing has been used in compressed sensing algorithms for solving (1.2.3) and its variants, where debiasing is performed after a support set I has been tentatively identified. Debiasing is the process of solving a least squares problem restricted to the support set I, i.e., we solve

$$\min \quad \|A_I x_I - b\|_2, \tag{2.4.3}$$

where A_I is a submatrix of A whose columns correspond to the support index set I, and x_I is a subvector of x corresponding to I.

Our debiasing procedure for the matrix completion problem differs from the procedure used in compressed sensing since the concept of a support set is not applicable. When we do debiasing, we fix the matrices U^k and V^k in the singular value decomposition of X^k and

then solve a least squares problem to determine the correct singular values $\sigma \in \mathbb{R}^r_+$; i.e., we solve

$$\min_{\sigma \geq 0} \quad \|\mathcal{A}(U^k \mathrm{Diag}(\sigma) V^{k^\top}) - b\|_2, \tag{2.4.4}$$

where r is the rank of current matrix X^k. Because debiasing can be costly, we use a rule proposed in [104] to decide when to do it. In the continuation framework, we know that in each subproblem with a fixed μ, $\|X_{k+1} - X_k\|_F$ converges to zero, and $\|g\|_2$ converges to μ when X_k converges to the optimal solution of the subproblem. We therefore choose to do debiasing when $\|g\|_2/\|X_{k+1} - X_k\|_F$ becomes large because this indicates that the change between two consecutive iterates is relatively small. Specifically, we call for debiasing in the solver FPC3 (see Section 2.7) when $\|g\|_2/\|X_{k+1} - X_k\|_F > 10$.

2.5 Bregman iterative algorithm

Algorithm FPC is designed to solve (2.1.12), an optimal solution of which approaches an optimal solution of the nuclear norm minimization problem (2.1.2) as μ goes to zero. However, by incorporating FPC into a Bregman iterative technique, we can solve (2.1.2) by solving a limited number of instances of (2.1.12), each corresponding to a different b.

Given a convex function $J(\cdot)$, the Bregman distance [11] of the point u from the point v is defined as

$$D_J^p(u,v) := J(u) - J(v) - < p, u - v >, \tag{2.5.1}$$

where $p \in \partial J(v)$ is some subgradient in the subdifferential of J at the point v.

Bregman iterative regularization was introduced by Osher et al. in the context of image processing [79]. Specifically, in [79], the Rudin-Osher-Fatemi [83] model

$$u = \text{argmin}_u \quad \mu \int |\nabla u| + \frac{1}{2}\|u - b\|_2^2 \tag{2.5.2}$$

was extended to an iterative regularization model by replacing the total variation functional

$$J(u) = \mu TV(u) = \mu \int |\nabla u|,$$

by the Bregman distance with respect to $J(u)$. This Bregman iterative regularization procedure recursively solves

$$u^{k+1} \leftarrow \min_u D_J^{p^k}(u, u^k) + \frac{1}{2}\|u - b\|_2^2 \tag{2.5.3}$$

for $k = 0, 1, \ldots$ starting with $u^0 = \mathbf{0}$ and $p^0 = \mathbf{0}$. Since (2.5.3) is a convex programming problem, the optimality conditions are given by $\mathbf{0} \in \partial J(u^{k+1}) - p^k + u^{k+1} - b$, from which we get the update formula for p^{k+1} :

$$p^{k+1} := p^k + b - u^{k+1}. \tag{2.5.4}$$

Therefore, the Bregman iterative scheme is given by

$$\begin{cases} u^{k+1} \leftarrow \min_u D_J^{p^k}(u, u^k) + \dfrac{1}{2}\|u-b\|_2^2 \\ p^{k+1} = p^k + b - u^{k+1}. \end{cases} \tag{2.5.5}$$

Interestingly, this turns out to be equivalent to the iterative process

$$\begin{cases} b^{k+1} = b + (b^k - u^k) \\ u^{k+1} \leftarrow \min_u J(u) + \dfrac{1}{2}\|u-b^{k+1}\|_2^2, \end{cases} \tag{2.5.6}$$

which can be easily implemented using existing algorithms for (2.5.2) with different inputs b. Interestingly, Bregman methods (2.5.5) and (2.5.6) were later found to be equivalent to the classical augmented Lagrangian method (see, e.g., [106]).

Subsequently, Yin et al. [106] proposed solving the basis pursuit problem (1.2.3) by applying the Bregman iterative regularization algorithm to

$$\min_x \quad J(x) + \frac{1}{2}\|Ax-b\|_2^2$$

for $J(x) = \mu\|x\|_1$, and obtained the following two equivalent iterative schemes analogous to (2.5.5) and (2.5.6), respectively:

- Version 1:
  - $x^0 \leftarrow \mathbf{0}, p^0 \leftarrow \mathbf{0}$,
  - for $k = 0, 1, \ldots$ do

- $x^{k+1} \leftarrow \text{argmin}_x D_J^{p^k}(x, x^k) + \frac{1}{2}\|Ax - b\|_2^2$
- $p^{k+1} \leftarrow p^k - A^\top(Ax^{k+1} - b)$

- Version 2:
  - $b^0 \leftarrow \mathbf{0}, x^0 \leftarrow \mathbf{0}$,
  - for $k = 0, 1, \ldots$ do
  - $b^{k+1} \leftarrow b + (b^k - Ax^k)$
  - $x^{k+1} \leftarrow \text{argmin}_x J(x) + \frac{1}{2}\|Ax - b^{k+1}\|_2^2$.

One can also use the Bregman iterative regularization algorithm applied to the unconstrained problem (2.1.12) to solve the nuclear norm minimization problem (2.1.2). That is, one iteratively solves (2.1.12) by

$$X^{k+1} \leftarrow \min_X D_J^{p^k}(X, X^k) + \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2, \tag{2.5.7}$$

and updates the subgradient p^{k+1} by

$$p^{k+1} := p^k - \mathcal{A}^*(\mathcal{A}(X^{k+1}) - b), \tag{2.5.8}$$

where $J(X) = \mu\|X\|_*$.

Equivalently, one can also use the following iterative scheme:

$$\begin{cases} b^{k+1} \leftarrow b + (b^k - \mathcal{A}(X^k)) \\ X^{k+1} \leftarrow \arg\min_X \mu\|X\|_* + \frac{1}{2}\|\mathcal{A}(X) - b^{k+1}\|_2^2. \end{cases} \tag{2.5.9}$$

Thus, our Bregman iterative algorithm for nuclear norm minimization (2.1.2) can be outlined as follows. The last step can be solved by Algorithm FPC.

Algorithm 2.2 Bregman Iterative Algorithm

$b^0 \leftarrow \mathbf{0}, X^0 \leftarrow \mathbf{0}$,
for $k = 0, 1, \ldots$ **do**
  $b^{k+1} \leftarrow b + (b^k - \mathcal{A}(X^k))$,
  $X^{k+1} \leftarrow \arg\min_X \mu\|X\|_* + \frac{1}{2}\|\mathcal{A}(X) - b^{k+1}\|_2^2$.
end for

2.6 An approximate SVD based FPC algorithm: FPCA

Computing singular value decompositions is the main computational cost in Algorithm FPC. Consequently, instead of computing the full SVD of the matrix Y in each iteration, we implemented a variant of algorithm FPC in which we compute only a rank-k_s approximation to Y, where k_s is a heuristically determined parameter. We call this approximate SVD based FPC algorithm (FPCA). This approach greatly reduces the computational effort required by the algorithm. Specifically, we compute an approximate SVD by a fast Monte Carlo algorithm: the Linear Time SVD algorithm developed by Drineas et al. [30]. For a given matrix $A \in \mathbb{R}^{m\times n}$, and parameters $c_s, k_s \in \mathbb{Z}^+$ with $1 \le k_s \le c_s \le n$ and $\{p_i\}_{i=1}^n$, $p_i \ge 0, \sum_{i=1}^n p_i = 1$, this algorithm returns an approximation to the largest k_s singular values

and corresponding left singular vectors of the matrix A in linear $O(m+n)$ time. The Linear Time SVD Algorithm is outlined below.

Algorithm 2.3 Linear Time Approximate SVD Algorithm [30]

Input: $A \in \mathbb{R}^{m\times n}$, $c_s, k_s \in \mathbb{Z}^+$ s.t.$1 \le k_s \le c_s \le n$, $\{p_i\}_{i=1}^n$ s.t.$p_i \ge 0, \sum_{i=1}^n p_i = 1$.
Output: $H_k \in \mathbb{R}^{m\times k_s}$ and $\sigma_t(C), t = 1,\ldots,k_s$.
for $t = 1$ to c_s **do**
  Pick $i_t \in 1,\ldots,n$ with $Pr[i_t = \alpha] = p_\alpha, \alpha = 1,\ldots,n$.
  Set $C^{(t)} = A^{(i_t)}/\sqrt{c_s p_{i_t}}$.
end for
Compute $C^\top C$ and its SVD; say $C^\top C = \sum_{t=1}^{c_s} \sigma_t^2(C) y^t y^{t\top}$.
Compute $h^t = Cy^t/\sigma_t(C)$ for $t = 1,\ldots,k_s$.
Return H_{k_s}, where $H_{k_s}^{(t)} = h^t$, and $\sigma_t(C), t = 1,\ldots,k_s$.

The outputs $\sigma_t(C), t = 1,\ldots,k_s$ are approximations to the largest k_s singular values and $H_{k_s}^{(t)}, t = 1,\ldots,k$ are approximations to the corresponding left singular vectors of the matrix A. Thus, the SVD of A is approximated by

$$A \approx A_{k_s} := H_{k_s}\mathrm{Diag}(\sigma(C))(A^\top H_{k_s}\mathrm{Diag}(1/\sigma(C))^\top.$$

Drineas et al. [30] prove that with high probability, the following estimate holds for both $\xi = 2$ and $\xi = F$:

$$\|A - A_{k_s}\|_\xi^2 \le \min_{D:\mathrm{rank}(D)\le k_s} \|A - D\|_\xi^2 + poly(k_s, 1/c_s)\|A\|_F^2, \tag{2.6.1}$$

where $poly(k_s, 1/c_s)$ is a polynomial in k_s and $1/c_s$. Thus, A_{k_s} is a approximation to the best rank-k_s approximation to A. (For any matrix $M \in \mathbb{R}^{m\times n}$ with SVD $M = \sum_{i=1}^r \sigma_i u_i v_i^\top$,

where $\sigma_1 \geq \ldots \geq \sigma_r > 0, u_i \in \mathbb{R}^m, v_i \in \mathbb{R}^n$, the best rank-k approximation to M is given by $\bar{M} = \sum_{i=1}^{k} \sigma_i u_i v_i^\top$).

Note that in this algorithm, we compute an exact SVD of a smaller matrix $C^\top C \in \mathbb{R}^{c_s \times c_s}$. Thus, c_s determines the speed of this algorithm. If we choose a large c_s, we need more time to compute the SVD of $C^\top C$. However, the larger c_s is, the more likely are the $\sigma_t(C), t = 1, \ldots, k_s$ to be close to the largest k_s singular values of the matrix A since the second term in the right hand side of (2.6.1) is smaller. In our numerical experiments, we found that we could choose a relatively small c_s so that the computational time was reduced without significantly degrading the accuracy. In our tests, we obtained very good results by choosing $c_s = 2r_{\max} - 2$, where $r_{\max} = \lfloor (m+n-\sqrt{(m+n)^2 - 4p})/2 \rfloor$ is, for a given number of entries sampled, the largest rank of $m \times n$ matrices for which the matrix completion problem has a unique solution.

There are many ways to choose the probabilities p_i. In our numerical experiments in Section 2.7, we used the simplest one, i.e., we set all p_i equal to $1/n$. For other choices of p_i, see [30] and the references therein.

In our numerical experiments, we set k_s using the following procedure. In the k-th iteration, when computing the approximate SVD of $Y^k = X^k - \tau g^k$, we set k_s equal to the number of components in $\bar{s}_{k-1}$ that are no less than $\varepsilon_{k_s} \max\{\bar{s}_{k-1}\}$, where ε_{k_s} is a small positive number and $\max\{\bar{s}_{k-1}\}$ is the largest component in the vector $\bar{s}_{k-1}$ used to form $X^k = U^{k-1} \mathrm{Diag}(\bar{s}_{k-1}) V^{k-1^\top}$. Note that k_s is non-increasing in this procedure. However, if k_s is too small at some iteration, the non-expansive property (2.3.1) of the shrinkage operator S_ν may be violated since the approximate SVD is not a valid approximation when

k_s is too small. Thus, in algorithm FPCA, if (2.3.1) is violated 10 times, we increase k_s by 1. Our numerical experience indicates that this technique makes our algorithm very robust.

Our numerical results in Section 2.7 show that this approximate SVD based FPC algorithm: FPCA, is very fast, robust, and significantly outperforms other solvers (such as SDPT3) in recovering low-rank matrices. One reason for this is that in the approximate SVD algorithm, we compute a low-rank approximation to the original matrix. Hence, the iterative matrices produced by our algorithm are more likely to be of low-rank than an exact solution to the nuclear norm minimization problem (2.1.5), or equivalently, to the SDP (2.1.7), which is exactly what we want. Some convergence/recoverability properties of a variant of FPCA that uses a truncated SVD rather than a randomized SVD at each step, are discussed in [46].

2.7 Numerical results

In this section, we report on the application of our FPC, FPCA and Bregman iterative algorithms to a series of matrix completion problems of the form (2.1.4) to demonstrate the ability of these algorithms to efficiently recover low-rank matrices.

To illustrate the performance of our algorithmic approach combined with exact and approximate SVD algorithms, different stopping rules, and with or without debiasing, we tested the following solvers.

- FPC1. Exact SVD, no debiasing, stopping rule: (2.4.2).
- FPC2. Exact SVD, no debiasing, stopping rule: (2.4.1) and (2.4.2).

- FPC3. Exact SVD with debiasing, stopping rule: (2.4.2).
- FPCA. Approximate SVD, no debiasing, stopping rule: (2.4.2).
- Bregman. Bregman iterative method using FPC2 to solve the subproblems.

2.7.1 FPC and Bregman iterative algorithms for random matrices

In our first series of tests, we created random matrices $M \in \mathbb{R}^{m\times n}$ with rank r by the following procedure: we first generated random matrices $M_L \in \mathbb{R}^{m\times r}$ and $M_R \in \mathbb{R}^{n\times r}$ with i.i.d. Gaussian entries and then set $M = M_L M_R^\top$. We then sampled a subset Ω of p entries uniformly at random. For each problem with $m \times n$ matrix M, measurement number p and rank r, we solved 50 randomly created matrix completion problems. We use $SR = p/(mn)$, i.e., the number of measurements divided by the number of entries of the matrix, to denote the sampling ratio. We also list $FR = r(m+n-r)/p$, i.e. the dimension of the set of rank r matrices divided by the number of measurements, in the tables. Note that if $FR > 1$, then there is always an infinite number of matrices with rank r with the given entries, so we cannot hope to recover the matrix in this situation. We use $r_{\max}$ to denote the largest rank such that $FR \leq 1$, i.e., $r_{\max} = \lfloor (m+n-\sqrt{(m+n)^2-4p})/2 \rfloor$. We use NS to denote the number of matrices that are recovered successfully. We use AT to denote the average time (seconds) for the examples that are successfully solved.

We used the relative error

$$rel.err. := \frac{\|X_{opt} - M\|_F}{\|M\|_F}$$

to estimate the closeness of X_{opt} to M, where X_{opt} is the "optimal" solution to (2.1.5) produced by our algorithms. We declared M to be recovered if the relative error was less than 10^{-3}, which is the criterion used in [81] and [16]. We use RA, RU, RL to denote the average, largest and smallest relative error of the successfully recovered matrices, respectively.

We summarize the parameter settings used by the algorithms in Table 2.1. We use I_m to denote the maximum number of iterations allowed for solving each subproblem in FPC, i.e., if the stopping rules (2.4.2) (and (2.4.1)) are not satisfied after I_m iterations, we terminate the subproblem and decrease μ to start the next subproblem.

Table 2.1: Parameters in Algorithm FPC

FPC	$\bar{\mu} = 10^{-8}, \eta_\mu = 1/4, \mu_1 = \eta_\mu \|\mathcal{A}^* b\|_2, \tau = 1,$ $xtol = 10^{-10}, gtol = 10^{-4}, I_m = 500, X_0 = \mathbf{0}$
Approx SVD	$c_s = 2r_{\max} - 2, \varepsilon_{k_s} = 10^{-2}, p_i = 1/n, \forall i$

All numerical experiments were run in MATLAB 7.3.0 on a Dell Precision 670 workstation with an Intel Xeon(TM) 3.4GHZ CPU and 6GB of RAM.

The comparisons between FPC1, FPC2, FPC3 and SDPT3 for small matrix completion problems are presented in Table 2.2. From Table 2.2 we can see that FPC1 and FPC2 achieve almost the same recoverability and relative error, which means that as long as we set $xtol$ to be very small (like 10^{-10}), we only need to use (2.4.2) as the stopping rule for the inner iterations. That is, use of stopping rule (2.4.1) does not affect the performance of the algorithm. Of course FPC2 costs more time than FPC1 since more iterations are sometimes needed to satisfy the stopping rules in FPC2. While FPC3 can improve the

recoverability, it costs more time for performing debiasing. SDPT3 seems to obtain more accurate solutions than FPC1, FPC2 or FPC3.

Table 2.2: Comparisons of FPC1, FPC2, FPC3 and SDPT3 for randomly created small matrix completion problems (m=n=40, p=800, SR=0.5)

r	FR	Solver	NS	AT	RA	RU	RL
1	0.0988	FPC1	50	1.81	1.67e-9	1.22e-8	6.06e-10
		FPC2	50	3.61	1.32e-9	1.20e-8	2.55e-10
		FPC3	50	16.81	1.06e-9	2.22e-9	5.68e-10
		SDPT3	50	1.81	6.30e-10	3.46e-9	8.72e-11
2	0.1950	FPC1	42	3.05	1.01e-6	4.23e-5	8.36e-10
		FPC2	42	17.97	1.01e-6	4.23e-5	2.78e-10
		FPC3	49	16.86	1.26e-5	3.53e-4	7.62e-10
		SDPT3	44	1.90	1.50e-9	7.18e-9	1.82e-10
3	0.2888	FPC1	35	5.50	9.72e-9	2.85e-8	1.93e-9
		FPC2	35	20.33	2.17e-9	1.41e-8	3.88e-10
		FPC3	42	16.87	3.58e-5	7.40e-4	1.34e-9
		SDPT3	37	1.95	2.66e-9	1.58e-8	3.08e-10
4	0.3800	FPC1	22	9.08	7.91e-5	5.46e-4	3.57e-9
		FPC2	22	18.43	7.91e-5	5.46e-4	4.87e-10
		FPC3	29	16.95	3.83e-5	6.18e-4	2.57e-9
		SDPT3	29	2.09	1.18e-8	7.03e-8	7.97e-10
5	0.4688	FPC1	1	10.41	2.10e-8	2.10e-8	2.10e-8
		FPC2	1	17.88	2.70e-9	2.70e-9	2.70e-9
		FPC3	5	16.70	1.78e-4	6.73e-4	6.33e-9
		SDPT3	8	2.26	1.83e-7	8.12e-7	2.56e-9
6	0.5550	FPC1	0	—	—	—	—
		FPC2	0	—	—	—	—
		FPC3	0	—	—	—	—
		SDPT3	1	2.87	6.58e-7	6.58e-7	6.58e-7

To illustrate the performance of our Bregman iterative algorithm, we compare the results of using it versus using FPC2 in Table 2.3. From our numerical experience, for those problems for which the Bregman iterative algorithm greatly improves the recoverability, the Bregman iterative algorithm usually takes 2 to 3 iterations. Thus, in our numerical tests, we fixed the number of subproblems solved by our Bregman algorithm to 3. Since

our Bregman algorithm achieves as good a relative error as the FPC algorithm, we only report how many of the examples that are successfully recovered by FPC, are improved greatly by using our Bregman iterative algorithm. In Table 2.3, NIM is the number of examples that the Bregman iterative algorithm outperformed FPC2 greatly (the relative errors obtained from FPC2 were at least 10^4 times larger than those obtained by the Bregman algorithm). From Table 2.3 we can see that for more than half of the examples successfully recovered by FPC2, the Bregman iterative algorithm improved the relative errors greatly (from $[10^{-10}, 10^{-9}]$ to $[10^{-16}, 10^{-15}]$). Of course the run times for the Bregman iterative algorithm were about three times that for algorithm FPC2, since the former calls the latter three times to solve the subproblems.

Table 2.3: Numerical results for the Bregman iterative method for small matrix completion problems (m=n=40, p=800, SR=0.5)

Problem			FPC2		Bregman	
r	FR	NIM (NS)	RU	RL	RU	RL
1	0.0988	32 (50)	2.22e-9	2.55e-10	1.87e-15	3.35e-16
2	0.1950	29 (42)	5.01e-9	2.80e-10	2.96e-15	6.83e-16
3	0.2888	24 (35)	2.77e-9	3.88e-10	2.93e-15	1.00e-15
4	0.3800	10 (22)	5.51e-9	4.87e-10	3.11e-15	1.30e-15

In the following, we discuss the numerical results obtained by our approximate SVD based FPC algorithm (FPCA). We will see from these numerical results that FPCA achieves much better recoverability and is much faster than any of the solvers FPC1, FPC2, FPC3 or SDPT3.

We present the numerical results of FPCA for small (m=n=40) and medium (m=n=100) problems in Tables 2.4, and 2.5 respectively. Since we found that $xtol = 10^{-6}$ is small

enough to guarantee very good recoverability, we set $xtol = 10^{-6}$ in algorithm FPCA and used only (2.4.2) as stopping rule for the inner iterations. From these tables, we can see that our FPCA algorithm is much more powerful than SDPT3 for randomly created matrix completion problems. When $m = n = 40$ and $p = 800$, and the rank r was less than or equal to 8, FPCA recovered the matrices in all 50 examples. When rank $r = 9$, it failed on only one example. Even for rank $r = 10$, which is almost the largest rank that satisfies $FR \leq 1$, FPCA still recovered the solution in more than 60% of the examples. However, SDPT3 started to fail to recover the matrices when the rank $r = 2$. When $r = 6$, there was only one example out of 50 where the correct solution matrix was recovered. When $r \geq 7$, none of the 50 examples could be recovered. For the medium sized matrices ($m = n = 100$) we used $p = 2000$, which is only a 20% measurement rate, FPCA recovered the matrices in all 50 examples when $r \leq 6$. For $r = 7$, FPCA recovered the matrices in most of the examples (49 out of 50). When $r = 8$, more than 60% of the matrices were recovered successfully by FPCA. Even when $r = 9$, FPCA still recovered 1 matrix. However, SDPT3 could not recover all of the matrices even when the rank $r = 1$ and none of the matrices were recovered when $r \geq 4$. When we increased the number of measurements to 3000, FPCA recovered the matrices in all 50 examples up to rank $r = 12$. When $r = 13, 14$, FPCA still recovered most of them. However, SDPT3 started to fail for some matrices when $r = 3$. When $r \geq 8$, SDPT3 failed to recover any of the matrices. We can also see that for the medium sized problems, FPCA was much faster than SDPT3.

Table 2.4: Numerical results for FPCA and SDPT3 for randomly created small matrix completion problems (m=n=40, p=800, SR=0.5)

Problems		FPCA					SDPT3				
r	FR	NS	AT	RA	RU	RL	NS	AT	RA	RU	RL
1	0.0988	50	3.49	3.92e-7	1.43e-6	2.72e-7	50	1.84	6.30e-10	3.46e-9	8.70e-11
2	0.1950	50	3.60	1.44e-6	7.16e-6	4.41e-7	44	1.93	1.50e-9	7.18e-9	1.82e-10
3	0.2888	50	3.97	1.91e-6	4.07e-6	9.28e-7	37	1.99	2.66e-9	1.58e-8	3.10e-10
4	0.3800	50	4.03	2.64e-6	8.14e-6	1.54e-6	29	2.12	1.18e-8	7.03e-8	8.00e-10
5	0.4688	50	4.16	3.40e-6	7.62e-6	1.52e-6	8	2.30	1.83e-7	8.12e-7	2.60e-9
6	0.5550	50	4.45	4.08e-6	7.62e-6	2.26e-6	1	2.89	6.58e-7	6.58e-7	6.58e-7
7	0.6388	50	4.78	6.04e-6	1.57e-5	2.52e-6	0	—	—	—	—
8	0.7200	50	4.99	8.48e-6	5.72e-5	3.72e-6	0	—	—	—	—
9	0.7987	49	5.73	2.58e-5	5.94e-4	5.94e-6	0	—	—	—	—
10	0.8750	30	7.20	8.64e-5	6.04e-4	8.48e-6	0	—	—	—	—
11	0.9487	0	—	—	—	—	0	—	—	—	—

2.7.2 Comparison of FPCA and SVT

In this subsection we compare our FPCA algorithm against the SVT algorithm proposed in [14]. The SVT code was downloaded from *http://svt.caltech.edu*. We constructed two sets of test problems. One set contained "easy" problems. These problems are "easy" because the matrices are of very low-rank compared to the matrix size and the number of samples, and hence they are easy to recover. For all problems in this set, FR was less than 0.34. The other set contained "hard" problems, i.e., problems that are very challenging. These problems involved matrices that are not of very low rank and for which only a very limited number of entries were samples. For this set of problems, FR ranged from 0.40 to 0.87. The maximum iteration number in SVT was set to be 1000. All other parameters were set to their default values in SVT. The parameters of FPCA were set somewhat loosely for easy problems. Specifically, we set $\bar{\mu} = 10^{-4}, xtol = 10^{-4}, \tau = 2, I_m = 10$ and

Table 2.5: Numerical results for FPCA and SDPT3 for randomly created medium matrix completion problems (m=n=100)

Problems				FPCA					SDPT3				
p	r	SR	FR	NS	AT	RA	RU	RL	NS	AT	RA	RU	RL
2000	1	0.2	0.0995	50	4.93	5.80e-6	1.53e-5	2.86e-6	47	15.10	1.55e-9	1.83e-8	1.40e-10
2000	2	0.2	0.1980	50	5.26	6.10e-6	9.36e-6	4.06e-6	31	16.02	7.95e-9	8.69e-8	5.20e-10
2000	3	0.2	0.2955	50	5.80	7.48e-6	1.70e-5	4.75e-6	13	19.23	1.05e-4	9.70e-4	9.08e-10
2000	4	0.2	0.3920	50	9.33	1.09e-5	5.14e-5	6.79e-6	0	—	—	—	—
2000	5	0.2	0.4875	50	5.42	1.61e-5	8.95e-5	8.12e-6	0	—	—	—	—
2000	6	0.2	0.5820	50	7.02	2.62e-5	7.07e-5	8.72e-6	0	—	—	—	—
2000	7	0.2	0.6755	49	8.69	7.69e-5	5.53e-4	1.11e-5	0	—	—	—	—
2000	8	0.2	0.7680	32	10.94	1.97e-4	8.15e-4	2.29e-5	0	—	—	—	—
2000	9	0.2	0.8595	1	11.75	4.38e-4	4.38e-4	4.38e-4	0	—	—	—	—
2000	10	0.2	0.9500	0	—	—	—	—	0	—	—	—	—
3000	1	0.3	0.0663	50	7.73	1.97e-6	3.15e-6	1.22e-6	50	36.68	2.01e-10	9.64e-10	7.52e-11
3000	2	0.3	0.1320	50	7.85	2.68e-6	8.41e-6	1.44e-6	50	36.50	1.13e-9	2.97e-9	1.77e-10
3000	3	0.3	0.1970	50	8.10	2.82e-6	4.38e-6	1.83e-6	46	38.50	1.28e-5	5.89e-4	2.10e-10
3000	4	0.3	0.2613	50	8.94	3.57e-6	5.62e-6	2.64e-6	42	41.28	4.60e-6	1.21e-4	4.53e-10
3000	5	0.3	0.3250	50	9.12	4.06e-6	8.41e-6	2.78e-6	32	43.92	7.82e-8	1.50e-6	1.23e-9
3000	6	0.3	0.3880	50	9.24	4.84e-6	9.14e-6	3.71e-6	17	49.60	3.44e-7	4.29e-6	3.68e-9
3000	7	0.3	0.4503	50	9.41	5.72e-6	1.09e-5	3.96e-6	3	59.18	1.43e-4	4.28e-4	1.57e-7
3000	8	0.3	0.5120	50	9.62	6.37e-6	1.90e-5	4.43e-6	0	—	—	—	—
3000	9	0.3	0.5730	50	10.35	6.32e-6	1.60e-5	4.56e-6	0	—	—	—	—
3000	10	0.3	0.6333	50	10.93	8.45e-6	3.79e-5	5.59e-6	0	—	—	—	—
3000	11	0.3	0.6930	50	11.58	1.41e-5	6.84e-5	6.99e-6	0	—	—	—	—
3000	12	0.3	0.7520	50	12.17	1.84e-5	1.46e-4	8.84e-6	0	—	—	—	—
3000	13	0.3	0.8103	48	15.24	5.12e-5	6.91e-4	1.25e-5	0	—	—	—	—
3000	14	0.3	0.8680	39	18.85	2.35e-4	9.92e-4	2.05e-5	0	—	—	—	—
3000	15	0.3	0.9250	0	—	—	—	—	0	—	—	—	—
3000	16	0.3	0.9813	0	—	—	—	—	0	—	—	—	—

all other parameters were set to the values given in Table 2.1. Relative errors and times were averaged over 5 runs. In this subsection, all test matrices were square, i.e., $m = n$.

From Table 2.6, we can see that for the easy problems except for one problem which is exceptionally sparse as well as having low rank, FPCA was much faster and usually provided more accurate solutions than SVT.

For hard problems, all parameters of FPCA were set to the values given in Table 2.1, except that we set $xtol = 10^{-6}$ since this value is small enough to guarantee very good recoverability. Also, for small problems (i.e., $\max\{m,n\} < 1000$), we set $I_m = 500$; and for large problems (i.e., $\max\{m,n\} \geq 1000$), we set $I_m = 20$. We use "—" to indicate that

Table 2.6: Comparison of FPCA and SVT on easy problems

Problems					FPCA		SVT	
n	r	p	SR	FR	rel.err.	time	rel.err.	time
100	10	5666	0.57	0.34	4.27e-5	0.39	1.64e-3	30.40
200	10	15665	0.39	0.25	6.40e-5	1.38	1.90e-4	9.33
500	10	49471	0.20	0.20	2.48e-4	8.01	1.88e-4	23.77
1000	10	119406	0.12	0.17	5.04e-4	18.49	1.68e-4	41.81
1000	50	389852	0.39	0.25	3.13e-5	120.64	1.63e-4	228.79
1000	100	569900	0.57	0.33	2.26e-5	177.17	1.71e-4	635.15
5000	10	597973	0.02	0.17	1.58e-3	1037.12	1.73e-4	121.39
5000	50	2486747	0.10	0.20	5.39e-4	1252.70	1.59e-4	1375.33
5000	100	3957533	0.16	0.25	2.90e-4	2347.41	1.74e-4	5369.76

the algorithm either diverges or does not terminate in one hour. Relative errors and times were averaged over 5 runs.

Table 2.7: Comparison of FPCA and SVT on hard problems

Problems				FPCA		SVT	
n	r	SR	FR	rel.err.	time	rel.err.	time
40	9	0.5	0.80	1.21e-5	5.72	5.01e-1	3.05
100	14	0.3	0.87	1.32e-4	19.00	8.31e-1	316.90
1000	20	0.1	0.40	2.46e-5	116.15	—	—
1000	30	0.1	0.59	2.00e-3	128.30	—	—
1000	50	0.2	0.49	1.04e-5	183.67	—	—

From Table 2.7, we can see that for the hard problems, SVT either diverged or did not solve the problems in less than one hour, or it yielded a very inaccurate solution. In contrast, FPCA always provided a very good solution efficiently.

We can also see that FPCA was able to efficiently solve large problems ($m = n = 1000$) that could not be solved by SDPT3 due to the large size of the matrices and the large number of constraints.

2.7.3 Results for real data matrices

In this section, we consider matrix completion problems based on two real data sets: the Jester joke data set [44] and the DNA data set [85]. The Jester joke data set contains 4.1 million ratings for 100 jokes from 73,421 users and is available on the website http://www.ieor.berkeley.edu/~Egoldberg/jester-data/. Since the number of jokes is only 100, but the number of users is quite large, we randomly selected n_u users to get a modestly sized matrix for testing purpose. As in [88], we randomly held out two ratings for each user. Since some entries in the matrix are missing, we cannot compute the relative error as we did for the randomly created matrices. Instead, we computed the Normalized Mean Absolute Error (NMAE) as in [44] and [88]. The Mean Absolute Error (MAE) is defined as

$$MAE = \frac{1}{2N}\sum_{i=1}^{N} |\hat{r}^i_{i_1} - r^i_{i_1}| + |\hat{r}^i_{i_2} - r^i_{i_2}|, \tag{2.7.1}$$

where r^i_j and $\hat{r}^i_j$ are the withheld and predicted ratings of movie j by user i, respectively, for $j = i_1, i_2$. NMAE is defined as

$$NMAE = \frac{MAE}{r_{\max} - r_{\min}}, \tag{2.7.2}$$

where $r_{\min}$ and $r_{\max}$ are lower and upper bounds for the ratings. Since all ratings are scaled to the range $[-10, +10]$, we have $r_{\min} = -10$ and $r_{\max} = 10$.

The numerical results for the Jester data set using FPC1 and FPCA are presented in

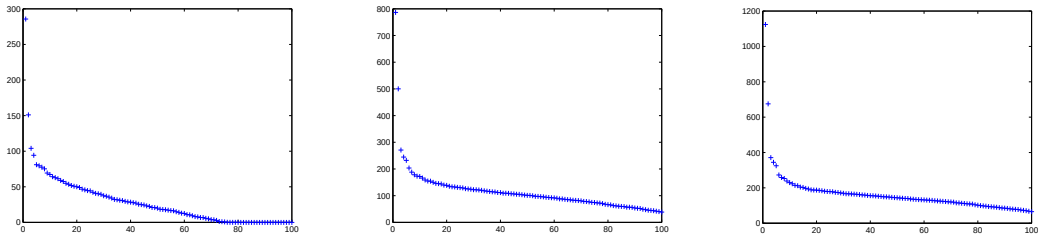

Figure 2.1: Distribution of the singular values of the recovered matrices for the Jester data set using FPC1. Left:100 users, Middle: 1000 users, Right: 2000 users

Tables 2.8 and 2.9, respectively. In these two tables, $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest positive singular values of the recovered matrices, and *rank* is the rank of the recovered matrices. The distributions of the singular values of the recovered matrices are shown in Figures 2.1 and 2.2. From Tables 2.8 and 2.9 we can see that by using FPC1 and FPCA to recover these matrices, we can get relatively low NMAEs, which are comparable to the results shown in [88] and [44].

Table 2.8: Numerical results for FPC1 for the Jester joke data set

num.user	num.samp	samp.ratio	rank	$\sigma_{\max}$	$\sigma_{\min}$	NMAE	Time
100	7172	0.7172	79	285.65	3.49e-4	0.1727	34.30
1000	71152	0.7115	100	786.37	38.43	0.1667	304.81
2000	140691	0.7035	100	1.1242e+3	65.06	0.1582	661.66

Table 2.9: Numerical results for FPCA for the Jester joke data set (c_s is the number of rows we picked for the approximate SVD)

num.user	num.samp	samp.ratio	ε_{k_s}	c_s	rank	$\sigma_{\max}$	$\sigma_{\min}$	NMAE	Time
100	7172	0.7172	10^{-2}	25	20	295.14	32.68	0.1627	26.73
1000	71152	0.7115	10^{-2}	100	85	859.27	48.04	0.2008	808.52
1000	71152	0.7115	10^{-4}	100	90	859.46	44.62	0.2101	778.56
2000	140691	0.7035	10^{-4}	200	100	1.1518e+3	63.52	0.1564	1.1345e+3

We also used two data sets of DNA microarrays from [85]. These data sets are available on the website http://cellcycle-www.stanford.edu/. The first microarray data set is a matrix

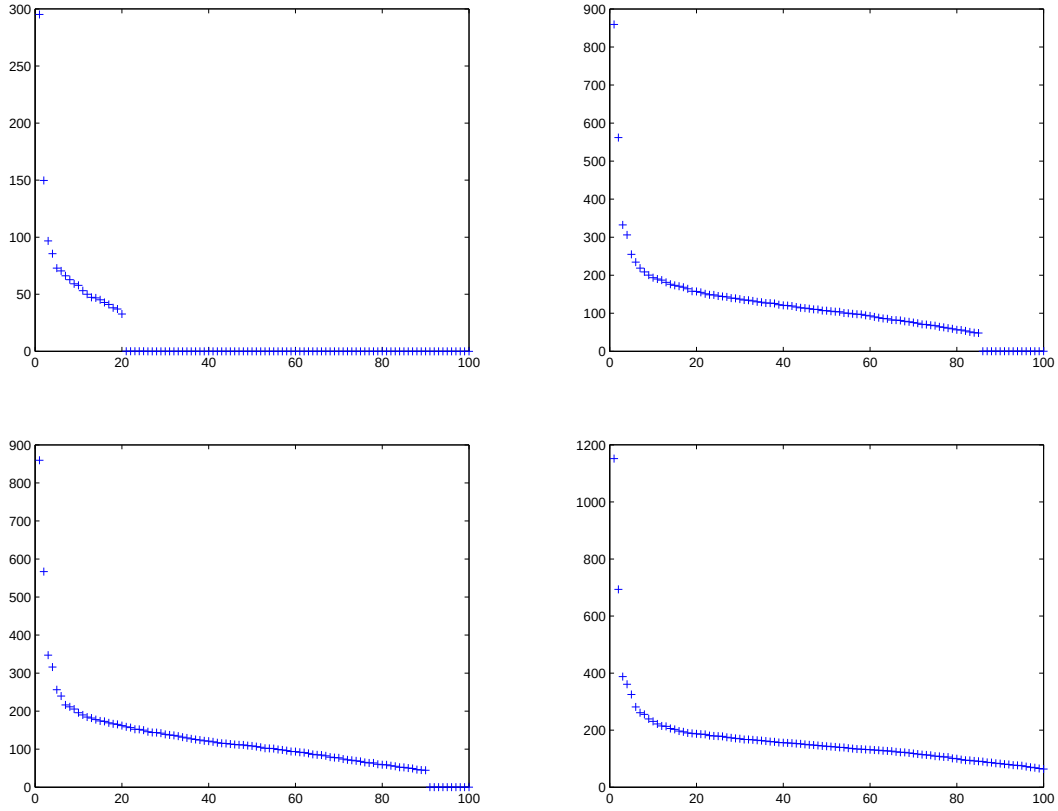

Figure 2.2: Distribution of the singular values of the recovered matrices for the Jester data set using FPCA. Upper Left: 100 users, $\varepsilon_{k_s} = 10^{-2}, c_s = 25$; Upper Right: 1000 users, $\varepsilon_{k_s} = 10^{-2}, c_s = 100$; Bottom Left: 1000 users, $\varepsilon_{k_s} = 10^{-4}, c_s = 100$; Bottom Right: 2000 users, $\varepsilon_{k_s} = 10^{-4}, c_s = 200$

that represents the expression of 6178 genes in 14 experiments based on the Elutriation data set in [85]. The second microarray data set is based on the Cdc15 data set in [85], and represents the expression of 6178 genes in 24 experiments. However, some entries in these two matrices are missing. For evaluating our algorithms, we created complete matrices by deleted all rows containing missing values. This is similar to how the DNA microarray data set was preprocessed in [95]. The resulting complete matrix for the Elutriation data set was 5766×14. The complete matrix for the Cdc15 data set was 4381×24. We must point out that these DNA microarray matrices are neither low-rank nor even approximately low-rank although such claims have been made in some papers. The distributions of the singular values of these two matrices are shown in Figure 2.3. From this figure we can see that in each microarray matrix, only one singular value is close to zero, while the others are far away from zero. Thus there is no way to claim that the rank of the Elutriation matrix is less than 13, or the rank of the Cdc15 matrix is less than 23. Since these matrices are not low-rank, we cannot expect our algorithms to recover these matrices by sampling only a small portion of their entries. Thus we needed to further modify the data sets to yield low-rank matrices. Specifically, we used the best rank-2 approximation to the Elutriation matrix as the new complete Elutriation matrix and the best rank-5 approximation to the Cdc15 matrix as the new complete Cdc15 matrix. The numerical results for FPCA for recovering these two matrices are presented in Table 2.10. In the FPCA algorithm, we set $\varepsilon_{k_s} = 10^{-2}$ and $xtol = 10^{-6}$. For the Elutriation matrix, we set $c_s = 115$ and for the Cdc15 matrix, we set $c_s = 88$. The observed entries were randomly sampled. From Table 2.10 we can see that by taking 60% of the entries of the matrices, our FPCA algorithm can recover these matrices

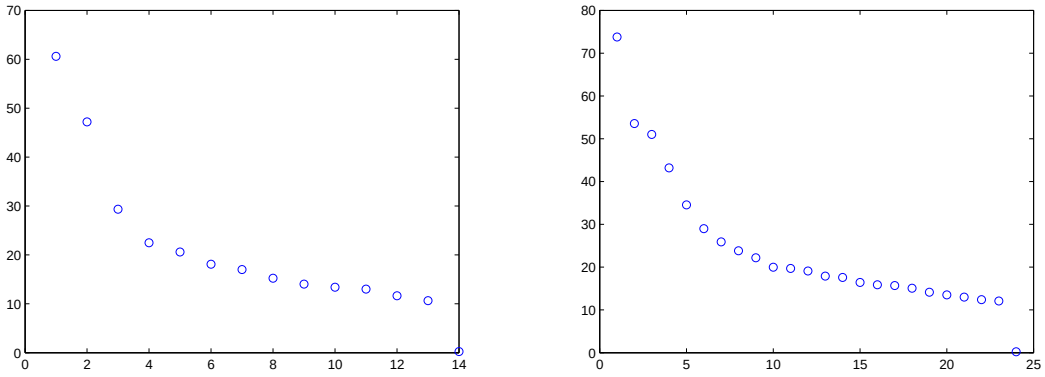

Figure 2.3: Distribution of the singular values of the matrices in the original DNA microarray data sets. Left: Elutriation matrix; Right: Cdc15 matrix.

very well, yielding relative errors as low as 10^{-5} and 10^{-6}, which is promising for practical use.

Table 2.10: Numerical results of FPCA for DNA microarray data sets

Matrix	m	n	p	rank	SR	FR	rel.err	Time
Elutriation	5766	14	48434	2	0.6	0.2386	1.83e-5	218.01
Cdc15	4381	24	63086	5	0.6	0.3487	7.95e-6	189.32

To graphically illustrate the effectiveness of FPCA, we applied it to image inpainting [5]. Grayscale images and color images can be expressed as matrices and tensors, respectively. In grayscale image inpainting, the grayscale value of some of the pixels of the image are missing, and we want to fill in these missing values. If the image is of low-rank, or of numerical low-rank, we can solve the image inpainting problem as a matrix completion problem (2.1.4). In our test we applied SVD to the 512×512 image in Figure 2.4(a), and truncated this decomposition to get the rank-40 image which is shown in Figure 2.4(b). Figure 2.4(c) is a masked version of the image in Figure 2.4(a), where one half of the pixels in Figure 2.4(a) were masked uniformly at random. Figure 2.4(d) is the image obtained

from Figure 2.4(c) by applying FPCA. Figure 2.4(d) is a low-rank approximation to Figure 2.4(a) with a relative error of $8.41e-2$. Figure 2.4(e) is a masked version of the image in Figure 2.4(b), where one half of the pixels in Figure 2.4(b) were masked uniformly at random. Figure 2.4(f) is the image obtained from Figure 2.4(e) by applying FPCA. Figure 2.4(f) is an approximation to Figure 2.4(b) with a relative error of $3.61e-2$. Figure 2.4(g) is another masked image obtained from Figure 2.4(b), where 4 percent of the pixels were masked in a non-random fashion. Figure 2.4(h) is the image obtained from Figure 2.4(g) by applying FPCA. Figure 2.4(g) is an approximation to Figure 2.4(b) with a relative error of $1.70e-2$.

2.8 Conclusion

In this chapter, we derived a fixed-point continuation algorithm and a Bregman iterative algorithm for solving the linearly constrained nuclear norm minimization problem, which is a convex relaxation of the NP-hard linearly constrained matrix rank minimization problem. The convergence of the fixed-point iterative scheme was established. By adopting an approximate SVD technique, we obtained a very powerful algorithm (FPCA) for the matrix rank minimization problem. On matrix completion problems, FPCA greatly outperforms SDP solvers such as SDPT3 in both speed and recoverability of low-rank matrices.

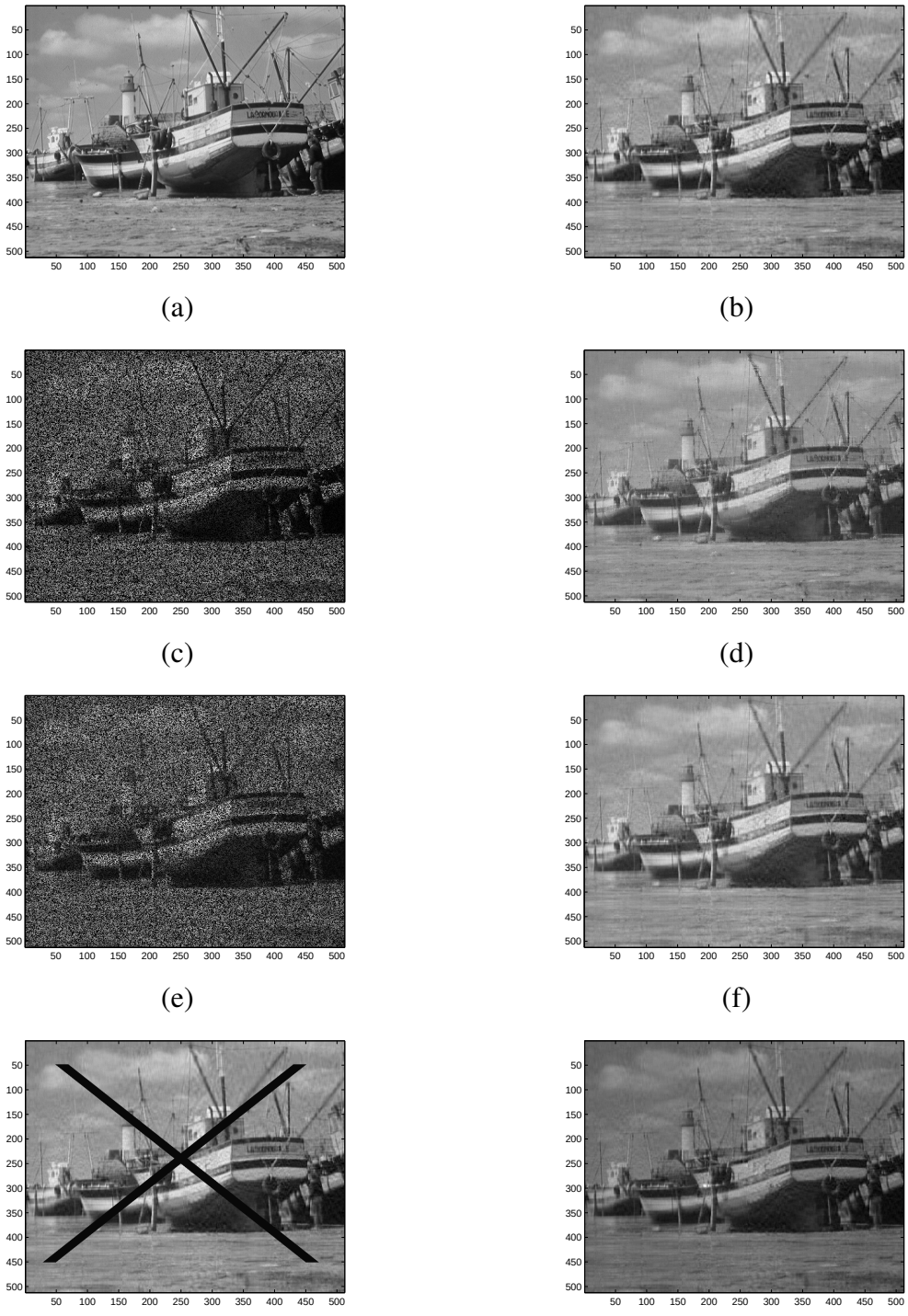

Figure 2.4: (a): Original 512×512 image with full rank; (b): Original image truncated to be of rank 40; (c): 50% randomly masked original image; (d): Recovered image from 50% randomly masked original image ($rel.err = 8.41e-2$); (e): 50% randomly masked rank 40 image; (f): Recovered image from 50% randomly masked rank 40 image ($rel.err = 3.61e-2$); (g): Deterministically masked rank 40 image (SR = 0.96); (h): Recovered image from deterministically masked rank 40 image ($rel.err = 1.70e-2$).

Chapter 3

Convergence and Recoverability of FPCA and Its Variants

3.1 Introduction

In this chapter, we study the convergence/recoverability properties and numerical performance of FPCA and some of its variants for solving the affinely constrained matrix rank minimization problem. For convenience, we restate the matrix rank minimization and its convex relaxations as follows.

The matrix rank minimization problem can be cast as:

$$\min_{X\in\mathbb{R}^{m\times n}} \quad \text{rank}(X) \quad \text{s.t.} \quad \mathcal{A}(X) = b, \tag{3.1.1}$$

where $b \in \mathbb{R}^p$ and $\mathcal{A} : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^p$ is a linear map. Without loss of generality, we assume

that $m \leq n$ throughout this chapter. A convex relaxation of (3.1.1) is the following nuclear norm minimization problem:

$$\min \quad \|X\|_* \quad \text{s.t.} \quad \mathcal{A}(X) = b. \tag{3.1.2}$$

A penalty variant of (3.1.2) is

$$\min \quad \mu\|X\|_* + \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2. \tag{3.1.3}$$

The FPCA algorithm (Algorithm 2.3) can be restated as follows:

Algorithm 3.1 Fixed-Point Continuation with Approximate SVD for MRM (FPCA)

Initialization: Set $X := X^0$.
for $\mu = \mu_1, \mu_2, \ldots, \mu_L = \bar{\mu}$ **do**
  while not converged **do**
    $Y := X - \tau\mathcal{A}^*(\mathcal{A}X - b)$.
    choose r.
    $X := S_{\tau\mu}(R_r(Y))$.
  end while
end for

We can see that FPCA makes use of three techniques, hard thresholding, soft shrinkage and continuation. These three techniques have different properties which, when combined, produce a very robust and efficient algorithm with great recoverability properties. By using only one or two of these three techniques, we get different variants of FPCA. We will study two of these variants, Iterative Hard Thresholding (IHT) and Iterative Hard Thresholding with soft Matrix Shrinkage (IHTMS) in Sections 3.3 and 3.4, respectively, and FPCA with given rank r (FPCAr) in Section 3.5.

Before we discuss these algorithms, we need to define restricted isometry property and present some consequences of it that our analysis relies on.

3.2 Restricted Isometry Property

In compressed sensing and matrix rank minimization, the restricted isometry property (RIP) of the matrix A or linear operator $\mathcal{A}$ plays a key role in the relationship between the original combinatorial problem and its convex relaxation and their optimal solutions.

The definition of the RIP for matrix rank minimization is:

Definition 3.2.1. *For every integer r with* $1 \leq r \leq m$*, the linear operator* $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ *is said to satisfy the Restricted Isometry Property with the restricted isometry constant* $\delta_r(\mathcal{A})$ *if* $\delta_r(\mathcal{A})$ *is the minimum constant that satisfies*

$$(1-\delta_r(\mathcal{A}))\|X\|_F^2 \leq \|\mathcal{A}X\|_2^2 \leq (1+\delta_r(\mathcal{A}))\|X\|_F^2, \tag{3.2.1}$$

for all $X \in \mathbb{R}^{m\times n}$ *with* $rank(X) \leq r$*.* $\delta_r(\mathcal{A})$ *is called the RIP constant. Note that* $\delta_s \leq \delta_t$*, if* $s \leq t$*.*

The RIP concept and the RIP constant $\delta_r(\mathcal{A})$ play a central role in the theoretical developments of this chapter. We first note that if the operator $\mathcal{A}$ has a nontrivial kernel, i.e., there exists $X \in \mathbb{R}^{m\times n}$ such that $\mathcal{A}X = 0$ and $X \neq 0$, then $\delta_n(\mathcal{A}) \geq 1$. Second, if we represent $\mathcal{A}$ in the coordinate form $(\mathcal{A}X)_i = \mathrm{Tr}(A_iX), i = 1,\ldots,p$, then $\delta_r(\mathcal{A})$ is related to the joint kernel of the matrices A_i. For example, if there exists a matrix $X \in \mathbb{R}^{m\times n}$ with rank r

such that $A_iX = 0, i = 1, \ldots, p$, then $\delta_r(\mathcal{A}) \geq 1$. Our results in this chapter do not apply to such a pathological case.

For matrix rank minimization (3.1.1), Recht et al. [81] proved the following results.

Theorem 3.2.2 (Theorem 3.3 in [81]). *Suppose that* $rank(X) \leq r$, $r \geq 1$ *and* $\delta_{5r}(\mathcal{A}) < 0.1$. *Then* (3.1.1) *and the corresponding nuclear norm minimization* (3.1.2) *have the same optimal solution.*

Theorem 3.2.3 (Theorem 4.2 in [81]). *Fix* $\delta \in (0,1)$. *If* $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ *is a nearly isometric random map (see Definition 4.1 in [81]), then for every* $1 \leq r \leq m$, *there exist constants* $c_0, c_1 > 0$ *depending only on* δ *such that, with probability at least* $1 - \exp(-c_1 p)$, $\delta_r(\mathcal{A}) \leq \delta$ *whenever* $p \geq c_0 r(m+n)\log(mn)$.

Theorems 3.2.2 and 3.2.3 indicate that if $\mathcal{A}$ is a nearly isometric random map, then with very high probability, $\mathcal{A}$ will satisfy the RIP with a small RIP constant and thus we can solve (3.1.1) by solving its convex relaxation (3.1.2). For example, if A is the matrix version of the operator $\mathcal{A}$, and its entries A_{ij} are independent, identically distributed (i.i.d.) Gaussian, i.e., $A_{ij} \sim \mathcal{N}(0, 1/p)$, then $\mathcal{A}$ is a nearly isometric random map. For other nearly isometric random maps, see [81].

In Section 3.7, we will show empirically that when the entries of A are i.i.d. Gaussian, the algorithms proposed in this chapter can solve the matrix rank minimization problem (3.1.1) very well.

In our proofs of the convergence of FPCA variants, we need $\mathcal{A}$ to satisfy the RIP. Before

we describe some properties of the RIP that we will use in our proofs, we need the following definitions.

Definition 3.2.4 (**Orthonormal basis of a subspace).** *Given a set of rank-one matrices* $\Psi = \{\psi_1, \ldots, \psi_r\}$*, there exists a set of orthonormal matrices* $\Gamma = \{\gamma_1, \ldots, \gamma_s\}$*, i.e.,* $\langle \gamma_i, \gamma_j \rangle = 0$*, for* $i \neq j$ *and* $\|\gamma_i\|_F = 1$ *for all i, such that span*$(\Gamma) =$ *span*(Ψ)*. We call* Γ *an* **orthonormal basis** *for the subspace span*(Ψ)*. We use* $P_\Gamma X$ *to denote the projection of X onto the subspace span*(Γ)*. Note that* $P_\Gamma X = P_\Psi X$ *and rank*$(P_\Gamma X) \leq r, \forall X \in \mathbb{R}^{m\times n}$*.*

Definition 3.2.5 (**SVD basis of a matrix).** *Assume that the rank-r matrix* X_r *has the singular value decomposition* $X_r = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$*.* $\Gamma := \{u_1 v_1^\top, u_2 v_2^\top, \ldots, u_r v_r^\top\}$ *is called an* **SVD basis** *for the matrix* X_r*. Note that elements in* Γ *are orthonormal rank-one matrices.*

We now list some important properties of linear operators that satisfy RIP. [1]

Proposition 3.2.6. *Suppose that the linear operator* $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ *satisfies the RIP with constant* $\delta_r(\mathcal{A})$*. Let* Ψ *be an arbitrary orthonormal subset of* $\mathbb{R}^{m\times n}$ *such that rank*$(P_\Psi X) \leq r, \forall X \in \mathbb{R}^{m\times n}$*. Then, for all* $b \in \mathbb{R}^p$ *and* $X \in \mathbb{R}^{m\times n}$*, the following properties hold:*

$$\|P_\Psi \mathcal{A}^* b\|_F \leq \sqrt{1+\delta_r(\mathcal{A})}\|b\|_2 \tag{3.2.2}$$

$$(1-\delta_r(\mathcal{A}))\|P_\Psi X\|_F \leq \|P_\Psi \mathcal{A}^* \mathcal{A} P_\Psi X\|_F \leq (1+\delta_r(\mathcal{A}))\|P_\Psi X\|_F. \tag{3.2.3}$$

[1] Propositions 3.2.6 and 3.2.8 were first proposed by Lee and Bresler without proof in [61]. Proofs of Propositions 3.2.6 and 3.2.8 were provided later in [60].

Proof. We prove (3.2.2) first. Since for any $X \in \mathbb{R}^{m\times n}$, $\text{rank}(P_\Psi X) \leq r$, we have

$$\begin{aligned}
|\langle X, P_\Psi \mathcal{A}^* b\rangle| &= |\langle \mathcal{A} P_\Psi X, b\rangle| \\
&\leq \|\mathcal{A} P_\Psi X\|_2 \|b\|_2 \\
&\leq \sqrt{1+\delta_r(\mathcal{A})}\|P_\Psi X\|_F \|b\|_2 \\
&\leq \sqrt{1+\delta_r(\mathcal{A})}\|X\|_F \|b\|_2.
\end{aligned}$$

Thus

$$\|P_\Psi \mathcal{A}^* b\|_F = \max_{\|X\|_F=1} |\langle X, P_\Psi \mathcal{A}^* b\rangle| \leq \sqrt{1+\delta_r(\mathcal{A})}\|b\|_2.$$

To prove (3.2.3), note that by the RIP,

$$(1-\delta_r(\mathcal{A}))\|P_\Psi X\|_F^2 \leq \|\mathcal{A} P_\Psi X\|_F^2 \leq (1+\delta_r(\mathcal{A}))\|P_\Psi X\|_F^2,$$

which means the eigenvalues of $P_\Psi \mathcal{A}^* \mathcal{A} P_\Psi$ restricted to $\text{span}(\Psi)$ are in the interval $[1-\delta_r(\mathcal{A}), 1+\delta_r(\mathcal{A})]$. Thus (3.2.3) holds. □

Proposition 3.2.7. *Suppose that the linear operator $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ satisfies the RIP with constant $\delta_r(\mathcal{A})$. Let Ψ, Ψ' be arbitrary orthonormal subsets of $\mathbb{R}^{m\times n}$ such that rank$(P_{\Psi\cup\Psi'} X) \leq r$, for any $X \in \mathbb{R}^{m\times n}$. Then the following inequality holds*

$$\|P_\Psi \mathcal{A}^* \mathcal{A}(I-P_\Psi)X\|_F \leq \delta_r(\mathcal{A})\|(I-P_\Psi)X\|_F, \forall X \in \text{span}(\Psi'). \tag{3.2.4}$$

Proof. First, we prove

$$|\langle \mathcal{A}(I-P_\Psi)X, \mathcal{A}P_\Psi Y\rangle| \le \delta_r(\mathcal{A})\|(I-P_\Psi)X\|_F\|P_\Psi Y\|_F, \forall Y \in \mathbb{R}^{m\times n}, X \in \text{span}(\Psi'). \tag{3.2.5}$$

(3.2.5) holds obviously if $(I-P_\Psi)X = 0$ or $P_\Psi Y = 0$. Thus we can assume $(I-P_\Psi)X \neq 0$ and $P_\Psi Y \neq 0$. Define $\hat{X} = \frac{(I-P_\Psi)X}{\|(I-P_\Psi)X\|_F}$ and $\hat{Y} = \frac{P_\Psi Y}{\|P_\Psi Y\|_F}$; then we have $\left\|\hat{X}\right\|_F = 1$, $\left\|\hat{Y}\right\|_F = 1$ and $\langle \hat{X}, \hat{Y}\rangle = 0$. Since $\hat{X} \in \text{span}(\Psi \cup \Psi')$ and $\hat{Y} \in \text{span}(\Psi)$, we have rank $\left(\hat{X}+\hat{Y}\right) \le r$ and rank $\left(\hat{X}-\hat{Y}\right) \le r$. Hence by RIP,

$$\begin{aligned} 2(1-\delta_r(\mathcal{A})) = (1-\delta_r(\mathcal{A}))\left\|\hat{X}+\hat{Y}\right\|_F^2 &\le \left\|\mathcal{A}\hat{X}+\mathcal{A}\hat{Y}\right\|_2^2 \\ &\le (1+\delta_r(\mathcal{A}))\left\|\hat{X}+\hat{Y}\right\|_F^2 = 2(1+\delta_r(\mathcal{A})). \end{aligned}$$

and

$$\begin{aligned} 2(1-\delta_r(\mathcal{A})) = (1-\delta_r(\mathcal{A}))\left\|\hat{X}-\hat{Y}\right\|_F^2 &\le \left\|\mathcal{A}\hat{X}-\mathcal{A}\hat{Y}\right\|_2^2 \\ &\le (1+\delta_r(\mathcal{A}))\left\|\hat{X}-\hat{Y}\right\|_F^2 = 2(1+\delta_r(\mathcal{A})). \end{aligned}$$

Therefore we have

$$\langle \mathcal{A}\hat{X}, \mathcal{A}\hat{Y}\rangle = \frac{\left\|\mathcal{A}\hat{X}+\mathcal{A}\hat{Y}\right\|_2^2 - \left\|\mathcal{A}\hat{X}-\mathcal{A}\hat{Y}\right\|_2^2}{4} \le \delta_r(\mathcal{A})$$

and

$$-\langle \mathcal{A}\hat{X}, \mathcal{A}\hat{Y}\rangle = \frac{\left\|\mathcal{A}\hat{X} - \mathcal{A}\hat{Y}\right\|_2^2 - \left\|\mathcal{A}\hat{X} + \mathcal{A}\hat{Y}\right\|_2^2}{4} \leq \delta_r(\mathcal{A}).$$

Thus, $|\langle \mathcal{A}\hat{X}, \mathcal{A}\hat{Y}\rangle| \leq \delta_r(\mathcal{A})$ and (3.2.5) holds.

Finally we have, for any $X \in \operatorname{span}(\Psi')$,

$$\begin{aligned}
\|P_\Psi \mathcal{A}^* \mathcal{A}(I - P_\Psi)X\|_F &= \max_{\|Y\|_F=1} |\langle P_\Psi \mathcal{A}^* \mathcal{A}(I - P_\Psi)X, Y\rangle| \\
&= \max_{\|Y\|_F=1} |\langle \mathcal{A}(I - P_\Psi)X, \mathcal{A}P_\Psi Y\rangle| \\
&\leq \delta_r(\mathcal{A}) \|(I - P_\Psi)X\|_F ,
\end{aligned}$$

i.e., (3.2.4) holds, which completes the proof. □

Proposition 3.2.8. *If a linear map $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ satisfies*

$$\|\mathcal{A}X\|_2^2 \leq (1 + \delta_r(\mathcal{A}))\|X\|_F^2, \quad \forall X \in \mathbb{R}^{m\times n}, rank(X) \leq r, \tag{3.2.6}$$

then

$$\|\mathcal{A}X\|_2 \leq \sqrt{1 + \delta_r(\mathcal{A})}\left(\|X\|_F + \frac{1}{\sqrt{r}}\|X\|_*\right), \quad \forall X \in \mathbb{R}^{m\times n}. \tag{3.2.7}$$

Proof. This proof essentially follows that given by Needell and Tropp in [74]. Let $B^s := \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = s, \|X\|_F \leq 1\}$ be the unit ball of rank-s matrices in $\mathbb{R}^{m\times n}$. Define

the convex hull of the unit norm matrices with rank at most r as:

$$S := \operatorname{conv}\left\{ \bigcup_{s \le r} B^s \right\} \subset \mathbb{R}^{m \times n}.$$

By (3.2.6), we know that the operator norm

$$\|\mathcal{A}\|_{S \to 2} = \max_{X \in S} \|\mathcal{A}X\|_2 \le \sqrt{1 + \delta_r(\mathcal{A})}.$$

Define another convex set

$$K := \{X \in \mathbb{R}^{m \times n} : \|X\|_F + \frac{1}{\sqrt{r}} \|X\|_* \le 1\} \subset \mathbb{R}^{m \times n},$$

and consider the operator norm

$$\|\mathcal{A}\|_{K \to 2} = \max_{X \in K} \|\mathcal{A}X\|_2.$$

The content of the proposition is the claim that $K \subset S$.

Choose a matrix $X \in K$ with SVD $X = U \operatorname{Diag}(\sigma) V^\top$. Let I_0 index the r largest components of σ, breaking ties lexicographically. Let I_1 index the next largest r components, and so forth. Note that the final block I_J may have fewer than r components. We may assume that $\sigma|_{I_j}$ is nonzero for each j. This partition induces a decomposition

$$X = U[\operatorname{Diag}(\sigma|_{I_0}) + \sum_{j=1}^{J} \operatorname{Diag}(\sigma|_{I_j})]V^\top = \lambda_0 Y_0 + \sum_{j=1}^{J} \lambda_j Y_j,$$

where $\lambda_j = \|U\mathrm{Diag}(\sigma|_{I_j})V^\top\|_F$ and $Y_j = \lambda_j^{-1}U\mathrm{Diag}(\sigma|_{I_j})V^\top$. By construction, each matrix Y_j belongs to S because it's rank is at most r and it has unit Frobenius norm. We will prove that $\sum_j \lambda_j \leq 1$, which implies that X can be expressed as a convex combination of matrices from the set S. So $X \in S$ and $K \subset S$.

Fix j in the range $\{1,2,\ldots,J\}$. It follows that $\sigma|_{I_j}$ contains at most r elements and $\sigma|_{I_{j-1}}$ contains exactly r elements. Therefore,

$$\lambda_j = \|\sigma|_{I_j}\|_2 \leq \sqrt{r}\|\sigma|_{I_j}\|_\infty \leq \sqrt{r}\cdot\frac{1}{r}\|\sigma|_{I_{j-1}}\|_1.$$

Summing these relations, we obtain,

$$\sum_{j=1}^{J}\lambda_j \leq \frac{1}{\sqrt{r}}\|\sigma|_{I_{j-1}}\|_1 \leq \frac{1}{\sqrt{r}}\|X\|_*.$$

It is obvious that $\lambda_0 = \|\sigma|_{I_0}\|_2 \leq \|X\|_F$. We now conclude that

$$\sum_{j=0}^{J}\lambda_j \leq \|X\|_F + \frac{1}{\sqrt{r}}\|X\|_* \leq 1$$

because $X \in K$. This implies that $X \in S$ and $K \subset S$, and thus completes the proof. □

3.3 Iterative Hard Thresholding

In the following three sections, we assume that the rank r of the optimal solution is given and we compute the best rank-r approximation to Y in each iteration. In Section 3.6, we

give a heuristic for choosing r in each iteration if r is unknown and use the fast Monte Carlo algorithm proposed in [30] and described in Section 2.6 to compute a rank-r approximation to Y.

In this section, we study a variant of FPCA that we call Iterative Hard Thresholding (IHT) because of its similarity to the algorithm in [8] for compressed sensing.

If in FPCA, we assume that the rank r is given, we do not do any continuation or soft shrinkage, and always choose the stepsize τ equal to one, then FPCA becomes Algorithm 3.2 (IHT). At each iteration of IHT, we first perform a gradient step $Y^{k+1} := X^k - \mathcal{A}^*(\mathcal{A}X^k - b)$, and then apply hard thresholding to the singular values of Y^{k+1}, i.e., we only keep the largest r singular values of Y^{k+1}, to get X^{k+1}.

Algorithm 3.2 Iterative Hard Thresholding (IHT)

Initialization: Given X^0, r.
for k = 0,1,... **do**
  $Y^{k+1} := X^k - \mathcal{A}^*(\mathcal{A}X^k - b)$.
  $X^{k+1} := R_r(Y^{k+1})$
end for

As previously mentioned, IHT is closely related to an algorithm proposed by Blumensath and Davies [8] for compressed sensing. Their algorithm for solving (1.2.1) performs the following iterative scheme:

$$\begin{cases} y^{k+1} = x^k - \tau A^\top (Ax^k - b) \\ x^{k+1} = H_r(y^{k+1}), \end{cases} \tag{3.3.1}$$

where $H_r(y)$ is the hard thresholding operator that sets all but the largest (in magnitude) r

elements of y to zero. Clearly, IHT for matrix rank minimization and compressed sensing are the same except that the shrinkage operator in the matrix case is applied to the singular values, while in the compressed sensing case it is applied to the solution vector.

To prove the convergence/recoverability properties of IHT for matrix rank minimization, we need the following lemma.

Lemma 3.3.1. *Suppose $X := R_r(Y)$ is the best rank-r approximation to the matrix Y, and Γ is an SVD basis of X. Then for any rank-r matrix X_r and SVD basis Γ_r of X_r, we have*

$$\|P_B X - P_B Y\|_F \leq \|P_B X_r - P_B Y\|_F, \tag{3.3.2}$$

where B is any orthonormal set of matrices satisfying $span(\Gamma \cup \Gamma_r) \subseteq span(B)$.

Proof. Since X is the best rank-r approximation to Y and $rank(X_r) = r$, $\|X - Y\|_F \leq \|X_r - Y\|_F$. Hence,

$$\|P_B(X-Y)\|_F^2 + \|(I-P_B)(X-Y)\|_F^2 \leq \|P_B(X_r-Y)\|_F^2 + \|(I-P_B)(X_r-Y)\|_F^2.$$

Since $(I-P_B)X = 0$ and $(I-P_B)X_r = 0$, this reduces to (3.3.2). □

For IHT, we have the following convergence results, whose proofs essentially follow those given by Blumensath and Davies [8] for IHT for compressed sensing. Our first result considers the case where the desired solution X_r satisfies a perturbed linear system of equations $\mathcal{A}X_r + e = b$.

Theorem 3.3.2. *Suppose that $b = \mathcal{A}X_r + e$, where X_r is a rank-r matrix, and $\mathcal{A}$ has the RIP with $\delta_{3r}(\mathcal{A}) \leq \alpha/\sqrt{8}$ where $\alpha \in (0,1)$. Then, at iteration k, IHT will recover an approximation X^k satisfying*

$$\left\|X_r - X^k\right\|_F \leq \alpha^k \left\|X_r - X^0\right\|_F + \frac{\beta}{1-\alpha}\|e\|_2, \tag{3.3.3}$$

where $\beta := 2\sqrt{1+\alpha/\sqrt{8}}$. Furthermore, after at most $k^* := \left\lceil \log_{1/\alpha}\left(\left\|X_r - X^0\right\|_F / \|e\|_2\right)\right\rceil$ iterations, IHT estimates X^r with accuracy

$$\left\|X_r - X^{k^*}\right\|_F \leq \frac{1-\alpha+\beta}{1-\alpha}\|e\|_2. \tag{3.3.4}$$

Proof. Let Γ_r and Γ^k denote SVD bases of X_r and X^k, respectively, and B_k denote an orthonormal basis of the subspace $\text{span}(\Gamma_r \cup \Gamma^k)$. Let $Z^k := X_r - X^k$ denote the residual at iteration k. Since $P_{B_{k+1}}X_r = X_r$ and $P_{B_{k+1}}X^{k+1} = X^{k+1}$, it follows first from the triangle inequality and then from Lemma 3.3.1 that

$$\begin{aligned}\left\|X_r - X^{k+1}\right\|_F &\leq \left\|P_{B_{k+1}}X_r - P_{B_{k+1}}Y^{k+1}\right\|_F + \left\|P_{B_{k+1}}X^{k+1} - P_{B_{k+1}}Y^{k+1}\right\|_F \\ &\leq 2\left\|P_{B_{k+1}}X_r - P_{B_{k+1}}Y^{k+1}\right\|_F.\end{aligned} \tag{3.3.5}$$

Using the fact that $b = \mathcal{A}X_r + e$, $Y^{k+1} = X^k - \mathcal{A}^*(\mathcal{A}X^k - \mathcal{A}X_r - e) = X^k + \mathcal{A}^*(\mathcal{A}Z^k + e)$.

Hence, from (3.3.5),

$$\begin{aligned}
\left\|X_r - X^{k+1}\right\|_F &\leq 2\left\|P_{B_{k+1}}X_r - P_{B_{k+1}}Y^{k+1}\right\|_F \\
&\leq 2\left\|P_{B_{k+1}}X_r - P_{B_{k+1}}X^k - P_{B_{k+1}}\mathcal{A}^*\mathcal{A}(P_{B_{k+1}}Z^k + (I - P_{B_{k+1}})Z^k) - P_{B_{k+1}}\mathcal{A}^*e\right\|_F \\
&\leq 2\left\|P_{B_{k+1}}Z^k - P_{B_{k+1}}\mathcal{A}^*\mathcal{A}(P_{B_{k+1}}Z^k + (I - P_{B_{k+1}})Z^k)\right\|_F + 2\left\|P_{B_{k+1}}\mathcal{A}^*e\right\|_F \\
&\leq 2\left\|(I - P_{B_{k+1}}\mathcal{A}^*\mathcal{A}P_{B_{k+1}})P_{B_{k+1}}Z^k\right\|_F + 2\left\|P_{B_{k+1}}\mathcal{A}^*\mathcal{A}(I - P_{B_{k+1}})Z^k\right\|_F \\
&\quad + 2\left\|P_{B_{k+1}}\mathcal{A}^*e\right\|_F.
\end{aligned}$$

Since $\operatorname{rank}(P_{B_{k+1}}X) \leq 2r, \forall X \in \mathbb{R}^{m\times n}$, by applying (3.2.2) in Proposition 3.2.6 we get,

$$\left\|P_{B_{k+1}}\mathcal{A}^*e\right\|_F \leq \sqrt{1+\delta_{2r}(\mathcal{A})}\|e\|_2.$$

Since $P_\Psi P_\Psi = P_\Psi$, it follows from (3.2.3) in Proposition 3.2.6 that the eigenvalues of the linear operator $P_\Psi\mathcal{A}^*\mathcal{A}P_\Psi$ are in the interval $[1-\delta_r(\mathcal{A}), 1+\delta_r(\mathcal{A})]$. Letting $\Psi = B_{k+1}$, it follows that the eigenvalues of $P_{B_{k+1}}\mathcal{A}^*\mathcal{A}P_{B_{k+1}}$ lie in the interval $[1-\delta_{2r}(\mathcal{A}), 1+\delta_{2r}(\mathcal{A})]$. Hence the eigenvalues of $I - P_{B_{k+1}}\mathcal{A}^*\mathcal{A}P_{B_{k+1}}$ are bounded above by $\delta_{2r}(\mathcal{A})$ and it follows that

$$\left\|(I - P_{B_{k+1}}\mathcal{A}^*\mathcal{A}P_{B_{k+1}})P_{B_{k+1}}Z^k\right\|_F \leq \delta_{2r}(\mathcal{A})\left\|P_{B_{k+1}}Z^k\right\|_F.$$

Also, since $P_{B_k}Z^k = Z^k$, $Z^k \in \operatorname{span}(B_k)$ and $\operatorname{rank}(P_{B_k\cup B_{k+1}}X) \leq 3r, \forall X \in \mathbb{R}^{m\times n}$, by applying

Proposition 3.2.7 we get

$$\left\|P_{B_{k+1}}\mathcal{A}^*\mathcal{A}(I-P_{B_{k+1}})Z^k\right\|_F \leq \delta_{3r}(\mathcal{A})\left\|(I-P_{B_{k+1}})Z^k\right\|_F.$$

Thus, since $\delta_{2r}(\mathcal{A}) \leq \delta_{3r}(\mathcal{A})$,

$$\begin{aligned}\left\|X_r - X^{k+1}\right\|_F &\leq 2\delta_{2r}(\mathcal{A})\left\|P_{B_{k+1}}Z^k\right\|_F + 2\delta_{3r}(\mathcal{A})\left\|(I-P_{B_{k+1}})Z^k\right\|_F + 2\sqrt{1+\delta_{2r}(\mathcal{A})}\|e\|_2 \\ &\leq 2\sqrt{2}\delta_{3r}(\mathcal{A})\left\|Z^k\right\|_F + 2\sqrt{1+\delta_{3r}(\mathcal{A})}\|e\|_2.\end{aligned}$$

By assumption, $\delta_{3r}(\mathcal{A}) \leq \alpha/\sqrt{8}$; hence we have

$$\left\|Z^{k+1}\right\|_F \leq \alpha\left\|Z^k\right\|_F + \beta\|e\|_2. \tag{3.3.6}$$

Iterating this inequality, we get (3.3.3).

From (3.3.3), the recovery accuracy $\left\|Z^k\right\|_F \leq \frac{1-\alpha+\beta}{1-\alpha}\|e\|_2$, if $\alpha^k\left\|X_r - X^0\right\|_F \leq \|e\|_2$. Hence for $k^* := \left\lceil \log_{1/\alpha}\left(\left\|X_r - X^0\right\|_F / \|e\|_2\right)\right\rceil$, (3.3.4) holds. □

Remark 3.3.1. Note that in Theorem 3.3.2, convergence is guaranteed for any $\alpha \in (0,1)$. For the choice $\alpha = \frac{1}{2}$, $\beta = 2\sqrt{1+1/\sqrt{32}} \approx 2.1696$. Thus (3.3.3) becomes

$$\left\|X_r - X^k\right\|_F \leq 2^{-k}\left\|X_r - X^0\right\|_F + 4.3392\|e\|_2,$$

and (3.3.4) becomes

$$\left\|X_r - X^{k^*}\right\|_F \leq 5.3392\|e\|_2.$$

For an arbitrary matrix X, we have the following result.

Theorem 3.3.3. *Suppose that $b = \mathcal{A}X + e$, where X is an arbitrary matrix, and $\mathcal{A}$ has the RIP with $\delta_{3r}(\mathcal{A}) \leq \alpha/\sqrt{8}$ where $\alpha \in (0,1)$. Let X_r be the best rank-r approximation to X. Then, at iteration k, IHT will recover an approximation X^k satisfying*

$$\left\|X - X^k\right\|_F \leq \alpha^k \left\|X_r - X^0\right\|_F + \gamma\tilde{\varepsilon}_r, \tag{3.3.7}$$

where $\gamma := \frac{\beta^2}{2(1-\alpha)} + 1$, $\beta := 2\sqrt{1+\alpha/\sqrt{8}}$, and

$$\tilde{\varepsilon}_r = \|X - X_r\|_F + \frac{1}{\sqrt{r}}\|X - X_r\|_* + \|e\|_2, \tag{3.3.8}$$

is called the unrecoverable energy (see [74]). Furthermore, after at most $k^* := \left\lceil \log_{1/\alpha}\left(\left\|X_r - X^0\right\|_F / \tilde{\varepsilon}_r\right)\right\rceil$ iterations, IHT estimates X with accuracy

$$\left\|X - X^{k^*}\right\|_F \leq (1+\gamma)\tilde{\varepsilon}_r. \tag{3.3.9}$$

Proof. From Theorem 3.3.2 with $\tilde{e} = \mathcal{A}(X - X_r) + e$ instead of e, we have

$$\left\|X_r - X^k\right\|_F = \left\|Z^k\right\|_F \leq \alpha^k \left\|X_r - X^0\right\|_F + \frac{\beta}{1-\alpha}\|\tilde{e}\|_2.$$

By Proposition 3.2.8, we know that

$$\|\tilde{e}\|_2 \leq \|\mathcal{A}(X-X_r)\|_F + \|e\|_2 \leq \sqrt{1+\delta_r(\mathcal{A})}\left(\|X-X_r\|_F + \frac{1}{\sqrt{r}}\|X-X_r\|_*\right) + \|e\|_2.$$

Thus we have from the triangle inequality and (3.3.8)

$$\begin{aligned}\left\|X-X^k\right\|_F &\leq \left\|X_r-X^k\right\|_F + \|X-X_r\|_F \\ &\leq \alpha^k\left\|X_r-X^0\right\|_F + \frac{\beta}{1-\alpha}\|\tilde{e}\|_2 + \|X-X_r\|_F \\ &\leq \alpha^k\left\|X_r-X^0\right\|_F + \left(\frac{\beta}{1-\alpha}\sqrt{1+\delta_r(\mathcal{A})}+1\right)\tilde{\varepsilon}_r \\ &\leq \alpha^k\left\|X_r-X^0\right\|_F + \gamma\tilde{\varepsilon}_r.\end{aligned}$$

This proves (3.3.7).

Furthermore, $\left\|X-X^k\right\|_F \leq (1+\gamma)\tilde{\varepsilon}_r$ if $\alpha^k\left\|X_r-X^0\right\|_F \leq \tilde{\varepsilon}_r$. Therefore, for $k^* := \left\lceil \log_{1/\alpha}\left(\left\|X_r-X^0\right\|_F/\tilde{\varepsilon}_r\right)\right\rceil$, (3.3.9) holds. □

Remark 3.3.2. For the choice $\alpha = \frac{1}{2}$, $\beta = 2\sqrt{1+1/\sqrt{32}} \approx 2.1696$ and $\gamma = \frac{\beta^2}{2(1-\alpha)} + 1 \approx 5.7072$. Thus (3.3.7) holds as

$$\left\|X-X^k\right\|_F \leq 2^{-k}\left\|X_r-X^0\right\|_F + 5.7072\tilde{\varepsilon}_r,$$

and (3.3.9) holds as

$$\left\|X-X^{k^*}\right\|_F \leq 6.7072\tilde{\varepsilon}_r.$$

Similar bounds on the RIP constant for an approximate recovery were obtained by Lee and Bresler [60, 61] for affinely constrained matrix rank minimization and by Lee and Bresler for ellipsoidally constrained matrix rank minimization [62]. The results in Theorems 3.3.2 and 3.3.3 improve the previous results for affinely constrained matrix rank minimization in [60, 61]. Specifically, Theorems 3.3.2 and 3.3.3 require the RIP constant $\delta_{3r}(\mathcal{A}) < 1/\sqrt{8} \approx 0.3536$, while the result in [60,61] requires $\delta_{4r}(\mathcal{A}) \leq 0.04$ and the result in [62] requires $\delta_{3r}(\mathcal{A}) < 1/(1+4/\sqrt{3}) \approx 0.3022$ for recovery in the noisy case. The IHT algorithm for matrix rank minimization has also been independently studied by Meka, Jain and Dhillon in [71], who obtained very different results than those in Theorems 3.3.2 and 3.3.3.

3.4 Iterative Hard Thresholding with Matrix Shrinkage

We study another variant of FPCA in this section. If in each iteration of IHT, we perform matrix shrinkage to $R_r(Y)$ with fixed thresholding $\mu > 0$, we get the following algorithm (Algorithm 3.3), which we call Iterative Hard Thresholding with Matrix Shrinkage (IHTMS). Note that $S_\mu(R_r(Y)) = R_r(S_\mu(Y)), \forall r, \mu$ and Y.

Algorithm 3.3 Iterative Hard Thresholding with Matrix Shrinkage (IHTMS)

Initialization: Given X^0, μ and r.
for k = 0,1,... **do**
  $Y^{k+1} := X^k - \mathcal{A}^*(\mathcal{A}X^k - b)$.
  $X^{k+1} := R_r(S_\mu(Y^{k+1}))$.
end for

For IHTMS, we have the following convergence results.

Theorem 3.4.1. *Suppose that $b = \mathcal{A}X_r + e$, where X_r is a rank-r matrix, and $\mathcal{A}$ has the RIP with $\delta_{3r}(\mathcal{A}) \leq \alpha/\sqrt{8}$ where $\alpha \in (0,1)$. Then, at iteration k, IHTMS will recover an approximation X^k satisfying*

$$\left\|X_r - X^k\right\|_F \leq \alpha^k \left\|X_r - X^0\right\|_F + \frac{1}{1-\alpha}(\beta\|e\|_2 + 2\mu\sqrt{m}), \tag{3.4.1}$$

where $\beta := 2\sqrt{1+\alpha/\sqrt{8}}$. Furthermore, after at most
$k^* := \left\lceil \log_{1/\alpha}\left(\left\|X_r - X^0\right\|_F / (\|e\|_2 + 2\mu\sqrt{m})\right)\right\rceil$ iterations, IHTMS estimates X_r with accuracy

$$\left\|X_r - X^{k^*}\right\|_F \leq \frac{1-\alpha+\beta}{1-\alpha}\|e\|_2 + \frac{2-\alpha}{1-\alpha}2\mu\sqrt{m}. \tag{3.4.2}$$

Proof. Using the same notation as in the proof of Theorem 3.3.2, we know that $P_{B_{k+1}}X_r = X_r$ and $P_{B_{k+1}}X^{k+1} = X^{k+1}$. Using the triangle inequality we get,

$$\begin{aligned}\left\|X_r - X^{k+1}\right\|_F \leq &\left\|P_{B_{k+1}}X_r - P_{B_{k+1}}Y^{k+1}\right\|_F \\ &+ \left\|P_{B_{k+1}}X^{k+1} - P_{B_{k+1}}S_\mu(Y^{k+1})\right\|_F \\ &+ \left\|P_{B_{k+1}}S_\mu(Y^{k+1}) - P_{B_{k+1}}Y^{k+1}\right\|_F.\end{aligned} \tag{3.4.3}$$

Since X^{k+1} is the best rank-r approximation to $S_\mu(Y^{k+1})$, by applying Lemma 3.3.1 we get

$$\begin{aligned}\left\|P_{B_{k+1}}X^{k+1}-P_{B_{k+1}}S_\mu(Y^{k+1})\right\|_F &\le \left\|P_{B_{k+1}}X_r-P_{B_{k+1}}S_\mu(Y^{k+1})\right\|_F \\ &\le \left\|P_{B_{k+1}}X_r-P_{B_{k+1}}Y^{k+1}\right\|_F \\ &\quad+\left\|P_{B_{k+1}}S_\mu(Y^{k+1})-P_{B_{k+1}}Y^{k+1}\right\|_F.\end{aligned} \tag{3.4.4}$$

Therefore, by combining (3.4.3), (3.4.4) and noticing that

$$\left\|P_{B_{k+1}}S_\mu(Y^{k+1})-P_{B_{k+1}}Y^{k+1}\right\|_F \le \left\|S_\mu(Y^{k+1})-Y^{k+1}\right\|_F \le \mu\sqrt{m},$$

we have

$$\left\|X_r-X^{k+1}\right\|_F \le 2\left\|P_{B_{k+1}}X_r-P_{B_{k+1}}Y^{k+1}\right\|_F+2\mu\sqrt{m}.$$

Using an argument identical the one below (3.3.5) in the proof of Theorem 3.3.2, we get

$$\left\|X_r-X^{k+1}\right\|_F \le 2\sqrt{2}\delta_{3r}(\mathcal{A})\left\|Z^k\right\|_F+2\sqrt{1+\delta_{3r}(\mathcal{A})}\|e\|_2+2\mu\sqrt{m}.$$

Now since $\delta_{3r}(\mathcal{A}) \le \alpha/\sqrt{8}$, we have

$$\left\|Z^{k+1}\right\|_F \le \alpha\left\|Z^k\right\|_F+\beta\|e\|_2+2\mu\sqrt{m},$$

which implies that (3.4.1) holds. Hence (3.4.2) holds if

$k^* := \left\lceil \log_{1/\alpha}\left(\left\|X_r-X^0\right\|_F/(\|e\|_2+2\mu\sqrt{m})\right)\right\rceil$. □

For an arbitrary matrix X, we have the following results.

Theorem 3.4.2. *Suppose that $b = \mathcal{A}X + e$, where X is an arbitrary matrix, and $\mathcal{A}$ has the RIP with $\delta_{3r}(\mathcal{A}) \leq \alpha/\sqrt{8}$ where $\alpha \in (0,1)$. Let X_r be the best rank-r approximation to X. Then, at iteration k, IHTMS will recover an approximation X^k satisfying*

$$\left\|X - X^k\right\|_F \leq \alpha^k \left\|X_r - X^0\right\|_F + \gamma\tilde{\varepsilon}_r + \frac{2\mu\sqrt{m}}{1-\alpha}, \tag{3.4.5}$$

where $\gamma := \frac{\beta^2}{2(1-\alpha)} + 1$, $\beta := 2\sqrt{1+\alpha/\sqrt{8}}$, and $\tilde{\varepsilon}_r$ is defined by (3.3.8)*. Furthermore, after at most $k^* := \left\lceil \log_{1/\alpha}\left(\left\|X_r - X^0\right\|_F / (\tilde{\varepsilon}_r + 2\mu\sqrt{m})\right)\right\rceil$ iterations, IHTMS estimates X with accuracy*

$$\left\|X - X^{k^*}\right\|_F \leq (1+\gamma)\tilde{\varepsilon}_r + \frac{2-\alpha}{1-\alpha}2\mu\sqrt{m}. \tag{3.4.6}$$

Proof. The proof of (3.4.5) is identical to the proof of (3.3.7) in Theorem 3.3.3, except that (3.4.1) is used instead of (3.3.3). It also immediately follows from (3.4.5) that (3.4.6) holds for $k^* := \left\lceil \log_{1/\alpha}\left(\left\|X_r - X^0\right\|_F / (\tilde{\varepsilon}_r + 2\mu\sqrt{m})\right)\right\rceil$. □

3.5 FPCA with Given Rank r

In this section, we study the FPCA when rank r is known and a unit stepsize $\tau = 1$ is always chosen. This is equivalent to applying a continuation strategy to μ in IHTMS. We call this algorithm FPCAr (see Algorithm 3.4 below). The parameter η_μ determines the

rate of reduction of the consecutive μ_j in continuation, i.e.,

$$\mu_{j+1} = \max\{\mu_j \eta_\mu, \bar{\mu}\}, j = 1, \ldots, L-1 \tag{3.5.1}$$

For FPCAr, we have the following convergence results.

Algorithm 3.4 FPCA with given rank r (FPCAr)

Initialization: $X^0_{(1)}, r, \mu_1 > \mu_2 \ldots > \mu_L = \bar{\mu}$.
for j = 1,...,L **do**
  Set $\mu = \mu_j$.
  for k = 0,1,..., until convergence **do**
    $Y^{k+1}_{(j)} := X^k_{(j)} - \mathcal{A}^*\left(\mathcal{A}X^k_{(j)} - b\right)$.
    $X^{k+1}_{(j)} := S_\mu\left(R_r\left(Y^{k+1}_{(j)}\right)\right)$.
  end for
  Set $X^0_{(j+1)} = X^{k+1}_{(j)}$.
end for
Output: $X^* := X^0_{(L+1)}$.

Theorem 3.5.1. *Suppose that $b = \mathcal{A}X_r + e$, where X_r is a rank-r matrix, and $\mathcal{A}$ has the RIP with $\delta_{3r}(\mathcal{A}) \leq \alpha/\sqrt{8}$ where $\alpha \in (0,1)$. Also, suppose in FPCAr, after K_j iterations with fixed $\mu = \mu_j$, we obtain a solution $X^{(K_j)}_{(j)}$ that is then set to the initial point $X^0_{(j+1)}$ for the next continuation subproblem $\mu = \mu_{j+1}$. Then FPCAr will recover an approximation $X^{(K_L)}_{(L)}$ that satisfies*

$$\begin{aligned}\left\|X_r - X^{(K_L)}_{(L)}\right\|_F \leq \left(\alpha^{\Sigma_{j=1}^L K_j}\right)\left\|X_r - X^0\right\|_F + \left(\sum_{j=2}^{L} \alpha^{\Sigma_{l=j}^L K_l} + 1\right)\frac{\beta}{1-\alpha}\|e\|_2 \\ + \left(\sum_{j=2}^{L}\left(\alpha^{\Sigma_{l=j}^L K_l}\right)\mu_{j-1} + \mu_L\right)\frac{2\sqrt{m}}{1-\alpha},\end{aligned} \tag{3.5.2}$$

where $\beta := 2\sqrt{1+\alpha/\sqrt{8}}$.

Proof. For $X_{(1)}^{(K_1)}$, which is obtained by setting $\mu = \mu_1$ in the first K_1 iterations, we get from Theorem 3.4.1, that if $\delta_{3r}(\mathcal{A}) \leq \alpha/\sqrt{8}$,

$$\left\|X_r - X_{(1)}^{(K_1)}\right\|_F \leq \alpha^{K_1} \left\|X_r - X^0\right\|_F + \frac{\beta}{1-\alpha}\|e\|_2 + \frac{2\mu_1\sqrt{m}}{1-\alpha}. \tag{3.5.3}$$

Then from iteration $K_1 + 1$ to $K_1 + K_2$, we fix $\mu = \mu_2$. Again by Theorem 3.4.1, we get

$$\left\|X_r - X_{(2)}^{(K_2)}\right\|_F \leq \alpha^{K_2} \left\|X_r - X_{(1)}^{(K_1)}\right\|_F + \frac{\beta}{1-\alpha}\|e\|_2 + \frac{2\mu_2\sqrt{m}}{1-\alpha}. \tag{3.5.4}$$

By substituting (3.5.3) into (3.5.4), we get

$$\begin{aligned}\left\|X_r - X_{(2)}^{(K_2)}\right\|_F \leq & \alpha^{(K_1+K_2)} \left\|X_r - X^0\right\|_F + \left(\alpha^{K_2} + 1\right) \frac{\beta}{1-\alpha}\|e\|_2 \\ & + \left(\alpha^{K_2}\mu_1 + \mu_2\right) \frac{2\sqrt{m}}{1-\alpha}.\end{aligned}$$

Repeating this procedure we get (3.5.2). □

Theorem 3.5.1 shows that as long as μ_L is small and K_L is large, the recovery error will be very small. For an arbitrary matrix X, we have the following convergence result.

Theorem 3.5.2. *Suppose that $b = \mathcal{A}X + e$, where X is an arbitrary matrix. Let X_r be the best rank-r approximation to X. With the same notation and under the same conditions as*

in Theorem 3.5.1, FPCAr will recover an approximation $X_{(L)}^{(K_L)}$ *that satisfies*

$$\left\|X - X_{(L)}^{(K_L)}\right\|_F \leq \left(\alpha^{\Sigma_{j=1}^{L} K_j}\right) \left\|X_r - X^0\right\|_F + \left(\left(\sum_{j=2}^{L} \alpha^{\Sigma_{l=j}^{L} K_l} + 1\right)\gamma + 1\right)\tilde{\varepsilon}_r + \left(\sum_{j=2}^{L}\left(\alpha^{\Sigma_{l=j}^{L} K_l}\right)\mu_{j-1} + \mu_L\right)\frac{2\sqrt{m}}{1-\alpha},$$

where $\gamma := \frac{\beta^2}{2(1-\alpha)} + 1$, $\beta := 2\sqrt{1+\alpha/\sqrt{8}}$, *and* $\tilde{\varepsilon}_r$ *is defined by* (3.3.8).

Proof. We skip the proof here since it is similar to the proof of Theorem 3.3.3. □

3.6 Practical Issues

In practice, the rank r of the optimal solution is usually unknown. Thus, in every iteration, we need to determine r appropriately. We propose some heuristics for doing this here. We start with $r := r_{\max} = \lfloor (m+n-\sqrt{(m+n)^2-4p})/2 \rfloor$, which is for a given number of entries sampled, the largest rank of $m \times n$ matrices for which the matrix completion problem has a unique solution. So X^1 is a rank-$r_{\max}$ matrix. For the k-th iteration ($k \geq 2$), r is chosen as the number of singular values of X^{k-1} that are greater than $\varepsilon_s \sigma_1^{k-1}$, where σ_1^{k-1} is the largest singular value of X^{k-1} and $\varepsilon_s \in (0,1)$ is a given tolerance. Sometimes the given tolerance truncates too many of the singular values, so we need to increase r occasionally. One way to do this is to increase r by 1 whenever the non-expansive property (see [68]) of the shrinkage operator S_μ is violated some fixed number of times, say 10. In the numerical experiments described in Section 3.7, we used another strategy; i.e., we increased r by 1 whenever the Frobenius norm of the gradient g increased by more than 10 times. We

tested this heuristic for determining r extensively. It enables our algorithms to achieve very good recoverability and appears to be very robust. For many examples, our algorithms can recover matrices whose rank is almost $r_{\max}$ with a limited number of measurements.

Another issue in practice is concerned with the SVD computation. Note that in IHT, IHTMS and FPCA, we need to compute the best rank-r approximation to Y^{k+1} at every iteration. This can be very expensive even if we use a state-of-the-art code like PROPACK [59], especially when the rank of the matrix is relatively large. Therefore, we used instead the Monte Carlo algorithm LinearTimeSVD [30] described in Section 2.6 to approximate the best rank-r approximation (see Algorithm 2.3). Although PROPACK is more accurate than this Monte Carlo method (Algorithm 2.3), we observed from our numerical experiments that our algorithms are very robust and are not very sensitive to the accuracy of the approximate SVDs.

In the j-th inner iteration in FPCA we solve problem (3.1.3) for a fixed $\mu = \mu_j$; and stop when

$$\frac{\|X^{k+1} - X^k\|_F}{\max\{1, \|X^k\|_F\}} < xtol, \tag{3.6.1}$$

where $xtol$ is a small positive number. We then decrease μ and go to the next inner iteration.

3.7 Numerical Experiments

In this section, we present numerical results for the algorithms discussed above and provide comparisons with the SDP solver SDPT3 [92]. We use IHTr, IHTMSr, FPCAr to denote algorithms in which the rank r is specified, and IHT, IHTMS, FPCA to denote those in which r is determined by the heuristics described in Section 3.6. We tested these six algorithms on both randomly created and realistic matrix rank minimization problems (3.1.1). IHTr, IHT, IHTMSr and IHTMS were terminated when (3.6.1) holds. FPCAr and FPCA were terminated when both (3.6.1) holds and $\mu_k = \bar{\mu}$. All numerical experiments were run in MATLAB 7.3.0 on a Dell Precision 670 workstation with an Intel xeon(TM) 3.4GHZ CPU and 6GB of RAM. All CPU times reported in this section are in seconds.

3.7.1 Randomly Created Test Problems

We tested some randomly created problems to illustrate the recoverability/convergence properties of our algorithms. The random test problems (3.1.1) were created in the following manner. We first generated random matrices $M_L \in \mathbb{R}^{m\times r}$ and $M_R \in \mathbb{R}^{n\times r}$ with i.i.d. Gaussian entries $\sim \mathcal{N}(0,1)$ and then set $M = M_L M_R^\top$. We then created a matrix $A \in \mathbb{R}^{p\times mn}$ with i.i.d. Gaussian entries $A_{ij} \sim \mathcal{N}(0,1/p)$. Finally, the observation b was set equal to $b := A\mathrm{vec}(M)$. We use $SR = p/(mn)$, i.e., the number of measurements divided by the number of entries of the matrix, to denote the sampling ratio. We also list $FR = r(m+n-r)/p$, i.e. the dimension of the set of rank r matrices divided by the number of measurements, in the tables. Note that if $FR > 1$, then there is always an infinite number of matrices with

rank r satisfying the p linear constraints, so we cannot hope to recover the matrix in this situation. We also report the relative error

$$rel.err. := \frac{\|X_{opt} - M\|_F}{\|M\|_F}$$

to indicate the closeness of X_{opt} to M, where X_{opt} is the optimal solution to (3.1.1) produced by our algorithms. We declared M to be recovered if the relative error was less than 10^{-3}. We solved 10 randomly created matrix rank minimization problems for each set of (m,n,p,r). We used NS to denote the number of matrices that were recovered successfully. The average time and average relative error of the successfully solved problems are also reported.

The parameters used in the algorithms are summarized in Table 3.1.

Table 3.1: Parameters used in the algorithms

parameter	value	description
$\bar{\mu}$	10^{-8}	parameter in Algorithms 3.1 and 3.4
η_μ	0.25	parameter in (3.5.1)
ε_s	0.01	parameter in LinearTimeSVD
c_s	$2r_{\max} - 2$	parameter in LinearTimeSVD
p_i	$1/n, \forall i$	parameter in LinearTimeSVD
$xtol$	10^{-6}	parameter in (3.6.1)

We first compare the solvers discussed above that specify the rank r with the SDP solver SDPT3 [92]. The results for a set of small problems with $m = n = 60$, 20 percent sampling (i.e., SR = 0.2 and p = 720) and different ranks are presented in Table 3.2. Note that for this set of parameters (m,n,p), the largest rank that satisfies $FR < 1$ is $r_{\max} = 6$.

From Table 3.2 we can see that the performance of our methods is very robust and

Table 3.2: Comparison between IHTr, IHTMSr and FPCAr with SDPT3

Prob		SDPT3			IHTr			IHTMSr			FPCAr		
r	FR	NS	time	rel.err.	NS	time	rel.err.	NS	time	rel.err.	NS	time	rel.err.
1	0.17	10	122.93	2.31e-10	10	2.60	1.67e-05	10	2.59	1.67e-05	10	4.63	9.00e-06
2	0.33	10	124.26	3.46e-09	10	4.97	1.99e-05	10	4.98	2.11e-05	10	6.06	1.51e-05
3	0.49	3	149.74	2.84e-07	10	10.04	2.38e-05	10	9.95	2.27e-05	10	10.64	2.35e-05
4	0.64	0	—	—	10	22.99	2.88e-05	10	22.72	3.05e-05	10	23.29	2.93e-05
5	0.80	0	—	—	10	75.86	3.89e-05	10	84.13	3.95e-05	10	79.46	3.94e-05

quite similar in terms of their recoverability properties. They are also much faster and their abilities to recover the matrices are much better than SDPT3. For ranks less than or equal to 5, which is almost the largest rank guaranteeing $FR < 1$, IHTr, IHTMSr and FPCAr can recover all randomly generated matrices with a relative error of the order of $1e-5$. However, SDPT3 can only recover all matrices with a rank equal to 1 or 2. When the rank r increases to 3, SDPT3 can only recover 3 of the 10 matrices. When the rank r increases to 4 or 5, none of the 10 matrices can be recovered by SDPT3.

To verify the theoretical results in Sections 3.3, 3.4 and 3.5, we plotted the log of the approximation error $\|X^k - X^*\|_F$ achieved by each of the algorithms IHTr, IHTMSr and FPCAr versus the iteration number k in Figure 3.1 for one of 10 randomly created problems involving a matrix of rank 2. From this figure, we can see that $\log \|X^k - X^*\|_F$ is approximately a linear function of the iteration number k. This implies that our theoretical results in Sections 3.3, 3.4 and 3.5 approximately hold in practice.

For the same set of test problems, Tables 3.3, 3.4, and 3.5 present comparisons of IHTr versus IHT, IHTMSr versus IHTMS and FPCAr versus FPCA.

From these tables we see that by using our heuristics for determining the rank r at every iteration, algorithms IHT, IHTMS and FPCA perform similarly to algorithms IHTr,

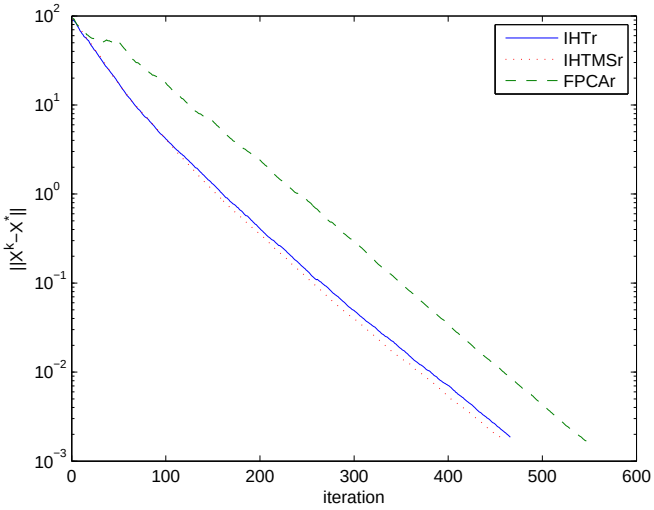

Figure 3.1: Approximation error versus the iteration number for a problem where the rank equaled 2

Table 3.3: Comparison between IHTr and IHT

Prob		IHTr			IHT		
r	FR	NS	time	rel.err.	NS	time	rel.err.
1	0.17	10	2.60	1.67e-05	10	4.24	1.74e-05
2	0.33	10	4.97	1.99e-05	10	7.00	1.92e-05
3	0.49	10	10.04	2.38e-05	10	13.27	2.32e-05
4	0.64	10	22.99	2.88e-05	10	28.06	2.93e-05
5	0.80	10	75.86	3.89e-05	10	96.32	4.00e-05

IHTMSr and FPCAr which make use of knowledge of the true rank r. Specifically, algorithms IHT, IHTMS and FPCA are capable of recovering low-rank matrices very well even when we do not know their rank.

Choosing r is crucial in algorithms IHTr, IHTMSr and FPCAr as it is in greedy algorithms for matrix rank minimization and compressed sensing. In Table 3.6 we present results on how the choice of r affects the performance of algorithms IHTr, IHTMSr and FPCAr when the true rank of the matrix is not known. In Table 3.6, the true rank is 3 and

Table 3.4: Comparison between IHTMSr and IHTMS

Prob		IHTMSr			IHTMS		
r	FR	NS	time	rel.err.	NS	time	rel.err.
1	0.17	10	2.59	1.67e-05	10	3.98	1.77e-05
2	0.33	10	4.98	2.11e-05	10	6.95	2.04e-05
3	0.49	10	9.95	2.27e-05	10	12.65	2.30e-05
4	0.64	10	22.72	3.05e-05	10	27.12	2.86e-05
5	0.80	10	84.13	3.95e-05	10	94.13	4.10e-05

Table 3.5: Comparison between FPCAr and FPCA

Prob		FPCAr			FPCA		
r	FR	NS	time	rel.err.	NS	time	rel.err.
1	0.17	10	4.63	9.00e-06	10	4.66	8.88e-06
2	0.33	10	6.06	1.51e-05	10	6.15	1.55e-05
3	0.49	10	10.64	2.35e-05	10	11.50	2.24e-05
4	0.64	10	23.29	2.93e-05	10	25.66	2.88e-05
5	0.80	10	79.46	3.94e-05	10	83.91	3.87e-05

the results for choices of the rank from 1 to 6 are presented. The rows labeled IHT, IHTMS and FPCA present the results for these algorithms which use the heuristics in Section 3.6 to determine the rank r. From Table 3.6 we see that if we specify a rank that is smaller than the true rank, then all of the algorithms IHTr, IHTMSr and FPCAr are unable to successfully recover the matrices (i.e., the relative error is greater than 1e-3). Specifically, since for the problems tested the true rank of the matrix was 3, the algorithms failed when r was chosen to be either 1 or 2. If the chosen rank is slightly greater than the true rank (i.e., the rank was chosen to be 4 or 5), all the three algorithms IHTr, IHTMSr and FPCAr still worked. However, the relative errors and times were much worse than those produced by the heuristics based solvers IHT, IHTMS and FPCA. When the chosen rank was too large (i.e., was chosen to be 6), IHTr, IHTMSr and FPCAr were only able to recover the matrices

Table 3.6: Comparison when the given rank is different from the true rank of 3

Given rank	NS	time	rel.err.
IHTr			
1	0	—	—
2	0	—	—
3	10	10.04	2.38e-05
4	10	21.42	3.42e-05
5	10	63.53	5.51e-05
6	4	109.00	4.44e-04
IHT	10	13.27	2.32e-05
IHTMSr			
1	0	—	—
2	0	—	—
3	10	9.95	2.27e-05
4	10	22.53	3.40e-05
5	10	67.89	5.93e-05
6	1	116.62	6.04e-04
IHTMS	10	12.65	2.30e-05
FPCAr			
1	0	—	—
2	0	—	—
3	10	10.64	2.35e-05
4	10	21.26	3.46e-05
5	10	63.67	5.99e-05
6	3	108.02	4.04e-04
FPCA	10	11.50	2.24e-05

in 4, 1 and 3 out of 10 problems, respectively. However, IHT, IHTMS and FPCA always recovered the matrices.

3.7.2 A Video Compression Problem

We tested the performance of our algorithms on a video compression problem. By stacking each frame of the video as a column of a large matrix, we get a matrix M whose j-th column

corresponds to the j-th frame of the video. Due to the correlation between consecutive frames of the video matrix, M is expected to be of low rank. Hence we should be able to recover the video by only taking a limited number of measurements. The video used in our experiment was downloaded from the website http://media.xiph.org/video/derf. The original colored video consisted of 300 frames where each frame was an image stored in an RGB format, as a $144 \times 176 \times 3$ array. Since this video data was too large for our use, we preprocessed it in the following way. We first converted each frame from an RGB format into a grayscale image, so each frame was a 144×176 matrix. We then used only the portion of each frame corresponding to a 39×47 submatrix of pixels in the center of each frame, and took only the first 20 frames. Consequently, the matrix M had $m = 1833$ rows and $n = 20$ columns. We then created a Gaussian sampling matrix $A \in \mathbb{R}^{p \times (mn)}$ as in Section 3.7.1 with $p = 1833 * 20 * 0.4 = 14664$ rows (i.e., we used sampling ratio $SR = 0.4$) and computed $b = A\text{vec}(M) \in \mathbb{R}^p$. This 14664×36660 matrix A was close to the size limit of what could be created by calling the MATLAB function $A = randn(p, mn)$ on our computer. Although the matrix M was expected to be of low rank, it was only approximately of low rank. Therefore, besides comparing the recovered matrices with the original matrix M, we also compared them with the best rank-5 approximation of M. Since the relative error of the best rank-5 approximation of M was $2.33e-2$, we cannot expect to get a more accurate solution. Therefore, we set $xtol$ equal to 0.002 for this problem. The results of our numerical tests are reported in Table 3.7. The ranks reported in the table are the ranks of the recovered matrices. The reported relative errors and CPU times are averages over 5 runs. We do not report any results for SDPT3, because the problem is far too large to be solved

by an SDP solver. From Table 3.7 we see that our algorithms were able to recover the matrix M very well, achieving relative errors that were of the same order as that obtained by the best rank-5 approximation.

Table 3.7: Results on recovery of compressed video

Solvers	rank	rel.err.	time
IHTr	5	6.87e-2	645
IHT	5	9.76e-2	949
IHTMSr	5	6.72e-2	688
IHTMS	5	9.69e-2	804
FPCAr	5	5.10e-2	514
FPCA	5	5.17e-2	1296

In Figure 3.2, the three images in the first column correspond to three particular frames in the original video. The images in the second column correspond to these frames in the rank-5 approximation matrix of the video. The images in the third column correspond to these frames in the matrix recovered by FPCA. The other five solvers recovered images that were very similar visually to FPCA so we do not show them here. From Figure 3.2 we see that FPCA recovers the video very well by taking only 40% as many measurements as there are pixels in the video.

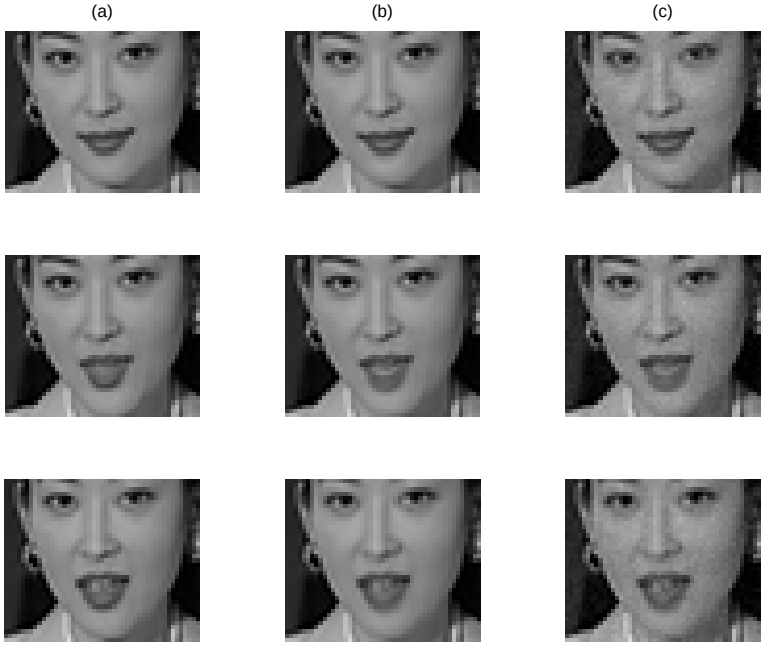

Figure 3.2: Comparison of frames 4, 12 and 18 of (a) the original video, (b) the best rank-5 approximation and (c) the matrix recovered by FPCA

Chapter 4

Fast Alternating Linearization Methods for Minimizing The Sum of Two Convex Functions

4.1 Introduction

We consider the special case of the composite convex problem (1.2.10) with $K = 2$ in this chapter, i.e.,

$$\min \quad F(x) \equiv f(x) + g(x). \tag{4.1.1}$$

Algorithms for solving problem (4.1.1) have been studied extensively in the literature. For large-scale problems, for which the problems

$$\min_{x\in\mathbb{R}^n} \quad \tau f(x)+\frac{1}{2}\|x-z\|_2^2 \tag{4.1.2}$$

and

$$\min_{x\in\mathbb{R}^n} \quad \tau g(x)+\frac{1}{2}\|x-z\|_2^2 \tag{4.1.3}$$

are relatively easy to solve, where $\tau>0, z\in\mathbb{R}^n$, the class of alternating direction methods that are based on variable splitting combined with the augmented Lagrangian method are particularly important. In these methods, one splits the variable x into two variables, i.e., one introduces a new variable y and rewrites Problem (4.1.1) as

$$\min\{f(x)+g(y) : x-y=0\}. \tag{4.1.4}$$

Since Problem (4.1.4) is an equality constrained problem, the augmented Lagrangian method can be used to solve it. Given a penalty parameter $1/\mu$, at the k-th iteration, the augmented Lagrangian method minimizes the augmented Lagrangian function

$$\mathcal{L}_\mu(x,y;\lambda) := f(x)+g(y)-\langle\lambda,x-y\rangle+\frac{1}{2\mu}\|x-y\|^2, \tag{4.1.5}$$

with respect to x and y, i.e., it solves the subproblem

$$(x^k, y^k) \quad := \arg\min_{x,y} \mathcal{L}_\mu(x, y; \lambda^k) \tag{4.1.6}$$

and then updates the Lagrange multiplier λ^k via:

$$\lambda^{k+1} \quad := \lambda^k - \tfrac{1}{\mu}(x^k - y^k). \tag{4.1.7}$$

Minimizing $\mathcal{L}_\mu(x, y; \lambda)$ with respect to x and y jointly is often not easy. In fact, it certainly is not any easier than solving the original problem (4.1.1). However, if one minimizes $\mathcal{L}_\mu(x, y; \lambda)$ with respect to x and y alternatingly, one needs to solve problems of the form (4.1.2) and (4.1.3), which as we have already discussed, is often easy to do. Such an alternating direction augmented Lagrangian method (ADAL) for solving (4.1.4) is given below as Algorithm 4.1.

Algorithm 4.1 Alternating Direction Augmented Lagrangian Method (ADAL)

Choose μ, λ^0 and $x^0 = y^0$.
for $k = 0, 1, \cdots$ **do**
  $x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$
  $y^{k+1} := \arg\min_y \mathcal{L}_\mu(x^{k+1}, y; \lambda^k)$
  $\lambda^{k+1} := \lambda^k - \frac{1}{\mu}(x^{k+1} - y^{k+1})$
end for

The history of alternating direction methods (ADMs) goes back to the 1950s for solving PDEs [29, 80] and to the 1970s for solving variational problems associated with PDEs [41,43]. ADMs have also been applied to solving variational inequality problems by Tseng [96, 97] and He et al. [51, 53]. Recently, with the emergence of compressive sensing and

subsequent great interest in ℓ_1 minimization [18, 26], ADMs have been applied to ℓ_1 and total variation regularized problems arising from signal processing and image processing. The papers of Goldstein and Osher [48], Afonso et al. [1] and Yang and Zhang [105] are based on the alternating direction augmented Lagrangian framework (Algorithm 4.1), and demonstrate that ADMs are very efficient for solving ℓ_1 and TV regularized problems. The work of Yuan [108] and Yuan and Yang [109] showed that ADMs can also efficiently solve ℓ_1-regularized problems arising from statistics and data analysis. More recently, Wen, Goldfarb and Yin [103] and Malick et al. [70] applied alternating direction augmented Lagrangian methods to solve semidefinite programming (SDP) problems. The results in [103] show that these methods greatly outperform interior point methods on several classes of well-structured SDP problems. Furthermore, He et al. proposed an alternating direction based contraction method for solving separable linearly constrained convex problems [52]. For a recent survey on the application of ADMs to distributed optimization and machine learning, see [10].

Another important and related class of algorithms for solving (4.1.1) is based on operator-splitting. The aim of these algorithms is to find an x such that

$$0 \in T_1(x) + T_2(x), \tag{4.1.8}$$

where T_1 and T_2 are maximal monotone operators. This is a more general problem than (4.1.1) and ADMs for it have been the focus of a substantial amount of research; e.g.,

see [21–23, 32, 33, 66, 86]. Since the first-order optimality conditions for (4.1.1) are

$$0 \in \partial f(x) + \partial g(x), \tag{4.1.9}$$

where $\partial f(x)$ denotes the subdifferential of $f(x)$ at the point x, a solution to Problem (4.1.1) can be obtained by solving Problem (4.1.8). For example, see [32–34,66,86] and references therein for more information on this class of algorithms.

While global convergence results for various splitting and alternating direction algorithms have been established under appropriate conditions, our interest here is on iteration complexity bounds for such algorithms. By an iteration complexity bound we mean a bound on the number of iterations needed to obtain an ε-optimal solution which is defined as follows.

Definition 4.1.1. *$x_\varepsilon \in \mathcal{C}$ is called an ε-optimal solution to*

$$\min_x \quad F(x), \quad s.t. \quad x \in \mathcal{C}, \tag{4.1.10}$$

if $F(x_\varepsilon) - F(x^*) \leq \varepsilon$, where x^* is an optimal solution to (4.1.10).

Complexity bounds for first-order methods for solving convex optimization problems have been given by Nesterov and many others. In [75, 76], Nesterov gave first-order algorithms for solving smooth unconstrained convex minimization problems with an iteration complexity of $O(\sqrt{L/\varepsilon})$, where L is the Lipschitz constant of the gradient of the objective function, and showed that this is the best complexity that is obtainable when only first-order

information is used. These methods can be viewed as accelerated gradient methods where a combination of past iterates are used to compute the next iterate. Similar techniques were then applied to nonsmooth problems [4, 77, 78, 99] and corresponding optimal complexity results were obtained. The ISTA (Iterative Shrinkage/Thresholding Algorithm) and FISTA (Fast Iterative Shrinkage/Thresholding Algorithm) algorithms proposed by Beck and Teboulle in [4] are designed for solving (4.1.1) when one of the functions (say $f(x)$) is smooth and the other is not. It is proved in [4] that the number of iterations required by ISTA and FISTA to get an ε-optimal solution to problem (4.1.1) are respectively $O(L(f)/\varepsilon)$ and $O(\sqrt{L(f)/\varepsilon})$, under the assumption that $\nabla f(x)$ is Lipschitz continuous with Lipschitz constant $L(f)$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L(f)\|x-y\|_2, \quad \forall x, y \in \mathbb{R}^n.$$

ISTA computes a sequence $\{x^k\}$ via the iteration

$$x^{k+1} := \arg\min_x Q_f(x, x^k), \tag{4.1.11}$$

where

$$Q_f(u, v) := g(u) + f(v) + \langle \nabla f(v), u - v \rangle + \frac{1}{2\mu}\|u - v\|^2, \tag{4.1.12}$$

while FISTA computes $\{x^k\}$ via the iteration

$$\begin{cases} x^k & := & \arg\min_x Q_f(x, y^k) \\ t_{k+1} & := & \left(1+\sqrt{1+4t_k^2}\right)/2 \\ y^{k+1} & := & x^k + \left(\frac{t_k-1}{t_{k+1}}\right)(x^k - x^{k-1}) \end{cases} \tag{4.1.13}$$

starting with $t_1 = 1$, $y^1 = x^0 \in \mathbb{R}^n$ and $k = 1$.

Note that ISTA and FISTA treat the functions $f(x)$ and $g(x)$ very differently. At each iteration they both linearize the function $f(x)$ but never directly minimize it, while they do minimize the function $g(x)$ in conjunction with the linearization of $f(x)$ and a proximal (penalty) term. These two methods have proved to be efficient for solving the CS problem (1.2.4) (see e.g., [4, 50]) and the nuclear norm minimization problem (1.2.8) (see e.g., [68, 93]). ISTA and FISTA work well in these areas because $f(x)$ is quadratic and is well approximated by linearization. However, for the RPCA problem (1.2.12) where two nonsmooth functions are involved, ISTA and FISTA (after first smoothing one of the functions) do not work as well. As we shall show in Sections 4.2 and 4.3, our ADMs are very effective in solving RPCA problems. For the SICS problem (1.2.13), intermediate iterates X^k may not be positive definite, and hence the gradient of $f(X) = -\log\det(X) + \langle \Sigma, X\rangle$ may not be well defined at X^k. Therefore, ISTA and FISTA cannot be used to solve the SICS problem (1.2.13). In [84], it is shown that SICS problems can be very efficiently solved by our ADM approach.

Our contribution. In this chapter, we propose both basic and accelerated (i.e., *fast*)

versions of first-order alternating linearization methods (ALMs) based on an alternating direction augmented Lagrangian approach for solving (4.1.1) and analyze their iteration complexities. Our basic methods require at most $O(L/\varepsilon)$ iterations to obtain an ε-optimal solution, while our fast methods require at most $O(\sqrt{L/\varepsilon})$ iterations with only a very small increase in the computational effort required at each iteration. Thus, our fast methods are as efficient as *optimal* first-order methods in terms of iteration complexity. These algorithms can also be viewed as extensions of the ISTA and FISTA algorithms in [4]. The complexity bounds we obtain for our algorithms are similar to (and as much as a factor of two better that) those in [4].

At each iteration, our algorithms alternatively minimize two different approximations to the original objective function, obtained by keeping one function unchanged and linearizing the other one. Our basic algorithm is similar in many ways to the alternating linearization method proposed by Kiwiel et al. [58]. In particular, the approximate functions minimized at each step of Algorithm 3.1 in [58] have the same form as those minimized in our algorithm. However, our basic algorithm differs from the one in [58] in the way that the proximal terms are chosen, and our accelerated algorithms are very different. Moreover, no complexity bounds have been given for the algorithm in [58]. To the best of our knowledge, the complexity results in this chapter are the first ones that have been given for a Gauss-Seidel type alternating direction method. Complexity results for related Jacobi type alternating direction methods are given in Chapter 5 (see also [45]).

Organization. The rest of this chapter is organized as follows. In Sections 4.2 and 4.3 we propose our alternating linearization methods based on alternating direction augmented

Lagrangian methods and give convergence/complexity bounds for them. We compare the performance of our ALMs to other competitive first-order algorithms on an image deblurring problem in Section 4.4. In Section 4.5, we apply our ALMs to solve very large RPCA problems arising from background extraction in surveillance video and matrix completion. Finally, we offer some conclusion in Section 4.6.

4.2 Alternating Linearization Methods

In each iteration of the ADAL method, Algorithm 4.1, the Lagrange multiplier λ is updated just once, immediately after the augmented Lagrangian is minimized with respect to y. For the alternating direction approach to be symmetric with respect to x and y, one should also update λ after solving the subproblem with respect to x. Such a symmetric version of the ADAL method is given below as Algorithm 4.2.

Algorithm 4.2 Symmetric Alternating Direction Augmented Lagrangian Method (SADAL)

Choose μ, λ^0 and $x^0 = y^0$.
for $k = 0, 1, \cdots$ **do**
  $x^{k+1} := \arg\min_x \mathcal{L}_\mu(x, y^k; \lambda^k)$
  $\lambda^{k+\frac{1}{2}} := \lambda^k - \frac{1}{\mu}(x^{k+1} - y^k)$
  $y^{k+1} := \arg\min_y \mathcal{L}_\mu(x^{k+1}, y; \lambda^{k+\frac{1}{2}})$
  $\lambda^{k+1} := \lambda^{k+\frac{1}{2}} - \frac{1}{\mu}(x^{k+1} - y^{k+1})$
end for

This ADAL variant is described and analyzed in [43]. Moreover, it is shown in [43] that Algorithms 4.1 and 4.2 are equivalent to the Douglas-Rachford [29] and Peaceman-Rachford [80] methods, respectively applied to the optimality condition (4.1.9) for problem

(4.1.1). If we assume that both $f(x)$ and $g(x)$ are differentiable, it follows from the first-order optimality conditions for the two subproblems in lines 3 and 5 of Algorithm 4.2 that

$$\lambda^{k+\frac{1}{2}} = \nabla f(x^{k+1}) \quad \text{and} \quad \lambda^{k+1} = -\nabla g(y^{k+1}). \tag{4.2.1}$$

Substituting (4.2.1) into Algorithm 4.2, we get the following alternating linearization method (ALM) which is equivalent to the SADAL method (Algorithm 4.2) when both f and g are differentiable. In Algorithm 4.3, $Q_f(u,v)$ is defined by (4.1.12) and

$$Q_g(u,v) := f(u) + g(v) + \langle \nabla g(v), u - v \rangle + \frac{1}{2\mu}\|u - v\|_2^2. \tag{4.2.2}$$

Algorithm 4.3 Alternating Linearization Method (ALM)

Choose μ and $x^0 = y^0$.
for $k = 0, 1, \cdots$ **do**
  $x^{k+1} := \arg\min_x Q_g(x, y^k)$
  $y^{k+1} := \arg\min_y Q_f(y, x^{k+1})$
end for

In Algorithm 4.3, we alternatively replace the functions g and f by their linearizations plus a proximal regularization term to get an approximation to the original function F. Thus, our ALM algorithm can also be viewed as a proximal-point algorithm.

We show in the following that the iteration complexity of Algorithm 4.3 is $O(1/\varepsilon)$ for obtaining an ε-optimal solution for (4.1.1). First, we need the following lemmas before we prove these complexity results. The first lemma is the following fundamental property for smooth functions in the class $C^{1,1}$; see e.g., [6].

Lemma 4.2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex function in the class $C^{1,1}$ with Lipschitz constant $L(f)$. Then for any $L \geq L(f)$,*

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|_2^2, \quad \forall x, y \in \mathbb{R}^n.$$

Lemma 4.2.2. *The following inequalities hold for any u, v, $\mu \leq 1/\max\{L(f), L(g)\}$:*

$$2\mu(F(u) - F(p_f(v))) \geq \|p_f(v) - u\|^2 - \|v - u\|^2, \tag{4.2.3}$$

$$2\mu(F(u) - F(p_g(v))) \geq \|p_g(v) - u\|^2 - \|v - u\|^2, \tag{4.2.4}$$

where

$$p_f(v) := \arg\min_u \quad Q_f(u, v), \tag{4.2.5}$$

and

$$p_g(v) := \arg\min_u \quad Q_g(u, v). \tag{4.2.6}$$

Proof. Since $\mu \leq 1/\max\{L(f), L(g)\}$, from Lemma 4.2.1 we have

$$f(p_f(v)) \leq f(v) + \langle p_f(v) - v, \nabla f(v) \rangle + \frac{1}{2\mu}\|p_f(v) - v\|^2,$$

which implies that $F(p_f(v)) \leq Q_f(p_f(v), v)$. Then it follows that

$$
\begin{aligned}
& F(u) - F(p_f(v)) \qquad (4.2.7)\\
\geq & F(u) - Q_f(p_f(v), v)\\
= & f(u) + g(u) - (f(v) + \langle p_f(v) - v, \nabla f(v)\rangle + \frac{1}{2\mu}\|p_f(v) - v\|^2 + g(p_f(v)))\\
\geq & \langle u - v, \nabla f(v)\rangle + \langle u - p_f(v), \nabla g(p_f(v))\rangle - \langle p_f(v) - v, \nabla f(v)\rangle - \frac{1}{2\mu}\|p_f(v) - v\|^2\\
= & \langle u - p_f(v), \nabla f(v) + \nabla g(p_f(v))\rangle - \frac{1}{2\mu}\|p_f(v) - v\|^2\\
= & \langle u - p_f(v), -\frac{1}{\mu}(p_f(v) - v)\rangle - \frac{1}{2\mu}\|p_f(v) - v\|^2\\
= & \frac{1}{2\mu}(\|p_f(v) - u\|^2 - \|v - u\|^2),
\end{aligned}
$$

where the second inequality is from the convexity of functions f and g, the third equality is due to the first-order optimality conditions of (4.2.5):

$$\nabla g(p_f(v)) + \nabla f(v) + \frac{1}{\mu}(p_f(v) - v) = 0. \qquad (4.2.8)$$

Thus we proved (4.2.3). (4.2.4) can be proved similarly. □

Now we are ready to give the complexity result of ALM.

Theorem 4.2.3. *Assume ∇f and ∇g are both Lipschitz continuous with Lipschitz constants $L(f)$ and $L(g)$, respectively. The sequence $\{x^k, y^k\}$ generated via ALM with $\mu \leq$*

$1/\max\{L(f),L(g)\}$ *satisfy*

$$F(y^k) - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{4\mu k}. \tag{4.2.9}$$

Moreover, if $1/(\beta\max\{L(f),L(g)\}) \leq \mu \leq 1/\max\{L(f),L(g)\}$ *where* $\beta \geq 1$*, the number of iterations needed to get an* ε*-optimal solution is at most* $\lceil C/\varepsilon \rceil$*, where* $C = \beta\max\{L(f),L(g)\}\|x^0 - x^*\|^2/4$.

Proof. In (4.2.3), by letting $u = x^*, v = x^{n+1}$, we get

$$2\mu(F(x^*) - F(y^{n+1})) \geq \|y^{n+1} - x^*\|^2 - \|x^{n+1} - x^*\|^2. \tag{4.2.10}$$

Similarly, by letting $u = x^*, v = y^n$ in (4.2.4) we get

$$2\mu(F(x^*) - F(x^{n+1})) \geq \|x^{n+1} - x^*\|^2 - \|y^n - x^*\|^2. \tag{4.2.11}$$

Taking the summation of (4.2.10) and (4.2.11) we get

$$2\mu(2F(x^*) - F(x^{n+1}) - F(y^{n+1})) \geq \|y^{n+1} - x^*\|^2 - \|y^n - x^*\|^2. \tag{4.2.12}$$

Again in (4.2.3), by letting $u = v = x^{n+1}$ we get

$$2\mu(F(x^{n+1}) - F(y^{n+1})) \geq \|y^{n+1} - x^{n+1}\|^2 \geq 0. \tag{4.2.13}$$

Similarly, by letting $u = v = y^n$ in (4.2.4) we get

$$2\mu(F(y^n) - F(x^{n+1})) \geq \|x^{n+1} - y^n\|^2 \geq 0. \tag{4.2.14}$$

Summing (4.2.13) and (4.2.14) we get

$$F(y^n) - F(y^{n+1}) \geq 0. \tag{4.2.15}$$

Combining (4.2.12) and (4.2.13) we get

$$2\mu(2F(x^*) - 2F(y^{n+1})) \geq \|y^{n+1} - x^*\|^2 - \|y^n - x^*\|^2. \tag{4.2.16}$$

Summing (4.2.16) over $n = 0, 1, \ldots, k-1$ and combining with (4.2.15) we get

$$2\mu(2kF(x^*) - 2kF(y^k)) \geq \|y^k - x^*\|^2 - \|y^0 - x^*\|^2 \geq -\|x^0 - x^*\|^2. \tag{4.2.17}$$

It follows immediately from (4.2.17) that (4.2.9) holds. □

Remark 4.2.1. More general problems of the form

$$\begin{aligned} \min \quad & f(x) + g(y) \\ \text{s.t.} \quad & Ax + y = b \end{aligned} \tag{4.2.18}$$

are easily handled by our approach, since one can express (4.2.18) as

$$\min \quad f(x)+g(b-Ax).$$

Remark 4.2.2. If a convex constraint $x \in \mathcal{C}$, where $\mathcal{C}$ is a convex set is added to problem (4.1.1), and we impose this constraint in the two subproblems in Algorithm 4.3, i.e., we impose $x \in \mathcal{C}$ in the subproblem with respect to x and $y \in \mathcal{C}$ in the subproblem with respect to y, the complexity results in Theorem 4.2.3 continue to hold. The only changes in the proof are in Lemma 4.2.2. If there is a constraint $x \in \mathcal{C}$, then (4.2.3) and (4.2.4) hold for any $u \in \mathcal{C}$ and $v \in \mathcal{C}$. Also in the proof of Lemma 4.2.2, the third equality in (4.2.7) becomes a "$\geq$" inequality due to the fact that the optimality conditions (4.2.8) become

$$\langle \nabla g(p_f(v)) + \nabla f(v) + \frac{1}{\mu}(p_f(v)-v), u-p_f(v)\rangle \geq 0, \forall u \in \mathcal{C}.$$

Remark 4.2.3. Although Algorithm 4.3 assumes that the Lipschitz constants are known, and hence that an upper bound for μ is known, this can be relaxed by using the backtracking technique in [4] to estimate μ at each iteration.

4.3 Fast Alternating Linearization Methods

In this section, we propose a fast alternating linearization method (FALM) which computes an ε-optimal solution to problem (4.1.1) in $O(\sqrt{L/\varepsilon})$ iterations, while keeping the work at each iteration almost the same as that required by ALM.

FALM is an accelerated version of ALM for solving (4.1.1), or equivalently (4.1.4), when $f(x)$ and $g(x)$ are both differentiable, and is given below as Algorithm 4.4. Clearly, FALM is also a Gauss-Seidel type algorithm. In fact, it is a successive over-relaxation type algorithm since $(t_k-1)/t_{k+1} > 0, \forall k \geq 2$.

Algorithm 4.4 Fast Alternating Linearization Method (FALM)

Choose μ and $x^0 = y^0 = z^1$, set $t_1 = 1$.
for $k = 1, 2, \cdots$ **do**
  $x^k := \arg\min_x Q_g(x, z^k)$
  $y^k := \arg\min_y Q_f(y, x^k)$
  $t_{k+1} := (1 + \sqrt{1+4t_k^2})/2$
  $z^{k+1} := y^k + \frac{t_k - 1}{t_{k+1}}(y^k - y^{k-1})$
end for

In the following two lemmas and one theorem, we derive iteration-complexity bounds for Algorithm 4.4.

Lemma 4.3.1. *Assume ∇f and ∇g are both Lipschitz continuous with Lipschitz constants $L(f)$ and $L(g)$, respectively. The sequences $\{x^k, y^k\}$ generated via Algorithm 4.4 with $\mu \leq 1/\max\{L(f), L(g)\}$ satisfy*

$$2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1}) \geq \|\mathbf{u}^{k+1}\|^2 - \|\mathbf{u}^k\|^2, \tag{4.3.1}$$

where $v_k := F(x^k) + F(y^k) - 2F(x^*)$, $\mathbf{u}^k := t_k y^k - (t_k - 1)\delta^{k-1} - x^*$, $\delta^k = (x^k + y^k)/2$.

Proof. In (4.2.3), by letting $u = \delta^k$, $v = x^{k+1}$ we get

$$2\mu(F(\delta^k) - F(y^{k+1}) \geq \|y^{k+1} - \delta^k\|^2 - \|x^{k+1} - \delta^k\|^2. \tag{4.3.2}$$

By letting $u = \delta^k$, $v = z^{k+1}$ in (4.2.4) we get

$$2\mu(F(\delta^k) - F(x^{k+1})) \geq \|x^{k+1} - \delta^k\|^2 - \|z^{k+1} - \delta^k\|^2 \tag{4.3.3}$$

Taking the summation of (4.3.2) and (4.3.3), and using the facts that $\delta^k = (x^k + y^k)/2$ and $F(\cdot)$ is convex, we obtain,

$$2\mu(v_k - v_{k+1}) \geq \|y^{k+1} - \delta^k\|^2 - \|z^{k+1} - \delta^k\|^2 \tag{4.3.4}$$

Again, in (4.2.4), by letting $u = x^*$, $v = z^{k+1}$ we get

$$2\mu(F(x^*) - F(x^{k+1})) \geq \|x^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2. \tag{4.3.5}$$

By letting $u = x^*$, $v = x^{k+1}$ in (4.2.3) we get

$$2\mu(F(x^*) - F(y^{k+1}) \geq \|y^{k+1} - x^*\|^2 - \|x^{k+1} - x^*\|^2. \tag{4.3.6}$$

Taking the summation of (4.3.5) and (4.3.6) we obtain,

$$-2\mu v_{k+1} \geq \|y^{k+1} - x^*\|^2 - \|z^{k+1} - x^*\|^2. \tag{4.3.7}$$

We then multiply (4.3.4) by t_k^2, (4.3.7) by t_{k+1}, and take the summation of the resulting two

inequalities. We can get the following result by using the fact $t_k^2 = t_{k+1}(t_{k+1}-1)$:

$$\begin{aligned} 2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1}) \geq \; & t_{k+1}^2 \|y^{k+1} - z^{k+1}\|^2 \\ & + 2t_{k+1}\langle y^{k+1} - z^{k+1}, t_{k+1}z^{k+1} - (t_{k+1}-1)\delta^k - x^*\rangle. \end{aligned} \tag{4.3.8}$$

Applying the Pythagoras relation

$$\|\mathbf{b}-\mathbf{a}\|^2 + 2\langle \mathbf{b}-\mathbf{a}, \mathbf{a}-\mathbf{c}\rangle = \|\mathbf{b}-\mathbf{c}\|^2 - \|\mathbf{a}-\mathbf{c}\|^2$$

to the right-hand side of the last inequality with

$$\mathbf{a} := t_{k+1}z^{k+1}, \mathbf{b} := t_{k+1}y^{k+1}, \mathbf{c} := (t_{k+1}-1)\delta^k - x^*,$$

we get

$$2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1}) \geq \|t_{k+1}y^{k+1} - (t_{k+1}-1)\delta^k - x^*\|^2 - \|t_{k+1}z^{k+1} - (t_{k+1}-1)\delta^k - x^*\|^2.$$

Therefore, with $t_{k+1}z^{k+1} := t_{k+1}\delta^k + t_k(y^k - \delta^{k-1}) - (\delta^k - \delta^{k-1})$ and $\mathbf{u}^k := t_k y^k - (t_k - 1)\delta^{k-1} - x^*$, (4.3.1) follows immediately. □

Lemma 4.3.2. *Let $\{a_k, b_k\}$ be positive sequences of reals satisfying*

$$a_k - a_{k+1} \geq b_{k+1} - b_k, \forall k \geq 1, \text{ with } a_1 + b_1 \leq c, c \geq 0.$$

Then $a_k \leq c$ for every $k \geq 1$.

Theorem 4.3.3. *Assume ∇f and ∇g are both Lipschitz continuous with Lipschitz constants $L(f)$ and $L(g)$, respectively. The sequences $\{x^k, y^k\}$ generated via Algorithm 4.4 with $\mu \leq 1/\max\{L(f), L(g)\}$ satisfy*

$$\min\{F(x^k), F(y^k)\} - F(x^*) \leq \frac{\|x^0 - x^*\|^2}{\mu(k+1)^2}. \tag{4.3.9}$$

Moreover if $1/(\beta \max\{L(f), L(g)\}) \leq \mu \leq 1/\max\{L(f), L(g)\}$ where $\beta \geq 1$, the number of iterations needed to get an ε-optimal solution is at most $\lceil \sqrt{C/\varepsilon} \rceil - 1$, where $C = \beta \max\{L(f), L(g)\}\|x^0 - x^*\|^2$.

Proof. Define

$$a_k := 2\mu t_k^2 v_k, b_k := \|\mathbf{u}^k\|^2, c := \|x^0 - x^*\|^2.$$

From Theorem 4.3.1 we know that for every $k \geq 1$,

$$a_k - a_{k+1} \geq b_{k+1} - b_k.$$

If $a_1 + b_1 \leq c$ holds, then according to Lemma 4.3.2, we obtain that,

$$2\mu t_k^2 v^k \leq \|x^0 - x^*\|^2,$$

which combined with $t_k \geq (k+1)/2$ yields

$$F(x^k) + F(y^k) - 2F(x^*) = v_k \leq \frac{2\|x^0 - x^*\|^2}{\mu(k+1)^2}.$$

The desired result follows immediately. Thus, all we need to do now is to prove $a_1 + b_1 \leq c$, where $a_1 = 2\mu v_1$, $b_1 = \|\mathbf{u}^1\|^2 = \|y^1 - x^*\|^2$. Note that (4.3.7) implies that

$$-2\mu v_1 \geq \|y^1 - x^*\|^2 - \|z^1 - x^*\|^2.$$

Then we have

$$a_1 + b_1 = 2\mu v_1 + \|y^1 - x^*\|^2 \leq \|z^1 - x^*\|^2 = \|x^0 - x^*\|^2,$$

and (4.3.9) follows immediately. □

4.4 Comparison of ALM, FALM, ISTA, FISTA, SADAL and SALSA

In this section we compare the performance of our basic and fast ALMs against ISTA, FISTA, SADAL (Algorithm 4.2) and an alternating direction augmented Lagrangian method SALSA described in [1] on a benchmark wavelet-based image deblurring problem from [37]. In this problem, the original image is the well-known Cameraman image of size 256×256 and the observed image is obtained after imposing a uniform blur of size 9×9 (denoted by the operator R) and Gaussian noise (generated by the function *randn* in MATLAB with a seed of 0 and a standard deviation of 0.56). Since the coefficient of the wavelet transform of the image is sparse in this problem, one can try to reconstruct the image u

from the observed image b by solving the problem:

$$\bar{x} := \arg\min_{x} \quad \frac{1}{2}\|Ax-b\|_2^2 + \rho\|x\|_1, \tag{4.4.1}$$

and setting $u := W\bar{x}$, where $A := RW$ and W is the inverse discrete Haar wavelet transform with four levels. By defining $f(x) := \frac{1}{2}\|Ax-b\|_2^2$ and $g(x) := \rho\|x\|_1$, it is clear that (4.4.1) can be expressed in the form of (4.1.1) and can be solved by ISTA, FISTA, SALSA and SADAL (Algorithm 4.2). However, in order to use ALM (Algorithm 4.3) and FALM (Algorithm 4.4), we need to smooth $g(x)$ first, since these two algorithms require both f and g to be smooth. Here we apply the smoothing technique introduced by Nesterov [77] since this technique guarantees that the gradient of the smoothed function is Lipschitz continuous. A smoothed approximation to the ℓ_1 function $g(x) := \rho\|x\|_1$ with smoothness parameter $\sigma > 0$ is

$$g_\sigma(x) := \max\{\langle x, z\rangle - \frac{\sigma}{2}\|z\|_2^2 : \|z\|_\infty \leq \rho\}. \tag{4.4.2}$$

It is easy to show that the optimal solution $z_\sigma(x)$ of (4.4.2) is

$$z_\sigma(x) = \min\{\rho, \max\{x/\sigma, -\rho\}\}. \tag{4.4.3}$$

According to Theorem 1 in [77], the gradient of g_σ is given by $\nabla g_\sigma(x) = z_\sigma(x)$ and is Lipschitz continuous with Lipschitz constant $L(g_\sigma) = 1/\sigma$. After smoothing g, we can

apply Algorithms 4.3 and 4.4 to solve the smoothed problem:

$$\min_x f(x) + g_\sigma(x). \tag{4.4.4}$$

We have the following theorem about the ε-optimal solutions of problems (4.4.1) and (4.4.4).

Theorem 4.4.1. *Let* $\sigma = \frac{\varepsilon}{n\rho^2}$ *and* $\varepsilon > 0$. *If* $x(\sigma)$ *is an* $\varepsilon/2$*-optimal solution to* (4.4.4)*, then* $x(\sigma)$ *is an* ε*-optimal solution to* (4.4.1).

Proof. Let $D_g := \max\{\frac{1}{2}\|z\|_2^2 : \|z\|_\infty \le \rho\} = \frac{1}{2}n\rho^2$ and x^* and $x^*(\sigma)$ be optimal solution to problems (4.4.1) and (4.4.4), respectively. Note that

$$g_\sigma(x) \le g(x) \le g_\sigma(x) + \sigma D_g, \forall x \in \mathbb{R}^n. \tag{4.4.5}$$

Using the inequalities in (4.4.5) and the facts that $x(\sigma)$ is an $\varepsilon/2$-optimal solution to (4.4.4) and $\sigma D_g = \frac{\varepsilon}{2}$, we have

$$\begin{aligned} f(x(\sigma)) + g(x(\sigma)) - f(x^*) - g(x^*) &\le f(x(\sigma)) + g_\sigma(x(\sigma)) + \sigma D_g - f(x^*) - g_\sigma(x^*) \\ &\le f(x(\sigma)) + g_\sigma(x(\sigma)) + \sigma D_g - f(x^*(\sigma)) - g_\sigma(x^*(\sigma)) \\ &\le \varepsilon/2 + \sigma D_g = \varepsilon. \end{aligned}$$

Thus, to find an ε-optimal solution to (4.4.1), we can apply Algorithms 4.3 and 4.4 to find an $\varepsilon/2$-optimal solution to (4.4.4) with $\sigma = \frac{\varepsilon}{n\rho^2}$. The iteration complexity results in

Theorems 4.2.3 and 4.3.3 hold since the gradient of g_σ is Lipschitz continuous. However, the numbers of iterations needed by ALM and FALM to obtain an ε-optimal solution to (4.4.1) become $O(1/\varepsilon^2)$ and $O(1/\varepsilon)$, respectively, due to the fact that the Lipschitz constant $L(g_\sigma) = 1/\sigma = \frac{n\rho^2}{\varepsilon} = O(1/\varepsilon)$.

When Algorithms 4.3 and 4.4 are applied to solve (4.4.4), (4.1.3) with g replaced by g_σ is also easy to solve; its optimal solution is

$$x := z - \tau \min\{\rho, \max\{-\rho, \frac{z}{\tau+\sigma}\}\}.$$

Since ALM is equivalent to SADAL when both functions are smooth, we implemented ALM as SADAL when we solved (4.4.4). We also applied SADAL to the nonsmooth problem (4.4.1). In all algorithms, we set the initial points $x^0 = y^0 = \mathbf{0}$, and in FALM we set $z^1 = \mathbf{0}$. MATLAB codes for SALSA, FISTA and ISTA (modified from FISTA) were downloaded from http://cascais.lx.it.pt/∼mafonso/salsa.html and their default inputs were used. Moreover, λ^0 was set to $\mathbf{0}$ in Algorithm 4.2. Also, whenever $g(x)$ was smoothed, we set $\sigma = 10^{-6}$. μ was set to 1 in all the algorithms since the Lipschitz constant of the gradient of function $\frac{1}{2}\|RW(\cdot) - b\|_2^2$ was known to be 1. We set μ to 1 even for the smoothed problems. Although this violates the requirement $\mu \leq \frac{1}{L(g_\sigma)}$ in Theorems 4.2.3 and 4.3.3, we see from our numerical results reported below that ALM and FALM still work very well. All of the algorithms tested were terminated after 1000 iterations. The (nonsmoothed) objective function values in (4.4.1) produced by these algorithms at iterations: 10, 50, 100, 200, 500, 800 and 1000 for different choices of ρ are presented in Tables 4.1 and 4.2.

The CPU times (in seconds) and the number of iterations required to reduce the objective function value to below $1.04e+5$ and $8.60e+5$ are reported respectively in the last columns of Tables 4.1 and 4.2.

All of our codes were written in MATLAB and run in MATLAB 7.3.0 on a Dell Precision 670 workstation with an Intel Xeon(TM) 3.4GHZ CPU and 6GB of RAM.

Table 4.1: Comparison of the algorithms for solving (4.4.1) with $\rho = 0.01$

solver	obj in k-th iteration							cpu (iter)
	10	50	100	200	500	800	1000	
FALM	1.767249e+5	1.040955e+5	9.899843e+4	9.516208e+4	9.186355e+4	9.073086e+4	9.028790e+4	23.1 (51)
FISTA	1.723109e+5	1.061116e+5	1.016385e+5	9.752858e+4	9.372093e+4	9.233719e+4	9.178455e+4	26.0 (69)
ALM	4.585705e+5	1.481379e+5	1.233182e+5	1.116683e+5	1.047410e+5	1.025611e+5	1.016589e+5	208.1 (581)
ISTA	2.345290e+5	1.267048e+5	1.137827e+5	1.079721e+5	1.040666e+5	1.025107e+5	1.018068e+5	196.8 (510)
SALSA	8.772957e+5	1.549462e+5	1.267379e+5	1.132676e+5	1.054600e+5	1.031346e+5	1.021898e+5	223.9 (663)
SADAL	2.524912e+5	1.271591e+5	1.133542e+5	1.068386e+5	1.021905e+5	1.004005e+5	9.961905e+4	113.5 (332)

Table 4.2: Comparison of the algorithms for solving (4.4.1) with $\rho = 0.1$

solver	obj in k-th iteration							cpu (iter)
	10	50	100	200	500	800	1000	
FALM	9.876315e+5	8.629257e+5	8.369244e+5	8.210375e+5	8.097621e+5	8.067903e+5	8.058290e+5	25.7 (54)
FISTA	9.924884e+5	8.830263e+5	8.501727e+5	8.288459e+5	8.126598e+5	8.081259e+5	8.066060e+5	30.1 (79)
ALM	1.263787e+6	9.521381e+5	9.172737e+5	8.910902e+5	8.639917e+5	8.528932e+5	8.482666e+5	211.8 (588)
ISTA	1.048956e+6	9.396822e+5	9.161787e+5	8.951970e+5	8.700864e+5	8.589587e+5	8.541664e+5	293.8 (764)
SALSA	1.680608e+6	9.601661e+5	9.230268e+5	8.956607e+5	8.674579e+5	8.558580e+5	8.509770e+5	230.2 (671)
SADAL	1.060130e+6	9.231803e+5	8.956150e+5	8.735746e+5	8.509601e+5	8.420295e+5	8.383270e+5	112.5 (335)

From Tables 4.1 and 4.2 we see that in terms of the value of the objective function achieved after a specified number of iterations, the performance FALM is always slightly better than that of FISTA and is much better than the performance of the other algorithms. Since, on the two test problems, FALM is always better than ALM, and FISTA is always better than ISTA, we can conclude that the Nesterov-type acceleration technique greatly speeds up the basic algorithms on these problems. Moreover, although sometimes in the early iterations FISTA (ISTA) is better than FALM (ALM), it is always worse than the latter

two algorithms when the iteration number is large. We also illustrate our comparisons graphically by plotting in Figure 4.1 the objective function value versus the number of iterations taken by these algorithms for solving (4.4.1) with $\rho = 0.1$. From Figure 4.1 we see clearly that for this problem, ALM outperforms ISTA, FALM outperforms FISTA, FALM outperforms ALM.

From the CPU times and the iteration numbers in the last columns of Tables 4.1 and 4.2 we see that, the fast versions are always much better than the basic versions of the algorithms. Since iterations of FISTA cost less than those of FALM, we see that although FISTA takes 35% (4%) more iterations than FALM in the last column of Table 4.1 (4.2) it takes only 7% more time (20% less time).

4.5 Applications

In this section, we describe how ALM and FALM can be applied to problems that can be formulated as RPCA problems to illustrate the use of Nesterov-type smoothing when the functions f and g do not satisfy the smoothness conditions required by the theorems in Sections 4.2 and 4.3. Our numerical results show that our methods are able to solve huge problems that arise in practice; e.g., one problem involving roughly 40 million variables and 20 million linear constraints is solved in about three-quarters of an hour. We alse describe application of our methods to the SICS problem.

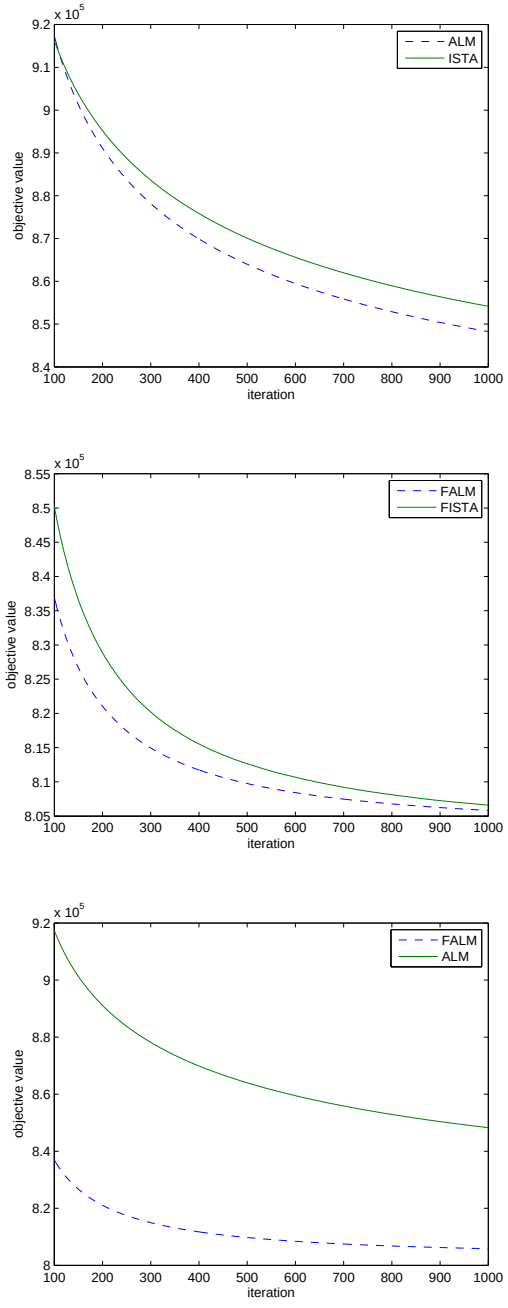

Figure 4.1: Comparison of the algorithms

4.5.1 Applications in Robust Principal Component Analysis

We restate the robust PCA problem (1.2.12) here for convenience:

$$\min_{X,Y\in\mathbb{R}^{m\times n}} \quad \|X\|_* + \rho\|Y\|_1 \quad \text{s.t.} \quad X+Y=M. \tag{4.5.1}$$

In order to apply Algorithms 4.3 and 4.4 to (4.5.1), we need to smooth both the nuclear norm $f(X) := \|X\|_*$ and the ℓ_1 norm $g(Y) := \rho\|Y\|_1$. We again apply Nesterov's smoothing technique as in Section 4.4. $g(Y)$ can be smoothed in the same way as the vector ℓ_1 norm in Section 4.4. We use $g_\sigma(Y)$ to denote the smoothed function with smoothness parameter $\sigma > 0$. A smoothed approximation to $f(X)$ with smoothness parameter $\sigma > 0$ is

$$f_\sigma(X) := \max\{\langle X, W\rangle - \frac{\sigma}{2}\|W\|_F^2 : \|W\| \le 1\}. \tag{4.5.2}$$

It is easy to show that the optimal solution $W_\sigma(X)$ of (4.5.2) is

$$W_\sigma(X) = U\mathrm{Diag}(\min\{\gamma, 1\})V^\top, \tag{4.5.3}$$

where $U\mathrm{Diag}(\gamma)V^\top$ is the singular value decomposition (SVD) of X/σ. According to Theorem 1 in [77], the gradient of f_σ is given by $\nabla f_\sigma(X) = W_\sigma(X)$ and is Lipschitz continuous with Lipschitz constant $L(f_\sigma) = 1/\sigma$. After smoothing f and g, we can apply Algorithms 4.3 and 4.4 to solve the following smoothed problem:

$$\min\{f_\sigma(X) + g_\sigma(Y) : X+Y=M\}. \tag{4.5.4}$$

We have the following theorem about ε-optimal solutions of problems (4.5.1) and (4.5.4).

Theorem 4.5.1. *Let* $\sigma = \frac{\varepsilon}{2\max\{\min\{m,n\},mn\rho^2\}}$ *and* $\varepsilon > 0$. *If* $(X(\sigma),Y(\sigma))$ *is an* $\varepsilon/2$*-optimal solution to* (4.5.4)*, then* $(X(\sigma),Y(\sigma))$ *is an* ε*-optimal solution to* (4.5.1).

Proof. Let $D_f := \max\{\frac{1}{2}\|W\|_F^2 : \|W\| \leq 1\} = \frac{1}{2}\min\{m,n\}$, $D_g := \max\{\frac{1}{2}\|Z\|_F^2 : \|Z\|_\infty \leq \rho\} = \frac{1}{2}mn\rho^2$ and (X^*,Y^*) and $(X^*(\sigma),Y^*(\sigma))$ be optimal solution to problems (4.5.1) and (4.5.4), respectively. Note that

$$f_\sigma(X) \leq f(X) \leq f_\sigma(X) + \sigma D_f, \forall X \in \mathbb{R}^{m\times n} \tag{4.5.5}$$

and

$$g_\sigma(Y) \leq g(Y) \leq g_\sigma(Y) + \sigma D_g, \forall Y \in \mathbb{R}^{m\times n}. \tag{4.5.6}$$

Using the inequalities in (4.5.5) and (4.5.6) and the facts that $(X(\sigma),Y(\sigma))$ is an $\varepsilon/2$-optimal solution to (4.5.4) and $\sigma\max\{D_f,D_g\} = \frac{\varepsilon}{4}$, we have

$$\begin{aligned}
&f(X(\sigma)) + g(Y(\sigma)) - f(X^*) - g(Y^*) \\
\leq& f_\sigma(X(\sigma)) + g_\sigma(Y(\sigma)) + \sigma D_f + \sigma D_g - f_\sigma(X^*) - g_\sigma(Y^*) \\
\leq& f_\sigma(X(\sigma)) + g_\sigma(Y(\sigma)) + \sigma D_f + \sigma D_g - f_\sigma(X^*(\sigma)) - g_\sigma(Y^*(\sigma)) \\
\leq& \varepsilon/2 + \sigma D_f + \sigma D_g \leq \varepsilon/2 + \varepsilon/4 + \varepsilon/4 = \varepsilon.
\end{aligned}$$

□

Thus, according to Theorem 4.5.1, to find an ε-optimal solution to (4.5.1), we need to find an $\varepsilon/2$-optimal solution to (4.5.4) with $\sigma = \frac{\varepsilon}{2\max\{\min\{m,n\},mn\rho^2\}}$. We can apply Algorithms 4.3 and 4.4 to solve (4.5.4). However, the numbers of iterations needed by ALM and FALM to obtain an ε-optimal solution to (4.5.1) become $O(1/\varepsilon^2)$ and $O(1/\varepsilon)$, respectively, due to the fact that the Lipschitz constant $L(f_\sigma) = 1/\sigma = \frac{2\max\{\min\{m,n\},mn\rho^2\}}{\varepsilon} = O(1/\varepsilon)$.

The two subproblems at iteration k of Algorithm 4.3 when applied to (4.5.4) reduce to

$$\begin{aligned} X^{k+1} := \arg\min_X \quad & f_\sigma(X) + g_\sigma(Y^k) + \langle \nabla g_\sigma(Y^k), M - X - Y^k \rangle \\ & + \tfrac{1}{2\mu}\|X + Y^k - M\|_F^2, \end{aligned} \tag{4.5.7}$$

and

$$\begin{aligned} Y^{k+1} := \arg\min_Y \quad & f_\sigma(X^{k+1}) + \langle \nabla f_\sigma(X^{k+1}), M - X^{k+1} - Y \rangle \\ & + \tfrac{1}{2\mu}\|X^{k+1} + Y - M\|_F^2 + g_\sigma(Y). \end{aligned} \tag{4.5.8}$$

The first-order optimality conditions for (4.5.7) are:

$$W_\sigma(X) - Z_\sigma(Y^k) + \frac{1}{\mu}(X + Y^k - M) = 0, \tag{4.5.9}$$

where $W_\sigma(X)$ and $Z_\sigma(Y)$ are defined in (4.5.3) and (4.4.3). It is easy to check that

$$X := U\mathrm{Diag}(\gamma - \frac{\mu\gamma}{\max\{\gamma, \mu+\sigma\}})V^\top \tag{4.5.10}$$

satisfies (4.5.9), where $U\mathrm{Diag}(\gamma)V^\top$ is the SVD of the matrix $\mu Z_\sigma(Y^k) - Y^k + M$. Thus,

solving the subproblem (4.5.7) corresponds to an SVD. If we define $B := \mu W_\sigma(X^{k+1}) - X^{k+1} + M$, it is easy to verify that

$$Y_{ij} = B_{ij} - \mu \min\{\rho, \max\{-\rho, \frac{B_{ij}}{\sigma+\mu}\}\} \text{ for } i = 1,\ldots,m \text{ and } j = 1,\ldots,n \qquad (4.5.11)$$

satisfies the first-order optimality conditions for (4.5.8): $-W_\sigma(X^{k+1}) + \frac{1}{\mu}(X^{k+1} + Y - M) + Z_\sigma(Y) = 0$. Thus, solving the subproblem (4.5.8) can be done very cheaply. The two subproblems at the k-th iteration of Algorithm 4.4 can be done in the same way and the main computational effort in each iteration of both ALM and FALM corresponds to an SVD.

4.5.2 RPCA with Missing Data

In some applications of RPCA, some of the entries of M in (4.5.1) may be missing (e.g., in low-rank matrix completion problems where the matrix is corrupted by noise). Let Ω be the index set of the entries of M that are observable and define the projection operator $\mathcal{P}_\Omega$ as: $(\mathcal{P}_\Omega(X))_{ij} = X_{ij}$, if $(i,j) \in \Omega$ and $(\mathcal{P}_\Omega(X))_{ij} = 0$ otherwise. It has been shown under some randomness hypotheses that the low rank $\bar{X}$ and sparse $\bar{Y}$ can be recovered with high probability by solving (see Theorem 1.2 in [15]),

$$(\bar{X}, \bar{Y}) := \arg\min_{X,Y}\{\|X\|_* + \rho\|Y\|_1 : \mathcal{P}_\Omega(X+Y) = \mathcal{P}_\Omega(M)\}. \qquad (4.5.12)$$

To solve (4.5.12) by ALM or FALM, we need to transform it into the form of (4.5.1). For this we have

Theorem 4.5.2. $(\bar{X}, \mathcal{P}_\Omega(\bar{Y}))$ *is an optimal solution to* (4.5.12) *if*

$$(\bar{X}, \bar{Y}) = \arg\min_{X,Y}\{\|X\|_* + \rho\|\mathcal{P}_\Omega(Y)\|_1 : X + Y = \mathcal{P}_\Omega(M)\}. \tag{4.5.13}$$

Proof. Suppose (X^*, Y^*) is an optimal solution to (4.5.12). We claim that $Y^*_{ij} = 0, \forall (i,j) \notin \Omega$. Otherwise, $(X^*, \mathcal{P}_\Omega(Y^*))$ is feasible to (4.5.12) and has a strictly smaller objective function value than (X^*, Y^*), which contradicts the optimality of (X^*, Y^*). Thus, $\|\mathcal{P}_\Omega(Y^*)\|_1 = \|Y^*\|_1$. Now suppose that $(\bar{X}, \mathcal{P}_\Omega(\bar{Y}))$ is not optimal to (4.5.12); then we have

$$\|X^*\|_* + \rho\|\mathcal{P}_\Omega(Y^*)\|_1 = \|X^*\|_* + \rho\|Y^*\|_1 < \|\bar{X}\|_* + \rho\|\mathcal{P}_\Omega(\bar{Y})\|_1. \tag{4.5.14}$$

By defining a new matrix $\tilde{Y}$ as

$$\tilde{Y}_{ij} = \begin{cases} Y^*_{ij}, & (i,j) \in \Omega \\ -X^*_{ij}, & (i,j) \notin \Omega, \end{cases}$$

we have that $(X^*, \tilde{Y})$ is feasible to (4.5.13) and $\|\mathcal{P}_\Omega(\tilde{Y})\|_1 = \|\mathcal{P}_\Omega(Y^*)\|_1$. Combining this with (4.5.14), we obtain

$$\|X^*\|_* + \rho\|\mathcal{P}_\Omega(\tilde{Y})\|_1 < \|\bar{X}\|_* + \rho\|\mathcal{P}_\Omega(\bar{Y})\|_1,$$

which contradicts the optimality of $(\bar{X}, \bar{Y})$ to (4.5.13). Therefore, $(\bar{X}, \mathcal{P}_\Omega(\bar{Y}))$ is optimal to (4.5.12). □

The only differences between (4.5.1) and (4.5.13) lie in that the matrix M is replaced by $\mathcal{P}_\Omega(M)$ and $g(Y) = \rho\|Y\|_1$ is replaced by $\rho\|\mathcal{P}_\Omega(Y)\|_1$. A smoothed approximation $g_\sigma(Y)$ to $g(Y) := \rho\|\mathcal{P}_\Omega(Y)\|_1$ is given by

$$g_\sigma(Y) := \max\{\langle \mathcal{P}_\Omega(Y), Z\rangle - \frac{\sigma}{2}\|Z\|_F^2 : \|Z\|_\infty \le \rho\}, \tag{4.5.15}$$

and

$$(\nabla g_\sigma(Y))_{ij} = \min\{\rho, \max\{(\mathcal{P}_\Omega(Y))_{ij}/\sigma, -\rho\}\}, \text{ for } 1 \le i \le m \text{ and } 1 \le j \le n. \tag{4.5.16}$$

According to Theorem 1 in [77], $\nabla g_\sigma(Y)$ is Lipschitz continuous with $L_\sigma(g) = 1/\sigma$. Thus the convergence and iteration complexity results in Theorems 4.2.3 and 4.3.3 apply. The only changes in Algorithms 4.3 and 4.4 are: replacing M by $\mathcal{P}_\Omega(M)$ and computing Y^{k+1} using (4.5.11) with B is replaced by $\mathcal{P}_\Omega(B)$.

4.5.3 Numerical Results on RPCA Problems

In this section, we report numerical results obtained using the ALM method to solve RPCA problems with both complete and incomplete data matrices M. We compare the performance of ALM with the exact ADM (EADM) and the inexact ADM (IADM) methods in [64]. The MATLAB codes of EADM and IADM were downloaded from $http://watt.csl.illinois.edu/\sim perceive/matrix-rank/sample\_code.html$ and their default settings were used. To further accelerate ALM, we adopted the continuation strategy used in

EADM and IADM. Specifically, we set $\mu_{k+1} := \max\{\bar{\mu}, \eta\mu_k\}$, where $\mu_0 = \|M\|/1.25, \bar{\mu} = 10^{-6}$ and $\eta = 2/3$ in our numerical experiments. Although in some iterations this violates the requirement $\mu \leq \min\{\frac{1}{L(f_\sigma)}, \frac{1}{L(g_\sigma)}\}$ in Theorems 4.2.3 and 4.3.3, we see from our numerical results reported below that ALM and FALM still work very well. We also found that by adopting this updating rule for μ, there was not much difference between the performance of ALM and that of FALM. So we only compare ALM with EADM and IADM. As in Section 4.4, since we applied ALM to a smoothed problem, we implemented ALM as SADAL. The initial point in ALM was set to $(X^0, Y^0) = (M, \mathbf{0})$ and the initial Lagrange multiplier was set to $\Lambda^0 = -\nabla g_\sigma(Y^0)$. We set the smoothness parameter $\sigma = 10^{-6}$. Solving subproblem (4.5.7) requires computing an SVD (see (4.5.10)). However, we do not have to compute the whole SVD, as only the singular values that are larger than the threshold $\tau = \frac{\mu\gamma}{\max\{\gamma, \mu+\sigma\}}$ and the corresponding singular vectors are needed. We therefore use PROPACK [59], which is also used in EADM and IADM, to compute these singular values and corresponding singular vectors. To use PROPACK, one has to specify the number of leading singular values (denoted by sv_k) to be computed at iteration k. We here adopt the strategy suggested in [64] for EADM and IADM. This strategy starts with $sv_0 = 100$ and updates sv_k via:

$$sv_{k+1} = \begin{cases} svp_k + 1, & \text{if } svp_k < sv_k \\ \min\{svp_k + round(0.05d), d\}, & \text{if } svp_k = sv_k, \end{cases}$$

where $d = \min\{m, n\}$ and svp_k is the number of singular values that are larger than the threshold τ.

In all our experiments ρ was chosen equal to $1/\sqrt{m}$. We stopped ALM, EADM and IADM when the relative infeasibility was less than 10^{-7}, i.e., $\|X+Y-M\|_F < 10^{-7}\|M\|_F$.

Background Extraction from Surveillance Video

Extracting the almost still background from a sequence frames of video is a basic task in video surveillance. This problem is difficult due to the presence of moving objects in the foreground in the video. Interestingly, as shown in [15], this problem can be formulated as a RPCA problem (4.5.1). By stacking the columns of each frame into a long vector, we get a matrix M whose columns correspond to the sequence of frames of the video. This matrix M can be decomposed into the sum of two matrices $M := \bar{X} + \bar{Y}$. The matrix $\bar{X}$, which represents the background in the frames, should be of low rank due to the correlation between frames. The matrix $\bar{Y}$, which represents the moving objects in the foreground in the frames, should be sparse since these objects usually occupy a small portion of each frame. We apply ALM to solve (4.5.1) for two videos introduced in [63].

Our first example is a sequence of 200 grayscale frames of size 144×176 from a video of a hall at an airport. Thus the matrix M is in $\mathbb{R}^{25344\times 200}$. The second example is a sequence of 320 color frames from a video taken at a campus. Since the video is colored, each frame is an image stored in the RGB format, which is a $128 \times 160 \times 3$ cube. The video is then reshaped into a 128×160 by 3×320 matrix, i.e., $M \in \mathbb{R}^{20480\times 960}$. Some frames of the videos and the recovered backgrounds and foregrounds are shown in Figure 4.5.3. We only show the frames produced by ALM, because EADM and IADM produce visually identical results. From these figures we can see that ALM can effectively separate the nearly still

Figure 4.2: In the first 3 columns: (a) Video sequence. (b) Static background recovered by our ALM. Note that the man who kept still in the 200 frames stays as in the background. (c) Moving foreground recovered by our ALM. In the last 3 columns: (a) Video sequence. (b) Static background recovered by our ALM. (c) Moving foreground recovered by our ALM.

background from the moving foreground. Table 4.3 summarizes the numerical results on these problems. The CPU times are reported in the form of $hh:mm:ss$. From Table 4.3 we see that although ALM is slightly worse than IADM, it is much faster than EADM in terms of both the number of SVDs and CPU times. We note that the numerical results in [15] show that the model (4.5.1) produces much better results than other competing models for background extraction in surveillance video.

Table 4.3: Comparison of ALM and EADM on surveillance video problems

			Exact ADM		Inexact ADM		ALM	
Problem	m	n	SVDs	CPU	SVDs	CPU	SVDs	CPU
Hall (gray)	25344	200	550	40:15	38	03:47	43	04:03
Campus (color)	20480	960	651	13:54:38	40	43:35	46	46:49

Random Matrix Completion Problems with Grossly Corrupted Data

For the matrix completion problem (4.5.12), we set $M := A + E$, where the rank r matrix $A \in \mathbb{R}^{n\times n}$ was created as the product $A_L A_R^\top$, of random matrices $A_L \in \mathbb{R}^{n\times r}$ and $A_R \in \mathbb{R}^{n\times r}$ with i.i.d. Gaussian entries $\mathcal{N}(0,1)$ and the sparse matrix E was generated by choosing its support uniformly at random and its nonzero entries uniformly i.i.d. in the interval $[-500, 500]$. In Table 4.4, $rr := \text{rank}(A)/n$, $spr := \|E\|_0/n^2$, the relative errors $relX := \|X - A\|_F/\|A\|_F$ and $relY := \|Y - E\|_F/\|E\|_F$, and the sampling ratio of Ω, $SR = m/n^2$. The m indices in Ω were generated uniformly at random. We set $\rho = 1/\sqrt{n}$ and stopped ALM when the relative infeasibility $\|X + Y - \mathcal{P}_\Omega(M)\|_F/\|\mathcal{P}_\Omega(M)\|_F < 10^{-5}$ and for our continuation strategy, we set $\mu_0 = \|\mathcal{P}_\Omega(M)\|_F/1.25$. The test results obtained using ALM to solve (4.5.13) with the nonsmooth functions replaced by their smoothed approximations are given in Table 4.4. From Table 4.4 we see that ALM recovered the test matrices from a limited number of observations. Note that a fairly high number of samples was needed to obtain small relative errors due to the presence of noise. The number of iterations needed was almost constant (around 36), no matter the size of the problems. The CPU times (in seconds) needed are also reported.

4.6 Conclusion

In this chapter, we proposed both basic and accelerated versions of alternating linearization methods for minimizing the sum of two convex functions. Our basic methods require at most $O(1/\varepsilon)$ iterations to obtain an ε-optimal solution, while our accelerated methods

Table 4.4: Numerical results for noisy matrix completion problems

rr	*spr*	iter	relX	relY	cpu	iter	relX	relY	cpu
		$SR = 90\%, n = 500$				$SR = 80\%, n = 500$			
0.05	0.05	36	4.60e-5	4.25e-6	137	36	3.24e-5	4.31e-6	153
0.05	0.1	36	4.68e-5	5.29e-6	156	36	4.40e-5	4.91e-6	161
0.1	0.05	36	4.04e-5	3.74e-6	128	36	1.28e-3	1.33e-4	129
0.1	0.1	36	6.00e-4	4.50e-5	129	35	1.06e-2	7.59e-4	124
		$SR = 90\%, n = 1000$				$SR = 80\%, n = 1000$			
0.05	0.05	37	3.10e-5	3.96e-6	1089	37	2.27e-5	4.14e-6	1191
0.05	0.1	37	3.20e-5	4.93e-6	1213	37	3.00e-5	4.66e-6	1271
0.1	0.05	37	2.68e-5	3.34e-6	982	37	1.75e-4	2.49e-5	994
0.1	0.1	37	3.64e-5	4.51e-6	1004	36	4.62e-3	4.63e-4	965

require at most $O(1/\sqrt{\varepsilon})$ iterations with only a small additional amount of computational effort at each iteration. Numerical results on image deblurring, background extraction from surveillance video and matrix completion with grossly corrupted data are reported. These results demonstrate the efficiency and the practical potential of our algorithms.

Chapter 5

Fast Multiple Splitting Algorithms for Convex Optimization

5.1 Introduction

Many convex optimization problems that arise in practice take the form of a sum of convex functions. Often one function is an energy that one wants to minimize and the other functions are regularization terms to make the solution have certain properties. For example, Tikhonov regularization [91] is usually applied to ill-conditioned inverse problems to make them well-posed, compressed sensing [18, 26] uses ℓ_1 regularization to obtain sparse solutions, and problems arising from medical imaging adopt both ℓ_1 and total variation (TV) as regularization terms [69]. In this chapter, we propose and analyze splitting/alternating

direction algorithms for solving the following convex optimization problem:

$$\min_{x\in\mathbb{R}^n} \quad F(x) \equiv \sum_{i=1}^{K} f_i(x), \tag{5.1.1}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}, i = 1,\ldots,K$, are convex functions. When the functions f_i's are well-structured, a well established way to solve problem (5.1.1) is to split the variable x into K variables by introducing $K-1$ new variables and then apply an augmented Lagrangian method to solve the resulting problem. Decomposition of the augmented Lagrangian function can then be accomplished by applying an alternating direction method (ADM) to minimize it.

Problem (5.1.1) is closely related to the following inclusion problem:

$$0 \in T_1(x) + \cdots + T_K(x), \tag{5.1.2}$$

where $T_1,\ldots,T_K$ are set-valued maximal monotone operators. The goal of problem (5.1.2) is to find a zero of the sum of K maximal monotone operators. Note that the optimality conditions for (5.1.1) are

$$0 \in \sum_{i=1}^{K} \partial f_i(x);$$

hence, these conditions can be satisfied by solving a problem of the form (5.1.2).

In the extensive literature on splitting and ADM algorithms, the case $K = 2$ predominates. The algorithms for solving (5.1.2) when $K = 2$ are usually based on operator splitting techniques. Important operator splitting algorithms include the Douglas-Rachford

[22, 29, 32], Peaceman-Rachford [80], double-backward [21] and forward-backward class [40, 98] of algorithms. Alternating direction methods (ADM) within an augmented Lagrangian framework for solving (5.1.1) are optimization analogs/variants of the Douglas-Rachford and Peaceman-Rachford splitting methods. These algorithms have been studied extensively for the case of $K = 2$, and were first proposed in the 1970s for solving optimization problems arising from numerical PDE problems [41,43]. We refer to [47] and the references therein for more information on splitting and ADM algorithms for the case of $K = 2$.

Although there is an extensive literature on operator splitting methods, very few convergence results have been published on methods for finding a zero of a sum of more than two maximal monotone operators. The principal exceptions, are the Jacobi-like method of Spingarn [86] and more recently, the general projective splitting methods of Eckstein and Svaiter [34]. The algorithm addressed in [86] first reduces problem (5.1.2) to the sum of two maximal monotone operators by defining new subspaces and operators, and then applies a Douglas-Rachford splitting algorithm to solve the new problem. The projective splitting methods in [34] do not reduce problem (5.1.2) to the case $K = 2$. Instead, by using the concept of an extended solution set, it is shown in [34] that solving (5.1.2) is equivalent to finding a point in the extended solution set, and a separator-projection algorithm is given to do this.

Global convergence results for variable splitting ADMs and operator splitting algorithms for the case of $K = 2$ have been proved under various assumptions. However, except for the fairly recently proposed gradient methods in [78] and related iterative shrink-

age/thresholding algorithms in [4] and the alternating linearization methods in [47], complexity bounds for these methods had not been established. These complexity results are extensions of the seminal results of Nesterov [75, 76], who first showed that certain first-order methods that he proposed could obtain an ε-optimal solution of a smooth convex programming problem in $O(1/\sqrt{\varepsilon})$ iterations. Moreover, he showed that his methods were optimal in the sense that this iteration complexity was the best that could be obtained using only first-order information. Nesterov's optimal gradient methods are accelerated gradient methods that use a combination of previous points to compute the new point at each iteration. By combining these methods with smoothing techniques, optimal complexity results were obtained for solving nonsmooth problems in [77, 99].

In this chapter, we propose two classes of multiple variable-splitting algorithms based on alternating direction and alternating linearization techniques that can solve problem (5.1.1) for general $K(K \geq 2)$ and we present complexity results for them. (Note that the complexity results in [4, 47, 78] and Chapter 4 are only for problem (5.1.1) when $K = 2$). The algorithms in the first class can be viewed as alternating linearization methods in the sense that at each iteration these algorithms perform K minimizations of an approximation to the original objective function F by keeping one of the functions $f_i(x)$ unchanged and linearizing the other $K-1$ functions. An alternating linearization method for minimizing the sum of two convex functions was studied by Kiwiel et al. [58]. However, our algorithms differ greatly from the one in [58] in the way that the proximal terms are chosen. Moreover, our algorithms are more general as they can solve general problems with $K(K \geq 2)$ functions. Furthermore, we prove that the iteration complexity of this class of

splitting algorithms is $O(1/\varepsilon)$ for an ε-optimal solution. To the best of our knowledge, this is the first complexity result of this type for splitting/alternating direction type algorithms. The algorithms in our second class are accelerated versions of the algorithms in our first class and have $O(1/\sqrt{\varepsilon})$ iteration complexities. This class of splitting algorithms is also new as are the complexity results.

Our new algorithms have, in addition, several practical advantages. First, they are all parallelizable. Thus, although at each iteration we solve K subproblems, the CPU time required should be approximately equal to the time required to solve the most difficult of the subproblems if we have K processors that can work in parallel. Second, since every function f_i is minimized once at each iteration, our algorithms may need fewer iterations to converge than operator splitting algorithms such as FPC [50, 68],TVCMRI [69], ISTA and FISTA [4]. The numerical results in [1] for the case of $K = 2$ support this conclusion.

The rest of this chapter is organized as follows. In Section 5.2 we propose a class of splitting algorithms based on alternating direction and alternating linearization methods for solving (5.1.1) and prove that they require $O(1/\varepsilon)$ iterations to obtain an ε-optimal solution. In Section 5.3 we propose accelerated splitting algorithms for solving (5.1.1) and prove they have $O(1/\sqrt{\varepsilon})$ complexities. Numerical results are presented in Section 5.4. Finally, we summarize our results in Section 5.5.

5.2 A class of multiple-splitting algorithms

By introducing new variables, i.e., splitting variable x into K different variables, problem (5.1.1) can be rewritten as:

$$\begin{aligned} \min \quad & \sum_{i=1}^{K} f_i(x^i) \\ s.t. \quad & x^i = x^{i+1}, i = 1, \ldots, K-1. \end{aligned} \tag{5.2.1}$$

In Sections 2 and 3, we focus on splitting and ADM algorithms for solving (5.2.1) and their complexity results.

We make the following assumptions throughout Sections 5.2 and 5.3.

Assumption 5.2.1.

- *$f_i(\cdot) : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, K$ is a smooth convex function of the type $C^{1,1}$, i.e. continuously differentiable with Lipschitz continuous gradient:*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L(f_i)\|x - y\|, \forall x, y \in \mathbb{R}^n,$$

  where $L(f_i)$ is the Lipschitz constant.

- *Problem* (5.1.1) *is solvable, i.e., $X_* := \arg\min F \neq \emptyset$.*

The following notation is adopted throughout Sections 5.2 and 5.3.

Definition 5.2.2. *We define $\tilde{f}_i(u, v)$ as the linear approximation to $f_i(u)$ at a point v plus a*

proximal term:

$$\tilde{f}_i(u,v) := f_i(v) + \langle \nabla f_i(v), u-v \rangle + \frac{1}{2\mu}\|u-v\|^2,$$

where μ is a penalty parameter. We use $Q_i(v^1,\ldots,v^{i-1},u,v^{i+1},\ldots,v^K)$ to denote the following approximation to the function $F(u)$:

$$Q_i(v^1,\ldots,v^{i-1},u,v^{i+1},\ldots,v^K) := f_i(u) + \sum_{j=1,j\neq i}^{K} \tilde{f}_j(u,v^j),$$

i.e., Q_i is an approximation to the function F, where the i-th function f_i is unchanged but the other functions are approximated by a linear term plus a proximal term. We use $p_i(v^1,\ldots,v^{i-1},v^{i+1},\ldots,v^K)$ to denote the minimizer of $Q_i(v^1,\ldots,v^{i-1},u,v^{i+1},\ldots,v^K)$ with respect to u, i.e.,

$$p_i(v^1,\ldots,v^{i-1},v^{i+1},\ldots,v^K) := \arg\min_u Q_i(v^1,\ldots,v^{i-1},u,v^{i+1},\ldots,v^K). \tag{5.2.2}$$

With the above notation, we have the following lemma which follows from a fundamental property of a smooth function in the class $C^{1,1}$; see e.g., [6].

Lemma 5.2.3. *For $\tilde{f}_i$ defined as in Definition 5.2.2 and $\mu \leq 1/\max_{1\leq i\leq K} L(f_i)$, we have for $i = 1,\ldots,K$,*

$$f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x-y \rangle + \frac{L(f_i)}{2}\|x-y\|^2 \leq \tilde{f}_i(x,y), \forall x,y \in \mathbb{R}^n.$$

The following key lemma is crucial for the proofs of our complexity results. Our proofs of this lemma and most of the results that follow in this and the remaining sections of the chapter closely follow proofs given in [4] for related lemmas and theorems.

Lemma 5.2.4. *For any $i = 1,\dots,K$, $u, v^1,\dots,v^{i-1},v^{i+1},\dots,v^K \in \mathbb{R}^n$ and $\mu \le 1/\max_{1\le i\le K} L(f_i)$, we have,*

$$2\mu(F(u)-F(p)) \ge \sum_{j=1,j\neq i}^{K} \left(\|p-u\|^2 - \|v^j-u\|^2\right), \tag{5.2.3}$$

where $p := p_i(v^1,\dots,v^{i-1},v^{i+1},\dots,v^K)$.

Proof. From Lemma 5.2.3 we know that $F(p) \le Q_i(v^1,\dots,v^{i-1},p,v^{i+1},\dots,v^K)$ holds for all i and $v^1,\dots,v^{i-1},v^{i+1},\dots,v^K \in \mathbb{R}^n$. Thus, for any $u \in \mathbb{R}^n$ we have,

$$\begin{aligned}
F(u)-F(p) &\ge F(u) - Q_i(v^1,\dots,v^{i-1},p,v^{i+1},\dots,v^K) \qquad (5.2.4)\\
&= f_i(u) - f_i(p) + \sum_{j=1,j\neq i}^{K} \left(f_j(u) - f_j(v^j) - \left\langle \nabla f_j(v^j), p-v^j\right\rangle + \frac{1}{2\mu}\|p-v^j\|^2 \right)\\
&\ge \langle \nabla f_i(p), u-p\rangle + \sum_{j=1,j\neq i}^{K} \left(\left\langle \nabla f_j(v^j), u-v^j\right\rangle - \left\langle \nabla f_j(v^j), p-v^j\right\rangle + \frac{1}{2\mu}\|p-v^j\|^2 \right)\\
&= \langle \nabla f_i(p), u-p\rangle + \sum_{j=1,j\neq i}^{K} \left(\left\langle \nabla f_j(v^j), u-p\right\rangle + \frac{1}{2\mu}\|p-v^j\|^2 \right)\\
&= \sum_{j=1,j\neq i}^{K} \left(\left\langle -\frac{1}{\mu}(p-v^j), u-p\right\rangle - \frac{1}{2\mu}\|p-v^j\|^2 \right),
\end{aligned}$$

where the second inequality is due to the convexity of the functions $f_j, j = 1,\dots,K$ and the

last equality is from the first-order optimality conditions for problem (5.2.2), i.e.,

$$\nabla f_i(p) + \sum_{j=1, j\neq i}^{K} \left(\nabla f_j(v^j) + \frac{1}{\mu}(p - v^j) \right) = 0. \tag{5.2.5}$$

Then using the identity

$$\|a-c\|^2 - \|b-c\|^2 = \|a-b\|^2 + 2\langle a-b, b-c\rangle, \tag{5.2.6}$$

we get the following inequality:

$$\begin{aligned} 2\mu(F(u) - F(p)) &\geq \sum_{j=1, j\neq i}^{K} \left(\langle -2(p-v^j), u-p\rangle - \|p-v^j\|^2 \right) \\ &= \sum_{j=1, j\neq i}^{K} \left(\|p-u\|^2 - \|v^j-u\|^2 \right). \end{aligned}$$

□

Our multiple-splitting algorithms (MSA) for solving (5.2.1) are outlined in Algorithm 5.1, where $D^{(k)} \in \mathbb{R}^{K\times K}$ is a doubly stochastic matrix, i.e.,

$$D^{(k)}_{ij} \geq 0, \sum_{j=1}^{K} D^{(k)}_{ij} = 1, \sum_{i=1}^{K} D^{(k)}_{ij} = 1, \forall i, j = 1, \ldots, K.$$

One natural choice of $D^{(k)}$ is to take all of its components equal to $1/K$. In this case, all $w^i_{(k)}, i = 1, \ldots, K$ are equal to $\sum_{i=1}^{K} x^i_{(k)}/K$, i.e., the average of the current K iterates.

At iteration k, Algorithm 5.1 computes K points $x^i_{(k)}, i = 1, \ldots, K$ by solving K sub-

Algorithm 5.1 A Class of Multiple-Splitting Algorithms (MSA)

Set $x_0 = x^1_{(0)} = \ldots = x^K_{(0)} = w^1_{(0)} = \ldots = w^K_{(0)}$ and $\mu \leq 1/\max_{1\leq i\leq K} L(f_i)$.
for $k = 0, 1, \cdots$ **do**
  for $i = 1, \ldots, K$ **do**
    compute $x^i_{(k+1)} := p_i(w^i_{(k)}, \ldots, w^i_{(k)})$,
  end for
  compute

$$\left(w^1_{(k+1)}, \ldots, w^K_{(k+1)}\right) := \left(x^1_{(k+1)}, \ldots, x^K_{(k+1)}\right) D^{(k+1)}.$$

end for

problems. For many problems in practice, these K subproblems are expected to be very easy to solve. Another advantage of the algorithm is that it is parallelizable since given $w^i_{(k)}, i = 1, \ldots, K$, the K subproblems in Algorithm 5.1 can be solved simultaneously. Algorithm (5.1) can be viewed as an alternating linearization method since at each iteration, K subproblems are solved, and each subproblem corresponds to minimizing a function involving linear approximations to some of the functions. Although Algorithm 5.1 assumes the Lipschitz constants are known, and hence that μ is known, this assumption can be relaxed by using the backtracking technique in [4] to estimate μ at each iteration.

We prove in the following that the number of iterations needed by Algorithm 5.1 to obtain an ε-optimal solution is $O(1/\varepsilon)$.

Theorem 5.2.5. *Suppose x^* is an optimal solution to problem* (5.2.1)*. For any choice of $\mu \leq 1/\max_{1\leq i\leq K} L(f_i)$, the sequence $\{x^i_{(k)}, w^i_{(k)}\}_{i=1}^K$ generated by Algorithm 5.1 satisfies:*

$$\min_{i=1,\ldots,K} F(x^i_{(k)}) - F(x^*) \leq \frac{(K-1)\|x_0 - x^*\|^2}{2\mu k}. \tag{5.2.7}$$

Thus, the sequence $\{\min_{i=1,\ldots,K} F(x^i_{(k)})\}$ *produced by Algorithm 5.1 converges to* $F(x^*)$. *Moreover, if* $\mu \geq \beta/\max_i\{L(f_i)\}$ *where* $0 < \beta \leq 1$, *the number of iterations needed to obtain an* ε*-optimal solution is at most* $\lceil C/\varepsilon \rceil$, *where* $C = \frac{(K-1)\max_i\{L(f_i)\}\|x_0 - x^*\|^2}{2\beta}$.

Proof. In (5.2.3), by letting $u = x^*, v^j = w^i_{(n)}, j = 1,\ldots,K, j \neq i$, we have $p = x^i_{n+1}$ and

$$\begin{aligned} 2\mu(F(x^*) - F(x^i_{(n+1)})) &\geq \sum_{j=1, j\neq i}^{K} \left(\|x^i_{(n+1)} - x^*\|^2 - \|w^i_{(n)} - x^*\|^2 \right) \\ &= (K-1)\left(\|x^i_{(n+1)} - x^*\|^2 - \|w^i_{(n)} - x^*\|^2 \right). \end{aligned} \tag{5.2.8}$$

Using the definition of $w^i_{(n)}$ in Algorithm 5.1, we have

$$\begin{aligned} \sum_{i=1}^{K} \|w^i_{(n+1)} - x^*\|^2 &= \sum_{i=1}^{K} \left\| \sum_{j=1}^{K} D^{(n+1)}_{ji} x^j_{(n+1)} - x^* \right\|^2 \\ &= \sum_{i=1}^{K} \left\| \sum_{j=1}^{K} D^{(n+1)}_{ji} (x^j_{(n+1)} - x^*) \right\|^2 \\ &\leq \sum_{i=1}^{K} \sum_{j=1}^{K} D^{(n+1)}_{ji} \|x^j_{(n+1)} - x^*\|^2 \\ &= \sum_{j=1}^{K} \|x^j_{(n+1)} - x^*\|^2, \end{aligned} \tag{5.2.9}$$

where the second and the last equalities are due to the fact that $D^{(n+1)}$ is a doubly stochastic matrix and the inequality is due to the convexity of the function $\|\cdot\|^2$.

Thus by summing (5.2.8) over $i = 1, \ldots, K$ we obtain

$$2\mu \left(KF(x^*) - \sum_{i=1}^{K} F(x^i_{(n+1)}) \right) \geq (K-1) \left(\sum_{i=1}^{K} \|x^i_{(n+1)} - x^*\|^2 - \sum_{i=1}^{K} \|w^i_{(n)} - x^*\|^2 \right) \tag{5.2.10}$$

$$\geq (K-1) \left(\sum_{i=1}^{K} \|w^i_{(n+1)} - x^*\|^2 - \sum_{i=1}^{K} \|w^i_{(n)} - x^*\|^2 \right),$$

where the last inequality is due to (5.2.9).

Summing (5.2.10) over $n = 0, 1, \ldots, k-1$, and using the fact that $w^i_{(0)} = x_0, i = 1, \ldots, K$, yields

$$2\mu \left(kKF(x^*) - \sum_{n=0}^{k-1} \sum_{i=1}^{K} F(x^i_{(n+1)}) \right) \geq (K-1) \sum_{i=1}^{K} \left(\|w^i_{(k)} - x^*\|^2 - \|w^i_{(0)} - x^*\|^2 \right) \tag{5.2.11}$$

$$\geq -K(K-1)\|x_0 - x^*\|^2.$$

In (5.2.3), by letting $u = v^j = w^i_{(n)}, j = 1, \ldots, K, j \neq i$, we get $p = x^i_{(n+1)}$ and

$$2\mu \left(F(w^i_{(n)}) - F(x^i_{(n+1)}) \right) \geq \sum_{j=1, j\neq i}^{K} \|x^i_{(n+1)} - w^i_{(n)}\|^2 \geq 0. \tag{5.2.12}$$

From the way we compute $w^i_{(n)}, i = 1, \ldots, K$, and the facts that F is convex and $D^{(n)}$ is a

doubly stochastic matrix, we get

$$\begin{aligned}\sum_{i=1}^{K} F(w_{(n)}^{i}) &= \sum_{i=1}^{K} F\left(\sum_{j=1}^{K} D_{ji}^{(n)} x_{(n)}^{j}\right) \\ &\leq \sum_{i=1}^{K}\sum_{j=1}^{K} D_{ji}^{(n)} F(x_{(n)}^{j}) \\ &= \sum_{j=1}^{K} F(x_{(n)}^{j}).\end{aligned} \tag{5.2.13}$$

Now summing (5.2.12) over $i = 1, \ldots, K$ and using (5.2.13), we obtain

$$2\mu\left(\sum_{i=1}^{K} F(x_{(n)}^{i}) - \sum_{i=1}^{K} F(x_{(n+1)}^{i})\right) \geq 0. \tag{5.2.14}$$

This shows that the sums $\sum_{i=1}^{K} F(x_{(n)}^{i})$ are non-increasing as n increases. Hence,

$$\sum_{n=0}^{k-1}\sum_{i=1}^{K} F(x_{(n+1)}^{i}) \geq k\sum_{i=1}^{K} F(x_{(k)}^{i}). \tag{5.2.15}$$

Finally, combining (5.2.11) and (5.2.15) yields

$$2\mu\left(kKF(x^*) - k\sum_{i=1}^{K} F(x_{(k)}^{i})\right) \geq -K(K-1)\|x_0 - x^*\|^2.$$

Hence,

$$\min_{1,\ldots,K} F(x_{(k)}^{i}) - F(x^*) \leq \frac{1}{K}\sum_{i=1}^{K} F(x_{(k)}^{i}) - F(x^*) \leq \frac{(K-1)\|x_0 - x^*\|^2}{2\mu k}.$$

It then follows that $\min_{i=1,\ldots,K} F(x_{(k)}^{i}) - F(x^*) \leq \varepsilon$, if $k \geq \lceil C/\varepsilon \rceil$, where

$C = \frac{(K-1)\max_i\{L(f_i)\}\|x_0 - x^*\|^2}{2\beta}$, and hence that for any $k \geq \lceil C/\varepsilon \rceil$, $x_{(k)} := \arg\min\{x^i_{(k)} | F(x^i_{(k)}), i = 1, \ldots, K\}$ is an ε-optimal solution. □

Remark 5.2.1. If in the original problem (5.1.1), x is subject to a convex constraint $x \in \mathcal{C}$, where $\mathcal{C}$ is a convex set, we can impose this constraint in every subproblem in MSA and obtain the same complexity result. The only changes in the proof are in Lemma 5.2.4. If there is a constraint $x \in \mathcal{C}$, then (5.2.3) and (5.2.4) hold for any $u \in \mathcal{C}$ and the last equality in (5.2.4) becomes a "$\geq$" inequality due to the fact that the optimality conditions (5.2.5) become

$$\left\langle \nabla f_i(p) + \sum_{j=1, j\neq i}^{K} \left(\nabla f_j(v^j) + \frac{1}{\mu}(p - v^j) \right), u - p \right\rangle \geq 0, \forall u \in \mathcal{C}.$$

Unfortunately, this extension is not very practical, since for it to be useful, adding the constraint in every subproblem would most likely make most of these subproblems difficult to solve.

5.3 A class of fast multiple-splitting algorithms

In this section, we give a class of fast multiple-splitting algorithms (FaMSA) for solving problem (5.2.1) that require at most $O(1/\sqrt{\varepsilon})$ iterations to obtain an ε-optimal solution while requiring a computational effort at each iteration that is roughly the same as Algorithm 5.1. Our fast multiple-splitting algorithms are outlined in Algorithm 5.2, where $D^{(k)} \in \mathbb{R}^{K\times K}$ is a doubly stochastic matrix.

Algorithm 5.2 A Class of Fast Multiple-Splitting Algorithms (FaMSA)

Set $x_0 = x^i_{(0)} = \hat{w}^i_{(0)} = w^i_{(1)}, i = 1,\dots,K, t_1 = 1$, and choose $\mu \le 1/\max_{1\le i\le K} L(f_i)$
for $k = 1, 2, \cdots$ **do**
  for $i = 1,\dots,K$ **do**
    compute $x^i_{(k)} = p_i(w^i_{(k)},\dots,w^i_{(k)})$
  end for
  compute $\left(\hat{w}^1_{(k)},\dots,\hat{w}^K_{(k)}\right) := \left(x^1_{(k)},\dots,x^K_{(k)}\right) D^{(k)}$
  compute $t_{k+1} = (1+\sqrt{1+4t_k^2})/2$
  for $i = 1,\dots,K$ **do**
    compute $w^i_{(k+1)} := \hat{w}^i_{(k)} + \frac{1}{t_{k+1}}\left(t_k(x^i_{(k)} - \hat{w}^i_{(k-1)}) - (\hat{w}^i_{(k)} - \hat{w}^i_{(k-1)})\right)$.
  end for
end for

To establish the $O(1/\sqrt{\varepsilon})$ iteration complexity of FaMSA, we need the following lemma.

Lemma 5.3.1. *Suppose x^* is an optimal solution to problem* (5.2.1)*. For any choice of $\mu \le \max_{1\le i\le K} L(f_i)$, the sequence $\{x^i_{(k)}, w^i_{(k)}, \hat{w}^i_{(k)}\}_{i=1}^K$ generated by Algorithm 5.2 satisfies:*

$$2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1}) \ge (K-1)\sum_{i=1}^{K}\left(\|u^i_{k+1}\|^2 - \|u^i_k\|^2\right), \tag{5.3.1}$$

where $v_k := \sum_{i=1}^K F(x^i_{(k)}) - KF(x^*)$ and $u^i_k := t_k x^i_{(k)} - (t_k - 1)\hat{w}^i_{(k-1)} - x^*, i = 1,\dots,K$.

Proof. In (5.2.3), by letting $u = \hat{w}^i_{(k)}, v^j = w^i_{(k+1)}, j = 1,\dots,K, j \ne i$, we get $p = x^i_{(k+1)}$ and

$$\begin{aligned} 2\mu\left(F(\hat{w}^i_{(k)}) - F(x^i_{(k+1)})\right) &\ge \sum_{j=1, j\ne i}^{K}\left(\|x^i_{(k+1)} - \hat{w}^i_{(k)}\|^2 - \|w^i_{(k+1)} - \hat{w}^i_{(k)}\|^2\right) \\ &= (K-1)\left(\|x^i_{(k+1)} - \hat{w}^i_{(k)}\|^2 - \|w^i_{(k+1)} - \hat{w}^i_{(k)}\|^2\right). \end{aligned} \tag{5.3.2}$$

Summing (5.3.2) over $i = 1,\dots,K$, and using the facts that F is convex and $D^{(k)}$ is a

doubly stochastic matrix, we get

$$2\mu\left(\sum_{i=1}^{K}F(x^i_{(k)})-\sum_{i=1}^{K}F(x^i_{(k+1)})\right)\geq 2\mu\left(\sum_{i=1}^{K}F(\hat{w}^i_{(k)})-\sum_{i=1}^{K}F(x^i_{(k+1)})\right)$$
$$\geq (K-1)\sum_{i=1}^{K}\left(\|x^i_{(k+1)}-\hat{w}^i_{(k)}\|^2-\|w^i_{(k+1)}-\hat{w}^i_{(k)}\|^2\right),$$

i.e.,

$$2\mu(v_k-v_{k+1})\geq (K-1)\sum_{i=1}^{K}\left(\|x^i_{(k+1)}-\hat{w}^i_{(k)}\|^2-\|w^i_{(k+1)}-\hat{w}^i_{(k)}\|^2\right). \tag{5.3.3}$$

In (5.2.3), by letting $u=x^*, v^j=w^i_{(k+1)}$, we get $p=x^i_{(k+1)}$ and

$$2\mu\left(F(x^*)-F(x^i_{(k+1)})\right)\geq \sum_{j=1,j\neq i}^{K}\left(\|x^i_{(k+1)}-x^*\|^2-\|w^i_{(k+1)}-x^*\|^2\right) \tag{5.3.4}$$
$$=(K-1)\left(\|x^i_{(k+1)}-x^*\|^2-\|w^i_{(k+1)}-x^*\|^2\right).$$

Summing (5.3.4) over $i=1,\ldots,K$ we obtain

$$-2\mu v_{k+1}\geq (K-1)\sum_{i=1}^{K}\left(\|x^i_{(k+1)}-x^*\|^2-\|w^i_{(k+1)}-x^*\|^2\right). \tag{5.3.5}$$

Now multiplying (5.3.3) by t_k^2 and (5.3.5) by t_{k+1}, adding the resulting two inequalities,

using the relation $t_k^2 = t_{k+1}(t_{k+1}-1)$, and the identity (5.2.6), we get

$$
\begin{aligned}
& 2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1}) \qquad (5.3.6)\\
\geq & (K-1)\sum_{i=1}^{K} t_{k+1}(t_{k+1}-1)\left(\|x^i_{(k+1)} - \hat{w}^i_{(k)}\|^2 - \|w^i_{(k+1)} - \hat{w}^i_{(k)}\|^2\right)\\
& + (K-1)\sum_{i=1}^{K} t_{k+1}\left(\|x^i_{(k+1)} - x^*\|^2 - \|w^i_{(k+1)} - x^*\|^2\right)\\
= & (K-1)\sum_{i=1}^{K} t_{k+1}(t_{k+1}-1)\left(\|x^i_{(k+1)} - w^i_{(k+1)}\|^2 + 2\left\langle x^i_{(k+1)} - w^i_{(k+1)}, w^i_{(k+1)} - \hat{w}^i_{(k)}\right\rangle\right)\\
& + (K-1)\sum_{i=1}^{K} t_{k+1}\left(\|x^i_{(k+1)} - w^i_{(k+1)}\|^2 + 2\left\langle x^i_{(k+1)} - w^i_{(k+1)}, w^i_{(k+1)} - x^*\right\rangle\right)\\
= & (K-1)\sum_{i=1}^{K} t_{k+1}^2\|x^i_{(k+1)} - w^i_{(k+1)}\|^2\\
& + 2(K-1)t_{k+1}\sum_{i=1}^{K}\left\langle x^i_{(k+1)} - w^i_{(k+1)}, t_{k+1}w^i_{(k+1)} - (t_{k+1}-1)\hat{w}^i_{(k)} - x^*\right\rangle\\
= & (K-1)\sum_{i=1}^{K}\left(\|t_{k+1}x^i_{(k+1)} - (t_{k+1}-1)\hat{w}^i_{(k)} - x^*\|^2 - \|t_{k+1}w^i_{(k+1)} - (t_{k+1}-1)\hat{w}^i_{(k)} - x^*\|^2\right).
\end{aligned}
$$

From the way we compute $w^i_{(k+1)}$ in Algorithm 5.2, i.e.,

$$
w^i_{(k+1)} := \hat{w}^i_{(k)} + \frac{1}{t_{k+1}}\left(t_k(x^i_{(k)} - \hat{w}^i_{(k-1)}) - (\hat{w}^i_{(k)} - \hat{w}^i_{(k-1)})\right),
$$

it follows that

$$
t_{k+1}w^i_{(k+1)} - (t_{k+1}-1)\hat{w}^i_{(k)} - x^* = t_k x^i_{(k)} - (t_k-1)\hat{w}^i_{(k-1)} - x^*.
$$

Thus, from (5.3.6) and the definition of u_k^i it follows that

$$\begin{aligned}
&2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1}) \\
\geq &(K-1)\sum_{i=1}^{K}\left(\|t_{k+1}x_{(k+1)}^i - (t_{k+1}-1)\hat{w}_{(k)}^i - x^*\|^2 - \|t_k x_{(k)}^i - (t_k-1)\hat{w}_{(k-1)}^i - x^*\|^2\right) \\
= &(K-1)\sum_{i=1}^{K}\left(\|u_{k+1}^i\|^2 - \|u_k^i\|^2\right).
\end{aligned}$$

This completes the proof. □

Before proving our main complexity theorem to Algorithm 5.2, we note that the sequence $\{t_k\}$ generated by Algorithm 5.2 clearly satisfies $t_{k+1} \geq t_k + \frac{1}{2}$, and hence $t_k \geq (k+1)/2$ for all $k \geq 1$ since $t_1 = 1$.

Theorem 5.3.2. *Suppose x^* is an optimal solution to problem* (5.2.1)*. For any choice of $\mu \leq \max_{1\leq i\leq K} L(f_i)$, the sequence $\{x_{(k)}^i, w_{(k)}^i, \hat{w}_{(k)}^i\}_{i=1}^K$ generated by Algorithm 5.2 satisfies:*

$$\min_{i=1,\ldots,K} F(x_{(k)}^i) - F(x^*) \leq \frac{2(K-1)\|x_0 - x^*\|^2}{\mu(k+1)^2}. \tag{5.3.7}$$

Thus, the sequence $\{\min_{i=1,\ldots,K} F(x_{(k)}^i)\}$ produced by Algorithm 5.2 converges to $F(x^*)$. Moreover, if $\mu \geq \beta/\max_i\{L(f_i)\}$ where $0 < \beta \leq 1$, the number of iterations needed to obtain an ε-optimal solution is at most $\lfloor\sqrt{C/\varepsilon}\rfloor$, where $C = 2(K-1)\max_i\{L(f_i)\}\|x_0 - x^*\|^2/\beta$.

Proof. By rewriting (5.3.1) as

$$2\mu t_{k+1}^2 v_{k+1} + (K-1)\sum_{i=1}^{K}\|u_{k+1}^i\|^2 \leq 2\mu t_k^2 v_k + (K-1)\sum_{i=1}^{K}\|u_k^i\|^2,$$

we get

$$\begin{aligned}
2\mu\left(\frac{k+1}{2}\right)^2 v_k &\leq 2\mu t_k^2 v_k + (K-1)\sum_{i=1}^{K}\|u_k^i\|^2 \\
&\leq 2\mu t_1^2 v_1 + (K-1)\sum_{i=1}^{K}\|u_1^i\|^2 \\
&= 2\mu v_1 + (K-1)\sum_{i=1}^{K}\|x_{(1)}^i - x^*\|^2 \\
&\leq (K-1)\sum_{i=1}^{K}\|w_{(1)}^i - x^*\|^2 \\
&= K(K-1)\|x_0 - x^*\|^2,
\end{aligned}$$

where the first inequality is due to $t_k \geq (k+1)/2$, the first equality is from the facts that $t_1 = 1$ and $u_1^i = x_{(1)}^i - x^*$, the third inequality is from letting $k = 0$ in (5.3.5) and the last equality is due to $w_{(1)}^i = x_0, i = 1, \ldots, K$.

Thus, from $v_k = \sum_{i=1}^{K} F(x_{(k)}^i) - KF(x^*)$ we get

$$\sum_{i=1}^{K} F(x_{(k)}^i) - KF(x^*) \leq \frac{2K(K-1)\|x_0 - x^*\|^2}{\mu(k+1)^2},$$

which implies that

$$\min_{i=1,\ldots,K} F(x_{(k)}^i) - F(x^*) \leq \frac{1}{K}\sum_{i=1}^{K} F(x_{(k)}^i) - F(x^*) \leq \frac{2(K-1)\|x_0 - x^*\|^2}{\mu(k+1)^2},$$

i.e., (5.3.7) holds.

Moreover, it follows that if $C/(k+1)^2 \leq \varepsilon$, i.e., $k \geq \lfloor\sqrt{C/\varepsilon}\rfloor$, then $\min_{i=1,\ldots,K} F(x_{(k)}^i) -$

$F(x^*) \leq \varepsilon$, where $C = 2(K-1)\max_i\{L(f_i)\}\|x_0 - x^*\|^2/\beta$. This implies that for any $k \geq \lfloor\sqrt{C/\varepsilon}\rfloor$, $x_{(k)} := \arg\min\{x^i_{(k)}|F(x^i_{(k)}), i = 1,\ldots,K\}$ is an ε-optimal solution. □

Remark 5.3.1. Although we have assumed that the Lipschitz constants $L(f_i)$ are known, and hence that μ is chosen in Algorithm 5.2 to be smaller than $1/\max_{1\leq i\leq K}\{L(f_i)\}$, this can be relaxed by using the backtracking technique in [4] that chooses a μ at each iteration that is smaller than the μ used at the previous iteration and for which $F(p) \leq Q_i(w^i_{(k)},\ldots,w^i_{(k)},p,w^i_{(k)},\ldots,w^i_{(k)})$ for all i.

5.3.1 A variant of the fast multiple-splitting algorithm

In this section, we present a variant of the fast multiple-splitting algorithm (Algorithm 5.2) that is much more efficient and requires much less memory than Algorithm 5.2 for problems in which K is large. This variant uses $D^{(k)} := 1/Kee^\top$, where e is the n-dimensional vector with all ones, and replaces $x^i_{(k)}$ in the last line of Algorithm 5.2 by $\hat{w}^i_{(k)}$; i.e., in the last line of Algorithm 5.2, we compute $w^i_{(k+1)}$ for $i = 1,\ldots,K$ by the formula:

$$w^i_{(k+1)} := \hat{w}^i_{(k)} + \frac{t_k - 1}{t_{k+1}}(\hat{w}^i_{(k)} - \hat{w}^i_{(k-1)}).$$

It is easy to see that in this variant, the $\hat{w}^i_{(k)}, i = 1,\ldots,K$ are all the same and the $w^i_{(k+1)}, i = 1,\ldots,K$ are all the same. We call this variant FaMSA-s, where s refers to the fact that this variant computes a "single" vector $\hat{w}^k$ and a single vector $w_{(k+1)}$ at the k-th iteration. It is given below as Algorithm 5.3.

Algorithm 5.3 A variant of FaMSA (FaMSA-s)

Set $x_0 = x^i_{(0)} = \hat{w}_{(0)} = w_{(1)}, i = 1,\ldots,K, t_1 = 1$, and choose $\mu \leq 1/\max_{1\leq i\leq K} L(f_i)$
for $k = 1,2,\cdots$ **do**
  for $i = 1,\ldots,K$ **do**
    compute $x^i_{(k)} = p_i(w_{(k)},\ldots,w_{(k)})$
  end for
  compute $\hat{w}_{(k)} := \frac{1}{K}\sum_{i=1}^{K} x^i_{(k)}$
  compute $t_{k+1} = (1+\sqrt{1+4t_k^2})/2$
  for $i = 1,\ldots,K$ **do**
    compute $w_{(k+1)} := \hat{w}_{(k)} + \frac{t_k-1}{t_{k+1}}(\hat{w}_{(k)} - \hat{w}_{(k-1)})$.
  end for
end for

It is easy to verify that the following analog of Lemma 5.3.1 applies to Algorithm FaMSA-s.

Lemma 5.3.3. *Suppose x^* is an optimal solution to problem* (5.2.1)*. The sequence $\{w_{(k)}, \hat{w}_{(k)}\}$ generated by Algorithm FaMSA-s satisfies:*

$$2\mu(t_k^2 v_k - t_{k+1}^2 v_{k+1}) \geq (K-1)\sum_{i=1}^{K}\left(\|u_{k+1}\|^2 - \|u_k\|^2\right),$$

where $v_k := K(F(\hat{w}_{(k)}) - F(x^*))$ and $u_k := t_k\hat{w}_{(k)} - (t_k-1)\hat{w}_{(k-1)} - x^*$.

Proof. The proof is very similar to the proof of Lemma 5.3.1; hence, we leave it to the reader. The main difference is that instead of using the inequality $\sum_{i=1}^{K} F(\hat{w}^i_{(k)}) \leq \sum_{i=1}^{K} F(x^i_{(k)})$ to replace the sum involving $\hat{w}^i_{(k)}$, we use the fact that $KF(\hat{w}_{k+1}) \leq \sum_{i=1}^{K} F(x^i_{(k+1)})$ to replace the sum involving $x^i_{(k+1)}$ in the proof. □

From Lemma 5.3.3, Theorem 5.3.2 with $\hat{w}^i_{(k)}$ and $w^i_{(k)}$, respectively, for $i = 1,\ldots,K$ replaced by $\hat{w}_{(k)}$ and $w_{(k)}$ follows immediately for FaMSA-s.

Remark 5.3.2. Although for the above results we required all functions to be in the class of $C^{1,1}$, our algorithms can still be applied to solve nonsmooth problems by first smoothing all nonsmooth functions using the smoothing techniques described in Chapter 4.

5.4 Numerical experiments

We present some preliminary numerical experiments in this section. Specifically, we apply our MSA and FaMSA algorithms to solve the Fermat-Weber problem and a total variation and wavelet based image deblurring problem. All numerical experiments were run in MATLAB 7.3.0 on a Dell Precision 670 workstation with an Intel Xeon(TM) 3.4GHZ CPU and 6GB of RAM.

5.4.1 The Fermat-Weber problem

The Fermat-Weber (F-W) problem can be cast as:

$$\min F(x) \equiv \sum_{i=1}^{K} \|x - c^i\|, \tag{5.4.1}$$

where $c^i \in \mathbb{R}^n, i = 1, \ldots, K$ are K given points. Problem (5.4.1) can be reformulated as a second-order cone programming (SOCP) problem and thus solved in polynomial time by an interior-point method. Since there are K cones, the size of a standard form SOCP formulation for this problem is quite large for large K and n. Since $f_i(x) = \|x - c^i\|, i = 1, \ldots, K$ are not smooth, to apply our MSA and FaMSA algorithms, we need to smooth

them first. Here we adopt the smoothing technique discussed in Chapter 4; we approximate $f_i(x)$ by the smooth function

$$f_i^{\rho}(x) := \max\{\langle x - c^i, y\rangle - \frac{\rho}{2}\|y\|^2 : \|y\| \leq 1\}, \tag{5.4.2}$$

where $\rho > 0$ is a smoothness parameter. The gradient of f_i^{ρ}, $\nabla f_i^{\rho}(x) = y_i^*$, where y_i^* is the optimal solution to the optimization problem in (5.4.2). It is easy to show that $y_i^* = \frac{x-c^i}{\max\{\rho, \|x-c^i\|\}}$. Moreover, $\nabla f_i^{\rho}(x)$ is Lipschitz continuous with constant $L(f_i^{\rho}) = 1/\rho$. Now we can apply MSA, FaMSA and FaMSA-s to solve

$$\min \sum_{i=1}^{K} f_i^{\rho}(x). \tag{5.4.3}$$

The i-th subproblem in all of these algorithms corresponds to solving the following problem:

$$\begin{aligned} &p_i(w_{(k)}^i, \ldots, w_{(k)}^i) \\ &:= \arg\min_u f_i^{\rho}(u) + \textstyle\sum_{j=1, j\neq i}^{K} \left(f_j^{\rho}(w_{(k)}^i) + \langle \nabla f_j^{\rho}(w_{(k)}^i), u - w_{(k)}^i\rangle + \frac{1}{2\mu}\|u - w_{(k)}^i\|^2 \right). \end{aligned} \tag{5.4.4}$$

It is easy to check that the optimal solution to problem (5.4.4) is given by

$$u^* := \begin{cases} c^i + \dfrac{\rho(K-1)}{\mu + \rho(K-1)}(z_{(k)}^i - c^i), & \text{if } \|z_{(k)}^i - c^i\| \leq \rho + \frac{\mu}{K-1} \\ c^i + \dfrac{(K-1)\|z_{(k)}^i - c^i\| - \mu}{(K-1)\|z_{(k)}^i - c^i\|}(z_{(k)}^i - c^i), & \text{if } \|z_{(k)}^i - c^i\| > \rho + \frac{\mu}{K-1}, \end{cases}$$

where

$$z^i_{(k)} := w^i_{(k)} - \frac{\mu}{K-1} \sum_{j=1, j\neq i}^{K} \frac{w^i_{(k)} - c^j}{\max\{\rho, \|w^i_{(k)} - c^j\|\}}.$$

If we choose the doubly stochastic matrix $D^{(k)}$ to be $D^{(k)} := 1/Kee^\top$ in MSA as we do in FaMSA-s, all $w^i_{(k)}$'s are the same in MSA as they are in FaMSA-s. Hence, computing $x^i_{(k)}$, for $i = 1, \ldots, K$ in both algorithms can be done efficiently as follows.

$$\begin{cases} \hat{z}_{(k)} &= \sum_{j=1}^{K} \frac{w_{(k)} - c^j}{\max\{\rho, \|w_{(k)} - c^j\|\}} \\ z^i_{(k)} &= w_{(k)} - \frac{\mu}{K-1}(\hat{z} - \frac{w_{(k)} - c^i}{\max\{\rho, \|w_{(k)} - c^i\|\}}), \forall i = 1, \ldots, K \\ x^i_{(k)} &= c^i + (1 - \frac{\mu}{\max\{(K-1)\|z^i_{(k)} - c^i\|, \mu + \rho(K-1)\}})(z^i_{(k)} - c^i), \forall i = 1, \ldots, K \end{cases} \tag{5.4.5}$$

We compared the performance of MSA and FaMSA-s with the classical gradient method (Grad) and Nesterov's accelerated gradient method (Nest) for solving (5.4.3). The classical gradient method for solving (5.4.3) with step size $\tau > 0$ is:

$$x^{k+1} = x^k - \tau \sum_{j=1}^{K} \nabla f^\rho_j(x^k).$$

The variant of Nesterov's accelerated gradient method that we used is the following:

$$\begin{cases} x^k &= y^{k-1} - \tau \sum_{j=1}^{K} \nabla f^\rho_j(y^{k-1}) \\ y^k &= x^k - \frac{k-1}{k+2}(x^k - x^{k-1}). \end{cases}$$

We created random problems to test the performance of MSA, FaMSA-s, Grad and Nest as follows. Vectors $c^i \in \mathbb{R}^n, i = 1, \ldots, K$ were created with i.i.d. Gaussian entries from

$\mathcal{N}(0,n)$. The seed for generating random numbers in MATLAB was set to 0. We set the smoothness parameter ρ equal to 10^{-3}. The initial points $x^i, i = 1,\ldots,K$ were set to the average of all of the c^i's, i.e., $x^i_{(0)} = \frac{1}{K}\sum_{i=1}^{K} c^i$. We chose $D^{(k)}_{ij} = 1/K, i, j = 1,\ldots,K$ for all k in MSA. To compare the number of iterations needed by MSA and FaMSA-s, we first solved (5.4.1) by Mosek [72] after converting it into an SOCP problem to get the optimal solution x^*, and then terminated MSA, FaMSA-s, Grad and Nest when the relative error of the objective function value at the k-th iterate,

$$relerr := \frac{|\min_{i=1,\ldots,K} F(x^i_{(k)}) - F(x^*)|}{F(x^*)},$$

was less than 10^{-6}. We tested the performance of these four solvers for different choices of τ, which is the step size for Grad and Nest. Note that since the w^i's are the same in MSA with $D^{(k)} = \frac{1}{K}ee^\top$ for all k and in FaMSA-s, these two methods can be viewed as linearization methods in which the single function $\sum_{j=1,j\neq i}^{K} f_j(x)$ is linearized at the point w with only one proximal term $\frac{K-1}{2\mu}\|x-w\|$ in the i-th subproblem. So the step size for MSA and FaMSA-s is $\mu/(K-1)$. Hence, the parameter μ for MSA and FaMSA-s was set to $\mu = \tau(K-1)$ in our numerical tests.

Our results are presented in Table 5.1. The CPU times reported are in seconds. These results show that for the F-W problem, our implementations of MSA and FaMSA-s take roughly between two and three times as much time to solve each problem as taken by Grad and Nest, respectively. This is not surprising since it is clear that the computation of each set of K vectors $z^i_{(k)}$ and $x^i_{(k)}$ for $i = 1,\ldots,K$ in (5.4.5) is roughly comparable to a single

computation of the gradient, i.e., the K gradients of $f_i^{\rho}(x)$, for $i = 1, \ldots, K$. Moreover, for the simple F-W objective function, not much is gained by minimizing only one out of the K individual functions $f_i^{\rho}(x)$, $i = 1, \ldots, K$, when K is large as it is in our tests. Note that the number of iterations required by MSA and Grad were exactly the same on our set of test problems. When K is of a moderate size and the individual functions are more complicated, MSA should require fewer iterations than Grad.

The purpose of this set of tests was not to demonstrate any advantage that our algorithms might have over gradient methods. Rather, they were performed to validate our algorithms and show that the accelerated variants like algorithm Nest can reduce the number of iterations required to solve problems of the form (5.1.1). This is quite clear from the results reported in Table 5.1. We further note that FaMSA-s often takes one to three fewer iterations than Nest. Note that for some problems, the multiple-splitting algorithm took only one iteration to converge. The reason was that for these problems, the number of points was much larger than the dimension of the space. Therefore, the points were very compact and fairly uniformly distributed around the initial point; hence that point was quite likely to be very close to the optimal solution.

5.4.2 An image deblurring problem

In this section, we report the results of applying our multiple-splitting algorithms to a benchmark total variation and wavelet-based image deblurring problem from [37]. In this problem, the original image is the well-known Cameraman image of size 256×256 and the observed image is obtained after imposing a uniform blur of size 9×9 (denoted by

the operator A) and Gaussian noise (generated by the function *randn* in MATLAB with a seed of 0 and a standard deviation of 0.56). Since the vector of coefficients of the wavelet transform of the image is sparse in this problem and the total variation norm of the image is expected to be small, one can try to reconstruct the image x from the observed image b by solving the problem:

$$\min \quad \alpha \mathrm{TV}(x) + \beta \|\Phi x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{5.4.6}$$

where $\mathrm{TV}(x) := \sum_{ij} \sqrt{(x_{i+1,j} - x_{ij})^2 + (x_{ij} - x_{i,j+1})^2}$ is the total variation of x, Φ is the wavelet transform, A denotes the deblurring kernel and $\alpha > 0$, $\beta > 0$ are weighting parameters. Problem (5.4.6) involves minimizing the sum of three convex functions with $f_1(x) = \alpha \mathrm{TV}(x)$, $f_2(x) = \beta \|\Phi x\|_1$ and $f_3(x) = \frac{1}{2}\|Ax - b\|_2^2$.

To apply our multiple-splitting algorithms to solve (5.4.6), our theory requires all the functions to be smooth functions. So we needed to smooth the TV and the ℓ_1 functions first. We adopted the following way to smooth the TV function, widely used in the literature for doing this:

$$f_1^{\delta}(x) := \alpha \sum_{ij} \sqrt{(x_{i+1,j} - x_{ij})^2 + (x_{ij} - x_{i,j+1})^2 + \delta}.$$

The ℓ_1 function was smoothed in the way described in Chapter 4:

$$f_2^{\sigma}(x) := \beta \max_u \{\langle \Phi x, u \rangle - \frac{\sigma}{2}\|u\|^2 : \|u\|_\infty \leq 1\}.$$

Thus, the smooth version of problem (5.4.6) was:

$$\min_{x} \quad f_1^{\delta}(x) + f_2^{\sigma}(x) + f_3(x). \tag{5.4.7}$$

However, when we applied our multiple-splitting algorithms to (5.4.7), we actually performed the following computation on the k-th iteration:

$$\begin{cases} x^{k+1} := \arg\min_x & f_1(x) + \langle \nabla f_2^{\sigma}(w^k), x - w^k \rangle + \frac{1}{2\mu}\|x - w^k\|^2 \\ & + \langle \nabla f_3(w^k), x - w^k \rangle + \frac{1}{2\mu}\|x - w^k\|^2 \\ y^{k+1} := \arg\min_y & f_2^{\sigma}(y) + \langle \nabla f_1^{\delta}(w^k), y - w^k \rangle + \frac{1}{2\mu}\|y - w^k\|^2 \\ & + \langle \nabla f_3(w^k), y - w^k \rangle + \frac{1}{2\mu}\|y - w^k\|^2 \\ z^{k+1} := \arg\min_z & f_3(z) + \langle \nabla f_1^{\delta}(w^k), z - w^k \rangle + \frac{1}{2\mu}\|z - w^k\|^2 \\ & + \langle \nabla f_2^{\sigma}(w^k), z - w^k \rangle + \frac{1}{2\mu}\|z - w^k\|^2 \\ w^{k+1} := & (x^{k+1} + y^{k+1} + z^{k+1})/3. \end{cases} \tag{5.4.8}$$

Note that in (5.4.8), when we linearized the TV function, we used the smoothed TV function $f_1^{\delta}(\cdot)$, i.e., we computed the gradient of $f_1^{\delta}(\cdot)$. But when we solved the first subproblem, we used the nonsmooth TV function $f_1(\cdot)$, because there are efficient algorithms for solving this nonsmooth problem. Specifically, this subproblem can be reduced to:

$$x^{k+1} := \arg\min_x \frac{\alpha\mu}{2}\mathrm{TV}(x) + \frac{1}{2}\|x - \left(w^k - \mu(\nabla f_2^{\sigma}(w^k) + \nabla f_3(w^k)\right)/2)\|^2,$$

which is a standard TV-denoising problem. In our tests, we perform 10 iterations of the

algorithm proposed by Chambolle in [20] to approximately solve this problem. The second subproblem in (5.4.8) can be reduced to:

$$y^{k+1} := \arg\min_{y} \frac{\mu}{2} f_2^{\sigma}(y) + \frac{1}{2}\|y - (w^k - \mu\left(\nabla f_1^{\delta}(w^k) + \nabla f_3(w^k)\right)/2)\|^2. \qquad (5.4.9)$$

It is easy to check that the solution of (5.4.9) is given by:

$$y^{k+1} := \Phi^{\top}\left(\Phi\bar{w}^k - \frac{\mu\beta}{2}\tilde{w}^k\right)$$

where $(\tilde{w}^k)_j = \max\{-1, \min\{1, \frac{2(\Phi\bar{w}^k)_j}{2\sigma+\beta\mu}\}\}$ and $\bar{w}^k = w^k - \mu\left(\nabla f_1^{\delta}(w^k) + \nabla f_3(w^k)\right)/2$. The third subproblem in (5.4.8) corresponds to solving the following linear system:

$$(A^{\top}A + 2/\mu I)z = A^{\top}b - \nabla f_1^{\delta}(w^k) + 2/\mu w^k - \nabla f_2^{\delta}(w^k).$$

Solving this linear system is easy since the operator A has a special structure and thus $(A^{\top}A + 2/\mu I)$ can be inverted efficiently.

In our tests, we set $\alpha = 0.001$, $\beta = 0.035$ and used smoothing parameters $\delta = \sigma = 10^{-4}$. The initial points were all set equal to 0. We compared the performance of MSA, FaMSA, FaMSA-s and Grad for different μ and step sizes τ. In these comparisons, we simply terminated the codes after 500 iterations. The objective function value and the improvement signal noise ratio (ISNR) at different iterations are reported in Table 5.2. The ISNR is defined as $ISNR := 10\log_{10}\frac{\|x-b\|^2}{\|x-\bar{x}\|^2}$, where x is the reconstructed image and $\bar{x}$ is the true image. As we did for F-W problem, we always used $\mu = \tau(K-1)$ and since there were

three functions in this problem, we used $\mu = 2\tau$. For large μ, we did not report the results for all of the iterations since the comparisons are quite clear from the selected iterations. See Figure 5.1 for additional and more complete comparisons. We make the following observations from Table 5.2. For $\mu = 0.1$, FaMSA-s achieved the best objective function value in about 200 iterations and 152 CPU seconds. The best ISNR was also achieved by FaMSA-s, in about 300 iterations and 227 seconds. MSA and Grad were not able to obtain an acceptable solution in 500 iterations. In fact, they were only able to reduce the objective function to twice the near-optimal value of 3.86×10^4 achieved by FaMSA-s. For $\mu = 0.5$, FaMSA-s achieved the best objective function value and ISNR in 100 iterations and 76 seconds and 125 iterations and 94 seconds, respectively. Again, MSA and Grad did not achieve acceptable results even after 500 iterations. For $\mu = 1$, MSA achieved the best objective function value, 3.73×10^4, after 500 iterations and 349 CPU seconds, while the best ISNR was achieved by FaMSA-s in 80 iterations and 61 seconds. Also, the best objective function value achieved by FaMSA-s was at the 60-th iteration after only 47 CPU seconds. We also note that for $\mu = 0.1, 0.5$ and 1, MSA was always better than Grad and FaMSA-s was always slightly better than FaMSA. Another observation was that MSA always decreased the objective function value for $\mu = 0.1, 0.5$ and 1, while FaMSA and FaMSA-s always achieved near-optimal results in a relatively small number of iterations and then started getting worse. However, in practice, one would always terminate FaMSA and FaMSA-s once the objective function value started increasing. For $\mu = 5$, MSA gave very good results while the other three solvers diverged immediately. Specifically, the best objective function value 3.73×10^4 was achieved by MSA in 120 iterations and 80 CPU

seconds, and the best ISNR was achieved by MSA in 200 iterations and 132 CPU seconds. Thus, based on these observations, we conclude that FaMSA-s attains a nearly optimal solution very quickly for small μ while MSA is more stable for large μ.

We also plotted some figures to graphically illustrate the performance of these solvers. Figures (a), (b) and (c) in Figure 5.1 plot the objective function value versus the iteration number for $\mu = 0.1, 0.5$ and 1, respectively. Figures (d), (e) and (f) in Figure 5.1 plot ISNR versus the iteration number for $\mu = 0.1, 0.5$ and 1. We did not plot graphs for $\mu = 5$, since FaMSA, FaMSA-s and Grad diverged from the very first iteration. From Figure 5.1 we can see the comparisons clearly. Basically, these figures show that FaMSA and FaMSA-s achieve a nearly optimal solution very quickly. We can also see from (b), (c), (e) and (f) that FaMSA-s is always slightly better than FaMSA and MSA is always better than Grad.

We also tested setting $D^{(k)}$ to the identity matrix in MSA and FaMSA, but this choice, as expected, did not give as good results.

To see how MSA performed for the deblurring problem (5.4.7), we show the original (a), blurred (b) and reconstructed (c) cameraman images in Figure 5.2. The reconstructed image (c) is the one that was obtained by applying MSA with $\mu = 5$ after 200 iterations. The ISNR of the reconstructed image is 5.3182. From Figure 5.2 we see that MSA was able to recover the blurred image very well.

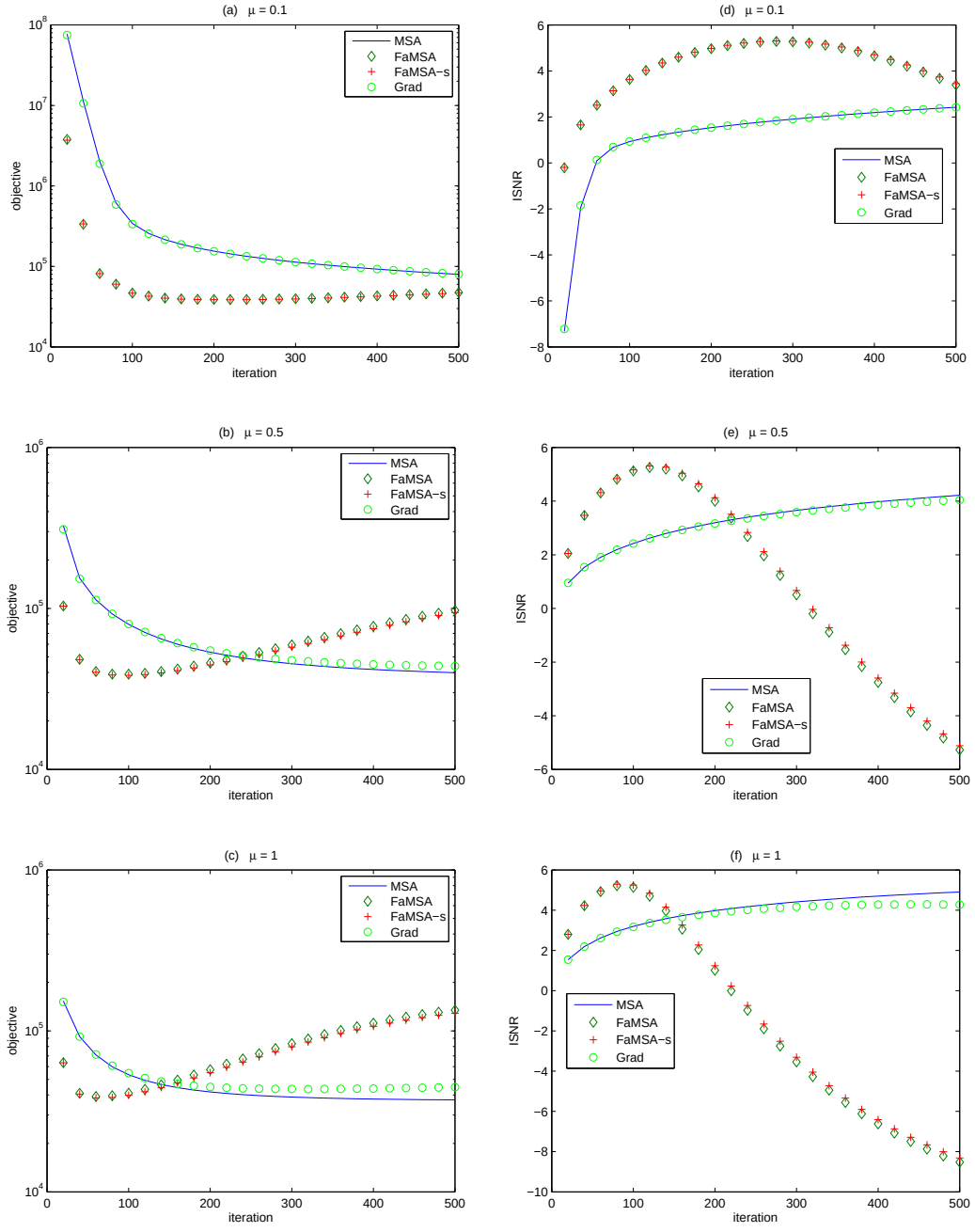

Figure 5.1: Comparison of MSA, FaMSA-s and Grad for different μ.

(a)

(b)

(c)

Figure 5.2: Using MSA to solve (5.4.7). (a): Original image; (b): Blurred image; (c): Reconstructed image by MSA

5.5 Conclusions

In this chapter, we proposed two classes of multiple-splitting algorithms based on alternating directions and optimal gradient techniques for minimizing the sum of K convex functions. Complexity bounds on the number of iterations required to obtain an ε-optimal solution for these algorithms were derived. Our algorithms are all parallelizable, which is attractive for practical applications involving large-scale optimization problems.

Table 5.1: Comparison of MSA, FaMSA-v, Grad and Nest on solving Fermat-Weber problem (5.4.3)

Problem		Mosek	MSA			FaMSA-s			Grad			Nest		
n	K	time	iter	relerr	time	iter	relerr	time	iter	relerr	time	iter	relerr	time
$\tau = 0.001$														
50	50	0.85	500	4.1e-05	0.73	107	8.4e-07	0.17	500	4.1e-05	0.21	109	9.0e-07	0.05
50	100	3.40	500	5.6e-06	1.44	69	9.9e-07	0.21	500	5.6e-06	0.42	72	8.5e-07	0.07
50	200	0.96	427	9.9e-07	2.44	47	8.8e-07	0.28	427	9.9e-07	0.69	49	8.6e-07	0.09
100	100	1.78	500	8.9e-06	1.68	94	9.8e-07	0.33	500	8.9e-06	0.51	97	9.1e-07	0.10
100	200	4.48	500	1.6e-06	3.35	60	9.2e-07	0.42	500	1.6e-06	1.00	62	9.2e-07	0.13
100	400	9.40	198	1.0e-06	2.68	34	9.5e-07	0.47	198	1.0e-06	0.79	36	9.2e-07	0.15
200	200	22.22	500	2.3e-06	4.36	75	9.9e-07	0.67	500	2.3e-06	1.39	77	9.9e-07	0.22
200	400	45.55	275	1.0e-06	4.81	41	9.9e-07	0.73	275	1.0e-06	1.54	43	9.8e-07	0.25
200	800	100.15	41	1.0e-06	1.44	15	9.7e-07	0.54	41	1.0e-06	0.46	16	9.8e-07	0.18
300	300	102.64	419	1.0e-06	6.73	52	9.9e-07	0.85	419	1.0e-06	2.22	54	9.9e-07	0.29
300	600	194.99	24	1.0e-06	0.79	11	9.9e-07	0.37	24	1.0e-06	0.26	12	9.9e-07	0.14
300	1200	401.54	1	5.8e-07	0.08	1	5.8e-07	0.08	1	5.8e-07	0.03	1	5.8e-07	0.03
$\tau = 0.01$														
50	50	0.84	238	9.9e-07	0.36	32	8.3e-07	0.06	238	9.9e-07	0.11	34	7.5e-07	0.02
50	100	3.36	93	9.9e-07	0.29	20	9.6e-07	0.07	93	9.9e-07	0.08	22	7.6e-07	0.03
50	200	0.96	42	9.9e-07	0.26	13	9.2e-07	0.09	42	9.9e-07	0.07	15	5.9e-07	0.03
100	100	1.78	160	1.0e-06	0.55	28	9.0e-07	0.11	160	1.0e-06	0.17	30	8.1e-07	0.04
100	200	4.48	62	9.8e-07	0.43	17	9.2e-07	0.13	62	9.8e-07	0.13	19	7.5e-07	0.05
100	400	9.46	20	9.5e-07	0.28	9	9.1e-07	0.13	20	9.5e-07	0.09	10	9.2e-07	0.05
200	200	22.37	91	1.0e-06	0.81	22	9.2e-07	0.21	91	1.0e-06	0.26	23	1.0e-06	0.07
200	400	45.56	28	9.7e-07	0.50	11	9.9e-07	0.21	28	9.7e-07	0.16	13	8.4e-07	0.08
200	800	99.38	4	1.0e-06	0.16	4	8.6e-07	0.16	4	1.0e-06	0.05	4	9.4e-07	0.05
300	300	100.48	42	9.9e-07	0.69	15	9.3e-07	0.26	42	9.9e-07	0.23	16	9.5e-07	0.09
300	600	194.88	3	9.7e-07	0.11	3	9.4e-07	0.11	3	9.7e-07	0.04	3	9.6e-07	0.04
300	1200	402.16	1	5.4e-07	0.08	1	5.4e-07	0.08	1	5.4e-07	0.03	1	5.4e-07	0.03
$\tau = 0.1$														
50	50	0.84	23	9.5e-07	0.05	9	3.4e-07	0.03	23	9.4e-07	0.02	10	5.4e-07	0.01
50	100	3.41	9	7.7e-07	0.04	5	6.1e-07	0.03	9	7.7e-07	0.02	6	5.1e-07	0.01
50	200	0.95	4	5.3e-07	0.04	3	2.9e-07	0.03	4	5.2e-07	0.01	3	1.0e-06	0.01
100	100	1.80	16	8.6e-07	0.07	8	3.6e-07	0.04	16	8.6e-07	0.02	9	4.1e-07	0.02
100	200	4.48	6	8.3e-07	0.05	4	7.2e-07	0.04	6	8.3e-07	0.02	5	5.2e-07	0.02
100	400	9.40	2	6.4e-07	0.04	2	4.2e-07	0.04	2	6.4e-07	0.02	2	6.4e-07	0.02
200	200	22.25	9	9.4e-07	0.09	6	5.8e-07	0.07	9	9.3e-07	0.03	6	9.4e-07	0.02
200	400	45.61	3	7.9e-07	0.07	3	5.0e-07	0.07	3	7.9e-07	0.02	3	6.9e-07	0.03
200	800	99.77	1	5.0e-07	0.05	1	5.0e-07	0.05	1	5.0e-07	0.02	1	5.0e-07	0.02
300	300	100.37	4	9.9e-07	0.08	4	6.7e-07	0.08	4	9.9e-07	0.03	4	8.4e-07	0.03
300	600	197.72	1	7.0e-07	0.05	1	7.0e-07	0.05	1	7.0e-07	0.02	1	7.0e-07	0.02
300	1200	412.49	1	2.1e-07	0.08	1	2.1e-07	0.08	1	2.1e-07	0.03	1	2.1e-07	0.03

Table 5.2: Comparison of MSA, FaMSA, FaMSA-s and Grad on solving TV-deblurring problem

	MSA		FaMSA		FaMSA-s		Grad	
Iter	obj	ISNR	obj	ISNR	obj	ISNR	obj	ISNR
$\mu = 0.1, \tau = 0.05$								
100	3.42e+005	0.9311	4.67e+004	3.6310	4.66e+004	3.6332	3.36e+005	0.9344
200	1.55e+005	1.5340	3.89e+004	4.9693	**3.86e+004**	4.9821	1.55e+005	1.5341
300	1.13e+005	1.9057	3.98e+004	5.2695	3.94e+004	**5.2989**	1.13e+005	1.9043
400	9.25e+004	2.1905	4.30e+004	4.6587	4.26e+004	4.7075	9.28e+004	2.1871
500	7.97e+004	2.4235	4.76e+004	3.3881	4.70e+004	3.4500	8.02e+004	2.4175
$\mu = 0.5, \tau = 0.25$								
25	2.41e+005	1.1343	7.70e+004	2.4777	7.69e+004	2.4784	2.36e+005	1.1408
50	1.29e+005	1.7359	4.31e+004	3.9343	4.28e+004	3.9416	1.29e+005	1.7376
75	9.66e+004	2.1260	3.92e+004	4.7122	3.88e+004	4.7324	9.67e+004	2.1250
100	7.96e+004	2.4243	3.90e+004	5.1257	**3.84e+004**	5.1638	8.00e+004	2.4198
125	6.92e+004	2.6659	3.97e+004	5.2558	3.90e+004	**5.3160**	6.98e+004	2.6569
150	6.21e+004	2.8682	4.12e+004	5.0880	4.04e+004	5.1737	6.30e+004	2.8538
175	5.71e+004	3.0416	4.33e+004	4.6478	4.23e+004	4.7576	5.82e+004	3.0207
200	5.34e+004	3.1928	4.58e+004	3.9964	4.46e+004	4.1258	5.48e+004	3.1646
225	5.06e+004	3.3266	4.86e+004	3.2006	4.73e+004	3.3442	5.22e+004	3.2902
250	4.85e+004	3.4463	5.18e+004	2.3223	5.03e+004	2.4758	5.03e+004	3.4009
275	4.67e+004	3.5545	5.54e+004	1.4132	5.37e+004	1.5723	4.88e+004	3.4991
300	4.54e+004	3.6529	5.93e+004	0.5078	5.74e+004	0.6705	4.76e+004	3.5869
500	3.99e+004	4.2186	9.74e+004	-5.2730	9.43e+004	-5.1193	4.38e+004	4.0416
$\mu = 1, \tau = 0.5$								
20	1.53e+005	1.5382	6.35e+004	2.7991	6.33e+004	2.8006	1.51e+005	1.5443
40	9.23e+004	2.1927	4.10e+004	4.2214	4.05e+004	4.2361	9.22e+004	2.1932
60	7.09e+004	2.6220	3.91e+004	4.9205	**3.84e+004**	4.9591	7.13e+004	2.6158
80	5.99e+004	2.9413	3.96e+004	5.2175	3.86e+004	**5.2890**	6.08e+004	2.9258
100	5.34e+004	3.1933	4.10e+004	5.1371	3.98e+004	5.2488	5.47e+004	3.1664
120	4.93e+004	3.4003	4.33e+004	4.6922	4.19e+004	4.8439	5.10e+004	3.3597
140	4.64e+004	3.5751	4.62e+004	3.9649	4.45e+004	4.1489	4.85e+004	3.5186
160	4.44e+004	3.7258	4.94e+004	3.0524	4.75e+004	3.2595	4.68e+004	3.6515
180	4.29e+004	3.8578	5.32e+004	2.0449	5.10e+004	2.2668	4.57e+004	3.7637
200	4.18e+004	3.9748	5.74e+004	1.0116	5.50e+004	1.2419	4.49e+004	3.8592
220	4.09e+004	4.0795	6.20e+004	-0.0045	5.93e+004	0.2311	4.44e+004	3.9407
240	4.02e+004	4.1741	6.70e+004	-0.9780	6.41e+004	-0.7394	4.40e+004	4.0103
260	3.96e+004	4.2602	7.22e+004	-1.8951	6.90e+004	-1.6561	4.37e+004	4.0695
280	3.92e+004	4.3388	7.77e+004	-2.7506	7.43e+004	-2.5136	4.36e+004	4.1197
300	3.88e+004	4.4111	8.34e+004	-3.5436	7.97e+004	-3.3102	4.35e+004	4.1620
500	**3.73e+004**	4.9042	1.35e+005	-8.5246	1.29e+005	-8.3127	4.47e+004	4.2742
$\mu = 5, \tau = 2.5$								
20	2.54e+007	-2.7911	1.10e+023	-157.9048	8.53e+018	-116.7985	5.63e+015	-84.9895
40	4.91e+005	3.7130	1.37e+040	-328.8532	8.05e+031	-246.5444	6.03e+022	-155.2917
60	4.69e+004	4.4065	3.59e+057	-503.0389	1.68e+045	-379.7479	6.55e+029	-225.6465
80	3.80e+004	4.6991	1.29e+075	-678.5934	4.93e+058	-514.4122	7.15e+036	-296.0278
100	3.74e+004	4.9027	5.53e+092	-854.9100	1.74e+072	-649.8897	7.84e+043	-366.4253
120	**3.73e+004**	5.0513	2.65e+110	-1031.7135	6.92e+085	-785.8864	8.61e+050	-436.8334
140	3.73e+004	5.1600	1.37e+128	-1208.8552	2.99e+099	-922.2437	9.47e+057	-507.2490
160	3.74e+004	5.2373	7.52e+145	-1386.2455	1.37e+113	-1058.8660	1.04e+065	-577.6699
180	3.76e+004	5.2888	4.31e+163	-1563.8262	6.62e+126	-1195.6913	1.15e+072	-648.0947
200	3.78e+004	**5.3182**	2.55e+181	-1741.5574	3.31e+140	-1332.6769	1.27e+079	-718.5224
500	4.27e+004	4.4523	Inf	-Inf	Inf	-Inf	5.70e+184	-1775.0426

Chapter 6

Conclusions

We proposed and studied several algorithms for solving sparse and low-rank optimization problems in this thesis.

In Chapter 2, we proposed a fixed-point continuation algorithm for solving matrix rank minimization problems. By adopting a Monte Carlo algorithm for approximately computing SVD in each iteration, we got a very efficient and powerful algorithm that can recover low-rank matrices using a limited number of observations effectively.

In Chapter 3, we studied several greedy type algorithms for solving matrix rank minimization problems. Their convergence and recoverability properties were analyzed. Our results on convergence and recoverability improved the previous results on matrix rank minimization.

In Chapter 4, we proposed alternating linearization methods for minimizing the sum of two convex functions. Under the assumption that the objective function has a Lipschitz continuous gradient, we analyzed for the first time the iteration complexity results

for alternating direction type methods. We proved that the basic and accelerated versions of our alternating linearization methods need respectively $O(1/\varepsilon)$ and $O(1/\sqrt{\varepsilon})$ iterations for an ε-optimal solution. Numerical results on compressed sensing and robust principal component analysis problems showed the efficiency of our methods.

In Chapter 5, we proposed multiple splitting algorithms for more general convex composite optimization problems, i.e., minimizing the sum of K convex functions for any $K \geq 2$. The $O(1/\varepsilon)$ and $O(1/\sqrt{\varepsilon})$ iteration complexity for both basic and accelerated multiple splitting algorithms were analyzed. Numerical results on Fermat-Weber problem and TV-denoising problem showed the advantages of our methods.

Bibliography

[1] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, *Fast image recovery using variable splitting and constrained optimization*, IEEE Transactions on Image Processing **19** (2010), no. 9, 2345?-2356.

[2] F. R. Bach, *Consistency of trace norm minimization*, Journal of Machine Learning Research **9** (2008), no. Jun, 1019–1048.

[3] O. Banerjee, L. El Ghaoui, and A. d'Aspremont, *Model selection through sparse maximum likelihood estimation for multivariate gaussian for binary data*, Journal of Machine Learning Research **9** (2008), 485–516.

[4] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sciences **2** (2009), no. 1, 183–202.

[5] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester, *Image inpainting*, Proceedings of SIGGRAPH 2000, New Orleans, USA (2000).

[6] D. P. Bertsekas, *Nonlinear programming, 2nd ed*, Athena Scientific, Belmont, Massachusetts, 1999.

[7] T. Blumensath and M. E. Davies, *Gradient pursuits*, IEEE Transactions on Signal Processing **56** (2008), no. 6, 2370–2382.

[8] T. Blumensath and M. E. Davies, *Iterative hard thresholding for compressed sensing*, Applied and Computational Harmonic Analysis **27** (2009), no. 3, 265–274.

[9] J. M. Borwein and A. S. Lewis, *Convex analysis and nonlinear optimization*, Springer-Verlag, 2003.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, to appear (2011).

[11] L. Bregman, *The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming*, USSR Computational Mathematics and Mathematical Physics **7** (1967), 200–217.

[12] S. Burer and R. D. C. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Mathematical Programming (Series B) **95** (2003), 329–357.

[13] ———, *Local minima and convergence in low-rank semidefinite programming*, Mathematical Programming **103** (2005), no. 3, 427–444.

[14] J. Cai, E. J. Candès, and Z. Shen, *A singular value thresholding algorithm for matrix completion*, SIAM J. on Optimization **20** (2010), no. 4, 1956–1982.

[15] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, Preprint available at http://arxiv.org/abs/0912.3599 (2009).

[16] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics **9** (2009), 717–772.

[17] E. J. Candès and J. Romberg, *ℓ_1-MAGIC: Recovery of sparse signals via convex programming*, Tech. report, Caltech, 2005.

[18] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on Information Theory **52** (2006), 489–509.

[19] E. J. Candès and T. Tao, *The power of convex relaxation: near-optimal matrix completion*, IEEE Trans. Inform. Theory **56** (2009), no. 5, 2053–2080.

[20] A. Chambolle, *An algorithm for total variation minimization and applications*, Journal of Mathematical Imaging and Vision **20** (2004), 89–97.

[21] P. L. Combettes, *Solving monotone inclusions via compositions of nonexpansive averaged operators*, Optimization **53** (2004), 475–504.

[22] P. L. Combettes and Jean-Christophe Pesquet, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE Journal of Selected Topics in Signal Processing **1** (2007), no. 4, 564–574.

[23] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, SIAM Journal on Multiscale Modeling and Simulation **4** (2005), no. 4, 1168–1200.

[24] Rice compressed sensing website, *http://dsp.rice.edu/cs*.

[25] W. Dai and O. Milenkovic, *Subspace pursuit for compressive sensing signal reconstruction*, IEEE Trans. on Information Theory **55** (2009), no. 5, 2230–2249.

[26] D. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory **52** (2006), 1289–1306.

[27] D. Donoho, Y. Tsaig, I. Drori, and J.-C. Starck, *Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit*, Tech. report, Stanford University, 2006.

[28] D. L. Donoho and Y. Tsaig, *Fast solution of ℓ_1-norm minimization problems when the solution may be sparse*, IEEE Transactions on Information Theory **54** (2008), no. 11, 4789–4812.

[29] J. Douglas and H. H. Rachford, *On the numerical solution of the heat conduction problem in 2 and 3 space variables*, Transactions of the American Mathematical Society **82** (1956), 421–439.

[30] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices ii: Computing low-rank approximations to a matrix*, SIAM J. Computing **36** (2006), 158–183.

[31] J. Eckstein, *Splitting methods for monotone operators with applications to parallel optimization*, Ph.D. thesis, Massachusetts Institute of Technology, 1989.

[32] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program. **55** (1992), 293–318.

[33] J. Eckstein and B. F. Svaiter, *A family of projective splitting methods for sum of two maximal monotone operators*, Math. Program. Ser. B **111** (2008), 173–199.

[34] _____, *General projective splitting methods for sums of maximal monotone operators*, SIAM J. Control Optim. **48** (2009), no. 2, 787–811.

[35] M. Fazel, *Matrix rank minimization with applications*, Ph.D. thesis, Stanford University, 2002.

[36] M. Fazel, H. Hindi, and S. Boyd, *A rank minimization heuristic with application to minimum order system approximation*, Proceedings of the American Control Conference, vol. 6, 2001, pp. 4734–4739.

[37] M. Figueiredo and R. Nowak, *An EM algorithm for wavelet-based image restoration*, IEEE Transactions on Image Processing **12** (2003), 906–916.

[38] 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 on Selected Topics in Signal Processing **1** (2007), no. 4.

[39] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical lasso*, Biostatistics (2007).

[40] D. Gabay, *Applications of the method of multipliers to variational inequalities*, Augmented Lagrangian Methods: Applications to the Solution of Boundary Value Problems (M. Fortin and R. Glowinski, eds.), North-Hollan, Amsterdam, 1983.

[41] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite-element approximations*, Comp. Math. Appl. **2** (1976), 17–40.

[42] L. El Ghaoui and P. Gahinet, *Rank minimization under LMI constraints: A framework for output feedback problems*, Proceedings of the European Control Conference, 1993.

[43] R. Glowinski and P. Le Tallec, *Augmented lagrangian and operator-splitting methods in nonlinear mechanics*, SIAM, Philadelphia, Pennsylvania, 1989.

[44] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, *Eigentaste: A constant time collaborative filtering algorithm*, Information Retrieval **4** (2001), no. 2, 133–151.

[45] D. Goldfarb and S. Ma, *Fast multiple splitting algorithms for convex optimization*, Tech. report, Department of IEOR, Columbia University. Preprint available at http://arxiv.org/abs/0912.4570, 2009.

[46] ———, *Convergence of fixed point continuation algorithms for matrix rank minimization*, Foundations of Computational Mathematics **11** (2011), no. 2, 183–210.

[47] D. Goldfarb, S. Ma, and K. Scheinberg, *Fast alternating linearization methods for minimizing the sum of two convex functions*, Tech. report, Department of IEOR, Columbia University. Preprint available at http://arxiv.org/abs/0912.4571, 2010.

[48] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci. **2** (2009), 323–343.

[49] E. T. Hale, W. Yin, and Y. Zhang, *A fixed-point continuation method for ℓ_1-regularized minimization with applications to compressed sensing*, Tech. report, CAAM TR07-07, 2007.

[50] ———, *Fixed-point continuation for ℓ_1-minimization: Methodology and convergence*, SIAM Journal on Optimization **19** (2008), no. 3, 1107–1130.

[51] B. S. He, L.-Z. Liao, D. Han, and H. Yang, *A new inexact alternating direction method for monotone variational inequalities*, Math. Program. **92** (2002), 103–118.

[52] B. S. He, M. Tao, M. Xu, and X. Yuan, *Alternating direction based contraction method for generally separable linearly constrained convex programming problems*, Optimization-online: http://www.optimization-online.org/DB_HTML/2009/11/2465.html (2009).

[53] B. S. He, H. Yang, and S. L. Wang, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, Journal of optimization theory and applications **106** (2000), no. 2, 337–356.

[54] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms ii: Advanced theory and bundle methods*, Springer-Verlag, New York, 1993.

[55] R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge University Press, 1985.

[56] R. H. Keshavan, A. Montanari, and S. Oh, *Matrix completion from a few entries*, IEEE Trans. on Info. Theory **56** (2010), 2980–2998.

[57] S. J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, *A method for large-scale ℓ_1-regularized least-squares*, IEEE Journal on Selected Topics in Signal Processing **4** (2007), no. 1, 606–617.

[58] K. C. Kiwiel, C. H. Rosa, and A. Ruszczynski, *Proximal decomposition via alternating linearization*, SIAM J. Optimization **9** (1999), no. 3, 668–689.

[59] R. M. Larsen, *PROPACK - software for large and sparse SVD calculations*, Available from http://sun.stanford.edu/∼rmunk/PROPACK.

[60] K. Lee and Y. Bresler, *ADMIRA: atomic decomposition for minimum rank approximation*, ArXiv preprint: arXiv:0905.0044 (2009).

[61] K. Lee and Y. Bresler, *Efficient and guaranteed rank minimization by atomic decomposition*, preprint, available at arXiv: 0901.1898v1 (2009).

[62] K. Lee and Y. Bresler, *Guaranteed minimum rank approximation from linear observations by nuclear norm minimization with an ellipsoidal constraint*, Arxiv preprint arXiv:0903.4742 (2009).

[63] L. Li, W. Huang, I. Gu, and Q. Tian, *Statistical modeling of complex backgrounds for foreground object detection*, IEEE Trans. on Image Processing **13** (2004), no. 11, 1459–1472.

[64] Z. Lin, M. Chen, L. Wu, and Y. Ma, *The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices*, preprint (2009).

[65] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some of its algorithmic applications*, Combinatorica **15** (1995), 215–245.

[66] P. L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis **16** (1979), 964–979.

[67] Z. Liu and L. Vandenberghe, *Interior-point method for nuclear norm approximation with application to system identification.*, SIAM Journal on Matrix Analysis and Applications **31** (2009), no. 3, 1235–1256.

[68] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, Mathematical Programming Series A **128** (2011), 321–353.

[69] S. Ma, W. Yin, Y. Zhang, and A. Chakraborty, *An efficient algorithm for compressed MR imaging using total variation and wavelets*, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (2008), 1–8.

[70] J. Malick, J. Povh, F. Rendl, and A. Wiegele, *Regularization methods for semidefinite programming*, SIAM Journal on Optimization **20** (2009), 336–356.

[71] R. Meka, P. Jain, and I. S. Dhillon, *Guaranteed rank minimization via singular value projection*, Arxiv preprint, available at http://arxiv.org/abs/0909.5457 (2009).

[72] Mosek ApS Inc., *The Mosek optimization tools, ver 6.*, 2009.

[73] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM Journal on Computing **24** (1995), 227–234.

[74] D. Needell and J. A. Tropp, *CoSaMP: Iterative signal recovery from incomplete and inaccurate samples*, Applied and Computational Harmonic Analysis **26** (2009), 301–321.

[75] Y. E. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence* $O(1/k^2)$, Dokl. Akad. Nauk SSSR **269** (1983), 543–547.

[76] ———, *Introductory lectures on convex optimization*, **87** (2004), xviii+236, A basic course. MR MR2142598 (2005k:90001)

[77] ———, *Smooth minimization for non-smooth functions*, Math. Program. Ser. A **103** (2005), 127–152.

[78] ———, *Gradient methods for minimizing composite objective function*, CORE Discussion Paper 2007/76 (2007).

[79] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, SIAM Journal on Multiscale Modeling and Simulation **4** (2005), no. 2, 460–489.

[80] D. H. Peaceman and H. H. Rachford, *The numerical solution of parabolic elliptic differential equations*, SIAM Journal on Applied Mathematics **3** (1955), 28–41.

[81] B. Recht, M. Fazel, and P. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Review **52** (2010), no. 3, 471–501.

[82] J. D. M. Rennie and N. Srebro, *Fast maximum margin matrix factorization for collaborative prediction*, Proceedings of the International Conference of Machine Learning, 2005.

[83] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D **60** (1992), 259–268.

[84] K. Scheinberg, S. Ma, and D. Goldfarb, *Sparse inverse covariance selection via alternating linearization methods*, Proceedings of the Neural Information Processing Systems (NIPS), 2010.

[85] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, *Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization*, Molecular Biology of the Cell **9** (1998), 3273–3297.

[86] J. E. Spingarn, *Partial inverse of a monotone operator*, Appl. Math. Optim. **10** (1983), 247–265.

[87] N. Srebro, *Learning with matrix factorizations*, Ph.D. thesis, Massachusetts Institute of Technology, 2004.

[88] N. Srebro and T. Jaakkola, *Weighted low-rank approximations*, Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), 2003.

[89] J. F. Sturm, *Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones*, Optimization Methods and Software **11-12** (1999), 625–653.

[90] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal Royal Statistical Society B **58** (1996), 267–288.

[91] A. N. Tikhonov and V. Y. Arsenin, *Solutions of ill-posed problems*, Winston, New York, 1977.

[92] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3 - a Matlab software package for semidefinite programming*, Optimization Methods and Software **11** (1999), 545–581.

[93] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, Pacific J. Optimization **6** (2010), 615–640.

[94] J. Tropp, *Just relax: Convex programming methods for identifying sparse signals*, IEEE Transactions on Information Theory **51** (2006), 1030–1051.

[95] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, *Missing value estimation methods for DNA microarrays*, Bioinformatics **17** (2001), no. 6, 520–525.

[96] P. Tseng, *Further applications of a splitting algorithm to decomposition in variational inequalities and convex programming*, Mathematical Programming **48** (1990), 249–263.

[97] ———, *Applications of a splitting algorithm to decomposition in convex programming and variational inequalities*, SIAM J. Control and Optimization **29** (1991), no. 1, 119–138.

[98] ———, *A modified forward-backward splitting method for maximal monotone mappings*, SIAM Journal on Control and Optimization **38** (2000), no. 2, 431–446.

[99] ———, *On accelerated proximal gradient methods for convex-concave optimization*, submitted to SIAM J. Optim. (2008).

[100] R. H. Tütüncü, K.-C. Toh, and M. J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Mathematical Programming Series B **95** (2003), 189–217.

[101] E. van den Berg and M. P. Friedlander, *Probing the Pareto frontier for basis pursuit solutions*, SIAM J. on Scientific Computing **31** (2008), no. 2, 890–912.

[102] M. Wainwright, P. Ravikumar, and J. Lafferty, *High-dimensional graphical model selection using ℓ_1-regularized logistic regression*, NIPS **19** (2007), 1465–1472.

[103] Z. Wen, D. Goldfarb, and W. Yin, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Mathematical Programming Computation **2** (2010), 203–230.

[104] 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** (2009), no. 4, 1832–1857.

[105] J. Yang and Y. Zhang, *Alternating direction algorithms for ℓ_1 problems in compressive sensing*, SIAM Journal on Scientific Computing **33** (2011), no. 1, 250–278.

[106] 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** (2008), no. 1, 143–168.

[107] M. Yuan and Y. Lin, *Model selection and estimation in the Gaussian graphical model*, Biometrika **94** (2007), no. 1, 19–35.

[108] X. Yuan, *Alternating direction methods for sparse covariance selection*, (2009), Preprint available at http://www.optimization-online.org/DB_HTML/2009/09/2390.html.

[109] X. Yuan and J. Yang, *Sparse and low rank matrix decomposition via alternating direction methods*, preprint (2009).