Inverse Problems and Imaging
Volume 3, No. 3, 2009, 487–503
doi:10.3934/ipi.2009.3.487

COORDINATE DESCENT OPTIMIZATION FOR ℓ^1 MINIMIZATION WITH APPLICATION TO COMPRESSED SENSING; A GREEDY ALGORITHM

Yingying Li and Stanley Osher

UCLA Mathematics Department
Box 951555
Los Angeles, CA 90095-1555, USA

(Communicated by Zuowei Shen)

Abstract. We propose a fast algorithm for solving the Basis Pursuit problem, $\min_u\{|u|_1 : Au = f\}$, which has application to compressed sensing. We design an efficient method for solving the related unconstrained problem $\min_u E(u) = |u|_1 + \lambda\|Au - f\|_2^2$ based on a greedy coordinate descent method. We claim that in combination with a Bregman iterative method, our algorithm will achieve a solution with speed and accuracy competitive with some of the leading methods for the basis pursuit problem.

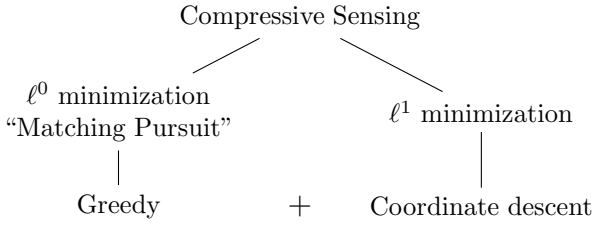
1. **Introduction.** We wish to solve the compressed sensing (CS) problem [3, 12, 23, 24]

$$\min_{u\in\mathbb{R}^n} |u|_0 \quad \text{subject to } Au = f. \tag{1}$$

A breakthrough for this NP problem is that when A satisfies certain conditions, then the ℓ^0 problem (1) is equivalent to

$$\min_{u\in\mathbb{R}^n} |u|_1 \quad \text{subject to } Au = f, \tag{2}$$

see [5, 8, 9, 10]. Thus there are two main approaches in solving (1). One is to approach (1) directly using greedy selection algorithms [17, 14, 11, 22]; the other is through ℓ^1 minimization methods, like coordinate descent, also [16, 4, 13]. The ℓ^0 approach is faster but unreliable: since (1) is nonconvex, there is no guarantee of existence and it is difficult to find global minimizers (and not local minimizers). The ℓ^1 approach is often slower but guaranteed to find a global minimizer for certain kinds of matrices A and is stable to noise.



2000 *Mathematics Subject Classification.* Primary: 90C25; Secondary: 65K05.
Key words and phrases. Basis Pursuit, shrinkage, greedy sweep, Bregman iteration, constrained problem.
This research was supported by ONR Grant N00014710810 and the Department of Defense.

To exploit both of their advantages, we propose a greedy coordinate descent method to solve the ℓ^1 minimization problem. In [28], the authors use the steepest descent method to find the updating coordinate which makes the most negative gradient. This is also a greedy coordinate descent method. Our paper also uses greedy selection to choose the coordinate, but uses to another criterion to optimize.

We use greedy coordinate descent for solving (3) to solve the Basis Pursuit problem [6], which also arises in compressed sensing. Recently, many efficient algorithms have been developed for solving this. These include the linearized Bregman method [29, 18] and the fixed point continuation method (FPC) [16] together with a Bregman iterative method [29, 1]. Our proposed method solves (2) by solving a sequence of subproblems (3), which is the same as in [29]. We show that our proposed method is competitive in speed and accuracy for an important class of matrices A.

1.1. **Coordinate descent.** In multivariable minimization, *coordinate descent* methods minimize the objective by solving a sequence of scalar minimization subproblems. Each subproblem improves the estimate of the solution by minimizing along a selected coordinate with all other coordinates fixed. Coordinate descent is attractive because it is simple: scalar minimization is easier than multivariable minimization.

Coordinate descent is efficient when the subproblems can be solved quickly. For some applications, the subproblem solutions can be expressed in closed form. This is the case for the lasso problem

$$\min_{u\in\mathbb{R}^n} E(u) = |u|_1 + \lambda\|Au - f\|_2^2 \tag{3}$$

where $f \in \mathbb{R}^m$ and A is an $m \times n$ matrix with $m < n$ (the matrix is wide). The solution of the coordinate subproblems involves a shrinkage [15, 28]. There are other fast algorithms for solving the lasso problem, see [6, 7, 19, 20, 21].

Generally, each coordinate must be visited several times to reach a minimum. The order in which coordinates are visited is called the *sweep pattern*. In many applications, the sweep pattern is sequential where the coordinates are visited in order—this is called *pathwise* coordinate descent [15] or cyclic coordinate descent. But for some problems, the choice of sweep pattern has a significant effect on the rate of convergence. The the essential feature in this work is coordinate descent with an adaptive choice of sweep pattern, which tends to converge faster than sequential or random sweep order.

1.2. **Outline.** This work is structured as follows. In §2 we introduce the pathwise coordinate descent method [15] for solving (3). In §2.3, we describe our refined method with a better sweep pattern for solving the same problem and compare computational costs in §2.5. In §3, we solve (2) by using a Bregman iterative method, then discuss numerical performance. We give a denoising example in §3.3 and an example including huge dynamic range in §3.4. In §4, we briefly introduce two other fast methods and compare them with the proposed method. Finally, we give the conclusion in §5.

2. Solving the unconstrained problem.

2.1. **The coordinate subproblem.** The coordinate descent method optimizes the objective function through a sequence of one-dimensional optimizations. The

method is essentially the analog of the Gauss-Seidel matrix algorithm for optimization. We can apply this strategy to

$$\min_{u\in\mathbb{R}^n} E(u) = |u|_{\ell^1} + \lambda\|Au - f\|_{\ell^2}^2, \tag{4}$$

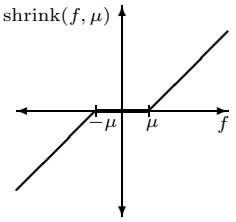
where $f \in \mathbb{R}^m$, λ is a scalar parameter and A is a $m \times n$ matrix with $m < n$ (the matrix is wide). It is easy to see that the objective function $E(u)$ is convex, so a local minimum is a global minimum. For (4), there is a simple closed formula for the solution of each coordinate subproblem, which makes the coordinate descent method efficient for solving it.

The intuition for using coordinate descent comes from the simple solution of the same problem in one dimension,

$$\min_{x\in\mathbb{R}} E(x) = |x| + \lambda(x - f)^2.$$

The solution is: $x = \text{shrink}(f, \frac{1}{2\lambda})$, where the shrink operator is

$$\text{shrink}(f,\mu) = \begin{cases} f-\mu, & \text{if} \quad f > \mu; \\ 0, & \text{if} \quad -\mu \le f \le \mu; \\ f+\mu, & \text{if} \quad f < -\mu. \end{cases}$$



For each coordinate subproblem, we freeze all components of u except the jth component u_j. Let a_j denote the jth column of A, a_{ij} the element of A in the ith row and jth column, and f_i the ith component of f, the problem is to minimize

$$\begin{aligned}\min_{u_j} E(u) &= \lambda\sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_{ij}u_j - f_i\right)^2 + \sum_{i=1}^{n}|u_i| \\ &= \lambda\big(\|a_j\|_2^2 u_j^2 - 2\beta_j u_j + \|f\|_2^2\big) + |u_j| + \sum_{i\neq j}|u_i|,\end{aligned}$$

where $\beta_j = \sum_i a_{ij}(f_i - \sum_{k\neq j} a_{ik}u_k)$. The optimal value for u_j with all the other coordinates fixed is

$$\tilde{u}_j = \frac{1}{\|a_j\|_2^2}\,\text{shrink}\left(\beta_j, \tfrac{1}{2\lambda}\right). \tag{5}$$

Here, we assume $\|a_j\|_2^2 \neq 0$; otherwise $a_{ij} = 0$ for any i. Then we can delete the whole jth column of A since the value of u_j does not affect f at all and u_j has to be zero to get the ℓ^1 minimum. So this formula (5) corrects the jth component of u, which decreases the energy function $E(u)$.

2.2. **Pathwise coordinate descent.** Coordinate descent applied with the sequential sweep order is called pathwise coordinate descent [15].

Algorithm (*Pathwise Coordinate Descent*):

While "not converge"
  For $i = 1, 2, \ldots, n$,
    $\beta_j \leftarrow \sum_i a_{ij}(f_i - \sum_{k\neq j} a_{ik}u_k)$
    $u_j \leftarrow \frac{1}{\|a_j\|_2^2}\,\text{shrink}\left(\beta_j, \frac{1}{2\lambda}\right)$

2.3. **Adaptive and relatively greedy sweeping.** For the application of compressed sensing [3, 12, 22], we ultimately seek a sparse signal. But sequential sweeps also change the coordinates which should be zero during the sweep; the process of finding the solution does not give preference to sparsity, see Figure 1. After one sequential sweep (512 iterations), almost every component becomes nonzero (circles in the left plot and the middle plot for the zoom-in version), but the exact signal is zero for most of the components (dots). To exploit the sparsity of the solution, we propose an adaptive sweep. Instead of proceeding through all the coordinates sequentially, we choose the coordinate that decreases the energy the most. The plot on the right side shows the numerical performance of using the same number of adaptive sweeps. We see that the result in the right figure keeps more sparsity than the result in the left figure.

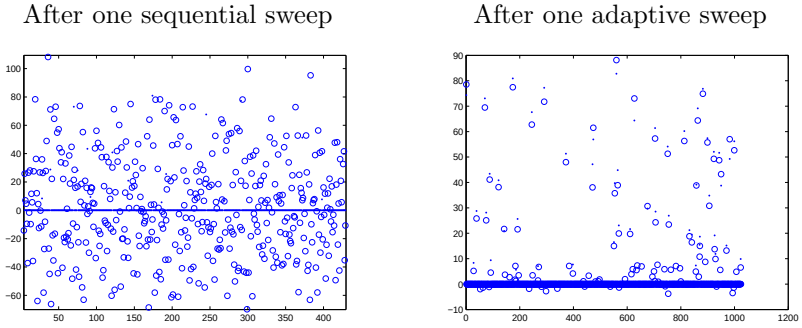


FIGURE 1. The dots represent the exact solution of (4) at $\lambda = 1000$ and the circles are the results after 512 iterations. *Left:* the result after one sequential sweep (512 iterations). *Right:* after 512 adaptive steps.

The formula (5) provides the best choice of the jth coordinate. Therefore, the energy decrease obtained in updating the jth coordinate is

$$\begin{aligned}\Delta E_j &= E(u_1,\dots,u_j,\dots,u_n) - E(u_1,\dots,\tilde{u}_j,\dots,u_n)\\ &= \lambda\|a_j\|_2^2\left(u_j - \frac{\beta_j}{\|a_j\|_2^2}\right)^2 + |u_j| - \lambda\|a_j\|_2^2\left(\tilde{u}_j - \frac{\beta_j}{\|a_j\|_2^2}\right)^2 - |\tilde{u}_j|\\ &= \lambda\|a_j\|_2^2(u_j-\tilde{u}_j)\left(u_j+\tilde{u}_j - \frac{2\beta_j}{\|a_j\|_2^2}\right) + |u_j| - |\tilde{u}_j|. \qquad (6)\end{aligned}$$

We select the coordinate that produces the largest decrease in energy,

$$j^* = \arg\max_j \Delta E_j.$$

Previous work has considered other ways to choose the updated coordinate. In [28], the authors choose the updated coordinate according to the most negative directional derivative along each coordinate in the forward or backward directions. Denoting e_k as the kth standard basis vector, the objective function $E(u)$ has directional derivatives

$$d_{e_k}E = \lim_{\sigma\downarrow 0}\frac{E(u+\sigma e_k)-E(u)}{\sigma} = 2\lambda(\|a_k\|_2^2 u_k - \beta_k) + \begin{cases} 1, & u_k \geq 0,\\ -1, & u_k < 0,\end{cases}$$

$$d_{-e_k}E = \lim_{\sigma\downarrow 0}\frac{E(u-\sigma e_k)-E(u)}{\sigma} = -2\lambda(\|a_k\|_2^2 u_k - \beta_k) + \begin{cases} -1, & u_k > 0, \\ 1, & u_k \le 0. \end{cases}$$

The updated coordinate is then selected as

$$j^* = \arg\min_j\big(\min\{d_{e_j}E, d_{-e_j}E\}\big). \tag{7}$$

However, this selection is not very cheap. Our selection strategy is somewhat cheaper for the compressed sensing problem.

A computational savings is obtained through updating β_j, rather than entirely recomputing it through every iteration. We rewrite β_j as

$$\beta_j = (A^T f)_j - a_j^T Au + (A^T A)_{j,j}u_j = (A^T f)_j - a_j^T Au + \|a_j\|_2^2 u_j. \tag{8}$$

then

$$\begin{aligned}\beta_j^{k+1} - \beta_j^k &= (A^T f)_j - a_j^T Au^{k+1} + \|a_j\|_2^2 u_j^{k+1} - \big((A^T f)_j - a_j^T Au^k + \|a_j\|_2^2 u_j^k\big) \\ &= a_j^T A(u^k - u^{k+1}) + \|a_j\|_2^2(u_j^{k+1} - u_j^k).\end{aligned}$$

Suppose we choose the p^{th} coordinate to update in the k^{th} iteration, then $u^{k+1}-u^k$ is nonzero only in the p^{th} coordinate, so $u^{k+1}-u^k = (u_p^{k+1}-u_p^k)e_p$, where e_p is the p^{th} standard basis vector.

$$\begin{aligned}\beta_j^{k+1} - \beta_j^k &= (u_p^k - u_p^{k+1})a_j^T Ae_p + \|a_j\|_2^2(u_j^{k+1} - u_j^k) \\ &= \big((A^T A)_{j,p} - \delta_{j,p}\|a_j\|_2^2\big)(u_p^k - u_p^{k+1}) \\ &= \begin{cases} (A^T A)_{j,p}(u_p^k - u_p^{k+1}) & j \ne p \\ 0 & j = p. \end{cases}\end{aligned}$$

and

$$\begin{aligned}\beta^{k+1} - \beta^k &= (u_p^k - u_p^{k+1})A^T Ae_p + \delta_{j,p}\|a_p\|_2^2(u_p^{k+1} - u_p^k) \\ &= (u_p^{k+1} - u_p^k)(\|a_p\|_2^2 I - A^T A)e_p.\end{aligned}$$

Therefore, we get the algorithm for using an adaptive sweep,

Precompute: $w_j = \|a_j\|_2^2$;
Initialization: $u = 0$, $\beta = A^\top f$;
Iterate until converge:
$\tilde{u} = \text{shrink}(\frac{\beta}{w_j}, \frac{1}{2\lambda w_j})$;
$j = \arg\max_i \Delta E_i$ or $j = \arg\min_i \{d_{e_i}E, d_{-e_i}E\}$,
then $u_i^{k+1} = u_i^k$, $i \ne j$,
$u_j^{k+1} = \tilde{u}_j$;
$\beta^{k+1} = \beta^k - w_j|u_j^k - \tilde{u}_j|\,(A^\top A)e_j$,
$\beta_j^{k+1} = \beta_j^k$.

If A is a column normalized matrix, then the computation simplifies to $w_i = 1$ and

$$\Delta E_j = \lambda(u_j - \tilde{u}_j)(u_j + \tilde{u}_j - 2\beta_j) + |u_j| - |\tilde{u}_j|, \tag{9}$$

To simplify the computation even further, we observe that a large difference $|u_j - \tilde{u}_j|$ indicates that changing u_j to $\tilde{u}_j$ probably yields a large decrease in E. So we use the following simplification for choosing the updating component based on the relative error:

$$\Delta E_j \approx \lambda|u_j - \tilde{u}_j|. \tag{10}$$

We call this a relatively greedy choice, because we have essentially removed the absolute magnitudes of u_j and $\tilde{u}_j$ from the decision process. This simplification makes the computation easier than computing the exact ΔE. We have two adaptive sweeps (9) and (10) to process coordinate descent for solving (3). The first updates the coordinate which makes the energy decrease the most, the second chooses the coordinate that produces the biggest difference $|u_j - \tilde{u}_j|$. For the compressed sensing problem, numerical experience suggests that they both need a similar number of iterations to get the same accuracy but the latter (10) takes less time compared to the other two ways of greedy selections, (7) and (9). For simplification, we shall use column normalized matrices for all the numerical tests in this paper and the latter sweep in the coordinate method, which is called a refined sweep. If we do not normalize A initially, then we can approximate ΔE by $\|a_j\|^2|u_j^k - \tilde{u}_j|$. If we precompute $B = A^\top A$, then $(A^\top A)e_j$ is the jth column of B in the algorithm.

Algorithm (*Coordinate descent with a refined sweep*):

Precompute: $w_j = \|a_j\|_2^2$;
Normalization: $A(\cdot, i) = A(\cdot, i)/w_i$;
Initialization: $u = 0$, $\beta = A^\top f$;
Iterate until converge:
$\tilde{u} = \text{shrink}(\beta, \frac{1}{2\lambda})$;
$j = \arg\max_i |u_i - \tilde{u}_i|$,
then $u_i^{k+1} = u_i^k$, $i \neq j$,
$u_j^{k+1} = \tilde{u}_j$;
$\beta^{k+1} = \beta^k - |u_j^k - \tilde{u}_j|(A^\top A)e_j$,
$\beta_j^{k+1} = \beta_j^k$.

2.4. **Convergence of greedy coordinate descent.** Consider minimizing functions of the form

$$\min_{u_i \in \mathbb{R}} H(u_1, u_2, \ldots, u_n) = F(u_1, u_2, \ldots, u_n) + \sum_{i=1}^{n} g_i(u_i), \tag{11}$$

where F is differentiable and convex, and g_i is convex. Tseng (1988) proved pathwise coordinate descent converges to a minimizer of H [25, 26]. Here, with several additional conditions on H, the following theorem shows that our greedy coordinate descent method also converges to a minimizer of H.

Theorem 2.1. *Suppose $F(u_1, u_2, \ldots, u_n)$ is smooth and convex, $|\frac{\partial^2 F}{\partial u_i \partial u_j}|_\infty \leq M$, and H is strictly convex with respect to any one variable u_i, then the statement that $u = (u_1, u_2, \ldots, u_n)$ is an optimal solution of (11) is equivalent to the statement that every component u_i is an optimal solution of H with respect to variable u_i for any i.*

Proof. Suppose $u = (u_1, u_2, \ldots, u_n)$ satisfies the condition that every component u_i is an optimal solution of H with respect to variable u_i. That is,

$$H(u_1, u_2, \ldots, u_i + \epsilon_i, \ldots, u_n) - H(u_1, u_2, \ldots, u_i, \ldots, u_n) \geq 0, \text{ for any } \epsilon_i.$$

Since H is strictly convex along any one coordinate, u_i is the unique solution of $\min_{u_i} H$. Therefore,

$$H(u_1, u_2, \ldots, u_i + \epsilon_i, \ldots, u_n) - H(u_1, u_2, \ldots, u_i, \ldots, u_n) > 0, \text{ for any } \epsilon_i \neq 0. \tag{12}$$

From (12), we obtain by the mean value theorem

$$\begin{aligned}
&H(u_1, u_2, \ldots, u_i + \epsilon_i, \ldots, u_n) - H(u_1, u_2, \ldots, u_i, \ldots, u_n) \\
&= F(u_1, u_2, \ldots, u_i + \epsilon_i, \ldots, u_n) - F(u_1, u_2, \ldots, u_i, \ldots, u_n) + g_i(u_i + \epsilon_i) - g_i(u_i) \\
&= \epsilon_i \frac{\partial F}{\partial u_i}(u) + \frac{\epsilon_i^2}{2} \frac{\partial^2 F}{\partial {u_i}^2}(\tilde{u}) + g_i(u_i + \epsilon_i) - g_i(u_i) > 0.
\end{aligned}$$

for some $\tilde{u}$ between u_i and $u_i + \epsilon_i$. When $\epsilon_i > 0$, then

$$\frac{\partial F}{\partial u_i}(u) + \frac{\epsilon_i}{2} \frac{\partial^2 F}{\partial {u_i}^2}(\tilde{u}) + \frac{g_i(u_i + \epsilon_i) - g_i(u_i)}{\epsilon_i} > 0$$

Now we want to prove

$$\frac{\partial F}{\partial u_i}(u) + \frac{g_i(u_i + \epsilon_i) - g_i(u_i)}{\epsilon_i} \geq 0, \quad \text{for any} \quad \epsilon_i > 0.$$

If it is not true, there is an ϵ_i such that

$$\frac{\partial F}{\partial u_i}(u) + \frac{g_i(u_i + \epsilon_i) - g_i(u_i)}{\epsilon_i} = c < 0.$$

Since g_i is convex, we have

$$\frac{g_i(u_i + \eta_i) - g_i(u_i)}{\eta_i} \leq \frac{g_i(u_i + \epsilon_i) - g_i(u_i)}{\epsilon_i} \quad \text{for} \quad 0 < \eta_i \leq \epsilon_i.$$

Therefore,

$$\begin{aligned}
&\frac{\partial F}{\partial u_i}(u) + \frac{\eta_i}{2} \frac{\partial^2 F}{\partial {u_i}^2}(\tilde{u}) + \frac{g_i(u_i + \eta_i) - g_i(u_i)}{\eta_i} \\
&\leq \frac{\partial F}{\partial u_i}(u) + \frac{\eta_i}{2} \frac{\partial^2 F}{\partial {u_i}^2}(\tilde{u}) + \frac{g_i(u_i + \epsilon_i) - g_i(u_i)}{\epsilon_i} \\
&= c + \frac{\eta_i}{2} \frac{\partial^2 F}{\partial {u_i}^2}(\tilde{u}) \quad \text{for} \quad 0 < \eta_i \leq \epsilon_i.
\end{aligned}$$

Let $\eta_i = -\frac{c}{M}$, then $|\frac{\eta_i}{2} \frac{\partial^2 F}{\partial {u_i}^2}(\tilde{u})| \leq \frac{\eta_i}{2} M = -\frac{c}{2}$, which means

$$\frac{\partial F}{\partial u_i}(u) + \frac{\eta_i}{2} \frac{\partial^2 F}{\partial {u_i}^2}(\tilde{u}) + \frac{g_i(u_i + \eta_i) - g_i(u_i)}{\eta_i} \leq \frac{c}{2} < 0,$$

which is a contradiction. Therefore,

$$\frac{\partial F}{\partial u_i}(u) + \frac{g_i(u_i + \epsilon_i) - g_i(u_i)}{\epsilon_i} \geq 0 \quad \text{for any} \quad \epsilon_i > 0.$$

The analogous result for $\epsilon_i < 0$ can be obtained similarly,

$$\frac{\partial F}{\partial u_i}(u) + \frac{g_i(u_i + \epsilon_i) - g_i(u_i)}{\epsilon_i} \leq 0 \quad \text{for any} \quad \epsilon_i \leq 0.$$

In conclusion, we have

$$\epsilon_i \frac{\partial F}{\partial u_i}(u) + g_i(u_i + \epsilon_i) - g_i(u_i) \geq 0 \quad \text{for any} \quad \epsilon_i.$$

Therefore, by the convexity of F,

$$\begin{aligned}
&H(u_1+\epsilon_1,u_2+\epsilon_2,\dots,u_n+\epsilon_n)-H(u_1,u_2,\dots,u_n)\\
&=F(u_1+\epsilon_1,u_2+\epsilon_2,\dots,u_n+\epsilon_n)-F(u_1,u_2,\dots,u_n)+\sum_i\big(g_i(u_i+\epsilon_i)-g_i(u_i)\big)\\
&\geq\sum_i\epsilon_i\frac{\partial F}{\partial u_i}(u)+\sum_i\big(g_i(u_i+\epsilon_i)-g_i(u_i)\big)\\
&\geq 0.
\end{aligned}$$

□

Now checking the objective function E in (4), $F=\lambda\|Au-f\|_2^2$ is smooth and convex, $\frac{\partial^2 F}{\partial u_i\partial u_j}=(A^\top A)_{ij}$ is bounded, E with respect to any single variable u_i has the form $\lambda(u_i-b)^2+|u_i|_1$, which is strictly convex for any value of b and $\lambda\neq 0$. Thus, E satisfies all the conditions in the theorem.

Lemma 2.2. *Denote* $v=(u_1,u_2,\dots,u_{i-1},u_{i+1},\dots,u_n)$ *and*

$$f(v)=\arg\min_{u_i}H(u_1,u_2,\dots,u_i,\dots,u_n),$$

then f *is continuous for any* i.

Proof. Suppose there is a sequence (v^k) that converges to v, we want to prove that $f(v^k)$ converges to $f(v)$. Denote $\tilde{u}_i^k=f(v^k)$, $\tilde{u}_i=f(v)$ and $\tilde{H}(v,u_i)=H(u_1,u_2,\dots,u_i,\dots,u_n)$. Since $H(v^k,\tilde{u}_i^k)\leq H(v^k,\tilde{u}_i)$ and $\lim_k H(v^k,\tilde{u}_i)=H(v,\tilde{u}_i)$, we have for some M,

$$|H(v^k,\tilde{u}_i^k)|\leq|H(v,\hat{u}_i)|+M\qquad\text{for all } k.$$

Then, since H has bounded level sets, there exists T such that $\left\|(v^k,\tilde{u}_i^k)\right\|_\infty\leq T$. Therefore, $(\tilde{u}_i^k)$ has a convergent subsequence $(\tilde{u}_i^{p_k})$, $\tilde{u}_i^{p_k}\to\hat{u}_i$. Then

$$\begin{aligned}
H(v,\hat{u}_i)&=\lim_{k\to\infty}H(v^{p_k},\hat{u}_i^{p_k})\\
&\leq\lim_{k\to\infty}H(v^{p_k},\tilde{u}_i)=H(v,\tilde{u}_i),
\end{aligned}$$

but $\arg\min_u H(v,u)$ is unique, so $\hat{u}_i=\tilde{u}_i$. Therefore, every cluster point of the bounded sequence $(\tilde{u}_i^k)$ converges to $\tilde{u}_i$. Then $(\tilde{u}_i^k)$ converges to $\tilde{u}_i$, so f is continuous. □

Theorem 2.3. *If* $\lim_{u_i\to\infty}H(u_1,u_2,\dots,u_n)=\infty$ *for any* i*, then the greedy coordinate descent method based on the selection rule:* $j=\arg\max_i\Delta H_i$ *converges to an optimal solution of* (11).

Proof. It is easy to get

$$\min_{u\in\mathbb{R}^p}H(u)\leq H(u^{p+1})\leq H(u^p).\tag{13}$$

$H(u^p)$ is nonincreasing and bounded from below, so it converges to some $\hat{H}$,

$$\lim_{p\to\infty}H(u^p)=\hat{H}.\tag{14}$$

Since $\lim_{u_i\to\infty}H(u)=\infty$, we have $H(u^p)\leq M$ implies $\sup\|u^p\|_\infty$ is bounded (the function H has bounded level sets). Therefore, the sequence (u^p) is also bounded, which implies there exists a subsequence $(u^{p(k)})$ converging to a limit $\hat{u}$. Since H is continuous, $H(\hat{u})=\hat{H}$.

Define

$$\Delta H_i(u) = H(u_1, \ldots, u_i, \ldots, u_n) - H(u_1, \ldots, \tilde{u}_i, \ldots, u_n),$$

where $\tilde{u}_i = \arg\min_{u_i} H(u)$ is the optimal solution of H with respect to variable u_i with all the other $u_j\,(i \neq j)$ fixed. Using the lemma, $\tilde{u}_i$ is continuous on $v = (u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_n)$, so $\Delta H_i(u) = H(u_1, \ldots, u_i, \ldots, u_n) - H(u_1, \ldots, \tilde{u}_i, \ldots, u_n)$ is continuous.

$$\Delta H_i(\hat{u}) = \Delta H_i(\lim_{k\to\infty} u^{p(k)}) = \lim_{k\to\infty} \Delta H_i(u^{p(k)}).$$

For the sweep pattern $j = \arg\max_i \Delta E$ for choosing the updated coordinate,

$$\Delta H_i(u^{p(k)}) \leq H(u^{p(k)}) - H(u^{p(k)+1}) = 0$$

and

$$\Delta H_i(\hat{u}) = \lim_{k\to\infty} \Delta H_i(u^{p(k)}) \leq \lim_{k\to\infty} H(u^{p(k)}) - H(u^{p(k)+1}) = 0.$$

According to Theorem 2.1, $\hat{u}$ is an optimal solution to (11). □

Remark 1. For our problem, $H(u) = |u|_1 + \lambda\|Au - f\|_2^2$, so $\lim_{|u_i|\to\infty} H(u) = \infty$ for any i. So the greedy coordinate descent method works here. Also in this case, $\tilde{u}_i = f(v)$ is obtained by a shrinkage, which is clearly continuous.

We can prove the convergence of coordinate descent when using the sweep pattern $j = \arg\max_i |u_i - \tilde{u}_i|$ for solving $H(u) = |u|_1 + \lambda\|Au - f\|_2^2$.

Lemma 2.4. *Consider function $H(x) = (ax - b)^2 + |x|$, where $a > 0$. It is easy to know that $\tilde{x} = \arg\min_x H(x) = \text{shrink}(\frac{b}{a}, \frac{1}{2a^2})$. Now we claim that for any x,*

$$|x - \tilde{x}| \leq \frac{1}{a}\big(H(x) - H(\tilde{x})\big)^{1/2}.$$

Proof. Assume that $\frac{b}{a} \geq 0$, then $\tilde{x} = \arg\min_x H(x) = \text{shrink}(\frac{b}{a}, \frac{1}{2a^2}) \geq 0$.

If $\tilde{x} = 0$, then $\frac{b}{a} \leq \frac{1}{2a^2}$ so $2ab \leq 1$.

$$\begin{aligned} H(x) - H(\tilde{x}) &= a^2x^2 + |x| - 2abx \\ &\geq a^2x^2 + (1 - 2ab)|x| \\ &\geq a^2x^2 \\ &= a^2(x - \tilde{x})^2. \end{aligned}$$

If $\tilde{x} > 0$, then $\tilde{x} = \frac{b}{a} - \frac{1}{2a^2}$ so $b = a\tilde{x} + \frac{1}{2a}$. We have

$$\begin{aligned} H(x) - H(\tilde{x}) &= (ax - b)^2 + |x| - (a\tilde{x} - b)^2 - \tilde{x} \\ &\geq (ax - b)^2 + x - (a\tilde{x} - b)^2 - \tilde{x} \\ &= a(x - \tilde{x})(ax + a\tilde{x} - 2b + \tfrac{1}{a}) \\ &= a(x - \tilde{x})(ax - a\tilde{x} - \tfrac{1}{2a} + a\tilde{x} - a\tilde{x} - \tfrac{1}{2a} + \tfrac{1}{a}) \\ &= a^2(x - \tilde{x})^2. \end{aligned}$$

When $\frac{b}{a} \leq 0$, the analysis is the same. □

Theorem 2.5. *The greedy coordinate descent method with the selection rule: $j = \arg\max_i |u_i - \tilde{u}_i|$ converges to an optimal solution of*

$$\min_{u\in\mathbb{R}^n} H(u) = |u|_{\ell^1} + \lambda\|Au - f\|_{\ell^2}^2, \tag{15}$$

Proof. We know that the function H restricted on the ith coordinate is a function like $\lambda\|a_i\|_2^2(x-b)^2+|x|$, where a_i is the ith column of A, so by Lemma 2

$$|u_i-\tilde{u}_i|\le\frac{(H(v,u_i)-H(v,\tilde{u}_i))^{1/2}}{\|a_i\|_2},\quad \textit{for any } v.$$

We have $\lim_{k\to\infty}|u^k-u^{k+1}|_\infty\le\lim_{k\to\infty}\frac{(H(u^k)-H(u^{k+1}))^{1/2}}{\min_j\|a_j\|_2}=0$, so (u^k) is a Cauchy sequence and converges to $\hat{u}$. Denote $u^k=(v_i^k,u_i^k)$ and $\hat{u}=(v_i,u_i)$, then $\lim(v_i^k,u_i^k)=(\hat{v}_i,\hat{u}_i)$. Let

$$\begin{aligned}\tilde{u}_i=f_i(v_i)&=\arg\min_{u_i}H(v_i,u_i)\\&=\frac{1}{\|a_i\|_2^2}\operatorname{shrink}\Big(\sum_j a_{ji}(f_j-\textstyle\sum_{k\neq i}a_{jk}u_k),\frac{1}{2\lambda}\Big),\end{aligned}$$

Since shrinkage is continuous, f_i is continuous and $f_i(v_i^k)\to f_i(\hat{v}_i)$. Denote $\tilde{u}_i^k=f_i(v_i^k)$, then

$$\begin{aligned}|\hat{u}_i-f_i(\hat{v}_i)|&=\lim_{k\to\infty}|u_i^k-f_i(v_i^k)|=\lim_{k\to\infty}|u_i^k-\tilde{u}_i^k|\\&\le\lim_{k\to\infty}\max_j|u_j^k-\tilde{u}_j^k|\\&=\lim_{k\to\infty}\|u^k-u^{k+1}\|_\infty\\&\le\lim_{k\to\infty}\frac{(H(u^k)-H(u^{k+1}))^{1/2}}{\min_j\|a_j\|_2}=0\qquad\text{for any } i.\end{aligned}$$

So $\hat{u}_i=f_i(\hat{v}_i)$. This means $\hat{u}$ is optimal in each coordinate. By Theorem 2.1, $\hat{u}$ is optimal. □

2.5. **Comparison of numerical speed.** Table 1 compares the runtimes using the pathwise coordinate descent and the refined coordinate descent for solving

$$\min_{u\in\mathbb{R}^n}|u|_1+\lambda\|Au-f\|_2^2. \tag{16}$$

In this experiment, we form a 256×512 matrix A by choosing elements uniformly distributed in $[0,1]$ and then normalizing columns, f by a 512-sample input signal with 26 uniformly distributed nonzero components. The stopping conditions for two different methods are almost the same, which means the relative residuals and the relative errors obtained from the two algorithms are almost the same.

λ	Pathwise	(7)	(9)	(10)
0.1	43	0.33	0.24	0.18
0.5	42	0.31	0.22	0.17
1	49	0.26	0.19	0.16
10	242	0.56	0.42	0.33
100	1617	4.0	2.7	2.2

TABLE 1. Runtime in seconds of pathwise coordinate descent and adaptive sweep methods (7), (9), (10) for solving (16) when they achieve the same accuracy.

From the table, we see that the proposed method (10) is much faster than pathwise coordinate descent for any parameter λ. The proposed method is also faster than the other two adaptive sweep methods included by (7) and by (9).

All methods slow down with increasing λ. We need large λ to approximate the constrained problem $\min\{|u|_{\ell^1},\ \text{s.t. } Au = f\}$. Unfortunately, when λ is very big, the threshold is $\frac{1}{\lambda}$ and consequently advances in the coordinate descent are small and the convergence is slow. However, applying Bregman iteration, the methods speed up considerably for moderate λ and converge rapidly to a solution of the constrained problem. We will discuss this in detail in §3.2 and §3.3.

3. Solving the constrained problem.

3.1. The unconstrained problem and the constrained problem.

We consider the unconstrained problem

$$\min_{u\in\mathbb{R}^n} |u|_1 + \lambda\|Au - f\|_2^2 \tag{17}$$

and the constrained problem

$$\min_{u} |u|_1 \quad \text{s.t. } Au = f. \tag{18}$$

Table 2 shows how increasing λ in (17) affects speed and accuracy. We use the refined coordinate descent method and compare the relative residuals $\|Au - f\|_2 / \|f\|_2$ and errors $\|u - u_{\text{exact}}\|_2 / \|u_{\text{exact}}\|_2$. Here, A is a 256×512 column normalized matrix uniformly distributed in $[0, 1]$, and the sparse solution to (18) is a 512-sample input signal with 26 nonzero components. We use the stopping time $\|u^k - u^{k-1}\|_{\ell^\infty} \leq 1\times10^{-6}$.

λ	Relative Residual	Relative Error	Runtime (s)
0.1	8.5×10^{-3}	2.3×10^{-2}	0.18
0.5	1.7×10^{-3}	4.7×10^{-2}	0.17
1	8.5×10^{-4}	2.3×10^{-3}	0.16
10	8.5×10^{-5}	2.3×10^{-4}	0.33
10^2	8.6×10^{-6}	2.3×10^{-5}	1.26
10^4	5.4×10^{-7}	1.0×10^{-6}	113.00
10^8	5.7×10^{-8}	$\mathbf{8.0 \times 10^{-2}}$	0.87

TABLE 2. The relative residuals and the relative errors for different λ using the refined method.

From the table, we see that the relative residual and the relative error goes to zero as λ increases, which implies the solution of (17) converges to the solution of (18). Figure 2 shows the convergence graphically.

By choosing large λ, the unconstrained problem (17) has more weight in the penalty term and its solution approximates the solution of (18), see Figure 2. However, when λ is very big, the algorithm needs more accuracy and the algorithm converges slowly to the sparse solution of the constrained problem (18). From Table 2, we see that it takes much longer time when $\lambda = 10^4$ and when $\lambda = 10^8$ the relative error is surprisingly bad even through the relative residual is very small. In Figure 2, we see that the numerical result and the exact solution are very close to each other when $\lambda = 100$.

If we want more accuracy, we have to use bigger λ, which will take much more computation. To overcome this inefficiency, we use a Bregman iterative method.

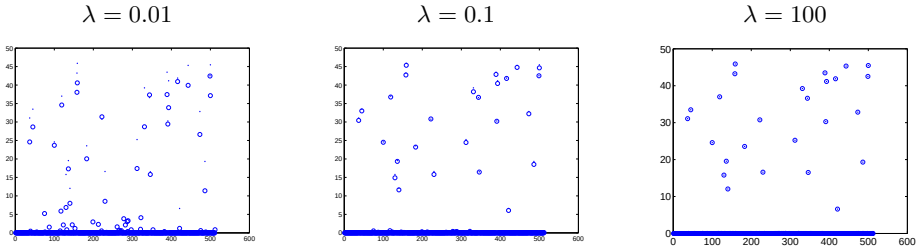


FIGURE 2. The circles are the numerical outputs at a certain λ and the points represent the true solution. If the numerical outputs are very close to the true solution, then the points should be in the center of the circles.

3.2. **Bregman iterative method.** The Bregman distance [1] is defined as

$$D_J^p(v,u) = J(v) - J(u) - \langle v-u, p\rangle, \quad p \in \partial_u J$$

where $p \in \partial J(u)$ is an element in the subgradient of J at the point u.

The Bregman iterative method for solving the basis pursuit problem (18) was proposed in [29]. Here, $J(u) = |u|_1$ and $H(u,f) = \lambda\|Au-f\|_2^2$.

Algorithm (*Proposed*):

1: Initialize: $k=0$, $u^0 = \mathbf{0}$, $f^0 = \mathbf{0}$.
2: **while** $\frac{\|Au-f\|_2}{\|f\|_2} >$ tolerance **do**
3: Solve $u^{k+1} \to \arg\min_u |u|_1 + \lambda\|Au - f^k\|_2^2$ by coordinate descent
4: $f^{k+1} \to f + (f^k - Au^{k+1})$
5: $k \to k+1$
6: **end while**

The largest computational cost is in step 3, which we solve by refined coordinate descent. One advantage of using Bregman iteration is that it will converge to the exact solution of the constrained problem for any choice of λ in the subproblem (step 3). So λ should be selected such that the subproblem can be solved efficiently. When λ is very big, the convergence is slow. This effect is evident in Table 1 and Table 2. In the other extreme, if λ is too small, then more Bregman iterations are needed. So the best value for λ is something intermediate. Numerical experiences show that, by choosing an appropriate λ such that both the subproblems are solved quickly and few Bregman iterations are needed, the constrained problem can be solved with more accuracy and faster than solving (17) with a big λ.

Another advantage of the proposed algorithm is helpful for its speed. Using the Bregman iterative method for solving (18), it is not necessary to solve the subproblem (step 3) very accurately, which is very time-consuming. Here, we pick the stopping condition as $\|\Delta u\|_\infty \le 10^{-5}$.

3.3. **Numerical results.** Now we implement the proposed algorithm to solve (18). For every Bregman iteration, we solve an unconstrained problem using the coordinate descent method, then update f and go to the next Bregman iteration. We repeat this process until convergence. Table 3 shows the accuracy at each Bregman iteration using the pathwise coordinate descent method with a fixed λ. Table 4 shows the accuracy using the refined coordinate descent method. The numerical

examples are the same as in the previous section and we use a fixed $\lambda = 0.01$ for Table 3 and $\lambda = 0.1$ for Table 4.

Bregman Step	Relative Residual	Relative Error
1	8.0×10^{-3}	1.8×10^{-1}
2	1.8×10^{-7}	1.7×10^{-6}
3	2.3×10^{-8}	2.1×10^{-7}

TABLE 3. The relative residuals and errors for solving (18) after each Bregman iteration using the pathwise coordinate descent method. The total computational time after 3 Bregman iterations is 46s.

Bregman Step	Relative Residual	Relative Error
1	8.0×10^{-3}	1.8×10^{-1}
2	1.3×10^{-8}	1.6×10^{-7}
3	8.8×10^{-9}	9.9×10^{-8}

TABLE 4. The relative residuals and errors for solving (18) after each Bregman iterative step using the refined method. We obtain very accurate solution with 0.3s computational cost.

From the two tables, we see that the numerical results have very high accuracy after only 3 Bregman iterative steps, so it solves the constrained problem (18). Compared to method of solving (18) by using very big λs, this is much more efficient and accurate.

3.4. **Denoising.** If the input data f is noisy, and we know the noise level, then we can stop before convergence to denoise, see [29, 18]. Assume the noise level is known $\|f - f_{exact}\|_2^2 = \epsilon$, then we use coordinate descent to minimize $E(u) = |u|_1 + \lambda\|Au - f\|_2^2$ with the stopping condition $\|Au - f\|_2^2 \le \epsilon$. The numerical example uses the same matrix A and u_{exact} as in the previous, see Figure 3.

3.5. **A special numerical example.** In this example, our original sparse signal has a high dynamical range. u_{exact} has 80 spikes, whose values are generated by multiplying a uniform random number in $[0, 1]$ with another one randomly picked from $\{1, 10, \ldots, 10^{10}\}$. The matrix A is a normalized Gaussian matrix of size 1200×4000. By choosing a big $\lambda = 10^6$, our recovered result is very accurate, see Figure 4. It stopped after 776 iterations (runtime 0.25s) and $\frac{\|Au-f\|_2}{\|f\|_2} = 4.26\times10^{-14}$, $\frac{\|u-u_{\text{exact}}\|_2}{\|u_{\text{exact}}\|_2} = 3.65 \times 10^{-14}$ and $\|u - u_{\text{exact}}\|_\infty = 1.64 \times 10^{-4}$.

4. **Comparison to other methods.** We introduce two other fast methods and then compare their numerical speed with the proposed method for solving (18). The fixed point continuation (FPC [16, 27]) method is efficient for solving (17). The method works in the following way:

$$\min_u E(u) = \underbrace{\tfrac{1}{2}\|Au - f\|_2^2}_{\phi_1} + \underbrace{\mu|u|_1}_{\phi_2},$$

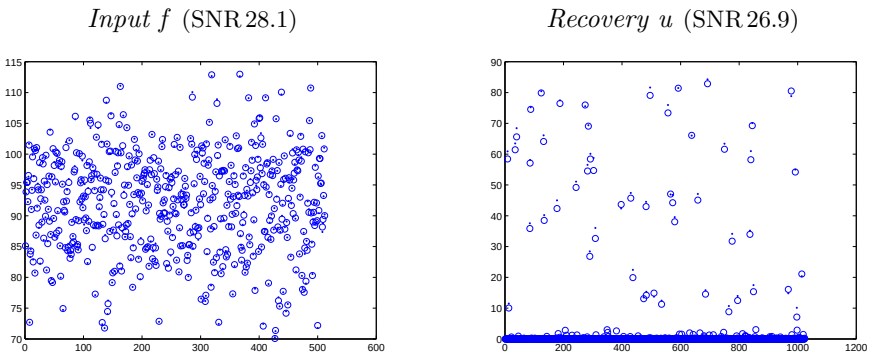


FIGURE 3. For the left picture, the circles are the numerical noisy inputs and the points represent the clean f; for the right plot, the circles are the numerical results and the points represent the exact solution. *Parameters*: $\lambda = 0.3$, runtime 0.35s.

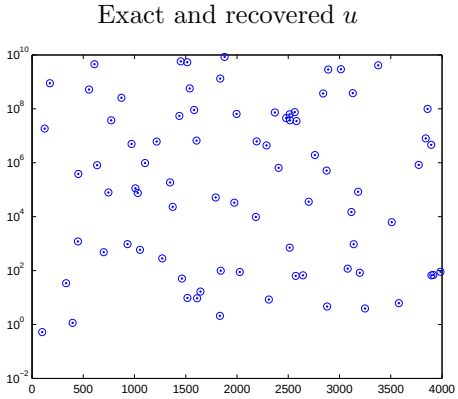


FIGURE 4. The recovery of signal with high dynamical range. The dots are the original signal and the circles are recovered result. *Parameters*: $\lambda = 10^6$, runtime 0.25s.

where $\mu = \frac{1}{2\lambda}$. We denote $T_1 = \partial\phi_1 = \mu\,\mathrm{sgn}(u)$, $T_2 = \partial\phi_2 = A^\top(Au - f)$. Then

$$0 = \partial E(u) \Leftrightarrow 0 = (u + \tau T_1(u)) - (u - \tau T_2(u))$$
$$\Leftrightarrow u = (I + \tau T_1)^{-1}(I - \tau T_2)u.$$

The fixed point iterative method obtains the exact solution by updating u as

$$u^{k+1} := (I + \tau T_1)^{-1}(I - \tau T_2)u^k.$$

It turns out that $(I + \tau T_1)^{-1} = \mathrm{shrink}(\cdot, \tau/\mu)$, where $\nu = \tau/\mu$. So

$$u^{k+1} = \mathrm{shrink}(u^k - \tau A^\top(Au^k - f), \nu).$$

To solve (17), we update u using the above formula until convergence. For the application to compressed sensing, we use FPC to solve the subproblem (step 3) in the Bregman iterative step. If we think of the coordinate method as Gauss-Seidel,

then FPC method is indeed successive over-relaxation (SOR). So FPC needs fewer iterations, but the computational cost of one FPC iteration is $O(nm)$ and the computational cost of one coordinate iteration is $O(n)$. The FPC method has a new version (FPC_BB) [27] to accelerate convergence, and we use the new version in the comparison (Table 5).

The linearized Bregman method [29, 18, 2] solves the constrained problem

$$\min \mu|u|_1 + \frac{1}{2\delta}\|u\|_2^2, \text{ s.t. } Au = f,$$

where λ is a positive scalar parameter. When δ is large enough, its solution is the solution of (18), see [2, 30]. The algorithm is a two-line code:

$$\begin{aligned} v^0 &= 0; u^0 = 0. \\ u_i^{k+1} &\leftarrow \text{shrink}(v_i^k, \mu\delta), \quad i = 1, \ldots, n, \\ v^{k+1} &\leftarrow v^k + \delta A^\top (f - Au^{k+1}). \end{aligned}$$

Now we compare the three different methods. For the numerical tests, A is an $m \times n$ normalized Gaussian matrix, and the original sparse signal u_{exact} has round$(0.05 \times n)$ spikes, which have amplitudes uniformly distributed in $[0, n]$. When A is a normalized Gaussian matrix instead of a uniformly distributed in $[0, 1]$ as in some previous numerical examples, all three methods converge faster, which does not effect the comparison result. For the proposed method, we assume $A^\top A$ is computed ahead, which is not included in the total time showing in the table.[1]

$m \times n$	512×1024	1024×2048	2048×4096
LB	0.8	5.9	10
FPC_BB	0.3	1.3	6.2
Proposed	0.07	0.21	0.94

TABLE 5. The runtime (in seconds) comparison of FPC_BB, linearized Bregman and proposed methods.

Generally, the FPC and proposed method use few Bregman iterations but both of them need lots of iterations to solve the subproblem (step 3); the linearized Bregman method has closed form for the subproblem but it needs many more Bregman iterations than the other two. For the same type of data as used in Table 5 ($m = 0.5 \times n$), FPC converges with a fixed number of iterations. However the proposed method needs a number of iterations which linearly depends on the number of spikes of u_{exact} (round$(0.05 \times n)$). But the computational cost of one FPC iteration is $O(mn)$ and the computational cost of one iteration with the proposed method is $O(n)$. Both of them use $O(mn)$ computation for the total cost, but our proposed method is faster than FPC.

If the matrix A is a DCT or FFT matrix, then it takes $O(n \log n)$ to multiply A with a vector. The total cost of FPC becomes $O(n \log n)$, but the total time of the proposed method is still $O(mn)$, which makes FPC faster than the proposed method with big matrices, see Table 6 ($m = 0.5 \times n$). The linearized Bregman iterative method has two parameters μ and δ, which are crucial to the speed.

Actually, for recovering a sparse signal, there is no need to compute the entire $B = A^\top A$. We only need parts of the columns of B. To save the computation, we

[1] All the experiments are done on an AMD Athlon 64x2 Dual Core 5600+ and 2GB RAM and coded in MATLAB R2007b.

$m \times n$	512×1024	1024×2048	2048×4096	4096×8192
FPC_BB	0.05	0.08	0.16	0.73
Proposed	0.03	0.12	0.46	2.04

TABLE 6. The runtime (in seconds) comparison of FPC_BB and proposed methods.

only compute the column when we need, and then save it in case we need it again. Moreover, to compute the columns of B and the other computation in the proposed algorithm is separable, which gives an advantage for possible parallel computation.

5. **Conclusion.** In this article, we developed a fast coordinate descent method to solve (17). By increasing the penalty parameter λ, we can solve the basis pursuit problem (18), but at the cost of long runtimes. So instead, we combine a Bregman iterative method with a greedy coordinate optimization method, which allows us to solve (18) for small λ without losing accuracy and efficiency. Numerical examples indicate that this is a very efficient method.

Acknowledgments. Thanks to Pascal Getreuer for his helpful suggestions.

REFERENCES

[1] L. Brègman, *The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming*, USSR Computation Mathematics and Mathematical Physics, **7** (1967), 200–217.
[2] J. Cai, S. Osher and Z. Shen, *Linearized bregman iterations for compressed sensing*, Math. Comp., **78** (2009), 1515–1536.
[3] E. Candès, J. Romberg and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on Information Theory, **52** (2006), 5406–5425.
[4] E. Candès and T. Tao, "Decoding by Linear Programming," IEEE Transactions on Information Theory, Dec. 2005.
[5] E. Candès and J. Romberg, *Sparsity and incoherence in compressive sampling*, Inverse Problems, **23** (2007), 969–985.
[6] S. Chen, D. Donoho and M. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., **20** (1998), 33–61.
[7] J. Claerbout and F. Muir, *Robust modeling with erratic data*, Geophysics, **38** (1973), 826–844.
[8] R. Chartrand, *Nonconvex compressed sensing and error correction*, in Proceedings of the 32nd Inter- national Conference on Acoustics, Speech, and Signal Processing (ICASSP), 889–892, 2007.
[9] P. Combettes and J. Pesquet, *Proximal thresholding algorithm for minimization over orthonormal bases*, SIAM J. Optim., **18** (2007), 1351–1376.
[10] J. Darbon and M. Sigelle, *Image restoration with discrete constrained total variation, Part I: Fast and exact optimization*, J. Math. Imaging Vision, **26** (2006), 261–276.
[11] M. Davies and T. Blumensath, *Faster and greedier: Algorithm for sparse reconstruction of large datasets*, Invited paper to the third IEEE International Symposium on Communications, Control, and Signal Processing, March 2008.
[12] D. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory, **52** (2006), 1289–1306.
[13] D. Donoho, *For most large underdetermined systems of linear equations, the minimal ell-1 norm near-solution approximates the sparsest near-solution*, Communications on Pure and Applied Mathematics, **59** (2006), 907–934.
[14] D. Donoho, Y. Tsaig, I. Drori and J. Strack, "Sparse Solutions of Underdetermined Linear Equations by Stagewise Orthogonal Matching Pursuit," IEEE Trans. on Signal Processing, 2006.

[15] J. Friedman, T. Hastie, H. Hofling and R. Tibshirani, *Pathwise coordinate optimization*, The Annals of Applied Statistics, **1** (2007), 302–332.
[16] E. Hale, W. Yin and Y. Zhang, *A Fixed-Point continuation method for ℓ^1-regularized minimization with applications to compressed sensing*, CAAM Technical Report, 07-07.
[17] S. Mallat and Z. Zhang, *Matching pursuit with time-frequency dictionaries*, IEEE Transactions on Signal Processing, **41** (1993), 489–509.
[18] S. Osher, Y. Mao, B. Dong and W. Yin, *Fast linearized Bregman iteration for compressive sensing and sparse denoising*, UCLA CAM Report 08-37.
[19] F. Santosa and W. Symes, *Linear inversion of band-limited reection seimograms*, SIAM J. Sci. Statist. Comput., **7** (1986), 1307–1330.
[20] H. Taylor, S. Banks and J. McCoy, *Deconvolution with the ℓ_1 norm*, Geophysics, **44** (1979), 39–52.
[21] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Roy. Statist. Soc. Ser. B, **58** (1996), 267–288.
[22] J. Tropp, *Greed is good: Algorithmic results for sparse approximation*, IEEE Transactions on Information Theory, **50** (2004), 2231–2242.
[23] J. Tropp, *Just relax: Convex programming methods for identifying sparse signals*, IEEE Transactions on Information Theory, **51** (2006), 1030–1051.
[24] J. A. Tropp and A. C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Trans. on Info. Theory, **53** (2007), 4655–4666.
[25] P. Tseng, "Coordinate Ascent for Maximizing Nondifferentiable Concave Functions," Technical Report LIDS-P; 1840, Massachusetts Institute of Technology. Laboratory for Information and Decision Systems, 1988.
[26] P. Tseng, *Convergence of block coordinate descent method for nondifferentiable maximization*, J. Opt. Theory and Applications, **109** (2001), 474–494.
[27] Z. Wen, W. Yin, D. Goldfarb and Y. Zhang, *A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation*, Preprint, 2009.
[28] T. Wu and K. Lange, *Coordinate descent algorithm for lasso penalized regression*, The Annals of Applied Statistics, **2** (2008), 224–244.
[29] W. Yin, S. Osher, D. Goldfarb and J. Darbon, *Bregman iterative algorithm for ℓ^1 minimization with applications to compressed sensing*, SIAM J. Imaging Sciences, 143–168, 2008.
[30] W. Yin, *On dual penalty methods for basis pursuit*, Preprint, 2008.

Received February 2009; revised June 2009.

E-mail address: yingyingli@math.ucla.edu
E-mail address: sjo@math.ucla.edu