INERTIAL ITERATIVE THRESHOLDING WITH APPLICATIONS TO SPARSE AND LOW-RANK SIGNAL RECOVERY

BY

PATRICK ROYCE JOHNSTONE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor Pierre Moulin

ABSTRACT

This thesis is concerned with a class of methods known collectively as iterative thresholding algorithms. These methods have been used by researchers for several decades to solve various optimization problems that arise in signal processing, inverse problems, pattern recognition and other related fields. One such problem of great interest is compressed sensing, where the goal is to recover a signal that is known to be sparse from fewer linear measurements than the dimension of the signal. Another is low-rank matrix completion where one wants to recover a low-rank matrix from a subset of revealed entries. A third example is robust principle component analysis (RPCA) where one is given a data matrix and would like to decompose it into a low-rank component and a sparse component. Other examples include total-variation denoising and deblurring, and ℓ_1-regularized regression.

Iterative thresholding methods have low complexity, but they typically take many iterations to converge, especially on ill-conditioned problems. In this thesis we explore how *inertia* can be used to accelerate iterative thresholding algorithms. A second problem with iterative thresholding algorithms is they tend to become trapped in undesirable local minima when the problem is non-convex. We discuss how inertia can help iterative thresholding methods to avoid local minima and propose several schemes to solve well-known non-convex problems.

To my family, for their love and support.

ACKNOWLEDGMENTS

First, I thank my adviser, Professor Pierre Moulin, for his support and wisdom. I'd like to thank Professor Olgica Milenkovic for her insights and advice. Next I'd like to thank Amin Emad for his help with developing several ideas and for many illuminating discussions. Algorithm 6 was Amin's idea. Next I thank my office mates in our Beckman home. Finally I thank my family for being there through tough times and supporting me from the other side of the world. Without them I could not achieve anything.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Every year humans are acquiring, generating, communicating and storing increasingly huge amounts of data. For instance there are now over a billion users on the social network Facebook. These users 'like' more than a staggering 30 thousand brands and organizations every minute [1]. Other websites such as Google, Amazon and Netflix also acquire enormous amounts of data every day. The (over)abundance of data in a wide variety of fields is placing pressure on signal processing algorithms to keep up.

Compressed sensing and low-rank matrix recovery are attempts to address the problem of acquiring certain high dimensional data. Both techniques are based on the following principle: signals encountered in a wide variety of applications are highly structured. In spite of their high dimensionality, many signals we encounter rely on relatively few intrinsic parameters in known basis expansions. Indeed many compression schemes depend on this observation. For instance JPEG2000 relies on the fact that most images can be accurately represented by very few parameters in a wavelet basis [2].

In low-rank matrix recovery, the measure of structure is rank. A low-rank matrix can be represented by far fewer parameters than its ambient dimension. From the singular value decomposition (SVD) it can be shown that an $m \times n$ matrix of rank r depends on only $r(m+n-r)$ degrees of freedom. When r is small this is far fewer than $m \times n$. In compressed sensing the objects of interest are vectors and the measure of structure is *sparsity*, which counts the number of nonzero elements in the vector. A vector with k non-zero entries has essentially $k+1$ intrinsic parameters: k unknown values and 1 parameter encoding the support set. Other measures of structure have also been suggested [3]. The unifying concept is this: lots of structure means very few intrinsic parameters.

Low rank matrix recovery and compressed sensing are different from traditional compression techniques in a fundamental way, although they both

depend on structure. In JPEG2000 one must have the full image in order to transform it to a wavelet basis for compression. Similarly one must have the full matrix to compute its SVD and compress. Matrix recovery and compressed sensing attempt to *acquire* the signal in a compressed form.

It has been shown that convex optimization methods can be deployed successfully in compressed sensing and matrix recovery under certain conditions. Unfortunately these convex optimization methods are not currently scalable to problems with billions (or even millions) of unknowns. For several years now there has been a push for alternatives that can scale to truly massive data sets. Iterative thresholding methods are one possible approach that can scale to large dimensions.

In this thesis we propose several simple low complexity modifications to several iterative thresholding methods which provide accelerated convergence and improved recovery performance. For the ℓ_1-regularized least-squares problem the modified algorithm is shown to achieve an asymptotic linear convergence rate. Furthermore this rate is shown to be faster than that of the well-known iterative soft thresholding (IST) algorithm. Remarkably, the improvement is gained with negligible additional computations. The speed-up is significant when the sub-matrix corresponding to the support of the solution is ill-conditioned. The result extends the analysis pioneered by Polyak in [4] of the heavy-ball method to a problem which is not smooth, not quadratic and not strongly-convex.

1.1 Outline of Thesis

The thesis is organized as follows. In chapter 2 we introduce proximal splitting methods for solving certain convex optimization problems. We then introduce the notion of inertia and its use in proximal methods. In sections 2.3 and 2.4 we apply inertial proximal methods to develop new approaches to robust PCA and low-rank matrix recovery respectively. In sections 2.5 to 2.8 we develop a new algorithm based on inertial proximal methods for the well-known LASSO problem with applications to compressed sensing. We also determine the rate of convergence of the method and provide experiments comparing it with several well-known alternative methods. In chapter 3 we consider how inertia can be used in non-convex approaches to low-rank

matrix recovery and compressed sensing. We provide numerical experiments which attest to the potential of inertia in these problems.

CHAPTER 2

INERTIAL ITERATIVE SOFT THRESHOLDING

2.1 Proximal Splitting Methods

We are interested in the following problem:

$$\min_x F(x) \triangleq f(x) + r(x) \tag{2.1}$$

where f is convex and smooth and r is convex but not necessarily smooth and $x \in \mathbb{R}^N$. We require that r is lower semi-continuous which allows r to be the indicator for a closed convex set $\mathcal{C}$ meaning $r(x)$ is zero when $x \in \mathcal{C}$ and ∞ otherwise. A minimum of F must satisfy the inclusion

$$0 \in \nabla f(x) + \partial r(x). \tag{2.2}$$

where ∇ and ∂ are the gradient and sub-gradient operators respectively. The forward-backward algorithm is a well-known iterative algorithm for solving problem (2.1) consisting of two steps. The "forward" step involves computation of a gradient descent step with respect to the smooth component f. The "backward" step is the application of a proximal operator to the result of the forward step. The proximity operator of a convex function r is defined explicitly as follows:

$$\text{prox}_r(y) \triangleq \arg\min_z r(z) + \frac{1}{2}\|z - y\|^2. \tag{2.3}$$

Since r is convex the above problem is strongly convex and prox is a well defined function. Implicitly if $x = \text{prox}_r(y)$ then

$$0 \in \partial r(x) + x - y. \tag{2.4}$$

The proximity operator can be thought of as a generalization of projection onto a convex set. Indeed if $r(x)$ is the indicator function of a closed convex set, then the proximity operator is the convex projection onto that set. For many functions of interest the proximity operator can be computed efficiently [5].

The iterations of the basic forward-backward algorithm are

$$x^{k+1} = \text{prox}_{\tau_k r}\left(x^k - \tau_k \nabla f(x^k)\right) \tag{2.5}$$

starting at some arbitrary point x_0 and with explicit prescriptions on the step-size τ_k. If (x^*, τ^*) is a fixed point of the algorithm, it obeys

$$x^* \quad = \quad \text{prox}_{\tau^* r}\left(x^* - \tau^* \nabla f(x^*)\right) \tag{2.6}$$

which implies

$$\begin{aligned} 0 \quad &\in \quad \tau^* \partial r(x^*) + x^* - (x^* - \tau^* \nabla f(x^*)) && (2.7) \\ &\in \quad \partial r(x^*) + \nabla f(x^*). && (2.8) \end{aligned}$$

Therefore if the algorithm converges, it converges to a minimum of (2.1). Convergence can be proved with the additional assumption that ∇f is Lipschitz continuous. If ∇f is L-Lipschitz continuous and $0 < \tau_k < 2/L$ for all k, then x^k converges to a minimum of (2.1). Furthermore if f is strongly convex, then x^k converges at a geometric (linear) rate [5].

When f is not strongly convex it is difficult to obtain estimates of the rate of convergence of the iterates x^k [5]. However if one considers specific choices of $r(x)$, fast rates can be established. For instance if $r(x) = \lambda \|x\|_1$ then x^k converges to a minimizer at a linear rate so long as $0 < \tau_k < 2/L$ [6].

We can also consider the behavior of the objective values $F(x^k) - F(x^*)$. For the forward-backward algorithm this is known to be $O(1/k)$ [5]. Fast Iterative Soft Thresholding (FISTA) is a so-called accelerated variation of the forward-backward algorithm because it achieves $F(x^k) - F(x^*) = O(1/k^2)$ [7]. This rate is optimal in the sense that it is the best rate possible for some specific choices of $F(x)$. However the rate at which the iterates x^k of FISTA converge is unknown. Furthermore if the minimizer of (2.1) is not unique then convergence of $F(x^k)$ does not guarantee convergence of x^k.

Algorithm 1 FISTA [7]

Require: Lipschitz constant L of ∇f and a stopping criterion.

1: Fix x_0, set $z_0 = x_0$, $t_0 = 1$.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad y_k = z_k - \frac{1}{L}\nabla f(z_k)$
4: $\quad x^{k+1} = \text{prox}_{L^{-1}r} y_k$
5: $\quad t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
6: $\quad z_{k+1} = x^{k+1} + \frac{t_k - 1}{t_{k+1}}(x^{k+1} - x^k)$
7: **end for**
8: **return** x^{k+1}

2.2 Inertial Proximal Splitting

In [8], Moudafi and Oliny introduced an inertial variant of the forward-backward algorithm which was inspired by the heavy ball method proposed by Polyak [4]. Consider the following ODE:

$$\dot{x} = -\alpha \nabla f(x) \tag{2.9}$$

which is a continuous analog of the basic gradient descent algorithm

$$x^{k+1} = x^k - \tau_k \nabla f(x^k). \tag{2.10}$$

The heavy ball method begins instead with the ODE

$$m\ddot{x} + \gamma \dot{x} = -\alpha \nabla f(x) \tag{2.11}$$

which describes the path of a particle of mass m experiencing friction with constant γ and subject to a conservative force field $\nabla f(x)$. It is often referred to as the heavy ball with friction ODE [4]. Equation (2.9) is the limit of (2.11) as mass m goes to zero.

We are interested in designing algorithms for digital computers and must consider discrete versions of (2.9) and (2.11). The standard gradient descent algorithm arises from an explicit discretization of (2.9). Similarly, Polyak's heavy ball method results from an explicit discretization of (2.11). The

iterates of Polyak's HB method are

$$x^{k+1} = x^k - \tau_k \nabla f(x^k) + \beta_k (x^k - x^{k-1}) \tag{2.12}$$

The optimal choice of τ_k and β_k are explicit constants that depend on the condition number of the Hessian. The conjugate gradient method is a special case of Polyak's HB method where the parameters τ_k and β_k are chosen at each iterate by a line search. For a strongly convex objective function, gradient descent, HB and conjugate gradient all achieve a linear convergence rate; however, Polyak's method and conjugate gradient achieve a significantly better geometric constant. The advantage is significant when the Hessian at the minimum is ill-conditioned [4]. The extra inertia term helps to smooth out the zig-zagging of the gradient on ill conditioned problems. Furthermore the rate achieved by HB and conjugate gradient on strongly convex functions is the best possible for any iterative first order method which computes the next iteration as a linear combination of previous estimates and gradients [4].

Moudafi and Oliny extended Polyak's method to proximal splitting problems including (2.1). We will refer to Moudafi and Oliny's algorithm as the inertial splitting method (ISM) which is outlined in algorithm (2) for general F. To analyze the algorithm we need to introduce the concept of a γ-*co-coercive* operator [9].

Definition An operator $T : \mathcal{H} \to \mathcal{H}$ where $\mathcal{H}$ is a Hilbert space is γ-*co-coercive* with respect to a set $S \subset \mathcal{H}$ if there exists $\gamma > 0$ such that $\langle T(x) - T(y), x - y \rangle \geq \gamma \|T(x) - T(y)\|^2$ for all $x \in \mathcal{H}, y \in S$.

A monotone operator satisfies $\langle T(x) - T(y), x - y \rangle \geq 0$ for all x and y. Thus a co-coercive operator is monotone. A strongly monotone operator satisfies $\langle T(x) - T(y), x - y \rangle \geq c\|x - y\|^2$ for some $c > 0$ and for all x and y. Thus a co-coercive operator is not necessarily strongly monotone. However a strongly monotone Lipschitz operator is co-coercive [9]. In general if T is γ-co-coercive with respect to $\mathcal{H}$ then it is Lipschitz continuous with constant $L \leq 1/\gamma$. If $f(x) = \frac{1}{2}\|y - Ax\|_2^2$ then $\nabla f(x) = A^T(Ax - y)$. $\nabla f(x)$ is γ-co-coercive with respect to $\mathbb{R}^n$ for $\gamma \leq 1/\lambda_{\max}(A^T A)$ and Lipschitz continuous with Lipschitz parameter $1/\gamma$ [9].

Algorithm 2 Moudafi and Oliny's ISM [8]

Require: γ, co-coerciveness constant of $\nabla f(x)$ with respect to the solution set $X^* = \{x \in \mathbb{R}^n : 0 \in \nabla f(x) + \partial r(x)\}$. $\tau_{\min}$, sequences β_k and τ_k satisfying $0 \leq \beta_k < 1$, $0 < \tau_{\min} \leq \tau_k \leq 2\gamma$ and a stopping criterion.

1: Fix x_0.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad x^{k+1} = \text{prox}_{\tau_k r}\left(x^k - \tau_k \nabla f(x^k) + \beta_k(x^k - x^{k-1})\right)$
4: **end for**
5: **return** x^{k+1}

Moudafi and Oliny analyzed the convergence of their algorithm and the result is stated in the following theorem.

Theorem 1 ([8]) *Suppose ∇f is γ-co-coercive with respect to the solution set. If the following condition holds*

$$\sum_{k=1}^{\infty} \beta_k \|x^k - x^{k-1}\|^2 < \infty \tag{2.13}$$

then the output x^k of algorithm 2 converges to a minimizer of (2.1) .

Inequality (2.13) can be enforced by setting

$$\beta_k = \frac{1}{k^2 \|x^k - x^{k-1}\|^2}. \tag{2.14}$$

It was also shown in [10] that choosing the β_k's to be non-decreasing and $0 \leq \beta_k < 1/3$ ensures condition (2.13) is met.

It is worth pointing out the differences between algorithms 1 and 2. Both methods employ inertia at each iteration to produce an extrapolated point $x_k + \beta_k(x^k - x^{k-1})$. The main difference is that FISTA computes the next gradient at the extrapolated point, whereas ISM computes the gradient at the previous point. FISTA requires the step-size to be less than $1/L$ whereas ISM allows step-sizes up to $2/L$. FISTA provides a specific choice of the sequence β_k although this seems to be primarily to allow for theoretical guarantees and numerical evidence suggests that more straightforward choices, such as a constant value, work just as well. In terms of theoretical guarantees, unlike ISM, FISTA has no guarantee that x^k will converge. FISTA guarantees an optimal worst-case rate of convergence of the objective function. For ISM, the rate of convergence of the objective function values has not been determined

for general F. In section 2.6 we determine the rate of convergence of x^k and $F(x^k)$ for the specific choice $r(x) = \lambda\|x\|_1$.

2.3 Low Rank Plus Sparse Matrix Decomposition

The ISM algorithm can be applied to various non-smooth optimization problems of interest. We begin with robust PCA, which is the most general one to be considered in this paper. It is the problem of decomposing a matrix D into a low-rank component M and sparse component E [11]. A popular approach is to solve the following convex non-smooth problem.

$$\min_{M,E} \|M\|_* + \lambda\|E\|_1 + \frac{1}{2\gamma}\|D - (M + E)\|_F^2. \tag{2.15}$$

The nuclear norm $\|\cdot\|_*$ and ℓ_1 norm $\|\cdot\|_1$ are the "best" convex approximations to the rank function and ℓ_0 pseudo norm respectively. Under certain assumptions on D the minimizers M^* and E^* can be shown to have low-rank and high sparsity respectively, and satisfy $D \approx M^* + E^*$. In practice a continuation scheme is normally used whereby the value of γ is decreased towards 0 so that the error $\|D-(M+E)\|_F^2$ is controlled. We will not consider a continuation scheme although it can be incorporated into our algorithm easily. In [11] the authors showed that setting $\lambda = 1/\sqrt{\max\{n_1, n_2\}}$, where M is of size $n_1 \times n_2$, has good recovery performance.

Problem (2.15) is an instance of problem (2.1) with $x = (M, E)$, $f(x) = \frac{1}{2}\|D - (M + E)\|_F^2$, the coerciviness constant of $\nabla f(x)$ is 2 and $r(x) = \gamma\|M\|_* + \gamma\lambda\|E\|_1$. prox_g has a closed form solution given below.

$$\left\{\text{prox}_{\tau\|\cdot\|_1} Y\right\}_{ij} = \max\left(|Y_{ij}| - \tau, 0\right)\text{sgn}(Y_{ij}). \tag{2.16}$$

If $Y = USV^T$ is the SVD of Y, then

$$\text{prox}_{\tau\|\cdot\|_*} Y = U\text{prox}_{\tau\|\cdot\|_1}(S)\, V^T. \tag{2.17}$$

We can now specialize Moudafi and Oliny's algorithm to problem (2.15). We call the algorithm Inertial RPCA and it is shown as algorithm 3 below.

Algorithm 3 Inertial RPCA (I-RPCA)

Require: $\tau_{\min}$, sequences β_k and τ_k satisfying $0 \leq \beta_k < 1$, $0 < \tau_{\min} \leq \tau_k \leq 1$ and a stopping criterion.

Fix M_0, E_0.
for $k = 1, 2, \ldots$ **do**
  $G_M = M_k - \tau_k(M_k + E_k - D) + \beta_k (M_k - M_{k-1})$
  $G_E = E_k - \tau_k(M_k + E_k - D) + \beta_k (E_k - E_{k-1})$
  $M_{k+1} = \text{prox}_{\tau\|\cdot\|_*} G_M$
  $E_{k+1} = \text{prox}_{\tau\|\cdot\|_1} G_E$
end for
return M_{k+1}, E_{k+1}

2.3.1 Robust PCA Simulations

Consider U and V of size 20×2 with entries drawn i.i.d. $\mathcal{N}(0, 1)$ and form $M^* = UV^T$. Note that the rank of M is at most 2. Now form the 20×20 matrix E^* by randomly selecting a subset of 100 entries and setting them i.i.d. $\mathcal{N}(0, 1)$ and setting all other entries to 0. Let $L = M + E$. We attempt to decompose L into low-rank and sparse components by solving Problem (2.15) using the forward-backward algorithm, IRPCA and FISTA. The forward-backward algorithm corresponds to algorithm 3 with $\beta = 0$ and $\tau = 1$, for I-RPCA we chose $\beta = 1/4$ and $\tau = 1$. Note that this choice of β satisfies Moudafi and Oliny's sufficient condition, equation (2.13). For FISTA we used the recommended choices of β and τ from [7]. The results are shown in figure 2.1. The optimal function value was computed by running IST for 1000 iterations. Note that all three algorithms converged to the same minimizer, however I-RPCA was the fastest. While these experimental results are promising, more theoretical work is necessary to understand the rate of convergence behavior of I-RPCA.

2.4 Low Rank Matrix Recovery

Moudafi and Oliny's algorithm can also be applied to low-rank matrix recovery. In this problem we would like to recover a low-rank matrix given noisy linear measurements of the entries. When the linear measurements are samples of individual entries, this is low-rank matrix completion [12]. One

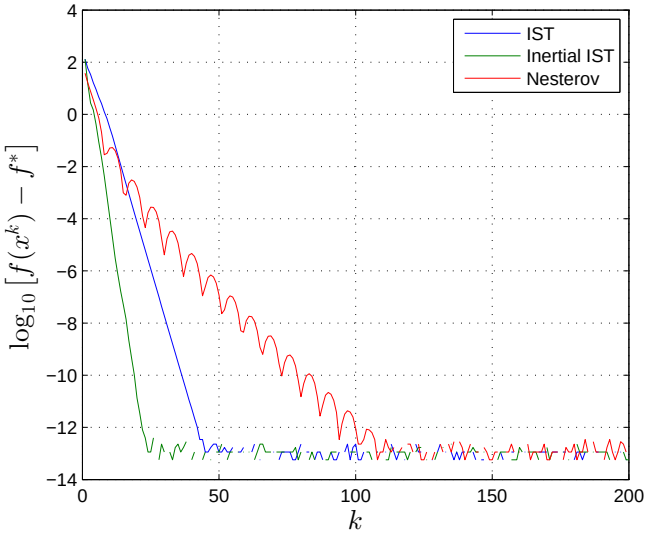

Figure 2.1: Numerical Simulation for Robust PCA

approach is to solve the nuclear norm regularized least squares problem

$$\min_M \gamma\|M\|_* + \frac{1}{2}\|\mathcal{A}(M) - y\|_2^2 \tag{2.18}$$

where $\mathcal{A}$ is the linear operator. Under certain conditions on the underlying low-rank matrix and the linear operator, the solution to (2.18) is the true matrix we want to recover [12].

Algorithm 4 Inertial Low-rank matrix recovery

Require: $\tau_{\min}$, sequences β_k and τ_k satisfying $0 \leq \beta_k < 1$, $0 < \tau_{\min} \leq \tau_k \leq 2/\lambda_{\max}(\mathcal{A}^*\mathcal{A})$ and a stopping criterion.

1: Fix M_0.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad G_M = M_k - \tau_k \mathcal{A}^* \left(\mathcal{A}\left(M_k\right) - y\right) + \beta_k \left(M_k - M_{k-1}\right)$
4: $\quad M_{k+1} = \text{prox}_{\tau\|\cdot\|_*} G_M$
5: **end for**
6: **return** M_{k+1}

As with robust PCA a continuation scheme can be incorporated into algorithm 4 to find an appropriate value for γ. Nesterov's method was applied to problem (2.18) in [13]. The resulting algorithm is an instance of algorithm 1 with the nuclear norm proximity operator. Numerical results for the low-rank matrix recovery problem are presented below.

2.4.1 Matrix Completion Simulation

We show a numerical experiment to compare the convergence rates of Algorithm 4 with FISTA and the forward-backward algorithm. Consider U and V of size 20×2 with entries drawn i.i.d. $\mathcal{N}(0, 1)$. We form $X = UV^T$ which is of rank at most 2. We choose a set of entries of size 200 randomly from all 400 entries and attempt to recover X from the known entries by solving problem 2.18 for $\gamma = 1$ using algorithm 4 with

1. $\beta = 0.3$, $\tau = 2$ (algorithm 4).
2. $\beta = 0$, $\tau = 2$ (the forward backward algorithm)
3. FISTA with recommended β_k and τ.

The results are shown in figure 2.2. The three algorithms converged to the same solution X^*. Note that $X^* \neq X$. In order to recover X we should use a continuation scheme to select a good choice of γ. Since we have noiseless measurements the continuation scheme would have to have $\gamma_k \to 0$. The purpose of this simulation is to demonstrate the various speeds with which the three algorithms converge to the solution of problem (2.18) for fixed γ. It is a different matter to determine whether the solution of (2.18) actually recovers X; see [12] for more details. As we can see, algorithm 4 achieved the fastest convergence rate. However more study is required to provide theoretical guarantees on the rate of convergence of these algorithms.

2.4.2 Low Rank Matrix Recovery Simulation

Consider U and V of size 20×2 with entries drawn i.i.d. $\mathcal{N}(0, 1)$. We from $X = UV^T$ which is of rank at most 2. We draw a matrix A of size 200×400 with entries drawn i.i.d. $\mathcal{N}(0, 1)$ and compute 200 linear measurements $y = AX$. Our goal is to recovery X from y by solving problem (2.18) using each of the three discussed algorithms. Specifically

1. $\beta = 0.95$, $\tau = 2$ (algorithm 4).
2. $\beta = 0$, $\tau = 2$ (the forward backward algorithm)
3. FISTA with recommended β_k and τ.

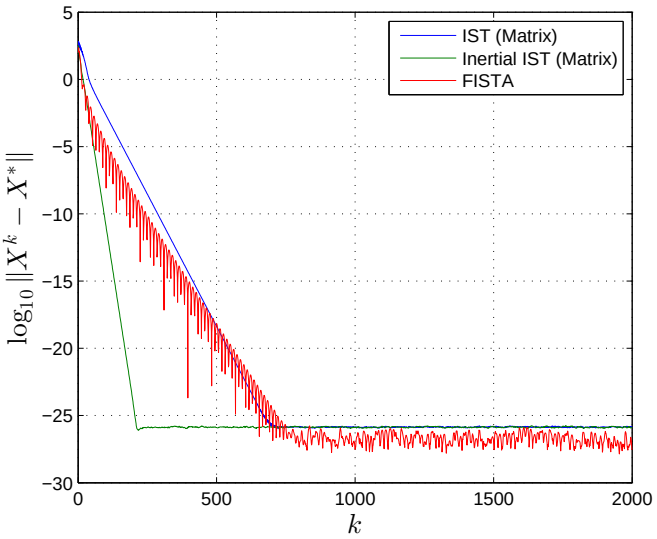

Figure 2.2: Numerical Simulation of Error Convergence for Matrix Completion

Note that we used a much larger value for β than for matrix completion. This was because we found in simulations that smaller values of β resulted in much slower convergence rates than FISTA. This tuning of β is not really a fair comparison and more theoretical work is required to determine the best choice of β in algorithm 4. In the following sections we provide precise optimal values for β for the problem which arises in compressed sensing where the ℓ_1 norm is used. However, the nuclear norm has very different characteristics and it seems that a different analysis is required to pin down the convergence rate in this case. Considering the performance shown in figure 2.3, observe that FISTA and algorithm 4 are superior to the forward backward algorithm in this experiment.

2.5 Compressed Sensing

In compressed sensing we wish to recover a sparse vector x from fewer linear measurements than the length of x. One approach is to solve the LASSO problem which tends to have a sparse solution and under certain conditions on the sparsity and the measurement matrix A will recover the true vector x [14]. In fact LASSO is of interest in many other contexts beyond compressed sensing such as regularized regression in statistics and machine learning. The

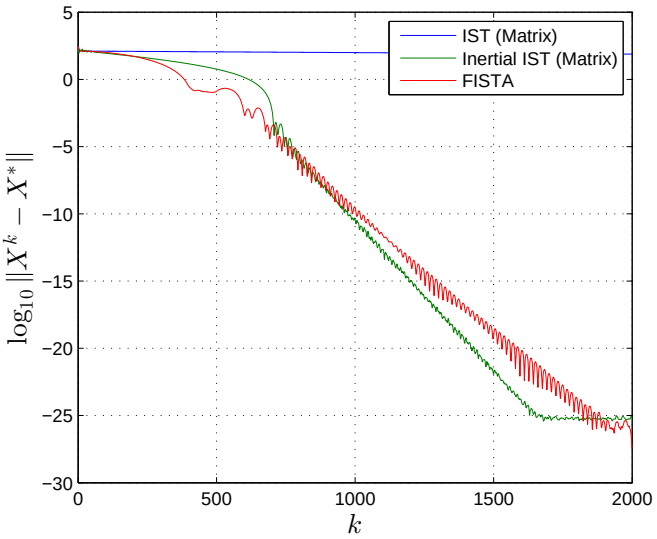

Figure 2.3: Numerical Simulation of Error Convergence for Matrix Recovery

LASSO problem is

$$\min_x \gamma\|x\|_1 + \frac{1}{2}\|y - Ax\|^2 \tag{2.19}$$

which is an instance of (2.1) with $f = \frac{1}{2}\|y - Ax\|^2$ and $r(x) = \gamma\|x\|_1$. The forward-backward algorithm specialized to this problem is known as iterative soft thresholding (IST). As was mentioned in section 2.1, for this particular problem the forward backward algorithm has been shown to converge at a linear rate under some assumptions on the limit point [6]. The convergence is affected by the conditioning of the sub-Hessian corresponding to the support of the limit.

The proximity operator for the ℓ_1 norm has been given in section 2.3 for matrices and is the same element-wise soft thresholding for vectors. Let the ℓ_1 proximity operator be given by $S_\lambda(y) \triangleq \text{prox}_{\lambda\|\cdot\|_1}(y)$.

2.6 Convergence for LASSO

We now show that Inertial-IST (I-IST) achieves a linear convergence rate for the LASSO problem. In section 2.7 we will show how the convergence rate can be made faster than IST through an intelligent choice of the sequences τ_k and β_k. First we will discuss the properties of the set of minimizers of

Algorithm 5 Inertial-IST

Require: $\tau_{\min}$, sequences β_k and τ_k satisfying $0 \leq \beta_k < 1$, $0 < \tau_{\min} \leq \tau_k \leq 2/\lambda_{\max}(A^TA)$ and a stopping criterion.

1: Fix x_0.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad g_k = x^k - \tau_k A^T(Ax^k - y)) + \beta_k(x^k - x^{k-1})$
4: $\quad x^{k+1} = S_{\tau_k\gamma}(g_k)$
5: **end for**
6: **return** x^{k+1}

problem (2.19). Let $g(x) \triangleq \nabla f(x)$.

Theorem 2 ([6]) *Let X^* be the set of optimal solutions of (2.19). $x^* \in X^*$ if and only if $g(x^*) = g^*$ where for all i*

$$\frac{g_i^*}{\gamma} \begin{cases} = -1 : sgn(x_i^*) > 0 \\ = +1 : sgn(x_i^*) < 0 \\ \in [-1, 1] \quad else. \end{cases} \tag{2.20}$$

Furthermore there exists some constant g^* such that

$$g(x^*) = g^* \ \forall x^* \in X^*. \tag{2.21}$$

Note that for all $i \in \text{supp}(x^*)$, $\text{sgn}(x_i^*) = -g_i^*/\gamma$. Let

$$h^k(x, y) \triangleq x - \tau_k \nabla f(x) + \beta_k(x - y) \tag{2.22}$$

and

$$\bar{h}^k(x) \triangleq x - \tau_k \nabla f(x). \tag{2.23}$$

In what follows assume $\tau_k \to \tau_*$ and $\beta_k \to \beta_*$. Let

$$\bar{h}(x) \triangleq x - \tau_* \nabla f(x). \tag{2.24}$$

Lemma 3 ([6])

$$\|\bar{h}^k(x) - \bar{h}(y)\| \leq \|x - y\| + o(1). \tag{2.25}$$

We will need the following properties of S_λ.

Lemma 4 ([6]) *If $|y| \geq \lambda$ and $sgn(x) \neq sgn(y)$ then*

$$|S_\lambda(x) - S_\lambda(y)| \leq |x - y| - \lambda. \tag{2.26}$$

Theorem 5 *Assume condition (2.13) holds. Then by theorem 2.1 the output of algorithm 5 converges to some x^* which minimizes (2.19). Let $supp(x^*) = \{i : x_i^* \neq 0\}$ then for all but finitely many k*

$$x_i^k = 0, \forall i \notin supp(x^*), \tag{2.27}$$

and

$$\begin{aligned} sgn\left(h_i^k(x^k, x^{k-1})\right) &= sgn\left(\bar{h}_i(x^*)\right) && (2.28)\\ &= sgn(x_i^*) && (2.29)\\ &= -\frac{\nabla f(x^*)_i}{\gamma} \ \forall i \in supp(x^*) && (2.30) \end{aligned}$$

Proof By theorem 1 $\|x^k - x^*\|^2 \to 0$, therefore $|x_i^k - x^*| \to 0$. Consider $i \in L$.

$$x_i^{k+1} = S_{\tau_k \gamma}\left(x_i^k - \tau_k \nabla f(x^k) + \beta_k (x_i^k - x_i^{k-1})\right) \tag{2.31}$$

Since $x_i^k \to 0$, for all $\epsilon > 0$ there exists a K_1 such that $|x_i^k| < \epsilon$ for all $k > K_1$. Take $\epsilon < \gamma \tau_{\min}$ implies $|x_i^k| = 0$ for all $k > K_1$, which proves statement (2.27).

For the second statement we note that for $i \in \text{supp}(x^*)$,

$$0 \neq x_i^* = \text{sgn}\left(\bar{h}_i(x^*)\right) \max\{|\bar{h}_i(x^*)| - \nu, 0\},$$

where $\nu = \gamma \tau_*$. Therefore $|\bar{h}_i(x^*)| > \nu$ for all $i \in \text{supp}(x^*)$. Let

$$e_i^k \triangleq |x_i^k - x_i^*|.$$

Since $\|x^k - x^*\| \to 0$ there exists some M such that $\|x^k - x^*\| < M$ for all k.

If

$$\operatorname{sgn}(h_i^k(x^k, x^{k-1})) \neq \operatorname{sgn}(\bar{h}_i(x^*)) \tag{2.32}$$

for some $i \in \operatorname{supp}(x^*)$, then Lemma 4 implies

$$\begin{aligned}
(e_i^{k+1})^2 &= \left|S_\nu \circ h_i^k(x^k, x^{k-1}) - S_\nu \circ h_i(x^*)\right|^2 \\
&\leq \left(|h_i^k(x^k, x^{k-1}) - h_i(x^*)| - \nu\right)^2 && (2.33) \\
&\leq \left|h_i^k(x^k, x^{k-1}) - h_i(x^*)\right|^2 - \nu^2 && (2.34) \\
&= |\bar{h}_i^k(x^k) - \bar{h}(x^*) + \beta_k(x_i^k - x_i^{k-1})|^2 \\
&\quad -\nu^2 && (2.35) \\
&\leq |\bar{h}_i^k(x^k) - \bar{h}(x^*)| + \beta_k^2|x_i^k - x_i^{k-1}|^2 \\
&\quad +2\beta_k|x_i^k - x_i^{k-1}||\bar{h}_i^k(x^k) - \bar{h}(x^*)| \\
&\quad -\nu^2. && (2.36)
\end{aligned}$$

Which implies

$$\begin{aligned}
\|x^{k+1} - x^*\|^2 &\leq \|x^k - x^*\|^2 + \beta_k^2\|x^k - x^{k-1}\|^2 \\
&\quad +2\beta_k\|x^k - x^{k-1}\|\|x^k - x^*\| - \nu^2 && (2.37) \\
&\leq \|x^k - x^*\|^2 + \beta_k^2\|x^k - x^{k-1}\|^2 \\
&\quad +2\beta_k\|x^k - x^{k-1}\|M - \nu^2. && (2.38)
\end{aligned}$$

Now $\beta_k^2\|x^k - x^{k-1}\|^2 = O(1/k)$ by condition (2.13). Therefore $\beta_k\|x^k - x^{k-1}\| = O(1/\sqrt{k})$ and

$$\sum_{j=1}^{k} \beta_j\|x^j - x^{j-1}\| = O(\sqrt{k}). \tag{2.39}$$

Using condition (2.13) there exists $M_2 > 0$ such that $\sum_{k=1}^{\infty} \beta_k\|x^k - x^{k-1}\|^2 < M_2$. An explicit value for M_2 was found in [10]. Thus

$$\|x^{k+1} - x^*\|^2 \leq \|x^0 - x^*\|^2 + M_2 - k\nu^2 + O(\sqrt{k}). \tag{2.40}$$

Therefore for all $\epsilon > 0$ there exists a K_2 such that for all $k > K_2$

$$(e_i^{k+1})^2 \leq (e_i^0)^2 - k(\nu^2 - \epsilon). \tag{2.41}$$

Which implies that if equation (2.32) holds for some i and $k > K_2$

$$\|x^{k+1} - x^*\|^2 \leq \|x^0 - x^*\|^2 - k(\nu^2 - \epsilon). \tag{2.42}$$

Take $\epsilon < \nu^2$ and we see that the number of iterations such that $\text{sgn}(h_i^k(x^k, x^{k-1})) \neq \text{sgn}(\bar{h}_i(x^*))$ for some i is at most

$$\max\left\{\frac{\|x^0 - x^*\|^2 + M_2}{\nu^2 - \epsilon}, K_2(\epsilon)\right\} \tag{2.43}$$

steps, for some $\epsilon > 0$.

Let

$$\omega = \min\{\nu - |h_i(x^*)| : i \notin \text{supp}(x^*)\} > 0. \tag{2.44}$$

Consider $i \notin \text{supp}(x^*)$. The second part of lemma 4 implies

$$\begin{aligned}(e_i^{k+1})^2 &= \left|S_\nu \circ h_i^k(x^k, x^{k-1}) - S_\nu \circ h_i(x^*)\right|^2 \\ &\leq \left(|h_i^k(x^k, x^{k-1}) - h_i(x^*)| - (\nu - h_i(x^*))\right)^2 \\ &\leq \left|h_i^k(x^k, x^{k-1}) - h_i(x^*)\right|^2 - \omega^2. \end{aligned} \tag{2.45}$$

Repeating the arguments used to prove statement (2.43) we can say the following. For all $\epsilon > 0$ there exists some K_1' such that if $x_i^k \neq 0$ for some $i \notin \text{supp}(x^*)$ we must have

$$\max\left\{\frac{\|x^0 - x^*\|^2 + M_2}{\omega^2 - \epsilon}, K_1'(\epsilon)\right\}. \tag{2.46}$$

We will now use theorem 5 to show that after finitely many iterations the output of algorithm 5 becomes equivalent to projected heavy ball subject to a specific quadrant constraint. Let $E \triangleq \text{supp}(x^*)$ and $L = E^c$. Let x_E be the $|E|$-dimensional vector with entries from the subset of x corresponding to E. Let $(x_E, \mathbf{0})$ mean the n-dimensional vector with values on E equal to x_E and zero on L.

Corollary 6 *Assume condition (2.13) holds. After finitely many iterations the iterations of algorithm 5 reduce to projected heavy ball for minimizing*

$\phi(x_E)$ *over constraint set* O_E *where*

$$\begin{aligned} \phi(x_E) &\triangleq -(g_E^*)^T x_E + f(x_E) && (2.47) \\ O_E &\triangleq \{x_E \in \mathbb{R}^{|E|} : -sgn(g_i^*)x_i \geq 0,\ \forall i \in E\}. && (2.48) \end{aligned}$$

Specifically $x^{k+1} = (x_E^{k+1}, \mathbf{0})$ *where*

$$x_E^{k+1} = P_{O_E}\left(x_E^k - \tau_k \nabla\phi(x_E^k) + \beta_k(x_E^k - x_E^{k-1})\right), \tag{2.49}$$

and P_{O_E} *is the orthogonal projector onto* O_E.

Proof From theorem 5 there exists a K such that for all $k > K$ equations (2.27) and (2.28) hold. Take $k > K$. Since $x_i^k = 0$ for all $i \in L$ it suffices to consider $i \in E$. For $i \in E$ we have $x_i^k \geq 0$ if $\text{sgn}(h_i^k(x^{k-1}, x^{k-2})) = 1$ (equivalently $g_i^* < 0$) and $x_i^k \leq 0$ if $\text{sgn}(h_i^k(x^{k-1}, x^{k-2})) = -1$ (equivalently $g_i^* > 0$). Therefore for any $i \in E : -g_i^* x_i^k \geq 0$ thus $x_E^k \in O_E$ for all $k > K$.

For $i \in E$ we calculate the quantity

$$\begin{aligned} y_i^{k+1} &\triangleq x_i^k - \tau_k \nabla\phi(x^k)_i + \beta(x_i^k - x_i^{k-1}) && (2.50) \\ &= x_i^k - \tau_k(-g_i^* + g_i(x^k)) + \beta_k(x_i^k - x_i^{k-1}) && (2.51) \\ &= h_i^k(x^k, x^{k-1}) - \gamma\tau_k(\frac{g_i^*}{\gamma}) && (2.52) \\ &= \text{sgn}(h_i^k(x^k, x^{k-1}))(|h_i(x^k)| - \gamma\tau_k). && (2.53) \end{aligned}$$

Therefore

$$x_i^{k+1} = S_{\gamma\tau_k}\left(h_i^k(x^k, x^{k-1})\right) = \begin{cases} y_i^{k+1} : -g_i^* y_i^{k+1} \geq 0 \\ 0 : \text{else} \end{cases} \tag{2.54}$$

equivalently

$$x_E^{k+1} = P_{O_E}(x_E^k - \tau\phi(x_E^k) + \beta_k(x_E^k - x_E^{k-1})). \tag{2.55}$$

If assumption (2.13) holds the sequence x^k generated by algorithm 5 converges. Thus there exists $K_3 > 0$ such that for all $k > K_3$ $x_E^k \in O_E$ and equation (2.49) is satisfied. Therefore the rate of convergence of the constrained heavy ball iterations (2.49) is equivalent to the rate of unconstrained heavy ball for minimizing $\phi(x_E)$.

Thus we see that the rate of convergence of algorithm 5 is ultimately determined by the sub-matrix of the Hessian of f corresponding to the support set E of the limit $x^* \in X^*$. This is also the Hessian of ϕ_E. In [6] the authors proved a similar result for IST which is a special case of algorithm 5 with $\beta_k = 0$, i.e. no momentum.

2.7 Optimal Choice of Parameters

We will now discuss the choice of the parameters β_k and τ_k. Let H be the Hessian of f, i.e. $H = A^T A$ and let H_E be the Hessian of ϕ_E, i.e H_E. Let $p = |E|$. Let $\lambda_1, \lambda_2, \ldots, \lambda_p$ be the eigenvalues of $H_E = A_E^T A_E$. Let $\lambda_{\min}^E = \min\{\lambda_i\}$ and $\lambda_{\max}^E = \max\{\lambda_i\}$ and assume $\lambda_{\min}^E > 0$, which must be true in compressed sensing if the recovery problem is feasible. Note that $\lambda_{\max}^E$ is typically much smaller than $\lambda_{\max} \triangleq \lambda_{\max}(A^T A)$. We will say that a sequence a^k has q-linear rate of convergence if

$$q = \limsup_{k \to \infty} \frac{\|a^{k+1} - a^*\|}{\|a^k - a^*\|}. \in (0, 1).$$

In [6] the authors showed that the rate of convergence of IST was q-linear with q equal to

$$q = \frac{\lambda_{\max} - \lambda_{\min}^E}{\lambda_{\max} + \lambda_{\min}^E} \tag{2.56}$$

corresponding to the choice $\tau_k = 2/\lambda_{\max}$ and $\beta_k = 0$ for all k. We will now attempt to choose the sequence β_k to improve q.

In [4] Polyak derived the optimal choice of $\tau_k = \tau^*$ and $\beta_k = \beta^*$ for the constant parameter case. The conjugate gradient algorithm arises when τ_k and β_k are chosen by a line search at each iteration. In choosing τ_k and β_k we must be careful to ensure that Moudafi and Oliny's conditions on the sequences are satisfied, namely $0 < \tau_{\min} \leq \tau_k \leq 2\gamma$ and $\sum \beta_k \|x^k - x^{k-1}\|^2 < \infty$, otherwise the I-IST algorithm is not guaranteed to converge.

To keep things simple we will fix τ_k and β_k to constant values for all

iterations. It can be shown that for $k > K_3$ the iterations (2.49) satisfy

$$\begin{bmatrix} x_E^{k+1} - x_E^* \\ x_E^k - x_E^* \end{bmatrix} = M \begin{bmatrix} x_E^k - x_E^* \\ x_E^{k-1} - x_E^* \end{bmatrix} \tag{2.57}$$

where

$$M = \begin{bmatrix} (1+\beta)I_{p\times p} - \tau A_E^T A_E & -\beta I_{p\times p} \\ I_{p\times p} & \mathbf{0}_{p\times p} \end{bmatrix}. \tag{2.58}$$

Therefore for $k > K_3$

$$\|x^k - x^*\| = \|x_E^k - x_E^*\| \leq C(q+\epsilon_k)^k \tag{2.59}$$

where q is the maximum magnitude of all eigenvalues of M and $\epsilon_k \to 0$ [4]. Let the eigenvalues of M be ρ_j. The problem we are trying to solve is

$$\min_{\beta,\tau} |\rho_{\max}(M)| \text{ subject to condition (2.13)}, \tau \leq 2\gamma. \tag{2.60}$$

From [10] we know that condition (2.13) can be enforced by making $\beta < 1/3$ so a simpler problem would be

$$\min_{\beta,\tau} |\lambda_{\max}(M)| \text{ subject to } \beta < 1/3, \tau \leq 2\gamma. \tag{2.61}$$

The eigenvalues $\rho_j, j = 1, 2, \ldots, 2n$ of M are equal to the eigenvalues of the 2×2 matrix

$$\begin{bmatrix} 1+\beta-\tau\lambda_i & -\beta \\ 1 & 0 \end{bmatrix}. \tag{2.62}$$

for $i = 1, 2, \ldots, n$. Therefore they are the $2n$ roots of the equations

$$\rho^2 - \rho(1+\beta-\tau\lambda_i) + \beta = 0, \quad i = 1, 2, \ldots, n. \tag{2.63}$$

Therefore

$$\rho_i = \frac{(1+\beta-\lambda_i\tau) \pm \sqrt{(1+\beta-\tau\lambda_i)^2 - 4\beta}}{2}. \tag{2.64}$$

If

$$(1+\beta-\tau\lambda_i)^2-4\beta<0 \quad \forall i \tag{2.65}$$

then $|\rho_i|=\sqrt{\beta}$ for all i. Simple algebra shows that (2.65) requires

$$\frac{(1-\sqrt{\beta})^2}{\lambda_i}\leq\tau\leq\frac{(1+\sqrt{\beta})^2}{\lambda_i} \quad \forall i \tag{2.66}$$

which in turn requires

$$\frac{(1-\sqrt{\beta})^2}{\lambda^E_{\min}}\leq\frac{(1+\sqrt{\beta})^2}{\lambda^E_{\max}}. \tag{2.67}$$

Let $\kappa_E=\lambda^E_{\max}/\lambda^E_{\min}$. Condition (2.67) implies

$$\beta\geq\left(\frac{\sqrt{\kappa_E}-1}{\sqrt{\kappa_E}+1}\right)^2 \tag{2.68}$$

which implies the optimal choice β^* as making an equality in (2.68) and τ^* as the left or right hand side of the inequality sandwich (2.67), since they are equal. However we also require $\tau^*\leq 2/\lambda_{\max}$ so we must enforce

$$\frac{(1-\sqrt{\beta})^2}{\lambda^E_{\min}}\leq\frac{2}{\lambda_{\max}} \tag{2.69}$$

which implies

$$\beta\geq\left(1-\sqrt{\frac{2}{\kappa'}}\right)^2 \tag{2.70}$$

where $\kappa'=\lambda_{\max}/\lambda^E_{\min}$. Therefore the optimal choice is

$$\beta^*=\max\left\{\left(\frac{\sqrt{\kappa_E}-1}{\sqrt{\kappa_E}+1}\right)^2,\left(1-\sqrt{\frac{2}{\kappa'}}\right)^2\right\} \tag{2.71}$$

and

$$\tau^*=2/\lambda_{\max}. \tag{2.72}$$

If $\beta^*\geq 1/3$ it is not possible to enforce (2.65) and satisfy Moudafi and

Oliny's sufficient condition for convergence. Therefore for any choice $\beta < 1/3$ the roots will be of varying magnitude and some will be real. Let ρ_i^+ and ρ_i^- be the roots corresponding to + and − respectively in equation (2.64). It can be seen that for any i if the two roots are real, $|\rho_i^+| > |\rho_i^-|$, otherwise $|\rho_i^+| = |\rho_i^-| = \sqrt{\beta}$. ρ_i^+ is monotone decreasing in λ_i. Therefore to establish the rate of convergence it suffices to consider ρ_1^+. $|\rho_1^+|$ is monotone decreasing in β for fixed τ and for $\beta \leq \beta^*$ defined in equation (2.68). Furthermore it is monotone decreasing in τ for fixed β. If β^* is greater than 1/3, the optimal choice while satisfying Moudafi and Oliny's constraints is $\beta = 1/3 - \epsilon$ for any small $\epsilon > 0$ and $\tau = 2/L$.

If $\beta^* < 1/3$ the rate of convergence is $\sqrt{\beta}$. Otherwise the rate is $|\rho_1^+|$ corresponding to the choice $\beta = 1/3 - \epsilon$ and $\tau = 2/L$.

Let us compare the rate of convergence of IST and our proposal I-IST. If $\beta^* < 1/3$ than I-IST attains q-linear convergence with rate

$$q_1 = \max\left\{\left(\frac{\sqrt{\kappa_E} - 1}{\sqrt{\kappa_E} + 1}\right), \left(1 - \sqrt{\frac{2}{\kappa'}}\right)\right\} \tag{2.73}$$

while IST achieves

$$q_2 = \frac{\kappa' - 1}{\kappa' + 1}. \tag{2.74}$$

Now since $\kappa' \gg \kappa_E$ q_1 is typically equal to the second argument in equation (2.73). Figure 2.4 compares q_1 and q_2. The improvement is more marked when κ' is large which is in keeping with the behavior of the heavy ball method established by Polyak [4].

IST corresponds to the choice $\beta = 0$ and $\tau = 2/L$. Since $|\rho_i^+|$ is decreasing in β for $\beta < \beta^*$, any positive choice of β will improve the rate of convergence. It should be noted that we confine $\beta < 1/3$ so that Moulani and Ofiny's sufficient condition for convergence is satisfied. However it is not clear if this is also a necessary condition and numerical experiments suggest larger values of β, such as $\beta = \beta^*$, work well and provide fast convergence. We suggest using the optimal choice of β given in equation 2.71 at first. If it appears the algorithm is diverging then one can reduce β until the algorithm is convergent, keeping in mind that if β is less than 1/3 convergence is guaranteed.

Finally we note that the values κ_E and κ' are not known apriori because they depend on the support of the solution x^*. In practice these quantities

or upper bounds for them must be estimated. This should not prove too difficult since ultimately one would have a desired sparsity level p in mind and the eigenvalues $\lambda^E_{\min}$ and $\lambda^E_{\max}$ can be estimated from the measurement matrix. This is also true of $\lambda_{\max}$ which is rarely computed exactly but is usually estimated. One simple approach would be to sample some number of sub-matrices of the size of the desired sparsity level and exactly compute the eigenvalues of these matrices and then average them together to estimate κ_E. Alternatively β could be adjusted on the fly as the support of the current iterate changes, based on the eigenvalues of the corresponding sub-matrix. Such modifications should yield improvements to the rate of convergence.

It is also worth mentioning that since the function ϕ has Lipschitz continuous gradient, for all x and y

$$\phi(x) \leq \phi(y) + \langle x - y, \nabla\phi(y)\rangle + \frac{L}{2}\|x - y\|^2. \tag{2.75}$$

Choosing $x = x^k$ and $y = x^*$ means that linear convergence of x^k implies linear convergence of $\phi(x^k_E)$ and therefore $F(x^k)$.

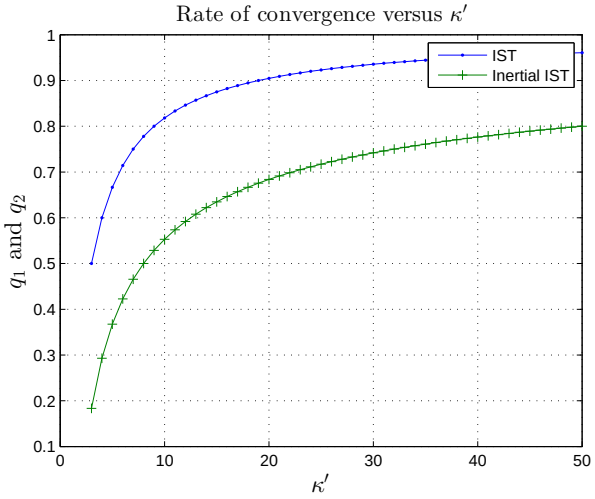

Figure 2.4: Comparison of q_1 and q_2 in Equations (2.74) and (2.73) for Varying κ'

2.8 LASSO Simulations

2.8.1 Parallel Coordinate Descent

We compare our proposed algorithm for solving the LASSO problem with several algorithms. Of particular interest are several algorithm variants described in [15] which are similar to our proposed algorithm, I-IST, but with important differences. There are three main variants proposed in [15], Parallel Coordinate Descent (PCD) and two modifications called PCD-SESOP and PCD-CG. SESOP stands for Sequential Subspace Optimization and CG stands for conjugate gradient. We will now describe each of these three variants.

The main idea of [15] is a parallel coordinate descent approach to solving the LASSO problem. Each coordinate is updated via a line search. However cycling through all coordinates is computationally expensive, so they are updated in parallel which gives rise to an iterative algorithm quite similar to IST but with several differences. Most importantly, PCD solves a smoothed version of the LASSO problem. The absolute value term in the ℓ_1 norm is replaced by

$$\phi_\epsilon(a) = |a| - \epsilon \log\left(1 + \frac{|a|}{\epsilon}\right). \tag{2.76}$$

As $\epsilon \to 0$, $\phi_\epsilon(a) \to |a|$. Note that the curvature can be quite large at the origin since $\phi_\epsilon''(0) = 1/\epsilon$. The smoothed LASSO problem solved by PCD is

$$\min_x \frac{1}{2}\|y - Ax\|^2 + \gamma \sum_{i=1}^{N} \phi_\epsilon(x_i). \tag{2.77}$$

Since the problem is smooth, the gradient and Hessian can be computed. The second difference is that, while IST uses a scalar step-size $\tau \leq 2/\lambda_{\max}$, PCD uses a diagonal matrix of different step-sizes for each component, which arises from the component-wise nature of the update. The thresholding operator which arises in PCD from solving problem (2.77) is not the same as the ℓ_1 proximity operator. Furthermore PCD employs line search techniques which are not easy to emulate with IST due to the non-smoothness of the ℓ_1 term.

The second variant involves SESOP and is relevant to I-IST. SESOP, se-

quential subspace optimization, refers to the fact that at each iteration the algorithm minimizes the smoothed function of (2.77) along a subspace. This subspace is the span of the current gradient plus M previous search directions, i.e.

$$d^{k-i+1} = x^{k-i+1} - x^{k-i}$$

for $i = 1, \ldots, M$. Since the objective function is smooth, one can perform a search within the subspace spanned by the current gradient and M previous directions using Newton's method or some approximation to it. So long as M is small, the complexity of this search is low.

The final variant, PCD-CG is similar to PCD-SESOP, except only one previous direction is used. The next direction is computed explicitly using the Polyak-Ribiere formula [16], as $d^{k+1} = \beta d^k + g^{k+1}$, with a specific formula for β which is a direct generalization of the conjugate gradient method to non-quadratic problems. A line search is then performed along d^{k+1} to generate the next iterate.

It is worth pointing out the differences between PCD-SESOP, PCD-CG and our proposed method, I-IST. Firstly, they solve different problems since PCD-SESOP and PCD-CG solve a smoothed approximation to LASSO. In order to meet the required accuracy of any given application, ϵ may have to be very small (we used 10^{-10} in our experiments) and reducing ϵ appears to slow down the convergence. However for most desired accuracies our experiments suggest this is not a major drawback of these methods.

Secondly, while all three methods employ previous search directions, the PCD variants do so with in the parallel coordinate descent method, which results in a different algorithm.

Thirdly, PCD-SESOP and PCD-CG require more computation per iteration than I-IST. We found that the running times of PCD-SESOP and PCD-CG were larger than I-IST (see tables 2.1, 2.2 and 2.3). However the per iteration complexity of all three algorithms is the of the same order, as they all involve two matrix multiplies per iteration at cost $O(mN)$. This is the dominant factor so long as $M \ll m$. For large m and N these matrix multiplies dominate, however in many applications the matrix multiplies can be performed with a fast transform like the FFT. Numerical experiments suggest that computing the function $\phi_\epsilon(x)$ can be slow, especially since it must be computed often in the line or subspace search performed in each iteration

of all three variants of PCD, because a backtracking technique is used. I-IST is also far simpler, involving fixed parameters and no line search techniques. It may be possible to improve I-IST by considering variable choices of the parameters, such as in the conjugate-gradient method applied to a quadratic problem. Such possibilities are for future work. As we show in the numerical experiment below, even with a simple fixed choice of the parameters, I-IST can achieve comparable performance to the more complicated PCD-SESOP and PCD-CG and in less total computation time.

Finally, to the best of our knowledge, all three PCD variants do not have theoretical guarantees for the LASSO problem. They do have a convergence guarantee for problem (2.77), but with no rate of convergence. The rate of convergence PCD and variants are known only for strongly convex quadratic functions. For the function given in (2.77) the rate of convergence is effected by the curvature of the function ϕ_ϵ which grows as $1/\epsilon$ at 0. This means for small ϵ the rate of convergence given in [15] could be very slow. On the other hand the analysis of I-IST given in this thesis holds for the LASSO problem and gives a precise value for the asymptotic linear convergence rate.

2.8.2 Gaussian Sensing Matrix

We ran the following simulation to compare I-IST with IST, FISTA, PCD and PCD-SESOP. In this problem A is a 500×4000 Gaussian matrix with entries distributed i.i.d. $\mathcal{N}(0, 1/\sqrt{m})$. We generate a length 4000 sparse vector x with 200 non-zero components which are drawn i.i.d. $\mathcal{N}(0, 1)$. We compute a noisy measurement $y = Ax + e$ where e has $\mathcal{N}(0, 0.0025)$ entries. We attempt to estimate x by solving problem (2.19) using I-IST, IST and FISTA and problem (2.77) using PCD, PCD-SESOP and PCD-CG. We set $\gamma = 0.05$ which was manually chosen to produce the true sparsity level. We compute a solution using the interior point method of [17] and use this as the optimal value of the objective function, f^*, at a tolerance level of 10^{-8}. That is, the solution of the interior point method is within 10^{-8} of the true minimum. Therefore 10^{-8} is the "noise-floor" observed in Figure 2.5. We stop each algorithm when the objective function is within 10^{-8} of the output of the interior point method.

We use IST with step size $\tau_L = 2/\lambda_{\max}(A^T A)$, FISTA with the recom-

mended settings [7], and PCD-SESOP with a search subspace of the past two previous directions. For I-IST we computed the optimal β and τ using equation (2.71) using an estimate of κ_E based on the sparsity level that was a simple average of 100 sampled sub-matrices. The optimal β is 0.6856 and $\tau = \tau_L$. We note that this choice of β does not satisfy the sufficient condition for convergence given by Moudafi and Oliny. As can be seen in figure 2.5, PCD-SESOP has the fastest convergence followed by I-IST. Both methods significantly outperform IST and FISTA. The linear convergence rate of I-IST is corresponds with that predicted by the theory of the previous section (equation 2.74).

While PCD-SESOP converges in fewer iterations than I-IST, we note that the total computation time of PCD-SESOP is higher, as shown in Table 2.1. Note that Table 2.1 refers to the time taken for each method to reach an objective function value within 10^{-8} of the objective function of the interior point method, as well as the number of iterations. We do not include the iteration count for the interior point method because it is not comparable, since the per-iteration complexity of this method is much higher than the other methods. The interior point is essentially a second-order method while the others are first-order. Another thing to note is that Inertial IST is not a descent method whereas the three variants of PCD are descent methods. As can be seen in figure 2.5 the early iterations of Inertial IST actually increase the objective function.

Table 2.1: Total Running Time

Mthd:	Int Pnt	IST	FISTA	I-IST	PCD	PCD-SESOP	PCD-CG
time (s):	3.4	6.47	3.94	0.57	3.56	3.46	2.4
iterations:	NA	1771	1038	147	436	133	130

2.8.3 DCT Sensing Matrix

In this simulation we randomly select 512 rows of the 4096×4096 discrete cosine transform (DCT) matrix to use as our sensing matrix. We consider two sparsity levels.

1. 250-sparse vector and $\gamma = 0.08$.

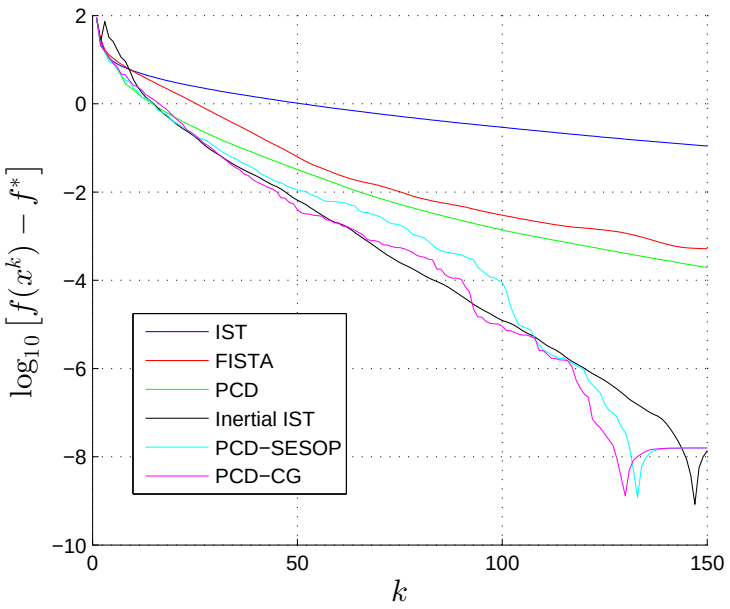

Figure 2.5: Objective Functions for LASSO

2. 25-sparse vector and $\gamma = 0.06$.

The ℓ_1 penalty constant γ was chosen so that the minimizer had the true sparsity k. The results are shown in figures 2.6 and 2.7. As before, f^* is computed to within 10^{-8} using the interior point method.

For the case of $k = 250$, I-IST, PCD-SESP and PCD-CG are the best performing algorithms. This is not surprising since, as k is large, the conditioning of the sub-matrix corresponding to the optimal support set is poor. Thus I-IST offers a significant speed-up over IST. In fact the optimal β is 0.68 and the predicted asymptotic rate of convergence of I-IST is thus

$$f(x^k) - f^* \leq (0.68)^k.$$

This rate is confirmed by looking at the slopes in Figure 2.6. On the other hand the asymptotic rate of convergence of IST is

$$f(x^k) - f^* \leq (0.94)^k,$$

thus I-IST offers a significant speed-up. PCD-SESOP and PCD-CG are also implicitly using momentum therefore it is not uprising that they also perform well when the support sub-matrix is ill-conditioned, which is the case when

k is large (γ is small).

In the second case with $k = 25$ the improvement provided by I-IST is not as significant, as the Hessian corresponding to the optimal support is not as poorly conditioned. The predicted asymptotic rate for IST is

$$f(x^k) - f^* \leq (0.64)^k,$$

while for I-IST

$$f(x^k) - f^* \leq (0.41)^k.$$

The three PCD variants perform equally well in this case and outperform I-IST. This results suggests that I-IST is mostly effective for large k but for small k the PCD methods appear to be more effective. We note that the case where the k is large is often when acceleration is most needed as most methods can be quite slow. However for small k there are many fast options, such as LAR [18], which converges in exactly k iterations. Tables 2.2 and 2.3 again show that I-IST converges more quickly than the PCD variants in terms of overall time.

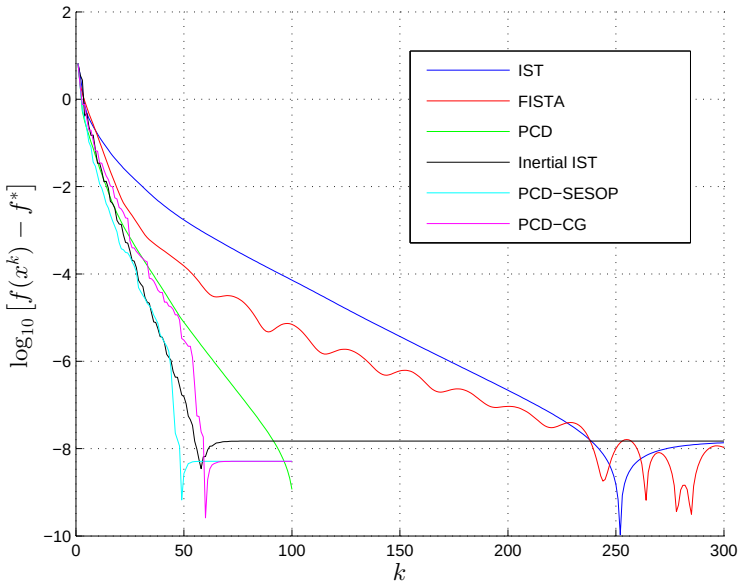

Figure 2.6: Objective Functions for LASSO: DCT Matrix, Sparsity Level $k = 250$

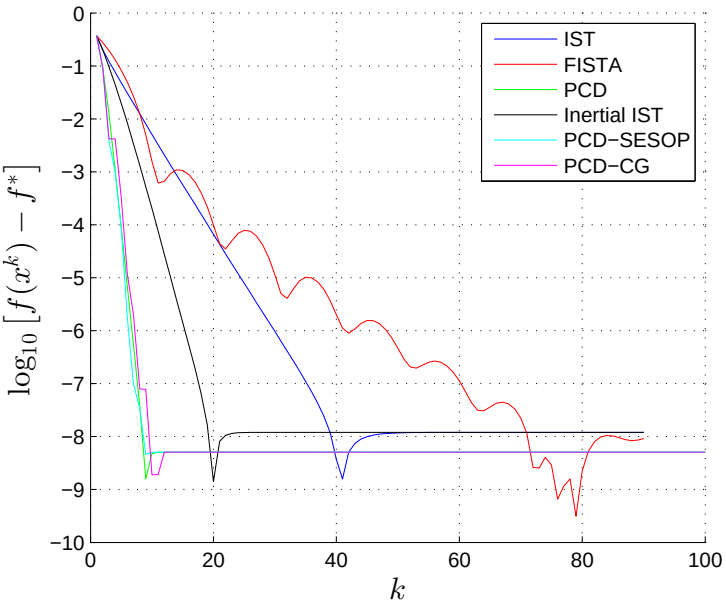

Figure 2.7: Objective Functions for LASSO: DCT Matrix, Sparsity $k = 25$

Table 2.2: Total Running Time: $k = 250$.

Mthd:	Int Pnt	IST	FISTA	I-IST	PCD	PCD-SESOP	PCD-CG
time (s):	3.67	0.96	1	0.25	1.6	0.93	1.2
iterations:	NA	240	240	60	48	60	90

Table 2.3: Total Running Time: $k = 25$.

Mthd:	Int Pnt	IST	FISTA	I-IST	PCD	PCD-SESOP	PCD-CG
time (s):	1.4	0.16	0.28	0.09	0.14	0.38	0.19
iterations:	NA	40	70	20	10	10	10

CHAPTER 3

NON-CONVEX METHODS WITH INERTIA

3.1 Inertial Iterative Hard Thresholding

In the previous sections we introduced an algorithm for solving the LASSO problem, problem (2.19). In compressed sensing the LASSO problem is used as a tractable convex surrogate for the intractable non-convex problem:

$$\min_x \gamma\|x\|_0 + \frac{1}{2}\|y - Ax\|^2 \tag{3.1}$$

where $\|x\|_0$ counts the number of non-zero elements of x. An equivalent problem is

$$\min_x \frac{1}{2}\|y - Ax\|^2 : \quad \|x\|_0 \leq s. \tag{3.2}$$

Problems (3.1) and (3.2) are equivalent in the sense that for every γ there exists some s such that the two problems have the same minimum and minimizer. In [19] Blumensath et al. analyzed algorithms for solving each of these problems as well as conditions on A such that the minimizer is a good estimate in a compressed sensing framework. Similar methods were also introduced in [20] in the context of image denoising and deblurring. For problem (3.1) they introduced and analyzed the so called iterative hard thresholding algorithm (IHT) which utilizes the hard thresholding operator $H_c(x)$ defined as follows:

$$\{H_c(x)\}_i = \begin{cases} x_i : & |x_i| \geq c \\ 0 : & |x_i| < c \end{cases} \tag{3.3}$$

IHT is described in Algorithm 6 corresponding to the choice $\beta_k = 0$ and $\tau_k = 1/\lambda_{\max}(A^T A)$. To solve problem (3.1) one chooses $c_k = \gamma^{1/2}$ and to solve

problem (3.2) one chooses c_k to be the s-th largest component in magnitude of g_k. For each problem and with the specified choice of c_k Blumensath et al. showed IHT converges to a local minimum asymptotically linearly. They also proved that, under some conditions on the RIP constant of A IHT recovers a sufficiently sparse vector from $Ax + e$ where e is bounded noise.

Algorithm 6 Inertial IHT (I-IHT)

Require: $\tau_{\min}$, sequences β_k and τ_k satisfying $0 \leq \beta_k < 1$, $0 < \tau_{\min} \leq \tau_k \leq 2/\lambda_{\max}(A^T A)$ and a stopping criterion.

1: Fix x_0.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad g_k = x^k - \tau_k A^T(Ax^k - y)) + \beta_k(x^k - x^{k-1})$
4: $\quad$ Choose c_k.
5: $\quad x^{k+1} = H_{c_k}(g_k)$
6: **end for**
7: **return** x^{k+1}

We propose using inertia to accelerate the convergence of Blumensath's IHT algorithm. Our proposed method is called Inertial IHT (I-IHT) and is described in Algorithm 6. Since after a finite number of iterations IHT converges on a support set, the linear convergence rate is then determined by the sub-matrix of the support and if this is ill-conditioned, convergence could be slow. The momentum can be chosen to counter this. Figure 3.1 shows numerical evidence support this possibility. In this simulation A was 100×200 with entries drawn i.i.d. from $N(0, 1/10)$. A 20-sparse vector x^* was drawn with random support and entries drawn i.i.d. from $N(0, 1)$ and the vector of noiseless measurements was formed as $y = Ax^*$. We ran IHT to solve problem (3.2) and I-IHT with $\beta = 0.3$. Future work would involve proving a theorem analogous to theorem 5 for I-IHT.

3.1.1 Proportional-Integral (PI) Feedback

Another potential advantage of momentum in non-convex problems is in helping to avoid local minima. Problem (3.2) has an overwhelming number of local minima and it is quite typical for iterative algorithms like IHT to become trapped in one of them. It was noticed in the neural networks community that including a momentum term in iterative algorithms for nonconvex problems

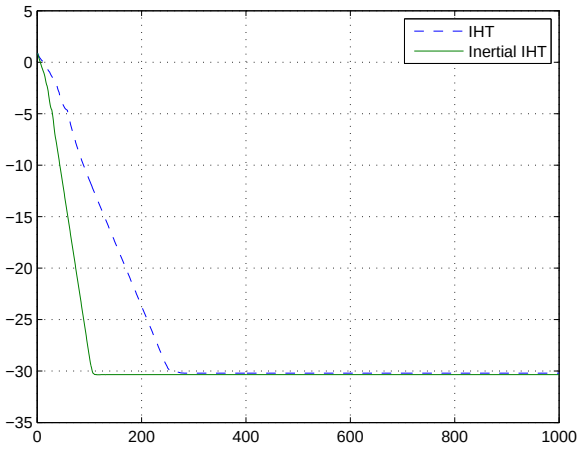

Figure 3.1: Numerical Simulation of Error Convergence of IHT and I-IHT

can help avoid shallow local minima [21].

An equivalent formulation of the unconstrained heavy ball algorithm (equation 2.11) is

$$p^{k+1} = \beta_k p^k - \tau_k \nabla f(x^k) \tag{3.4}$$

$$x^{k+1} = x^k + p^{k+1}. \tag{3.5}$$

With this formulation it can be seen that the update direction p^{k+1} is a weighted sum of all past anti-gradients. For fixed $\beta_k = \beta$,

$$p^{k+1} = \sum_{i=1}^{k} \beta^{k-i} \left(-\tau_k \nabla f(x^k) \right). \tag{3.6}$$

This also reveals a connection between the heavy ball method and the proportional integral (PI) controller [21]. If we think of $\nabla f(x) = 0$ as being an error of our system which we would like to drive to zero, the sum over past gradients is a discretized weighted integral of the error function.

An intuitive explanation why PI control can help avoid local minima is as follows. If β large the heavy ball iterates oscillate around a local minimum. They tend to "overshoot" the minimum. For convex problems where any local minimum is also a global minimum, this is an undesirable scenario as it slows the convergence rate. However for non-convex problems this underdamped scenario can be good if it helps to overcome the local maxima which surround

the local minimum. Of course it is possible that such oscillations might cause the iterates to escape the global minimum, so one must hope that the global minimum is "deeper" than competing local minima.

There may also be a connection between this idea and graduated non-convexity (GNC) [22]. GNC is an iterative non-convex solver whereby a non-convex function f is approximated by a sequence of simpler functions f_k which converge to f. At each iteration GNC minimizes the simpler function f_k using the previous minimizer x^{k-1} as the initialization. By minimizing a sequence of smoother functions it is hoped that GNC can avoid local minima and converge on the global minimum. In early iterations the approximators f_k capture only global properties of f. The PI method, by computing a smoothed gradient, would appear to be doing something similar in that minor fluctuations in the gradient as caused by shallow local minima are outweighed by large accumulations over many iterations caused by the global behavior of the gradient.

We propose a PI approach to solving problems (3.1) and (3.2) which is described in Algorithm 7. To appreciate the role of the momentum term, it is worth investigating the optimality properties of problem (3.2).

Lemma 7 ([19]) *Let $x^* \in \mathbb{R}^N$ and let $L = \{i : x_i^* = 0\}$ and $E = \{x_i^* \neq 0\}$. x^* is a local minimum of problem 3.2 if and only if*

$$|A_i^T(y - Ax^*)| \begin{cases} = 0 & i \in E \\ \leq c(x^*) & i \in L \end{cases} \tag{3.7}$$

where $c(x^*)$ is the s-th largest magnitude of x^*. Furthermore x^* is a fixed point of algorithm 7.

From Lemma 7 we see that a local minimum of problem (3.2) has gradient equal to zero on the non-zero entries but not necessarily equal to zero at other entries. For the noiseless case, if x^* is a true global minimum it must satisfy $y = Ax^*$ and therefore the gradient must be zero on L as well as E. However if x^* is not a global minimum than it is likely that $y \neq Ax^*$ in which case the gradient will be non-zero but less than the threshold $c(x^*)$ on L. The advantage of PI is that because of the integral term these non-zero errors will accumulate over iterations when the algorithm is caught in a local minimum. This non-zero offset will be enough to force the support to change if the weighted sum (equation 3.6) becomes larger than the threshold $c(x^*)$.

Algorithm 7 PI - IHT

Require: $\tau_{\min}$, sequences β_k and τ_k satisfying $0 \leq \beta_k < 1$, $0 < \tau_{\min} \leq \tau_k \leq 2/\lambda_{\max}(A^T A)$ and a stopping criterion.

1: Fix x_0.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad p^{k+1} = \beta_k p^k - \tau_k A^T(Ax^k - y)$
4: $\quad$ Choose c_k.
5: $\quad x^{k+1} = H_{c_k}\left(x^k + p^{k+1}\right)$
6: **end for**
7: **return** x^{k+1}

I-IHT is different from PI-IHT. IHT employs momentum in the direction updates whereas I-IHT adds momentum to the iterates x^k. The two methods become equivalent only when $c_k = 0$, that is without the $\|\cdot\|_0$ term. Inertial IST does not share the property of PI discussed in the previous paragraph. At a local minimum with non-zero error terms on the inactive entries of x^*, there is no accumulated error. This would suggest that PI-IHT is more effective than I-IHT in avoiding local minima and this is borne out in simulations (see figure 3.2).

We compared the three algorithms on a compressed sensing problem with A of size 100×200 with entries drawn i.i.d. from $N(0, 1/10)$. A 40-sparse vector x^* was drawn with random support and entries drawn i.i.d from $N(0, 1)$ and the vector of noiseless measurements was formed as $y = Ax^*$. Note that this is a much harder problem than the simulation in the previous section because the sparsity of the vector is much larger, leading to many more local minima. We ran IHT to solve problem (3.2), I-IHT with $\beta = 0.9$ and PI-IHT with $\beta = 0.5$ and $\beta = 0.9$. We note that only PI-IHT with $\beta = 0.9$ does not get trapped in a local minimum.

3.2 Low Rank Matrix Recovery

The low-rank matrix recovery problem can also be tackled via non-convex methods. One is interested in solving the following problem:

$$\min_X \frac{1}{2}\|y - \mathcal{A}(X)\|^2 : \quad \text{rank}(X) \leq s \tag{3.8}$$

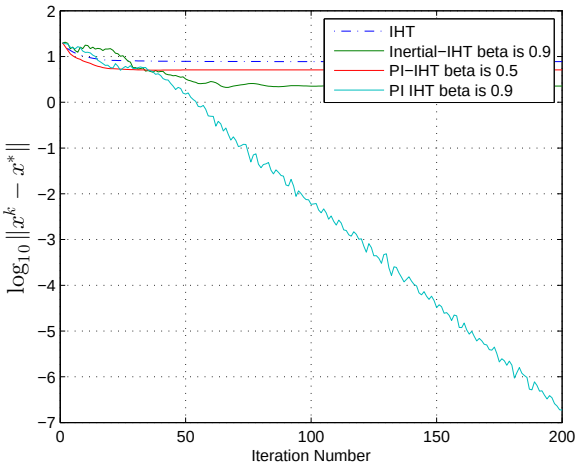

Figure 3.2: Comparison of IHT, I-IHT and PI-IHT

for which Problem (2.18) is a convex surrogate. Problem (3.8) is non-convex [12]. Analogously to the IHT algorithm, an iterative solver for problem (3.8) known as singular value projection (SVP) was developed in [23]. At each iteration SVP applies a hard thresholding operator to the singular values of a matrix. The operator $H_s(X)$ is then the best rank s approximation to X in the Frobenius sense. This motivates us to suggest an inertial version of SVP, which we elsewhere called Residual Feedback Iterative Thresholding (RFIT) [24].

In figure 3.3 we show the results of a simulation comparing RFIT and SVP. Two matrices U and V of size 20×2 and 2×20 respectively are generated with entries drawn i.i.d. $N(0,1)$ and a third matrix $X = UV$ is formed. Note that $\text{rank}(X) \leq 2$. An 80×400 matrix A is drawn with elements i.i.d. $N(0,1)$ and noiseless linear measurements are taken, $y = A\text{vec}(X)$, where $\text{vec}(X)$ is the 400×1 vector formed by stacking the columns of X on top of each other. We solve problem (3.8) with $s = 2$ using RFIT and SVP. Note that the number of degrees of freedom of a rank 2 20×20 matrix is $r(m + n - r) = 76$, so we have only slightly more measurements than are absolutely necessary. It is then not surprising that SVP falls into a local minimum. RFIT, however, converges to X at a linear rate. This suggests that the momentum helps RFIT to escape local minima in a way analogous to PI-IHT.

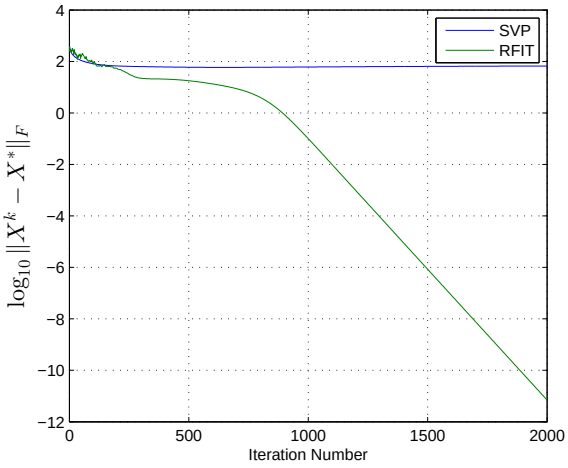

Figure 3.3: Comparison of SVP and RFIT

CHAPTER 4

CONCLUSIONS

In this thesis we proposed several algorithms for several non-smooth convex and non-convex optimization problems which are of great interest in many fields such as compressed sensing, low-rank matrix recovery and robust PCA. For the special case of ℓ_1-regularized least-squares (LASSO) we extended the analysis of Polyak and proved that our proposed algorithm achieves a linear convergence rate which is faster than the well known iterative soft thresholding algorithm, particularly on poorly conditioned problems. The improvement comes with minor additional computation. All of the introduced methods rely on the concept of inertia, which is shown to provide acceleration on convex problems and to help avoid local minima on non-convex problems. Future work may include:

1. Incorporating a continuation scheme on γ and β into the I-IST method.

2. Developing a variation of I-IST which uses adaptive choices of τ and β in the spirit of the conjugate-gradient method.

3. Establishing convergence rates for the non-convex algorithms proposed in chapter 3.

4. Exploring connections between these inertial non-convex methods and other non-convex methods such as graduated non-convexity which avoid local minima by exploring global properties of the objective function.

REFERENCES

[1] A. Press, "Number of active users at facebook over the years," 2012. [Online]. Available: http://finance.yahoo.com/news/number-active-users-facebook-over-years-214600186--finance.html

[2] M. Davenport, M. Duarte, Y. Eldar, and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge, UK: Cambridge University Press, 2012, ch. 1.

[3] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky, "The convex geometry of linear inverse problems," *Foundations of Computational Mathematics*, vol. 12, no. 6, pp. 805–849, 2012.

[4] B. T. Polyak, *Introduction to Optimization*. Optimization Software Inc., 1987.

[5] P. L. Combettes and J.-C. Pesquet, "Proximal Splitting Methods in Signal Processing," *ArXiv e-prints*, Dec. 2009.

[6] E. T. Hale, W. Yin, and Y. Zhang, "Fixed-point continuation for l-1 minimization: Methodology and convergence," *SIAM J. on Optimization*, vol. 19, no. 3, pp. 1107–1130, Oct. 2008.

[7] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1137/080716542

[8] A. Moudafi and M. Oliny, "Convergence of a splitting inertial proximal method for monotone operators," *Journal of Computational and Applied Mathematics*, vol. 155, no. 2, pp. 447 – 454, 2003.

[9] D. Zhu and P. Marcotte, "Co-coercivity and its role in the convergence of iterative schemes for solving variational inequalities," *SIAM J. Control and Optimization*, vol. 38, pp. 431–446, 2000.

[10] F. Alvarez and H. Attouch, "An inertial proximal method for maximal monotone operators via discretization of a nonlinear oscillator with damping," *Set-Valued Analysis*, vol. 9, no. 1-2, pp. 3–11, 2001.

[11] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, pp. 11:1–11:37, June 2011. [Online]. Available: http://doi.acm.org/10.1145/1970392.1970395

[12] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, pp. 717–772, Dec. 2009. [Online]. Available: http://dx.doi.org/10.1007/s10208-009-9045-5

[13] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of Optimization*, vol. 6, no. 615-640, p. 15, 2010.

[14] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[15] M. Elad, B. Matalon, and M. Zibulevsky, "Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization," *Applied and Computational Harmonic Analysis*, vol. 23, no. 3, pp. 346–367, 2007.

[16] S. Wright and J. Nocedal, *Numerical Optimization*. Springer New York, 1999, vol. 2.

[17] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale l 1-regularized least squares," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 606–617, 2007.

[18] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani et al., "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[19] T. Blumensath and M. E. Davies, "Iterative Thresholding for Sparse Approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1007/s00041-008-9035-z

[20] J. Liu and P. Moulin, "Complexity-regularized image restoration," in *ICIP (1)*, 1998, pp. 555–559.

[21] R. Vitthal, P. Sunthar, and C. D. Rao, "The generalized proportional-integral-derivative (pid) gradient descent back propagation algorithm," *Neural Networks*, vol. 8, no. 4, pp. 563–569, 1995.

[22] G. H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "Fast sparse representation based on smoothed l0 norm." in *ICA*, ser. Lecture Notes in Computer Science, M. E. Davies, C. J. James, S. A. Abdallah, and M. D. Plumbley, Eds., vol. 4666. Springer, 2007. [Online]. Available: http://dblp.uni-trier.de/db/conf/ica/ica2007.html#MohimaniBJ07 pp. 389–396.

[23] R. Meka, P. Jain, and I. S. Dhillon, "Guaranteed rank minimization via singular value projection," *CoRR*, vol. abs/0909.5457, 2009.

[24] P. R. Johnstone, A. Emad, O. Milenkovic, and P. Moulin, "Rfit: a new algorithm for matrix rank minimization," in *The 5th Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS 2013)*, Lausanne, Switzerland, July 2013.