Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 11, No. 3, pp. 527–548, July 2015

# PRECONDITION TECHNIQUES FOR ACCELERATED LINEARIZED BREGMAN ALGORITHMS*

Tao Sun, Hui Zhang and Lizhi Cheng

**Abstract:** The linearized Bregman algorithm (LBreg) is one of most efficient methods for solving the basis pursuit and related sparse optimization problems. A couple of its variants, generalizations, and accelerated versions appeared recently. In this paper, we study precondition techniques for accelerated linearized Bregman algorithms. Theoretically, we prove that our proposed preconditionally accelerated Bregman algorithms have the fastest speed in all basic linearized Bregman algorithms, which are applied to solve the equivalent augmented $\ell_1$ norm model. On the computational level, numerical experiments on sparse signal recovery and image deblurring demonstrate the efficiency of the proposed methods.

**Key words:** *Lagrange dual theory, linearized Bregman iteration, singular value decomposition, Cholesky decomposition, precondition, optimization*

**Mathematics Subject Classification:** *46N10 80M50*

## 1 Introduction

The linearized Bregman (LBreg) algorithm was proposed and analyzed in [4,5,22] and have been intensively studied from different perspectives recently, including different variants [1,23,24], accelerated versions [11,21,25], and generalizations [13,21,26]. In this paper, we are interested in its accelerations by applying precondition techniques.

### 1.1 The LBreg algorithm

The LBreg algorithm was originally designed for solving the famous basis pursuit problem [8]

$$\min_x\{\|x\|_1 : \ Ax = b\}. \tag{1.1}$$

But it actually returns the unique solution to the following strongly convex minimization with linear constraint

$$\min_x\{\|x\|_1 + \frac{1}{2\alpha}\|x\|_2^2 : \ Ax = b\}, \tag{1.2}$$

which is called *augmented $\ell_1$ norm model* in [12], where $\alpha$ is an augmented parameter; we assume that $A$ and $b$ are both nonzero though the paper. Exact regularization properties [10,19,21] show that there exists a finite parameter $\alpha$ such that problems (1.1) and (1.2)

*We are grateful for the support from the National Natural Science Foundation of Hunan Province, China (13JJ2001), and the Science Project of National University of Defense Technology (JC120201).

are equivalent. In particular, if a sparse vector $\hat{x}$ is a solution to (1.1), then $\alpha = 10\|\hat{x}\|_\infty$ is sufficient for equivalence under some very mild conditions on matrix $A$; more details please refer to [12].

The LBreg algorithm itself is not a quite new algorithm. In fact, it can be obtained by applying classic methods to (1.2). For example, when applying the Uzawa algorithm to (1.2), one can get the following "primal-dual" form of LBreg:

$$x^{(k+1)} \leftarrow \alpha \operatorname{shrink}(A^T y^{(k)}) \tag{1.3a}$$

$$y^{(k+1)} \leftarrow y^{(k)} + h(b - Ax^{(k+1)}) \tag{1.3b}$$

where $h$ is the step size and $\operatorname{shrink}_\mu$ is the well-known shrinkage or soft-thresholding operator with parameter $\mu > 0$; we omit $\mu$ when $\mu = 1$. When applying the gradient decent method to the Lagrange dual of (1.2), i.e.,

$$\min_y g(y) \triangleq -b^T y + \frac{\alpha}{2}\|\operatorname{shrink}(A^T y)\|_2^2, \tag{1.4}$$

one can get another equivalent form of LBreg

$$y^{(k+1)} \leftarrow y^{(k)} - h(-b + \alpha A \operatorname{shrink}(A^T y^{(k)})). \tag{1.5}$$

For applying LBreg to image deblurring, the author in [6] proposed the following alternative to deal with the case where $A$ fails to be full row-rank.

$$\min_x \{\| x \|_1 + \frac{1}{2\alpha}\| x \|_2^2 : A^T A x = A^T b\} \tag{1.6}$$

Other types of forms of LBreg can be found in [12].

## 1.2 Accelerated LBreg algorithms

Due to the strong convexity of the primal objective of (1.2), the dual objective of (1.4) is differntiable and has Lipschitz-continuous gradient [18]. Based on this key observation, accelerated LBreg algorithms were designed by utilizing some techniques such as the Barzilai-Borwein accelerated method and the limited memory BFGS method [21]. Recently, the Nesterov accelerated methods were incorporated into the LBreg algorithm that result in two accelerated versions of LBreg [11, 25]: Algorithm 1 and Algorithm 2.

**Algorithm 1** Nestorov accelerated linearized Bregman iteration(NLBreg)

**Require:** parameters $\alpha > 0, h > 0$
  **Initialization**: $y^0 = 0, \theta_0 = 1$
  **for**$k = 0, 1, 2, \ldots$
  $z^{k+1} = y^k - h(-b + Ax^k)$
  $\beta_{k+1} = \frac{1}{2}(1 - \theta_k)(\sqrt{\theta_k^2 + 4} - \theta_k)$
  $y^{k+1} = z^k + \beta_k(z^{k+1} - z^k)$
  $\theta_{k+1} = \frac{1}{2}\theta_k(\sqrt{\theta_k^2 + 4} - \theta_k)$
  $x^{k+1} = \alpha \operatorname{shrink}(A^T y^{k+1})$
  **end for**

Algorithm 1 is a simple application of Nesterov's accelerated methods [14,15] to minimize gradient Lipschitz-continuous function $g(y)$; while Algorithm 2 is based on the restricted

**Algorithm 2** Accelerated linearized Bregman iteration with restarts(RLBreg)

**Require:** parameters $\alpha > 0, h = \frac{1}{\alpha\|A\|_2^2}, K > 0$
**Initialization**: $y^0 = 0, \theta_0 = 1$
**for**$k = 0, 1, 2, \ldots$
restart Algorithm 1 after K iterations to obtain $x^{j,K}$; where $K = \sqrt{8e\alpha \parallel A \parallel_2^2 / \nu}$
set $x^{j+1,0} = x^{j,K}, y^{j+1,0} = x^{j,K}, \theta_0 = 1$
**end for**

strongly convex property with parameter $\nu$ of $g(y)$, which was discovered in [12] and further exploited in [25]. Since the number $K$ for restart is not easy to compute, the authors in [25] followed a restart scheme suggested in [16, 17]:

$$Gradient\ scheme: \langle \nabla f(y^{k-1}), y^k - y^{k-1} \rangle < 0. \tag{1.7}$$

Theorem 3.3 in paper [11] proves that Algorithm 1 has a sublinear convergence rate provided the stepsize satisfies $h \leq \frac{1}{\alpha\|A\|_2^2}$ and $b = Ax$ is consistent. After that, Theorem 8 in [25] shows that Algorithm 2 enjoys a linear convergence rate if the linear system $Ax = b$ is consistent, and numerical results in [25] show that Algorithm 2 has a much faster speed than Algorithm 1. Moreover, we would like to highlight that the speed of LBreg heavily depends on the condition number of $A$ [6]. Better condition number of $A$, faster speed of the convergence. This observation motivates us to improve the condition number of $A$ and further accelerate existing LBreg algorithms. Thus, we propose two classes of algorithms: the first class is to solve model (1.2), and the other one is to solve model (1.6). Our method could be summarized into the following:

**Step 1:** Obtain a linear system of equations $Bx = d$ equivalent to the linear system of model (1.2) (model (1.6)) such that $B$ has small condition number;

**Step 2:** Apply existing accelerated LBreg algorithms to

$$\min_x \{\| x \|_1 + \frac{1}{2\alpha}\| x \|_2^2 : Bx = d\}, \tag{1.8}$$

which is equivalent to model (1.2) (model (1.6)).

In Section 3.2 we will explain why the proposed algorithms are faster than the previous ones.

### 1.3 Organization

The rest of the paper is organized as follows. Section 2 proposes the accelerated algorithms. Section 3 shows that why the accelerated algorithms run faster. Section 4 consists of numerical experiments. Finally, Section 5 concludes this article.

## 2 Preconditionally Accelerated LBreg Algorithms

In this section, our method for the first class of algorithms can be divided into two steps, namely, precondition and acceleration. In the precondition step, by forced Cholesky decomposition, we get the preconditioner $P$. Then, we obtain a matrix $V$ whose rows are orthonormal and a vector $y$ satisfying

$$Vx = y, \tag{2.1}$$

which is equivalent to $Ax = b$ by the multiplication schemes $V = P^{-1}A$ and $y = P^{-1}b$. In the acceleration step, we apply the LBreg algorithm and its accelerated versions to the following optimization:

$$\min_{x\in R^n}\{\| x \|_1 + \frac{1}{2\alpha} \| x \|_2^2 : Vx = y\}, \tag{2.2}$$

which is equivalent to (2). In next section, we will prove that the convergence rate of LBreg and RLBreg will be maximized in this situation. Due to the fact that linear system in (1.6) is a little complicated, we improve the efficiency of decomposition based on the structure of $A^TA$.

### 2.1 A brief review of preconditioning strategies

Preconditioners are employed routinely to solve sparse linear systems $Mx = z$ using iterative methods whose convergence rate depends closely on the condition number of $M$. Hence one may attempt to transform the linear system into anther equivalent one which has more favorable spectral properties. A preconditioner is a matrix that can effect such a transformation. The preconditioner is usually a non-singular matrices $P$, which approximates the coefficient matrix $M$ in some way, employed to solve the equivalent linear system

$$P^{-1}Mx = P^{-1}z. \tag{2.3}$$

What should be pointed out is that $P$ should be chosen felicitously such that the inverse of $P$ can be easily computed and $P^{-1}M$ is sparse. In the following we introduce some common preconditioners [9].

**Jacobi preconditioning** If $M$ has widely varying diagonal entries, Jacobi preconditioning can be used. This simple preconditioner consists of just the diagonal of the coefficient matrix:

$$P_{i,j} = \begin{cases} M_{i,j}, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

As a generalization of the Jacobi preconditioner, divide $M$ into

$$M = \begin{pmatrix} M_{11} & \cdots & M_{1k} \\ \vdots & \ddots & \vdots \\ M_{k1} & \cdots & M_{kk} \end{pmatrix},$$

where the diagonal blocks $M_{ii}$ are square. Then, $diag(M_{11}, M_{22}, \cdots, M_{kk})$ is the corresponding preconditioner.

**SSOR preconditioning** The SSOR preconditioner is usually used to the symmetrical coefficient matrix which is decomposed as $M = D + L^T + L$ into its diagonal, lower and upper triangular part, respectively. The SSOR preconditioner is defined as

$$P(\omega) = \frac{1}{2-\omega}(\frac{1}{\omega}D + L)(\frac{1}{\omega}D)^{-1}(\frac{1}{\omega}D + L)^T,$$

where $0 < \omega < 2$.

**Incomplete factorization preconditioning** If $M$ is symmetrical, an incomplete Cholesky factorization $LL^T$ of $M$ is an approximation $M \approx LL^T$, where $L$ is of special sparsity form. When $M$ is nonsymmetric, there is a corresponding incomplete LU preconditioner.

However, all the preconditioners introduced above are difficult to be applied to the linear constrain of (1.2) directly, because matrix $A$ is not square or sparse and may be not full row-rank; they are also poor when applied to the linear constrain of (1.6). In next subsection, enlightened by the idea of incomplete factorization preconditioning, we develop a preconditioner for the underdetermined linear system.

## 2.2 Preconditioning for the underdetermined linear system: forced Cholesky decomposition

The first method we may consider is the SVD of $A$. Assume that $A = U\Sigma V$, where $U, V$ are orthonormal matrices and $\Sigma = \begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix} \in R^{m\times n}$, $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_r)$, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r > 0$. Set the preconditioner $P = U * diag(\Lambda, I_{m-r})$, we can get

$$\begin{pmatrix} I_r & 0 \end{pmatrix} Vx = \Lambda^{-1} \begin{pmatrix} I_r & 0 \end{pmatrix} U^T b. \tag{2.5}$$

It is easy to find that the computational bottleneck lies in the SVD of $A$. Actually, SVD is unwelcome in practice for its huge computational cost. To reduce the the cost of the computation, we find $V$ by Cholesky decomposition. In what follows, two cases are discussed depending on the rank of $A$.

### Case 1: $A$ is full row-rank

**Lemma 2.1** ([7, Cholesky decomposition theorem]). *Let $M \in R^{m\times m}$ be a positive definite matrix, then there exists a lower triangular $L$ satisfying*

$$M = LL^T. \tag{2.6}$$

Let $A \in R^{m\times n}$ be a full row-rank($m < n$). Let us consider the Cholesky decomposition of $AA^T = RR^T$. Obviously, $R^{-1}A$ is a row-orthonormal matrix, and $R^{-1}Ax = R^{-1}b$ equals to $Ax = b$. It's easy to know that computing $AA^T$ will cost about $m^2n$ and the Cholesky decomposition of $AA^T$ will cost about $\frac{1}{3}m^6$, and $R^{-1}A$ will cost about $m^2n$. Then totally, the float count required is about $2m^2n + \frac{1}{6}m^3$. On the contrary, the cost of computing the SVD of $A$ is about $4mn^2 + 8m^2n + 9m^3$ [7]. Note that $m \leq n$, we have

$$\frac{T_{Cholesky}}{T_{SVD}} \approx \frac{2m^2n + \frac{1}{6}m^3}{4mn^2 + 8m^2n + 9m^3} \leq \frac{1}{8}. \tag{2.7}$$

By this way, we cut down the cost of the computation.

### Case 2: $A$ is not full row-rank

When $A$ is not full row-rank, the Cholesky decomposition of $AA^T$ cannot be done for $AA^T$ is not positive definite. In this case, we consider the Cholesky decomposition of $AA^T + \varepsilon I_m$, where $\varepsilon$ is a small positive number. This method is also called forced Cholesky decomposition.

**Theorem 2.2.** *Assume $M \in R^{m\times n}$ is not full row-rank, $\bar{L}$ is the lower triangular matrix of forced Cholesky decomposition of $MM^T$. Denote that $\bar{V} = \bar{L}^{-1}M$, $r = rank(M)$. Then,*

$$\bar{V}_{i,j} = O(\sqrt{\varepsilon}), r + 1 \leq i \leq m. \tag{2.8}$$

The proof of Theorem 2.2 can be found in Appendix. Theorem 2.2 shows that when $\varepsilon$ is small enough, $\bar{V}_{i,j}(i \geq r)$ are also very small. Therefore, in the actual calculation we can set a threshold to eliminate some rows of $\bar{V}$ to reduce the scale of matrix $\bar{V}$. The speed of each iteration will be greatly improved for a "smaller" $\bar{V}$. Assume that $\bar{R}$ is the lower triangular matrix of forced Cholesky decomposition of $AA^T$, finally, we get

$$\begin{pmatrix} I_r & 0 \end{pmatrix} \bar{R}^{-1} Ax = \begin{pmatrix} I_r & 0 \end{pmatrix} \bar{R}^{-1} b. \tag{2.9}$$

When $\varepsilon$ is very small, (2.9) approximatively equals to $Ax = b$, and $\begin{pmatrix} \bar{R}_1^{-1} R_1 & 0 \end{pmatrix} Q$ is approximatively row-orthonormal. When $r = m$, $Case$ 2 returns to $Case$ 1, and hence the forced Cholesky decomposition can also work even if $A$ is full row-rank.

## 2.3 Acceleration

Below, we propose two accelerated algorithms for augmented $l_1$ model depending on the (in)consistency of the linear system $Ax = b$.

### 2.3.1 Accelerated algorithms for consistent linear system

The algorithms proposed here are all based on the results proposed in last subsection. Here

**Algorithm 3** Accelerated linearized Bregman algorithms for consistent linear system via precondition

**Require:** parameters $h > 0, \alpha > 0, \varepsilon > 0$, threshold $\zeta > 0$

**Initialization**: $y^0 = 0, \theta_0 = 1$

Step1. get the triangular matrix $\bar{R}$ by the Cholesky decomposition of $AA^T + \varepsilon I$

Step2. update: $(\bar{R})^{-1} A \to A$,$(\bar{R})^{-1} b \to b$

Step3. for $i = 1, 2, \ldots, m$,eliminate row $i$ of $A$ and $b$ if $|\frac{A(i,i)}{\|A_i\|}| < \zeta$

**Algorithm 3.1**: **P**recondition **L**inearized **B**regman iteration (P-LBreg)

Step4. run LBreg.

**Algorithm 3.2**: **P**recondition **N**esterov Accelerated **L**inearized **B**regman iteration (PN-LBreg)

Step4. run NLBreg.

**Algorithm 3.3**: **P**recondition Accelerated **L**inearized **B**regman iteration with **R**estarts (PR-LBreg)

Step4. run RLBreg.

**end**

parameters $h$, $\alpha$ in Step 4 are the same as the ones in Algorithms 1 and 2. In step 4, Algorithm 3 use one of LBreg, NLBreg and RLBreg. $\varepsilon$ is an important parameter for Algorithm 3: theoretically, a smaller $\varepsilon$ will produce a more accurate solution. Actually a very small $\varepsilon$ will cause large error in computation of Algorithm 3. Parameter $\zeta$ is a threshold. By Theorem 2.2, we know that $\zeta$ should be set as $O(\sqrt{\varepsilon})$. In numerical experiments, we set $\zeta = 10\sqrt{\varepsilon}$.

### 2.3.2 Accelerated algorithms for inconsistent linear system

We consider the QR decomposition of $A$, where $Q$ is the orthogonal matrix and $R$ is the lower triangular matrix, $\bar{R}$ is the lower triangular matrix of forced Cholesky decomposition

of $AA^T$. Because $A$ is not full-rank , $R = \begin{pmatrix} R_1 & 0 \\ R_2 & 0 \end{pmatrix} \in R^{m\times n}$, where $R_1 \in R^{r\times r}$ is a non-singular lower triangular matrix. Divide matrix $\bar{R}$ into blocks in the same way as $R$, $\bar{R} = \begin{pmatrix} \bar{R}_1 & 0 \\ \bar{R}_2 & \bar{R}_3 \end{pmatrix}$, where $\bar{R}_1 \in R^{r\times r}$. Then $A^TAx = A^Tb$ can be equivalently transformed to

$$\begin{pmatrix} R_1^TR_1 + R_2^TR_2 & 0 \\ 0 & 0 \end{pmatrix} Qx = \begin{pmatrix} R_1^T & R_2^T \\ 0 & 0 \end{pmatrix} b, \tag{2.10}$$

where $Q = \begin{pmatrix} Q_r \\ Q_{m-r} \end{pmatrix}$ and $Q_r$ is the first $r$ rows of $Q$. (2.10) can be expressed as follow:

$$Q_rx = (R_1^TR_1 + R_2^TR_2)^{-1} \begin{pmatrix} R_1^T & R_2^T \end{pmatrix} b. \tag{2.11}$$

From the deduction in Appendix, we can conclude that $\bar{R}_i - R_i = O(\varepsilon)(i = 1, 2)$. That means when $\varepsilon$ is very small, $\bar{R}_i$ is very close to $R_i(i = 1, 2)$. Let $d = (\bar{R}_1^T\bar{R}_1 + \bar{R}_2^T\bar{R}_2)^{-1} \begin{pmatrix} \bar{R}_1^T & \bar{R}_2^T \end{pmatrix} b$, and then

$$\bar{R_1}^{-1}R_1Q_rx = d \tag{2.12}$$

is approximatively equivalent to (2.11). (2.12) is consistent for that $\bar{R_1}^{-1}R_1Q_r$ is full row-rank.

Then, we can propose corresponding accelerated algorithms in the following. Except for the different ways to update $b$, Algorithms 4.1-4.3 are almost the same as Algorithms 3.1-3.3.

**Algorithm 4** Accelerated linearized Bregman algorithms inconsistent linear system via precondition

**Require:** parameters $h > 0, \alpha > 0, \varepsilon > 0$, threshold $\zeta > 0$

**Initialization**: $y^0 = 0, \theta_0 = 1$

Step1. get the triangular matrix $\bar{R}$ by the Cholesky decomposition of $AA^T + \varepsilon I$

Step2. update: $(\bar{R})^{-1}A \to A$

Step3. for $i = 1, 2, \ldots, m$, eliminate row $i$ of $A$ if $|\frac{A(i,i)}{\|A_i\|}| < \zeta$

Step4. update: $(\bar{R}_1^T\bar{R}_1 + \bar{R}_2^T\bar{R}_2)^{-1} \begin{pmatrix} \bar{R}_1^T & \bar{R}_2^T \end{pmatrix} b \to b$

**Algorithm 4.1**:**P**recondition **L**inearized **B**regman iteration with **I**nconsistent linear system (IP-LBreg)

Step5. run LBreg.

**Algorithm 4.2**:**P**recondition **N**esterov accelerated **L**inearized **B**regman iteration with **I**nconsistent linear system (IPN-LBreg)

Step5. run NLBreg.

**Algorithm 4.3**:**P**recondition Accelerated **L**inearized **B**regman iteration with **R**estarts with **I**nconsistent linear system (IPR-LBreg)

Step5. run RLBreg.

**end**

## 3 Why Faster?

In iteration (5), paper [12] shows that

$$\| y^k - y^k_{prj} \|_2 \leq \sqrt{1 - (\frac{\nu}{\alpha \| A \|_2^2})^2} \| y^{k-1} - y^{k-1}_{prj} \|_2, \tag{3.1}$$

where $y^k_{prj}$ is the projection of $y^k$ onto solution set of problem (4), $\nu$ is the RSC(defined in section 3.1) constant of the objective function of problem (4). Later, paper [25] improved the rate of convergence and obtained

$$\| y^k - y^k_{prj} \|_2 \leq \sqrt{1 - \frac{\nu}{\alpha \| A \|_2^2}} \| y^{k-1} - y^{k-1}_{prj} \|_2 . \tag{3.2}$$

Besides this, [25] also proved that RLBreg reaches $\varepsilon$-accuracy in $O(\sqrt{\frac{\alpha\|A\|_2^2}{\nu}} log(\frac{1}{\varepsilon}))$ iterations. It is easy to find that $\frac{\nu}{\|A\|_2^2}$ is the key to the efficiency of LBreg and RLBreg.

**Definition 3.1.** Let $M \in R^{m\times n}, z \in R^n, \alpha > 0$, and $Mx = z$ be consistent.

$$\omega_M := \nu_M / \| M \|_2^2, \tag{3.3}$$

where $\| M \|_2$ is the spectral radius of $M$.

From the discussion above we know that the larger $\omega_M$ is, the faster LBreg and RLBreg are.

## 3.1 Preliminaries

We first collect some definitions and lemmas required for upcoming analysis.

**Definition 3.2** ([12])**.** Let $M$ be a positive semi-definite matrix. Define

$$\lambda^{++}_{min}(M) := \min_{\lambda(M)\in S(M)} \{\lambda(M)\}, \tag{3.4}$$

where $S(M)$ is the set of all positive eigenvalues of $M$.

**Definition 3.3** ( [12])**.** (*Restricted strong convexity − RSC*($\nu$)). A convex function $f(x)$:$R^n \to R$ is restricted strongly convex with constant $\nu > 0$ if it is differentiable, and obeys:

$$\langle \nabla f(x) - \nabla f(x_{prj}), x - x_{prj} \rangle \geq \nu \| x - x_{prj} \|_2^2, \tag{3.5}$$

where $x_{prj} = prj_{\chi^*}(x)$ is the projection of $x$ onto the set $\chi^*$, and $\chi^*$ is the solution of problem $\min_x\{f(x)\}$.

**Lemma 3.4** ([12,25])**.** *The objective function of problem (1.4) is RSC, and the RSC constant is*

$$\nu_A = \lambda_A \cdot \min_{i\in supp(x^*)} \{\frac{\alpha|x^*_i|}{\alpha|x^*_i| + 2}\}. \tag{3.6}$$

*where* $\lambda_A = min\{\lambda^{++}_{min}(C^TC) | C$ *is a nonzero submatrix of* $A$ *of* $m$ *rows*$\}$, $x^*$ *is the unique solution of model (1.2),* $supp(x^*)$ *is support set of* $x^*$.

**Lemma 3.5** ([12])**.** *Let* $D \in R^{n\times n}$ *be a diagonal positive define matrix, and* $M \in R^{n\times n}$. *Then*

$$\lambda^{++}_{min}(M^TDM) = \min_{\|M^T\alpha\|=1} \{\alpha^T MM^T DMM^T \alpha\}. \tag{3.7}$$

### 3.2 Technique lemmas and main results

Before stating the main result, we need several technique lemmas.

**Lemma 3.6.** *Let convex function $f$ be differentiable, $U \in R^{m\times m}, UU^T = I$, and denote $g(y) = f(U^T y)$. If $f(x)$ is RSC($\nu$), then $g(y)$ is also RSC($\nu$).*

*Proof.* Assume that $\chi^* = \{x|\nabla f(x) = 0\}$, $\phi^* = \{y|\nabla f(U^T y) = 0\}$; obviously, $\chi^* = U\phi^*$. It's easy to know that $\chi^*$ is the solution of problem $\min_x\{f(x)\}$, and $\phi^*$ is the solution of problem $\min_y\{g(y)\}$. For any vector $y$, it holds that

$$prj_{\phi^*}(y) = arg\min_{\bar{y}\in\phi^*}\{\frac{1}{2} \parallel y - \bar{y} \parallel_2^2\}. \tag{3.8}$$

Then, for $UU^T = I$, we have

$$prj_{\phi^*}(y) = arg\min_{\bar{y}\in\phi^*}\{\frac{1}{2} \parallel U^T y - U^T\bar{y} \parallel_2^2\}, \tag{3.9}$$

Let $\hat{y} = U^T\bar{y}$; Then,

$$prj_{\phi^*}(y) = arg\min_{\hat{y}\in\chi^*}\{\frac{1}{2} \parallel U^T y - \hat{y} \parallel_2^2\}. \tag{3.10}$$

Namely $\forall y$, $U^T prj_{\phi^*}(y) = prj_{\chi^*}(U^T y)$.

Now, we verify that g(y) is RSC with constant $\nu$:

$$\begin{aligned}
\langle y - prj_{\phi^*}(y), \nabla g(y)\rangle &= \langle U^T y - U^T prj_{\phi^*}(y), \nabla f(U^T y)\rangle && (3.11a)\\
&= \langle U^T y - prj_{\chi^*}(U^T y), \nabla f(U^T y)\rangle && (3.11b)\\
&\geq \nu \parallel U^T y - prj_{\chi^*}(U^T y) \parallel_2^2 && (3.11c)\\
&= \nu \parallel U^T y - U^T prj_{\phi^*}(y) \parallel_2^2 && (3.11d)\\
&= \nu \parallel y - prj_{\phi^*}(y) \parallel_2^2 . && (3.11e)
\end{aligned}$$

□

**Lemma 3.7.** *Let $m < k < n$, $M \in R^{m\times n}, z \in R^m$, and $Mx = z$ be consistent. Denote that $\hat{M} = \begin{pmatrix} M \\ 0 \end{pmatrix} \in R^{k\times n}$, $\hat{z} = \begin{pmatrix} z \\ 0 \end{pmatrix} \in R^k$. Then $\omega_M = \omega_{\hat{M}}$.*

*Proof.* For $Mx = z$ is equivalent to $\hat{M}x = \hat{z}$, the solution to $\min_{x\in R^n}\{\parallel x \parallel_1 + \frac{1}{2\alpha}\parallel x \parallel_2^2 : Mx = z\}$ is also the solution to $\min_{x\in R^n}\{\parallel x \parallel_1 + \frac{1}{2\alpha}\parallel x \parallel_2^2 : \hat{M}x = \hat{z}\}$. By Lemma 3.4, what we should do is just to verify whether $\lambda_M = \lambda_{\hat{M}}$. If $\hat{C}$ is a nonzero submatrix of $\hat{M}$ of $k$ rows, it's easy to know $\hat{C} = \begin{pmatrix} C \\ 0 \end{pmatrix}$, where $C$ is a nonzero submatrix of $M$ of $m$ rows.

$$\begin{aligned}
\lambda_{\hat{M}} &= min\{\lambda_{min}^{++}(\hat{C}^T\hat{C})|\hat{C}\ is\ a\ nonzero\ submatrix\ of\ \hat{M}\ of\ k\ rows\}\\
&= min\{\lambda_{min}^{++}(C^T C)|C\ is\ a\ nonzero\ submatrix\ of\ M\ of\ m\ rows\}\\
&= \lambda_M && (3.12)
\end{aligned}$$

So we get $\omega_M = \omega_{\hat{M}}$. □

**Lemma 3.8.** *Let $Q \in R^{m\times m}, M \in R^{m\times n}, QQ^T = I$, then $\omega_M = \omega_{QM}$.*

*Proof.* It's easy to know that $\nu_{QM} = \nu_M$ by Lemma 3.6. For $QQ^T = I$, $\| QM \|_2 = \| M \|_2$. By Definition 3.2

$$\omega_{QM} = \nu_{QM} / \| QM \|_2^2 = \nu_M / \| M \|_2^2 = \omega_M. \tag{3.13}$$

□

**Lemma 3.9.** *Assume that row-orthonormal matrix $V_i \in R^{r\times n}$ (i=1,2) satisfy two equivalent linear systems $V_1 x = b_1$ and $V_2 x = b_2$. Then $\omega_{V_1} = \omega_{V_2}$.*

*Proof.* There exists matrix $Q$ satisfying $V_1 = QV_2$. Since $V_i$ is an row-orthonormal matrix, $I_r = V_1 V_1^T = QV_2 V_2^T Q^T = QQ^T$. Then $Q$ is also an orthogonal matrix. By Lemma 3.6, $\omega_{V_1} = \omega_{V_2}$. □

**Lemma 3.10.** *Let $M \in R^{m\times n}, z \in R^n$, and $Mx = z$ be consistent. Assume that the SVD of $M$ is $M = U\Sigma V$, where $U, V$ are orthogonal matrices and $\Sigma = \begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix} \in R^{m\times n}$, $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_r)$, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r > 0$, denote that $V_r = \begin{pmatrix} I_{r\times r} & 0_{r\times(m-r)} \end{pmatrix} V$, $z_r = \Lambda^{-1} \begin{pmatrix} I_{r\times r} & 0_{r\times(m-r)} \end{pmatrix} U^T z$. Then, the linear system $V_r x = z_r$ is equivalent to the linear system $Mx = z$, and $\omega_M \leq \omega_{V_r}$.*

*Proof.* For $U$ is orthogonal,

$$\Sigma V x = U^T z \Longrightarrow \begin{pmatrix} \Lambda V_r \\ 0 \end{pmatrix} x = U^T z. \tag{3.14}$$

It's easy to get $\omega_{\Sigma V} = \omega_M$ by Lemma 3.6. Actually, this is equivalent to

$$\Lambda V_r x = \begin{pmatrix} I_{r\times r} & 0_{r\times(m-r)} \end{pmatrix} U^T b. \tag{3.15}$$

By Lemma 3.7, we can get $\omega_{\Sigma V} = \omega_{\Lambda V_r}$. And (3.15) is also equivalent to

$$V_r x = \Lambda^{-1} \begin{pmatrix} I_{r\times r} & 0_{r\times(m-r)} \end{pmatrix} U^T z = z_r. \tag{3.16}$$

Next, we prove $\omega_{\Lambda V_r} \leq \omega_{V_r}$ by Lemma 3.4. We derive that

$$\begin{aligned}
\lambda_{\Lambda V_r} &= \min_C \{\lambda_{min}^{++}(C^T \Lambda^T \Lambda C)\} && (3.17a)\\
&= \min_C \min_{\|C^T\alpha\|=1} \{\alpha^T C C^T \Lambda^T \Lambda C C^T \alpha\} && (3.17b)\\
&\leq \min_C \min_{\|C^T\alpha\|=1} \{\alpha^T C C^T \Lambda^T \Lambda C C^T \alpha\} && (3.17c)\\
&= \lambda_1^2 \min_C \min_{\|C^T\alpha\|=1} \{\alpha^T C C^T C C^T \alpha\} && (3.17d)\\
&= \lambda_1^2 \min_C \{\lambda_{min}^{++}(C^T C)\} && (3.17e)\\
&= \lambda_1^2 \lambda_{V_r}, && (3.17f)
\end{aligned}$$

where $C$ is a nonzero submatrix of $V_r$ of $r$ rows. For all linear systems mentioned above are equivalent, so they enjoy the same solution $x^*$, by Lemma 3.4 $\nu_{\Sigma V_r} \leq \lambda_1^2 \nu_{V_r}$. Clearly, $\| V_r \|_2 = 1, \| \Sigma V_r \|_2 = \lambda_1$. Then,

$$\omega_{\Sigma V_r} = \frac{\nu_{\Sigma V_r}}{\| \Sigma V_r \|_2^2} \leq \frac{\lambda_1^2 \nu_{V_r}}{\lambda_1^2} = \frac{\nu_{V_r}}{1} = \frac{\nu_{V_r}}{\| V_r \|_2^2} = \omega_{V_r}. \tag{3.18}$$

Therefore,

$$\omega_M = \omega_{\Sigma V_r} \leq \omega_{V_r}. \tag{3.19}$$

□

**Definition 3.11.** Assume that $Ax = b$ is consistent with $A \in R^{m\times n}, b \in R^m$.

$$S_A := \{M | There\ exists\ z \in R^m\ satisfying\ that\ Mx = z\ equals\ to\ Ax = b\}. \tag{3.20}$$

The following theorem guarantees that the convergence speed of pretreated problem is superior to the original one.

**Theorem 3.12.** *Assume that $Ax = b$ is consistent and $A \in R^{m\times n}, b \in R^m, y \in R^r$. Row-orthonormal matrix $V \in R^{r\times n}$ satisfies the linear system $Vx = y$, which equals to $Ax = b$. Then*

$$\omega_V = \sup_{M\in S_A}\{\omega_M\}. \tag{3.21}$$

*Proof.* By Definition 3.11, $\forall M \in S_A$, there exists a vector $z$ satisfying $Mx = z$. Then, $Vx = y$ equals to $Mx = z$. Obviously, by Lemma 3.10, $Vx = y$ also equals to $V_r x = z_r$, where all notations are the same as in Lemma 3.10. By Lemma 3.9, $\omega_V = \omega_{V_r} \geq \omega_M$. □

Theorem **??** tells that the convergent rate of the LBreg or RLBreg applied to $\min_{x\in R^n}\{\| x \|_1 + \frac{1}{2\alpha}\| x \|_2^2 : Vx = y\}$ is lager than the one applied to $\min_{x\in R^n}\{\| x \|_1 + \frac{1}{2\alpha}\| x \|_2^2 : Mx = z, M \in S_A\}$. The $M$ is any matrix which belongs to $S_A$; of course includes $A$ itself. Therefore, Theorem **??** provides the theoretical support for Algorithms 3 and 4.

## 4 Numerical Demonstration

In this section, we present three examples in each subsection to demonstrate that our algorithms run faster.

### 4.1 Comparison of the algorithms for model (1.2)

The sparse signals $x^* \in R^{2400}$ with $N$ nonzero entries sampled independently from the standard Gaussian distribution. The examples have a sensing matrix $A$ formed by multiplication $A = BC$, where $B \in R^{1000\times r}$ and $C \in R^{r\times 2400}$ are both generated by Matlab function $randn(\cdot,\cdot)$. The following parameters are used as follow: $b = Ax^*, \alpha = 10 \| x^* \|_\infty$, $\varepsilon = 10^{-6}, \zeta = 10^{-2}$, and $h = 1/\alpha \| A \|_2^2$. All iterations were stopped if $\| Ax^k - b \|_2 < 10^{-12}$ or 5000 iterations. The primal solution relative error is calculated by

$$RE = \frac{\|x^k - x^*\|_2}{\|x^*\|_2}.$$

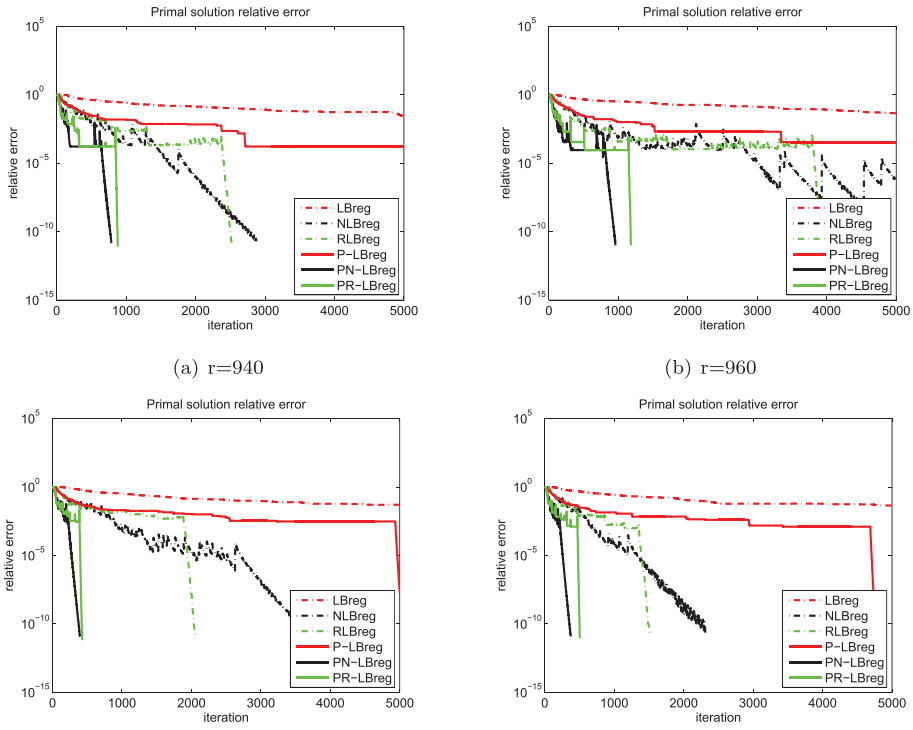
In numerical tests, we are also interested in the function value $f(x^k)$. Here, we employ the definition of the function relative error in [20]:

$$FRE = \frac{|f(x^k) - f(x^*)|}{|f(x^*)|}.$$

In test 1, we set $N = 150$ and $r = 940, 960, 980, 1000$. Figure 1 depicts primal solution relative error versus iteration $k$ of different $r$, Figure 2 depicts the function relative error versus iteration $k$ of different $r$. Table 1 reports the time(T) that each algorithm finally costs, the primal solution relative error (RE) and function relative error (FRE) that each algorithm finally gets.

Table 1: Time, primal solution relative error and function relative error for different $r$ in test 1

| scenario | r=940 | r=960 | r=980 | r=1000 |
|---|---|---|---|---|
| LBreg | 53.00s, 2.75E-2, 9.4E-3 | 54.70s, 4.54E-2, 1.50E-2 | 53.20s, 5.12E-2, 1.77E-2 | 53.20s, 4.53E-2, 1.4E-2 |
| P-LBreg | 50.80s, 1.66E-4, 2.85E-6 | 52.50s, 3.46E-4, 1.96E-5 | 53.20s, 1.77E-8, 7.92E-9 | 51.60s, 1.30E-11, 4.07E-12 |
| NLBreg | 31.10s, 1.95E-11, 3.65E-13 | 53.40s, 5.79E-7, 1.55E-12 | 39.20s, 1.70E-11, 4.72E-13 | 24.70s, 1.83E-11, 1.59E-13 |
| PN-LBreg | 9.71s, 1.46E-11, 3.77E-13 | 11.60s, 1.01E-11, 5.92E-13 | 6.20s, 1.23E-11, 3.67E-13 | 6.01s, 1.21E-11, 7.63E-13 |
| RLBreg | 27.50s, 1.61E-11, 2.86E-13 | 42.20s, 1.52E-11, 2.48E-13 | 22.10s, 1.49E-11, 9.72E-13 | 16.50s, 1.47E-11, 1.41E-13 |
| PR-LBreg | 10.50s, 8.12E-12, 8.94E-14 | 13.90s, 1.02E-11, 2.27E-13 | 6.48s, 6.64E-12, 8.47E-13 | 6.40s, 9.07E-12, 1.84E-13 |



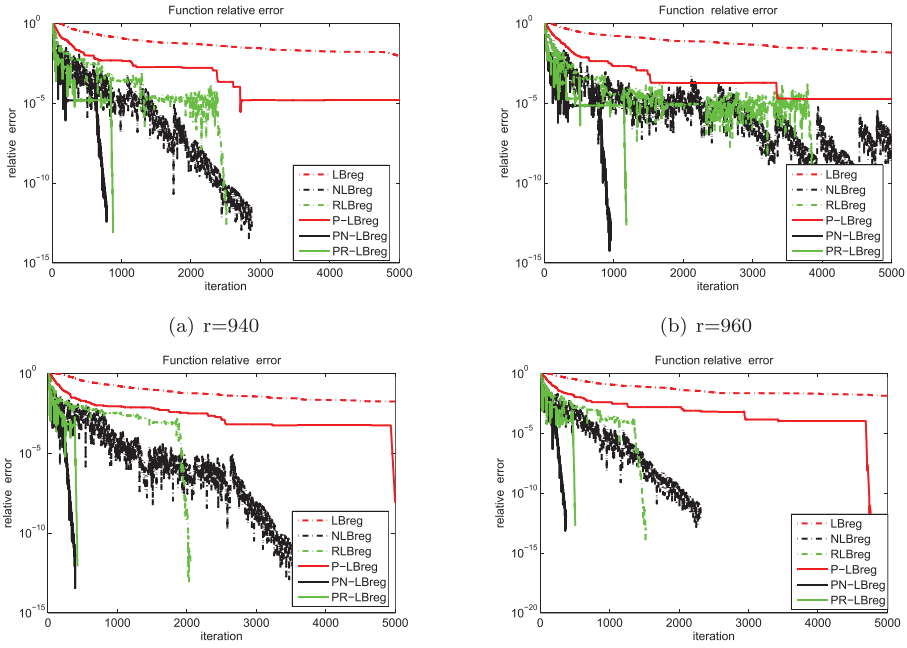

(a) r=940 (b) r=960

(c) r=980 (d) r=1000

Figure 1: Primal solution relative error of different $r$ in test 1

In test 2, the rank of the sensing matrix is set as $r = 960$, and sparsity levels of signal are set as $N = 110, 120, 130, 140$. The other parameters are the same as in test 1. Similarly, Figure 3 depicts the primal solution relative error versus iteration $k$ of different $N$, Figure 4 depicts the function relative error versus iteration $k$ of different $r$, and Table 2 reports $(T, RE, FRE)$ versus $N$. Now, we present the results of test 2 as follow:

The CPU times of P-LBreg, PN-LBreg and PR-LBreg in Table 1 and Table 2 include the time for pre-conditioning. It can be found that P-LBreg, PN-LBreg and PR-LBreg are much faster than LBreg, NLBreg and RLBreg.




(a) r=940 (b) r=960

(c) r=980 (d) r=1000

Figure 2: Function relative error of different $r$ in test 1

## 4.2 Comparison of the algorithms for model(1.6)

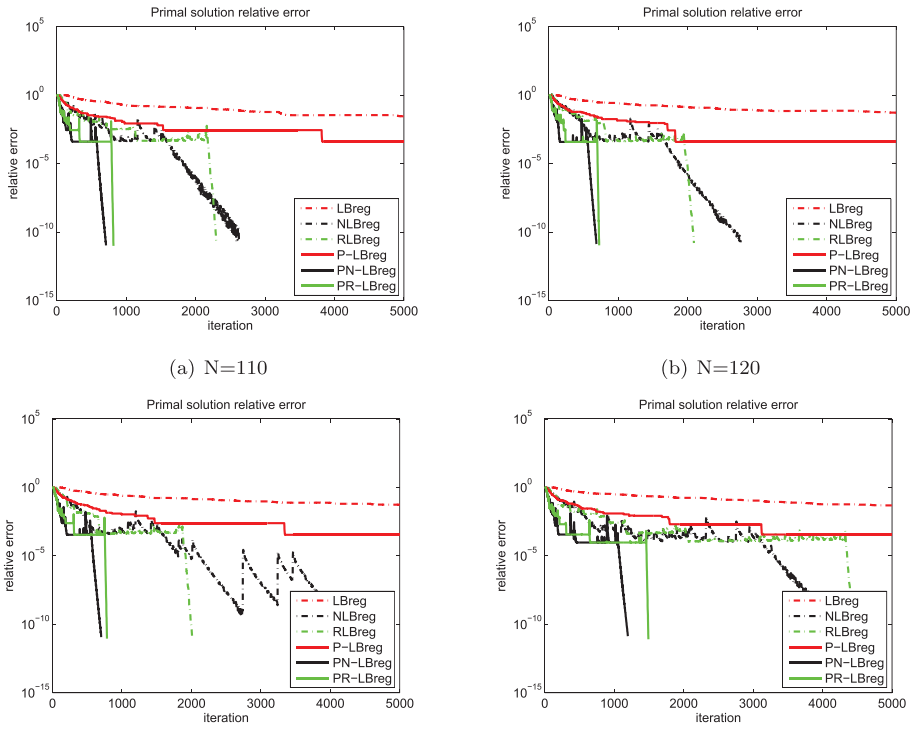
In this subsection the sparse signals $x^* \in R^{1200}$ with $N$ nonzero entries sampled independently from the standard Gaussian distribution. The tests in this subsection all have a sensing matrix $A$ with 500 rows and entries sampled independently from the standard Gaussian distribution. The rank of $A$ is $r$.

But we barge up against immediately. The first problem we encounter is how to create an inconsistent linear system. The second one is that if we get that system, how to get the solution of problem (1.6). In this part we set

$$b = Ax^* + 10^{-6}\frac{a}{\| a \|_2}, \tag{4.1}$$

Table 2: Time, primal solution relative error and function relative error for different $N$ in test 2

| scenario | N=110 | N=120 | N=130 | N=140 |
|---|---|---|---|---|
| LBreg | 53.40s, 2.84E-2, 1.20E-2 | 52.70s, 5.22E-2, 1.75E-2 | 53.00s, 5.53E-2, 1.64E-2 | 52.80s, 4.83E-2, 1.46E-2 |
| P-LBreg | 51.70s, 3.98E-4, 7.41E-5 | 51.60s, 3.79E-4, 4.83E-5 | 52.10s, 3.50E-4, 5.21E-5 | 51.90s, 3.63E-4, 4.32E-5 |
| NLBreg | 28.30s, 1.71E-11, 7.03E-13 | 29.30s, 1.76E-11, 4.98E-13 | 46.20s, 1.75E-11, 3.80E-13 | 45.30s, 1.68E-11, 2.06E-13 |
| PN-LBreg | 9.14s, 1.05E-11, 2.93E-13 | 8.91s, 1.33E-11, 1.70E-13 | 9.07s, 1.16E-11, 4.23E-13 | 14.10s, 1.37E-11, 3.28E-13 |
| RLBreg | 25.00s, 1.41E-11, 8.56E-14 | 22.30s, 1.58E-11, 3.31E-13 | 21.80s, 1.48E-11, 2.68E-13 | 47.40s, 1.37E-11, 2.06E-13 |
| PR-LBreg | 10.30s, 9.72E-12, 3.67E-13 | 9.26s, 1.06E-11, 1.75E-13 | 9.96s, 8.89E-12, 3.54E-13 | 17.20s, 8.16E-12, 8.28E-13 |




(a) N=110 (b) N=120

(c) N=130 (d) N=140

Figure 3: Primal solution relative error of different $N$ in test 2

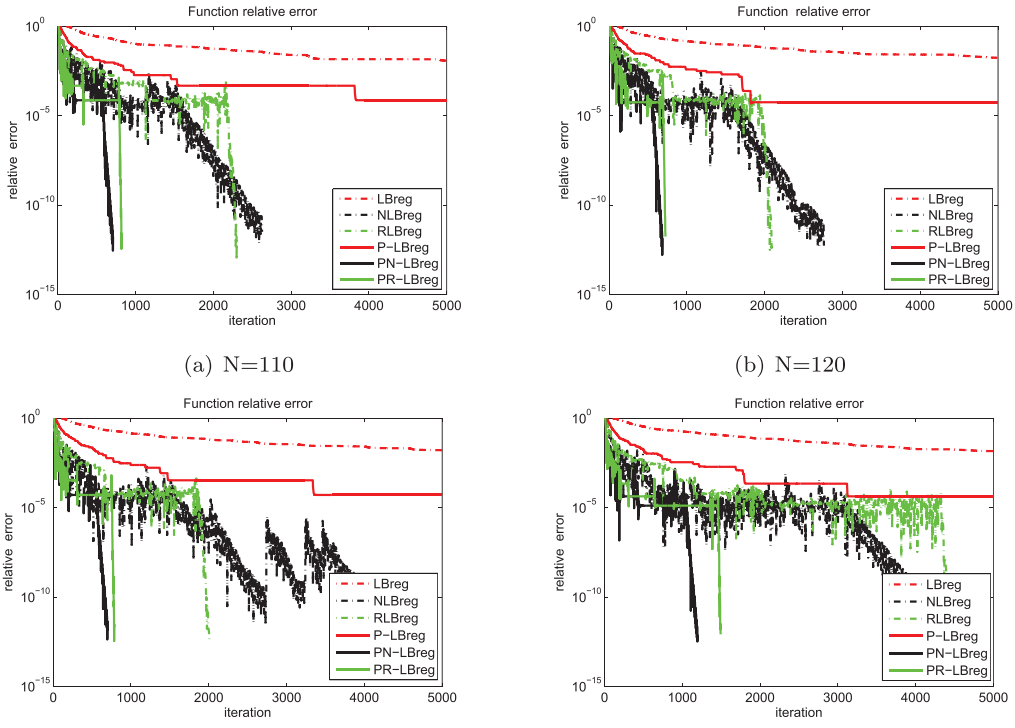
where $a$ is a nonzero random vector. (4.1) is consistent if and only if $a \in range(A)$, here $range(A) := \{Ax | x \in R^n\}$. Obviously, $dim(range(A)) = rank(A) < m$. Therefore, $P\{a \in range(A)\} = 0$. The tail added in (4.1) can make $Ax = b$ be an inconsistent linear system. On the other hand, the tail is very small, so we can regard $x^*$ as the solution of (1.6). Then by using this method we successfully solve the two problems we faced.

The other parameters here are as follows: $\alpha = 10 \parallel x^* \parallel_\infty, \varepsilon = 10^{-6}, \zeta = 0.01$, and the fixed step $h = 1/\alpha \parallel A \parallel_2^2$. All iterations were stopped if $\parallel Ax^k - b \parallel_2 < 10^{-6}$ or 5000 iterations.

In test 1, we set $N = 50$ and $r = 450, 460, 470, 480$. Figure 5 depicts the primal solution relative error versus iteration $k$ of different $N$, Figure 6 depicts the function relative error versus iteration $k$ of different $r$, and Table 3 reports $(T, RE, FRE)$ versus $r$. All notations are the same as before. The results of test 1 are as follows:

In test 2, the rank of the sensing matrix is set as $r = 450$, and sparsity levels of signal are set as $N = 35, 45, 55, 65$. The other parameters are the same as in test 1. Figure 7 depicts the primal solution relative error versus iteration $k$ of different $N$, Figure 8 depicts the function relative error versus iteration $k$ of different $r$, and Table 4 depicts $(T, RE, FRE)$ versus the sparsity of the signal.

The CPU times of IP-LBreg, IPN-LBreg and IPR-LBreg in Tables 3 and 4 also include the time for pre-conditioning. From the numerical results we get the same conclusion: IP-LBreg, IPN-LBreg and IPR-LBreg run much faster.




(a) N=110 (b) N=120

(c) N=130 (d) N=140

Figure 4: Function relative error of different $N$ in test 2

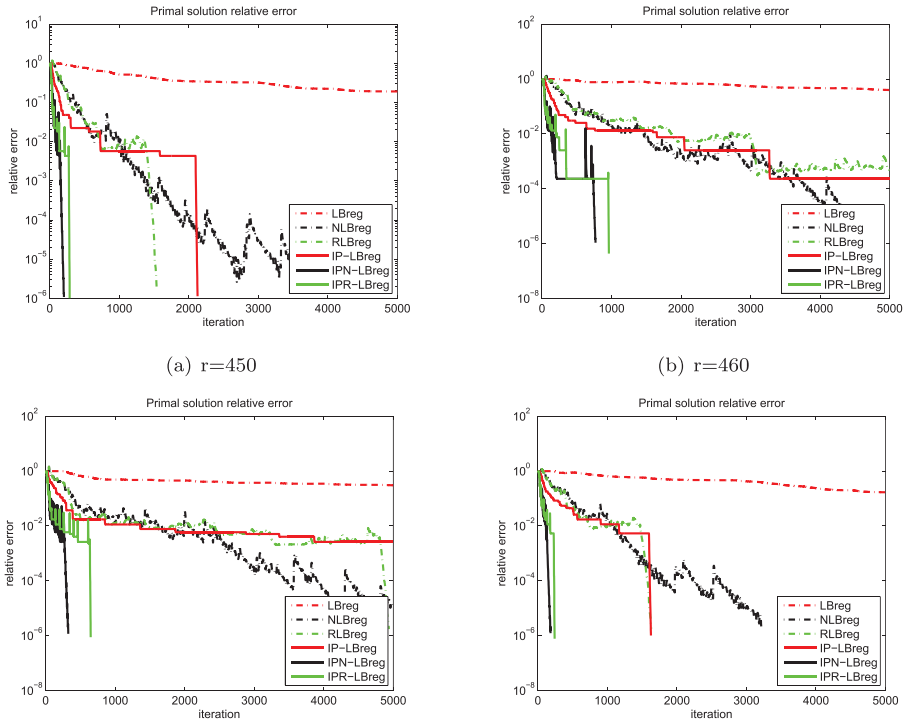
### 4.3 Application to image deblurring

The Frame-Based Deblurring model [6] can be expressed as follows:

$$\min_{u\in R^n}\{\| u \|_1 + \frac{1}{2\alpha}\| u \|_2^2 : BFu = g\}, \tag{4.2}$$

where $B$ is the linear blurring operator matrix, and $F$ is a tight frame matrix [2]. We omit the details about model (4.2) here, since the details can be found in [2,6]. Here, the blurring kernel is a $15\times 15$ Gaussian one with $\sigma = 2$ which can be easily generated by the MATLAB command $fspecial('Gaussian', 15, 2)$. The tight frame is chosen as the same as the one in [6]. The other parameters are set as the same as the ones in section 4.1, all iterations were stopped if $\| BFu^k - g \|_2 < 10^{-2}$ or 200 iterations. Figure 9 shows the results; and the CPU times contain the time for preconditioning.

The preconditioning implemented here is the same as what stated in Section 2.2: first, we get a triangular matrix $R$ by calculating the Cholesky decomposition of $BF(BF)^T + \varepsilon I$; then, we multiple both sides of the constrained linear system by $R^{-1}$. Actually, note that $F$ is a tight frame matrix which satisfies $FF^T = I$, we just need to calculate the Cholesky decomposition of $BB^T + \varepsilon I$. The numerical results on the fingerprint deblurring experiment coincide the conclusions we have got above. The preconditioned algorithms preform much better: they can get a better solution in less time.




(a) r=450 (b) r=460

(c) r=470 (d) r=480

Figure 5: Primal solution relative error of different $r$ in test 1

## 5 Conclusion

In this study, we have shown that LBreg and RLBreg can be accelerated impressively after applying precondition techniques to them. The main observation lies in that the smaller condition number of the sensing matrices, the faster speed of LBreg and RLBreg. We also consider the case, which arise in image restoration, for the inconsistent linear system. Numerical experiments on sparse signal recovery and image deblurring demonstrate that the proposed algorithms perform reasonably faster than the previous ones.

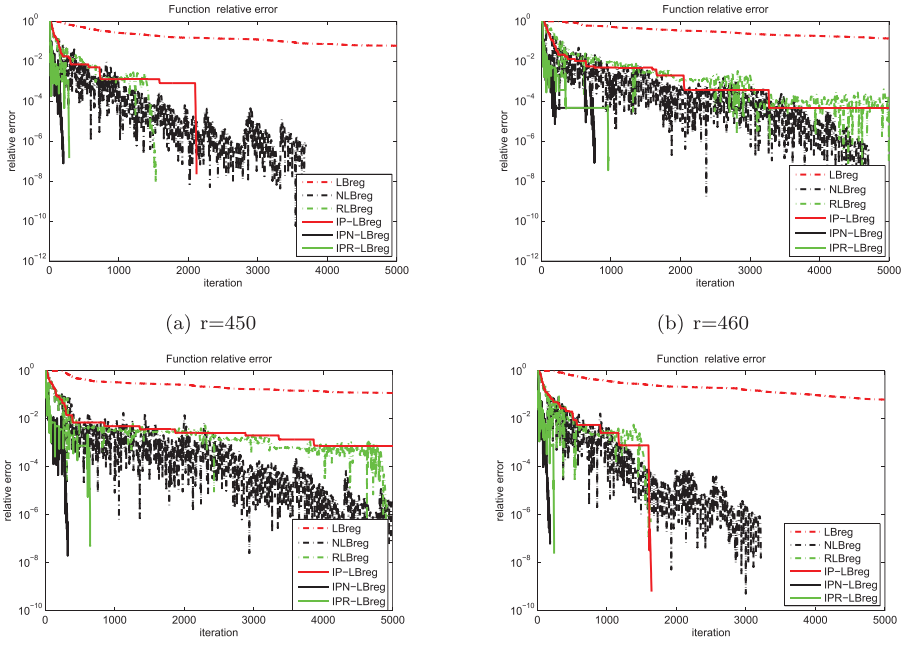
## Appendix

*The proof of Theorem 2.2:* Assume that the QR decomposition of $M$ is $M = LQ$, where $Q \in R^{n\times n}$ is the orthonormal matrix and $L = \begin{pmatrix} L_1 & 0 \\ L_2 & 0 \end{pmatrix} \in R^{m\times n}$, with $L_1 \in R^{r\times r}$ being a non-singular lower triangular matrix. Obviously,

$$MM^T + \varepsilon I_m = LL^T + \varepsilon I_m = \bar{L}\bar{L}^T. \tag{5.1}$$

Thus, it's easy to get

$$L_{1,1}^2 + \varepsilon = \bar{L}_{1,1}^2. \tag{5.2}$$




(a) r=450 (b) r=460

(c) r=470 (d) r=480

Figure 6: Function relative error of different $r$ in test 1

Let $L_{1,1} = sign(\bar{L}_{1,1})\sqrt{\bar{L}_{1,1}^2 - \varepsilon}$. Therefore,

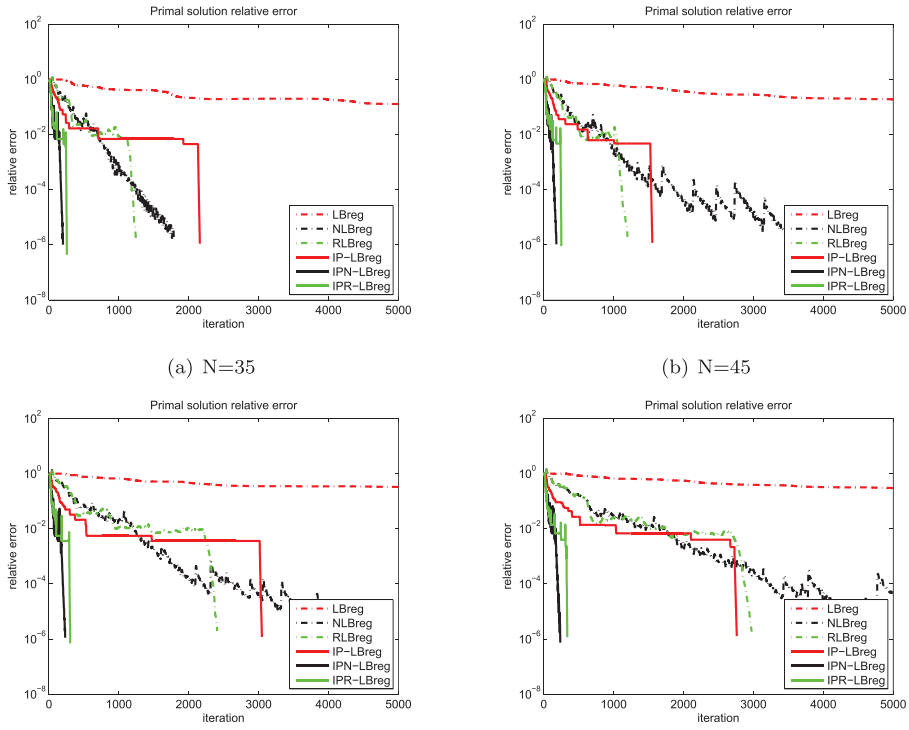
$$\bar{L}_{1,1} = L_{1,1} + O(\varepsilon). \tag{5.3}$$

By induction and the explicit scheme of Cholesky decomposition , while $1 \leq j \leq r$

$$\bar{L}_{i,j} = L_{i,j} + O(\varepsilon). \tag{5.4}$$

Table 3: Time, primal solution relative error and function relative error for different $N$ in test 1

| scenario | r=450 | r=460 | r=470 | r=480 |
|---|---|---|---|---|
| LBreg | 34.70s, 1.92E-1, 5.95E-2 | 35.70s, 3.98E-1, 1.38E-1 | 34.90s, 3.05E-1, 1.19E-1 | 37.10s, 1.70E-1, 6.40E-2 |
| IP-LBreg | 6.21s, 1.14E-6, 9.29E-8 | 14.70s, 2.31E-4, 4.60E-5 | 14.80s, 2.62E-3, 7.62E-4 | 5.05s, 1.02E-6, 1.13E-10 |
| NLBreg | 25.80s, 1.90E-6, 5.93E-10 | 33.10s, 1.60E-6, 4.32E-8 | 34.90s, 1.0E-5, 1.83E-9 | 22.40s, 1.88E-6, 5.26E-9 |
| IPN-LBreg | 0.90s, 1.09E-6, 7.90E-9 | 2.51s, 1.08E-6, 8.47E-8 | 1.28s, 1.18E-6, 2.02E-8 | 0.89s, 1.24E-6, 6.28E-8 |
| RLBreg | 11.00s, 1.91E-6, 9.29E-9 | 35.40s, 4.71E-4, 5.73E-8 | 34.40s, 1.75E-6, 9.07E-8 | 11.50s, 1.83E-6, 1.24E-7 |
| IPR-LBreg | 1.13s, 1.02E-6, 1.41E-8 | 3.07s, 4.35E-7, 3.53E-8 | 2.21s, 8.82E-7, 4.83E-8 | 1.05s, 7.75E-7, 2.39E-8 |




(a) N=35 (b) N=45

(c) N=55 (d) N=65

Figure 7: Primal solution relative error of different $N$ in test 2

By (5.1), while $i = r + 1$,

$$\sum_{j=1}^{r+1} \bar{L}_{r+1,j}^2 = \sum_{j=1}^{r+1} L_{r+1,j}^2 + \varepsilon$$

$$\sum_{j=1}^{r} L_{r+1,j}^2 + \bar{L}_{r+1,r+1}^2 + O(\varepsilon) = \sum_{j=1}^{r} L_{r+1,j}^2 + L_{r+1,r+1}^2 + \varepsilon$$

Because $L_{r+1,r+1} = 0$,

$$\bar{L}_{r+1,r+1} = O(\sqrt{\varepsilon}). \tag{5.5}$$

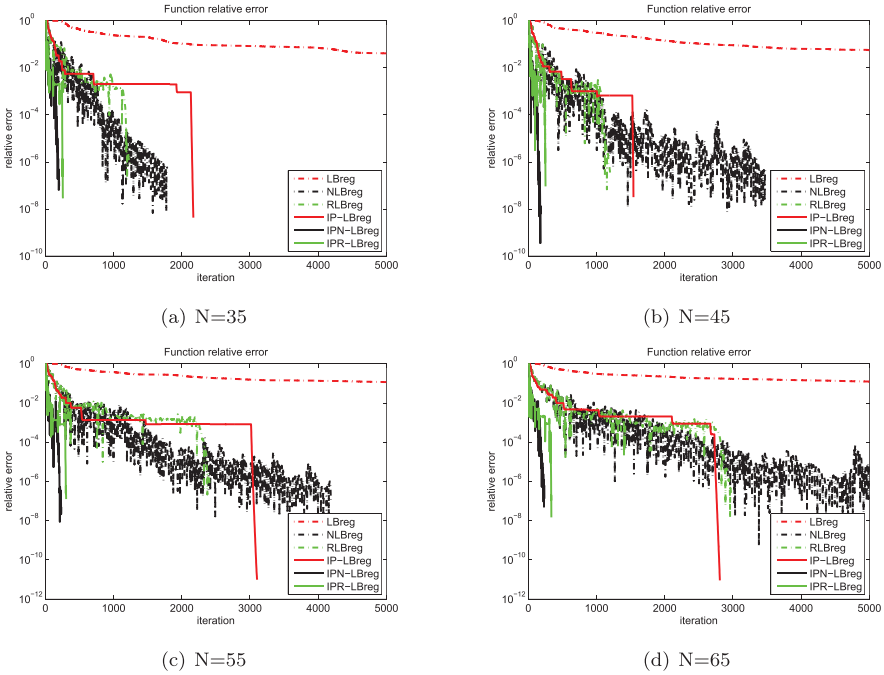
Similarly, using induction and the Cholesky decomposition explicit scheme, while $r + 1 \leq j \leq m$

$$\bar{L}_{i,j} = O(\sqrt{\varepsilon}), j \leq i. \tag{5.6}$$

Denote that $\bar{L} = \begin{pmatrix} \bar{L}_1 & 0 \\ \bar{L}_2 & \bar{R}_3 \end{pmatrix}$, where $\bar{L}_1 \in R^{r\times r}$. By the explicit scheme of inversion of block lower triangular matrix, we have

$$\bar{V} = \bar{L}^{-1}M = \begin{pmatrix} \bar{L}_1^{-1} & 0 \\ -\bar{L}_3^{-1}\bar{L}_2\bar{L}_1^{-1} & \bar{L}_3^{-1} \end{pmatrix} \begin{pmatrix} L_1 & 0 \\ L_2 & 0 \end{pmatrix} Q \tag{5.7a}$$

$$= \begin{pmatrix} \bar{L}_1^{-1}L_1 & 0 \\ -\bar{L}_3^{-1}(L_2 - \bar{L}_2\bar{L}_1^{-1}L_1) & 0 \end{pmatrix} Q. \tag{5.7b}$$




(a) N=35 (b) N=45

(c) N=55 (d) N=65

Figure 8: Function relative error of different $N$ in test 2

Now, we estimate $L_2 - \bar{L}_2\bar{L}_1^{-1}L_1$ as follows:

$$L_2 - \bar{L}_2\bar{L}_1^{-1}L_1 = L_2 - \bar{L}_2\bar{L}_1^{-1}(\bar{L}_1 - (\bar{L}_1 - L_1)) = L_2 - \bar{L}_2 + \bar{L}_2\bar{L}_1^{-1}(\bar{L}_1 - L_1). \tag{5.8}$$

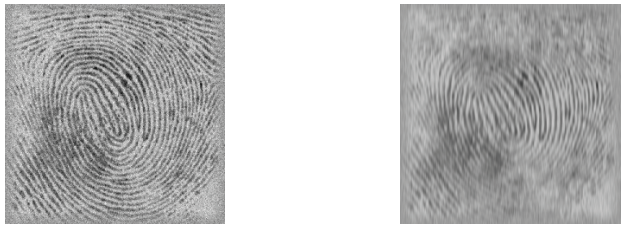
From the inferences above, the entries of $\bar{L}_1 - L_1$ and $\bar{L}_2 - L_2$ are all $O(\varepsilon)$ and $\bar{L}_2\bar{L}_1^{-1} \to L_2L_1^{-1}$ when $\varepsilon \to 0$. Therefore,

$$L_2 - \bar{L}_2\bar{L}_1^{-1}L_1 = O(\varepsilon). \tag{5.9}$$

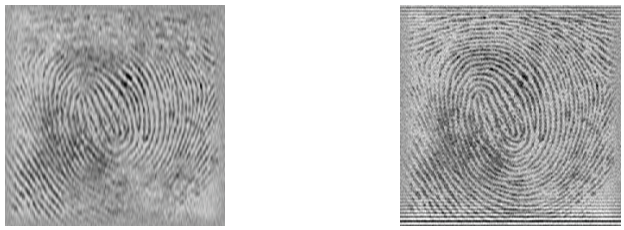
For the entries of $\bar{L}_3$ are $O(\sqrt{\varepsilon})$, by the explicit scheme of inversion of lower triangular

Table 4: Time, primal solution relative error and function relative error for different $N$ in test 2

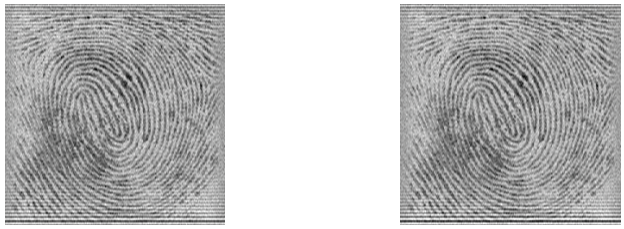
| scenario | N=35 | N=45 | N=55 | N=65 |
|---|---|---|---|---|
| LBreg | 37.30s, 1.29E-1, 4.15E-2 | 35.40s, 1.89E-1, 5.67E-2 | 37.40s, 3.29E-1, 1.18E-1 | 37.30s, 3.0E-1, 1.26E-1 |
| IP-LBreg | 6.66s, 1.11E-6, 2.75E-8 | 4.67s, 1.19E-6, 3.35E-7 | 8.73s, 1.20E-6, 6.12E-12 | 7.98s, 1.29E-6, 4.16E-11 |
| NLBreg | 12.70s, 1.75E-6, 7.05E-9 | 24.50s, 1.67E-6, 7.23E-9 | 29.50s, 1.80E-6, 8.08E-11 | 34.80s, 3.97E-5, 5.05E-9 |
| IPN-LBreg | 0.91s, 1.03E-6, 6.46E-8 | 0.83s, 1.06E-6, 3.60E-10 | 0.99s, 1.12E-6, 9.04E-9 | 0.99s, 7.46E-7, 5.31E-8 |
| RLBreg | 8.94s, 1.63E-6, 1.73E-7 | 8.73s, 1.97E-6, 6.94E-8 | 17.20s, 1.95E-6, 2.17E-7 | 21.10s, 1.85E-6,, 1.47E-8 |
| IPR-LBreg | 1.07s, 4.41E-7, 2.82E-8 | 1.06s, 9.03E-7, 9.18E-8 | 1.19s, 7.03E-7, 1.27E-7 | 1.28s, 1.17E-6, 1.37E-8 |



(a) The original image: fingerprint (b) The blurred image



(c) LBreg recovery t=1.42s SNR=1.98dB (d) P-LBreg recovery t=1.80s SNR=4.71dB



(e) PN-LBreg recovery t=1.21s SNR=5.16dB (f) PR-LBreg recovery t=1.28s SNR=4.75dB

Figure 9: The comparison of different recovery algorithms

matrix, it's easy to get that $(\bar{L}_3^{-1})_{i,j} = O(\varepsilon^{-\frac{1}{2}})$. Then,

$$\bar{L}_3^{-1}(L_2 - \bar{L}_2\bar{L}_1^{-1}L_1)_{i,j} = \sum_{k=1}^{r}(\bar{L}_3^{-1})_{i,k}(L_2 - \bar{L}_2\bar{L}_1^{-1}L_1)_{k,j} = O(\sqrt{\varepsilon}). \tag{5.10}$$

□

## References

[1] J.-F. Cai, E. Candès and Z. Shen, A singular value thresholding algorithm for matrix completion, *SIAM J. Optimiz.* 20 (2010) 1956–1982.

[2] J.-F. Cai, R.H. Chan and Z.W. Shen, A framelet-based image inpainting algorithm, *Appl. Comput. Harmon. Anal.* 24 (2008) 131–149.

[3] J.-F.Cai, S.Osher and S.W.Shen, Linearized Bregman iteration for frame-Based image deblurring, *SIAM J. Imaging Sci.* 2 (2009) 226–252.

[4] J.-F. Cai, S. Osher and Z. Shen, Linearized Bregman iterations for compressed sensing, *Math. Comput.* 78 (2009) 1515–1536.

[5] J.-F. Cai, S. Osher and Z. Shen, Convergence of the linearized Bregman iteration for $\ell_1$-norm minimization, *Math. Comput.* 78 (2009) 2127–2136.

[6] J.-F.Cai, S. Osher and S.W. Shen, Linearized Bregman iteration for frame-Based image deblurring, *SIAM J. Imaging Sci.* 2 (2009) 226–252.

[7] G.H. Golub and C.F. Van Loan, *Matrix computation*, The Johns Hopkins university press, 1992.

[8] S.S. Chen, D.L. Donoho and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM J. Sci. Comput.* 20 (1999) 33–61.

[9] J.H. Demmel, *Applied numerical linear algebra*, Tsinghua university press, Beijing, 2011.

[10] M.P. Friedlander and P. Tseng, Exact regularization of convex programs, *SIAM J. Optimiz.* 18 (2007) 1326–1350.

[11] B. Huang, S.Q. Ma and D. Goldfarb, Accelerated Linearized Bregman Method, *J. Sci. Comput.* 54 (2013) 428–453.

[12] M.J. Lai and W. Yin, Augmented $\ell_1$ and nuclear-norm models with a globally linearly convergent algorithm, *SIAM J. Imaging Sci.* 6 (2013) 1059–1091.

[13] D.A. Lorenz, F. Schopfer and S. Wenger, The linearized Bregman method via split feasibility problems: analysis and generalizations, *arXiv:1309.2094* (2013).

[14] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, Kluwer Academic Publishers, 2004.

[15] Y. Nesterov, *Gradient methods for minimizing composite objective function*, CORE discussion paper, 2007.

[16] B. O′Donoghue and E. Candès, Adaptive restart for accelerated gradient schemes, To appear in *Found. Comput. Math.*

[17] S. Osher, Y. Mao, B. Dong and W. Yin, Fast linearized bregman iteration for compressive sensing and sparse denoising, *Commun. Math. Sci.* 8 (2010) 93–111.

[18] R.T. Rockafellar, *Convex analysis*, Princeton University Press, Princeton, 1970.

[19] F. Schopfer, Exact regularization of polyhedral norms, *SIAM J. Optimiz.* 22 (2012) 1206–1223.

[20] Q. Tran-Dinh and V. Cevher, A Primal-Dual Algorithmic Framework for Constrained Convex Minimization, *arXiv:1406.5403v1* (2014).

[21] W. Yin, Analysis and generalizations of the linearized Bregman method, *SIAM J. Imaging Sci.* 3 (2010) 856–877.

[22] W. Yin, S. Osher, D. Goldfarb and J. Darbon, Bregman iterative algorithms for $\ell_1$-minimization with applications to compressed sensing, *SIAM J. Imaging Sci.* 1 (2008) 143–168.

[23] H. Zhang, J.-F. Cai, L.Z. Cheng and J. Zhu, Strongly convex programming for exact matrix completion and robust principal component analysis, *Inverse Probl. Imag.* 6 (2012) 357–372.

[24] H. Zhang, L.Z. Cheng and W. Zhu, A lower bound guaranteeing exact matrix completion via singular value thresholding algorithm, *Appl. Comput. Harmon. Anal.* 31 (2011) 454–459.

[25] H. Zhang and W. Yin, Gradient methods for convex minimization: better rates under weaker conditions, *UCLA CAM Report* (2013).

[26] H. Zhang and W. Yin, A dual algorithm for a class of augmented convex models, *UCLA CAM Report* (2013).

---

*Manuscript received 22 October 2013*
*revised 7 March 2014, 12 August 2014*
*accepted for publication 14 August 2014*

Tao Sun
College of Science, National University of Defense Technology
Changsha, 410073, Hunan, China
E-mail address: nudtsuntao@163.com

Hui Zhang
College of Science, National University of Defense Technology
Changsha, 410073, Hunan, China
E-mail address: hhuuii.zhang@gmail.com

Lizhi Cheng
The State Key Laboratory for High Performance Computation
National University of Defense Technology
Changsha, 410073, Hunan, China
E-mail address: clzcheng@nudt.edu.cn