Efficient Algorithms for Compressed Sensing and Matrix Completion

Ke Wei

St Hilda's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2014

Abstract

Compressed sensing and matrix completion are two new data acquisition techniques whose efficiency is achieved by exploring low dimensional structures in high dimensional data. Despite the combinatorial nature of compressed sensing and matrix completion, there has been significant development of computationally efficient algorithms which can produce accurate desired solutions to these problems. In this thesis, we are concerned with the development of low per iteration computational complexity algorithms for compressed sensing and matrix completion.

First, we derive a locally optimal stepsize selection rule for the simplest iterative hard thresholding algorithm for matrix completion, and obtain a simple yet efficient algorithm. It is observed to have average case performance superior in some aspects to other matrix completion algorithms.

To balance the fast convergence rates of more sophisticated recovery algorithms with the low per iteration computational cost of simple line-search algorithms, we introduce a family of conjugate gradient iterative hard thresholding algorithms for both compressed sensing and matrix completion. The theoretical results establish recovery guarantees for the restarted and projected variants of the algorithms, while the empirical performance comparisons establish significant computational advantages of the proposed methods over other hard thresholding algorithms.

Finally, we introduce an alternating steepest descent method and a scaled variant especially designed for the matrix completion problem based on a simple factorization model of the low rank matrix. The computational efficacy of this method is achieved by reducing the high per iteration computational cost of the second order method and fully exploring the numerical linear algebra structure in the algorithm. Empirical evaluations establish the effectiveness of the proposed algorithms, compared with other state-of-the-art algorithms.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Jared Tanner, for his guidance and generous support. Jared's pursuit of excellence and careful criticisms has greatly influenced me on how to do scientific research. I would follow this with special thanks to, Dr. Coralia Cartis, for her regular monitoring of my progress and constant encouragement. Further special thanks to Dr. Jeff Blanchard for the valuable opportunity of working together on some research projects.

I benefited a lot from the involvement in the Edinburgh Compressed Sensing (E-COS) group, especially from the regular reading groups. I am greatly appreciated for the opportunity of joining the lovely and active Numerical Analysis Group in Oxford, whose internal and external seminars have broadened my knowledge of the literature. Thanks a lot to all the members of both groups for bringing diversity to my experiences.

I am very grateful for the unforgettable days spent with Bubacarr Bah, Andrew Thompson and Martin Lotz in Edinburgh; special thanks goes to Andrew Thompson for reading substantial parts of the thesis. I am also grateful to all my friends who have supported me through the years and made my time in Edinburgh and Oxford enjoyable. In particular, I would like to thank Yujia Chen, Sheng Fang, Jinhan Xie, Kuan Xu, Yiming Yan and Shengxin Zhu for their company and encouragement. I want to thank my officemate Ingrid v. Glehn specially for giving me much advice about English grammar.

I would like to take this opportunity to thank my parents for their unconditional love and support. They have always believed in me and been fully supportive of all my decisions. I also want to thank my younger brother for his love and support. Finally my deepest gratitude goes to my wife Miaomiao for being always standing on my side and encouraging me through all the difficult times.

To my parents
for their unconditional love and support

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

Modern digital signal processing follows the celebrated Shannon-Nyquist-Whittaker theorem [101, 87], which shows that any band-limited signal can be exactly reconstructed from discrete samples if the sampling rate is twice the maximum component frequency of the signal. However, in many applications, the conventional approach usually involves the acquisition of very high dimensional data. Methods for more efficient data acquisition have received broad interest in the information community. This greater efficiency is achieved by approximating the high dimensional data by a low dimensional model and exploring the inherent simplicity of the model. Two notable examples of techniques which allow for more efficient data acquisition through the existence of such low dimensional models are: compressed sensing and low rank matrix completion.

1.1 Compressed sensing

Compressed sensing is a novel sampling paradigm which proposes to recover compressible or simply sparse signals from a number of linear measurements that is proportional to the desired compression rate rather than the ambient dimension containing the data. Moreover, the measurement process can be independent of the data to be measured. Since the pioneering work of Candès, Romberg and Tao [30] and Donoho [40], compressed sensing has received significant attentions and found a wide range of applications. For example, in medical imaging, the techniques of compressed sensing have been successfully used in Medical Resonance Imaging (MRI), which has substantially reduced the waiting timing of a patient in the MRI scanner [69, 70].

Other profound applications include but are not limited to the single-pixel camera [44], radar imaging [82], communication theory [7] and computational biology [36].

Let $x^o \in \mathbb{R}^n$ be the signal to be measured; x^o is *k-sparse* if x^o has at most k nonzero entries, denoted by $\|x^o\|_0 \leq k$ where $\|\cdot\|_0$ counts the number of nonzero entries in a vector. Let $A \in \mathbb{R}^{m\times n}$ be the sensing matrix from which we obtain n measurements $y = Ax^o$, where the number of measurements is less than the full dimension of the measured data, that is $m < n$. To recover the underlying k-sparse signal x^o, a natural approach is to solve for the sparsest solution to the underdetermined system:

$$\min_{x\in\mathbb{R}^N} \|x\|_0 \quad \text{subject to} \quad Ax = y. \tag{1.1}$$

Problem (1.1) is often referred to as the *ℓ_0-norm* or the *cardinality* minimization problem; and recovering x^o through the ℓ_0-norm minimization is often referred to as the *ℓ_0-decoder*. Due to the combinatorial nature of the cardinality objective, it often requires searching over all possible sparse patterns of x to find the global minimum of (1.1), and the computational complexity is exponential in the size of x. Indeed, it has been proven that (1.1) is an NP-hard problem [76].

1.1.1 Successful recovery of the ℓ_0-decoder

For the simplest sensing model $y = Ax^o$ with x^o being k-sparse, it is not hard to see that as long as there is a unique k-sparse solution to (1.1), the ℓ_0-decoder is guaranteed to reconstruct x^o. Recall that the *null space* of a matrix $A \in \mathbb{R}^{m\times n}$ consists of all the vectors $x \in \mathbb{R}^n$ such that $Ax = 0$, denoted $\mathcal{N}(A) = \{x : Ax = 0\}$. Let Σ_k denote the set of all k-sparse vectors, i.e. $\Sigma_k = \{x : \|x\|_0 \leq k\}$. The existence of a unique sparse solution to the ℓ_0-norm minimization problem (1.1) is fully determined by the null space condition.

Theorem 1.1 ([23])**.** For any $x^o \in \Sigma_k$ and $y = Ax^o$, there exists a unique k-sparse solution to (1.1) if and only if

$$\mathcal{N}(A) \cap \Sigma_{2k} = \emptyset. \tag{1.2}$$

The null space condition guarantees the successful recovery of the sparse vectors using the ℓ_0-decoder. However, it is difficult to to be verified and does not guarantee any other tractable approaches. Thus conditions based on more intuitive properties of the measurement matrix are of highly interest. First, let us consider the best sensing

matrix one can expect, that is, when $m = n$ and $A = \Phi$ is a unitary matrix. Let ϕ_i be the ith column of Φ. Then two key equivalent properties of Φ are

$$\phi_i^* \phi_j = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

and

$$\|\Phi x\|_2 = \|x\|_2, \ \forall x \in \mathbb{R}^n. \tag{1.4}$$

For an overcomplete matrix A with $m < n$, clearly neither (1.3) nor (1.4) can be true. However, it is desirable to have A preserve some properties close to (1.3) or (1.4). Coherence and the restricted isometry constant are adaptions of (1.3) and (1.4) respectively to an overcomplete matrix.

Definition 1.2 ([42])**.** For a given matrix $A \in \mathbb{R}^{m\times n}$, the coherence of A is defined as

$$\mu(A) = \max_{1\le i,j\le n, i\neq j} \frac{|a_i^* a_j|}{\|a_i\|_2 \|a_j\|_2}, \tag{1.5}$$

where a_i, a_j are the ith and jth columns of A respectively.

Coherence was first proposed by Donoho and Huo [42] for the concatenation of two unitary matrices, and then extended to a general overcomplete matrix by Donoho and Elad [41]. In [96], Tropp presented a block variant of coherence which is associated with sparsity levels of measured signals. A potential advantage of coherence is that it is easy to evaluate. If $A = [\Phi, \ \Psi]$ with $\Phi \in \mathbb{R}^{m\times m}$ and $\Psi \in \mathbb{R}^{m\times m}$ being both unitary, one has $1/\sqrt{m} \le \mu(A) \le 1$; and the lower bound can be achieved, for instance, if Φ is an identity matrix and Ψ is a Discrete Fourier matrix [42, 41]. For a general matrix $A \in \mathbb{R}^{m\times n}$ of full rank, in [89], it has been shown that

$$\mu(A) \ge \sqrt{\frac{n-m}{m(n-1)}}.$$

While coherence defines the maximum correlations between every two columns of A, the restricted isometry constant quantifies the deviation of A from an isometry operator when acting on k-sparse vectors.

Definition 1.3 (Restricted isometry constants (RICs), [31])**.** Let A be an $m \times n$ matrix. Then for every integer $0 < k \le m$, the restricted isometry constant, $\delta_k(A)$, of A with respect to k is defined as the smallest value such that

$$(1 - \delta_k(A)) \|x\|_2^2 \le \|Ax\|_2^2 \le (1 + \delta_k(A)) \|x\|_2^2 \tag{1.6}$$

holds for all $x \in \Sigma_k$.

Restricted isometry constants were originally defined in [31], and then independently generalized to the asymmetric RICs in [9] and [50]. For any index set $\Gamma \subset \{1, \cdots, N\}$, let A_Γ be the submatrix formed by the columns of A indexed by the set Γ. The restricted isometry constant of A can be expressed explicitly as

$$\delta_k(A) = \max\left(\max_{|\Gamma|=k} \lambda^{\max}\left(A_\Gamma^* A_\Gamma\right) - 1, 1 - \min_{|\Gamma|=k} \lambda^{\min}\left(A_\Gamma^* A_\Gamma\right)\right),$$

where $\lambda^{\max}\left(A_\Gamma^* A_\Gamma\right)$ and $\lambda^{\min}\left(A_\Gamma^* A_\Gamma\right)$ are the largest and smallest eigenvalues of $A_\Gamma^* A_\Gamma$ respectively. For the decay properties of RICs, see [8]. Although there are no deterministic ways known to design a matrix with small δ_k when the number of measurements is proportional to the sparsity level, for certain families of random matrices, e.g., Gaussian matrices and Bernoulli matrices, they have small RICs with overwhelmingly high probability if $k \leq C \cdot m/(\log(n/m) + 1)$ for some constant $C > 0$ [6].

The conditions for the existence of a unique solution to (1.1), i.e. successful recovery of the ℓ_0-decoder, can be established in terms of coherence and RICs as follows.

Theorem 1.4 ([23, 31])**.** For any $x^o \in \Sigma_k$ and $y = Ax^o$, there is a unique solution to (1.1) if

$$k < \frac{1}{2}\left(1 + \mu(A)^{-1}\right) \quad \text{or} \quad \delta_{2k}(A) < 1.$$

1.1.2 Alternatives to the ℓ_0-decoder

Since the ℓ_0-decoder is intractable, more computationally efficient algorithms are needed. One of the most widely studied methods is the ℓ_1-*decoder*, also known as *Basis Pursuit* in the signal processing literature, which proposes to recover k-sparse signals by solving the ℓ_1-norm minimization problem

$$\min_{x \in \mathbb{R}^n} \|x\|_1 \quad \text{subject to} \quad Ax = y, \tag{1.7}$$

where $\|x\|_1 = \sum_{i=1}^n |x_i|$ denotes the ℓ_1-norm of x. Problem (1.7) can be further cast into a linear programming and solved by the interior point method in polynomial time [22, 78]. The following theorem establishes the conditions for successful recovery of the ℓ_1-decoder in terms of coherence and RICs.

Theorem 1.5 ([96, 26])**.** For any $x^o \in \Sigma_k$ and $y = Ax^o$, x^o is a unique solution to (1.7) if

$$k < \frac{1}{2}\left(1 + \mu(A)^{-1}\right) \quad \text{or} \quad \delta_{2k}(A) < \sqrt{2} - 1.$$

Remark 1.6.

1. The RIC condition for the ℓ_1-decoder was firstly established in [31], and subsequently refined by several different authors [96, 42, 41, 31, 26, 48]. The result from [26] is chosen for its particular simplicity.

2. The ℓ_1-decoder and ℓ_0-decoder share the same unique solution x^o under the conditions in Thm. 1.5, a phenomenon often referred to as ℓ_1/ℓ_0 equivalence.

3. From the remark after Def. 1.2, we can see that $\mu(A)$ is proportional to $1/\sqrt{m}$ when $m \ll n$. Hence the condition based on the coherence guarantees the ℓ_1-decoder is able to recover a k-sparse signal if k is of the order $O(\sqrt{m})$. However, δ_{2k} is typically small for a wide range of random matrices if k is of the order $O(m/\log(n/m))$, which also implies the successful recovery of the ℓ_1-decoder. Therefore RICs are more powerful than coherence on predicting the sparsity levels of the signals that can be recovered. Since the conditions in Thm. 1.5 are typical for many other algorithms, RICs will be our analytical tool in this thesis.

In addition to the ℓ_1-decoder, there have been many greedy/iterative algorithms which are designed to target problem (1.1) directly when the desired sparsity level is known *a priori*. A partial list of the greedy algorithms for compressed sensing include Orthogonal Matching Pursuit (OMP) [97], Compressive Sampling Matching Pursuit (CoSaMP) [77], Subspace Pursuit (SP) [34], (Normalized) Iterative Hard Thresholding ((N)IHT) [19, 20] and Hard Thresholding Pursuit (HTP) [49]. In this section, we will briefly review IHT and NIHT. More details of the other algorithms will be introduced in the following chapters when necessary.

IHT, Alg. 1, is the simplest iterative hard thresholding algorithm which applies the projected gradient descent method with constant stepsize to the non-convex constrained problem

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax - y\|_2^2 \quad \text{subject to} \quad \|x\|_0 \leq k. \tag{1.8}$$

In each iteration of IHT, the current estimate x_{l-1} is updated along the gradient descent direction r_{l-1}, using a fixed stepsize α, followed by hard thresholding onto the space of k-sparse vectors. Computationally the hard thresholding operator is

composed of two parts. First the principal support of the approximation is identified, then the object is projected onto the principal support. The principal support of cardinality k, denoted here in pseudo-code by $\texttt{PrincipalSupport}_k(\cdot)$, consists of the indices for the k-largest entries in magnitude. Projecting onto the principal support set Γ, denoted by pseudo-code $\texttt{Proj}_\Gamma(\cdot)$, follows by setting to zero all the entries not on the support.

Algorithm 1 Iterative Hard Thresholding (IHT, [19])

Initilization: sparsity level k, $w_{-1} = A^*y$, $\Gamma_0 = \texttt{PrincipalSupport}_k(w_{-1})$, $x_0 = \texttt{Proj}_{\Gamma_0}(w_{-1})$, stepsize α, $l = 1$

Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ — gradient descent direction
2: $w_{l-1} = x_{l-1} + \alpha r_{l-1}$ — gradient descent update
3: $\Gamma_l = \texttt{PrincipalSupport}_k(w_{l-1})$ — proxy to the support set
4: $x_l = \texttt{Proj}_{\Gamma_l}(w_{l-1})$ — solution update

Output: x_l

IHT has a similar recovery guarantee to the ℓ_1-decoder in terms of RICs.

Theorem 1.7 ([19, 48])**.** For any $x^o \in \Sigma_k$ and $y = Ax^o$, the iterates of IHT converge to x^o provided

$$\delta_{3k}(A) < \frac{1}{2}.$$

NIHT for compressed sensing, Alg. 2, is an accelerated iterative hard thresholding algorithm, and corresponds to Alg. 1 with the stepsize selected adaptively to be locally optimal. During iteration l of NIHT, the current estimate x_{l-1} is updated along the gradient descent direction r_{l-1} with the stepsize α_{l-1} selected to be the steepest descent stepsize at x_{l-1} for problem (1.8) being restricted onto the current support set estimate Γ_{l-1}

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \left\| A\texttt{Proj}_{\Gamma_{l-1}}(x) - y \right\|_2^2, \tag{1.9}$$

and then followed by hard thresholding to a k-sparse vector.

NIHT has been proven to be able to recover the solution to (1.1) provided the measurement matrix has small RICs.

Theorem 1.8 ([20])**.** For any $x^o \in \Sigma_k$ and $y = Ax^o$, the iterates of NIHT converge to x^o provided

$$\delta_{3k}(A) < \frac{1}{5}.$$

Proof. See Thm. 2.1 in Chapter 2 for an analogous proof. □

Algorithm 2 Normalized Iterative Hard Thresholding (NIHT, [20])

Initilization: sparsity level k, $w_{-1} = A^*y$, $\Gamma_0 = \texttt{PrincipalSupport}_k(w_{-1})$, $x_0 = \texttt{Proj}_{\Gamma_0}(w_{-1})$, $l = 1$

Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ — gradient descent direction

2: $\alpha_{l-1} = \dfrac{\left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}{\left\|A\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}$ — compute the stepsize

3: $w_{l-1} = x_{l-1} + \alpha_{l-1}r_{l-1}$ — gradient descent update

4: $\Gamma_l = \texttt{PrincipalSupport}_k(w_{l-1})$ — proxy to the support set

5: $x_l = \texttt{Proj}_{\Gamma_l}(w_{l-1})$ — solution update

Output: x_l

1.2 Matrix completion

In compressed sensing, vectors of length n that have $k \ll n$ nonzero entries can be efficiently determined from less than n linear measurements. Similarly, $m \times n$ matrices with inherent simplicity can be uniquely determined from $p < mn$ measurements. A particularly trivial example is the simplicity model of matrices with few non-zero entries, which can be recast as standard compressed sensing by reforming the matrix as a sparse vector of length mn. In matrix completion, rather than the sparsity assumption as in compressed sensing, the observed matrix is assumed to be low rank. Let $X^o \in \mathbb{R}^{m\times n}$ be the matrix to be measured with $\text{rank}(X^o) < \min(m, n)$. Let $\mathcal{A}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ be a linear operator of the form

$$\mathcal{A}(X) := \begin{pmatrix} \langle A_1, X\rangle \\ \langle A_2, X\rangle \\ \vdots \\ \langle A_p, X\rangle \end{pmatrix}, \tag{1.10}$$

where $\{A_\ell\}_{\ell=1,\cdots,p}$ are p distinct $m \times n$ matrices making up the sensing operator, and $\langle A_\ell, X\rangle = \text{trace}(A_\ell^* X)$. Then we take $p < mn$ linear measurements of the low rank matrix X^o via $y = \mathcal{A}(X^o)$. Explicitly seeking the lowest rank matrix consistent with the measurements is expressed as

$$\min_{X\in\mathbb{R}^{m\times n}} \ \text{rank}(X) \ \text{ subject to } \ \mathcal{A}(X) = y. \tag{1.11}$$

There are two particularly interesting sensing operators. One is *Gaussian sensing*, in which the entries of each sensing matrix A_ℓ are sampled from the normal distribution. The other one is *entry sensing* where each sensing matrix A_ℓ is of the form $e_i e_j^*$, which

corresponds to directly measuring entries of X. More precisely, in entry sensing, we are concerned with whether it is possible to recover a low rank matrix from only a few known entries,

$$\begin{pmatrix} X^o_{1,1} & ? & X^o_{1,3} & \cdots & X^o_{1,n} \\ X^o_{2,1} & X^o_{2,2} & X^o_{2,3} & \cdots & ? \\ ? & X^o_{3,2} & ? & \cdots & X^o_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X^o_{m,1} & X^o_{m,2} & ? & \cdots & X^o_{m,n} \end{pmatrix}.$$

Explicitly seeking the lowest rank matrix consistent with the known entries is often directly expressed as

$$\min_{X \in \mathbb{R}^{m \times n}} \ \operatorname{rank}(X) \ \text{ subject to } \ P_\Omega(X) = P_\Omega(X^o), \tag{1.12}$$

where X^o is the underlying matrix to be reconstructed, Ω is a subset of indices for the known entries, and P_Ω denotes the sampling operator which acquires only the entries indexed by Ω.

When a subset of the matrix entries are directly measured, finding the unknown entries is typically referred to as "matrix completion" [29, 32]. Recovering the unknown matrix when each sensing matrix is dense is usually referred to as "(affine) rank minimization". However, for conciseness we will refer to recovery of a low rank matrix from both entry sensing and dense sensing as "matrix completion".

Matrices are natural representations for many datasets, and the low rank models are usually used to capture the principal information in the data matrices and provide low dimensional approximations. Therefore the matrix completion problem arises in a wide variety of practical contexts, such as model reduction [68], pattern recognition [46], and machine learning [3, 4]. A famous example is the *Netflix* problem [88]. In a recommender system for movies, users are encouraged to rate the movies based on their preferences; but users usually rate only very few of the movies so there are only a few scattered known entries for the data matrix. Thus one would like to complete the matrix so that the vender is able to recommend the movies that users like. Since users' preferences are usually determined by several factors, the data matrix can be assumed to be approximately low rank.

Compressed sensing can be viewed as a special case of matrix completion when the matrix is constrained to be diagonal. Therefore problem (1.11) is generally NP-hard [58], and it is worth studying other tractable approaches for recovering low rank matrices from a few number of independent measurements.

1.2.1 Nuclear Norm Minimization

Let $X = U\Sigma V^*$ be the singular value decomposition (SVD) of X. The nuclear norm, also known as the Schatten-1 norm, of X is defined as the sum of its singular values

$$\|X\|_* = \sum_i \sigma_i(X). \tag{1.13}$$

Analogous to the ℓ_1-decoder for compressed sensing, the nuclear norm minimization (NNM) method proposes to recover the low rank matrix by solving a convex optimization problem which replaces the rank objective in (1.11) with the nuclear norm of the matrix, namely,

$$\min_{X \in \mathbb{R}^{m \times n}} \quad \|X\|_* \quad \text{subject to} \quad \mathcal{A}(X) = y. \tag{1.14}$$

Problem (1.14) can be further cast into semidefinite programming (SDP) [85], and then solved using the interior point method [93].

The nuclear norm minimization method is amendable to detailed analysis for both dense and entry sensing. The restricted isometry constant analogue in matrix completion can be used to establish recovery guarantees for dense sensing.

Definition 1.9 (Restricted isometry constants (RICs), [85])**.** Let $\mathcal{A}(\cdot)$ be a linear map of $m \times n$ matrices to vectors of length p as defined in (1.10). For every integer $1 \leq r \leq \min(m, n)$, the restricted isometry constant, $\delta_r(\mathcal{A})$, of $\mathcal{A}(\cdot)$ is defined to be the smallest number such that[1]

$$(1 - \delta_r(\mathcal{A}))\, \|X\|_F^2 \leq \|\mathcal{A}(X)\|_2^2 \leq (1 + \delta_r(\mathcal{A}))\, \|X\|_F^2 \tag{1.15}$$

holds for all matrices of rank at most r.

A parallel result to Thm. 1.4 is given below which establishes the existence of the unique solution to the matrix completion problem (1.11).

Theorem 1.10 ([85])**.** Let $y = \mathcal{A}(X^o)$ with $\text{rank}(X^o) \leq r$. Then X^o is the unique solution to (1.11) if

$$\delta_{2r}(\mathcal{A}) < 1. \tag{1.16}$$

[1]The restricted isometry constant in the matrix completion setting is still denoted by $\delta(\cdot)$, with its matrix completion context making clear its distinction from the restricted isometry constant in compressed sensing.

Furthermore, the nuclear norm minimization is guaranteed to find the underlying low rank matrix if the sensing operator has small RICs.

Theorem 1.11 ([85])**.** Let $y = \mathcal{A}(X^o)$ with $\text{rank}(X^o) \leq r$. Then X^o is the unique solution to (1.14) if

$$\delta_{5r}(\mathcal{A}) < \frac{1}{10}. \tag{1.17}$$

It was first proved in [85] that several random measurement ensembles satisfy the RIC condition of Thm. 1.11 with overwhelmingly high probability provided the number of measurements $p \geq C \cdot r(m+n-r)\log(mn)$ for some constant $C > 0$. Typical examples include the ensembles in which each sensing matrix A_ℓ has i.i.d Gaussian $\mathcal{N}(0, 1/p)$ entries or i.i.d entries that are equal likely to take the value $1/\sqrt{p}$ or $-1/\sqrt{p}$. The result in Thm. 1.11 was refined to $p \geq C \cdot r(m+n-r)$ in [28] by employing a new covering argument.

However, the RIC based condition is only applicable for certain dense sensing operators, and not applicable for entry sensing. For any entry sensing operator $P_\Omega(\cdot)$ with $\Omega^c \neq \emptyset$, there exists a rank one matrix X which only has one nonzero entry in Ω^c such that $P_\Omega(X) = 0$, which implies $\delta_r(P_\Omega) = 1$ for any $r \geq 1$ and violates the condition in Thm. 1.11. More quantitative, non-RIC based estimates have been established for the convex relaxation (1.14) in entry sensing based on a notion of incoherence introduced in [29].

Definition 1.12. For a subspace $U \in \mathbb{R}^n$ of dimension r, let $\texttt{Proj}_U(\cdot)$ be the orthogonal projection onto U. Then the coherence of U is defined to be

$$\mu(U) = \frac{n}{r} \max_{1 \leq i \leq n} \left\| \texttt{Proj}_U(e_i) \right\|^2. \tag{1.18}$$

Given X^o be an $m \times n$ matrix of rank r, provided the singular vectors of X^o are weakly correlated with the canonical basis in terms of the coherence, Candès and Recht [29] proved that with high probability, the nuclear norm minimization method (1.14) is able to recover the unknown entries of X^o from uniformly random samples if $p \geq C \cdot r \max(m,n)^{6/5} \log(\max(m,n))$; later Candès and Tao [32] sharpened this result to $p \geq C \cdot r \max(m,n) \log^6(\max(m,n))$. Gross [56] extended the notion of entry sensing to a more general setting, and presented a novel technique which led to a simplified argument. Recht [84] then applied the techniques in [56] and derived the simplest argument for entry sensing.

Theorem 1.13 ([84])**.** Let X^o be a $m \times n$ matrix of rank r, and $X^o = U\Sigma V^*$ be the reduced singular value decomposition where $U \in \mathbb{R}^{m\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, $V \in \mathbb{R}^{n\times r}$. Assume that a) both $\mu(U)$ and $\mu(V)$ are bounded above by some positive number μ_0, and b) the matrix UV^* has a maximum entry bounded by $\mu_1\sqrt{r/\max(m,n)^2}$ for some positive number μ_1. Suppose p entries of X^o are sampled uniformly at random. Then if

$$p \geq 32 \max\{\mu_1^2, \mu_0\} r(m+n)\beta \log^2(2\max(m,n)),$$

for some $\beta > 1$, the minimiser to problem (1.14) is unique and equal to X^o with probability at least $1 - 6\log(\max(m,n))(m+n)^{2-2\beta} - \max(m,n)^{2-2\beta^{1/2}}$.

1.2.2 Non-convex Methods

Although there exists rigorous analysis for the ability of the nuclear norm minimization to recover a low rank matrix, the interior point method for (1.14) becomes inefficient if the size of the matrix is large because huge systems of linear equations need to be solved to compute the Newton directions [98]. As an alternative, there have been many algorithms which are designed to target (1.11) directly; many of them are adaptions of the algorithms for compressed sensing. For instance, IHT, HTP and CoSaMP have all been extended to matrix completion [52, 60, 66]. We present IHT as an example.

IHT for matrix completion, Alg. 3, applies the projected gradient method for the non-convex problem

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2}\|\mathcal{A}(x) - y\|_F^2 \quad \text{subject to} \quad \operatorname{rank}(X) \leq r. \tag{1.19}$$

In each iteration of IHT, the current estimate X_{l-1} is updated along the gradient descent direction R_{l-1}, using a constant stepsize α, followed by thresholding to the matrix of rank at most r that is nearest in the Frobenius norm. Computationally the hard thresholding operator is composed of two parts. First the principal component subspace is identified, then the matrix is projected onto the principal component subspace. The principal component subspace of dimension r can be identified in terms of either its column or row space. Here we make use of the column space, using the leading r left singular vectors, denoted here by $\texttt{PrincipalLeftSingularVectors}_r(\cdot)$. Projecting onto the principal subspace U, denoted in pseudo-code by $\texttt{Proj}_U(\cdot)$, follows by either multiplying the objective matrix on the left by the projection matrix UU^*, or by setting all but the r leading singular values of the objective matrix to zeroes.

Algorithm 3 IHT for matrix completion ([60, 52])

Initilization: rank r, $W_{-1} = \mathcal{A}^*(y)$, $U_0 = \texttt{PrincipalLeftSingularVectors}_r(W_{-1})$, $X_0 = \texttt{Proj}_{U_0}(W_{-1})$, stepsize α, $l = 1$

Iteration: During iteration l, **do**

1: $R_{l-1} = \mathcal{A}^*(y - \mathcal{A}(X_{l-1}))$ gradient descent direction
2: $W_{l-1} = X_{l-1} + \alpha R_{l-1}$ gradient descent update
3: $U_l = \texttt{PrincipalLeftSingularVectors}_r(W_{l-1})$ proxy to the subspace
4: $X_l = \texttt{Proj}_{U_l}(W_{l-1})$ solution update

Output: X_l

In Alg 3, the adjoint of the sensing operator, $\mathcal{A}^*(\cdot)$, is defined as

$$\mathcal{A}^*(y) := \sum_{\ell=1}^{p} y(\ell) A_\ell. \tag{1.20}$$

IHT for matrix completion has been independently studied in [60] and [52]. It has been proven in [60] that if the sensing operator $\mathcal{A}(\cdot)$ satisfies $\delta_{2r}(\mathcal{A}) \leq 1/3$ then IHT with stepsize $\alpha = 1/(1+\delta_{2r})$ is guaranteed to recover any rank r matrix measured by $\mathcal{A}(\cdot)$. However, this stepsize selection is not advocated for implementation as RICs are NP-hard to calculate [58]. In [52], IHT is analysed with the stepsize $\alpha = 1$, and the authors show that the iterates of IHT are guaranteed to converge to the underlying rank r matrix if $\delta_{3r} \leq 1/\sqrt{8}$. As a simple corollary of Thm. 2.1 in Chapter 2, the following result improves the results in [60, 52].

Corollary 1.14. Let $y = \mathcal{A}(X^o)$ with $\text{rank}(X^o) \leq r$. Then the iterates of IHT with a unit stepsize converge to X^o if

$$\delta_{3r}(\mathcal{A}) < \frac{1}{2}. \tag{1.21}$$

Proof. See Thm. 2.1 in Chapter 2 for an analogous proof. □

The iterative thresholding algorithms typically update a current rank r estimate along a line-search direction which departs from the manifold of rank r matrices[2], then projects back onto the rank r manifold by computing the nearest matrix in the Frobenius norm. The most direct implementation of the projection involves computing a partial singular value decomposition of an $m \times n$ matrix in each iteration, which has computational complexity $O(n^3)$ for $m \sim n$ [53]. For entry sensing, computing

[2]It is well known that the set of $m \times n$ rank r matrices is an $r(m+n-r)$ dimensional matrix manifold [24].

the SVD is the dominant cost in each iteration[3], which limits their applicability for large size matrix. To reduce the high computational cost of the SVD, one approach is to further exploit the local structure of the rank r manifold and develop Riemannian manifold algorithms for matrix completion. Examples include LRGeomCG [99], which is a nonlinear conjugate gradient method, and ScGrassMC [79], which uses a scaled gradient in the subspace update, as well as other geometries and metrics in [74, 75]. For more details, see Sec. 5.4.1 in Chapter 5.

There are other methods which can avoid computing the SVD by explicitly remaining on the rank r manifold using the factorization $X = YZ$ where $Y \in \mathbb{R}^{m\times r}$ and $Z \in \mathbb{R}^{r\times n}$. Based on this simple factorization model, rather than solving (1.11), algorithms are designed to solve the non-convex problem

$$\min_{\substack{Y\in\mathbb{R}^{m\times r}\\ Z\in\mathbb{R}^{r\times n}}} \frac{1}{2}\left\|P_\Omega(X^o) - P_\Omega(YZ)\right\|_F^2. \tag{1.22}$$

Algorithms for the solution of (1.22) usually follows an alternating minimization scheme, with PowerFactorization (PF) as a representative, see Alg. 4. In each iteration, PF minimizes over one component while the other one is held fixed.

Algorithm 4 PowerFactorization (PF, [57])

Initilization: rank r, $Y_0 \in \mathbb{R}^{m\times r}$, $Z_0 \in \mathbb{R}^{r\times n}$, $l = 1$

Iteration: During iteration l, **do**

1: Fix Z_{l-1}, solve $Y_l = \arg\min_{Y\in\mathbb{R}^{m\times r}} \left\|P_\Omega(X^o) - P_\Omega(YZ_{l-1})\right\|_F^2$ update Y component

2: Fix Y_l, solve $Z_l = \arg\min_{Z\in\mathbb{R}^{r\times n}} \left\|P_\Omega(X^o) - P_\Omega(Y_lZ)\right\|_F^2$ update Z component

Output: $X_l = Y_lZ_l$

Note that each subproblem in Alg. 4 is a standard least squares problem which can be solved efficiently by the conjugate gradient iterative method. In [57], a rank increment technique is also proposed to improve the performance of the algorithm.

1.3 Outline of the thesis

In Chapter 2, we propose an adaptive way to choose the stepsizes in IHT for matrix completion (Alg. 3), and introduce NIHT for matrix completion (Alg. 5), which is

[3]The related low rank approximation problem [85] has an $O(n^4)$ per iteration cost dominated by computing the sensing operator $\mathcal{A}(\cdot)$ and its adjoint $\mathcal{A}^*(\cdot)$; in this context, computing the SVD is of secondary cost.

an extension of NIHT for compressed sensing (Alg. 2). NIHT is proven to have a near optimal recovery guarantee using RICs (Thm. 2.1), and is observed to be able to recover a random low rank matrix from near the minimum number of measurements.

In Chapter 3, we introduce the CGIHT (Algs. 9, 10, and 11) family of algorithms for compressed sensing. CGIHT is designed to balance the low per iteration complexity of simple hard thresholding algorithms with the fast asymptotic convergence rates of more sophisticated sparse approximation algorithms. We establish uniform recovery guarantees and stability to noise for the restarted and projected variants of CGIHT (Algs. 10, 11) in terms of RICs (Thms. 3.3 and 3.4) of the measurement matrix. Extensive empirical tests utilizing the GPU software *GAGA* [13, 12] establishes the computational advantages of CGIHT in terms of both the sizes of the problems that can be recovered and the overall computational time.

In Chapter 4, the CGIHT family of algorithms are adapted for the matrix completion problem (Algs. 16, 17). A recovery guarantee is established for the projected variant of CGIHT in terms of RICs of the sensing operator (Thm. 4.1). CGIHT is observed to have average case performance superior to NIHT and other hard thresholding algorithms for matrix completion.

In Chapter 5, we present an alternating steepest descent algorithm, ASD (Alg. 20) and a scaled variant, ScaledASD (Alg. 21), for the fixed-rank matrix completion problem in entry sensing. Empirical evaluation of ASD and ScaledASD on both image inpainting and random problems shows they are competitive with other state-of-the-art matrix completion algorithms in terms of recoverable rank and overall computational time. In particular, ASD and ScaledASD are highly efficient for large size problems due to their low per iteration computational complexity, especially when computing the solutions to moderate accuracy. A preliminary convergence analysis is also presented (Thm. 5.3).

Finally, Chapter 6 concludes the thesis with summarizing the main contributions and pointing out some potential research directions for the future.

In this thesis, we are mainly considering the iterative hard thresholding algorithms for compressed sensing and matrix completion which are targeting the problems directly, but bear in mind that there are other potential candidates of fast algorithms, including the algorithms specifically for the convex relaxations and the recently introduced AMP algorithm [71]. The materials in this thesis have resulted in one journal

publication and four preprints which have been submitted.

Publications

1. *Normalized iterative hard thresholding for matrix completion*; SIAM J. Sci. Comput., 35(5) (2013), pp. S104 - S125. (with Jared Tanner)

Preprints

2. *CGIHT: Conjugate gradient iterative hard thresholding for compressed sensing and matrix completion*, submitted. (with Jeff Blanchard and Jared Tanner)
3. *Low rank matrix completion by alternating steepest descent methods*, submitted. (with Jared Tanner)
4. *Conjugate gradient iterative hard thresholding: Observed noise stability for compressed sensing*, submitted. (With Jeff Blanchard and Jared Tanner)
5. *Fast iterative hard thresholding for compressed sensing*, submitted.

Chapter 2

Normalized Iterative Hard Thresholding for Matrix Completion

Low rank matrices can be uniquely determined from fewer linear measurements, or entries, than the total number of entries in the matrix. Moreover, there is a growing literature of computationally efficient algorithms which can recover a low rank matrix from such limited information. Normalized iterative hard thresholding (NIHT) is a simple yet efficient hard thresholding algorithm for matrix completion which considers the same non-convex problem as IHT for matrix completion

$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2} \|\mathcal{A}(x) - y\|_F^2 \quad \text{subject to} \quad \text{rank}(X) \leq r, \tag{2.1}$$

where $y = \mathcal{A}(X^o)$ with $\text{rank}(X^o) \leq r$.

Chapter 2 is outlined as follows[1]. In Sec. 2.1, we present NIHT for problem (2.1). In Sec. 2.2, we evaluate the efficacy of NIHT in terms of the sizes of problems that can be recovered, and compare it with other selected algorithms. Section 2.3 presents the proof of the main theorem in this chapter.

2.1 Normalized iterative hard thresholding

The simplest hard thresholding algorithm for matrix completion is IHT, Alg. 3, where the stepsize α is selected as a fixed constant (such as $\alpha = 0.65$). IHT for matrix completion is a direct extension of IHT for compressed sensing, Alg. 1. The effectiveness

[1]Material in this chapter is published in [91], which is a joint authorship with J. Tanner whose permission has been obtained for the inclusion of the material.

of IHT for compressed sensing is determined by the selection of the stepsize α. Selecting α too small causes the algorithm to be both slow and more likely to converge to local minima rather than the global minimum, whereas selecting α too large can result in lack of convergence. Large scale empirical testing [72] of IHT for compressed sensing suggested $\alpha = 0.65$ is a good choice. Even more effective than the constant stepsize $\alpha = 0.65$ is selecting the stepsize adaptively. It has been observed that IHT for compressed sensing performs dramatically better if the stepsize is selected to be optimal when the current estimate has the same support set as the sparsest solution [20]; with this stepsize selection rule the method is referred to as NIHT for compressed sensing, Alg. 2. In this chapter we present heuristics for the iteration dependent selection of the stepsize α in Alg. 3, motivated by NIHT for compressed sensing; we also refer to this best performing heuristic simply as NIHT, Alg. 5, with its matrix completion context making clear its distinction from NIHT for compressed sensing.

Algorithm 5 NIHT for matrix completion

Initilization: rank r, $W_{-1} = \mathcal{A}^*(y)$, $U_0 = \texttt{PrincipalLeftSingularVectors}_r(W_{-1})$
$X_0 = \texttt{Proj}_{U_0}(W_{-1})$, $l = 1$

Iteration: During iteration l, **do**

1: $R_{l-1} = \mathcal{A}^*\left(y - \mathcal{A}(X_{l-1})\right)$ — gradient descent direction

2: $\alpha_{l-1} = \dfrac{\left\|\texttt{Proj}_{U_{l-1}}(R_{l-1})\right\|_F^2}{\left\|\mathcal{A}\left(\texttt{Proj}_{U_{l-1}}(R_{l-1})\right)\right\|_2^2}$ — local steepest descent stepsize

3: $W_{l-1} = X_{l-1} + \alpha_{l-1} R_{l-1}$ — gradient descent update

4: $U_l = \texttt{PrincipalLeftSingularVectors}_r(W_{l-1})$ — proxy to the subspace

5: $X_l = \texttt{Proj}_{U_l}(W_{l-1})$ — solution update

Output: X_l

2.1.1 Stepsize selection heuristic

When IHT converges to the minimal rank solution, each of the singular values and vectors of the current estimate X_l must also converge to the singular values and vectors of the minimum rank solution. Proximity to the correct singular vectors motivates selecting the stepsize as if the singular vectors had been correctly identified and the update was being used to improve the component in the subspace. Let X_l be the current estimate of rank r with the singular value decomposition $X_l = U_l \Sigma_l V_l^*$. Then for any matrix $X \in \mathbb{R}^{m \times n}$, the projection of X onto the leading r left and right singular vector spaces can be computed by

$$\texttt{Proj}_{U_l}(X) = U_l U_l^* X \tag{2.2}$$

and

$$\texttt{Proj}_{V_l}(X) = XV_lV_l^* \tag{2.3}$$

respectively. A matrix can be projected onto the span of both the left and right singular vectors by applying (2.2) and (2.3) simultaneously; for instance, if $R_l = \mathcal{A}^*(y - \mathcal{A}(X_l))$ is the gradient descent direction, then the projected gradient descent direction is given by $R_l^{uv} := U_lU_l^*R_lV_lV_l^*$. Alternatively, R_l can be projected onto the span of just the left or right singular vectors by applying only (2.2) or (2.3) respectively, with the projected search directions given by $R_l^u := U_lU_l^*R_l$ and $R_l^v := R_lV_lV_l^*$. Using any of these restricted search directions to update the current iterate X_l results in the next iterate also being restricted to the same projected spaces, which would not allow the iterates to converge to the desired low rank matrix unless the singular vectors had been correctly identified. The analogue in compressed sensing would be to only update the solution restricted to the support set of the past iterate. Although the projected search directions should not be used as the update directions, they provide useful information for selecting the stepsizes.

By selecting the steepest descent stepsizes along the projected directions, the three above mentioned projected gradient descent directions motivate three locally optimal stepsizes

$$\alpha_l^u := \frac{\left\|\texttt{Proj}_{U_l}(R_l)\right\|_F^2}{\left\|\mathcal{A}\left(\texttt{Proj}_{U_l}(R_l)\right)\right\|_2^2} \tag{2.4}$$

$$\alpha_l^v := \frac{\left\|\texttt{Proj}_{V_l}(R_l)\right\|_F^2}{\left\|\mathcal{A}\left(\texttt{Proj}_{V_l}(R_l)\right)\right\|_2^2} \tag{2.5}$$

$$\alpha_l^{uv} := \frac{\left\|\texttt{Proj}_{V_l}\left(\texttt{Proj}_{U_l}(R_l)\right)\right\|_F^2}{\left\|\mathcal{A}\left(\texttt{Proj}_{V_l}\left(\texttt{Proj}_{U_l}(R_l)\right)\right)\right\|_2^2}. \tag{2.6}$$

and three associated variants of NIHT, each of which uses the same unrestricted negative gradient search direction R_l, but with three different stepsize heuristics (2.4), (2.5) and (2.6). We refer to the three variants of NIHT associated with these stepsizes as: NIHT with column restriction when using (2.4), NIHT with row restriction when using (2.5), and NIHT with column and row restriction when using (2.6). NIHT with row restriction is observed to have similar empirical performance to NIHT with column restriction; and both of them are observed to be superior to NIHT with column and row restriction. Thus we only focus on NIHT with column restriction which we simply refer to as NIHT, see Alg. 5 for a formal description. In each iteration of NIHT,

the current estimate X_{l-1} is updated along the gradient descent direction R_{l-1}, using an adaptive stepsize calculated to be exact for a restricted subspace, followed by thresholding to the matrix of rank at most r that is nearest in the Frobenius norm.

Although Alg. 5 lacks a safeguard to ensure a decrease of $\|y - \mathcal{A}(X_l)\|_2$ in each iteration, each of the stepsizes in (2.4) - (2.6) can be bounded above and below for sensing operators $\mathcal{A}(\cdot)$ with bounded RICs (1.9). Thus NIHT can be proven to converge to the desired low rank matrix using a standard RIC based proof reminiscent of [19, 20, 52, 60]. Theorem 2.1 considers a typical model $y = \mathcal{A}(X^o) + e$, where $\text{rank}(X^o) \leq r$. In this model, X^o can be viewed as the rank r matrix which minimizes $\|y - \mathcal{A}(X)\|_2$ and e as the measurement error associated with restricting X to be at most rank r. Alternatively X^o can be viewed as a rank r matrix measured by $\mathcal{A}(\cdot)$ and that y is contaminated by additive noise; though, in this perspective, Thm. 2.1 does not specify any particular model for e.

Theorem 2.1. Let $\mathcal{A}(\cdot)$ be a linear map from $\mathbb{R}^{m\times n}$ with $p < mn$, and $y = \mathcal{A}(X^o)+e$ for any X^o with $\text{rank}(X^o) \leq r$. Define the following constant in terms of RICs of $\mathcal{A}(\cdot)$[2]

$$\mu = \frac{4\delta_{3r}}{1-\delta_r},\ \xi = \frac{2\left(1+\delta_{2r}\right)^{1/2}}{1-\delta_r}. \tag{2.7}$$

Then provided $\mu < 1$, the iterates of Alg. 5 satisfy

$$\|X_l - X^o\|_F \leq \mu^l \|X_0 - X^o\|_F + \frac{\xi}{1-\mu}\|e\|_2. \tag{2.8}$$

In particular, when $e = 0$, NIHT can recovery (within arbitrary precision) any rank r matrix measured by $\mathcal{A}(\cdot)$.

Proof. See Sec. 2.3 for a proof. □

2.2 Empirical performance comparisons

RIC based guarantees such as Thm 2.1 are uniform over all rank r matrices and are now common for matrix completion algorithms. The uniform nature of these results gives sufficient conditions for recovery. However, such theorems lack the quantitative precision sufficient to advise a practitioner as to how many linear measurements are necessary so that an algorithm is able to successfully recover a low rank matrix [10].

[2]In Thms. 2.1 and 4.1, the restricted isometry constant $\delta_{cr}(\mathcal{A})$ of the sensing ensemble $\mathcal{A}(\cdot)$ is abbreviated to δ_{cr} for simplicity.

For a matrix $X^o \in \mathbb{R}^{m\times n}$ of rank r, obviously it can be easily identified if mn linear measurements are obtained since the sensing process in invertible. On the other hand, an $m \times n$ rank r matrix lies in a manifold defined by $r(m+n-r)$ degrees of freedom [24]. If the subspace containing X^o is known *a priori*, X^o can be recovered by making appropriate measurements in the known subspace with the number of measurements equal to the number of degrees of freedom. Therefore the matrix completion problem inherently defines an *undersampling ratio* δ and an *oversampling ratio* ρ by comparing the number of measurements p with the maximum number of measurements required, mn, and the minimum number of measurements necessary, $r(m+n-r)$:

$$\delta = \frac{p}{mn} \quad \text{and} \quad \rho = \frac{r(m+n-r)}{p}. \tag{2.9}$$

The unit square $(\delta, \rho) \in [0,1]^2$ defines a *phase space* which indicates where the number of measurements is sufficient for the successful recovery of a low rank matrix to be possible when fewer measurements are taken than the ambient dimension. Moreover the *recovery phase transition curve* for an algorithm separates the phase space into two regions: success and failure. For problem instances (δ, ρ) below the recovery phase transition curve, the algorithm is observed to return a low rank matrix matching the desired solution. On the other hand, above the curve, the algorithm is observed to return an approximation which does not match the underlying low rank matrix. For instance, much more interestingly than Thm. 2.1, the recovery phase transitions for NIHT suggest it is able to recover random square matrices for nearly the largest achievable rank; that is, ρ near one[3], see Fig. 2.2. For Gaussian sensing the phase transition curve for NIHT appears to oscillate with ρ between 0.85 and 0.95 independent of δ and for entry sensing to slowly vary from about $\rho = 0.9$ at $\delta = 0.1$ to $\rho = 1$ at $\delta = 1$. Note that $\rho = 1$ corresponds to achieving the minimum sampling number of $p = r(m+n-r)$.

2.2.1 Experimental set-up

In this section, we present empirical recovery phase transition curves and investigate the rate of recovery for NIHT. The testing was conducted through a massive distribution of problem instances at high performance computing facility provided by the Edinburgh Compute and Data Facility (ECDF). Recall that the sensing operator in

[3]Note that guaranteed recovery for all rank r matrices is not possible for $\rho > 1/2$, but that algorithms can recovery most low rank matrices for $\rho > 1/2$ [45].

matrix completion is a linear map $\mathcal{A}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ where $y = \mathcal{A}(X)$ has each measurement defined by Frobenius norm inner product of a sensing matrix $A_\ell \in \mathbb{R}^{m\times n}$ and the matrix X : $y_\ell = \langle A_\ell, X\rangle$. The algorithms are tested with two representative sensing operator ensembles:

- $\mathcal{G}$: each of the p sensing matrices is a dense Gaussian matrix with entries drawn i.i.d from $\mathcal{N}(0, 1/mn)$;
- $\mathcal{E}$: each of the p sensing matrices only has one single nonzero entry of value 1 with the location chosen uniformly at random without replacement.

The sensing ensemble $\mathcal{G}$ is representative of dense sensing while $\mathcal{E}$ is an entry sensing operator. In the implementation, the sensing operator $\mathcal{E}$ is employed by directly measuring p distinct entries of the sensing matrix rather than through a series of inner products. To form a rank r testing matrix X^o, we compute the product of two random rank r matrices via

$$X^o = YZ \tag{2.10}$$

where $Y \in \mathbb{R}^{m\times r}$ and $Z \in \mathbb{R}^{r\times n}$ with their entries drawn i.i.d from standard normal distribution $\mathcal{N}(0, 1)$. In our tests, a problem instance is indexed by four parameters in Tab. 2.1.

Table 2.1: Parameters for problem instances in matrix completion.

(Op, N)	$Op \in \{\mathcal{G}, \mathcal{E}\}$ represents the sensing operator ensemble; N represents the low rank matrix ensemble formed by (2.10).
(m, n) or (γ, n)	(m, n) denotes the size of the testing matrix; $\gamma := \frac{m}{n}$ defines the ratio of m over n.
p or δ	p denotes the number of measurements, which is also determined by the undersampling ratio δ in (2.9).
r or ρ	r denotes the rank of a testing matrix, which is also determined by the oversampling ratio ρ in (2.9).

Moreover, the iterative algorithms tested terminate if one of the following conditions is satisfied: a maximum number of iterations is met, the relative residual is small $\|y - \mathcal{A}(X_l)\|_2 / \|y\|_2 < 10^{-5}$ or if the multiplicative average of the last fifteen convergence factors is greater than κ,

$$\left(\frac{\|y - \mathcal{A}(X_{l+15})\|_2}{\|y - \mathcal{A}(X_l)\|_2}\right)^{1/15} > \kappa. \tag{2.11}$$

Unless otherwise stated we set $\kappa = 0.999$.

2.2.2 NIHT empirical average case behavior

To evaluate NIHT's ability to recover $m \times n$ matrices of rank r, we conducted tests for problem instances with $Op \in \{\mathcal{G}, \mathcal{E}\}$ and $\gamma = 1/4, 1/2, 3/4$ and 1. The reconstruction algorithms are substantially faster for entry sensing, as dense sensing has an additional requirement of pmn scalar multiplications for the application of $\mathcal{A}(\cdot)$. For this reason the tests for dense Gaussian sensing is conducted for $n = 40$ and $n = 80$, whereas the tests for entry sensing are conducted for $n = 40, 80, 160, 200, 400$, and 800. For each (m, n) pair, we conduct tests with the number of measurements being $p = \delta \cdot mn$ for

$$\delta \in \{0.1, 0.2, \cdots, 0.9, 1\}.$$

We consider an algorithm to have successfully recovered the sensed rank r matrix X^o if it returns a rank r matrix $\hat{X}$ that is within 2×10^{-3} of the sensed matrix in the relative Frobenius norm[4], that is if

$$\frac{\left\|\hat{X} - X^o\right\|_F}{\left\|X^o\right\|_F} \leq 2 \times 10^{-3}. \tag{2.12}$$

For each triple (m, n, p), we select a value of r sufficiently small that the algorithm successfully recovers the sensed matrix in each of ten randomly drawn tests; we then increase the rank of the sensed matrices, conducting ten tests per rank, until the rank is sufficiently large that the algorithm fails to recover the sensed matrix in each of ten randomly drawn tests. We refer to the largest value of r for which the algorithm succeeded in each of ten tests as r_{min} and the smallest value of r for which the algorithm failed in each of the ten tests at r_{max}.

The values of r_{min} and r_{max} for NIHT, are displayed in Fig. 2.1 for Gaussian sensing with $n = 80$ (left panel) and entry sensing with $n = 800$ (right panel). The same data is displayed in Fig. 2.2, but with the vertical axis being the values of ρ calculated from $(r_{min} + r_{max})/2$. The exact values of r_{min}, r_{max} and associated ρ_{min}, ρ_{max} calculated from (2.9), as well as for smaller n, are listed in Tabs. 2.2 and 2.3 for Gaussian sensing and entry sensing respectively.

The remarkable effectiveness of NIHT is evident in Fig. 2.2 through the approximate phase transition, calculated using $(r_{min} + r_{max})/2$. Though the phase transition

[4]Greater accuracy conditions were tested for a subset of problems. In each instance it was observed that recovery within arbitrary precision was possible once a matrix was identified to within 2×10^{-3} relative error in the Frobenius norm.

for Gaussian sensing displayed in Fig. 2.2 (left panel) appears irregular due to the small value of $n = 80$ and associated large changes in ρ for a rank one change, it is surprising that the phase transitions remain between 0.85 and 0.95 for each of $\gamma = 1, 3/4, 1/2$, and $1/4$, as well as for each δ tested. The phase transition occurring between 0.85 and 0.95 indicates that irrespective of the degree of undersampling, δ, NIHT is able to recover rank r matrices for the number of measurements being a small multiple of the minimum number; that is $p = C \cdot r(m + n - r)$ for $C < 1.2$. NIHT exhibits a similarly high phase transition for entry sensing in Fig. 2.2 (right panel). The entry sensing tests for the larger value of $n = 800$ greatly reduces the sensitivity of ρ to small changes in r, resulting in much smoother observed phase transitions. For $\gamma = 1, 3/4$, and $1/2$ we observe phase transitions that slowly increasing from approximately $\rho = 0.9$ to 1 as δ increases from 0.2 to 1. The smaller value of $\gamma = 1/4$ show substantially reduced phase transitions for entry sensing, but in Sec. 2.2.4, we will observe that even this lower phase transition compares favourably to other matrix completion algorithms. The data in Tab. 2.2 for $m = n$ and $p = mn/10$ suggest that the lower phase transition for $\delta = 1/10$ may be an artifact of the small problem sizes. Similarly, we observe that increasing the problem size from $n = 80$ to 800 results in substantial increases in the phase transitions show in Fig. 2.5, including for $\gamma = 1/4$.

The stopping criteria (2.11) for the convergence facotr plays an important role in achieving the observed high phase transitions; setting κ to the very slow 0.995 is observed to noticeably reduce the largest recoverable rank as compared to the seemingly impractically slow $\kappa = 0.999$. Figure 2.4 shows the increase in the phase transitions when κ is increased from 0.995 (blue) to 0.999 (black). Figure 2.3 shows that this increase in κ allows for some of the tests that terminate with error greater than 2×10^{-3} to further iterate and decrease the error sufficiently for the relative error stopping criteria $\|b - \mathcal{A}(X^j)\|_2 / \|b\|_2 < 10^{-5}$ to be active. This raise in the phase transition by increasing κ comes at the cost of the maximum number of iterations increasing from approximately 1500 to 6500, and the associated time needed for completion. Although this, nearly four-fold, increase in the time to completion may seem inadvisable, no other algorithm was previously observed to be able to recover low rank matrices for such large ranks. The other algorithms tested are unable to increase the recoverable rank by extending the stopping criteria, see for instance the right panel of Fig. 2.3 where PF (Alg. 4) is observed to have clear separation between

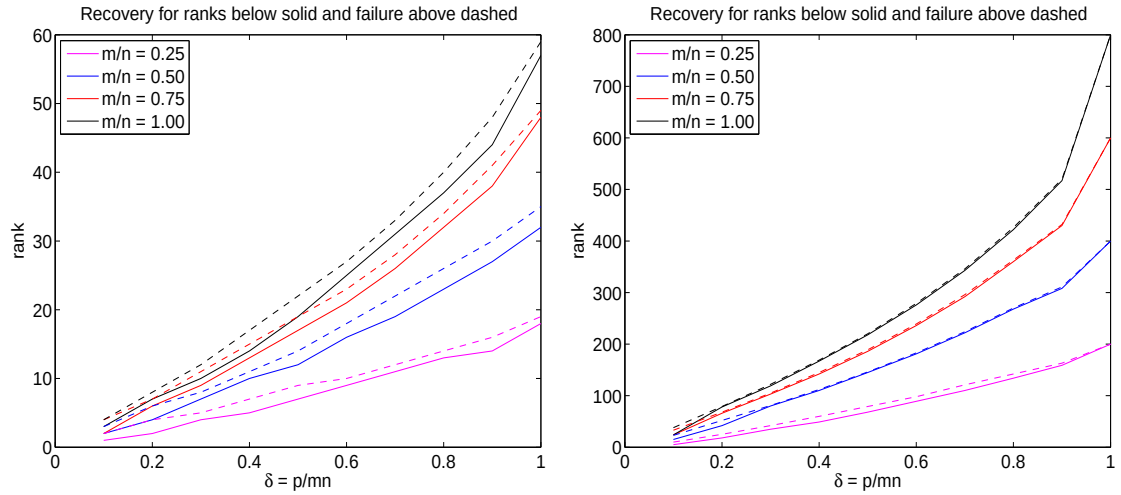


Figure 2.1: For each rank above the dashed lines NIHT failed to recover the sensed matrix in each of ten randomly drawn tests per rank, whereas for each rank below the solid lines NIHT succeeded in recovering the sensed matrix in each of ten randomly drawn tests per rank. The vertical axis is the rank of the sensed matrix and the horizontal axis is the undersampling ratio δ. Left panel: $\mathcal{G}$ with $n = 80$; right panel: $\mathcal{E}$ with $n = 800$.

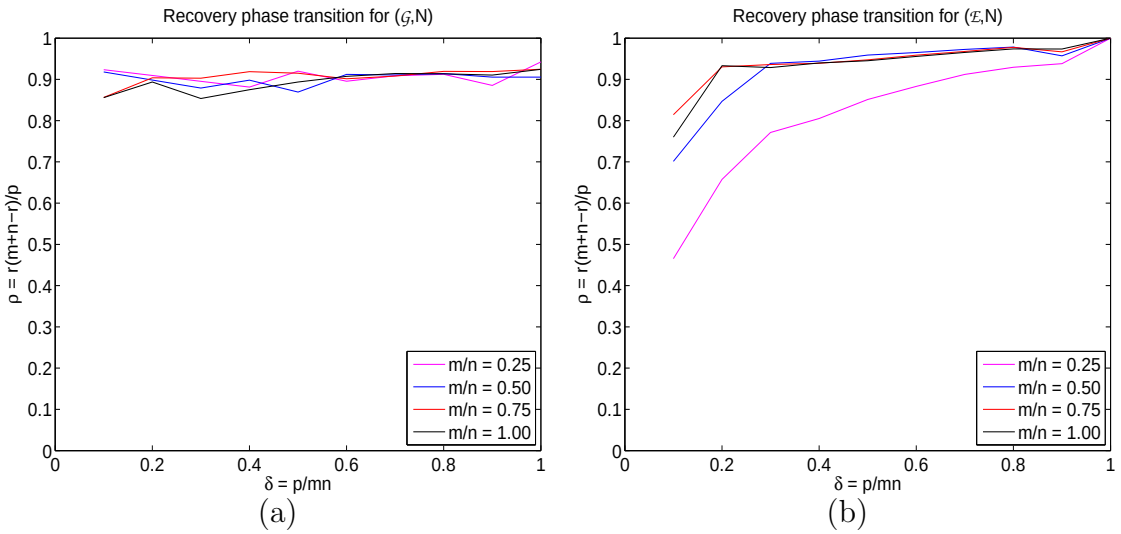


Figure 2.2: Phase transition of NIHT with horizontal axis δ and vertical axis ρ where ρ is calculated using the average of r_{min} and r_{max} from Tabs. 2.2 and 2.3. Left panel: $\mathcal{G}$ with $n = 80$; right panel: $\mathcal{E}$ with $n = 800$.

Table 2.2: For each listed (m, n, p) triple, NIHT with p Gaussian measurements is tested for ten randomly drawn $m \times n$ rank r matrices per rank and is observed to recover each of the ten measured matrices for all $r \leq r_{min}$ and failed to recover each of the ten measured matrices per rank for $r \geq r_{max}$.

NIHT with Gaussian Measurements													
m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}	m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}
10	40	40	0	0	0.000	0.000	20	80	160	1	2	0.619	1.225
		80	1	2	0.613	1.200			320	2	4	0.613	1.200
		120	2	3	0.800	1.175			480	4	5	0.800	0.990
		160	2	4	0.600	1.150			640	5	7	0.742	1.017
		200	3	5	0.705	1.125			800	7	9	0.814	1.024
		240	4	5	0.767	0.938			960	9	10	0.853	0.938
		280	5	7	0.804	1.075			1120	11	12	0.874	0.943
		320	6	7	0.825	0.941			1280	13	14	0.884	0.941
		360	7	9	0.836	1.025			1440	14	16	0.836	0.933
		400	8	10	0.840	1.000			1600	18	19	0.922	0.962
20	40	80	1	2	0.738	1.450	40	80	320	2	3	0.738	1.097
		160	2	3	0.725	1.069			640	4	6	0.725	1.069
		240	3	5	0.713	1.146			960	7	8	0.824	0.933
		320	4	6	0.700	1.012			1280	10	11	0.859	0.937
		400	6	7	0.810	0.927			1600	12	14	0.810	0.927
		480	8	9	0.867	0.956			1920	16	18	0.867	0.956
		560	9	11	0.820	0.963			2240	19	22	0.857	0.963
		640	11	13	0.842	0.955			2560	23	26	0.871	0.955
		720	13	15	0.849	0.938			2880	27	30	0.872	0.938
		800	16	18	0.880	0.945			3200	32	35	0.880	0.930
30	40	120	1	2	0.575	1.133	60	80	480	2	4	0.575	1.133
		240	3	4	0.838	1.100			960	6	7	0.838	0.970
		360	4	6	0.733	1.067			1440	9	11	0.819	0.985
		480	6	8	0.800	1.033			1920	13	15	0.860	0.977
		600	7	10	0.735	1.000			2400	17	19	0.871	0.958
		720	11	12	0.901	0.967			2880	21	23	0.868	0.934
		840	12	15	0.829	0.982			3360	26	28	0.882	0.933
		960	15	17	0.859	0.939			3840	32	34	0.900	0.939
		1080	18	21	0.867	0.953			4320	38	41	0.897	0.940
		1200	23	25	0.901	0.938			4800	48	49	0.920	0.929
40	40	160	1	2	0.494	0.975	80	80	640	3	4	0.736	0.975
		320	3	4	0.722	0.950			1280	7	8	0.837	0.950
		480	5	7	0.781	1.065			1920	10	12	0.781	0.925
		640	7	9	0.798	0.998			2560	14	17	0.798	0.950
		800	9	11	0.799	0.949			3200	19	22	0.837	0.949
		960	12	14	0.850	0.963			3840	25	27	0.879	0.935
		1120	15	17	0.871	0.956			4480	31	33	0.893	0.935
		1280	18	20	0.872	0.938			5120	37	40	0.889	0.938
		1440	21	24	0.860	0.933			5760	44	48	0.886	0.933
		1600	28	30	0.910	0.938			6400	57	59	0.917	0.931

Table 2.3: For each listed (m, n, p) triple, NIHT with p entry measurements is tested for ten randomly drawn $m \times n$ rank r matrices per rank and is observed to recover each of the ten measured matrices for all $r \leq r_{min}$ and failed to recover each of the ten measured matrices per rank for $r \geq r_{max}$.

NIHT with Entry Measurements																			
m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}	m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}	m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}
50	200	1000	0	2	0.000	0.496	100	400	4000	1	5	0.125	0.619	200	800	16000	5	10	0.311	0.619
		2000	1	5	0.124	0.613			8000	5	11	0.309	0.672			32000	18	25	0.552	0.762
		3000	3	9	0.247	0.723			12000	12	19	0.488	0.762			48000	35	42	0.704	0.838
		4000	6	14	0.366	0.826			16000	20	29	0.600	0.854			64000	49	60	0.728	0.881
		5000	10	19	0.480	0.878			20000	30	38	0.705	0.878			80000	68	79	0.792	0.909
		6000	16	23	0.624	0.870			24000	37	47	0.714	0.887			96000	89	98	0.845	0.921
		7000	20	29	0.657	0.916			28000	48	58	0.775	0.916			112000	110	121	0.874	0.950
		8000	27	34	0.753	0.918			32000	62	69	0.849	0.929			128000	134	142	0.907	0.952
		9000	34	39	0.816	0.914			36000	75	82	0.885	0.952			144000	159	163	0.929	0.947
		10000	50	50	1.000	1.000			40000	100	100	1.000	1.000			160000	200	200	1.000	1.000
100	200	2000	1	5	0.149	0.738	200	400	8000	4	10	0.298	0.738	400	800	32000	15	23	0.555	0.846
		4000	6	12	0.441	0.864			16000	18	25	0.655	0.898			64000	42	52	0.760	0.933
		6000	12	19	0.576	0.890			24000	32	40	0.757	0.933			96000	80	81	0.933	0.944
		8000	22	28	0.764	0.952			32000	47	56	0.812	0.952			128000	110	112	0.937	0.952
		10000	28	36	0.762	0.950			40000	65	73	0.869	0.962			160000	145	146	0.956	0.962
		12000	38	46	0.830	0.974			48000	84	92	0.903	0.974			192000	181	183	0.961	0.969
		14000	51	56	0.907	0.976			56000	107	112	0.942	0.976			224000	222	224	0.969	0.976
		16000	59	67	0.889	0.976			64000	130	135	0.955	0.981			256000	268	270	0.976	0.981
		18000	71	75	0.903	0.938			72000	154	156	0.954	0.962			288000	308	311	0.954	0.960
		20000	100	100	1.000	1.000			80000	200	200	1.000	1.000			320000	400	400	1.000	1.000
150	200	3000	2	8	0.232	0.912	300	400	12000	5	16	0.290	0.912	600	800	48000	24	33	0.688	0.940
		6000	11	17	0.622	0.944			24000	30	34	0.838	0.944			96000	66	68	0.917	0.944
		9000	22	26	0.802	0.936			36000	49	53	0.886	0.953			144000	103	105	0.928	0.944
		12000	32	37	0.848	0.965			48000	69	73	0.907	0.954			192000	142	145	0.930	0.948
		15000	45	48	0.915	0.966			60000	92	95	0.932	0.958			240000	186	189	0.941	0.954
		18000	57	60	0.928	0.967			72000	117	120	0.947	0.967			288000	236	239	0.954	0.963
		21000	71	75	0.943	0.982			84000	145	148	0.958	0.973			336000	292	296	0.963	0.973
		24000	84	91	0.931	0.982			96000	179	182	0.971	0.982			384000	360	363	0.975	0.980
		27000	99	103	0.920	0.942			108000	214	217	0.963	0.970			432000	430	432	0.966	0.968
		30000	150	150	1.000	1.000			120000	300	300	1.000	1.000			480000	600	600	1.000	1.000
200	200	4000	1	9	0.100	0.880	400	400	16000	7	19	0.347	0.927	800	800	64000	24	38	0.591	0.927
		8000	9	20	0.440	0.950			32000	36	40	0.860	0.950			128000	78	79	0.927	0.939
		12000	27	31	0.839	0.953			48000	58	62	0.897	0.953			192000	119	122	0.918	0.939
		16000	39	43	0.880	0.959			64000	81	85	0.910	0.950			256000	167	169	0.935	0.945
		20000	52	56	0.905	0.963			80000	108	111	0.934	0.956			320000	218	220	0.941	0.949
		24000	66	71	0.918	0.973			96000	137	140	0.946	0.963			384000	276	279	0.952	0.960
		28000	84	88	0.948	0.981			112000	170	174	0.956	0.973			448000	343	346	0.962	0.968
		32000	101	107	0.944	0.980			128000	210	214	0.968	0.980			512000	422	426	0.971	0.977
		36000	119	124	0.929	0.951			144000	257	260	0.969	0.975			576000	517	520	0.972	0.975
		40000	200	200	1.000	1.000			160000	400	400	1.000	1.000			640000	800	800	1.000	1.000

problem instances where the algorithm converges to the sensed matrix and when it returns a low rank matrix that differs substantially from the measured matrix.

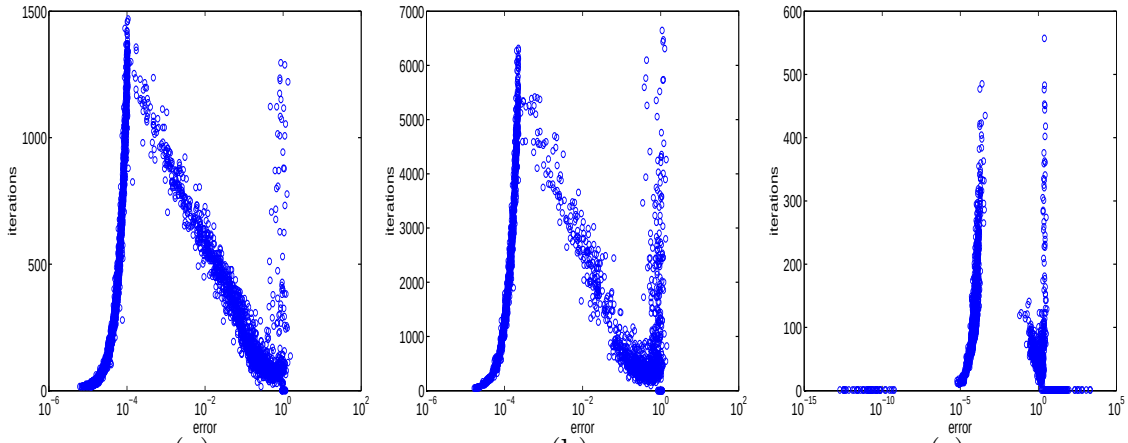


Figure 2.3: Iteration vs error for NIHT with Gaussian sensing using convergence factor stopping criteria 0.995 and 0.999 in the left and middle panels respectively, and for PF in the right panel. The plots include the results for each of the tests conducted for $n = 40$ and 80, comprising $10,320$ tests in the left panel, $2,500$ tests for the middle panel, and $2,205$ tests in the right panel.

Matrix completion is a rapidly evolving field with numerous algorithms introduced in the last few years. A recent review of matrix completion algorithms is presented in [73] along with extensive numerical comparisons. The focus of the numerical tests presented here is on quantifying the largest possible rank recoverable for an algorithm as a function of $\delta = p/mn$, and to explore if this phase transition is stable as the matrix size increases. This testing environment probes the algorithms in the region where their convergence rates are slow, causing the tests to have an unusually high computational burden. The tests presented in this chapter required 4.67 CPU years[5] run on hex core Intel X5650 CPUs with 24 GB of RAM.

The empirical behavior of NIHT has been presented in Sec. 2.2.2. Next we contrast the performance of NIHT with: fixed stepsize IHT ($\alpha = 0.65$), NNM (1.14) using a semidefinite programming formulation [85] and the software package SDPT3 [93], and PF. These three algorithms are selected as representative examples of hard thresholding algorithms, IHT, convex relaxations, NNM, and a quite distinct variant of optimization on the manifold of low rank matrices, PF. PF was selected due to

[5]All algorithms were written using Matlab R2011b with the default SVDs, as opposed to the potentially faster PROPACK [65].

both its simplicity and its remarkably high phase transition [57]. The primary focus of this comparison is to determine the largest rank recoverable by an algorithm for a given (m, n, p) triple. Timings are included for completeness. Each algorithm is tested as described in the first paragraph of Sec. 2.2.2.

2.2.3 Comparison of IHT and NIHT

We consider IHT with fixed stepsize $\alpha = 0.65$ as well as three variants of NIHT with iteration adaptive stepsizes as stated in (2.4) - (2.6). The RIC based analysis of NIHT with stepsize (2.4) presented in Sec. 2.3 is equally valid for stepsizes (2.5) and (2.6). Unfortunately, this worst case analysis gives little indication of their average case effectiveness. Tests were conducted for matrices with aspect ratio $\gamma = 1, 3/4, 1/2, 1/4,$ and $1/8$ for $n = 40$ and $n = 80$. The variant with stepsize (2.5) was able to recover matrices of the same rank as (2.4) when $\gamma = 1, 3/4,$ and $1/2$, but was only able to recover substantially lower rank matrices for the more rectangular aspect ratios $\gamma = 1/4$ and $1/8$. The NIHT variant with stepsize (2.6) was able to only recover matrices of greatly reduced rank as compared to (2.4) and (2.5). For conciseness we limit ourselves to presenting only the results for the adaptive stepsize variant (2.4), which we simply refer to as NIHT and state in Alg. 5.

Figures 2.4 and 2.5 display the empirical phase transition with ρ calculated using the average of r_{min} and r_{max} as described in Sec. 2.2.2. Figure 2.4 displays the phase transition with Gaussian sensing for NIHT with stopping criteria $\kappa = 0.999$ (black) and 0.995 (blue) as well as IHT with fixed stepsize $\alpha = 0.65$ (red) and $\kappa = 0.999$. Increasing κ is observed to substantially increase the recoverable rank. For the same stopping criteria, NIHT is observed to recover rank r matrices for r at least as high as IHT. The average timings over five random tests for the associated tests with $\kappa = 0.999$ and $m = n = 40$ are displayed in Fig. 2.8 Panels (b) and (d) for NIHT and IHT respectively, and the ratio of their average timings is displayed in Fig. 2.9 Panel (b). NIHT is observed to be faster than IHT except for the region of large δ and ρ where IHT fails to recover the solution to (2.1) but NIHT remains able to recover the solution to (2.1).

Figure 2.5 displays the phase transition for entry sensing and NIHT with $n = 800$ (blue) and $n = 80$ (black) as well as IHT with stepsize $\alpha = 0.65$ and $n = 80$ (red). NIHT and IHT are observed to be able to recover rank r matrices for nearly the same rank with $n = 80$. Increasing the matrix size from $n = 80$ to $n = 800$ results in a

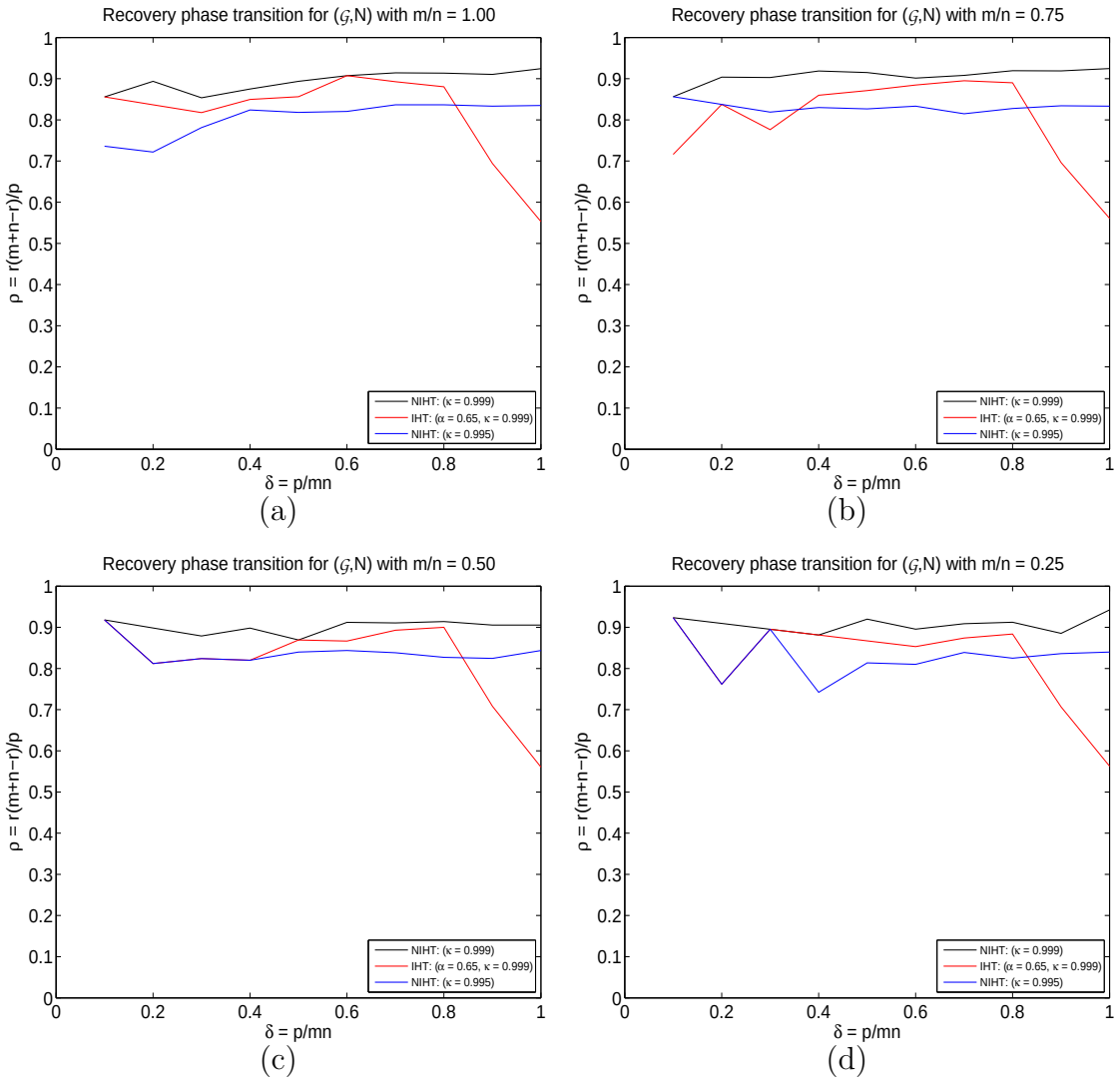


Figure 2.4: Phase transition for Gaussian sensing and algorithms: NIHT with stopping criteria $\kappa = 0.999$ and $\kappa = 0.995$ as well as IHT with stepsize $\alpha = 0.65$ and stopping criteria $\kappa = 0.999$. Horizontal axis δ and vertical axis ρ as defined in (2.9). Each transition is for $n = 80$ and the values of ρ shown are calculated using the average of r_{min} and r_{max}.

dramatic increase in the phase transitions, with each curve surprisingly close to the maximum achievable $\rho = 1$. All entry sensing tests use stopping criteria $\kappa = 0.999$. The average timings for the associated tests with $n = m = 80$ are displayed in Fig. 2.8 Panels (a) and (c) for NIHT and IHT respectively, and the ratio of their average timings is displayed in Fig. 2.9 Panel (a). NIHT is observed to always be faster than IHT, typically taking just under half the time.

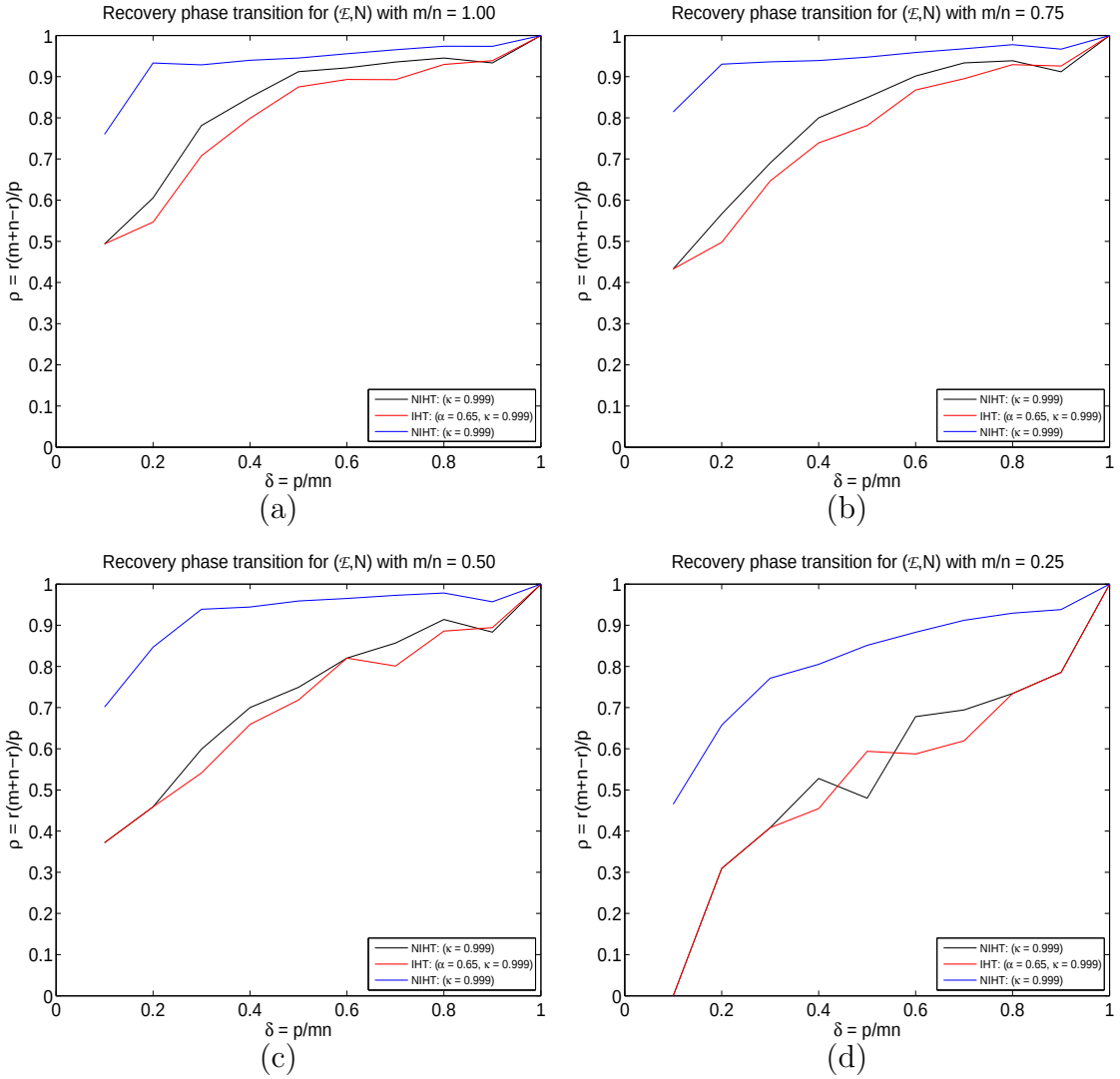


Figure 2.5: Phase transition for entry sensing and algorithms: NIHT with column projection for $n = 80$ (black) and $n = 800$ (blue) and IHT with stepsize $\alpha = 0.65$ (red) with $n = 80$. Horizontal axis δ and vertical axis ρ as defined in (2.9). The values of ρ shown are calculated using the average of r_{min} and r_{max}.

2.2.4 Comparison of NIHT with NNM and PF

Nuclear norm minimization (1.14), is the convex relaxation of the matrix completion problem (1.11), and is the most studied approach for matrix completion [29, 32, 85]. In particular, it is the only matrix completion formulation with a quantitatively accurate analysis [81] of the ability to recover the solution to (1.12). Having a formulation in terms of the well studied semidefinite programming, there are many existing algorithms and software packages that can be used to solve (1.14); moreover, numerous algorithms have been designed to solve NNM specifically for matrix completion, for example [25, 52]. These matrix completion focused methods for the solution of (1.14) are designed to more rapidly return the solution, but remain designed to give the solution to (1.14), and do not increase the range of the parameters δ and ρ where the solution to (1.14) corresponds to that of (1.11). With our focus of determining the largest recoverable rank for an algorithm, we use the well established software package SDPT3 [93], but are aware that specialized software is likely able to solve (1.14) in substantially less time. In addition to contrasting NIHT with NNM, we also compare NIHT with the very different manifold optimization method PF [57], see Alg. 4.

PF seeks to directly solve the minimum rank problem (1.11) and is a particularly simple example of a class of methods which are designed to remain on the manifold of rank r matrices throughout each iteration. In contrast, NIHT updates the solution with directions that result in intermediate updates which are not on the manifold of rank r matrices. Despite its simplicity, PF is capable of recovering matrices of surprisingly large rank.

Figure 2.6 displays the phase transition for Gaussian sensing and NIHT (black), PF (red), and NNM (blue). NIHT and PF use the stopping criteria $\kappa = 0.999$, and SDPT3 uses a tolerance based stopping criteria to solve NNM. In every instance PF is observed to be able to recover matrices of larger rank than can NNM. NIHT is observed to be able to recover even larger rank for all but the largest values of δ. The average timings over 5 random tests for NNM and PF with $m = n = 40$ are displayed in Fig. 2.8 Panels (f) and (h) respectively, and the ratio of their average timings as compared with NIHT is displayed in Fig. 2.9 Panels (d) and (f) for NNM and PF respectively. NIHT is observed to be faster than NNM for all but δ and ρ simultaneously large, where NIHT is observed to be extremely slow. NIHT is

observed to always be slower than PF for Gaussian sensing, typically three to seven times slower.

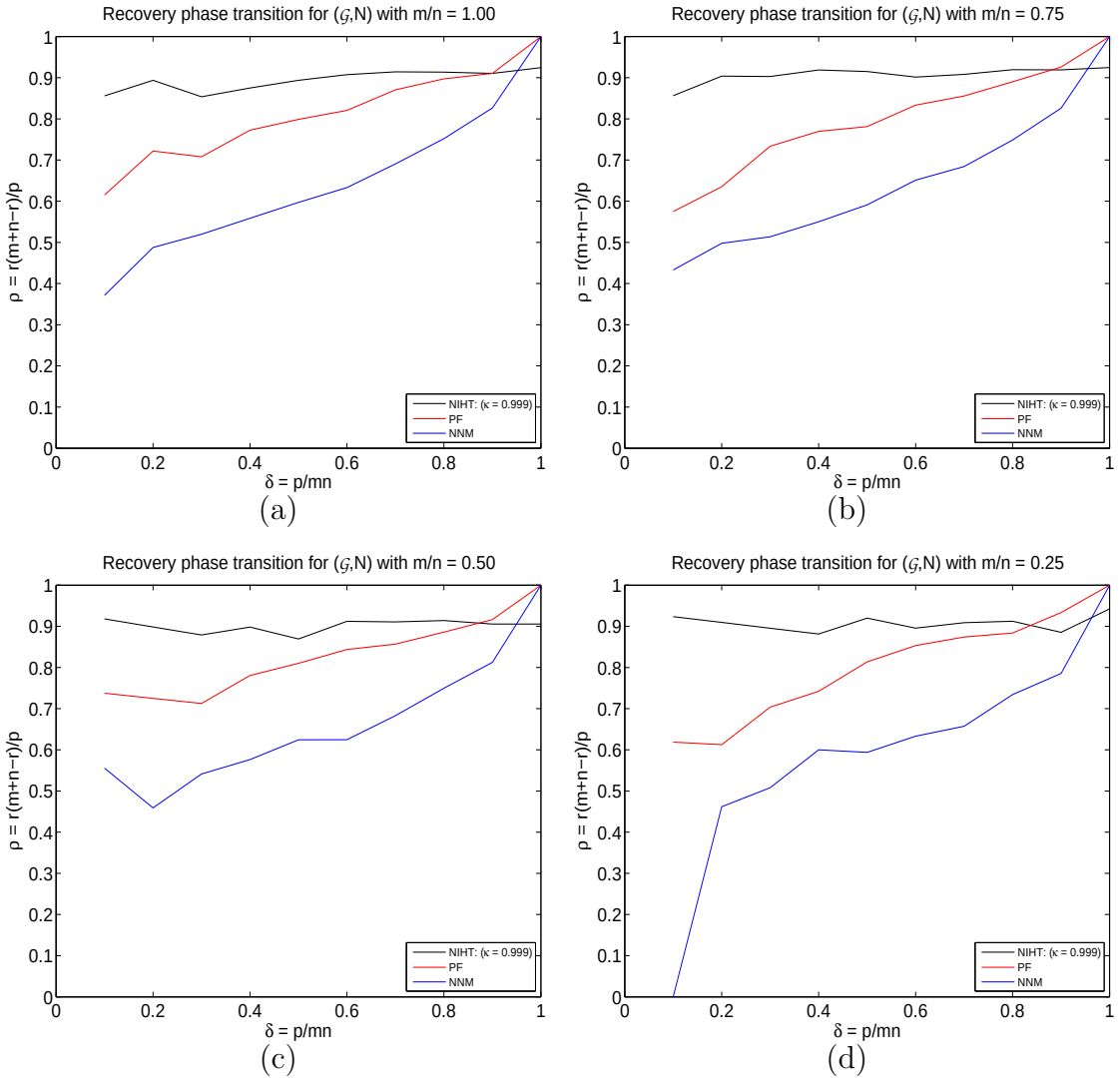


Figure 2.6: Phase transition for Gaussian sensing and algorithms: NIHT, PF and NNM, all with $n = 80$. Horizontal axis δ and vertical axis ρ as defined in (2.9). The values of ρ shown are calculated using the average of r_{min} and r_{max}.

Figure 2.7 displays the phase transition for entry sensing and NIHT (black), PF (red), and NNM (blue). Again, NIHT and PF use the stopping criteria $\kappa = 0.999$, and SDPT3 uses a tolerance based stopping criteria. Memory requirements limit the size of problems that NNM and PF are able to solve to $n = 80$. In every instance NIHT is observed to be able to recover matrices of a larger rank than can PF, which is able to recover matrices of larger rank than can NNM. The average timings for NNM and PF for $m = n = 80$ are displayed in Fig. 2.8 Panels (e) and (g) respectively, and the

ratio of their average timings as compared with NIHT is displayed in Fig. 2.9 Panels (c) and (e) for NNM and PF respectively. NIHT is observed to be faster than both NNM and PF, often more than ten times as fast; however, it should be noted that algorithms designed to solve (1.14) specifically for matrix completion can be expected to be substantially faster than that of SDPT3 and the use of iterative numerical linear algebra algorithms can be expected to accelerate PF.

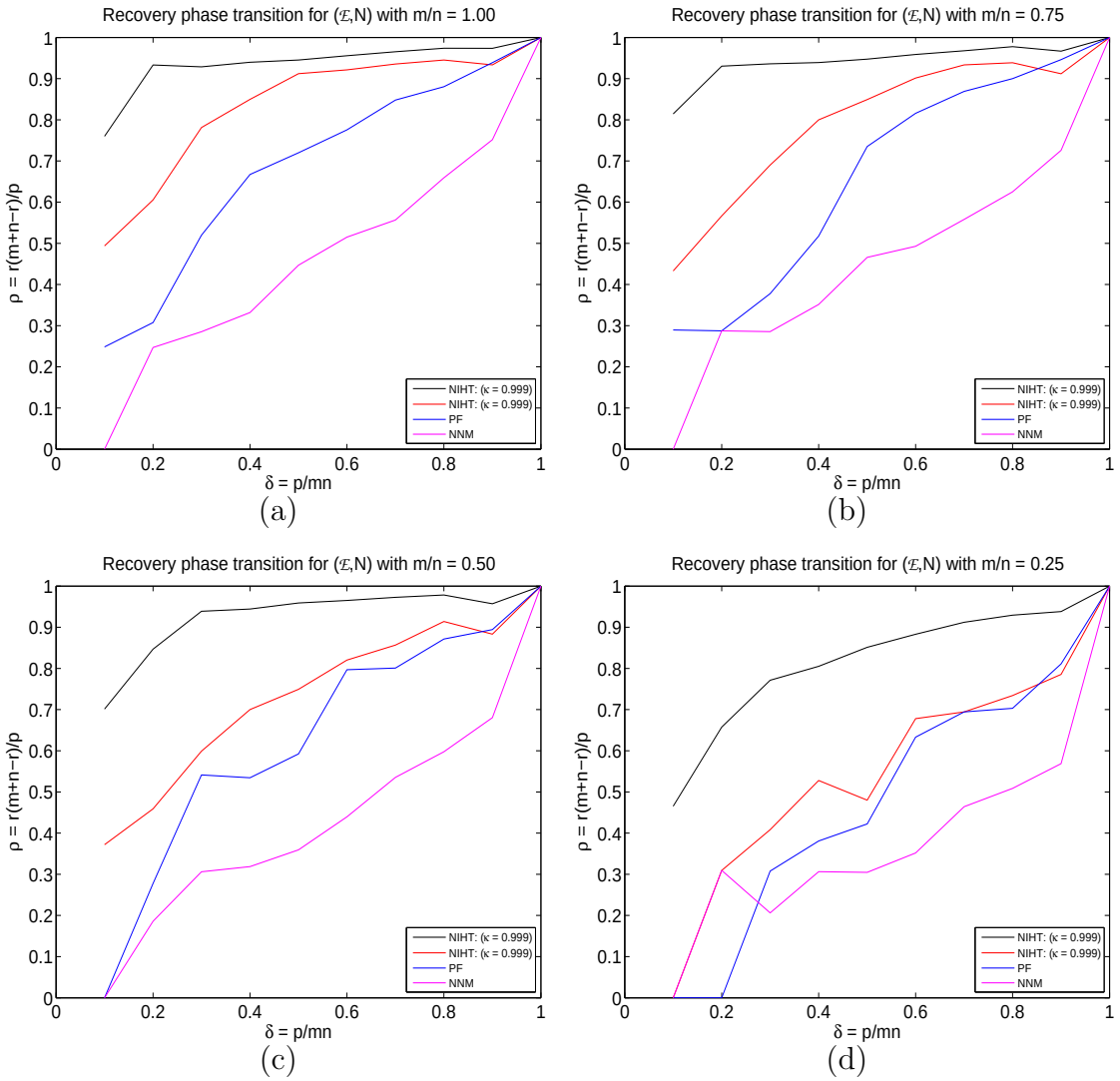


Figure 2.7: Phase transition for entry sensing and algorithms: NIHT with $n = 800$, and NNM and PF both with $n = 80$. Horizontal axis δ and vertical axis ρ as defined in (2.9). The values of ρ shown are calculated using the average of r_{min} and r_{max}.

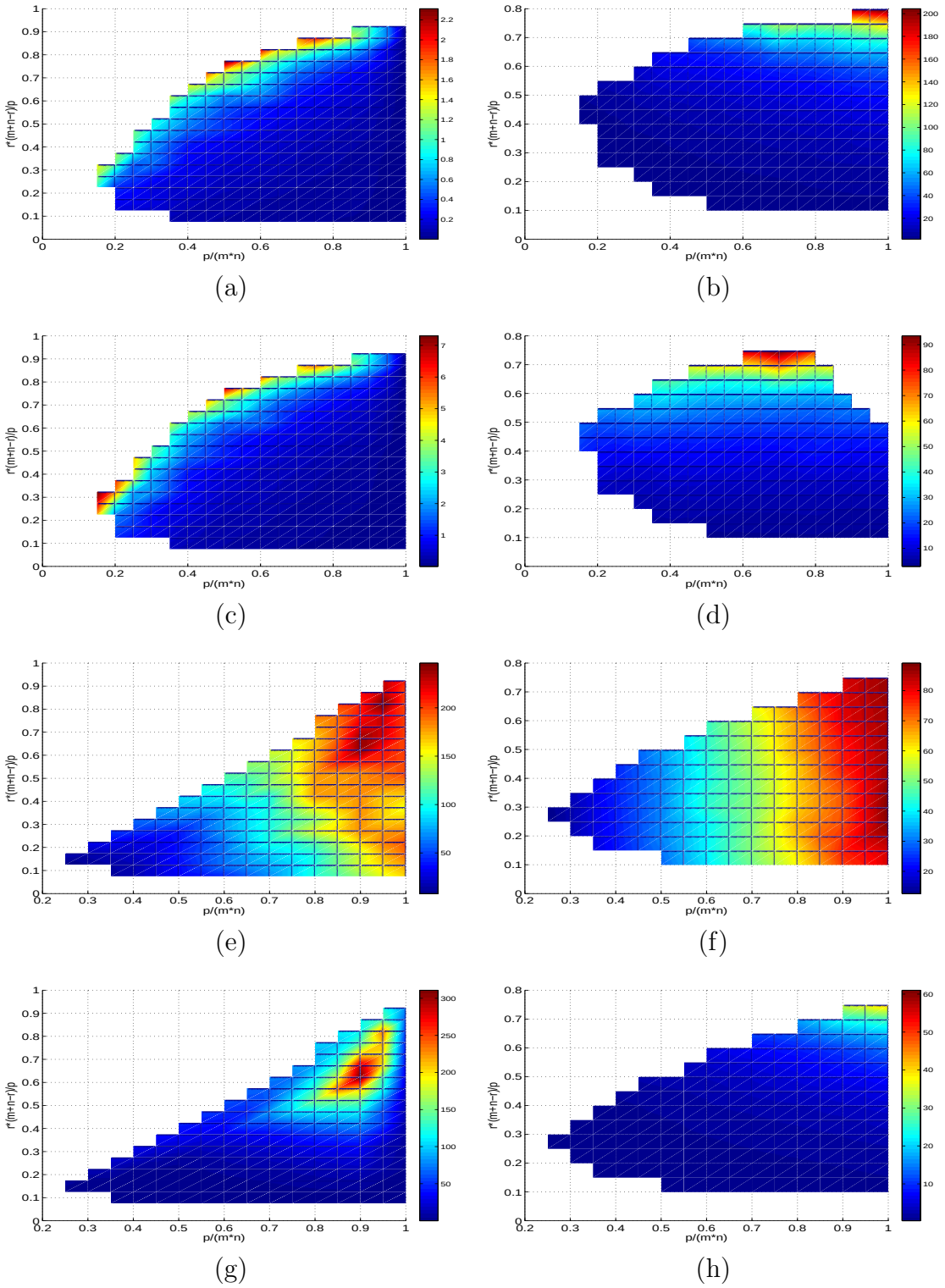


Figure 2.8: Average recovery time (s) for: (a) NIHT: entry sensing, (b) NIHT: Gaussian sensing, (c) IHT: entry sensing, (d) IHT: Gaussian sensing, (e) NNM: entry sensing, (f) NNM: Gaussian sensing, (g) PF: entry sensing, and (h) PF: Gaussian sensing. Entry sensing tests are for $m = n = 80$ and Gaussian sensing texts for $m = n = 40$. IHT uses the fixed stepsize $\alpha = 0.65$.

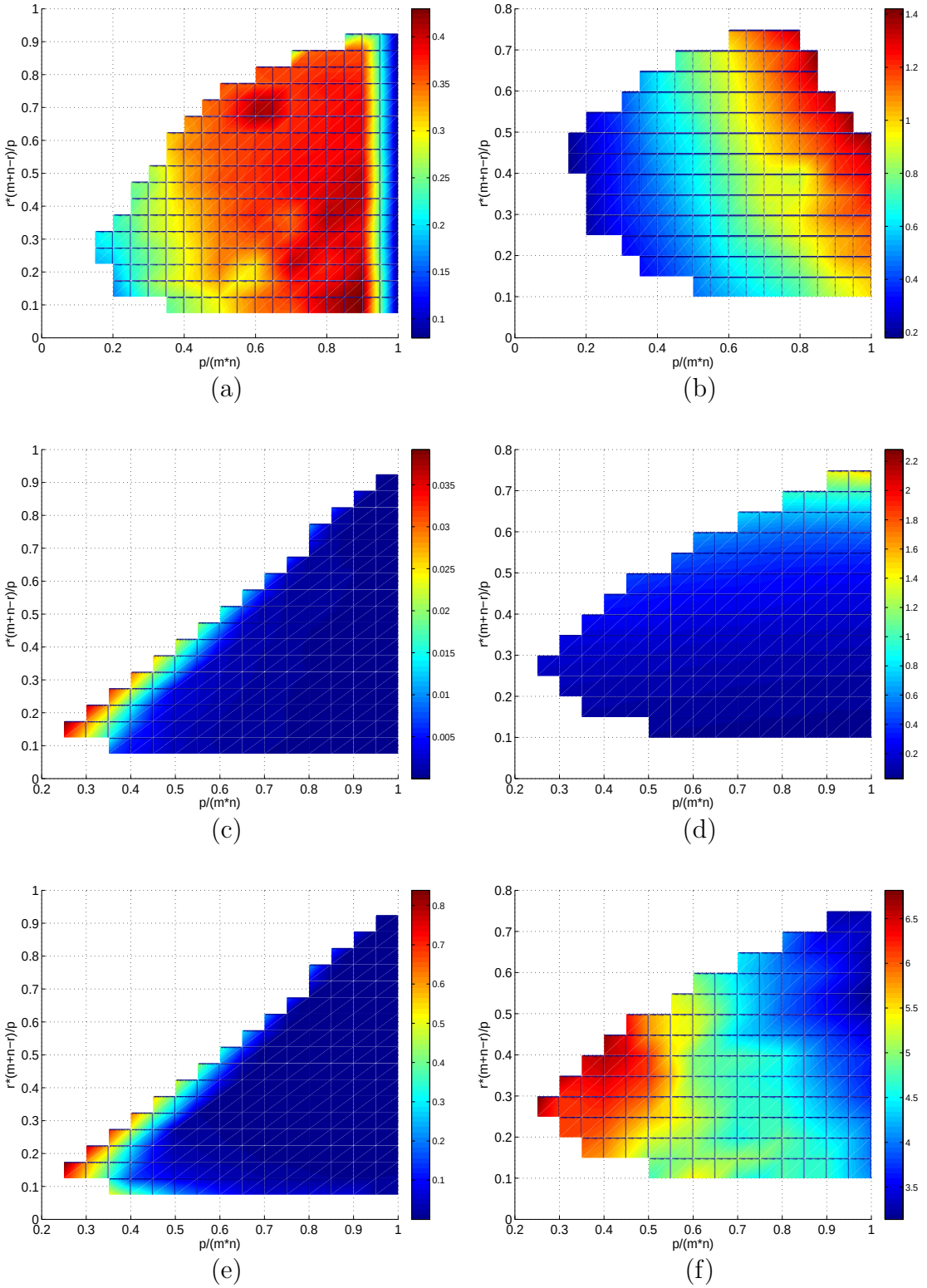


Figure 2.9: Average recovery ratio for NIHT divided by: (a) IHT: entry sensing, (b) IHT: Gaussian sensing, (c) NNM: entry sensing, (d) NNM: Gaussian sensing, (e) PF: entry sensing, and (f) PF: Gaussian sensing. Entry sensing tests are for $m = n = 80$ and Gaussian sensing texts for $m = n = 40$. IHT uses the fixed stepsize $\alpha = 0.65$.

2.3 Proof of Theorem 2.1

In this section, we present a proof to Thm. 2.1. This proof is similar to those in [19, 20, 52, 60], differing only in how the stepsize α_{l-1} is bounded, and is equally valid for each of the three variants of NIHT using the stepsizes (2.4) - (2.6).

Proof of Theorem 2.1. The proof proceeds by first deriving an inequality where $\|X_l - X^o\|_F$ is bounded by a factor multiplying $\|X_{l-1} - X^o\|_F$, and then showing that this multiplicative factor can be less than one if the sensing operator $\mathcal{A}(\cdot)$ has small RICs. Let X^o be the measured rank r matrix, with measurements $y = \mathcal{A}(X^o)$ and let

$$W_{l-1} = X_{l-1} + \alpha_{l-1}\mathcal{A}^*\left(y - \mathcal{A}\left(X_{l-1}\right)\right)$$

be the intermediate update to X_{l-1} in NIHT, as stated in Alg. 5, before it is projected to the rank r update X_l. Eckart-Young theorem [95, Thm. 5.9] ensures that X_l is the rank r matrix nearest to W_{l-1} in the Frobenius norm,

$$\|W_{l-1} - X_l\|_F^2 \leq \|W_{l-1} - X^o\|_F^2 . \tag{2.13}$$

Expanding $\|W_{l-1} - X_l\|_F^2$ and bounding it from above using (2.13) gives

$$\begin{aligned} \|W_{l-1} - X_l\|_F^2 &= \|W_{l-1} - X^o + X^o - X_l\|_F^2 \\ &= \|W_{l-1} - X^o\|_F^2 + \|X^o - X_l\|_F^2 \\ &\quad +2\left\langle W_{l-1} - X^o, X^o - X_l\right\rangle \\ &\leq \|W_{l-1} - X^o\|_F^2 . \end{aligned} \tag{2.14}$$

Cancelling $\|W_{l-1} - X^o\|_F^2$ in the above inequality gives the inequality

$$\begin{aligned} \|X_l - X^o\|_F^2 &\leq 2\left\langle W_{l-1} - X^o, X_l - X^o\right\rangle \\ &= 2\left\langle X_{l-1} - X^o, X_l - X^o\right\rangle - 2\alpha_{l-1}\left\langle \mathcal{A}^*\left(\mathcal{A}\left(X_{l-1} - X^o\right)\right), X_l - X^o\right\rangle \\ &\quad +2\alpha_{l-1}\left\langle \mathcal{A}^*\left(e\right), X_l - X^o\right\rangle \\ &= 2\left\langle X_{l-1} - X^o, X_l - X^o\right\rangle - 2\alpha_{l-1}\left\langle \mathcal{A}\left(X_{l-1} - X^o\right), \mathcal{A}\left(X_l - X^o\right)\right\rangle \\ &\quad +2\alpha_{l-1}\left\langle e, \mathcal{A}\left(X_l - X^o\right)\right\rangle . \end{aligned} \tag{2.15}$$

Let $Q_l \in \mathbb{R}^{m\times 3r}$ have orthogonal columns which span the space of all of the columns of X^o, X_{l-1}, and X_l, and let $P_Q^l := Q_l Q_l^*$ be the projection matrix to this column space; in particular, $P_Q^l X^o = X^o$, $P_Q^l X_{l-1} = X_{l-1}$, and $P_Q^l X_l = X_l$. Define

$\mathcal{A}_Q(Z) := \mathcal{A}(P_Q^l Z)$ which corresponds to replacing the sensing matrices $\{A_\ell\}_{\ell=1}^p$ of the unrestricted sensing operator $\mathcal{A}(\cdot)$ with the sensing matrices $\{P_Q^l A_\ell\}_{\ell=1}^p$ and the correspondingly associated adjoint operator $\mathcal{A}_Q^*(\cdot)$ as would follow in the definition (1.20) with A_ℓ replaced with $P_Q^l A_\ell$. With these projected operators we can express and further bound the inequality (2.15) as follows

$$\begin{aligned}
||X_l - X^o||_F^2 &\leq 2\left\langle X_{l-1} - X^o, X_l - X^o\right\rangle - 2\alpha_{l-1}\left\langle \mathcal{A}\left(X_{l-1} - X^o\right), \mathcal{A}\left(X_l - X^o\right)\right\rangle \\
&\quad +2\alpha_{l-1}\left\langle e, \mathcal{A}\left(X_l - X^o\right)\right\rangle \\
&= 2\left\langle X_{l-1} - X^o, X_l - X^o\right\rangle - 2\alpha_{l-1}\left\langle \mathcal{A}_Q\left(X_{l-1} - X^o\right), \mathcal{A}_Q\left(X_l - X^o\right)\right\rangle \\
&\quad +2\alpha_{l-1}\left\langle e, \mathcal{A}_Q\left(X_l - X^o\right)\right\rangle \\
&= 2\langle X_{l-1} - X^o, \left(I - \alpha_{l-1}\mathcal{A}_Q^*\mathcal{A}_Q\right)\left(X_l - X^o\right)\rangle \\
&\quad +2\alpha_{l-1}\left\langle e, \mathcal{A}\left(X_l - X^o\right)\right\rangle \\
&\leq 2\left\|I - \alpha_{l-1}\mathcal{A}_Q^*\mathcal{A}_Q\right\|_2 \cdot \left\|X_{l-1} - X^o\right\|_F \cdot \left\|X_l - X^o\right\|_F \\
&\quad +2|\alpha_{l-1}| \cdot \left\|\mathcal{A}\left(X_l - X^o\right)\right\|_2 \cdot \|e\|_2 \\
&\leq 2\left\|I - \alpha_{l-1}\mathcal{A}_Q^*\mathcal{A}_Q\right\|_2 \cdot \left\|X_{l-1} - X^o\right\|_F \cdot \left\|X_l - X^o\right\|_F \\
&\quad +2|\alpha_{l-1}| \cdot \left(1 + \delta_{2r}\right)^{1/2} \cdot \left\|X_l - X^o\right\|_F \cdot \|e\|_2, \qquad (2.16)
\end{aligned}$$

where the last inequality follows from the upper RIC bound on the matrix $X_l - X^o$ of rank at most $2r$

$$\left\|\mathcal{A}\left(X_l - X^o\right)\right\|_2^2 \leq \left(1 + \delta_{2r}\right)\left\|X_l - X^o\right\|_F^2.$$

Cancelling one power of $\left\|X_l - X^o\right\|_F$ from each side of (2.16) gives the desired bound on the error at step l as compared to the error at step $l-1$, namely

$$\left\|X_l - X^o\right\|_F \leq 2\left\|I - \alpha_{l-1}\mathcal{A}_Q^*\mathcal{A}_Q\right\|_2 \cdot \left\|X_{l-1} - X^o\right\|_F + 2|\alpha_{l-1}| \cdot \left(1 + \delta_{2r}\right)^{1/2} \cdot \|e\|_2, \quad (2.17)$$

and it only remains to bound $|\alpha_{l-1}|$ and the operator norm $\left\|I - \alpha_{l-1}\mathcal{A}_Q^*\mathcal{A}_Q\right\|_2$.

The operator $\mathcal{A}_Q^*\mathcal{A}_Q(\cdot)$ is self adjoint and acts on the projected space of rank $3r$ matrices defined by P_Q^l; as such, its spectrum satisfied the RIC bounds

$$1 - \delta_{3r} \leq \lambda(\mathcal{A}_Q^*\mathcal{A}_Q) \leq 1 + \delta_{3r}. \qquad (2.18)$$

Similarly, the inverse of the stepsize α_{l-1} is the ratio of the operator $\mathcal{A}(\cdot)$ acting on a rank r matrix $\texttt{Proj}_{U_{l-1}}\left(R_{l-1}\right)$, giving the RIC bounds

$$\frac{1}{1+\delta_r} \leq \alpha_{l-1} = \frac{\left\|\texttt{Proj}_{U_l}\left(R_{l-1}\right)\right\|_F^2}{\left\|\mathcal{A}\left(\texttt{Proj}_{U_l}\left(R_{l-1}\right)\right)\right\|_2^2} \leq \frac{1}{1-\delta_r}. \qquad (2.19)$$

Combining the bounds on the spectrum of $\mathcal{A}_Q^*\mathcal{A}_Q$ in (2.18) and the stepsize in (2.19), we bound the spectrum of $I - \alpha_{l-1}\mathcal{A}_Q^*\mathcal{A}_Q$ as

$$1 - \frac{1+\delta_{3r}}{1-\delta_r} \leq \lambda(I - \alpha_{l-1}\mathcal{A}_Q^*\mathcal{A}_Q) \leq 1 - \frac{1-\delta_{3r}}{1+\delta_r}. \tag{2.20}$$

The magnitude of the lower bound in (2.20) is greater than that of the upper bound, giving the operator bound

$$\left\|I - \alpha_{l-1}\mathcal{A}_Q^*\mathcal{A}_Q\right\|_2 \leq \frac{1+\delta_{3r}}{1-\delta_r} - 1 = \frac{\delta_{3r}+\delta_r}{1-\delta_r} \leq \frac{2\delta_{3r}}{1-\delta_r}. \tag{2.21}$$

Defining

$$\mu = \frac{4\delta_{3r}}{1-\delta_r},\ \xi = \frac{2\left(1+\delta_{2r}\right)^{1/2}}{1-\delta_r}, \tag{2.22}$$

and substituting (2.19) and (2.21) into (2.17) gives

$$\|X_l - X^o\|_F \leq \mu \|X_{l-1} - X^o\|_F + \xi\|e\|_2. \tag{2.23}$$

Consequently if $\mu < 1$,

$$\|X_l - X^o\|_F \leq \mu^l \|X_0 - X^o\|_F + \frac{\xi}{1-\mu}\|e\|_2. \tag{2.24}$$

□

Chapter 3

Conjugate Gradient Iterative Hard Thresholding for Compressed Sensing

In this chapter, we present the conjugate gradient iterative hard thresholding (CGIHT) family of algorithms for compressed sensing. The chapter is outlined as follows[1]. In Sec. 3.1, we briefly review the conjugate gradient iterative method for linear equations. In Sec. 3.2, we review another accelerated iterative hard thresholding algorithm for compressed sensing: HTP. The CGIHT family of algorithms for compressed sensing are presented in Sec. 3.3. In Sec. 3.4, we present the fast iterative hard thresholding (FIHT) algorithm for compressed sensing. The empirical results presented in Sec. 3.5 contrast CGIHT with other hard thresholding algorithms. Section 3.7 concludes this chapter with proofs of the recovery guarantees.

3.1 Conjugate gradient method

The conjugate gradient iterative method (CG, Alg. 6), which was proposed by Hestenes and Stiefel in 1952 [59], is a Krylov subspace method for solving the system of linear equations

$$Ax = y, \tag{3.1}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite. Let $\mathcal{K}_l$ be the lth Krylov subspace generated by (A, y):

$$\mathcal{K}_l = \langle y, Ay, \cdots, A^{l-2}y, A^{l-1}y\rangle, \tag{3.2}$$

[1]Material in this chapter has been prepared for publication and in preprint [15], which is a joint authorship with J. D. Blanchard and J. Tanner whose permission has been obtained for the inclusion of the material.

and x^o be the solution to (3.1). Then in iteration l, CG implicitly computes an approximation $x_l \in \mathcal{K}_l$ which is the solution to the minimization problem

$$\min_{x \in \mathcal{K}_l} (x - x^o)^T A (x - x^o).$$

Algorithm 6 Conjugate Gradient Iteration (CG)

Initilization: $x_0 = 0$, $r_{-1} = y$, $p_{-1} = 0$, $l = 1$

Iteration: During iteration l, **do**

1: $r_{l-1} = y - Ax_{l-1}$	gradient descent direction
2: $\beta_{l-1} = \left(r_{l-1}^T r_{l-1}\right) / \left(r_{l-2}^T r_{l-2}\right)$	compute orthogonalization weight
3: $p_{l-1} = r_{l-1} + \beta_{l-1} p_{l-2}$	conjugate gradient descent direction
4: $\alpha_{l-1} = \left(r_{l-1}^T r_{l-1}\right) / \left(p_{l-1}^T A p_{l-1}\right)$	steepest descent stepsize
5: $x_l = x_{l-1} + \alpha_{l-1} p_{l-1}$	solution update

Output: x_l

The well known conjugate properties of CG iteration are presented in Thm. 3.1, which is drawn from [95].

Theorem 3.1. Let the CG iteration (Alg. 6) be applied to a symmetric positive definite matrix problem $Ax = y$. As long as the iteration has not yet converged, the algorithm proceeds without divisions by zero, and we have the following identities of subspaces:

$$\begin{aligned} \mathcal{K}_l &= \langle x_1, x_2, \cdots, x_l \rangle = \langle p_0, p_1, \cdots, p_{l-1} \rangle \\ &= \langle r_0, r_1, \cdots, r_{l-1} \rangle = \langle y, Ay, \cdots, A^{l-1} y \rangle. \end{aligned} \tag{3.3}$$

Moreover, the residuals are orthogonal,

$$r_l^T r_j = 0 \quad (j < l), \tag{3.4}$$

and the search directions are "A-conjugate",

$$p_l^T A p_j = 0 \quad (j < l). \tag{3.5}$$

3.1.1 Optimization perspective of CG

The CG iteration can be interpreted as an optimization algorithm for the unconstrained problem

$$\min_{x \in \mathbb{R}^N} \ \psi(x) = \frac{1}{2} x^T A x - y^T x, \tag{3.6}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite. First note that the global minimum of (3.6) is the solution to the linear system (3.1). Since $\psi(x)$ is a strongly convex function, one can apply the gradient descent algorithm to optimize (3.6), see Alg. 7. In each iteration of Alg. 7, the current estimate is updated along the gradient descent direction with the steepest descent stepsize.

Algorithm 7 Steepest Decent Iteration (SD)

Initilization: $x_0 = 0$, $l = 1$

Iteration: During iteration l, **do**

1: $r_{l-1} = y - Ax_{l-1}$ gradient descent direction
2: $\alpha_{l-1} = (r_{l-1}^T r_{l-1})/(r_{l-1}^T A r_{l-1})$ steepest descent stepsize
3: $x_l = x_{l-1} + \alpha_{l-1} r_{l-1}$ solution update

Output: x_l

The CG method is also a line-search algorithm, see Step 2 in Alg. 6, and it updates the current estimate along the descent direction p_{l-1} which is a linear combination of the gradient descent direction and the prior search direction

$$p_{l-1} = r_{l-1} + \beta_{l-1} p_{l-2}. \tag{3.7}$$

The choice of the orthogonalization weight β_{l-1} forces the new search direction p_{l-1} to be A-orthogonal to the prior search direction p_{l-2} , that is

$$\langle p_{l-1}, Ap_{l-2}\rangle = \langle Ap_{l-1}, p_{l-2}\rangle = 0. \tag{3.8}$$

Property (3.8) follows directly from another equivalent formula for the orthogonalization weight β_{l-1} [54],

$$\beta_{l-1} = -\langle r_{l-1}, Ap_{l-2}\rangle \,/\, \langle p_{l-2}, Ap_{l-2}\rangle \,. \tag{3.9}$$

What makes the choice of β_{l-1} in CG outstanding is that minimizing $f(x)$ along p_{l-1} actually minimizes it over the Krylov subspace $\mathcal{K}_l$. Furthermore, from (3.3) in Thm. 3.1, one has

$$p_{l-1} \in \mathcal{K}_l = \langle r_0, r_1, \cdots, r_{l-1}\rangle, \tag{3.10}$$

which implies the search direction p_{l-1} in CG is an optimal linear combination of all the past gradient descent directions. It is remarkable that this optimality can be achieved by only memorizing the prior search direction p_{l-2}.

The memory technique is widely used and very effective in numerical linear algebra and optimization. Here are another two examples.

1. In numerical linear algebra, the Gauss-Seidel method for linear systems can be accelerated by taking an appropriate weighted average of the current estimate and the prior one; and it is further accelerated in Chebshev acceleration by taking some suitable linear combinations of all the past approximations [38].

2. In optimization, the fast (proximal) gradient method achieves the optimal convergence rate by starting each update from an appropriate convex combination of the last two approximations [78]. Similar to CG for linear systems, the nonlinear CG method can accelerate the gradient descent method by searching along a descent direction which is some linear combination of the gradient descent direction and the prior search direction [80].

3.2 Hard thresholding pursuit

NIHT, Alg. 2, can be viewed as an analogue of the steepest descent method for underdetermined linear systems, but with two major differences: a) the stepsize α_{l-1} is a local steepest descent stepsize, and b) the hard thresholding operator is employed to project the estimate onto the set of k-sparse vectors. The heuristic for the stepsize selection in NIHT guarantees the fast convergence rate once the true support set is identified and remains unchanged, since then NIHT is simply the steepest descent method for the overdetermined system $A_\Gamma x_\Gamma = y$, where A_Γ is the submatrix formed by the columns of A indexed by the support set of the underlying sparse solution.

Despite the simplicity of IHT and NIHT, they have numerous near optimal properties. Theorem 1.8 implies NIHT can reconstruct the solution to (1.1) from a number of measurements that is proportional to the sparsity of the measured data (see Remark 1.6). Moreover, for numerous measurement matrices, NIHT is observed to be able to solve (1.1) for the same problem sizes as more sophisticated algorithms, and is often able to solve it to moderate accuracy in less computational time than can more sophisticated variants. However, NIHT suffers from the slow asymptotic convergence rate of the steepest descent method if the sensing matrix is ill conditioned when restricted to the support set of the measured data. For example, Alg. 2 for compressed sensing has an asymptotic convergence factor per iteration of $\frac{\kappa-1}{\kappa+1}$ where $\kappa = \text{cond}\,(A_\Gamma^* A_\Gamma)$ denotes condition number [95, pg. 94] of $A_\Gamma^* A_\Gamma$.

Many of the more sophisticated hard thresholding algorithms have been designed in part to overcome this slow asymptotic convergence rate of NIHT. These more ad-

vanced algorithms typically achieve this with the inclusion of the conjugate gradient method to solve for the local minima of the objective in (1.9) while the support set is held fixed. A highly incomplete list of examples of such methods are: CoSaMP [77], SP [34], HTP [49]. Though such methods gain the fast convergence rate of the conjugate gradient method, they do so at the cost of higher per iteration complexity. It is observed in [14] that when the compressed sensing problem is solved to moderate accuracy, $\|b - Ax\|_2 / \|b\| \approx 10^{-4}$, the disadvantage of the high per iteration complexity causes early iterations, where the support set is typically changing, to result in an overall average computational time that is often as long or longer than that of NIHT. When approximate solutions are sought with $\|b - Ax\|_2 / \|b\| \gg 10^{-4}$, NIHT has even less computational time as compared with more sophisticated algorithms due to the support set identification portion being dominant. On the other hand, when approximate solutions are sought with $\|b - Ax\|_2 / \|b\| \ll 10^{-4}$, the more sophisticated algorithms can be substantially faster due to the asymptotic convergence rate more directly impacting the computational time.

HTP, Alg. 8, is the next simplest accelerated hard thresholding algorithm for (1.1). HTP corresponds to NIHT with an added pseudoinverse (Step 5) to compute the optimal values of the k nonzero entries given the current estimate of the support Γ_l. Typically the pseudoinverse is computed using the conjugate gradient method for the restricted system $A^*_{\Gamma_l} A_{\Gamma_l} x_{\Gamma_l} = A^*_{\Gamma_l} y$. HTP possesses a similar recovery guarantee to NIHT in terms of RICs of the measurement matrix.

Theorem 3.2 ([49])**.** For any $x^o \in \Sigma_k$ and $y = Ax^o$, the iterates of HTP converge to x^o provided

$$\delta_{3k}(A) < \frac{1}{\sqrt{3}}.$$

An important aspect of the computational effectiveness of HTP is to determine how many iterations of CG should be implemented per iteration to approximately solve the least squares subproblem. Properly controlling the computational cost of solving the subproblem is essential to obtain an overall faster algorithm than NIHT. Moreover, excessively solving the subproblem (1.9) when restricted to the current estimate for the support set can result in difficulty moving to a different support set; this can ultimately cause HTP to fail to recover sparse vectors of cardinality k that can be recovered by NIHT.

Algorithm 8 Hard Thresholding Pursuit (HTP, [49])

Initilization: sparsity level k, $w_{-1} = A^*y$, $\Gamma_0 = \texttt{PrincipalSupport}_k(w_{-1})$, $x_0 = \texttt{Proj}_{\Gamma_0}(w_{-1})$, $l = 1$

Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ — gradient descent direction

2: $\alpha_{l-1} = \frac{\left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}{\left\|A\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}$ — compute the stepsize

3: $w_{l-1} = x_{l-1} + \alpha_{l-1}r_{l-1}$ — gradient descent update

4: $\Gamma_l = \texttt{PrincipalSupport}_k(w_{l-1})$ — proxy to the support set

5: $(x_l)_{\Gamma_l} = A_{\Gamma_l}^{\dagger}y$, $(x_l)_{\Gamma_l^c} = 0$ — solution update

Output: x_l

3.3 Conjugate gradient iterative hard thresholding

In this section, we present new algorithms for the solution to (1.1), which we refer to collectively as CGIHT for compressed sensing. CGIHT is designed to balance the advantage of the low per iteration complexity of NIHT with the fast asymptotic convergence rate of HTP. CGIHT is observed to be able to more reliably recover k-sparse signals for large values of k than can NIHT, while having an average lower computational time. Moreover, despite the added complexity of CGIHT making explicit use of past search directions, there are variants of CGIHT with provable recovery guarantees analogous to those for other hard thresholding algorithms.

We begin our presentation of CGIHT for compressed sensing with its simplest variant which we simply refer to as CGIHT, see Alg. 9. In each iteration of CGIHT, the current estimate x_{l-1} is updated along the direction p_{l-1}, using a stepsize α_{l-1}, followed by thresholding onto the space of k-sparse vectors. The search direction p_{l-1} is selected to be exactly the residual r_0 in iteration $l = 1$, and is otherwise selected to be the residual r_{l-1} projected to be conjugate orthogonal to the past search direction when restricted to the current estimate of the support set Γ_{l-1}. This procedure is analogous to the conjugate gradient method, though lacking some of its important properties. In particular, the search directions do not in general remain conjugate orthogonal, except in the simplest case where the support set never changes [83, 103]; and this simplest variant of CGIHT lacks a proof of convergence to the measured k-sparse vector.

To recover the conjugate gradient property that past search directions maintain

Algorithm 9 CGIHT for compressed sensing

Initilization: sparsity level k, $w_{-1} = A^*y$, $\Gamma_0 = \texttt{PrincipalSupport}_k(w_{-1})$, $x_0 = \texttt{Proj}_{\Gamma_0}(w_{-1})$, $l = 1$

Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ gradient descent direction

2: if $l = 1$,

$\beta_{l-1} = 0$ set orthogonalization weight

else

$\beta_{l-1} = \frac{\left\langle A\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1}), A\texttt{Proj}_{\Gamma_{l-1}}(p_{l-2})\right\rangle}{\left\|A\texttt{Proj}_{\Gamma_{l-1}}(p_{l-2})\right\|_2^2}$ compute orthogonalization weight

3: $p_{l-1} = r_{l-1} + \beta_{l-1}p_{l-2}$ update search direction

4: $\alpha_{l-1} = \frac{\left\langle \texttt{Proj}_{\Gamma_{l-1}}(r_{l-1}), \texttt{Proj}_{\Gamma_{l-1}}(p_{l-1})\right\rangle}{\left\|A\texttt{Proj}_{\Gamma_{l-1}}(p_{l-1})\right\|_2^2}$ compute the stepsize

5: $w_{l-1} = x_{l-1} + \alpha_{l-1}p_{l-1}$ conjugate gradient descent update

6: $\Gamma_l = \texttt{PrincipalSupport}_k(w_{l-1})$ proxy to the support set

7: $x_l = \texttt{Proj}_{\Gamma_l}(w_{l-1})$ solution update

Output: x_l

conjugate orthogonality over a sequence of iterates acting the same support set requires restarting the conjugate gradient method when the support set Γ_{l-1} is different with Γ_{l-2}. We will present two restarted variants of CGIHT for compressed sensing, for which we are able to provide optimal order recovery proofs, similar to other hard thresholding algorithms in terms of RICs of the measurement matrix.

The first variant of CGIHT with a provably optimal order recovery guarantee is referred to as CGIHT restarted, see Alg. 10. The first iteration $l = 1$ of Alg. 9 and Alg. 10 are identical. Subsequent iterations differ in their choice of search directions. Alg. 10 uses the residual as the search direction in any iteration where the support set differs from that of the prior iteration, that is $\Gamma_{l-1} \neq \Gamma_{l-2}$. However, in iterations where the support set Γ_{l-1} is the same as at the prior iteration, the search direction p_{l-1} is selected to be the residual r_{l-1} projected to be conjugate orthogonal to the past search direction p_{l-2} when restricted to the support set Γ_{l-1}. Starting each instance of the orthogonalization with the first search direction having been the residual recovers the conjugate orthogonalization of all the search directions over a sequence of iterations where the support set remains unchanged; that is $\langle A_{\Gamma_{l-1}}p_{l-1}, A_{\Gamma_{l-1}}p_{l-j}\rangle = 0$ for j from 2 increasing until the first value of j such that $\Gamma_{l-j} \neq \Gamma_{l-1}$. Recovering this orthogonalization property allows for the use of an efficient formula for computing the stepsize α_{l-1} and the orthogonalization weight β_{l-1}, and enable us to provide an optimal order recovery proof using RICs of the measurement matrix.

Algorithm 10 CGIHT restarted for compressed sensing

Initilization: sparsity level k, $w_{-1} = A^*y$, $\Gamma_0 = \texttt{PrincipalSupport}_k(w_{-1})$, $x_0 = \texttt{Proj}_{\Gamma_0}(w_{-1})$, $l = 1$

Iteration: During iteration l, **do**

1: $r_{l-1} = A^*(y - Ax_{l-1})$ gradient descent direction

2: if $l = 1$ or $\Gamma_{l-1} \neq \Gamma_{l-2}$,

$\beta_{l-1} = 0$ set orthogonalization weight

else

$\beta_{l-1} = \dfrac{\left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}{\left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-2})\right\|_2^2}$ compute orthogonalization weight

3: $p_{l-1} = r_{l-1} + \beta_{l-1}p_{l-2}$ update search direction

4: $\alpha_{l-1} = \dfrac{\left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}{\left\|A\texttt{Proj}_{\Gamma_{l-1}}(p_{l-1})\right\|_2^2}$ compute the stepsize

5: $w_{l-1} = x_{l-1} + \alpha_{l-1}p_{l-1}$ conjugate gradient descent update

6: $\Gamma_l = \texttt{PrincipalSupport}_k(w_{l-1})$ proxy to the support set

7: $x_l = \texttt{Proj}_{\Gamma_l}(w_{l-1})$ solution update

Output: x_l

The recovery guarantee for Alg. 10, Thm. 3.3, considers the typical signal model $y = Ax^o + e$ where x^o has at most k nonzeroes. In this model x^o can be viewed as the k-sparse vector which minimizes $\|y - Ax\|_2$ and e as the discrepancy between y and Ax^o. Alternatively x^o can be viewed as a k-sparse vector measured by A and that y is contaminated by additive noise; though, in this perspective, Thm. 3.3 does not consider any particular model for e.

Theorem 3.3 (Recovery guarantee for CGIHT restarted for compressed sensing, Alg. 10.)**.** Let A be an $m \times n$ matrix with $m < n$, and $y = Ax^o + e$ for any x^o with at most k nonzeroes. Define the following constants in terms of the RICs of A[2]

$$\tau_1 = \frac{4\delta_{3k}}{1-\delta_k} + \frac{2\delta_k(1+\delta_k)}{(1-\delta_k)^2}, \qquad \tau_2 = \frac{4\delta_k(1-\delta_{3k})(1+\delta_k)}{(1-\delta_k)^3},$$
$$\mu = \frac{1}{2}\left(\tau_1 + \sqrt{\tau_1^2 + 4\tau_2}\right), \qquad \xi = 2\frac{(1+\delta_{2k})^{1/2}}{1-\delta_k}. \tag{3.11}$$

If the RICs of A satisfy

$$\frac{2\delta_{3k}(5+\delta_k)}{(1-\delta_k)^2} < 1, \tag{3.12}$$

then $\mu < 1$ and the iterates of Alg. 10 satisfy

$$\|x_l - x^o\|_2 \leq \mu^l \|x_0 - x^o\|_2 + \frac{\xi}{1-\mu}\|e\|_2. \tag{3.13}$$

[2]In Thms. 3.3, 3.4, and 3.5, the restricted isometry constant $\delta_{ck}(A)$ of the measurement matrix A is abbreviated to δ_{ck} for simplicity.

Proof. See Sec. 3.7.1 for a proof. □

Theorem 3.3 implies that if $e = 0$, the correct support set of x will be identified after logarithmically many iterations [10], and CGIHT restarted can recover (within arbitrary precision) any k-sparse vector measured by A. Once the correct support set Γ of the measured k-sparse vector has been identified and the support sets of all the successive approximate solutions stay on Γ, CGIHT restarted is simply the conjugate gradient method applied to the overdetermined linear system $A_\Gamma x_\Gamma = y$. Thus the asymptotic convergence factor is given by $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$, where $\kappa = \mathrm{cond}\,(A_\Gamma^* A_\Gamma)$. Note that this asymptotic rate is much smaller than the rate μ given in Thm. 3.3; the rate μ in Thm. 3.3 is the rate of ℓ_2 error contraction per iteration, including the iterations where the support set is incorrect.

The similarities and differences between Algs. 9 and 10 are summarized as follows.

1. Both Algs. 9 and 10 are analogues of the classical conjugate gradient method (Alg. 6) for finding sparsest solutions to underdetermined linear systems, but with the stepsize α_{l-1} and orthogonalization weight β_{l-1} computed to be locally optimal. However, Alg. 10 is less greedy than Alg. 9 since it uses a local CG update only when the support set of the current estimate is the same as that of the prior approximate solution.

2. The search direction p_{l-1} in both Algs. 9 and 10 can be expressed as some linear combinations of past residuals, that is

$$p_{l-1} \in \langle r_{l-1}, \cdots, r_0 \rangle\,, \tag{3.14}$$

although β_{l-1} will be set to zero when restarting occurs in Alg. 10. Thus both algorithms are seeking to accelerate NIHT by memorizing the past gradient descent directions.

3. Lacking the orthogonalization property on the support set estimate in Alg. 9 precludes the use of the most efficient formula for computing the conjugate gradient descent stepsize α_{l-1} and the orthogonalization weight β_{l-1} as in Alg. 10, resulting in one more matrix-vector product per iteration when the estimate of the support set changes.

As an alternative to the support set based restarting condition of Alg. 10, the conjugate gradient iterative hard thresholding method can be restarted based on a measure of the relative residual on the current support set. CGIHT projected, Alg. 11, corresponds to the nonlinear restarted CGIHT where restarting occurs when $\left\|r_{l-1} - \texttt{Proj}_{\Gamma_{l-1}}(p_{l-1})\right\|_2^2 / \left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2$ is sufficiently large. This restarting condition corresponds to the fraction of the current residual aligned with the current search direction, relative to the magnitude of the the residual on the current support set. Unlike Alg. 10 which has no tuning parameters, CGIHT projected has a restarting parameter θ controlling the computational effort used to decrease the residual on a given support set.

On one hand, CGIHT projected for compressed sensing can be viewed as a variant of HTP with the internal CG for the least squares subproblem terminating upon the relative fraction $\left\|r_{l-1} - \texttt{Proj}_{\Gamma_{l-1}}(p_{l-1})\right\|_2^2 / \left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2$ being sufficiently large. On the other hand, CGIHT projected can be interpreted as a probing CGIHT algorithm which predicts beforehand whether the estimate of the support set will change or not, and then determines next move based on the prediction.

Theorem 3.4 (Recovery guarantee for CGIHT projected for compressed sensing, Alg. 11.)**.** Let A be an $m \times n$ matrix with $m < n$, and $y = Ax^o + e$ for any x^o with at most k nonzeroes. Define the following constants in terms of the RICs of A and the restarting parameter c:

$$\mu = \frac{4(1+c)\delta_{3k}}{1-\delta_k}, \quad \theta_0 = \frac{2c\delta_{3k}}{1+\delta_{2k}}, \quad \text{and} \quad \xi = 2(1+\theta_0)\frac{(1+\delta_{2k})^{1/2}}{1-\delta_k}.$$

Then provided $\mu < 1$, the iterates of Alg. 11 with restarting condition $\theta < \theta_0$ satisfy

$$\|x_l - x^o\|_2 \leq \mu^l \|x_0 - x^o\|_2 + \frac{\xi}{1-\mu}\|e\|_2. \tag{3.15}$$

Proof. See Thm. 4.1 in Chapter 4 for an analogous proof. □

3.4 Fast iterative hard thresholding for compressed sensing

There are several accelerated hard thresholding algorithms for compressed sensing, such as CoSaMP [77], SP [34], and the Algebraic Pursuit (ALPS) family of algorithms [33], which differ from NIHT and HTP by having intermediate steps that further

Algorithm 11 CGIHT projected for compressed sensing

Initilization: sparsity level k, $w_{-1} = A^*y$, $\Gamma_0 = \texttt{PrincipalSupport}_k(w_{-1})$, $x_0 = \texttt{Proj}_{\Gamma_0}(w_{-1})$, RestartFlag $= 1$, set restart parameter θ, $l = 1$

Iteration: During iteration l, **do**

1: if $\frac{\left\|r_{l-1} - \texttt{Proj}_{\Gamma_{l-1}}(p_{l-1})\right\|_2^2}{\left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2} > \theta$

$\quad$ RestartFlag $= 1$ (set restart flag)

$\quad$ $\alpha_{l-1} = \frac{\left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}{\left\|A\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}$ (steepest descent update)

$\quad$ $w_{l-1} = x_{l-1} + \alpha_{l-1} r_{l-1}$ (gradient descent stepsize)

else

$\quad$ RestartFlag $= 0$ (set restart flag)

$\quad$ $\alpha_{l-1} = \frac{\left\|\texttt{Proj}_{\Gamma_{l-1}}(r_{l-1})\right\|_2^2}{\left\|A\texttt{Proj}_{\Gamma_{l-1}}(p_{l-1})\right\|_2^2}$ (steepest descent stepsize)

$\quad$ $w_{l-1} = x_{l-1} + \alpha_{l-1}\texttt{Proj}_{T_{l-1}}(p_{l-1})$ (conjugate gradient update)

2: $\Gamma_l = \texttt{PrincipalSupport}_k(w_{l-1})$ (proxy to the support set)

3: $x_l = \texttt{Proj}_{\Gamma_l}(w_{l-1})$ (solution update)

4: $r_l = A^*(y - Ax_l)$ (compute the residual)

5: if RestartFlag $= 1$,

$\quad$ $p_l = r_l$ (gradient descent direction update)

else

$\quad$ $\beta_l = \frac{\left\|\texttt{Proj}_{\Gamma_l}(r_l)\right\|_2^2}{\left\|\texttt{Proj}_{\Gamma_l}(r_{l-1})\right\|_2^2}$ (compute orthogonalization weight)

$\quad$ $p_l = r_l + \beta_l \texttt{Proj}_{\Gamma_l}(p_{l-1})$ (conjugate gradient descent direction update)

Output: x_l

update the values on support sets of cardinality greater than k. In this section, we present FIHT for compressed sensing, which is a variant of 1-ALPS(2) from the ALPS family of hard thresholding algorithms.

In [33], the author obtained a family of accelerated hard thresholding algorithms by integrating three building blocks: a) stepsize selection, b) memory exploitation, and c) further gradient descent or least squares update on the current estimate of the support set. 1-ALPS(2), Alg. 12, uses all the three building blocks, and is the version of ALPS observed to have the greatest overall computational efficacy. FIHT, Alg. 12, is a variant of 1-ALPS(2). FIHT differs from 1-ALPS(2) in its fourth step, where 1-ALPS(2) uses a support set of at least $2k$ by including k indices from the residual in its third step. As demonstrated by the empirical results in Sec. 3.5.3, FIHT is uniformly superior to 1-ALPS(2) in terms of both the sizes of the problems that are recoverable and overall computational time. Central to the fast asymptotic

Algorithm 12 FIHT: a variant of 1-ALPS(2) for compressed sensing

Initialization: sparsity level k, $w_{-1} = A^*y$, $\Gamma_0 = \texttt{PrincipalSupport}_k(w_{-1})$, $x_0 = \texttt{Proj}_{\Gamma_0}(w_{-1})$, $l = 1$

Iteration: During iteration l, **do**

1: if $l = 1$,
$\tau_{l-1} = 0$ (set momentum stepsize)
else
$\tau_{l-1} = \frac{\langle y - Ax_{l-1}, A(x_{l-1} - x_{l-2})\rangle}{\|A(x_{l-1} - x_{l-2})\|_2^2}$ (calculate momentum stepsize)

2: $v_{l-1} = x_{l-1} + \tau_{l-1}(x_{l-1} - x_{l-2})$ (new extrapolated point)

3: $\tilde{r}_{l-1} = A^*(y - Av_{l-1})$ (residual of extrapolated point)

4: $\tilde{\Gamma}_{l-1} = \texttt{Support}(v_{l-1})$ (support set for FIHT)
or
$\tilde{\Gamma}_{l-1} = \tilde{\Gamma}_{l-1} \cup \texttt{PrincipalSupport}_k\left(\texttt{Proj}_{\tilde{\Gamma}_{l-1}^c}(\tilde{r}_{l-1})\right)$ (support set for 1-ALPS(2))

5: $\tilde{\alpha}_{l-1} = \frac{\left\|\texttt{Proj}_{\tilde{\Gamma}_{l-1}}(\tilde{r}_{l-1})\right\|_2^2}{\left\|A\texttt{Proj}_{\tilde{\Gamma}_{l-1}}(\tilde{r}_{l-1})\right\|_2^2}$ (optimal stepsize restricted to support of w_{l-1})

6: $w_{l-1} = v_{l-1} + \tilde{\alpha}_{l-1}\tilde{r}_{l-1}$ (steepest descent step)

7: $\Gamma_l = \texttt{PrincipalSupport}_k(w_{l-1})$ (proxy to the support set)

8: $x_l = \texttt{Proj}_{\Gamma_l}(w_{l-1})$ (restriction to proxy support Γ_l)

9: $r_l = A^*(y - Ax_l)$ (residual of x_l)

10: $\alpha_l = \frac{\left\|\texttt{Proj}_{\Gamma_l}(r_l)\right\|_2^2}{\left\|A\texttt{Proj}_{\Gamma_l}(r_l)\right\|_2^2}$ (optimal stepsize restricted to Γ_l)

11: $x_l = x_l + \alpha_l\texttt{Proj}_{\Gamma_l}(r_l)$ (one more gradient descent on the restricted support)

Output: x_l

convergence of FIHT is the optimal selection of τ listed in the first step of FIHT. For fixed τ, the traditional 1-ALPS(2) has been proven to have a RIC based recovery guarantee analogous to other hard thresholding algorithms, see [64]; however, no such proof is currently available when the variable τ from the first step of Alg. 12 is used. In this section, we will show that FIHT with the variable τ also has an analogous recovery guarantee in terms of RICs of the measurement matrix.

Theorem 3.5 (Recovery guarantee for FIHT for compressed sensing, Alg. 12.)**.** Let A be an $m \times n$ matrix with $m < n$, and $y = Ax^o + e$ for any x^o with at most k nonzeroes. Define the following constants in terms of the RICs of A:

$$\mu = \frac{4\delta_{4k}\left(1+\delta_{2k}\right)}{\left(1-\delta_{2k}\right)^{3/2}\left(1-\delta_{3k}\right)^{1/2}}, \quad \xi = \frac{8\delta_{4k}\left(1+\delta_{2k}\right)^{1/2}}{\left(1-\delta_{2k}\right)^{3/2}\left(1-\delta_{3k}\right)^{1/2}} + \frac{4}{\left(1-\delta_{2k}\right)^{3/2}}. \quad (3.16)$$

Then provided $\mu < 1$, the iterates of Alg. 12 satisfy

$$\|x_l - x^o\|_2 \leq \mu^l \|x_0 - x^o\|_2 + \frac{\xi}{1-\mu}\|e\|_2. \quad (3.17)$$

Proof. See Sec. 3.7.2 for a proof. □

3.5 Empirical performance comparisons: noiseless case

While the theoretical guarantees establish sufficient conditions under which a particular greedy algorithm will successfully recover all the sparse vectors, the observed average case performance of the greedy algorithms is dramatically superior to the RIC based, uniformly sufficient conditions. The average case behaviour is often far more useful to practitioners since the theoretical guarantees require many more measurements than practically useful. In this section, we present empirical observations of CGIHT's efficacy for compressed sensing as compared with several leading hard thresholding algorithms. For the compressed sensing problem, the sparsity of the desired solution defines the minimum number of measurements required to capture the underlying information. If the support of the vector $x^o \in \mathbb{R}^n$ is known *a priori*, only $k = \|x^o\|_0$ measurements are necessary. On the other hand, the ambient dimension of the vector x^o defines the maximum number of measurements. Therefore, taking n measurements with $k < m < n$ defines the compressed sensing undersampling and oversampling ratios

$$\delta = \frac{m}{n} \quad \text{and} \quad \rho = \frac{k}{m}, \quad (3.18)$$

with the unit square $(\delta, \rho) \in [0, 1]^2$ defining the phase space for compressed sensing.

The empirical results will be presented in terms of *recovery phase transition curves* and *algorithm selection maps*. The recovery phase transition curve for each algorithm separates the phase space into two regions: success and failure. For problems instances with (δ, ρ) below the phase transition curve, the algorithm is observed to return an approximate solution matching the solution to (1.1). Alternatively, for problem instances with (δ, ρ) above the recovery phase transition curve, the algorithm is observed to return an approximation that is not a solution to (1.1). For each algorithm, the region of the phase space below the recovery phase transition curve is referred to as the *recovery region*. For a problem instance with sampling ratios (δ, ρ) in the intersection of the recovery regions for multiple algorithms, a practitioner must select an algorithm based on some criteria other than the ability to reconstruct a solution to (1.1). The algorithm selection maps of the phase space were introduced in [14] to identify the algorithm with least recovery time.

This section is structured as follows. In Sec. 3.5.1, we briefly review another hard thresholding algorithm CSMPSP, which is a hybrid of CoSaMP and SP. The experimental set-up is presented in Sec. 3.5.2. The empirical performance comparisons between FIHT and 1-ALPS(2) in Sec. 3.5.3 establish the computational advantages of FIHT over 1-ALPS(2). In Sec. 3.5.4, we compare the CGIHT variants with other leading hard thresholding algorithms, including FIHT, NIHT, HTP and CSMPSP.

3.5.1 CSMPSP: A hybrid of CoSaMP and SP

CoSaMP (Alg. 13, [77]) and SP (Alg. 14, [34]) are both support recovery algorithms which select the k largest magnitude entries of a vector obtained by applying a pseudoinverse to the measurement y. The columns of A selected for the pseudoinverse consist of two parts: a) the indices of the $2k$ (for CoSaMP) or k (for SP) largest magnitude entries of the vector obtained by applying A^* to the residual; b) the indices for the support set of the prior approximate solution. Compared with CoSaMP, SP computes an additional pseudoinverse on the selected support to obtain a new estimate in its fourth step.

Instead of testing CoSaMP and SP separately, we test a hybrid algorithm which has the advantageous properties of both algorithms, see CSMPSP in Alg. 15.

Algorithm 13 Compressive Sampling Matching Pursuit (CoSaMP, [77])

Initilization: $\Gamma_0 = \emptyset$, $r_0 = y$, $l = 1$

Iteration: During iteration l, **do**

1: $\tilde{\Gamma}_l = \Gamma_{l-1} \cup \{2k \text{ indices of largest magnitude entries of } A^* r_{l-1}\}$ support union

2: $\tilde{x} = A^\dagger_{\tilde{\Gamma}_l} y$ compute pseudoinverse

3: $\Gamma_l = \{k \text{ indices of largest magnitude entries of } \tilde{x}\}$ support estimate

4: $(x_l)_{\Gamma_l} = \tilde{x}_{\Gamma_l}$, $(x_l)_{\Gamma_l^c} = 0$ solution update

5: $r_l = y - Ax_l$ residual update

Output: x_l

Algorithm 14 Subspace Pursuit (SP, [34])

Initilization: $\Gamma_0 = \emptyset$, $r_0 = y$, $l = 1$

Iteration: During iteration l, **do**

1: $\tilde{\Gamma}_l = \Gamma_{l-1} \cup \{k \text{ indices of largest magnitude entries of } A^* r_{l-1}\}$ support union

2: $\tilde{x} = A^\dagger_{\tilde{\Gamma}_l} y$ compute pseudoinverse

3: $\Gamma_l = \{k \text{ indices of largest magnitude entries of } \tilde{x}\}$ support estimate

4: $(x_l)_{\Gamma_l} = A^\dagger_{\Gamma_l} y$, $(x_l)_{\Gamma_l^c} = 0$ solution update

5: $r_l = y - Ax_l$ residual update

Output: x_l

Algorithm 15 CSMPSP (a hybrid of CoSaMP and SP, [14])

Initilization: $\Gamma_0 = \emptyset$, $r_0 = y$, $l = 1$

Iteration: During iteration l, **do**

1: $\tilde{\Gamma}_l = \Gamma_{l-1} \cup \{k \text{ indices of largest magnitude entries of } A^* r_{l-1}\}$ support union

2: $\tilde{x} = A^\dagger_{\tilde{\Gamma}_l} y$ compute pseudoinverse

3: $\Gamma_l = \{k \text{ indices of largest magnitude entries of } \tilde{x}\}$ support estimate

4: $(x_l)_{\Gamma_l} = \tilde{x}_{\Gamma_l}$, $(x_l)_{\Gamma_l^c} = 0$ solution update

5: $r_l = y - Ax_l$ residual update

Output: x_l

3.5.2 Experimental set-up

The empirical performance comparisons of the hard thresholding algorithms for compressed sensing is conducted with the software *GAGA: GPU Accelerated Greedy Algorithms for Compressed Sensing*, Version 1.1.0 [13, 12]. The testing was conducted on a Linux machine with Intel Xeon E5-2643 CPUs @ 3.30 GHz, NVIDIA Tesla K10 GPUs, and executed from Matlab R2013a. As described in [14], the tested algorithms terminate when any one of the several stopping criteria is met: a maximum number of iterations is reached, the residual is small, the residual is no longer improving, the algorithm is diverging or the multiplicative average of the last fifteen linear convergence factors is near one,

$$\left(\frac{\|y - Ax_{l+15}\|_2}{\|y - Ax_l\|_2}\right)^{1/15} > 0.999. \tag{3.19}$$

The algorithms are tested with three random matrix ensembles:

- $\mathcal{N}$: dense Gaussian matrices with entries drawn i.i.d. from $\mathcal{N}(0, 1/m)$;
- $\mathcal{S}_7$: sparse matrices with 7 nonzero entries per column drawn with equal probability from $\{-1/\sqrt{7}, 1/\sqrt{7}\}$ and locations in each column chosen uniformly;
- DCT: m rows of the $n \times n$ Discrete Cosine Transform matrix are selected uniformly at random.

These three random matrix ensembles are representative of the random matrices frequently encountered in compressed sensing: $\mathcal{N}$ represents dense matrices, $\mathcal{S}_7$ represents sparse matrices, and DCT represents subsampling structured matrices with fast matrix-vector products.

The measured vectors x^o are taken from the random binary vector ensemble, B, which are formed by uniformly selecting k locations for the nonzero entries with values $\{-1, 1\}$ chosen with equal probability. A *problem class* (Mat, B) consists of a matrix ensemble, $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$, and a sparse vector drawn from the binary vector distribution, B. Alternative sparse vector ensembles, such as having nonzero entries drawn from a uniform or normal distribution, have been shown to have larger recovery regions in the (δ, ρ) phase space; related phase transitions and additional performance characteristics for alternative vector ensembles are available in [14]. In Sec. 3.5, we focus on the noiseless case, that is, the measured sparse vector x^o is from B, and the measurements are exactly given by $y = Ax^o$.

When the measured vector is taken from the vector ensemble B, the ℓ_∞ norm accurately determines when the support set of the approximation returned by a hard thresholding algorithm coincides with the support set of the measured vector. Additionally, when the ℓ_∞ norm is small, the algorithms have accurately identified the values of the nonzero entries. Therefore, for the problem classes (Mat, B), we say an algorithm has successfully recovered the measured vector if the output $\hat{x}$ differs by no more than 10^{-3} in any single component, namely $\|\hat{x} - x^o\|_\infty \leq 10^{-3}$ which implies that the correct support has been identified and that

$$\frac{\|\hat{x} - x^o\|_2}{\|x^o\|_2} \leq 10^{-3}. \tag{3.20}$$

In the tests for the compressed sensing problem, the parameter $\delta \in (0, 1)$ takes on thirty values

$$\delta \in \{0.001, 0.002, 0.004, 0.006, 0.008, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1, \ldots, 0.99\} \tag{3.21}$$

with eighteen additional uniformly spaced values of δ between 0.1 and 0.99. The parameter $\rho \in (0, 1)$ is sampled in two different ways, one to identify the recovery phase transition for each algorithm and the second for direct performance comparison throughout the recovery regions.

The empirical recovery phase transition curves are logistic regression curves identifying the 50% success rate for a given algorithm and problem class. In order to generate sufficient data for the logistic regression, the testing focuses on the phase transition region where the algorithm transitions from always succeeding to always failing. For a given problem class (Mat, B) and a specific value of n, the thirty values of m are chosen according to $m = \lceil \delta \cdot n \rceil$ with δ taken from (3.21). The phase transition region is found via a binary search which determines an interval $[k_{min}, k_{max}]$ so that the algorithm successfully recovers each of ten problem instances at $k < k_{min}$ and fails to recover any of ten problem instances for $k > k_{max}$. The phase transition region $[k_{min}, k_{max}]$ is then extensively tested with ten problem instances at $\max(50, \sqrt{m}/4)$ distinct, uniformly spaced values of $k \in (k_{min}, k_{max})$. When $k_{max} - k_{min} \leq 50$, every value of $k \in [k_{min}, k_{max}]$ is tested ten times. The results presented in this section were obtained in precisely the same manner as those reported in [14] where the interested reader will find further details regarding the experimental set-up, stopping criteria, and algorithm implementation.

The recovery phase transition curves define recovery regions where the associated algorithm is typically able to successfully approximate the sparse vector x^o. Algorithm selection is straightforward when faced with a problem instance in a region of the phase space where only one algorithm is typically successful. However, algorithm selection requires additional information in regions of the phase space where multiple algorithms are typically successful, i.e. in the intersection of the recovery regions for different algorithms. In order to compare the algorithms directly, the phase space is sampled on a mesh consisting of (δ, ρ) with δ from (3.21) and ρ taking the values

$$\rho \in \{\rho_j = 0.02j \mid j = 1, 2, \ldots, j^\star\} \tag{3.22}$$

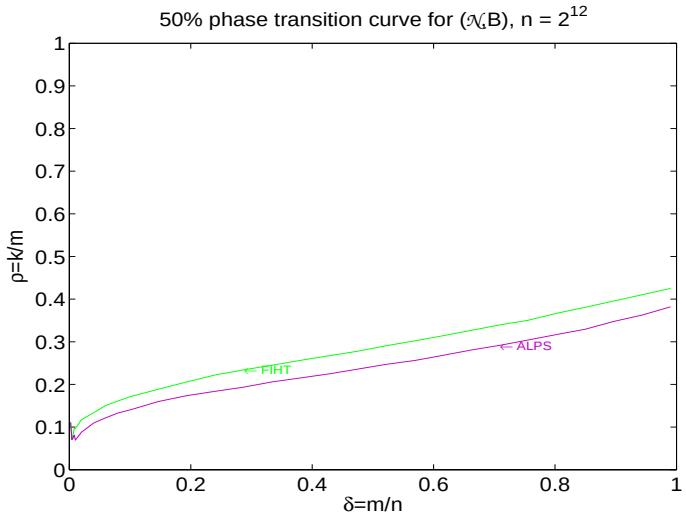
where $\rho_{j^\star}$ is the first value of ρ for which an algorithm fails to recover each of the problem instances tested. For every single (δ, ρ) in the joint recovery region, ten randomly drawn problem instances are tested for each algorithm; and the algorithm selection maps of the phase space identify the algorithms with least average computational time for successful recovery. Algorithm selection maps show consistent general trends across various problem sizes for each problem class (Mat, B) [14]; it is the general trends that are important in the selection maps rather than individual points.

3.5.3 Empirical performance comparisons between FIHT and 1-ALPS(2)

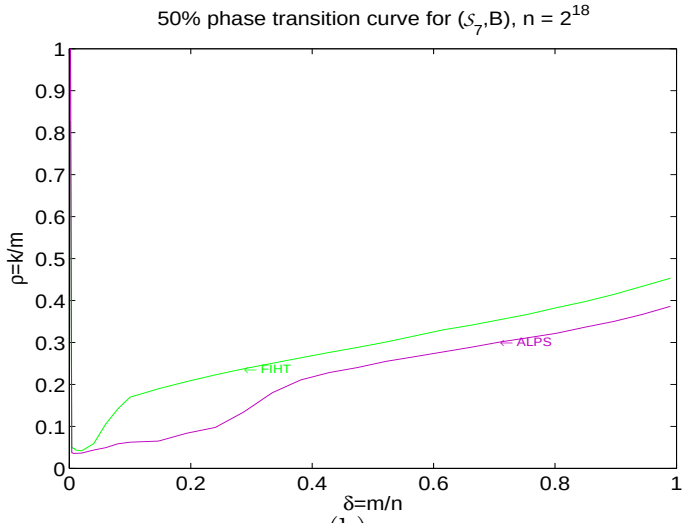
In this section, we compare the empirical performances of FIHT and 1-ALPS(2) on the sizes of the problems they are able to recover and the average computational time for them to successfully return an approximate solution.

3.5.3.1 Recovery phase transition curves

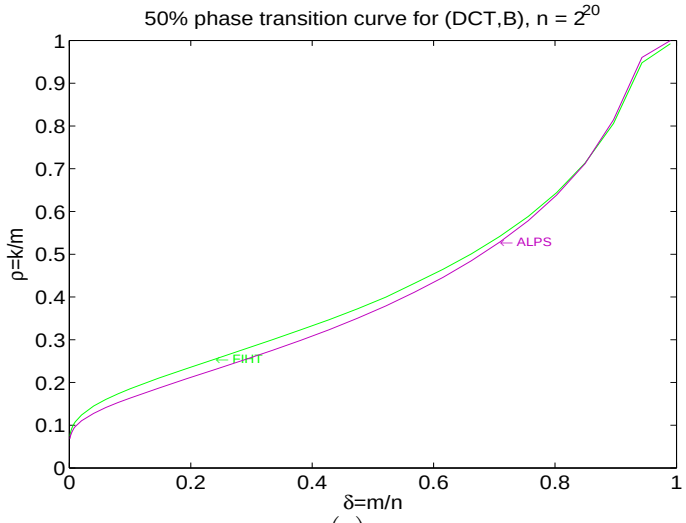
The phase transition curves of FIHT and 1-ALPS(2) are presented in Fig. 3.1. For matrix ensembles $\mathcal{N}$ and $\mathcal{S}_7$, Fig. 3.1 (a) and (b) show that the phase transition curves of FIHT is uniformly higher than the phase transition curves of 1-ALPS(2) for $0 < \delta < 1$. The discrepancy is particularly large when $0 < \delta \lesssim 0.3$ for the sparse matrix ensemble $\mathcal{S}_7$. For the DCT matrix, the phase transition curve of FIHT is higher than that of 1-ALPS(2) for $\delta \lesssim 0.85$, while 1-ALPS(2) has a slightly higher phase transition when $\delta \gtrsim 0.85$.



(a)



(b)

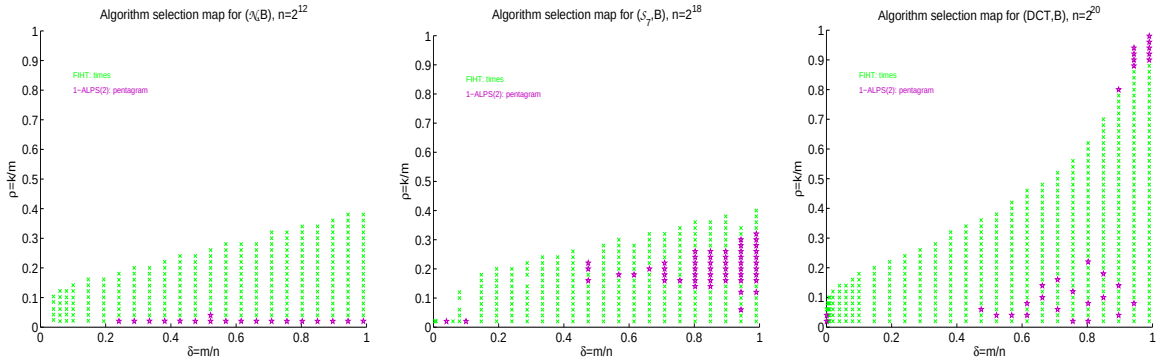


(c)

Figure 3.1: 50% recovery probability logistic regression curves of FIHT and 1-ALPS(2) for matrix ensembles (a) $\mathcal{N}$ with $n = 2^{12}$, (b) $\mathcal{S}_7$ with $n = 2^{18}$, and (c) DCT with $n = 2^{20}$.

3.5.3.2 Algorithms selection maps

Figure 3.2 presents the algorithm selection maps for FIHT and 1-ALPS(2). For Gaussian measurement matrix, FIHT marks most of the recovery regions except when ρ is extremely small. For the sparse sensing matrix, there is only a small region around $0.8 \lesssim \delta \lesssim 1,\ 0.15 \lesssim \rho \lesssim 0.25$ where 1-ALPS(2) is faster than FIHT. FIHT is essentially faster than 1-ALPS(2) for the DCT matrix when (δ, ρ) is in the intersection of their recovery regions. For completeness, Fig. 3.3 presents the least average computational time for FIHT and 1-ALPS(2) to return an approximate solution, and the recovery time ratios of FIHT and 1-ALPS(2) over the least recovery time.



(a) (b) (c)

Figure 3.2: Algorithm selection maps of FIHT and 1-ALPS(2) for matrix ensembles (a) $\mathcal{N}$ with $n = 2^{12}$, (b) $\mathcal{S}_7$ with $n = 2^{18}$, and (c) DCT with $n = 2^{20}$.

3.5.4 Empirical performance comparisons of CGIHT and other leading hard thresholding algorithms

This section compares the performance of CGIHT, CGIHT restarted, CGIHT projected, FIHT, NIHT, HTP and CSMPSP. CGIHT projected is implemented with the restarting parameter $\theta = 6$ for problem instances with $\delta \leq 0.5$ and $\theta = 3$ for $\delta > 0.5$. These values of θ were selected due to their favourable performance in preliminary tests, but have not been extensively tuned. Moreover, the values of θ have not been selected based on the RIC conditions of Thm. 3.3 where a computationally inefficient $\theta \ll 1$ would be suggested for a uniform recovery guarantee.

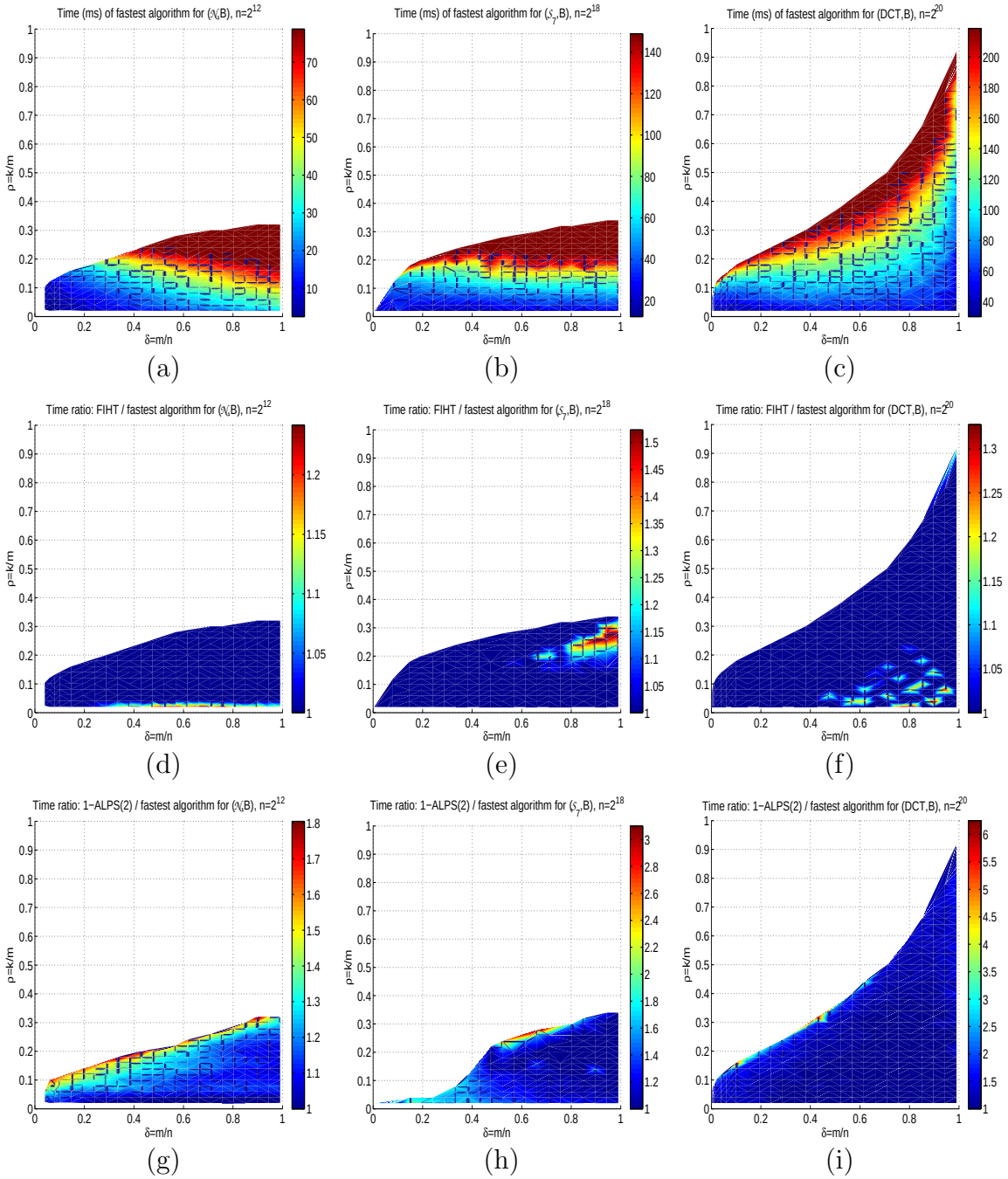
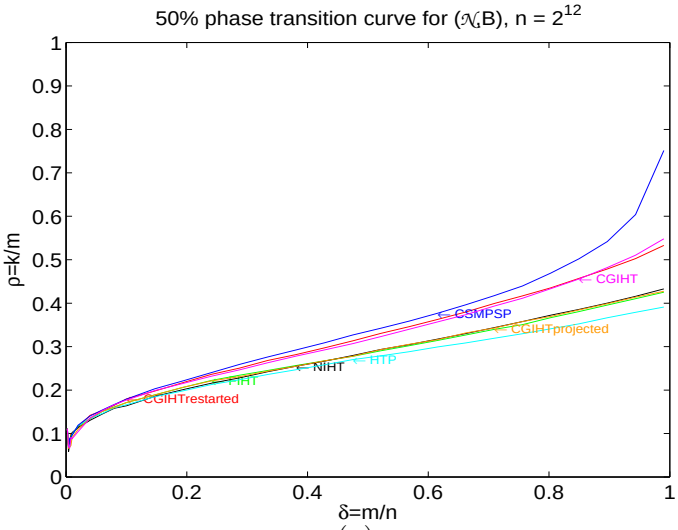
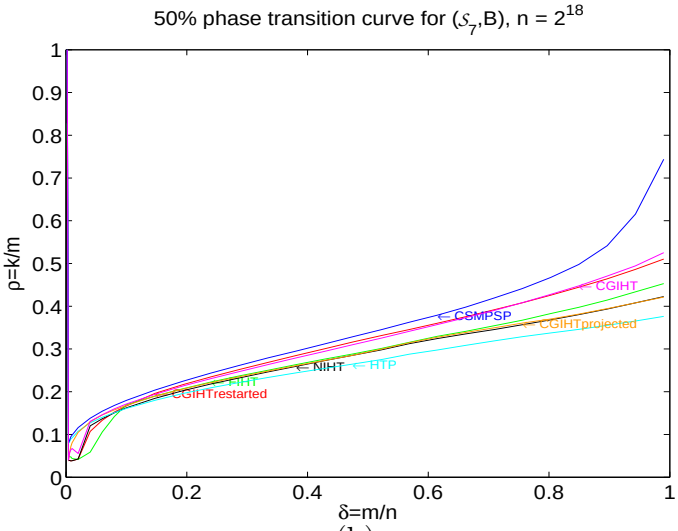


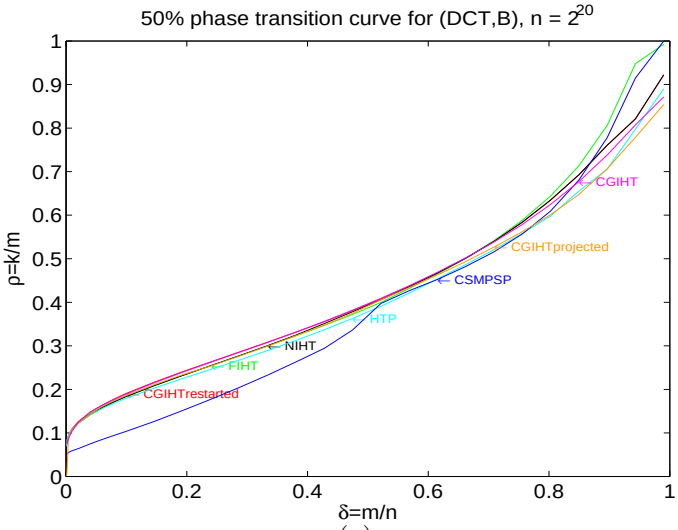
Figure 3.3: Least average computational time for FIHT and 1-ALPS(2) (a-c), and average recovery time ratio for FIHT (d-f) and 1-ALPS(2) compared to the least recovery time of those two algorithms. Matrix Ensembles: $\mathcal{N}$ with $n = 2^{12}$ (left panels), $\mathcal{S}_7$ with $n = 2^{18}$ (center panels), DCT with $n = 2^{20}$ (right panels).



(a)



(b)

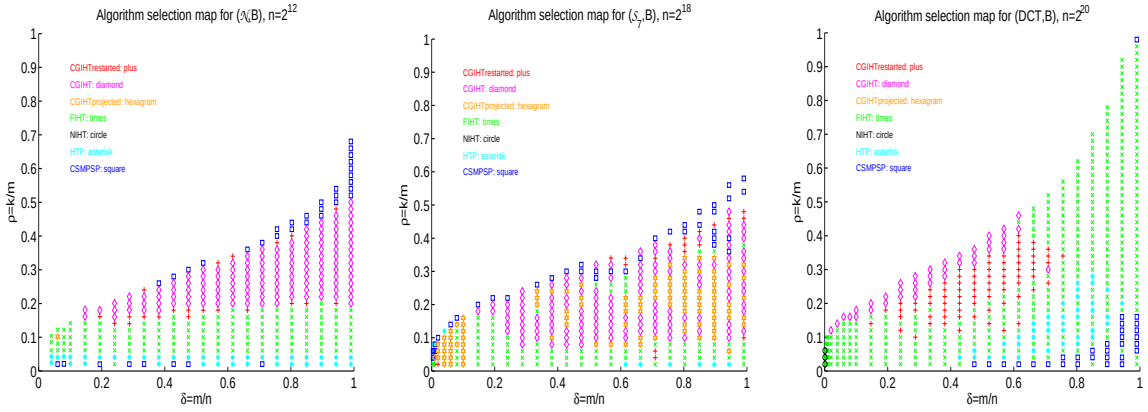


(c)

Figure 3.4: 50% recovery probability logistic regression curves for matrix ensembles (a) $\mathcal{N}$ with $n = 2^{12}$, (b) $\mathcal{S}_7$ with $n = 2^{18}$, and (c) DCT with $n = 2^{20}$.

3.5.4.1 Recovery phase transition curves

In [14], it was observed that NIHT, HTP and CSMPSP had similar recovery phase transition curves when $\delta \lesssim 0.1$ for matrix ensembles $\mathcal{N}$ and $\mathcal{S}_7$ while CSMPSP had an inferior phase transition in this region for the DCT matrix ensemble. For larger values of δ, CSMPSP typically had the highest recovery phase transition curve while NIHT and HTP continued to have similar phase transitions. In Fig. 3.4 we observe that the recovery phase transition curves for CGIHT and CGIHT restarted are superior to the phase transition curves of NIHT and HTP for essentially all $\delta \in (0, 1)$. In particular, the phase transition curves for CGIHT and CGIHT restarted are close to that of CSMPSP despite CSMPSP searching over a larger support set in each iteration. Figure 3.4 also shows that the recovery phase transition for CGIHT projected and FIHT are nearly identical to NIHT and HTP; again this is noteworthy in that FIHT searches over a larger support set when the support set changes.



(a) (b) (c)

Figure 3.5: Algorithm selection maps for matrix ensembles (a) $\mathcal{N}$ with $n = 2^{12}$, (b) $\mathcal{S}_7$ with $n = 2^{18}$, and (c) DCT with $n = 2^{20}$.

3.5.4.2 Algorithm selection maps

The algorithms selection maps for the CGIHT variants and other compared hard thresholding algorithms is presented in Fig. 3.5. For the problem class $(\mathcal{N}, B)$, when $\rho \lesssim 0.2$, the algorithm selection map in Fig. 3.5(a) recommends the use of FIHT while for $\rho \gtrsim 0.2$ CGIHT reliably recovers the sparse vector in the least time. The algorithm selection map in Fig. 3.5(b) depicts three clear regions of the phase space for the problem class $(\mathcal{S}_7, B)$. When undersampling by a factor of ten or more so that $\delta < 0.1$, CGIHT projected will successfully return a sparse solution in the least

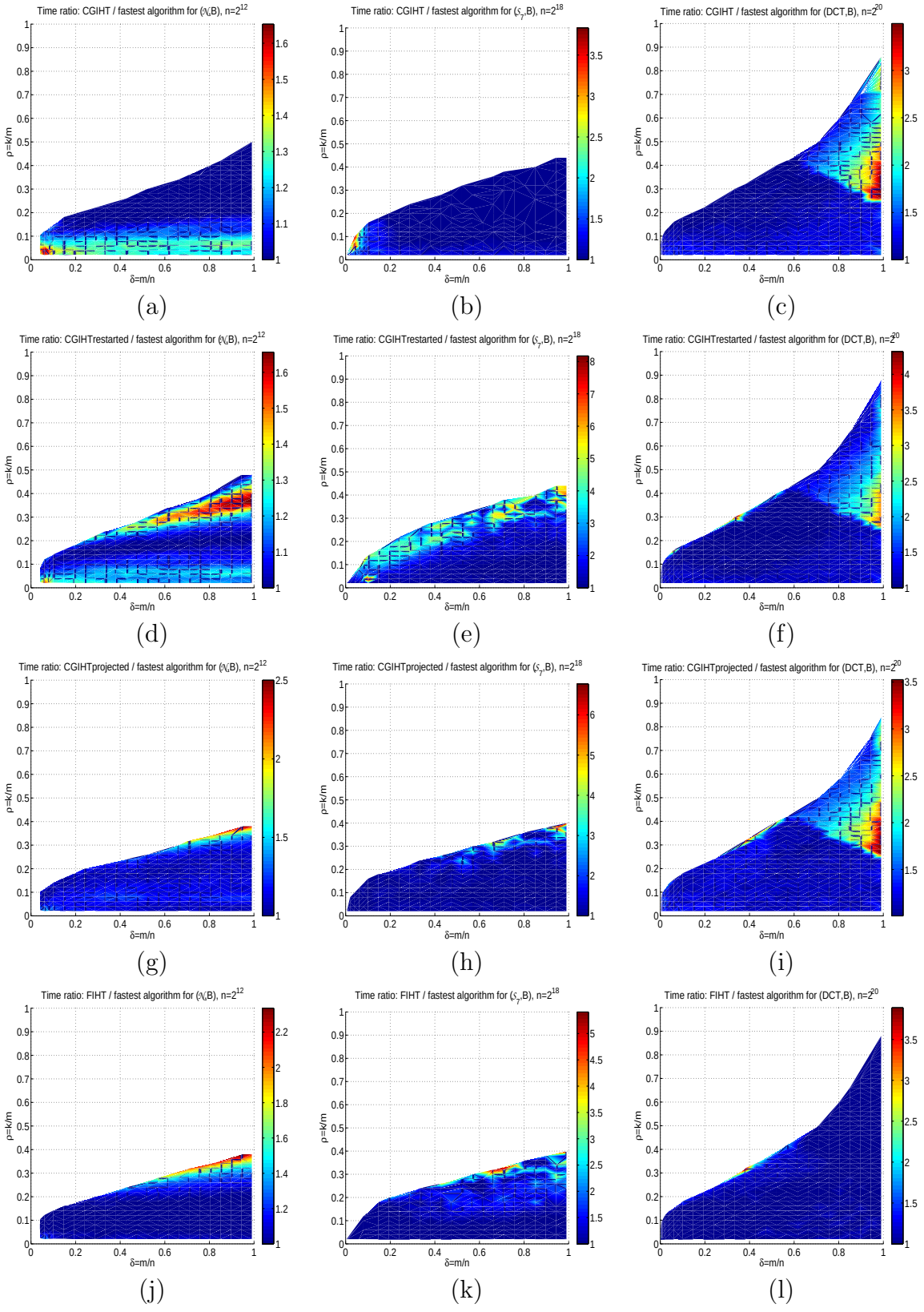


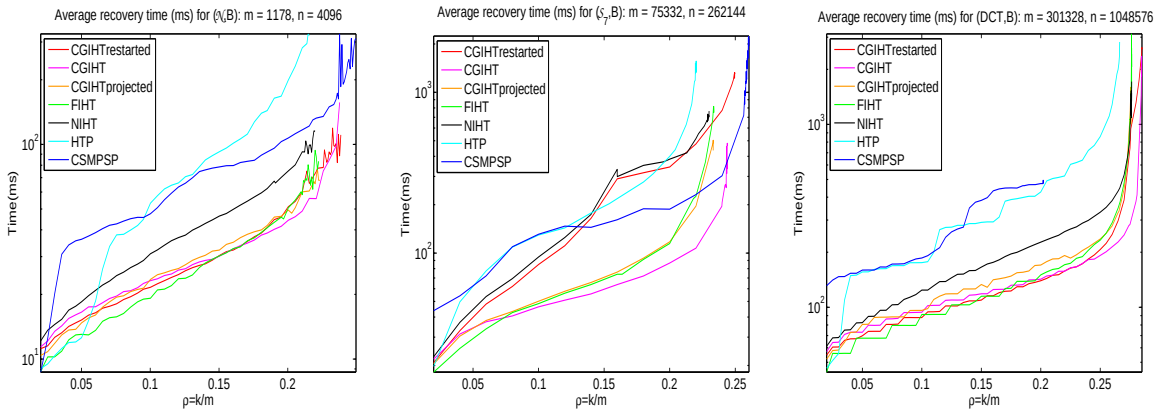
Figure 3.6: Average recovery time ratio for CGIHT (a-c), CGIHT restarted (d-f), CGIHT projected (g-i), and FIHT (j-l) compared to the fastest recovery time among all algorithms. Matrix Ensembles: $\mathcal{N}$ with $n = 2^{12}$ (left panels), $\mathcal{S}_7$ with $n = 2^{18}$ (center panels), DCT with $n = 2^{20}$ (right panels).

time. For $0.1 \lesssim \delta \lesssim 0.2$, or for $\delta \gtrsim 0.2$ with $\rho \lesssim 0.1$, FIHT will recover the measured vector in the least time. In all other cases, namely $\delta \gtrsim 0.2$ and $\rho \gtrsim 0.1$, CGIHT is recommended. For the problem class (DCT, B), FIHT is recommended through the majority of the recovery region. In the region of the phase space from $0.2 \lesssim \delta \lesssim 0.6$, CGIHT restarted or CGIHT are recommended as ρ approaches the phase transition curve of CGIHT.

3.5.4.3 Average computational time

While the algorithm selection maps identify the algorithm with the least average computational time to recover the vector, Fig. 3.6 provides a more complete picture for algorithm selection showing the ratio of a given algorithm's recovery time to the least recovery time of the algorithms tested. In particular, Fig. 3.6 shows that at each point in the (δ, ρ) phase space where recovery is possible, there is a variant of CGIHT that is either the fastest or within a few percent of being the fastest. For example, while the algorithm selection map advocates FIHT for problem class $(\mathcal{N}, B)$ and $\rho < 0.2$, Fig. 3.6 (a),(d), and (g) show that CGIHT, CGIHT restarted and CGIHT projected rarely require more than 1.25 times as long to find the solution, and never require more than 1.6 times as long as the least computational time among all algorithms. Moreover, for the problem class (DCT, B), while the algorithm selection map Fig. 3.5(c) seems to advocate FIHT throughout the majority of the phase space, the ratio maps Fig. 3.6 (c),(f) and (j) show that for $\delta < 0.7$, CGIHT, CGIHT restarted and FIHT have essentially the same recovery time. Last, the CGIHT variants and FIHT show greater differences in relative recovery time for the problem class $(\mathcal{S}_7, B)$, though with a variant of CGIHT nearly always having a smaller recovery time than FIHT. In summary, for the most interesting region of the phase space in compressed sensing, namely $\delta < 1/2$, the average recovery time for CGIHT, CGIHT restarted, CGIHT projected and FIHT are often within a few percent of one another for each problem class considered.

To further investigate the relative computational cost of the algorithms, we provide detailed information for each problem class with a single (m, n) pair. For fixed values of m, n, with a specific undersampling ratio of $\delta \approx 0.287$ (the columns in Fig. 3.5–3.6 closest to $\delta = 0.3$), a semi-log plot of the average recovery time is displayed against $\rho = k/m$ up to the maximum phase transition of the algorithms tested at the selected δ. For each problem class and each value of $\rho_j = .02 \cdot j$ with $j \leq 15$, ten problem



(a) (b) (c)

Figure 3.7: Average recovery time (ms) dependence on ρ for $\delta \approx 0.287$; (a) $\mathcal{N}$ with $n = 2^{12}$, (b) $\mathcal{S}_7$ with $n = 2^{18}$, (c) DCT with $n = 2^{20}$. Vertical scale in log(ms).

instances were tested with the average time required for recovery presented. Figure 3.7 shows two consistent trends across all three problem classes. First, the three variants of CGIHT and FIHT provide significantly improved recovery time when compared to NIHT, HTP and CSMPSP. Second, among these four accelerated hard thresholding algorithms, CGIHT demonstrates a clear computational advantage as ρ increases toward the recovery phase transition.

3.6 Empirical performance comparisons: noisy case

In this section[3], we test the stability of all the three variants of CGIHT to the additive noise, compared with the stability of NIHT, FIHT, HTP and CSMPSP. Here we consider the model $y = Ax^o + e$, where e is drawn from the sphere of radius $\epsilon\|Ax^o\|_2$ with ϵ being referred to as the "noise level". For the noisy measurements, a *problem class* is denoted by (Mat, B_ϵ), where $Mat \in \{\mathcal{N}, \mathcal{S}_7, DCT\}$ as detailed in Sec. 3.5.2, and B_ϵ indicates that the sparse vector x^o is drawn from B and the measurements are corrupted by the additive noise e with the noise level $\epsilon \in \{0.1, 0.2\}$.

The experimental set-up is essentially the same as those in Sec. 3.5.2. However, in addition to the stopping criteria described there, the algorithms also terminate if the ℓ_2 norm of the residual is identical (to single precision) to the ℓ_2 norm of one of the previous fifteen residuals. It was observed that when noise is present in the measurements, the tested algorithms frequently cycle between multiple limit points,

[3]Material in this section has been prepared for publication and in preprint [16], which is a joint authorship with J. D. Blanchard and J. Tanner whose permission has been obtained for the inclusion of the material.

and this stopping criteria is used to detect the cycle. In the empirical tests for the noisy measurements, as in [14], an algorithm is considered to be able to successfully recover x^o if it returns an approximation $\hat{x}$ that satisfies

$$\frac{\|\hat{x} - x^o\|_2}{\|x^o\|_2} < 0.001 + 2\epsilon. \tag{3.23}$$

For conciseness, only the recovery phase transition curves and algorithm selection maps are presented in this section.

3.6.1 Recovery phase transition curves

In Fig. 3.8, we compare the empirical phase transitions of the seven hard thresholding algorithms for the noisy measurements. In the most interesting region of the phase space, namely $\delta < 0.5$, the recovery phase transitions for CGIHT and CGIHT restarted are superior to NIHT, HTP, CGIHT projected and FIHT for all the three matrix ensembles. For matrix ensembles $\mathcal{N}$ and $\mathcal{S}_7$, CSMPSP has the highest phase transitions for all $\delta \in (0, 1)$ with the advantage increasing as $\delta \to 1$. As the noise level ϵ increases, the difference between these phase transition curves decreases.

Figure 3.9 presents the recovery phase transition curves of CGIHT, CGIHT restarted and CGIHT projected for both noiseless and noisy measurements. From it we observe that the phase transition curves of CGIHT variants decrease smoothly as the noise level increases. Moreover, for the moderate noise level $\epsilon = 0.1$, their phase transition curves track closely to the phase transition curves in the noise free case, with the discrepancy decreasing as $\delta \to 0$. Even for $\epsilon = 0.2$, the recovery regions capture a significant portion of the noise free recovery regions for all the three matrix ensembles, which implies the relative ℓ_2 error of the approximations from CGIHT, CGIHT restarted and CGIHT projected scales with the noise level.

3.6.2 Algorithm selection maps

In Fig. 3.10, we present the algorithm selection maps for all the three matrix ensembles and noise levels $\epsilon \in \{0.1, 0.2\}$. For the noiseless measurements, Fig. 3.5 shows FIHT and CGIHT variants share the recovery regions in the algorithm selection maps for all the three matrix ensembles. However, when the measurements are corrupted with additive noise, Fig. 3.10 shows CGIHT and CGIHT restarted mark almost all the regions of the phase space for matrix ensemble $\mathcal{N}$, and matrix ensemble $\mathcal{S}_7$ when

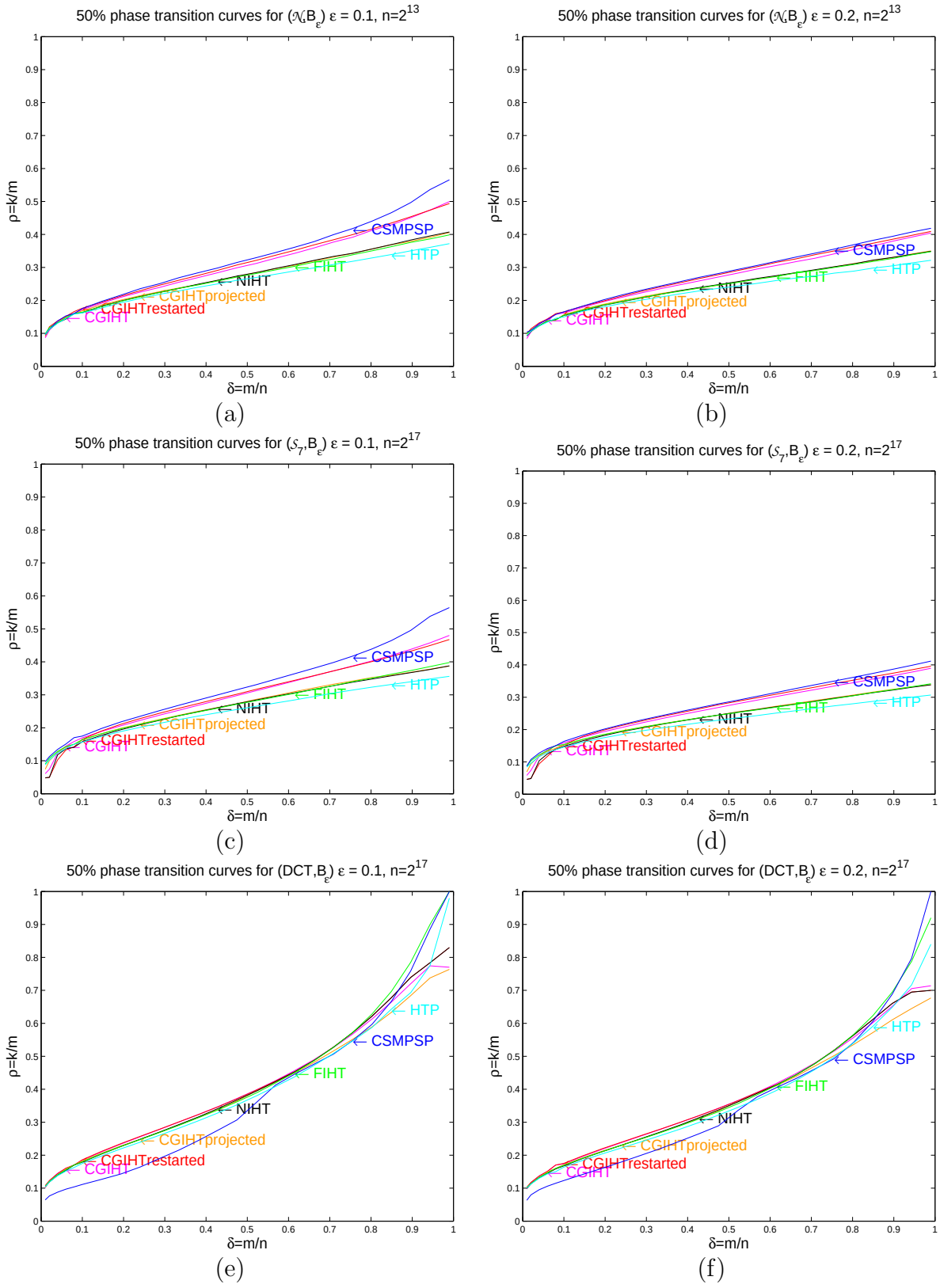


Figure 3.8: 50% recovery probability logistic regression curves for matrix ensembles $\mathcal{N}$ with $n = 2^{13}$ (top), $\mathcal{S}_7$ with $n = 2^{17}$ (middle), and DCT with $n = 2^{17}$ (bottom). (a), (c) and (e): $\epsilon = 0.1$, (b), (d) and (f): $\epsilon = 0.2$.

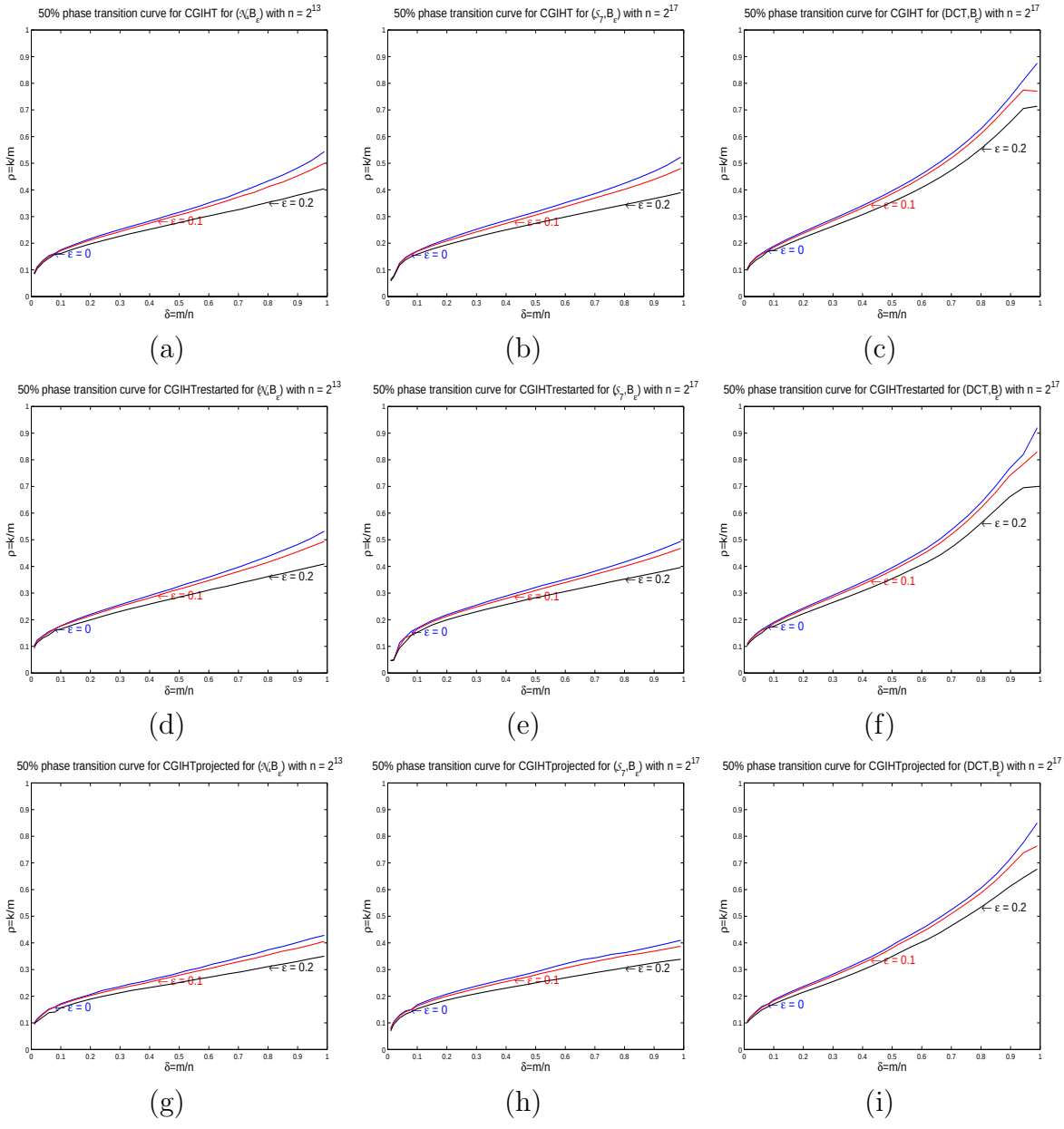


Figure 3.9: 50% recovery probability logistic regression curves for noiseless and noisy measurement ($\epsilon = 0.1,\ 0.2$) and matrix ensembles $\mathcal{N}$ with $n = 2^{13}$ (left panels), $\mathcal{S}_7$ with $n = 2^{17}$ (center panels), and DCT with $n = 2^{17}$ (right panels). (a)-(c): CGIHT, (d)-(f): CGIHT restarted, and (g)-(i): CGIHT projected.

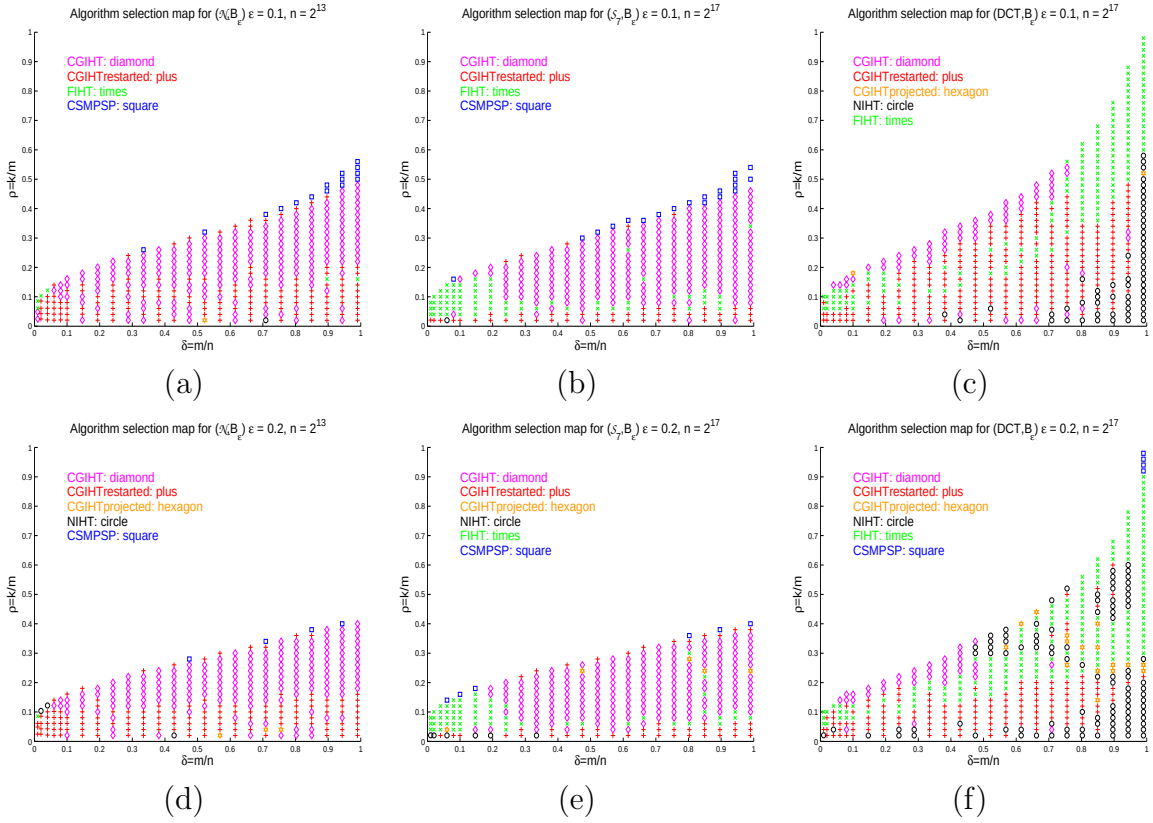


Figure 3.10: Algorithm selection maps for matrix ensembles $\mathcal{N}$ with $n = 2^{13}$ (left panels), $\mathcal{S}_7$ with $n = 2^{17}$ (center panels), and DCT with $n = 2^{17}$ (right panels). (a)-(c): $\epsilon = 0.1$, (d)-(f): $\epsilon = 0.2$.

$\delta \geq 0.2$. For matrix ensemble DCT and $\delta \leq 0.5$, CGIHT and CGIHT restarted continue to dominate the selection maps. FIHT (and the ALPS methods) are based on the optimal first order method for convex problems [78] which is known to lose its accelerated convergence rate in the presence of noise [39].

3.7 Proofs of Theorems 3.3 and 3.5

3.7.1 Proof of Theorem 3.3

The proof of Thm. 3.3 is partitioned into three steps: a technical lemma, bounds on the descent stepsizes and orthogonalization weights, and the analysis of the algorithm. The first two steps are presented as Lems 3.6 and 3.7. In CGIHT, each new approximation could possibly depend on all previous iterations. This will ultimately lead to a three term recurrence relation on the approximation error.

Lemma 3.6. Suppose $c_0, \eta, \tau_1, \tau_2 \geq 0$ and let $\mu = \frac{1}{2}\left(\tau_1 + \sqrt{\tau_1^2 + 4\tau_2}\right)$. Assume $c_1 \leq \mu c_0 + \eta$ and define $c_l = \tau_1 c_{l-1} + \tau_2 c_{l-2} + \eta$ for $l \geq 2$. If $\tau_1 + \tau_2 < 1$, then $\mu < 1$ and

$$c_l \leq \mu^l c_0 + \eta \sum_{i=0}^{l-1} \mu^i. \tag{3.24}$$

Proof. If $\tau_1 + \tau_2 < 1$, then $\mu < \frac{1}{2}(\tau_1 + \tau_2 + 1) < 1$. By assumption, (3.24) is valid for c_1. Assume it is also valid for c_j with $j \leq l-1$. Then, since $\mu = \tau_1 + \frac{\tau_2}{\mu}$,

$$c_l \leq \tau_1 \left(\mu^{l-1} c_0 + \eta \sum_{i=0}^{l-2} \mu^i\right) + \frac{\tau_2}{\mu}\mu \left(\mu^{l-2} c_0 + \eta \sum_{i=0}^{l-3} \mu^i\right) + \eta \leq \mu^l c_0 + \eta \sum_{i=0}^{l-1} \mu^i.$$

□

Central to the performance of CGIHT is the calculation of stepsizes and orthogonalization weights of the support set restricted conjugate gradient method. The stepsizes α_l are uniformly bounded near one with the same RIC bounds as the NIHT stepsize. The relative orthogonality is measured by β_l. When the support set has changed, β_l is defined to be zero, otherwise each β_l is uniformly bounded near zero. In the process of establishing these bounds on the stepsizes, we also bound the spectrum of a projection operator which appears regularly in this type of analysis.

Lemma 3.7. By the definition of RICs of the measurement matrix A, the stepsize is uniformly bounded by

$$\frac{1}{1+\delta_k} \leq \alpha_l \leq \frac{1}{1-\delta_k} \tag{3.25}$$

and the orthogonalization coefficients are uniformly bounded by

$$|\beta_l| \leq \frac{2\delta_k(1+\delta_k)}{(1-\delta_k)^2}. \tag{3.26}$$

Furthermore, if $Q, S \subset \{1, \ldots, n\}$ are two index sets and $ck = |Q \cup S|$, then for any z

$$\left\| \mathtt{Proj}_Q \left((I - \alpha_l A^* A) \mathtt{Proj}_S(z) \right) \right\|_2 \leq \frac{2\delta_{ck}}{1-\delta_k} \left\| \mathtt{Proj}_S(z) \right\|_2 . \tag{3.27}$$

Proof. When $\Gamma_l \neq \Gamma_{l-1}$ the stepsize α_l is the same as proposed by NIHT, and is bounded directly as

$$\frac{1}{1+\delta_k} \leq \alpha_l = \frac{\left\| \mathtt{Proj}_{\Gamma_l}(r_l) \right\|_2^2}{\left\| A\mathtt{Proj}_{\Gamma_l}(r_l) \right\|_2^2} \leq \frac{1}{1-\delta_k}$$

by inverting the standard RIC bounds of A. If $\Gamma_l = \Gamma_{l-1}$, we utilize two important inequalities

$$\left\| A\mathtt{Proj}_{\Gamma_{l-1}} (p_{l-1}) \right\|_2 \leq \left\| A\mathtt{Proj}_{\Gamma_{l-1}} (r_{l-1}) \right\|_2 , \tag{3.28}$$

$$\left\| \mathtt{Proj}_{\Gamma_{l-1}} (r_{l-1}) \right\|_2 \leq \left\| \mathtt{Proj}_{\Gamma_{l-1}} (p_{l-1}) \right\|_2 . \tag{3.29}$$

The first inequality (3.28) follows from noting that $A\mathtt{Proj}_{\Gamma_{l-1}} (p_{l-1})$ is orthogonal to $A\mathtt{Proj}_{\Gamma_{l-1}} (p_{l-2})$ by construction (orthogonalization after the application of A is referred to as conjugate orthogonal); consequently, multiplying $r_{l-1} = p_{l-1} - \beta_{l-1} p_{l-2}$ by A and noting the orthogonality gives $\| A\mathtt{Proj}_{\Gamma_{l-1}} (r_{l-1}) \|_\Gamma^2 = \| A\mathtt{Proj}_{\Gamma_{l-1}} (p_{l-1}) \|_2^2 + \beta_{l-1}^2 \| A\mathtt{Proj}_{\Gamma_{l-1}} (p_{l-2}) \|_2^2 \geq \| A\mathtt{Proj}_{T_{l-1}} (p_{l-1}) \|^2$. The second inequality (3.29) follows from applying the Cauchy-Schwartz inequality to

$$\begin{aligned} \left\langle \mathtt{Proj}_{\Gamma_{l-1}} (r_{l-1}) , \mathtt{Proj}_{\Gamma_{l-1}} (r_{l-1}) \right\rangle &= \left\langle \mathtt{Proj}_{\Gamma_{l-1}} (r_{l-1}) , \mathtt{Proj}_{\Gamma_{l-1}} (p_{l-1} - \beta_{l-1} p_{l-2}) \right\rangle \\ &= \left\langle \mathtt{Proj}_{\Gamma_{l-1}} (r_{l-1}) , \mathtt{Proj}_{\Gamma_{l-1}} (p_{l-1}) \right\rangle , \end{aligned} \tag{3.30}$$

where the last equality follows from the conjugate gradient property (see Thm. 3.1) that

$$\left\langle \mathtt{Proj}_{\Gamma_{l-1}} (r_{l-1}) , \mathtt{Proj}_{\Gamma_{l-1}} (p_{l-2}) \right\rangle = 0. \tag{3.31}$$

To establish the lower bound, we utilize (3.28) to observe

$$\frac{1}{\alpha_l} = \frac{\left\| A\mathtt{Proj}_{\Gamma_{l-1}}(p_{l-1}) \right\|_2^2}{\left\| \mathtt{Proj}_{\Gamma_{l-1}}(r_{l-1}) \right\|_2^2} \leq \frac{\left\| A\mathtt{Proj}_{\Gamma_{l-1}}(r_{l-1}) \right\|_2^2}{\left\| \mathtt{Proj}_{\Gamma_{l-1}}(r_{l-1}) \right\|_2^2} \leq 1 + \delta_k. \tag{3.32}$$

The upper bound follows from (3.29) since

$$\alpha_l = \frac{\left\| \mathtt{Proj}_{\Gamma_{l-1}}(r_{l-1}) \right\|_2^2}{\left\| A\mathtt{Proj}_{\Gamma_{l-1}}(p_{l-1}) \right\|_2^2} \leq \frac{\left\| \mathtt{Proj}_{\Gamma_{l-1}}(p_{l-1}) \right\|^2}{\left\| A\mathtt{Proj}_{\Gamma_{l-1}}(p_{l-1}) \right\|_2^2} \leq \frac{1}{1-\delta_k}. \tag{3.33}$$

Thus from (3.32) and (3.33), when $\Gamma_l = \Gamma_{l-1}$, α_l is also bounded by

$$\frac{1}{1+\delta_k} \leq \alpha_l \leq \frac{1}{1-\delta_k}. \tag{3.34}$$

For any index sets $Q, S \subset \{1, \ldots, n\}$, let $A_{Q\cup S}$ be the submatrix formed by the columns of A indexed by the set $Q \cup S$. Then for any l and any vector z, the $|Q|$ nonzeros in the vector $\mathtt{Proj}_Q\left((I - \alpha_l A^*A)\mathtt{Proj}_S(z)\right)$ are a subset of the $|Q \cup S|$ nonzeros in the vector $\left(I - \alpha_l A^*_{Q\cup S}A_{Q\cup S}\right)\mathtt{Proj}_S(z)$, so that

$$\begin{aligned} \left\| \mathtt{Proj}_Q\left((I - \alpha_l A^*A)\mathtt{Proj}_S(z)\right) \right\|_2 &\leq \left\| I - \alpha_l A^*_{Q\cup S}A_{Q\cup S} \right\|_2 \cdot \left\| \mathtt{Proj}_S(z) \right\|_2 \\ &\leq \frac{2\delta_{ck}}{1-\delta_k} \left\| \mathtt{Proj}_S(z) \right\|_2 \end{aligned} \tag{3.35}$$

where $ck = |Q \cup S|$. The bound on the operator $I - \alpha_l A^*_{Q\cup S}A_{Q\cup S}$ in terms of the RICs of A follows from Def. 1.3 and (3.34) as in [11, Lem. 5].

The orthogonalization factor β_l is equal to zero when $\Gamma_l \neq \Gamma_{l-1}$, and can be bounded when $\Gamma_l = \Gamma_{l-1}$ by the using alternative formula

$$\beta_l = -\frac{\left\langle \mathtt{Proj}_{\Gamma_l}(r_l), \mathtt{Proj}_{\Gamma_l}\left(A^*A\mathtt{Proj}_{\Gamma_l}(p_{l-1})\right) \right\rangle}{\left\| A\mathtt{Proj}_{\Gamma_l}(p_{l-1}) \right\|_2^2} \tag{3.36}$$

and expressing $\mathtt{Proj}_{\Gamma_l}(r_l)$ in terms of prior search directions. When $\Gamma_l = \Gamma_{l-1}$

$$\begin{aligned} \mathtt{Proj}_{\Gamma_l}(r_l) &= \mathtt{Proj}_{\Gamma_l}(r_{l-1}) - \alpha_{l-1}\mathtt{Proj}_{\Gamma_l}\left(A^*A\mathtt{Proj}_{\Gamma_l}(p_{l-1})\right) \\ &= \mathtt{Proj}_{\Gamma_l}(p_{l-1}) - \beta_{l-1}\mathtt{Proj}_{\Gamma_l}(p_{l-2}) - \alpha_{l-1}\mathtt{Proj}_{\Gamma_l}\left(A^*A\mathtt{Proj}_{\Gamma_l}(p_{l-1})\right) \\ &= \mathtt{Proj}_{\Gamma_l}\left((I - \alpha_{l-1}A^*A)\mathtt{Proj}_{\Gamma_l}(p_{l-1})\right) - \beta_{l-1}\mathtt{Proj}_{\Gamma_l}(p_{l-2}). \end{aligned} \tag{3.37}$$

where the first equality follows by substituting $x_l = \mathtt{Proj}_{\Gamma_l}(x_{l-1} + \alpha_{l-1}p_{l-1})$ into r_l, the second equality by substituting $r_{l-1} = p_{l-1} - \beta_{l-1}p_{l-2}$, and the final equality by

rearrangement. Inserting (3.37) for $\texttt{Proj}_{\Gamma_l}(r_l)$ into (3.36) gives

$$\begin{aligned}
|\beta_l| &= \frac{\left|\left\langle \texttt{Proj}_{\Gamma_l}\left((I-\alpha_{l-1}A^*A)\texttt{Proj}_{\Gamma_l}(p_{l-1})\right), \texttt{Proj}_{\Gamma_l}\left(A^*A\texttt{Proj}_{\Gamma_l}(p_{l-1})\right)\right\rangle\right|}{\|A\texttt{Proj}_{\Gamma_l}(p_{l-1})\|_2^2} \\
&\leq \frac{\left\|\texttt{Proj}_{\Gamma_l}\left((I-\alpha_{l-1}A^*A)\texttt{Proj}_{\Gamma_l}(p_{l-1})\right)\right\|_2 \left\|\texttt{Proj}_{\Gamma_l}\left(A^*A\texttt{Proj}_{\Gamma_l}(p_{l-1})\right)\right\|_2}{\left\|A\texttt{Proj}_{\Gamma_l}(p_{l-1})\right\|_2^2} \\
&\leq \frac{2\delta_k}{1-\delta_k}\left\|\texttt{Proj}_{\Gamma_l}(p_{l-1})\right\|_2 \frac{(1+\delta_k)\left\|\texttt{Proj}_{\Gamma_l}(p_{l-1})\right\|_2}{(1-\delta_k)\left\|\texttt{Proj}_{\Gamma_l}(p_{l-1})\right\|_2^2} \\
&= \frac{2\delta_k(1+\delta_k)}{(1-\delta_k)^2} \qquad (3.38)
\end{aligned}$$

where the first equality follows by noting that $A\texttt{Proj}_{\Gamma_l}(p_{l-1})$ and $A\texttt{Proj}_{\Gamma_l}(p_{l-2})$ are orthogonal, the first inequality follows from the Cauchy-Schwarz inequality, and the second inequality from (3.27) with $Q = S = \Gamma_l$. □

With Lems. 3.6 and 3.7, the proof of Thm. 3.3 is similar to the proofs of other hard thresholding algorithms. Following Foucart's general outline [48], the approximation error is bounded by observing that the hard thresholding operator produces the best k-sparse approximation to the current update. The proof is complicated by the fact that the current search direction can depend on all previous search directions. This is handled by establishing a recurrence relation and invoking Lem. 3.6.

Proof of Theorem 3.3. The proof begins following the proof of an analogous theorem for IHT [48]. Note that

$$\begin{aligned}
\|x_l - w_{l-1}\|_2^2 &= \|x_l - x^o + x^o - w_{l-1}\|_2^2 \\
&= \|x_l - x^o\|_2^2 + \|x^o - w_{l-1}\|_2^2 + 2\langle x_l - x^o, x^o - w_{l-1}\rangle \\
&\leq \|x^o - w_{l-1}\|_2^2 \qquad (3.39)
\end{aligned}$$

where the inequality follows from x_l being, by definition, the k-sparse vector nearest to w_{l-1}. Canceling $\|x^o - w_{l-1}\|^2$ in (3.39) and rearranging gives the inequality

$$\|x_l - x^o\|_2^2 \leq 2\langle x_l - x^o, w_{l-1} - x^o\rangle = 2\left\langle x_l - x^o, \texttt{Proj}_{\Gamma_l \cup T}(w_{l-1} - x^o)\right\rangle,$$

where $\texttt{Proj}_{\Gamma_l \cup T}(\cdot)$ reflects the sparsity of $x_l - x^o$. Applying the Cauchy-Schwarz inequality and cancelling a power of $\|x_l - x^o\|_2$ gives

$$\|x_l - x^o\| \leq 2\left\|\texttt{Proj}_{\Gamma_l \cup \Gamma}(w_{l-1} - x^o)\right\|_2. \qquad (3.40)$$

In anticipation of substituting $w_{l-1} = x_{l-1} + \alpha_{l-1}p_{l-1}$ into (3.40), we note that p_{l-1} can be expressed as

$$p_{l-1} = r_{l-1} + \sum_{j=0}^{l-2} r_j \left(\Pi_{q=j+1}^{l-1}\beta_q\right). \tag{3.41}$$

Equation (3.40) can then be expressed purely in terms of x_j for $j \le l$ by substituting $w_{l-1} = x_{l-1}+\alpha_{l-1}p_{l-1}$ into (3.40) with p_{l-1} given by (3.41) and replacing all instances of r_j with $A^*A(x^o - x_j) + A^*e$,

$$\begin{aligned}
\|x_l - x^o\|_2 &\le 2\left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}\left(x_{l-1} - x^o + \alpha_{l-1}r_{l-1} + \alpha_{l-1}\sum_{j=0}^{l-2} r_j\left(\Pi_{q=j+1}^{l-1}\beta_q\right)\right)\right\|_2 \\
&\le 2\left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}\left((I-\alpha_{l-1}A^*A)(x_{l-1}-x^o)\right)\right\|_2 + 2|\alpha_{l-1}|\left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}\left(A^*e\right)\right\|_2 \\
&\quad +2|\alpha_{l-1}|\sum_{j=0}^{l-2}\left(\Pi_{q=j+1}^{l-1}\beta_q\right)\left(\left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}\left(A^*A(x^o-x_j)\right)\right\|_2 + \left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}\left(A^*e\right)\right\|_2\right)
\end{aligned} \tag{3.42}$$

We proceed by bounding each of the terms in the final inequality of (3.42). Let $\epsilon_\alpha = \frac{1}{1-\delta_k}$, $\epsilon_\beta = \frac{2\delta_k(1+\delta_k)}{(1-\delta_k)^2}$, and $\epsilon_\lambda = \frac{2\delta_{3k}}{1-\delta_k}$ denote the upper bounds established by Lem. 3.7. Then we have

$$\left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}\left((I-\alpha_{l-1}A^*A)(x_{l-1}-x^o)\right)\right\|_2 \le \epsilon_\lambda\|x_{l-1}-x^o\|_2, \tag{3.43}$$

$$|\alpha_{l-1}|\left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}(A^*e)\right\|_2 \le \epsilon_\alpha\left(1+\delta_{2k}\right)^{1/2}\|e\|_2, \tag{3.44}$$

$$|\alpha_{l-1}|\sum_{j=0}^{l-2}\left(\Pi_{q=j+1}^{l-1}\beta_q\right)\left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}(A^*A(x^o-x_j))\right\|_2 \le \epsilon_\alpha(1+\delta_{3k})\sum_{j=0}^{l-2}\epsilon_\beta^{l-j-1}\|x^o-x_j\|_2, \tag{3.45}$$

$$|\alpha_{l-1}|\sum_{j=0}^{l-2}\left(\Pi_{q=j+1}^{l-1}\beta_q\right)\left\|\mathtt{Proj}_{\Gamma_l\cup\Gamma}(A^*e)\right\|_2 \le \epsilon_\alpha(1+\delta_{2k})^{1/2}\|e\|_2\sum_{j=0}^{l-2}\epsilon_\beta^{l-j-1}. \tag{3.46}$$

Applying (3.43)–(3.46) to (3.42), gathering like terms, and reindexing the sums,

$$\|x_l-x^o\|_2 \le 2\epsilon_\lambda\|x_{l-1}-x^o\|_2+2\epsilon_\alpha(1+\delta_{3k})\sum_{j=1}^{l-1}\epsilon_\beta^j\|x_{l-j-1}-x^o\|_2+2\epsilon_\alpha(1+\delta_{2k})^{1/2}\|e\|_2\sum_{j=0}^{l-1}\epsilon_\beta^j. \tag{3.47}$$

A simplified argument focused only the first iterate yields

$$\|x_1 - x^o\|_2 \le 2\epsilon_\lambda\|x_0 - x^o\|_2 + 2\epsilon_\alpha(1+\delta_{2k})^{1/2}\|e\|_2. \tag{3.48}$$

Seeking a bound on $\|x_l - x^o\|_2$ purely in terms of $\|x_0 - x^o\|_2$, define $c_0 = \|x_0 - x^o\|_2$, $c_1 = 2\epsilon_\lambda c_0 + \xi\|e\|_2$, and recursively define

$$c_l = 2\epsilon_\lambda c_{l-1} + 2\epsilon_\alpha(1+\delta_{3k})\sum_{j=1}^{l-1}\epsilon_\beta^j c_{l-j-1} + 2\epsilon_\alpha(1+\delta_{2k})^{1/2}\|e\|_2\sum_{j=0}^{l-1}\epsilon_\beta^j \qquad \text{for } l \geq 2. \quad (3.49)$$

Note that (3.48) shows that $\|x_1 - x^o\|_2 \leq c_1$ and (3.47) ensures $\|x_j - x^o\|_2 \leq c_j$ for $j \geq 2$. By computing $c_l - \epsilon_\beta c_{l-1}$ and isolating c_l, (3.49) can be rewritten as a three term recurrence relation

$$c_l = (2\epsilon_\lambda + \epsilon_\beta)c_{l-1} + 2(\epsilon_\alpha(1+\delta_{3k}) - \epsilon_\lambda)\epsilon_\beta c_{l-2} + 2\epsilon_\alpha(1+\delta_{2k})^{1/2}\|e\|_2. \quad (3.50)$$

Now define

$$\tau_1 = 2\epsilon_\lambda + \epsilon_\beta, \quad \tau_2 = 2\left(\epsilon_\alpha(1+\delta_{3k}) - \epsilon_\lambda\right)\epsilon_\beta, \quad \text{and} \quad \xi = 2\epsilon_\alpha(1+\delta_{2k})^{1/2},$$

so that $c_l = \tau_1 c_{l-1} + \tau_2 c_{l-2} + \xi\|e\|_2$. Let $\mu = \frac{1}{2}\left(\tau_1 + \sqrt{\tau_1^2 + 4\tau_2}\right)$ and observe that since $2\epsilon_\lambda < \tau_1 < \mu$, $c_1 \leq \mu c_0 + \xi\|e\|_2$. If $\tau_1 + \tau_2 < 1$, then Lem. 3.6 implies $\mu < 1$ and

$$\|x_l - x^o\|_2 \leq c_l \leq \mu^l c_0 + \xi\|e\|_2\sum_{i=0}^{l-1}\mu^i \leq \mu^l c_0 + \frac{\xi}{1-\mu}\|e\|_2 = \mu^l\|x_0 - x^o\|_2 + \frac{\xi}{1-\mu}\|e\|_2. \quad (3.51)$$

Finally, note that μ and ξ are equivalent to the definitions in (3.11) and the sufficient condition $\tau_1 + \tau_2 < 1$ is satisfied when

$$\frac{2\delta_{3k}(5+\delta_k)}{(1-\delta_k)^2} < 1. \quad (3.52)$$

□

3.7.2 Proof of Theorem 3.5

Proof of Theorem 3.5. There are three building blocks in Alg. 12: a) computing v_{l-1} as a linear combination of x_{l-1} and x_{l-2} (Step 1 and 2), b) a regular projected gradient descent step starting from v_{l-1} (Step 3 to 8), and c) one more steepest descent restricted on the current support set (Step 9 to 11), and each block can be processed independently. To process the first block, similar to that in [18], we utilize the observation that the residual decreases from x_{l-1} to v_{l-1}; and the rest of the proof resembles that for (N)IHT in [48].

First, rather than attempting to bound τ_{l-1} which varies in each iteration, the following observation can be drawn from the expression for τ_{l-1} in Alg. 12,

$$\|y - Av_{l-1}\|_2 \leq \|y - Ax_{l-1}\|_2. \tag{3.53}$$

Moreover, one has

$$\begin{aligned} \|v_{l-1} - x^o\|_2 &\leq (1-\delta_{3k})^{-1/2} \left(\|y - Av_{l-1}\|_2 + \|e\|_2\right), & (3.54)\\ \|y - Ax_{l-1}\|_2 &\leq (1+\delta_{2k})^{1/2} \|x_{l-1} - x^o\|_2 + \|e\|_2, & (3.55) \end{aligned}$$

by the RIC bounds of A and the fact $v_{l-1} - x^o$ is a sparse vector of cardinality at most $3k$ and $x_{l-1} - x^o$ is a sparse vector of cardinality at most $2k$. Consequently,

$$\|v_{l-1} - x^o\|_2 \leq \left(\frac{1+\delta_{2k}}{1-\delta_{3k}}\right)^{1/2} \|x_{l-1} - x^o\|_2 + 2(1-\delta_{3k})^{-1/2} \|e\|_2 . \tag{3.56}$$

The contraction of the second block follows a similar argument for NIHT, but starting with a sparse vector v_{l-1} of cardinality at most $2k$. A complete proof is presented here for the reader's convenience. Let $\tilde{x}_l$ be the intermediate approximate solution obtained from Step 8 in Alg. 12. Then repeating the argument in (3.39) gives

$$\|\tilde{x}_l - x^o\|_2^2 \leq 2 \langle \tilde{x}_l - x^o, w_{l-1} - x^o \rangle . \tag{3.57}$$

Substituting $w_{l-1} = v_{l-1} + \tilde{\alpha}_{l-1}\tilde{r}_{l-1}$ into (3.57) leads to

$$\begin{aligned} \|\tilde{x}_l - x^o\|_2^2 &\leq 2 \langle \tilde{x}_l - x^o, v_{l-1} + \tilde{\alpha}_{l-1}\tilde{r}_{l-1} - x^o \rangle \\ &= 2 \langle \tilde{x}_l - x^o, v_{l-1} + \tilde{\alpha}_{l-1} A^* (y - Av_{l-1}) - x^o \rangle \\ &= 2 \langle \tilde{x}_l - x^o, v_{l-1} - x^o \rangle + 2 \langle \tilde{x}_l - x^o, \tilde{\alpha}_{l-1} A^* (Ax + e - Av_{l-1}) \rangle \\ &= 2 \langle \tilde{x}_l - x^o, v_{l-1} - x^o \rangle - 2 \langle A(\tilde{x}_l - x^o), \tilde{\alpha}_{l-1} A (v_{l-1} - x^o) \rangle \\ &\quad + 2\tilde{\alpha}_{l-1} \langle A(\tilde{x}_l - x^o), e \rangle \\ &= 2 \langle \tilde{x}_l - x^o, v_{l-1} - x^o \rangle - 2 \langle A_{\Gamma_{4k}} (\tilde{x}_l - x^o), \tilde{\alpha}_{l-1} A_{\Gamma_{4k}} (v_{l-1} - x^o) \rangle \\ &\quad + 2 \langle \tilde{\alpha}_{l-1} A_{\Gamma_{2k}} (\tilde{x}_l - x^o), e \rangle \\ &= 2 \left\langle \tilde{x}_l - x^o, \left(I - \tilde{\alpha}_{l-1} A^*_{\Gamma_{4k}} A_{\Gamma_{4k}}\right)(v_{l-1} - x^o) \right\rangle \\ &\quad + 2\tilde{\alpha}_{l-1} \langle A_{\Gamma_{2k}} (\tilde{x} - x^o), e \rangle \\ &\leq 2 \|\tilde{x}_l - x^o\|_2 \cdot \left\|I - \tilde{\alpha}_{l-1} A^*_{\Gamma_{4k}} A_{\Gamma_{4k}}\right\|_2 \cdot \|v_{l-1} - x^o\|_2 \\ &\quad + 2\tilde{\alpha}_{l-1}(1+\delta_{2k})^{1/2} \|\tilde{x}_l - x^o\|_2 \cdot \|e\|_2 , \end{aligned} \tag{3.58}$$

where in the fourth equality $\Gamma_{4k} = \texttt{Support}\left(\tilde{x}_l\right) \cup \texttt{Support}\left(v_{l-1}\right) \cup \texttt{Support}\left(\tilde{x}\right)$ and $\Gamma_{2k} = \texttt{Support}\left(\tilde{x}_l\right) \cup \texttt{Support}\left(x\right)$. Cancelling one power of $\left\|\tilde{x}_l - \tilde{x}\right\|_2$ from both sides gives

$$\left\|\tilde{x}_l - x^o\right\|_2 \leq 2\left\|I - \tilde{\alpha}_{l-1} A_{\Gamma_{4k}}^* A_{\Gamma_{4k}}\right\|_2 \cdot \left\|v_{l-1} - x^o\right\|_2 + 2\tilde{\alpha}_{l-1}(1+\delta_{2k})^{1/2}\left\|e\right\|_2. \quad (3.59)$$

The descent stepsize $\tilde{\alpha}_{l-1}$ can be bounded as

$$\frac{1}{1+\delta_{2k}} \leq \tilde{\alpha}_{l-1} = \frac{\left\|\texttt{Proj}_{\tilde{\Gamma}_{l-1}}\left(\tilde{r}_{l-1}\right)\right\|_2^2}{\left\|A\texttt{Proj}_{\tilde{\Gamma}_{l-1}}\left(\tilde{r}_{l-1}\right)\right\|_2^2} \leq \frac{1}{1-\delta_{2k}}, \quad (3.60)$$

since in Alg. 12, $\tilde{\Gamma}_{l-1}$ is a support set with cardinality at most $2k$. Moreover, the spectral norm of $I - \tilde{\alpha}_{l-1} A_{\Gamma_{4k}}^* A_{\Gamma_{4k}}$ can be bounded as

$$\left\|I - \tilde{\alpha}_{l-1} A_{\Gamma_{4k}}^* A_{\Gamma_{4k}}\right\|_2 \leq \frac{2\delta_{4k}}{1-\delta_{2k}}. \quad (3.61)$$

Therefore,

$$\left\|\tilde{x}_l - x^o\right\|_2 \leq \frac{4\delta_{4k}}{1-\delta_{2k}}\left\|v_{l-1} - x^o\right\|_2 + \frac{2(1+\delta_{2k})^{1/2}}{1-\delta_{2k}}\left\|e\right\|_2. \quad (3.62)$$

From the fact the residual decreases in the third block, similarly to (3.56), one has

$$\left\|x_l - x^o\right\|_2 \leq \left(\frac{1+\delta_{2k}}{1-\delta_{2k}}\right)^{1/2}\left\|\tilde{x}_l - x^o\right\|_2 + 2(1-\delta_{2k})^{-1/2}\left\|e\right\|_2 \quad (3.63)$$

Combining (3.56), (3.62) and (3.63) together gives

$$\left\|x_l - x^o\right\|_2 \leq \mu\left\|x_{l-1} - x^o\right\|_2 + \xi\left\|e\right\|_2, \quad (3.64)$$

with

$$\mu = \frac{4\delta_{4k}\left(1+\delta_{2k}\right)}{\left(1-\delta_{2k}\right)^{3/2}\left(1-\delta_{3k}\right)^{1/2}}, \quad \xi = \frac{8\delta_{4k}\left(1+\delta_{2k}\right)^{1/2}}{\left(1-\delta_{2k}\right)^{3/2}\left(1-\delta_{3k}\right)^{1/2}} + \frac{4}{\left(1-\delta_{2k}\right)^{3/2}}. \quad (3.65)$$

Finally, one has

$$\left\|x_l - x^o\right\|_2 \leq \mu^l\left\|x_0 - x^o\right\|_2 + \frac{\xi}{1-\mu}\left\|e\right\|_2. \quad (3.66)$$

□

Chapter 4

Conjugate Gradient Iterative Hard Thresholding for Matrix Completion

In this chapter, we introduce the CGIHT algorithms for matrix completion, which are natural extensions of CGIHT for compressed sensing. The chapter is outlined as follows[1]. The formal descriptions of CGIHT for matrix completion are presented in Sec. 4.1, with a recovery guarantee provided for the projected variant. In Sec. 4.2, we present FIHT for matrix completion, which is a variant of Matrix ALPS II in [64]. In Sec. 4.3, we compare CGIHT with other hard thresholding algorithms. Section 4.4 contains the proof of the main theorem.

4.1 CGIHT: from compressed sensing to matrix completion

We begin our discussion of CGIHT for matrix completion with its simplest variant, Alg. 16, which mirrors Alg. 9. In each iteration of CGIHT the current estimate X_{l-1} is updated along the direction P_{l-1}, using a stepsize α_{l-1}, followed by hard thresholding to the matrix of rank at most r that is nearest in the Frobenius norm. As in Alg. 9, the search direction P_{l-1} is selected to be exactly the residual R_0 in iteration $l = 1$, and is otherwise selected to be the residual R_{l-1} projected to be conjugate orthogonal to the past search direction when restricted to the current estimate of subspace U_{l-1}. As with Alg. 9, Alg. 16 lacks the conjugate gradient property of past search directions

[1]Material in this chapter has been prepared for publication and in preprint [15], which is a joint authorship with J. D. Blanchard and J. Tanner whose permission has been obtained for the inclusion of the material.

remaining mutually orthogonal. In fact as the subspaces are continuously varying, there will typically be no two past subspaces U_l and U_{l-1} which are identical, and consequently only the past two search directions will remain orthogonal. Despite lacking orthogonalization over longer sequences of iterates, Alg. 16 nearly always has superior performance to the other variant of CGIHT for matrix completion in terms of both the problem size (m, n, p, r) it is able to recover and being able to recover the measured matrix in less time than the projected variant of CGIHT.

Algorithm 16 CGIHT for matrix completion

Initialization: Set $P_{-1} = 0$, $W_{-1} = \mathcal{A}^*(y)$,
$U_0 = \texttt{PrincipalLeftSingularVectors}_r(W_{-1})$, $X_0 = \texttt{Proj}_{U_0}(W_{-1})$, $l = 1$.

Iteration: During iteration l, **do**

1: $R_{l-1} = \mathcal{A}^*\left(y - \mathcal{A}(X_{l-1})\right)$ (gradient descent direction)

2: if $l = 1$,
$\beta_{l-1} = 0$, (set orthogonalization weight)
else
$\beta_{l-1} = -\frac{\left\langle \mathcal{A}\left(\texttt{Proj}_{U_{l-1}}(R_{l-1})\right), \mathcal{A}\left(\texttt{Proj}_{U_{l-1}}(P_{l-2})\right)\right\rangle}{\left\|\mathcal{A}\left(\texttt{Proj}_{U_{l-1}}(P_{l-2})\right)\right\|_2^2}$ (compute orthogonalization weight)

3: $P_{l-1} = R_{l-1} + \beta_{l-1} P_{l-2}$ (define the search direction)

4: $\alpha_{l-1} = \frac{\left\langle \texttt{Proj}_{U_{l-1}}(R_{l-1}), \texttt{Proj}_{U_{l-1}}(P_{l-1})\right\rangle}{\left\|\mathcal{A}\left(\texttt{Proj}_{U_{l-1}}(P_{l-1})\right)\right\|_2^2}$ (local steepest descent stepsize)

5: $W_{l-1} = X_{l-1} + \alpha_{l-1} P_{l-1}$ (gradient or conjugate gradient descent update)

6: $U_l = \texttt{PrincipalLeftSingularVectors}(W_{l-1})$ (proxy to the subspace)

7: $X_l = \texttt{Proj}_{U_l}(W_{l-1})$ (solution update)

Output: X_l

As stated previously, the matrix completion problem (1.11) differs from (1.1) fundamentally in that the subspace of low rank matrices is a continuously varying manifold whereas the space of k-sparse vectors is composed of $\binom{n}{k}$ finite dimensional subspaces corresponding to the possible support sets. As a consequence, the subspaces U_{l-1} and U_{l-2} will typically never be exactly the same and there is no exact analogue of Alg. 10 for matrix completion. However, the relative residual restarting condition of CGIHT projected, Alg. 11, can be extended to matrix completion, which we refer to as CGIHT projected for matrix completion, see Alg. 17. Similar to Alg. 11, CGIHT projected for matrix completion corresponds to nonlinear restarted conjugate gradient where restarting occurs when $\left\|R_{l-1} - \texttt{Proj}_{U-1}(P_{l-1})\right\|_F / \left\|\texttt{Proj}_{U-1}(R_{l-1})\right\|_F$ is sufficiently large. This restarting condition corresponds to the component of the search direction P_{l-1} contained in the image space of X_{l-1} minus R_{l-1} being small

in the Frobenius norm when compared with the size of the residual contained in the image space of X_{l-1}. Unlike CGIHT for matrix completion, Alg. 16, which has no tuning parameters, CGIHT projected has a restarting parameter θ controlling the computational effort used to decrease the residual on a given subspace.

Algorithm 17 CGIHT projected for matrix completion

Initialization: Set $W_{-1} = \mathcal{A}^*(y)$, $U_0 = \texttt{PrincipalLeftSingularVectors}_r(W_{-1})$, $X_0 = \texttt{Proj}_{U_0}(W_{-1})$, $R_0 = \mathcal{A}^*\left(y - \mathcal{A}(X_0)\right)$, $P_0 = R_0$, RestartFlag = 1, set restart parameter θ, $l = 1$.

Iteration: During iteration l, **do**

1: if $\frac{\left\|R_{l-1} - \texttt{Proj}_{U_{l-1}}(P_{l-1})\right\|_F}{\|\texttt{Proj}_{U_{l-1}}(R_{l-1})\|_F} > \theta$

$\quad$ RestartFlag = 1 (set restart flag)

$\quad$ $\alpha_{l-1} = \frac{\left\|\texttt{Proj}_{U_{l-1}}(R_{l-1})\right\|_F^2}{\left\|\mathcal{A}\left(\texttt{Proj}_{U_{l-1}}(R_{l-1})\right)\right\|_2^2}$ (local steepest descent stepsize)

$\quad$ $W_{l-1} = X_{l-1} + \alpha_{l-1} R_{l-1}$ (gradient descent update)

else

$\quad$ RestartFlag = 0 (set restart flag)

$\quad$ $\alpha_{l-1} = \frac{\left\|\texttt{Proj}_{U_{l-1}}(R_{l-1})\right\|_F^2}{\left\|\mathcal{A}\left(\texttt{Proj}_{U_{l-1}}(P_{l-1})\right)\right\|_2^2}$ (local steepest descent stepsize)

$\quad$ $W_{l-1} = X_{l-1} + \alpha_{l-1}\texttt{Proj}_{U_{l-1}}(P_{l-1})$ (conjugate gradient update)

2: $U_l = \texttt{PrincipalLeftSingularVectors}(W_{l-1})$ (proxy to the subspace)

3: $X_l = \texttt{Proj}_{U_l}(W_{l-1})$ (solution update)

4: $R_l = \mathcal{A}^*\left(y - \mathcal{A}(X_l)\right)$ (compute the residual)

5: if RestartFlag = 1,

$\quad$ $P_l = R_l$ (gradient descent direction)

else

$\quad$ $\beta_l = \frac{\left\|\texttt{Proj}_{U_l}(R_l)\right\|_F^2}{\left\|\texttt{Proj}_{U_l}(R_{l-1})\right\|_F^2}$ (compute orthogonalization weight)

$\quad$ $P_l = R_l + \beta_l \texttt{Proj}_{U_l}(P_{l-1})$ (conjugate gradient descent direction)

Output: X_l

CGIHT projected has a uniform recovery guarantee for the low rank matrix approximation problem provided the sensing operator $\mathcal{A}(\cdot)$ has sufficiently small RICs for low rank matrices.

Theorem 4.1 (Recovery guarantee for CGIHT projected for matrix completion, Alg. 17.)**.** Let $\mathcal{A}(\cdot)$ be a linear map from $\mathbb{R}^{m\times n}$ to $\mathbb{R}^p$ with $p < mn$, and $y = \mathcal{A}(X^o) + e$ for any X^o with $\text{rank}(X) \leq r$. Define the following constants in terms of the RICs of $\mathcal{A}(\cdot)$ and the restarting parameter c:

$$\mu = \frac{4(1+c)\delta_{3r}}{1-\delta_r}, \quad \theta_0 = \frac{2c\delta_{3r}}{1+\delta_{2r}}, \quad \text{and} \quad \xi = 2(1+\theta_0)\frac{(1+\delta_{2r})^{1/2}}{1-\delta_r}.$$

Then provided $\mu < 1$, the iterates of Alg. 17 with restarting condition $\theta < \theta_0$ satisfy

$$\|X_l - X^o\|_F \leq \mu^l \|X_0 - X^o\|_F + \frac{\xi}{1-\mu}\|e\|_2. \tag{4.1}$$

Proof. See Sec. 4.4 for a proof. □

As in Thm. 2.1, Thm. 4.1 considers the model $y = \mathcal{A}(X^o) + e$ where $\text{rank}(X^o) \leq r$. In this model X^o is typically viewed as the rank r matrix which minimizes $\|y - \mathcal{A}(X)\|_2$ and e as the model misfit. Alternatively X^o can be viewed as a rank r matrix measured by $\mathcal{A}(\cdot)$ and that y is contaminated by additive noise; though, in this perspective, Thm. 4.1 does not consider any particular model for e. Theorem 4.1 implies that if $e = 0$, CGIHT projected for matrix completion will recover the measured rank r matrix to arbitrary accuracy. As the manifold of rank r matrices is continuously varying, Alg. 17 will never exactly identify the correct image space, and does not exactly correspond to a traditional numerical linear algebra routine for computing a low rank matrix approximation. However, in iterations where the RestartFlag = 0, Alg. 17 does correspond to solving for the low rank approximation with a specified image space corresponding to that of U_{l-1}.

The restarting parameter θ plays an important role in CGIHT projected for matrix completion. This theorem requires $\theta < \frac{2c\delta_{3r}}{1+\delta_{2r}}$ and that $\mu = \frac{4(1+c)\delta_{3r}}{1-\delta_r} < 1$. First, consider two extreme cases of $c = 0$ or $c = \infty$. When $c = 0$, the restarting condition is met at every iteration and the algorithm is identical to NIHT for matrix completion. Moreover, the sufficient condition $\mu = \frac{4\delta_{3r}}{1-\delta_r} < 1$ is identical to the sufficient condition for NIHT for matrix completion, see Thm. 2.1. At the other extreme, the algorithm will never restart and instead will project the observations y onto the column space spanned by the r principal left singular vectors of $\mathcal{A}^*(y)$. This is the one step hard thresholding algorithm. For the values of $c \in (0, \infty)$, CGIHT traces between matrix completion versions of NIHT and HTP. As c increases, the sufficient condition $\mu < 1$ is satisfied by a smaller set of linear operators $\mathcal{A}(\cdot)$ while the algorithm is permitted to take more conjugate gradient steps on each subspace. Eventually, the subspace restricted conjugate gradient method will force $\|\texttt{Proj}_{U_{l-1}}(R_{l-1})\|_F$ to be small enough that the restarting criteria is satisfied. In other words, the parameter c determines an error tolerance for the projection of y onto the subspace spanned by U_{l-1}.

4.2 Fast iterative hard thresholding for matrix completion

In [64], the authors adapted the ALPS family of algorithms [33] to the matrix completion setting, and introduced the Matrix ALPS family of algorithms for matrix completion. Out of the Matrix ALPS algorithms, Matrix ALPS II, Alg. 18, is observed to have the greatest overall computational efficacy [64]. FIHT for matrix completion, Alg. 18, is a variant of Matrix ALPS II, which differs from Matrix ALPS II in its fourth step, where Matrix ALPS II uses a subspace of dimension of at least $2r$ by including another r dimensional subspace from the residual in its third step. However, as the subspace is continuously varying, the column subspaces for X_{l-1} and X_{l-2} will typically not be identical to each other, thus FIHT typically searches over a $2r$ dimensional subspace, while Matrix ALPS II typically searches over a $3r$ dimensional subspace. Since Matrix ALPS II requires an additional singular value decomposition in each iteration to include the extra r dimensional subspace which is computationally expensive, only FIHT is considered in this chapter.

Algorithm 18 FIHT: a variant of Matrix ALPS II for matrix completion

Initialization: Set $X_{-1} = 0$, $X_0 = \texttt{Proj}_{U_0}(\mathcal{A}^*(y))$, $R_0 = \mathcal{A}^*(y - \mathcal{A}(X_0))$, $l = 1$

Iteration: During iteration l, **do**

1: if $l = 1$,
$\tau_{l-1} = 0$ (set momentum stepsize)
else
$\tau_{l-1} = \frac{\langle y - \mathcal{A}(X_{l-1}), \mathcal{A}(X_{l-1} - X_{l-2})\rangle}{\|\mathcal{A}(X_{l-1} - X_{l-2})\|_2^2}$ (calculate momentum stepsize)

2: $V_{l-1} = X_{l-1} + \tau_{l-1}(X_{l-1} - X_{l-2})$ (new extrapolated point)

3: $R_{l-1} = \mathcal{A}^*(y - \mathcal{A}(V_{l-1}))$ (gradient descent direction)

4: $\tilde{U}_{l-1} = \texttt{LeftSingularVectors}(V_{l-1})$ (subspace for FIHT)
or
$\tilde{U}_{l-1} = \tilde{U}_{l-1} \cup \texttt{PrincipalLeftSingularVectors}_r\left(\texttt{Proj}_{\tilde{U}_{l-1}^{\perp}}(R_{l-1})\right)$
(subspace for Matrix ALPS II)

5: $\alpha_{l-1} = \frac{\left\|\texttt{Proj}_{\tilde{U}_{l-1}}(R_{l-1})\right\|_F^2}{\left\|\mathcal{A}\left(\texttt{Proj}_{\tilde{U}_{l-1}}(R_{l-1})\right)\right\|_2^2}$ (local steepest descent stepsize)

6: $W_{l-1} = V_{l-1} + \alpha_{l-1} R_{l-1}$ (steepest descent update)

7: $U_l = \texttt{PrincipalLeftSingularVectors}_r(W_{l-1})$ (proxy to the subspace)

8: $X_l = \texttt{Proj}_{U_l}(W_{l-1})$ (solution update)

Output: X_l

In [64], the authors proved the RIC based recovery guarantee for Matrix ALPS II with the constant momentum stepsize, that is when τ_{l-1} is fixed. In this chapter, we

will show that FIHT with the optimal variable τ_{l-1} also has a near optimal recovery guarantee in terms of RICs of the sensing operator. The proof equally applies to Matrix ALPS II with variable τ_{l-1}.

Theorem 4.2 (Recovery guarantee for FIHT for matrix completion, Alg. 18.)**.** Let $\mathcal{A}(\cdot)$ be a linear map from $\mathbb{R}^{m\times n}$ to $\mathbb{R}^p$ with $p < mn$, and $y = \mathcal{A}(X^o) + e$ for any X^o with $\text{rank}(X) \leq r$. Define the following constants in terms of the RICs of $\mathcal{A}(\cdot)$:

$$\mu = \frac{4\delta_k \left(1+\delta_{2k}\right)^{1/2}}{\left(1-\delta_{2k}\right)\left(1-\delta_{3k}\right)^{1/2}}, \quad \text{and} \quad \xi = \frac{8\delta_{4k}}{\left(1-\delta_{2k}\right)\left(1-\delta_{3k}\right)^{1/2}} + \frac{2\left(1+\delta_{2k}\right)^{1/2}}{1-\delta_{2k}}.$$

Then provided $\mu < 1$, the iterates of FIHT for matrix completion (Alg. 18) satisfy

$$\|X_l - X^o\|_F \leq \mu^l \|X_0 - X^o\|_F + \frac{\xi}{1-\mu}\|e\|_2. \tag{4.2}$$

Proof. The proof is based on two key observations: a) V_{l-1} is a low rank matrix with $\text{rank}(V_{l-1}) \leq 2r$, and b) the residual is non-increasing from X_{l-1} to V_{l-1}. See Thm. 3.5 in Chapter 3 for an analogous proof. □

4.3 Empirical performance comparisons

The experimental set-up in this section is essentially the same as that in Sec. 2.2.1. The testing was conducted through a massive distribution of problem instances at the IRIDIS High Performance Computing Facility provided by the e-Infrastructure South Centre for Innovation. Recall that a problem instance is indexed by four parameters (see Tab. 2.1):

$$\{(Op, N),\ (m, n),\ p \text{ or } \delta,\ r \text{ or } \rho\},$$

where $Op \in \{\mathcal{G}, \mathcal{E}\}$ represents the sensing operator ensemble, N represents the $m \times n$ rank r matrix ensemble formed as in (2.10), and $\delta = p/mn$ and $\rho = r(m+n-r)/p$ denote the undersampling and oversampling ratios respectively.

The empirical average case performance of NIHT for recovering a low rank matrix from near the minimum number of measurements is reported in Chapter 2; and in that chapter we also compare NIHT with other state-of-the-art methods demonstrating the superiority of NIHT in terms of phase transitions for both sensing ensembles $\mathcal{G}$ and $\mathcal{E}$. Due to the already large computational time of more than 5.6 CPU years required to generate the matrix completion data presented in this chapter, we restrict our testing to the matrix completion variants of CGIHT, CGIHT projected, NIHT and FIHT

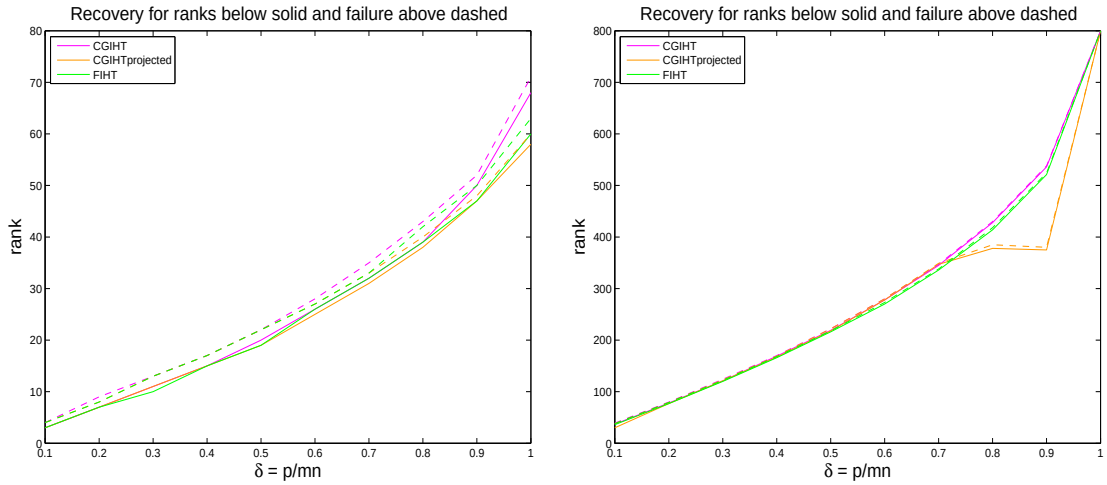
for the square matrices with $m = n = 80$ for Gaussian sensing and $m = n = 800$ for entry sensing. Rectangular matrices are observed to have lower phase transitions; see for instance Chapter 2 or [86, 2]. Indirect comparisons can be drawn with other algorithms by contrasting the results presented here and those in Chapter 2 and [64].

The algorithms terminate using the same stopping criteria detailed in Sec. 2.2.1. Preliminary testing suggested our choice of the restarting parameter θ in CGIHT projected, which was set to 5 for sensing ensemble $\mathcal{G}$; and for sensing ensemble $\mathcal{E}$ it was set to 3 if $\delta \leq 0.5$, otherwise it was set to 10. In this section, an algorithm is considered to have successfully recovered the matrix X^o when the algorithm returns a matrix $\hat{X}$ that satisfies (2.12) in Chapter 2.

4.3.1 Recovery phase transition curves

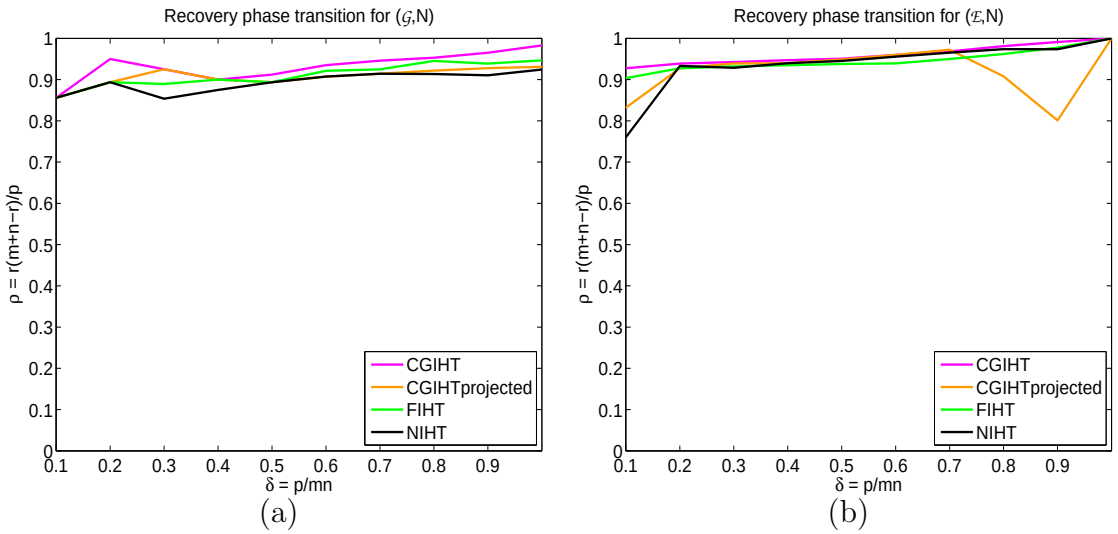
This section investigates when the matrix completion algorithms can successfully recover a rank r matrix from p linear measurements with p proportional to $r(m + n - r)$. For each problem class $(Oper, N)$ and (m, n) pair, we conduct tests with the undersampling ratio $\delta = p/(mn)$ taking ten equispaced values from 0.1 to 1.0. As in Chapter 2, for each triple (m, n, p), we start from a sufficiently small rank r that the algorithm can successfully recover each sensed matrix in all ten randomly drawn problem instances; we then increase the rank until the algorithm fails to recover the sensed matrix in each of ten random problem instances. We refer to the largest value of r for which the algorithm succeeded in each of the ten tests as r_{min} and the smallest value of r for which the algorithm failed in each of the ten tests at r_{max}. The values of r_{min} and r_{max} for CGIHT, CGIHT projected and FIHT are displayed in Fig. 4.1 for Gaussian sensing with $m = n = 80$ (left panel) and entry sensing with $m = n = 800$ (right panel). Figure 4.2 displays the empirical phase transitions of the four tested algorithms, with the vertical axis ρ being calculated from $(r_{min} + r_{max})/2$.

For the most interesting region of the phase space, namely $\delta < 1/2$, the recovery phase transition curves of CGIHT and CGIHT projected are always greater than 0.8 indicating both algorithms are able to successfully recover the randomly generated rank r matrices with the number of measurements $p = C \cdot r(m + n - r)$ for $C \leq 1.25$. For problem class $(\mathcal{G}, N)$ with $\delta < 1/2$, the recovery phase transition curves for CGIHT and CGIHT projected are at least as high as the phase transition curve for FIHT, which in turn is either equivalent or superior to that of NIHT. For all $\delta \in (0.1, 1)$, none of the other algorithms have a phase transition curve superior



(a) (b)

Figure 4.1: For each rank above the dashed lines the algorithm failed to recover the sensed matrix in each of ten randomly drawn tests per rank, whereas for each rank below the solid lines it succeeded in recovering the sensed matrix in each of ten randomly dawn tests per rank. The vertical axis is the rank of the sensed matrix and the horizontal axis is the undersampling ratio δ. Left panel: $\mathcal{G}$ with $n = 80$; right panel: $\mathcal{E}$ with $n = 800$.



(a) (b)

Figure 4.2: Phase transitions of the tested algorithms with horizontal axis δ and vertical axis ρ where ρ is calculated using the average of r_{min} and r_{max}. (a) $\mathcal{G}$ with $m = n = 80$, (b) $\mathcal{E}$ with $m = n = 800$.

to the phase transition curve of CGIHT for matrix completion (Alg. 16). Likewise for $(\mathcal{E}, N)$, CGIHT for matrix completion has the highest recovery phase transition curve for all values of $\delta \in (0.1, 1)$, although the phase transition curves for all four algorithms are very similar. For $\delta > 0.7$, the observed decrease in the phase transition curve for CGIHT projected is an artifact of the restarting parameter θ, and is likely caused by excessively solving the subproblem, and in doing so causing the subspace restricted conjugate gradient projections to converge to a non-optimal local minimum; decreasing the restarting parameter θ to 1 for $\delta > 0.7$ increases the phase transition curve to be that of NIHT, but at the cost of increased recovery time.

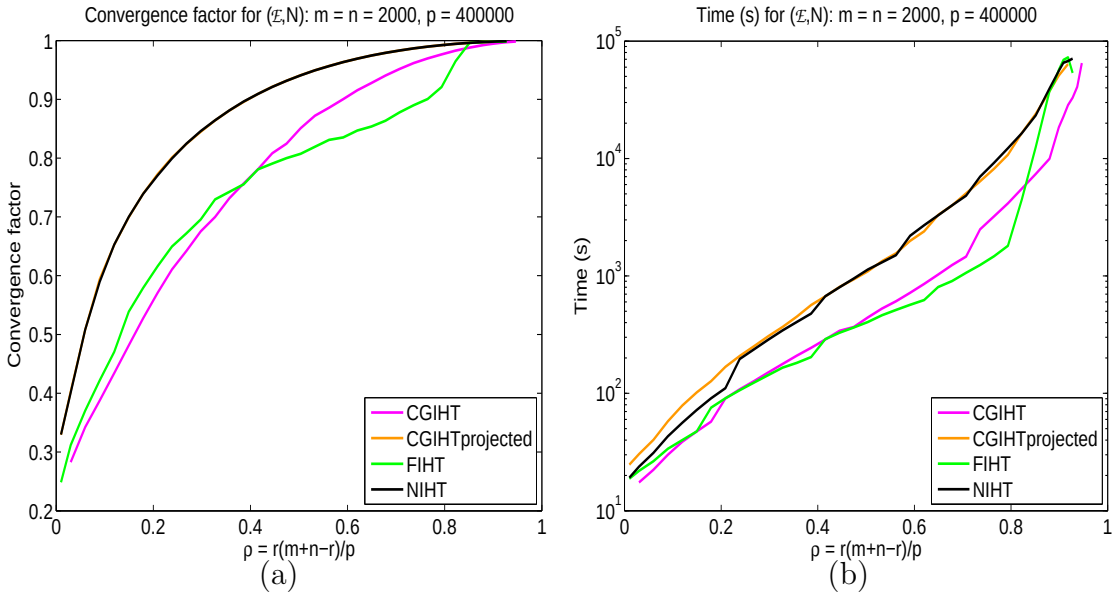


Figure 4.3: Average convergence factors and average recovery time (s) of CGIHT, CGIHT projected, FIHT and NIHT for problem class $(\mathcal{E}, N)$ with $m = n = 2000$, $p = mn/10$ and r ranging from 1 to 96. Horizontal axis ρ defined in (2.9) and convergence factors of left panel defined in (2.11). Vertical scale for (b) is log(s).

4.3.2 Recovery time dependence on ρ for $\delta = 1/10$

In this section, we explore the average recovery time and per iteration asymptotic convergence factors for CGIHT, CGIHT projected, FIHT and NIHT for matrix completion with a fixed value of $\delta = 1/10$. These experiments are performed with $m = n = 2000$, $p = mn/10$, and the rank r sampled from the set $\{r_j = 3j : j = 1, 2, \ldots, j^\star\}$ where $r_{j^\star}$ is the smallest rank for which an algorithm fails to recover a single rank $r_{j^\star}$ matrix in 100 random problem instances. Due to the computational burden of such large-scale testing, the results in this section were exclusively conducted for the

problem class $(\mathcal{E}, N)$. Figure 4.3(a) displays the average convergence factors computed according to (2.11) for each algorithm's successful recovery of a rank r matrix. Figure 4.3(b) provides a semi-log plot of the average computational recovery time.

The empirical results provided in Fig. 4.3 establish a computational advantage for CGIHT and FIHT when compared to NIHT and CGIHT projected which have indistinguishable convergence factors in Fig. 4.3(a). For the smallest ranks with $\rho \lesssim 0.5$, the accelerated algorithms CGIHT and FIHT have comparable average convergence factors and average recovery times. For $0.5 \lesssim \rho \lesssim 0.85$, FIHT appears to offer an advantage in terms of both the convergence factor and recovery time. However, theoretical results indicate that FIHT will lose this advantage in the presence of noise [39]. As the rank increases and forces ρ toward the recovery phase transition, CGIHT regains the computational advantage. The inferior rate of recovery for NIHT is expected since the other two algorithms are designed to accelerate the convergence rate of NIHT. It should be noted that these hard thresholding algorithms require a computationally expensive partial singular value decomposition in each iteration and algorithms which avoid this burdensome task are likely to improve average recovery times, see Chapter 5.

4.4 Proof of Theorem 4.1

By establishing the standard orthogonality relationships for the conjugate gradient method when restricted to a fixed column space, the matrix completion analogue of the uniform bounds on the stepsizes and orthogonalization weights in Alg. 17 follows the same general proof as that of Lem. 3.7.

Lemma 4.3. By the definition of RICs of the measurement operator $\mathcal{A}(\cdot)$, the stepsizes in Thm. 4.1 are uniformly bounded by

$$\frac{1}{1+\delta_r} \leq \alpha_l \leq \frac{1}{1-\delta_r} \tag{4.3}$$

and the orthogonalization weights are uniformly bounded by

$$|\beta_l| \leq \frac{2\delta_r(1+\delta_r)}{(1-\delta_r)^2}. \tag{4.4}$$

Furthermore, if Q, S define column spaces with $cr = \text{rank}(\texttt{Proj}_{Q\cup S})$, then for any $Z \in \mathbb{R}^{m\times n}$,

$$\left\|\texttt{Proj}_Q\left((I-\alpha_l\mathcal{A}^*\mathcal{A})\texttt{Proj}_S(Z)\right)\right\|_F \leq \frac{2\delta_{cr}}{1-\delta_r}\left\|\texttt{Proj}_S(Z)\right\|_F. \tag{4.5}$$

With Lem. 4.3, we prove the convergence guarantee for CGIHT projected for matrix completion, Alg. 17. Unlike in the discrete compressed sensing problem, CGIHT projected for matrix completion suffers from additional computational burdens and reduced empirical performance. Although we require the projected version in order to ensure orthogonality and establish a convergence result, the non projected version, CGIHT for matrix completion (Alg. 16), is recommended for implementations.

Proof of Theorem 4.1. The early steps of the proof of Thm. 4.1 follows the proof of Thm. 2.1 through to (2.15), leading to

$$\begin{aligned} \|X_l - X^o\|_F^2 &\leq 2\langle W_{l-1} - X^o, X_l - X^o\rangle \\ &= 2\left\langle \texttt{Proj}_{U_l \cup U}\left(W_{l-1} - X^o\right), X_l - X^o\right\rangle \\ &\leq \left\|\texttt{Proj}_{U_l \cup U}\left(W_{l-1} - X^o\right)\right\|_F \cdot \|X_l - X^o\|_F \end{aligned}$$

Cancelling one power of $\|X_l - X^o\|_F$ from both sides gives

$$\|X_l - X^o\|_F \leq 2\left\|\texttt{Proj}_{U_l \cup U}\left(W_{l-1} - X^o\right)\right\|_F. \tag{4.6}$$

We seek a bound on $\left\|\texttt{Proj}_{U_l \cup U}\left(W_{l-1} - X\right)\right\|_F$ which is split into two cases based on the value of RestartFlag. When RestartFlag $= 1$,

$$W_{l-1} - X^o = X_{l-1} - X^o + \alpha_{l-1}R_{l-1} = (I - \alpha_{l-1}\mathcal{A}^*\mathcal{A})(X_{l-1} - X^o) + \alpha_{l-1}\mathcal{A}^*(e).$$

Applying the triangle inequality, Lem. 4.3, and Def. 1.9,

$$\begin{aligned} \left\|\texttt{Proj}_{U_l \cup U}\left(X_{l-1} - X^o + \alpha_{l-1}R_{l-1}\right)\right\|_F &\leq \left\|\texttt{Proj}_{U_l \cup U}\left((I - \alpha_{l-1}\mathcal{A}^*\mathcal{A})\,\texttt{Proj}_{U_{l-1} \cup U}\left(X_{l-1} - X^o\right)\right)\right\|_F \\ &\quad + |\alpha_{l-1}|\left\|\texttt{Proj}_{U_l \cup U}\left(\mathcal{A}^*(e)\right)\right\|_F \\ &\leq \frac{2\delta_{3r}}{1-\delta_r}\|X_{l-1} - X^o\|_F + \frac{(1+\delta_{2r})^{1/2}}{1-\delta_r}\|e\|_2. \end{aligned} \tag{4.7}$$

Therefore, if RestartFlag $= 1$,

$$\left\|\texttt{Proj}_{U_l \cup U}\left(W_{l-1} - X^o\right)\right\|_F \leq \frac{2\delta_{3r}}{1-\delta_r}\|X_{l-1} - X^o\|_F + \frac{(1+\delta_{2r})^{1/2}}{1-\delta_r}\|e\|_2. \tag{4.8}$$

On the other hand, when RestartFlag $= 0$ (continuing with the same column space)

$$W_{l-1} - X^o = X_{l-1} - X^o + \alpha_{l-1}\texttt{Proj}_{U_{l-1}}P_{l-1} = X_{l-1} - X^o + \alpha_{l-1}R_{l-1} - \alpha_{l-1}\left(R_{l-1} - \texttt{Proj}_{U_{l-1}}(P_{l-1})\right). \tag{4.9}$$

When RestartFlag $= 0$, the restarting parameter gives the bound,

$$\frac{\left\|R_{l-1} - \texttt{Proj}_{U_{l-1}}(P_{l-1})\right\|_F}{\|\texttt{Proj}_{U_{l-1}}(R_{l-1})\|_F} \leq \theta < \theta_0 = \frac{2c\delta_{3r}}{1+\delta_{2r}}. \tag{4.10}$$

Writing $R_{l-1} = \mathcal{A}^*(\mathcal{A}(X - X_{l-1})) + \mathcal{A}^*(e)$, (4.10) leads to

$$\begin{aligned}
\left\|R_{l-1} - \texttt{Proj}_{U_{l-1}}(P_{l-1})\right\|_F &\leq \theta\|\texttt{Proj}_{U_{l-1}}(R_{l-1})\|_F \\
&\leq \theta\|\texttt{Proj}_{U_{l-1}}(\mathcal{A}^*(\mathcal{A}(X^o - X_{l-1})) + \mathcal{A}^*(e))\|_F \\
&\leq \theta\left(\|\texttt{Proj}_{U_{l-1}\cup U}(\mathcal{A}^*(\mathcal{A}(X^o - X_{l-1})))\|_F + \|\texttt{Proj}_{U_{l-1}}(\mathcal{A}^*(e))\|_F\right) \\
&\leq \theta\left((1+\delta_{2r})\|X_{l-1} - X^o\|_F + (1+\delta_r)^{1/2}\|e\|_2\right) \\
&\leq 2c\delta_{3r}\left(\|X_{l-1} - X^o\|_F + \frac{(1+\delta_r)^{1/2}}{1+\delta_{2r}}\|e\|_2\right).
\end{aligned} \tag{4.11}$$

So, in this case with RestartFlag $= 0$,

$$\begin{aligned}
\left\|\texttt{Proj}_{U_l\cup U}\left(W_{l-1} - X^o\right)\right\|_F &\leq \left\|\texttt{Proj}_{U_l\cup U}\left(X_{l-1} - X^o + \alpha_{l-1}R_{l-1}\right)\right\|_F \\
&\quad + |\alpha_{l-1}|\left\|R_{l-1} - \texttt{Proj}_{U_{l-1}}\left(P_{l-1}\right)\right\|_F \\
&\leq \left(\frac{2\delta_{3r}}{1-\delta_r}\|X_{l-1} - X^o\|_F + \frac{(1+\delta_{2r})^{1/2}}{1-\delta_r}\|e\|_2\right) \\
&\quad + \frac{2c\delta_{3r}}{1-\delta_r}\left(\|X_{l-1} - X^o\|_F + \frac{(1+\delta_r)^{1/2}}{1+\delta_{2r}}\|e\|_2\right) \\
&\leq \frac{2(1+c)\delta_{3r}}{1-\delta_r}\|X_{l-1} - X^o\|_F \\
&\quad + \frac{(1+\delta_{2r})^{1/2}}{1-\delta_r}\left(1 + \frac{2c\delta_{3r}}{1+\delta_{2r}}\right)\|e\|_2.
\end{aligned} \tag{4.12}$$

Clearly, the bound on $\|\texttt{Proj}_{U_l\cup U}(W_{l-1} - X^o)\|_F$ from (4.8) is smaller than the bound in (4.12); thus, (4.12) applies to both cases. Therefore, with

$$\mu = \frac{4(1+c)\delta_{3r}}{1-\delta_r} \quad \text{and} \quad \xi = 2\left(1+\theta_0\right)\frac{(1+\delta_{2r})^{1/2}}{1-\delta_r},$$

equations (4.6), (4.8), and (4.12) combine to show that

$$\|X_l - X^o\|_F \leq \mu\|X_{l-1} - X^o\|_F + \xi\|e\|_2. \tag{4.13}$$

If $\mu < 1$, a straightforward induction argument with (4.13) provides the desired final bound,

$$\|X_l - X^o\|_F \leq \mu^l\|X_0 - X^o\|_F + \frac{\xi}{1-\mu}\|e\|_2. \tag{4.14}$$

□

Chapter 5

Alternating Steepest Descent Methods for Matrix Completion

Many algorithms have been designed for the matrix completion problem[1]

$$\min_{X\in\mathbb{R}^{m\times n}} \operatorname{rank}(X) \text{ subject to } P_\Omega(X) = P_\Omega(X^o) \tag{5.1}$$

such as the nuclear norm minimization which reformulates (5.1) as a convex optimization and the greedy algorihtms which are targeting (5.1) directly [15, 21, 25, 32, 35, 52, 56, 57, 60, 63, 64, 74, 75, 79, 84, 91, 94, 99, 100]. In Chapters 2 and 4, we have presented three variants of hard thresholding algorithms which are able to recover an $m \times n$ rank r matrix from about $C \cdot (m+n-r)r$ measured entries with C being close to one. However, the iterative thresholding algorithms for matrix completion suffer from high per iteration computational cost of the SVD. In this chapter, we introduce an alternating steepest descent algorithm and a scaled variant for the fixed-rank matrix completion problem which can avoid computing the SVD by solving a different non-convex model

$$\min_{\substack{Y\in\mathbb{R}^{m\times r}\\ Z\in\mathbb{R}^{r\times n}}} \frac{1}{2}\left\|P_\Omega(X^o) - P_\Omega(YZ)\right\|_F^2. \tag{5.2}$$

Chapter 5 is outlined as follows[2]. In Sec. 5.1, we briefly review the alternating minimization algorithms PF and LMaFit for (5.2). The alternating steepest descent method (ASD) for (5.2) is presented in Sec. 5.2. In Sec. 5.3, we present ScaledASD, a version of ASD which is accelerated by scaling the search directions adaptively to improve the asymptotic convergence rate. Empirical results presented in Sec. 5.4 show

[1] In this chapter, we focus on entry sensing.

[2] Material in this chapter has been prepared for publication and in preprint [92], which is a joint authorship with J. Tanner whose permission has been obtained for the inclusion of the material.

that ASD and ScaledASD are highly efficient, particularly for large problems when solved to moderate accuracy. Preliminary convergence analysis for ASD and PF is presented in Sec. 5.5.

5.1 Alternating minimization

Alternating minimization is widely used for optimization problems due to its simplicity, low memory footprint and flexibility [47]. Without loss of generality, let us consider an objective function $f(Y, Z)$ with two variable components $Y \in \mathcal{Y}, Z \in \mathcal{Z}$, where $\mathcal{Y}$ and $\mathcal{Z}$ are the admissible sets. The alternating minimization method, also known as Gauss-Seidel or 2-block coordinate descent method, minimizes $f(Y, Z)$ by successive searches over $\mathcal{Y}$ and $\mathcal{Z}$. For problem (5.2), the objective function is

$$f(Y, Z) = \frac{1}{2} \|P_\Omega(X^o) - P_\Omega(YZ)\|_F^2, \tag{5.3}$$

with $\mathcal{Y} := \mathbb{R}^{m\times r}, \mathcal{Z} := \mathbb{R}^{r\times n}$. PF is a matrix completion algorithm which applies the alternating minimization method to (5.2), see Alg. 4 in Chapter 1. As a typical alternating minimization method, limit points of PF are necessarily stationary points.

Theorem 5.1. Any limit point of the sequence (Y_l, Z_l) generated by Alg. 4 is a stationary point.

Proof. See Sec. 5.5.2 for a proof. □

The per iteration computational cost of PF is dominated by the solution of the least squares subproblems in Alg. 4. To decrease this per iteration cost, [100] proposes the model

$$\min_{Y,Z,X} \frac{1}{2} \|X - ZY\|_F^2 \quad \text{subject to} \quad P_\Omega(X) = P_\Omega(X^o), \tag{5.4}$$

where $Y \in \mathbb{R}^{m\times r}$, $Z \in \mathbb{R}^{r\times n}$, $X \in \mathbb{R}^{m\times n}$, and the projection onto Ω is moved from the objective in (5.2) to a linear constraint. The alternating minimization approach applied to (5.4) gives the low-rank matrix fitting algorithm (LMaFit), see Alg. 19

LMaFit obtains new approximate solutions of Y and Z by solving least squares problems while X is held fixed, as done in PF. However, the subproblems in LMaFit have explicit solutions, which reduces the high per iteration computational cost of PF. Moreover, [100] proposes using a rank adaptive strategy and an over-relaxation scheme to further improve the performance of Alg. 19. The convergence of limit points to stationary points is similarly established for LMaFit, see Thm. 3.5 in [100].

Algorithm 19 Low-rank Matrix Fitting (LMaFit, [100])

Initilization: rank r, $Y_0 \in \mathbb{R}^{m\times r}$, $Z_0 \in \mathbb{R}^{r\times n}$, $l = 1$

Iteration: During iteration l, **do**

1: $Y_l = X_{l-1}Z_{l-1}^{\dagger} = \arg\min_{Y} \|YZ_{l-1} - X_{l-1}\|_F^2$ update Y

2: $Z_l = Y_l^{\dagger}X_{l-1} = \arg\min_{Z} \|Y_lZ - X_{l-1}\|_F^2$ update Z

3: $X_l = Y_lZ_l + P_\Omega(X^o - Y_lZ_l)$ update X

Output: $X_l = Y_lZ_l$

5.2 Alternating steepest descent

Solving the least squares subproblems in PF to high accuracy is both computationally expensive and potentially of limited value when the factor that is held fixed is inconsistent with the known entries $P_\Omega(X^o)$. To improve the computational efficiency we replace solving the least squares subproblems in PF with a single step of simple line-search along the gradient descent directions. ASD applies steepest gradient descent to $f(Y, Z)$ in (5.3) alternatively with respect to Y and Z. If $f(Y, Z)$ is written as $f_Z(Y)$ when Z is held constant and $f_Y(Z)$ when Y is held constant, the directions of gradient ascent are

$$\nabla f_Z(Y) = -\left(P_\Omega(X^o) - P_\Omega(YZ)\right)Z^T, \tag{5.5}$$

$$\nabla f_Y(Z) = -Y^T\left(\left(P_\Omega(X^o) - P_\Omega(YZ)\right)\right). \tag{5.6}$$

Let t_y, t_z be the steepest descent stepsizes for descent directions $-\nabla f_Y(X)$ and $-\nabla f_X(Y)$. Then

$$t_x = \frac{\|\nabla f_Z(Y)\|_F^2}{\|P_\Omega(\nabla f_Z(Y)Z)\|_F^2}, \tag{5.7}$$

$$t_y = \frac{\|\nabla f_Y(Z)\|_F^2}{\|P_\Omega(Y\nabla f_Y(Z))\|_F^2}, \tag{5.8}$$

which forms the alternating steepest descent algorithm together with the expressions for the gradients, see Alg. 20.

Remark 5.2. The denominators in $t_{y_{l-1}}$ and $t_{z_{l-1}}$ are zeroes if and only if the gradients are zeroes. This follows from the contradiction that otherwise $f(Y, Z)$ could be decreased without a lower bound which violates the nonnegativity of the objective. If the denominator of $t_{y_{l-1}}$ or $t_{z_{l-1}}$ is zero in an iteration, it indicates Y_{l-1} or Z_{l-1}

Algorithm 20 Alternating Steepest Descent (ASD)

Initilization: rank r, $Y_0 \in \mathbb{R}^{m\times r}$, $Z \in \mathbb{R}^{r\times n}$, $l = 1$

Iteration: During iteration l, **do**

1: $\nabla f_{Z_{l-1}}(Y_{l-1}) = -\left(P_\Omega(X^o) - P_\Omega(Y_{l-1}Z_{l-1})\right) Z_{l-1}^T$	gradient direction for Y
2: $t_{y_{l-1}} = \left\|\nabla f_{Z_{l-1}}(Y_{l-1})\right\|_F^2 / \left\|P_\Omega(\nabla f_{Z_{l-1}}(Y_{l-1})Z_{l-1})\right\|_F^2$	stepsize for Y
3: $Y_l = Y_{l-1} - t_{y_{l-1}}\nabla f_{Z_{l-1}}(Y_{l-1})$	update Y
4: $\nabla f_{Y_l}(Z_{l-1}) = -Y_l^T\left(P_\Omega(X^o) - P_\Omega(Y_l Z_{l-1})\right)$	gradient direction for Z
5: $t_{z_{l-1}} = \left\|\nabla f_{Y_l}(Z_{l-1})\right\|_F^2 / \left\|P_\Omega(Y_l\nabla f_{Y_l}(Z_{l-1}))\right\|_F^2$	stepsize for Z
6: $Z_l = Z_{l-1} - t_{z_{l-1}}\nabla f_{Y_l}(Z_{l-1})$	update Z

Output: $X_l = Y_l Z_l$

respectively are at a local minima and should be held fixed for that iteration by setting the offending stepsize to be zero while minimizing the other objective. If both denominators of $t_{y_{l-1}}$ or $t_{z_{l-1}}$ are zeroes then a stationary point has been reached.

Algorithm 20 consists of two main parts: the computations of the gradient and the steepest descent stepsize. Forming the gradient involves a matrix product between the residual and an $m \times r$ or $r \times n$ matrix. The computations of both the residual and the matrix product require to leading order $2|\Omega|r$ floating point operations (flops), where $|\Omega|$ denotes the number of sampled entries. Computing the stepsize also requires $2|\Omega|r$ flops for the denominator. Naively implemented this would require a per iteration complexity for Alg. 20 of $12|\Omega|r$ flops. However, the residual only needs to be computed once at the beginning of an iteration and then can be updated efficiently. Taking the first two steps as an example, the residual after Y_{l-1} is updated to Y_l is

$$\begin{aligned} P_\Omega(X^o) - P_\Omega(Y_l Z_{l-1}) &= P_\Omega(X^o) - P_\Omega\left(\left(Y_{l-1} - t_{y_{l-1}}\nabla f_{Z_{l-1}}(Y_{l-1})\right) Z_{l-1}\right) \\ &= P_\Omega(X^o) - P_\Omega\left(Y_{l-1}Z_{l-1}\right) + t_{y_{l-1}} P_\Omega\left(\nabla f_{Z_{l-1}}(Y_{l-1})\, Y_{l-1}\right). \end{aligned}$$

Fortunately $P_\Omega\left(\nabla f_{Z_{l-1}}(Y_{l-1})\, Y_{l-1}\right)$ is formed when computing $t_{y_{l-1}}$, so the new residual can be updated from the old one $P_\Omega(X^o) - P_\Omega\left(Y_{l-1}Z_{l-1}\right)$ without computing a matrix-matrix product. Therefore the leading order per iteration cost of Alg. 20 is $8|\Omega|r$ flops.

Similar to PF, limit points of ASD are also stationary points; that is limit points (Y, Z) of the iterates from ASD satisfy

$$\nabla f_Y(Z) = 0, \quad \nabla f_Z(Y) = 0. \tag{5.9}$$

Theorem 5.3. Any limit point of the sequence (Y_l, Z_l) generated by Alg. 20 is a stationary point.

Proof. See Sec. 5.5.1 for a proof. □

Note that Thms. 5.1 and 5.3 do not imply there always exists a convergent subsequence for any iterates generated by PF or ASD. In the implementations, if the iterates are observed to be divergent, the algorithms will be terminated immediately.

5.3 Scaled alternating steepest descent

In this section, we present an accelerated version of Alg. 20. To motivate its construction, let us consider the case when all the entries of X^o are observed. Under this assumption, problem (5.2) can be simplified to

$$\min_{\substack{Y\in\mathbb{R}^{m\times r}\\ Z\in\mathbb{R}^{r\times n}}} \frac{1}{2}\left\|X^o - YZ\right\|_F^2. \tag{5.10}$$

The Newton directions for problem (5.10) with respect to Y and Z are

$$(X^o - YZ)Z^T(ZZ^T)^{-1}, \tag{5.11}$$

$$(Y^TY)^{-1}Y^T(X^o - YZ), \tag{5.12}$$

which are the gradient descent directions scaled by $(ZZ^T)^{-1}$ and $(Y^TY)^{-1}$ respectively. However, when only partial entries of X^o are known, the Newton directions do not possess explicit formulas similar to (5.11) and (5.12); and solving each subproblem by Newton method is the same as solving a least squares problem as $f(Y,Z)$ is an exact quadratic function of Y or Z. Nevertheless, it is still possible to scale the gradient descent directions by $(ZZ^T)^{-1}$ and $(Y^TY)^{-1}$. ScaledASD uses the scaled gradient descent directions with exact line-search, see Alg. 21.

Remark 5.4.

1. The leading order per iteration computational cost of Alg. 21 remains $8|\Omega|r$ if $r \ll \min(m,n)$, and the residual can be updated as in ASD, see Sec. 5.2.

2. Although motivated from a different perspective, ScaledASD can also be viewed as a sequential version of the Riemannian gradient descent method suggested in [74], though with a new metric.

Algorithm 21 Scaled Alternating Steepest Descent (ScaledASD)

Initilization: rank r, $Y_0 \in \mathbb{R}^{m\times r}$, $Z \in \mathbb{R}^{r\times n}$, $l = 1$

Iteration: During iteration l, **do**

1: $\nabla f_{Z_{l-1}}(Y_{l-1}) = -\left(P_\Omega(X^o) - P_\Omega(Y_i Y_{l-1})\right) Z_{l-1}^T$ — gradient for Y

2: $d_{y_{l-1}} = -\nabla f_{Z_{l-1}}(Y_{l-1})(Z_{l-1}Z_{l-1}^T)^{-1}$ — search direction for Y

3: $t_{y_{l-1}} = -\langle \nabla f_{Z_{l-1}}(Y_{l-1}), d_{y_{l-1}}\rangle / \left\|P_\Omega(d_{y_{l-1}} Z_{l-1})\right\|_F^2$ — stepsize for Y

4: $Y_l = Y_{l-1} + t_{y_{l-1}} d_{y_{l-1}}$ — update Y

5: $\nabla f_{Y_l}(Z_{l-1}) = -Y_l^T\left(P_\Omega(X^o) - P_\Omega(Y_l Z_{l-1})\right)$ — gradient for Z

6: $d_{z_{l-1}} = (Y_l^T Y_l)^{-1}\nabla f_{Y_l}(Z_{l-1})$ — search direction for Z

7: $t_{z_{l-1}} = -\langle \nabla f_{Y_l}(Z_{l-1}), d_{z_{l-1}}\rangle / \left\|P_\Omega(Y_l d_{z_{l-1}})\right\|_F^2$ — stepsize for Z

8: $Z_l = Z_{l-1} + t_{z_{l-1}} d_{z_{l-1}}$ — update Z

Output: $X_l = Y_l Z_l$

5.4 Empirical performance comparisons

In this section, we present the empirical results on our gradient based algorithms and other algorithms. ASD and ScaledASD are implemented in Matlab with two subroutines written in C to take advantage of sparse structures. Other tested algorithms include PF [57], LMaFit [100], LRGeomCG [99] and ScGrassMC [79], which are all downloaded from the authors' websites and where applicable use the aforementioned C subroutines. For entry sensing, compared with the iterative hard thresholding algorithms (NIHT, CGIHT, FIHT) which have $O(n^3)$ per iteration complexity due to the singular value decompositions, each aforementioned algorithm tested in this section has $O(|\Omega|r)$ complexity per iteration if $m \sim n$ and $r \ll \min(m, n)$. LRGeomCG and ScGrassMC are both Riemannian optimization algorithms for matrix completion. In each iteration, ScGrassMC updates the left and right singular vector spaces of the low rank matrix by line-search methods, while LRGeomCG acts on the matrix directly. Because LRGeomCG can be viewed as a special iterative hard thresholding algorithm, as a representative of Riemannian methods for matrix completion, we briefly review LRGeomCG in Sec. 5.4.1. We compare these algorithms on both standard test images in Sec. 5.4.2 and random low rank matrices in Sec. 5.4.3.

5.4.1 LRGeomCG as a hard thresholding algorithm

LRGeomCG [99] is a non-linear conjugate gradient algorithm on a Riemannian manifold which targets the non-convex problem

$$\begin{aligned} \min_{X\in\mathbb{R}^{m\times n}} \quad & \frac{1}{2}\|P_\Omega(X^o) - P_\Omega(X)\|_F^2 \\ \text{s.t.} \quad & X \in \mathcal{M}_r := \{X \in \mathbb{R}^{m\times n} : \text{rank}(X) = r\}. \end{aligned} \tag{5.13}$$

It is formulated using several concepts from differential geometry, including tangent space, Riemannian gradient, vector transport, retraction. To avoid this conceptually complexity, we will decribe it directly as a variant of the iterative hard thresholding method. For the Riemainnian optimization ideas behind it, see [1, 99].

As stated previously, the computational bottleneck of the common iterative hard thresholding algorithms exists in the computation of the singular value decompositions in each iteration, which is of the order $O(n^3)$ for an $m \times n$ matrix with $m \sim n$. The SVD on a full $m \times n$ matrix is typically needed when projecting the intermediate matrix onto the rank r manifold, as the estimate is updated along a gradient descent direction which is not in any particular low dimensional subspace. However, if the line-search direction is restricted into a low dimensional subspace associated with the current estimate, it is likely to get an intermediate matrix which is low rank itself. Then we might be able to operate on a smaller size matrix when truncating the estimate to its nearest rank r approximation, provided the subspace is explicitly known. By exploring this subspace idea, the general gradient and conjugate gradient iterative hard thresholding algorithms for matrix completion are presented in Algs. 22 and 23 respectively. LRGeomCG can be recovered from Alg. 23 by selecting the subspace specified in (5.16) in each iteration.

Compared with NIHT (Alg. 5) in Chapter 2, the major difference in Alg. 22 is at Step 4, where the current estimate is updated along a projected gradient descent direction rather than the gradient descent direction. Furthermore, in Alg. 23, the search direction is selected to be an appropriate linear combination of the projected gradient descent direction and the prior search direction projected onto the current subspace (see Step 3 in Alg. 23). Motivated by the non-linear conjugate gradient method in convex optimization [80], there are several choices for the parameter β_{l-1}

Algorithm 22 General Gradient Descent for matrix completion

Initilization: rank r, $U_0 = \texttt{PrincipalLeftSingularVectors}_r(P_\Omega(X^o))$, $X_0 = \texttt{Proj}_{U_0}(P_\Omega(X^o))$, $l = 1$

Iteration: During iteration l, **do**

1: $R_{l-1} = P_\Omega(X^o) - P_\Omega(X_{l-1})$ (gradient descent direction)
2: $R^{\mathcal{S}}_{l-1} = \texttt{Proj}_{\mathcal{S}_{l-1}}(R_{l-1})$ (projected gradient descent direction)
3: $\alpha_{l-1} = \langle R_{l-1}, R^{\mathcal{S}}_{l-1}\rangle / \left\|P_\Omega\left(R^{\mathcal{S}}_{l-1}\right)\right\|_F^2$ (steepest descent stepsize along $R^{\mathcal{S}}_{l-1}$)
4: $W_{l-1} = X_{l-1} + \alpha_{l-1} R^{\mathcal{S}}_{l-1}$ (gradient descent update)
5: $U_l = \texttt{PrincipalLeftSingularVectors}_r(W_{l-1})$ (principal component subspace)
6: $X_l = \texttt{Proj}_{U_l}(W_{l-1})$ (estimate update)

Output: X_l

[1]. Explicitly imposing $P^{\mathcal{S}}_{l-1}$ to be conjugate orthogonal to $\texttt{Proj}_{\mathcal{S}_{l-1}}\left(P^{\mathcal{S}}_{l-2}\right)$ yields

$$\beta_{l-1} = \frac{\left\langle P_\Omega\left(R^{\mathcal{S}}_{l-1}\right), P_\Omega\left(\texttt{Proj}_{\mathcal{S}_{l-1}}\left(P^{\mathcal{S}}_{l-2}\right)\right)\right\rangle}{\left\|\texttt{Proj}_{\mathcal{S}_{l-1}}\left(P^{\mathcal{S}}_{l-2}\right)\right\|_F^2},$$

while other formulas for β_{l-1} include

$$\text{Fletcher-Reeves} \quad \beta^{FR}_{l-1} = \frac{\left\|R^{\mathcal{S}}_{l-1}\right\|_F^2}{\left\|R^{\mathcal{S}}_{l-2}\right\|_F^2}$$

$$\text{Polak-Ribière} \quad \beta^{PR}_{l-1} = \frac{\left\langle R^{\mathcal{S}}_{l-1},\ R^{\mathcal{S}}_{l-1} - \texttt{Proj}_{\mathcal{S}_{l-1}}\left(R^{\mathcal{S}}_{l-2}\right)\right\rangle}{\left\|R^{\mathcal{S}}_{l-2}\right\|_F^2}$$

$$\text{Polak-Ribière+} \quad \beta^{PR+}_{l-1} = \max\left\{\beta^{PR}_{l-1}, 0\right\}.$$

Algorithm 23 General Conjugate Gradient Descent for matrix completion

Initilization: rank r, $U_0 = \texttt{PrincipalLeftSingularVectors}_r(P_\Omega(X^o))$, $X_0 = \texttt{Proj}_{U_0}(P_\Omega(X^o))$, $l = 1$

Iteration: During iteration l, **do**

1: $R_{l-1} = P_\Omega(X^o) - P_\Omega(X_{l-1})$ (gradient descent direction)
2: $R^{\mathcal{S}}_{l-1} = \texttt{Proj}_{\mathcal{S}_{l-1}}(R_{l-1})$ (projected gradient descent direction)
3: $P^{S}_{l-1} = R^{\mathcal{S}}_{l-1} + \beta_{l-1}\texttt{Proj}_{\mathcal{S}_{l-1}}\left(P^{\mathcal{S}}_{l-2}\right)$ (conjugate gradient direction on $\mathcal{S}_{l-1}$)
4: $\alpha_{l-1} = \langle R_{l-1}, P^{\mathcal{S}}_{l-1}\rangle / \left\|P_\Omega(P^{\mathcal{S}}_{l-1})\right\|_F^2$ (steepest descent stepsize along $P^{\mathcal{S}}_{l-1}$)
5: $W_{l-1} = X_{l-1} + \alpha_{l-1} P^{\mathcal{S}}_{l-1}$ (conjugate gradient descent update)
6: $U_l = \texttt{PrincipalLeftSingularVectors}_r(W_{l-1})$ (principal component subspace)
7: $X_l = \texttt{Proj}_{U_l}(W_{l-1})$ (estimation update)

Output: X_l

Let X_l be the current estimate in Algs. 22 or 23, and $X_l = U_l \Sigma_l V_l^*$ be the singular

value decomposition. If $\mathcal{S}_l$ is selected to be the column space of X_l spanned by U_l:

$$\mathcal{S}_l = \{X \in \mathbb{R}^{m\times n} : X = U_l R^* \text{ with } R \in \mathbb{R}^{n\times r}\}, \tag{5.14}$$

the intermediate matrix W_l remains in $\mathcal{S}_l$, which implies W_l is already a rank r matrix, and no hard thresholding is needed. However, if $X^o \notin \mathcal{S}_0$, it can never be recovered by Algs. 22 and 23 since $S_l = S_{l-1} \cdots = S_1 = S_0$ does not include X^o in each iteration. Similar conclusion can be drawn if S_l is the row space of X_l spanned by V_l:

$$\mathcal{S}_l = \{X \in \mathbb{R}^{m\times n} : X = L V_l^* \text{ with } L \in \mathbb{R}^{m\times r}\}. \tag{5.15}$$

Therefore, a larger $\mathcal{S}_l$ is necessary to allow the subspace to update iteratively, so that it can capture more details of the underlying low rank matrix. When $\mathcal{S}_l$ is selected to be a direct sum of (5.14) and (5.15)

$$\mathcal{S}_l = \{X \in \mathbb{R}^{m\times n} : X = U_l R^* + L V_l^* \text{ with } R \in \mathbb{R}^{n\times r} \text{ and } L \in \mathbb{R}^{m\times r}\}, \tag{5.16}$$

Algs. 22 and 23 turn out to be the Riemmannian gradient and conjugate gradient methods for (5.13). In the matrix completion literature, the subspace of the form (5.16) first appears in [29], where it plays the role of the "support set" of a low rank matrix for the authors to establish the equivalence between the nuclear norm minimization and matrix completion, see Thm. 1.13. Here are several facts about $\mathcal{S}_l$.

Proposition 5.5 ([99])**.**

1. For any $X \in \mathcal{S}_l$, $\text{rank}(X) \leq 2r$.

2. $\mathcal{S}_l$ is the tangent space of the smooth manifold $\mathcal{M}_r$ at X_l; and the dimension of $\mathcal{S}_l$ is $r(m+n-r)$.

3. For any $X \in \mathbb{R}^{m\times n}$,

$$\begin{aligned} \texttt{Proj}_{\mathcal{S}_l}(X) &= U_l U_l^* X + X V_l V_j^* - U_l U_l^* X V_l V_l^*, & (5.17)\\ \texttt{Proj}_{\mathcal{S}_l^\perp}(X) &= (I_m - U_l U_l^*) X (I_n - V_l V_l^*), & (5.18) \end{aligned}$$

   where I_m and I_n are $m\times m$ and $n\times n$ identity matrices, and $\mathcal{S}_l^\perp$ is the orthogonal complement of $\mathcal{S}_l$.

In each iteration of Algs. 22 and 23, if $\mathcal{S}_l$ is selected to be the subspace in (5.16), an SVD is needed to project W_l onto the rank r matrix manifold. First note that in both algorithms, W_l can be expressed as

$$W_l = X_l + Z_l, \tag{5.19}$$

where $Z_l = \alpha_l R_l^{\mathcal{S}}$ in Alg. 22 and $Z_l = \alpha_l \left(R_l^{\mathcal{S}} + \beta_l \texttt{Proj}\left(P_{l-1}^{\mathcal{S}}\right)\right)$ in Alg. 23 are both in $\mathcal{S}_l$. Concequently,

$$\begin{aligned}
W_l &= U_l\Sigma_l V_l^* + Z_l \\
&= U_l\Sigma_l V_l^* + \texttt{Proj}_{\mathcal{S}_l}(Z_l) \\
&= U_l\Sigma_l V_l^* + U_l U_l^T Z_l + Z_l V_l V_l^* - U_l U_l^T Z_l V_l V_l^* \\
&= U_l\Sigma_l V_l^* + U_l U_l^T Z_l V_l V_l^* \\
&\quad + Z_l V_l V_l^* - U_l U_l^T Z_l V_l V_l^* \\
&\quad + U_l U_l^T Z_l - U_l U_l^T Z_l V_l V_l^* \\
&= U_l(\Sigma_l + U_l^T Z_l V_l)V_l^* + \underbrace{(I_m - U_l U_l^*) Z_l V_l}_{Y_1} V_l^* + U_l \underbrace{U_l^T Z_l (I_n - V_l V_l^*)}_{Y_2^*}.
\end{aligned}$$

If $Y_1 = Q_1 R_1$, $Y_2 = Q_2 R_2$ is the QR factorizations of $Y_1 \in \mathbb{R}^{m\times r}$ and $Y_2 \in \mathbb{R}^{n\times r}$, then $U_l^* Q_1 = 0$ and $V_l^* Q_2 = 0$. Therefore,

$$\begin{aligned}
W_l &= U_l(\Sigma_l + U_l^* Z_l V_l)V_l^* + Q_1 R_1 V_l^* + U_l R_2^* Q_2^* \\
&= \begin{bmatrix} U_l & Q_1 \end{bmatrix} \underbrace{\begin{bmatrix} \Sigma_l + U_l^* Z_l V_l & R_2^* \\ 0 & R_1 \end{bmatrix}}_{M_l} \begin{bmatrix} V_l^* \\ Q_2^* \end{bmatrix},
\end{aligned}$$

where M_l is a $2r \times 2r$ matrix. Since $\begin{bmatrix} U_l & Q_1 \end{bmatrix}$ and $\begin{bmatrix} V_l & Q_2 \end{bmatrix}$ are both unitary matrices, the SVD of W_l can be obtained from the SVD of M_l, which is a matrix of size much smaller than $\min(m, n)$ if $r \ll \min(m, n)$.

5.4.2 Image inpainting

In this section, we demonstrate the performance of the proposed algorithms on image inpainting problems. Image inpainting concerns filling in the unknown pixels of an image from an incomplete set of observed entries. All the aforementioned algorithms are tested on three standard 512×512 grayscale test images (Boat, Barbara and Lena), each projected to the nearest rank 50 image in the Frobenius norm. Two sampling schemes are considered, a) random sampling where 35% pixels of the low rank image

Table 5.1: Relative reconstruction errors for image inpainting

	ASD	ScaledASD	LMaFit	PF	LRGeomCG	ScGrassMC
Random	3.13e-02	7.04e-05	7.39e-05	6.28e-04	5.0e-05	6.81e-5
Cross	9.35e-02	1.11e-04	1.12e-04	9.04e-04	7.28e-05	1.14e-04

were sampled uniformly at random, and b) cross mask where 6.9% pixels of the image were masked in a non-random cross pattern. The relative residual tolerance was set to 10^{-5}. All algorithms were provided with the true rank of the image except LMaFit for random sampling which used warm starting strategies proposed by its authors in [100], where the hand tuned parameters chosen to give the best performance on the images tested were their increasing strategy with initial rank set to 25 and maximum rank set to 60. The tests were performed on the same computer as described in Sec. 5.4.3.2.

The reconstructed images for the Boat test image and each tested algorithm are displayed in Figs. 5.1 and 5.2 for the random sampling and cross mask respectively. The relative errors of the reconstructed images are presented in Tab. 5.1. The relative residual plotted against iteration number and computational time for each test image and algorithm tested are displayed in Figs. 5.3 and 5.4 respectively. Fig. 5.4 (a,c,e) show that ScaledASD and ScGrassMC have rapid initial decrease in the residual, and that LRGeomCG has the fastest asymptotic rate. For moderate accuracy ScaledASD and ScGrassMC require the least time for random sampling of the tested natural images, while for high accuracy LRGeomCG is preferable. It should be noted that the completed images are visually indistinguishable for relative residuals of about 10^{-2}, which is well before the fast asymptotic rate of LRGeomCG begins. Alternatively, for the cross mask, Fig. 5.4 (b,d,f) show that ScaledASD, PF and LMaFit are preferable throughout, though the fast asymptotic rate of LRGeomCG suggests it would be superior for a relative tolerance below 10^{-5}. Only ScaledASD is preferred for both the random sampling and cross mask, suggesting its usage for moderate accuracy, and LRGeomCG for higher accuracy.

5.4.3 Random matrix completion problems

In this section we conduct tests of randomly drawn rank r matrices using the model $X^o = YZ$, where $Y \in \mathbb{R}^{m\times r}$ and $Z \in \mathbb{R}^{r\times n}$ with Y and Z having their entries drawn i.i.d. from the normal distribution $\mathcal{N}(0,1)$. A random subset Ω of p entries from X^o

Figure 5.1: Image reconstruction for "Boat" from random sampling. The size of the image is 512×512. The desired rank is 50, and 35% entries of the low rank image are sampled uniformly at random.

Figure 5.2: Image reconstruction for "Boat" from cross masked. The size of the image is 512×512. The desired rank is 50, and 6.9% entries of the low rank image are masked in a non-random way.

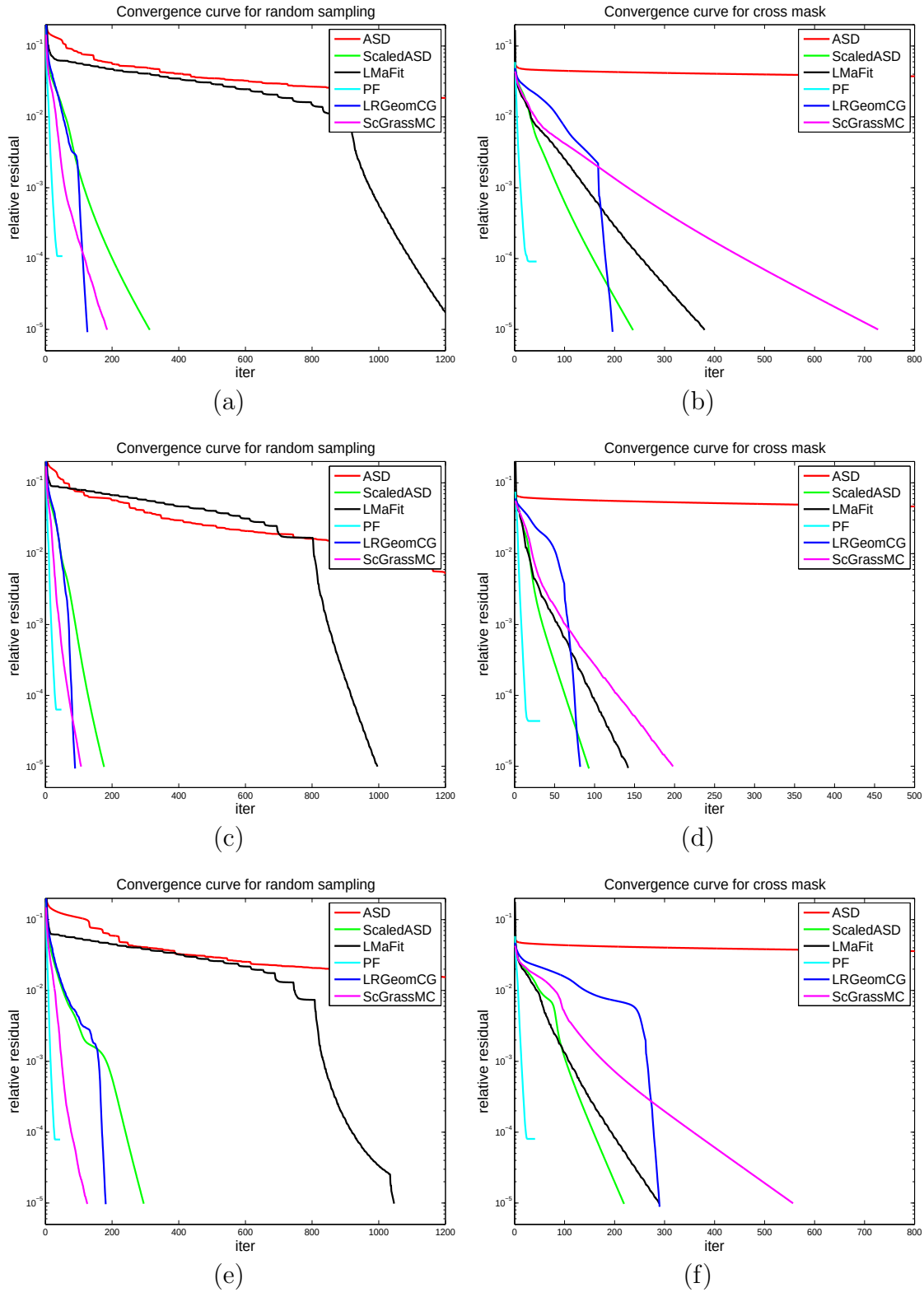


Figure 5.3: Relative residual as function of iterations for image recovery testing on "Boat" (top), "Barbara" (middle) and "Lena" (bottom). Left panel: random sampling; right panel: cross mask.

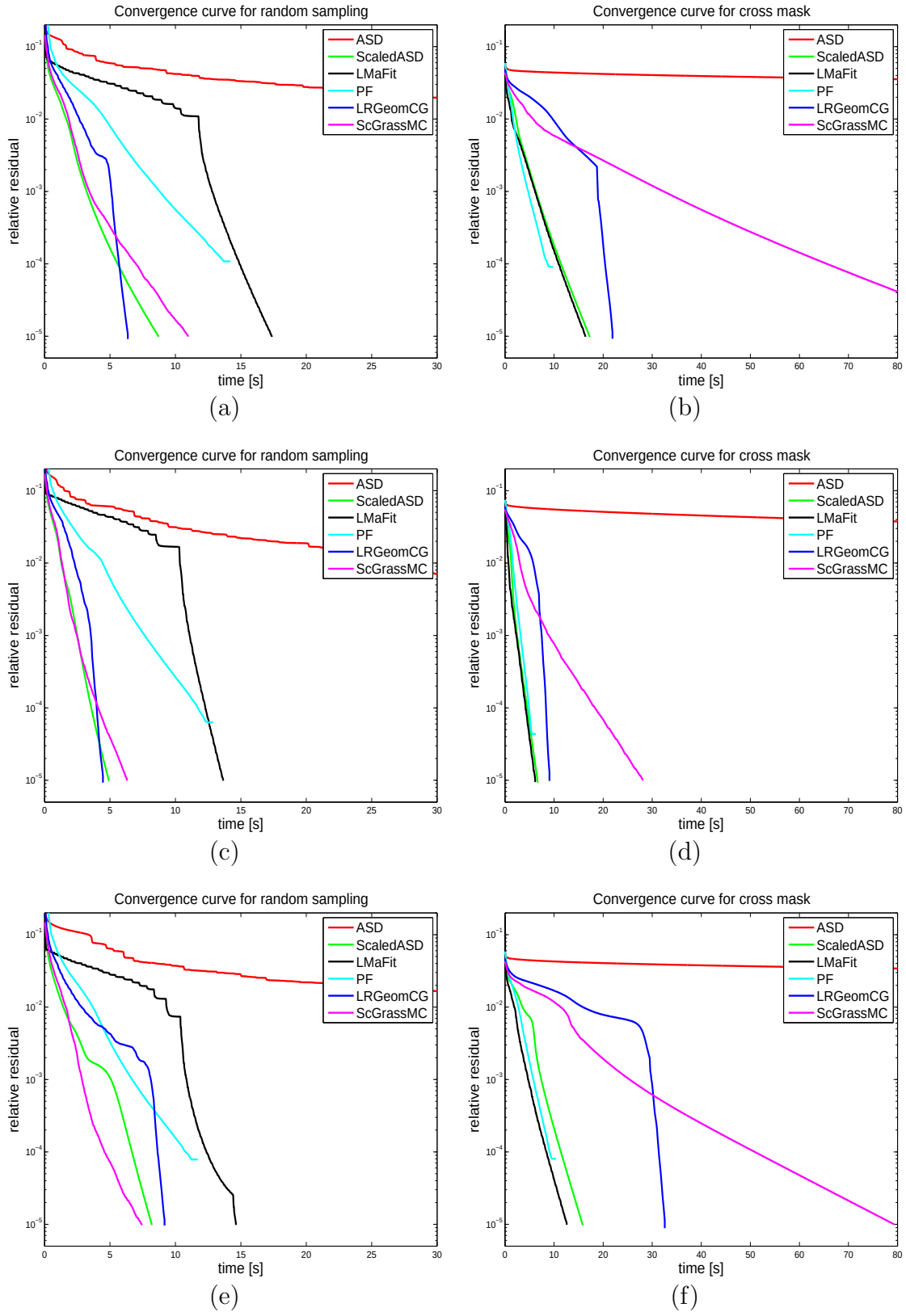


Figure 5.4: Relative residual as function of time for image recovery testing on "Boat" (top), "Barbara"(middle) and "Lena"(bottom). Left panel: random sampling; right panel: cross mask.

are sampled uniformly at random. For conciseness, the tests presented in this section consider square matrices as is typical in the literature.

5.4.3.1 Largest recoverable rank

Of central importance in matrix completion is what is the largest recoverable rank given (m, n, p), or equivalently, given (m, n, r) how many measurements of the entries are needed in order to reliably recover a rank r matrix. To quantify the optimality of such statements we use the same phase transition framework as in Chapters 2 and 4. First recall that the phase transition for matrix completion is defined through the undersampling and oversampling ratios:

$$\delta = \frac{p}{mn} \quad \text{and} \quad \rho = \frac{d_r}{p}, \tag{5.20}$$

where p is the number of sampled entries, and $d_r = r(m + n - r)$ is the degrees of freedom in a $m \times n$ rank r matrix. The region of greatest importance is severe undersampling, for $\delta \ll 1$, and we restrict our tests to two values $\delta \in \{0.05, 0.1\}$, which also allows us to test large matrices[3]. Due to the high computational cost of accurately computing an algorithm's phase transition we limit the testing here to the three of the most efficient algorithms ASD, LMaFit and LRGeomCG. Both ASD and LRGeomCG were provided with the true rank of the matrix, while the decreasing strategy with the initial rank set to $\lfloor 1.25r \rfloor$ was used in LMaFit as suggested in [100].

As in Chapter 2, for each triple (m, n, p), we started from a sufficiently small rank r so that the algorithm could successfully recover each of the sampled matrices in 100 random tests. The rank is then increased until it was large enough that the algorithm failed to recovery any of the 100 testing matrices to within the prescribed relative accuracy. In these tests an algorithm is considered to have successfully recovered the sampled matrix X^o if it returns a matrix $\hat{X}$ that is within 2×10^{-3} of X^o in the relative Frobenius norm, see (2.12) in Chapter 2. We refer to the largest rank for which the algorithm succeeded in each of the 100 tests as r_{min}, and the smallest rank for which the algorithm failed all the 100 tests as r_{max}. The exact values of r_{min}, r_{max} and associated ρ_{min}, ρ_{max} are listed in Tab. 5.2.

[3]The tests were conducted on the IRIDIS HPC facility provided by the e-Infrastructure South Centre for Innovation.The IRIDIS facility runs Linux nodes composed of two 6-core 2.4GHz Intel Westmere processors and approximately 22GB of usable memory per node. The data presented in Sec. 5.4.3 required 4.5 months of CPU time.

From Tab. 5.2, we can see that, similar to NIHT and CGIHT for matrix completion, both ASD and LGeomCG are able to recovering a randomly generated rank r matrix from only $C \cdot r(m+n-r)$ measurements of its entries with C being only slightly larger than 1, even when $\delta = 0.05$. In contrast, LMaFit has a substantially lower phase transition. Table 5.2 shows that these observations are consistent for different, moderately large size matrices.

Table 5.2: Phase transition table for ASD, LMaFit and LRGeomCG. For each (m, n, p) with $p = \delta \cdot mn$, the algorithm was observed to successfully recover each of the 100 randomly drawn $m \times n$ matrices of rank r if $r \leq r_{min}$ and failed to recover each of the randomly drawn matrices if $r \geq r_{max}$. The oversampling ratios ρ_{min} and ρ_{max} are computed using r_{min} and r_{max} by (5.20).

Configuration		ASD				LRGeomCG				LMaFit			
m = n	δ	r_{min}	r_{max}	ρ_{min}	ρ_{max}	r_{min}	r_{max}	ρ_{min}	ρ_{max}	r_{min}	r_{max}	ρ_{min}	ρ_{max}
1000	0.05	18	23	0.71	0.91	18	24	0.71	0.95	12	22	0.48	0.87
1000	0.10	43	48	0.84	0.94	44	48	0.86	0.94	30	44	0.59	0.86
2000	0.05	44	47	0.87	0.93	44	47	0.87	0.93	25	41	0.50	0.81
2000	0.10	92	95	0.90	0.93	92	96	0.90	0.94	54	84	0.53	0.82
4000	0.05	91	94	0.90	0.93	92	95	0.91	0.94	65	84	0.64	0.83
4000	0.10	186	190	0.91	0.93	188	191	0.92	0.93	141	172	0.69	0.84

5.4.3.2 Recovery time

In this section we evaluate the recovery time for the aforementioned algorithms when applied to the previously described randomly generated rank r matrices. The relative error, number of iterations, and computational time reported in this section are average results of ten random tests. All algorithms were supplied with the true rank, except LMaFit which used the decreasing strategy with the initial rank $\lfloor 1.25r \rfloor$ as advocated in [100]. The tolerance for relative residual, $\left\| P_\Omega(\hat{X} - X^o) \right\|_2 / \|P_\Omega(X^o)\|_2$, was set to 10^{-5} for each algorithm and other parameters were set to their default values. The simulation was conducted on a Linux machine with Intel Xeon E5-2643 CPUs @ 3.30 GHz and 64GB memory, and executed from Matlab R2013a.

The performance of ASD, ScaledASD and PF are compared in Tab. 5.3 for medium size matrices of $m = n = 1000$ with undersampling ratio $\delta \in \{0.1, 0.3, 0.5\}$ and the rank of the matrix is taken so that $p/d_r \approx 2.0$ and 1.27. Table 5.3 shows that PF takes many fewer iterations than ASD and ScaledASD; however, both ASD and ScaledASD can require much less computational time than PF due to their lower per iteration computational complexity. ScaledASD is observed to take fewer number of iterations than ASD and is typically faster, with the advantage of ScaledASD over ASD increases as $\delta = p/mn$ increases for p/d_r being approximately fixed. This is because the more entries of a matrix are known, the more likely it is that ScaledASD

will behave like a second order method; in particular, if all the entries have been observed, ScaledASD is actually an alternating Newton method for (5.10) while ASD is an alternating gradient descent method.

Next we compare the algorithms, excluding PF due to its slow convergence, on larger problem sizes and explore how their performance changes with additive noise. Tests with additive noise have the sampled entries $P_\Omega(X^o)$ corrupted by the vector

$$e = \epsilon \cdot \|P_\Omega(X^o)\|_2 \cdot \frac{w}{\|w\|_2}, \tag{5.21}$$

where the entries of w are i.i.d standard Gaussian random variables, and ϵ is referred to as the noise level. The tests conducted here consider ϵ to be either zero or 0.1. Tests are conducted for $m = n \in \{4000, 8000, 16000\}$, $r \in \{40, 80\}$, and $p/d_r \in \{3, 5\}$. The results are presented in Tabs. 5.4 and 5.5 for noise levels $\epsilon = 0$ and 0.1 respectively.

Table 5.4 shows that for $\epsilon = 0$, ASD and ScaledASD consistently require less time than LMaFit, ScaledASD always requires the least time for $p/d_r = 5$ and LRGeomCG requires the least time for $p/d_r = 3$. Table 5.5 shows that for $\epsilon = 0.1$, the performance is predictable, with out of the twelve (m, n, p, r) tuples tested: LMaFit requires the least time in four instances, ASD in one instance, ScaledASD in three instances, and LRGeomCG in four instances. Despite the efficiency of ScGrassMC for random sampling of the natural images in Fig. 5.4 (a,c,e), Tabs. 5.4 and 5.5 show that ScGrassMC consistently requires substantially greater computational time, for random rank r matrices, than the other algorithms tested. Tables 5.4 and 5.5 show the efficiency of ASD and ScaledASD when the random problems are solved to moderate accuracy. However, it is worth noting that these randomly generated matrices are well-conditioned with high probability; as ASD and ScaledASD are both first order methods, it can be expected that they will suffer from slow asymptotic convergence for severely ill-conditioned matrices, in which case a higher order method such as LRGeomCG or ScGrassMC may be preferable, see [21, 79] for a discussion of this phenomenon.

Table 5.3: Average computational time (s) and average number of iterations of ASD, ScaledASD and PF over ten random rank r matrices per (m, n, p, r) tuple for $m = 1000, n = 1000$, and $\delta \in \{0.1, 0.3, 0.5\}$.

p/d_r	2.05			1.28		
	rel.err	iter	time	rel.err	iter	time
p/mn	0.1					
r	25			40		
ASD	3.5e-5	103	1.5	9.0e-5	582	14.1
ScaledASD	3.5e-5	97	**1.5**	9.1e-5	533	**13.3**
PF	2.6e-5	29	11.0	8.1e-5	141	85.4
p/mn	0.3					
r	75			125		
ASD	2.6e-5	63	8.8	7.0e-5	373	88.7
ScaledASD	2.6e-5	48	**6.8**	7.0e-5	252	**61.6**
PF	2.1e-5	20	14.0	6.7e-5	93	110
p/mn	0.5					
r	130			220		
ASD	2.1e-5	64	26.3	5.8e-5	361	255
ScaledASD	2.3e-5	33	**13.7**	6.0e-5	155	**111**
PF	1.6e-5	16	16.5	5.7e-5	65	124

Table 5.4: Average computational time (s) and average number of iterations of ASD, ScaledASD, LMaFit and LRGeomCG over ten random rank r matrices per (m, n, p, r) tuple for $m = n \in \{4000, 8000, 16000\}, r \in \{40, 80\}$, and $p/d_r \in \{3, 5\}$. The noise level ϵ is equal to 0.

r	40						80					
p/d_r	3			5			3			5		
	rel.err	iter	time	rel.err	iter	time	rel.err	iter	time	rel.err	iter	time
$m = n$	4000											
ASD	2.1e-5	42	11.0	1.4e-5	23	9.6	2.0e-5	35	36.1	1.1e-5	20	33.7
ScaledASD	2.0e-5	41	10.6	1.4e-5	22	**9.2**	1.9e-5	33	34.1	1.2e-5	18	**30.4**
LMaFit	2.1e-5	68	12.0	1.3e-5	35	10.4	2.0e-5	55	37.8	1.2e-5	31	35.9
LRGeomCG	1.3e-5	23	**9.4**	1.0e-5	15	9.8	1.0e-5	21	**33.6**	6.9e-6	14	36.0
ScGrassMC	1.7e-5	24	76.1	1.3e-5	16	116	1.3e-5	23	967	1.1e-5	14	1609
$m = n$	8000											
ASD	2.1e-5	43	24.6	1.4e-5	23	21.1	1.9e-5	37	80.8	1.4e-5	20	70.9
ScaledASD	2.1e-5	43	24.4	1.4e-5	23	**20.8**	1.9e-5	36	78.6	1.4e-5	20	**68.6**
LMaFit	1.9e-5	73	29.2	1.5e-5	38	24.6	1.9e-5	62	94.0	1.3e-5	35	87.0
LRGeomCG	1.7e-5	23	**20.4**	8.1e-6	16	22.2	1.3e-5	22	**74.1**	1.2e-5	15	77.4
ScGrassMC	1.4e-5	29	179	8.9e-6	16	265	1.7e-5	23	2330	9.5e-6	15	3445
$m = n$	16000											
ASD	2.1e-5	44	61.4	1.4e-5	23	50.4	1.9e-5	38	201	1.1e-5	21	169
ScaledASD	2.0e-5	43	56.4	1.4e-5	23	**47.8**	2.0e-5	37	186	1.4e-5	20	**159**
LMaFit	2.2e-5	81	77.5	1.3e-5	41	62.6	1.7e-5	69	242	1.4e-5	37	210
LRGeomCG	1.6e-5	24	**47.5**	1.0e-5	16	49.9	1.4e-5	22	**167**	9.7e-6	15	178
ScGrassMC	1.5e-5	25	495	1.0e-5	16	704	1.4e-5	23	6063	1.0e-5	16	8681

Table 5.5: Average computational time (s) and average number of iterations of ASD, ScaledASD, LMaFit and LRGeomCG over ten random rank r matrices per (m, n, p, r) tuple for $m = n \in \{4000, 8000, 16000\}, r \in \{40, 80\}$, and $p/d_r \in \{3, 5\}$. The noise level ϵ is equal to 0.1.

r	40						80					
p/d_r	3			5			3			5		
	rel.err	iter	time	rel.err	iter	time	rel.err	iter	time	rel.err	iter	time
$m = n$	4000											
ASD	7.1e-2	21	5.6	5.0e-2	19	8.2	7.0e-2	21	21.6	4.9e-2	19	32.0
ScaledASD	7.1e-2	21	**5.6**	5.0e-2	19	8.3	7.0e-2	21	21.8	4.9e-2	19	32.2
LMaFit	7.1e-2	30	5.8	5.0e-2	20	**6.3**	7.0e-2	26	**18.8**	4.9e-2	16	**19.9**
LRGeomCG	7.1e-2	14	5.8	5.0e-2	11	7.2	7.0e-2	13	20.9	4.9e-2	11	28.4
ScGrassMC	7.1e-2	15	69.8	5.0e-2	10	113	7.0e-2	14	984	4.9e-2	9	1524
$m = n$	8000											
ASD	7.1e-2	21	**12.3**	5.0e-2	19	17.9	7.0e-2	21	46.2	5.0e-2	19	67.1
ScaledASD	7.1e-2	21	12.3	5.0e-2	19	18.0	7.0e-2	21	46.5	5.0e-2	19	67.5
LMaFit	7.1e-2	34	14.8	5.0e-2	22	15.7	7.0e-2	31	49.7	5.0e-2	20	**52.7**
LRGeomCG	7.1e-2	14	12.7	5.0e-2	11	**15.5**	7.0e-2	13	**44.7**	5.0e-2	11	60.8
ScGrassMC	7.1e-2	16	171	5.0e-2	10	264	7.0e-2	14	2337	5.0e-2	10	3623
$m = n$	16000											
ASD	7.1e-2	21	29.8	5.0e-2	20	41.9	7.1e-2	21	108	5.0e-2	19	157
ScaledASD	7.1e-2	21	**28.2**	5.0e-2	20	42.3	7.1e-2	21	**106**	5.0e-2	19	152
LMaFit	7.3e-2	29	31.1	5.0e-2	26	42.4	7.1e-2	32	119	5.0e-2	22	135
LRGeomCG	7.1e-2	15	29.4	5.0e-2	11	**35.7**	7.1e-2	14	106	5.0e-2	11	**133**
ScGrassMC	7.1e-2	16	651	5.0e-2	11	757	7.1e-2	15	6285	5.0e-2	10	9227

5.5 Proofs of Theorems 5.1 and 5.3

The convergence analysis of block coordinate descent methods for general unconstrained optimization problems has been studied in [55] based on stationary points. However, due to the simple structure of problem (5.2), for completeness we will offer a direct proof. Because the proof of Thm. 5.1 follows that of Thm. 5.3, we first present the proof of Thm. 5.3.

5.5.1 ASD convergence analysis: Proof of Theorem 5.3

Proof of Theorem 5.3. Suppose (Y_{l_k}, Z_{l_k}) is a subsequence of (Y_l, Z_l) such that

$$\lim_{l_k \to \infty} Y_{l_k} = Y^*, \lim_{l_k \to \infty} Z_{l_k} = Z^*,$$

for some (Y^*, Z^*). In order to prove (Y^*, Z^*) is a stationary point, by continuity of the gradients, it is sufficient to prove

$$\lim_{l_k \to \infty} \nabla f_{Y_{l_k}}(Z_{l_k}) = 0, \quad \lim_{l_k \to \infty} \nabla f_{Z_{l_k}}(Y_{l_k}) = 0. \tag{5.22}$$

Since (Y_{l_k}, Z_{l_k}) are bounded for all l_k, (5.22) follows immediately from the subsequent Lems. 5.6 and 5.7. □

Lemma 5.6. If (Y_{l_k}, Z_{l_k}) is a bounded sequence,

$$\lim_{l_k\to\infty} \nabla f_{Y_{l_k}}(Z_{l_k-1}) = 0, \ \lim_{l_k\to\infty} \nabla f_{Z_{l_k}}(Y_{l_k}) = 0 \tag{5.23}$$

Proof. By the quadratic form of $f(Y,Z)$ with respect to Y and Z, the decrease of $f(Y,Z)$ can be computed exactly in each iteration,

$$f(Y_l, Z_{l-1}) = f(Y_{l-1}, Z_{l-1}) - \frac{1}{2}\frac{\left\|\nabla f_{Z_{l-1}}(Y_{l-1})\right\|_F^4}{\left\|P_\Omega\left(\nabla f_{Z_{l-1}}(Y_{l-1})Z_{l-1}\right)\right\|_F^2}, \tag{5.24}$$

$$f(Y_l, Z_l) = f(Y_l, Z_{l-1}) - \frac{1}{2}\frac{\|\nabla f_{Y_l}(Z_{l-1})\|_F^4}{\|P_\Omega\left(Y_l\nabla f_{Y_l}(Z_{l-1})\right)\|_F^2}. \tag{5.25}$$

Summing up (5.24) and (5.25) from $l=0$ to $l=i$ gives

$$f(Y_i, Z_i) = f(Y_0, Z_0) - \frac{1}{2}\sum_{l=0}^{i-1}\left\{\frac{\|\nabla f_{Z_l}(Y_l)\|_F^4}{\|P_\Omega\left(\nabla f_{Z_l}(Y_l)Z_l\right)\|_F^2} + \frac{\left\|\nabla f_{Y_{l+1}}(Z_l)\right\|_F^4}{\left\|P_\Omega\left(Y_{l+1}\nabla f_{Y_{l+1}}(Z_l)\right)\right\|_F^2}\right\}.$$

The nonnegativity of $f(Y_i, Z_i)$ for all i implies

$$\sum_{l=0}^{\infty}\left\{\frac{\|\nabla f_{Z_l}(Y_l)\|_F^4}{\|P_\Omega\left(\nabla f_{Z_l}(Y_l)Z_l\right)\|_F^2} + \frac{\left\|\nabla f_{Y_{l+1}}(Z_l)\right\|_F^4}{\left\|P_\Omega\left(Y_{l+1}\nabla f_{Y_{l+1}}(Z_l)\right)\right\|_F^2}\right\} < \infty.$$

Consequently,

$$\lim_{l\to\infty}\frac{\|\nabla f_{Z_l}(Y_l)\|_F^4}{\|P_\Omega\left(\nabla f_{Z_l}(Y_l)Z_l\right)\|_F^2} = 0, \tag{5.26}$$

$$\lim_{l\to\infty}\frac{\|\nabla f_{Y_l}(Z_{l-1})\|_F^4}{\|P_\Omega\left(Y_l\nabla f_{Y_l}(Z_{l-1})\right)\|_F^2} = 0. \tag{5.27}$$

In particular, (5.26) and (5.27) are also true when l is replaced by the subsequence l_k. Since Z_{l_k} is bounded, one has

$$\begin{aligned}\left\|P_\Omega\left(\nabla f_{Z_{l_k}}(Y_{l_k})Z_{l_k}\right)\right\|_F &\le \left\|\nabla f_{Z_{l_k}}(Y_{l_k})Z_{l_k}\right\|_F \\ &\le \left\|\nabla f_{Z_{l_k}}(Y_{l_k})\right\|_F\|Z_{l_k}\|_F \le C\left\|\nabla f_{Z_{l_k}}(Y_{l_k})\right\|_F\end{aligned}$$

for some $C>0$. Thus

$$\lim_{l_k\to\infty}\frac{\left\|\nabla f_{Z_{l_k}}(Y_{l_k})\right\|_F^4}{C^2\left\|\nabla f_{Z_{l_k}}(Y_{l_k})\right\|_F^2} \le \lim_{l_k\to\infty}\frac{\left\|\nabla f_{Z_{l_k}}(Y_{l_k})\right\|_F^4}{\left\|P_\Omega\left(\nabla f_{Z_{l_k}}(Y_{l_k})Z_{l_k}\right)\right\|_F^2} = 0, \tag{5.28}$$

which gives $\lim_{l_k\to\infty}\nabla f_{Z_{l_k}}(Y_{l_k}) = 0$. Similarly $\lim_{l_k\to\infty}\nabla f_{Y_{l_k}}(Z_{l_k-1}) = 0$ since Y_{l_k} is bounded. □

Lemma 5.6 justifies the second limit in (5.22). The first limit in (5.22) follows directly from Lem. 5.7.

Lemma 5.7. If (Y_{l_k}, Z_{l_k}) are bounded,

$$\lim_{l_k \to \infty} \nabla f_{Y_{l_k}}(Z_{l_k}) = 0. \tag{5.29}$$

Proof. The difference between $\nabla f_{Y_{l_k}}(Z_{l_k})$ and $\nabla f_{Y_{l_k}}(Z_{l_k-1})$ is

$$\begin{aligned}
\nabla f_{Y_{l_k}}(Z_{l_k}) - \nabla f_{Y_{l_k}}(Z_{l_k-1}) &= -Y_{l_k}^T \left(P_\Omega(X^o) - P_\Omega(Y_{l_k} Z_{l_k})\right) \\
&+ Y_{l_k}^T \left(P_\Omega(X^o) - P_\Omega(Y_{l_k} Z_{l_k-1})\right) \\
&= Y_{l_k}^T P_\Omega(Y_{l_k}(Z_{l_k} - Z_{l_k-1})) \\
&= -t_{z_{l_k-1}} Y_{l_k}^T P_\Omega \left(Y_{l_k} \nabla f_{Y_{l_k}}(Z_{l_k-1})\right) \\
&= -\frac{\left\|\nabla f_{Y_{l_k}}(Z_{l_k-1})\right\|_F^2}{\left\|P_\Omega \left(Y_{l_k} \nabla f_{Y_{l_k}}(Z_{l_k-1})\right)\right\|_F^2} Y_{l_k}^T P_\Omega \left(Y_{l_k} \nabla f_{Y_{l_k}}(Z_{l_k-1})\right),
\end{aligned}$$

where the third equation follows from $Z_{l_k} = Z_{l_k-1} + t_{z_{l_k-1}} \nabla f_{Y_{l_k}}(Z_{l_k-1})$. Hence,

$$\begin{aligned}
\left\|\nabla f_{Y_{l_k}}(Z_{l_k}) - \nabla f_{Y_{l_k}}(Z_{l_k-1})\right\|_F &\leq \|Y_{l_k}\|_F \frac{\left\|\nabla f_{Y_{l_k}}(Z_{l_k-1})\right\|_F^2}{\left\|P_\Omega \left(Y_{l_k} \nabla f_{Y_{l_k}}(Z_{l_k-1})\right)\right\|_F} \\
&\leq C \frac{\left\|\nabla f_{Y_{l_k}}(Z_{l_k-1})\right\|_F^2}{\left\|P_\Omega \left(Y_{l_k} \nabla f_{Y_{l_k}}(Z_{l_k-1})\right)\right\|_F} \to 0
\end{aligned} \tag{5.30}$$

where the second inequality follows from the boundedness of Y_{l_k} and the limit follows from (5.27). Combining (5.30) together with (5.23) gives

$$\lim_{l_k \to \infty} \nabla f_{Y_{l_k}}(Z_{l_k}) = 0.$$

□

5.5.2 PF convergence analysis: Proof of Theorem 5.1

Proof of Theorem 5.1. Let (Y_l, Z_l) be a sequence from Alg. 4, with a subsequence (Y_{l_k}, Z_{l_k}) converging to (Y^*, Z^*). Because Z_l minimize $f_{Y_l}(Z)$ over all the $Z \in \mathbb{R}^{r \times n}$,

$$\nabla f_{Y_l}(Z_l) = 0,$$

for all l. By the continuity of the gradients,

$$\nabla f_{Y^*}(Z^*) = \lim_{l_k \to \infty} \nabla f_{Y_{l_k}}(Z_{l_k}) = 0.$$

To prove $\lim_{l_k \to \infty} \nabla f_{Z_{l_k}}(Y_{l_k}) = 0$, let $(\tilde{Y}_l, Z_{l-1})$ be the solution obtained by one step steepest descent as in Alg. 20 from (Y_{l-1}, Z_{l-1}), and $(Y_l, \tilde{Z}_l)$ be the solution obtained by one step steepest descent from (Y_l, Z_{l-1}). Then,

$$\begin{aligned} f(Y_l, Z_{l-1}) &\leq f(\tilde{Y}_l, Z_{l-1}) = f(Y_{l-1}, Z_{l-1}) - \frac{1}{2} \frac{\left\| \nabla f_{Z_{l-1}}(Y_{l-1}) \right\|_F^4}{\left\| P_\Omega \left(\nabla f_{Z_{l-1}}(Y_{l-1}) Z_{l-1} \right) \right\|_F^2}, \\ f(Y_l, Z_l) &\leq f(Y_l, \tilde{Z}_l) = f(Y_l, Z_{l-1}) - \frac{1}{2} \frac{\| \nabla f_{Y_l}(Z_{l-1}) \|_F^4}{\| P_\Omega \left(Y_l \nabla f_{Y_l}(Z_{l-1}) \right) \|_F^2}, \end{aligned}$$

where the inequalities follow from that Y_l and Z_l, by the definition, are the minimal points, and the equalities follow from the same argument as in (5.24) and (5.25). Thus,

$$f(Y_i, Z_i) \leq f(Y_0, Z_0) - \frac{1}{2} \sum_{l=0}^{i-1} \left\{ \frac{\| \nabla f_{Z_l}(Y_l) \|_F^4}{\| P_\Omega \left(\nabla f_{Z_l}(Y_l) Z_l \right) \|_F^2} + \frac{\left\| \nabla f_{Y_{l+1}}(Z_l) \right\|_F^4}{\left\| P_\Omega \left(Y_{l+1} \nabla f_{Y_{l+1}}(Z_l) \right) \right\|_F^2} \right\}.$$

Repeating the argument for Lem. 5.6 yields

$$\nabla f_{Z^*}(Y^*) = \lim_{l_k \to \infty} \nabla f_{Z_{l_k}}(Y_{l_k}) = 0.$$

□

Chapter 6

Conclusions and future directions

This thesis focuses on design and analysis of efficient numerical algorithms for compressed sensing and matrix completion. We close by summarizing the contributions of the thesis and pointing out some potential directions for future work.

For compressed sensing, we have introduced a family of conjugate gradient iterative hard thresholding algorithms (Chapter 3) which combines the low per iteration computational complexity of simple projected gradient descent algorithms with the fast asymptotic convergence rates of more sophisticated algorithms. We provide recovery guarantees in terms of RICs of the measurement matrix for the restarted and projected variants of CGIHT. The large scale empirical tests using the GPU software package *GAGA* establish significant computational advantages of CGIHT in terms of average case phase transitions and overall computational time.

The efficacy of CGIHT for compressed sensing suggests it may well be similarly effective for other sparse approximation problems, such as the model based compressed sensing [5] or the compressed sensing problem in signal space [17, 37]. In a different direction, the non-restarted variant of CGIHT for compressed sensing is the best performing variant, but unfortunately lacks a recovery guarantee. Developing a recovery guarantee is likely to require an analysis of the fraction of a support set correctly identified in each iteration. In the guarantee analysis for the projected variant of CGIHT, the tuning parameter depends on RICs of the measurement matrix. Developing a recovery guarantee without this dependence may also require detailed analysis of the support set in each iteration.

For matrix completion, we have extended NIHT for compressed sensing to the matrix completion setting (Chapter 2). NIHT for matrix completion is observed to be able to recover a random low rank matrix from near the minimum number of

measurements; and the empirical phase transition of NIHT is clearly higher than that of the nuclear norm minimization, particularly for the severely undersampling regions. The observed phase transition of the nuclear norm minimization is consistent with the theory [2, 81]. It is known that the nuclear norm minimization fails with high probability to recover a low rank matrix in the region between the phase transitions of itself and NIHT. This different phase transition phenomenon from the observed universality of phase transitions in compressed sensing [14, 43, 90] may deeply root in the fact that all $m \times n$ matrices of rank r form a smooth manifold in $\mathbb{R}^{m \times n}$, while the set of k-sparse vectors consists of separable k dimensional subspaces; and it requires further investigations by exploring the manifold ideas.

NIHT's ability to recover the largest rank matrices comes at the cost of allowing the method to converge at an extremely slow rate as the rank becomes large. The ability of increasing the recovery region at the cost of slow convergence rates suggests the need for accelerated variants of NIHT. CGIHT for matrix completion (Chapter 4) can further accelerate NIHT without much added computational cost by taking advantage of the fast convergence rate and low per iteration computational cost of the traditional conjugate gradient method. CGIHT also improves the empirical phase transitions of NIHT for both dense and entry sensing.

In Chapter 5, we introduce two alternating minimization algorithms ASD and ScaledASD for matrix completion. The convergence of limit points to stationary points is established. Although ASD and ScaledASD are implemented for fixed rank matrix completion problems, the common rank increasing and decreasing heuristics can be incorporated into them as well. The efficacy of ASD and ScaledASD suggests they may also be effective for other related problems, such as tensor completion [67, 51, 62] or separation of sparse outliers from low rank matrices [27, 102]. For ScaledASD, it would be interesting to try different scaling techniques, especially for more diverse applications. Moreover, a recent paper [61] proves that PowerFactorization with some specific initial point is guaranteed to find the objective matrix with high probability if the number of measurements is sufficiently large; a similar analysis for ASD appears a constructive approach.

Bibliography

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[2] D. Amelunxen, M. Lotz, M. B. McCoy, and J. A. Tropp. Living on the edge: A geometric theory of phase transitions in convex optimization. arXiv:1303.6672, 2014.

[3] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[4] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, 2007.

[5] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

[6] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2007.

[7] C. Berger, Z. Wang, J. Huang, and S. Zhou. Applications of compressive sensing to sparse channel estimation. *IEEE Communications Magazine*, 48(11):164–174, 2010.

[8] J. D. Blanchard, C. Cartis, and J. Tanner. Decay properties for restricted isometry constants. *IEEE Signal Processing Letters*, 16(7):572–575, 2009.

[9] J. D. Blanchard, C. Cartis, and J. Tanner. Compressed sensing: How sharp is restricted isometry property? *SIAM Review*, 53(1):105–125, 2011.

[10] J. D. Blanchard, C. Cartis, J. Tanner, and A. Thompson. Phase transitions for greedy sparse approximation algorithms. *Applied and Computational Harmonic Analysis*, 30(2):188–203, 2011.

[11] J. D. Blanchard, M. Cermak, D. Hanle, and Y. Jing. Greedy algorithms for joint sparse recovery. *IEEE Transactions on Signal Processing*, 62(7):1694–1704, 2014.

[12] J. D. Blanchard and J. Tanner. GAGA: GPU accelerated greedy algorithms. Version 1.1.0 [online], `www.gaga4cs.org`, 2013.

[13] J. D. Blanchard and J. Tanner. GPU accelerated greedy algorithms for compressed sensing. *Mathematical Programming Computation*, 5(3):267–304, 2013.

[14] J. D. Blanchard and J. Tanner. Performance comparisons of greedy algorithms for compressed sensing. *Numerical Linear Algebra with Applications*, 2014. In press.

[15] J. D. Blanchard, J. Tanner, and K. Wei. CGIHT: Conjugate gradient iterative hard thresholing for compressed sensing and matrix completion. Numerical analysis group preprint 14/08, University of Oxford, 2014.

[16] J. D. Blanchard, J. Tanner, and K. Wei. Conjugate gradient iterative hard thresholding: Observed noise stability for compressed sensing. Numerical analysis group preprint 14/03, University of Oxford, 2014.

[17] T. Blumensath. Sampling and reconstructing signals from a union of linear subspaces. *IEEE Transactions on Information Theory*, 57(7):4660 – 4671, 2011.

[18] T. Blumensath. Accelerated iterative hard threshoding. *Signal Processing*, 92:752–756, 2012.

[19] T. Blumensath and M. E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[20] T. Blumensath and M. E. Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):298–309, 2010.

[21] N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In *Advances in Neural Information Processing Systems*, 2011.

[22] S. Boyd and L. Vandenverghe. *Convex Optimization*. Cambridge, 2010.

[23] A. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signal and images. *SIAM Review*, 51(1):34–81, 2009.

[24] W. Bruns and U. Vetter. *Determinantal Rings (Lecture Notes in Mathematics, 1327)*. Springer-Verlag, 1988.

[25] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[26] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus de l'Acandemie Des Sciences*, Serie I:589–592, 2008.

[27] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(1):1–37, 2011.

[28] E. J. Candès and Y. Plan. Tight oracle bounds for low-rank matrix recovery from a minimal number of random measurements. *IEEE Transactions on Information Theory*, 57(4):2342–2359, 2009.

[29] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[30] 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(2):489–509, 2006.

[31] E. J. Candès and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[32] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–1080, 2009.

[33] V. Cevher. An ALPS view of sparse recovery. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2011.

[34] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009.

[35] W. Dai, O. Milenkovic, and E. Kerman. Subspace evolution and transfer (SET) for low-rank matrix completion. *IEEE Transactions on Signal Processing*, 59(7):3120–3132, 2011.

[36] W. Dai, M. A. Sheikh, O. Milenkovic, and R. G. Baraniuk. Compressive sensing DNA microarrays. *Eurosip Journal on Bioinformatics and Systems Biology*, Vol. 2009, 2009.

[37] M. A. Davenport, D. Needell, and M. B. Wakin. Signal space CoSaMP for sparse recovery with redundant dictionaries. *IEEE Transactions on Information Theory*, 59(10):6820–6829, 2013.

[38] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[39] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming Series A*, 2014.

[40] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[41] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Science*, 100(5):2197–2202, 2003.

[42] D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

[43] D. L. Donoho and J. Tanner. Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. *Philosophical Transactions of the Royal Society A*, 367(1906):4273–4293, 2009.

[44] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, 2008.

[45] Y. C. Eldar, D. Needell, and Y. Plan. Unicity conditions for low-rank matrix recovery. *Applied and Computational Harmonic Analysis*, 33(2):309–314, 2012.

[46] L. Eldén. *Matrix Methods in Data Mining and Pattern Recognization.* SIAM, 2007.

[47] R. Escalante and M. Raydan. *Alternating Projection Methods.* SIAM, 2011.

[48] S. Foucart. Sparse recovery algorithms: Sufficient conditions in terms of restricted isometry constants. In *Approximation Theory XIII: San Antonio 2010, Springer Proceedings in Mathematics*, 2010.

[49] S. Foucart. Hard thresholding pursuit: An algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

[50] S. Foucart and M.-J. Lai. Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 26(3):395–407, 2009.

[51] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[52] D. Goldfarb and S. Ma. Convergence of fixed-point continuation algorithms for matrix rank minimization. *Foundations of Computational Mathematics*, 11(2):183–210, 2011.

[53] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins Press, 2013.

[54] A. Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, 1997.

[55] L. Grippo and M. Sciandrone. Gloabally convergence block-coordinate techniques for unconstrained optimization. *Optimization Methods and Software*, 10(4):587–637, 1999.

[56] D. Gross. Recovering low-rank matrices from few coefficients in any basis. *IEEE Transactions on Information Theory*, 57(3):1548–1566, 2011.

[57] J. P. Haldar and D. Hernando. Rank-constrained solutions to linear matrix equations using PowerFactorization. *IEEE Signal Processing Letters*, 16(7):584–587, 2009.

[58] N. J. A. Harvey, D. R. Karger, and S. Yekhanin. The complexity of matrix completion. In *Proceedings of the seventeenth annual ACM-SIAM symposium on discrete algorithms*, 2006.

[59] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[60] P. Jain, R. Meka, and I. Dhillon. Guaranteed rank minimization via singular value projection. In *Proceedings of the Neural Information Processing Systems Conference*, 2010.

[61] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. arXiv:1212.0467, 2012.

[62] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 2013. In press.

[63] A. Kyrillidis and V. Cevher. Matrix ALPS: Accelerated low rank and sparse matrix reconstruction. arXiv:1203.3864v4, 2012.

[64] A. Kyrillidis and V. Cevher. Matrix recipes for hard thresholding methods. *Journal of Mathematical Imaging and Vision*, 48(2):235–265, 2014.

[65] R. M. Larsen. PROPACK (software package). `http://sun.stanford.edu/~rmunk/PROPACK/`.

[66] K. Lee and Y. Bresler. ADMiRA: Atomic decomposition for minimum rank approximation. *IEEE Transactions on Information Theory*, 128(1):4402–4416, 2010.

[67] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013.

[68] 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(3):1235–1256, 2009.

[69] M. Lustig, D. L. Donoho, and J. Pauly. The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[70] M. Lustig, D. L. Donoho, J. Santos, and J. Pauly. Compressed sensing MRI. *IEEE Signal Processing Magazine*, 25(2):72–82, 2008.

[71] A. Maleki. Approximate message passing algorithms for compressed sensing. thesis, 2010.

[72] A. Maleki and D. L. Donoho. Optimally tuned iterative reconstruction algorithms for compressed sensing. *IEEE Journal of Selected Topics in Signal Processing*, 56(9):330–341, 2010.

[73] M. Michenkova. Numerical algorithms for low-rank matrix completion problems. `http://www.math.ethz.ch/~kressner/students/michenkova.pdf`, 2011.

[74] B. Mishra, K. A. Apuroop, and R. Sepulchre. A Riemannian geometry for low-rank matrix completion. arXiv:1306.2672, 2013.

[75] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre. Fixed-rank matrix factorizations and Riemannian low-rank optimization. *Computational Statistics*, 2013.

[76] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[77] D. Needel and J. A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[78] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.

[79] T. Ngo and Y. Saad. Scaled gradients on Grassmann manifolds for matrix completion. In *Advances in Neural Information Processing Systems*, 2012.

[80] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[81] S. Oymak and B. Hassibi. New null space results and recovery thresholds for matrix rank minimization. arXiv:1011.6326, 2010.

[82] V. Patel, G. Easley, D. Healy, and R. Chellappa. Compressed synthetic aperture radar. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):244–254, 2010.

[83] M. J. D. Powell. Some convergence properties of the conjugate gradient method. *Mathematical Programming*, 11(1):42–49, 1976.

[84] B. Recht. A simpler approach to matrix completion. *The Journal of Machine Learning Research*, 12:3413–3430, 2011.

[85] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[86] B. Recht, W. Xu, and B. Hassibi. Null space conditions and thresholds for rank minimization. *Mathematical Programming Series B*, 127:175–211, 2011.

[87] C. E. Shannon. Communincations in the presence of noise. *Proceedings of the IEEE*, 86(2):10–21, 1998.

[88] ACM SIGKDD and Netflix. Proceedings of kdd cup and workshop. `http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings.html`.

[89] T. Strohmer and R. W. Heath. Grassmannian frames with applications to coding and communication. *Applied and Computational Harmonic Analysis*, 12(3):257–275, 2004.

[90] B. Sturm. Sparse vector distributions and recovery from compressed sensing. arXiv:1103.6246v2, 2011.

[91] J. Tanner and K. Wei. Normalized iterative hard thresholding for matrix completion. *SIAM Journal on Scientific Computing*, 35(5):S104–S125, 2013.

[92] J. Tanner and K. Wei. Low rank matrix completion by alternating steepest descent methods. Numerical analysis group preprint 14/10, University of Oxford, 2014.

[93] K. C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 4.0(beta) (software package). `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`, 2006.

[94] K. C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 6:615–640, 2010.

[95] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[96] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[97] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[98] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[99] B. Vandereycken. Low rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[100] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a non-linear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361, 2012.

[101] J. Whittaker. *Interpolatory Function Theory, Issue 33*. The University Press, 1935.

[102] S. Yuan, Z. Wen, and Y. Zhang. Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optimization Methods and Software*, 29(2):239–263, 2014.

[103] Y. X. Yuan. Analysis on the conjugate gradient method. *Optimization Methods and Software*, 2(1):19–29, 1993.