MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Pages 000–000
S 0025-5718(XX)0000-0

A DUAL SPLIT BREGMAN METHOD FOR FAST ℓ^1 MINIMIZATION

YI YANG, MICHAEL MÖLLER, AND STANLEY OSHER

Abstract. In this paper we propose a new algorithm for fast ℓ^1 minimization as frequently arising in compressed sensing. Our method is based on a split Bregman algorithm applied to the dual of the problem of minimizing $\|u\|_1 + \frac{1}{2\alpha}\|u\|^2$ such that u solves the under determined linear system $Au = f$, which was recently investigated in the context of linearized Bregman methods.

Furthermore, we provide a convergence analysis for split Bregman methods in general and show with our compressed sensing example that a split Bregman approach to the primal energy can lead to a different type of convergence than split Bregman applied to the dual, thus making the analysis of different ways to minimize the same energy interesting for a wide variety of optimization problems.

1. Introduction

The field of compressed sensing and techniques using sparsity has recently gained a lot of attention in various areas of research like image processing, inverse problems and data analysis. Representing the solution to an under determined linear system in an appropriate basis can mathematically be expressed as finding the sparsest solution to

$$Au = f, \tag{1.1}$$

with a matrix $A \in \mathbf{R}^{m\times n}$, data $f \in \mathbf{R}^m$ and the unknown $u \in \mathbf{R}^n$, usually with m much smaller than n. Sparsity refers to the number of non-zero components in the solution to (1.1) and is measured by the so called ℓ^0-norm, which is not a norm in the mathematical sense. Minimizing this ℓ^0-norm unfortunately leads to a highly non-convex problem which is extremely computationally expensive to solve exactly. Therefore, one often uses convex relaxation and minimizes the ℓ^1-norm instead,

$$\min_u \|u\|_1 \qquad \text{subject to } Au = f. \tag{1.2}$$

Under various conditions or requirements at the matrix A the equivalence between the ℓ^1 and the ℓ^0 minimizing solutions can be shown. We refer to [8, 9, 11, 10] and the references therein.

For some practical applications, e.g., probabilistic, data mining, hyperspectral imaging, only non-negative solutions u are physically meaningful such that the

2010 *Mathematics Subject Classification.* Primary 49M29, 65K10, 90C25; Secondary 65F22.

Key words and phrases. ℓ^1 Minimization, Optimization Algorithms, Sparsity, Compressed Sensing, Calculus of Variation.

model is changed to minimizing

$$\min_u \|u\|_1 \qquad \text{subject to } Au = f,\ u \geq 0. \tag{1.3}$$

Although (1.2) and (1.3) are much easier to solve than the ℓ^0 minimization problem, the ℓ^1 norm is not differentiable and thus the minimization can still be challenging particularly in high dimensions. In the next section we will summarize the most important algorithms for solving (1.2) before presenting our new approach in Section 3. We will analyze the convergence speed of our method and related split Bregman methods theoretically in Section 4, before showing numerical results for both plain ℓ^1 minimization and non-negative ℓ^1 minimization in Section 5. Finally, we conclude and suggest future areas of research in Section 6.

2. Numerical methods for ℓ^1 minimization

Many methods for ℓ^1 minimization have been proposed. In this work we will focus on the split Bregman method, the adaptive inverse scale space method, the fixed point continuation method with Bregman iteration as well as the linearized Bregman method and its generalizations, which we will all summarize in this section. For other ℓ^1 minimization techniques we refer, for example, to the website `http://nuit-blanche.blogspot.com` and the references discussed there.

2.1. **Split Bregman method.**

Bregman iteration [21, 25, 7] has recently gained a lot of attention due to its efficiency for many ℓ^1 related minimization problems. It is in general formulated for problems of the form

$$\min_u J(u) \qquad \text{such that } Au = f, \tag{2.1}$$

for a convex functional $J(u)$. Bregman iteration constructs a sequence u_k of minimizers

$$u_k = \arg\min_u \frac{\lambda}{2}\|Au - f\|^2 + J(u) - \langle p_{k-1}, u\rangle, \tag{2.2}$$

where p_{k-1} is an element of the subdifferential of J at u_{k-1}, i.e., $p_{k-1} \in \partial J(u_{k-1}) = \{p : J(u) - J(u_{k-1}) - \langle p, u - u_{k-1}\rangle \geq 0\ \forall u\}$. The resulting update for the subgradient p_k shows the equivalence to the well known augmented Lagrangian method (c.f. [25, 13]). The update scheme (2.2) can be rewritten into a simple two step procedure of the form

$$u_k = \arg\min_u \frac{\lambda}{2}\|Au - f + b_{k-1}\|^2 + J(u), \tag{2.3}$$

$$b_k = b_{k-1} + Au_k - f. \tag{2.4}$$

In [14] Goldstein and Osher proposed to use the above Bregman iteration to solve problems of the form

$$\min_u H(Au) + J(u) \tag{2.5}$$

with convex H and J by introducing a new variable d (splitting) and enforcing $d = u$ by Bregman iteration

$$(u_k, d_k) = \arg\min_{(u,d)} H(Au) + J(d) + \|u - d + b_{k-1}\|^2, \tag{2.6}$$

$$b_k = b_{k-1} + u_k - d_k, \tag{2.7}$$

where (2.6) is often done in an alternating fashion. Applying this scheme to our problem (1.2) (i.e. $H(Au)$ being the indicator function of $Au = f$ and $J(d) = \alpha\|d\|_1$) we obtain Algorithm 1. Notice that α is a parameter that influences the convergence speed but not the final solution. $A^\dagger$ denotes the pseudo inverse which for a typical compressed sensing matrix can be computed as $A^\dagger = A^t(AA^t)^{-1}$, and therefore only requires the inversion of the relatively small and well conditioned matrix AA^t. In case we want to solve the non-negative problem (1.3), we simply replace $\text{shrink}(u_k+b_{k-1},\alpha) = \text{sign}(u_k+b_{k-1})\max(|u_k+b_{k-1}|-\alpha, 0)$ by $\text{shrink}^+(u_k+b_{k-1},\alpha) = \max(u_k + b_{k-1} - \alpha, 0)$ in Algorithm 1.

Algorithm 1 Split Bregman

```
1. **Parameters:** $A,\ f,\ \alpha > 0,\ \text{threshold} > 0$
2. **Initialization:** $u_0 = 0$, $d_0 = 0$, $b_0 = 0$
while $\|Ad_k - f\| > \text{threshold}$ do
    Compute $u_k = A^\dagger\big(f + A(b_{k-1} - d_{k-1})\big) + d_{k-1} - b_{k-1}$
    Compute $d_k = \text{shrink}(u_k + b_{k-1}, \alpha)$
    Update $b_k = b_{k-1} + u_k - d_k$
end while
return $d_k$
```

2.2. **Adaptive inverse scale space method.** The so called inverse scale space (ISS) flow as first analyzed in [2] can be seen as the continuous version of Bregman iteration. Looking at the optimality condition for problem (2.2) we obtain

$$0 = \lambda A^t(Au_k - f) + p_k - p_{k-1}, \tag{2.8}$$

$$\Leftrightarrow \quad \frac{p_k - p_{k-1}}{\lambda} = A^t(f - Au_k). \tag{2.9}$$

Interpreting λ as a time step of a backward Euler discretization we obtain a continuous flow of the form

$$\partial_t p(t) = A^t(f - Au(t)), \qquad p(t) \in \partial\|u(t)\|_1. \tag{2.10}$$

It was found in [3] that the above equation can be solved exactly without any discretization: The solution stays piecewise constant in time intervals $[t_k, t_{k+1}]$, leading to a piecewise linear behavior of the subgradient $p(t)$. Since the subgradient has to evolve continuously $p(t_{k+1})$ can be determined and due to the specific structure of the subdifferential of the ℓ^1-norm, the corresponding $u(t_{k+1})$ can only be non-zero at indices i where $|p_i(t_{k+1})| = 1$. Thus, $u(t_{k+1})$ is the solution of a low dimensional non-negative least squares problem. The resulting adaptive inverse scale space algorithm (aISS) is presented as Algorithm 2 below, where P_I denotes the projection onto the index set I. For more details on the derivation we refer to [3].

Algorithm 2 Adaptive Inverse Scale Space Method

```
1. Parameters: $A,\ f,$ threshold $\geq 0$
2. Initialization: $t_1 = 1/\left\|A^t f\right\|_\infty,\ p(t_1) = t_1\ A^t f,\ I_1 = \{i \mid |p_i(t_1)| = 1\}$
while $\|Au(t_k) - f\| >$ threshold do
    Compute $u(t_k) = \arg\min_u \left\{\|AP_{I_k} u - f\|^2\right\}$ subject to $u(t_k)p(t_k) \geq 0$.
    Obtain $t_{k+1}$ as
        $t_{k+1} = \min\{t \mid t > t_k, \exists j : |p_j(t)| = 1, u_j(t_k) = 0, p_j(t) \neq p_j(t_k)\},$
    Update the dual variable $p(t)$ via (2.11) with $t = t_{k+1}$.
(2.11)      $p_j(t) = p_j(t_k) + (t - t_k)e_j \cdot A^t(f - Au(t_k)).$
    Compute $I_{k+1} = \{i \mid |p_i(t_{k+1})| = 1\}$.
end while
return $u(t_k)$
```

2.3. **Fixed point continuation with Bregman iteration.** As shown above, in each Bregman iteration (2.3) we need to solve the following unconstrained problem:

$$\min_u \frac{\lambda}{2}\|Au - f + b_{k-1}\|^2 + \|u\|_1. \tag{2.12}$$

Generally, the solution to (2.12) can not be determined explicitly, but requires an optimization algorithm. A simple iterative method which utilizes operator splitting was proposed in [15]. This fixed point continuation algorithm (FPC) is also called proximal gradient method (PGM). To summarize FPC let us briefly introduce the idea of proximal mappings.

The proximal mapping (prox-operator) of a convex function h is defined as

$$\mathbf{prox}_h(u) = \arg\min_x \ h(x) + \frac{1}{2}\|x - u\|^2. \tag{2.13}$$

Let us assume we want to minimize a function $f(u)$ of the form

$$f(u) = g(u) + h(u), \tag{2.14}$$

for a convex and differentiable $g(u)$, and a convex $h(u)$ with inexpensive prox-operator. Then the proximal gradient algorithm computes the minimizer of $f(u)$ iteratively via

$$u_k = \mathbf{prox}_{t_k h}(u_{k-1} - t_k \nabla g(u_{k-1})), \tag{2.15}$$

where t_k is the step size. A variant of FPC is the so called fast iterative shrinkage-thresholding algorithm (FISTA) [1],

$$u_k = \mathbf{prox}_{t_k h}(y_{k-1} - t_k \nabla g(y_{k-1})), \tag{2.16}$$

$$y_k = u_k + \frac{k-1}{k+2}(u_k - u_{k-1}), \tag{2.17}$$

which uses extrapolation on the FPC results and is usually several times faster than PGM for solving the unconstrained problem. For our problem where $g(u) = \frac{\lambda}{2}\|Au - f + b_{k-1}\|^2$, $h(u) = \|u\|_1$, the prox-operator becomes the shrinkage-operator,

$$\mathbf{prox}_{th}(u) = \text{shrink}(u, t), \tag{2.18}$$

and we arrive at Algorithm 3.

Algorithm 3 FPC with Bregman iteration

```
1. **Parameters:** $A$, $f$, $0 < t_k < \frac{2}{\|A^tA\|}$, threshold $> 0$, Number of inner iterations $N$
2. **Initialization:** $u_0 = 0$, $b_0 = 0$,
**while** $\|Au_k - f\| >$ threshold **do**
    Set $v_0 = u_{k-1}$.
    For $i = 1$ to $N$, compute
(2.19)    $v_i = \text{shrink}(v_{i-1} - t_{k-1}\lambda A^t(Av_{i-1} - f + b_{k-1}), t_{k-1}).$
    Update $u_k = v_N$.
    Update $b_k = b_{k-1} + Au_k - f$
**end while**
**return** $u_k$
```

Here the spectral norm $\|A^tA\|$ is defined as the largest singular value of A^tA. Notice that the above algorithm needs to restrict the step size $t_k < \frac{2}{\|A^tA\|}$ to guarantee stability. For ill conditioned matrices with large $\|A^tA\|$ this can become a severe restriction. Algorithm 3 is formulated using FPC, but we can also replace FPC with FISTA by using the updates (2.16) and (2.17) instead.

2.4. **Linearized Bregman (LB) iteration and its generalization.** Another iterative method for solving the constrained problem (2.1) is linearized Bregman iteration as introduced in [5, 6, 25]. It uses a first order Taylor expansion of the quadratic term in (2.2) leading to a minimization

$$u_k = \arg\min_u \frac{\lambda}{2}\|Au_{k-1}-f\|^2+\lambda\langle A(u-u_{k-1}), Au_{k-1}-f\rangle+J(u)-\langle p_{k-1}, u\rangle, \tag{2.20}$$

and hence implying the subgradient update

$$p_k = p_{k-1} + \lambda A^t(f - Au_{k-1}). \tag{2.21}$$

For our case, i.e. $J(u) = \|u\|_1$, the energy (2.20) we minimize might become unbounded from below. Therefore, one uses an elastic net, replacing the ℓ^1 norm by $J(u) = \|u\|_1 + \frac{1}{2\alpha}\|u\|^2$, leading to algorithms for the solution of

$$\min_u \|u\|_1 + \frac{1}{2\alpha}\|u\|^2 \text{ such that } Au = f \tag{2.22}$$

It has been shown in [24] that the solution of this elastic net linearized Bregman iteration (Algorithm 4) coincides with the ℓ^1 minimizing solution, if α is chosen sufficiently large. Recently, the authors in [19] gave a detailed proof that $\alpha > 10\|x\|_\infty$ leads to the equivalence of the models with and without the elastic net.

Algorithm 4 Linearized Bregman

```
1. Parameters: $A,\ f,\ \alpha > 0,\ 0 < \delta < \frac{2}{\|A^tA\|}$, threshold $> 0$
2. Initialization: $u_0 = 0$, $v_0 = 0$,
while $\|Au_k - f\| >$ threshold do
    Compute $v_k = v_{k-1} - A^t(Au_{k-1} - f)$
    Compute $u_k = \delta \cdot \text{shrink}(v_k, \alpha/\delta)$
end while
return $u_k$
```

Here the time step again has to meet $\delta < 2/\|A^tA\|$ as a stability requirement. Each iteration of the linearized Bregman algorithm is simple and fast, however the algorithm can stagnate sometimes. The "kicking" technique introduced in [22] can remove the stagnation and give faster convergence. However, even with "kicking" this algorithm can still take a long time for large problems due to the relatively small time step. It was proposed in [24] to explore the dual problem of (2.22):

$$\min_{y,z} \ -f^Ty + \frac{\alpha}{2}\|A^ty - z\|^2 \text{ such that } z \in [-1,1]^n. \tag{2.23}$$

Notice that z can be expressed by $\text{Proj}_{[-1,1]^n}(A^ty)$. Once the solution (y_α, z_α) is achieved, we can get the solution u_α of (2.22) simply by setting

$$u_\alpha = \alpha \cdot \text{shrink}(A^ty_\alpha, 1). \tag{2.24}$$

For nonnegative case we just need to change shrink to shrink^+.

The detailed analysis of the dual formulation in [24] allowed Yin to interpret the linearized Bregman algorithm as a gradient descend algorithm on the dual problem (2.23). Therefore, the idea in [24] was to use acceleration techniques for gradient-based optimization methods such as line search, Barzilai-Borwein and limited memory BFGS to solve the dual problem. Yin proposed two efficient algorithms called LB-BBLS and LB-BFGS. The numerical experiments in [24] indicate that they are usually much faster than linearized Bregman with "kicking". Further acceleration techniques based on the equivalence between the gradient descend and linearized Bregman method were recently proposed in [18].

3. LB-SB

The motivation for our approach is directly related to the linearized Bregman method being equivalent to gradient descend on the dual formulation. Inspired by [24], we try to find an algorithm which can solve (2.23) efficiently. Over the last couple of years many researchers found the alternating directions of multipliers method (ADMM), which is equivalent to the split Bregman method, to often result in a faster minimization technique than gradient descend based methods. Thus, the starting point for our algorithm is the above dual formulation, which, after replacing z by $\text{Proj}_{[-1,1]^n}(A^ty)$, is

$$\min_{y} \ -f^Ty + \frac{\alpha}{2}\|A^ty - \text{Proj}_{[-1,1]^n}(A^ty)\|^2. \tag{3.1}$$

with the relation

$$u = \alpha \cdot \text{shrink}(A^ty, 1) \tag{3.2}$$

between the primal and dual variables.

In order to solve (3.1) efficiently our idea is to apply the split Bregman method to the minimization problem (3.1). We first define $d = A^t y$, and the problem (3.1) is equivalent to

$$\min_{y,d} -f^T y + \frac{\alpha}{2}\|d - \mathrm{Proj}_{[-1,1]^n}(d)\|^2 \text{ such that } d = A^t y. \tag{3.3}$$

Then using Bregman iteration, we get

$$\begin{aligned} (y_k, d_k) &= \arg\min_{y,d}\{\lambda(-f^T y + \frac{\alpha}{2}\|d - \mathrm{Proj}_{[-1,1]^n}(d)\|^2) + \frac{1}{2}\|d - A^t y - b_{k-1}\|^2\}, \\ b_k &= b_{k-1} + A^t y_k - d_k, \end{aligned} \tag{3.4}$$

and do the minimization for y and d in an alternating fashion

$$\begin{aligned} d_k &= \arg\min_{d}\{\frac{\lambda\alpha}{2}\|d - \mathrm{Proj}_{[-1,1]^n}(d)\|^2 + \frac{1}{2}\|d - A^t y_{k-1} - b_{k-1}\|^2\}, \\ y_k &= \arg\min_{y,d}\{-\lambda f^T y + \frac{1}{2}\|d_k - A^t y - b_{k-1}\|^2\}. \end{aligned} \tag{3.5}$$

The optimality conditions to the above optimization problems yield

$$0 = d + \lambda\alpha \cdot \mathrm{shrink}(d,1) - (A^t y_{k-1} + b_{k-1}), \tag{3.6}$$

$$0 = (AA^t)y_k - (Ad_k - Ab_{k-1} + \lambda f). \tag{3.7}$$

In the typical compressed sensing case (AA^t) will be relatively small and well conditioned such that we can compute

$$y_k = (AA^t)^{-1}(Ad_k - Ab_{k-1} + \lambda f). \tag{3.8}$$

The solution of (3.6) can be considered element wise by a distinction of two cases. If $|(A^t y_{k-1} + b_{k-1})_i| \le 1$ we can simply choose

$$d_i = (A^t y_{k-1} + b_{k-1})_i \tag{3.9}$$

and since for this component $\mathrm{shrink}(d_i, 1) = 0$ it meets the optimality condition. In the other case, $|(A^t y_{k-1} + b_{k-1})_i| > 1$, $|d_i|$ has to be greater than 1. Furthermore, it is easy to see that the optimality can only be satisfied if $\mathrm{sign}(d_i) = \mathrm{sign}((A^t y_{k-1} + b_{k-1})_i)$. Therefore, $\mathrm{shrink}(d_i, 1) = d_i - \mathrm{sign}((A^t y_{k-1} + b_{k-1})_i)$ and hence

$$d_i = \frac{(A^t y_{k-1} + b_{k-1})_i + \mathrm{sign}((A^t y_{k-1} + b_{k-1})_i)\lambda\alpha}{1 + \lambda\alpha}. \tag{3.10}$$

Our algorithm becomes

Algorithm 5 LB-SB

```
1. Parameters: $A$, $f$, $\alpha > 0$, threshold $> 0$
2. Initialization: $y_0 = 0$, $b_0 = 0$, $\lambda = 10$.
while $\|Au_k - f\| >$ threshold do
    Solve $d + \lambda\alpha \cdot \mathrm{shrink}(d,1) = A^t y_{k-1} + b_{k-1}$ for $d_k$ with (3.9) or (3.10).
    Compute $y_k = (AA^t)^{-1}(Ad_k - Ab_{k-1} + \lambda f)$.
    Update $b_k = b_{k-1} + A^t y_k - d_k$.
    Compute $u_k = \alpha \cdot \mathrm{shrink}(A^t y_k, 1)$.
end while
return $u_k$
```

A similar algorithm can be derived for the non-negative case. The projection becomes a projection onto the non-negative part of the unit ball and therefore shrinkage becomes non-negative shrinkage. Thus, we obtain the update formula

$$d_i = \begin{cases} \max\left((A^t y_{k-1} + b_{k-1})_i, 0\right) & \text{if } (A^t y_{k-1} + b_{k-1})_i \leq 1, \\ \frac{(A^t y_{k-1}+b_{k-1})_i + \lambda\alpha}{1+\lambda\alpha} & \text{else.} \end{cases} \tag{3.11}$$

The rest of the algorithm remains the same and we determine

$$u_k = \alpha \cdot \text{shrink}^+(A^t y_k, 1) \tag{3.12}$$

4. Convergence analysis

As we have seen our algorithm is the split Bregman algorithm applied to the dual of the constrained minimization of the elastic net regularized energy. One of the main questions would of course be why we use the elastic net formulation and why we are looking at the dual instead of the primal energy. Therefore, it is natural to investigate the type of convergence of each method.

The convergence analysis of Bregman iteration in general can be found in [21]. In [4] the authors derive an additional error estimate based on the assumption of a so called *source condition*, which we will explain in more detail in the next subsection. For the split Bregman algorithm in particular, it has been shown in [23] that the algorithm converges based on its equivalence to the Augmented Lagrangian method and that solving (2.6) in an alternating fashion converges due to an equivalence to Douglas-Rachford Splitting. However, to the best knowledge of the authors, little work has been done on analyzing the convergence speed split Bregman schemes in general. We are particularly interested in developing a framework to analyze different types of convergence for different types of split Bregman methods - for instance for minimizing the same energy with the splitting applied to the primal or to the dual energy. Our analysis will be based on a generalization of the results from [20]. To analyze the difference between the different split Bregman methods to determine ℓ^1 minimizing solutions, let us first summarize some results of [4] for Bregman iteration in general.

4.1. **Error estimates for Bregman iteration.**

In [4] Burger, Resmerita and He derived an error estimate for the Bregman iteration that is used for solving

$$\min_u J(u) \text{ such that } Au = f, \tag{4.1}$$

for a convex functional $J(u)$, i.e. an error estimate for an iteration of the form

$$\begin{aligned} u_{k+1} &= \arg\min_u J(u) + \frac{\lambda}{2}\|Au - f + b_k\|^2, \\ b_{k+1} &= b_k + f - Au_k. \end{aligned} \tag{4.2}$$

The requirement for the error estimate is that the true solution to Problem (4.1) must meet a so called source condition, which is defined as follows.

Definition 4.1. We say that an element $\tilde{u}$ meets a source condition (SC) if there exists an element q such that $A^t q \in \partial J(\tilde{u})$.

With the definition of a SC Theorem 4.1 in [4] reads as follows:

Theorem 4.2. [4] *Let $\tilde{u}$ be a solution to $Au = f$ and assume that $\tilde{u}$ meets a SC. Then the following error estimate holds for the iterative scheme (4.2):*

$$D_J^{p_k}(\tilde{u}, u_k) \leq \frac{\|q\|^2}{2\lambda k}, \tag{4.3}$$

where q is the source element from the SC and $p_k = \lambda \sum_{j=1}^k A^t(f - Au_j) \in \partial J(u_k)$.

As first shown in [20] we can use this estimate to derive a convergence speed for Bregman iteration in general. In some cases this allows us to draw conclusions about the question whether the split Bregman method is more efficient if applied to the primal problem or to the dual problem. In the next section we will prove a convergence result for a very general type of energy minimization problem and then give a recipe on how to apply the convergence results to specific examples.

4.2. A general framework for the convergence analysis of split Bregman methods.

Split Bregman can be applied in various cases of energy minimization. Typically, we are looking at minimization problems of the form

$$\min_u H(u) + R(Tu), \tag{4.4}$$

for convex functionals $H(\cdot)$ and $R(\cdot)$, and a linear operator T. Common examples are $H(u) = \frac{\alpha}{2}\|Au - f\|^2$ and $R(Tu)$ being TV regularization or - with $T = I$ being the identity - ℓ^1 regularization.

The idea of split Bregman is to avoid the difficulty of having to minimize (4.4) all at once by introducing a new variable $d = Tu$, leading to the constrained minimization problem

$$\min_{u,d} H(u) + R(d) \text{ such that } Tu = d, \tag{4.5}$$

which is solved by Bregman iteration

$$\begin{aligned} (u_{k+1}, d_{k+1}) &= \arg\min_{u,d} H(u) + R(d) + \frac{\lambda}{2}\|Tu - d + b_k\|^2, \\ b_{k+1} &= b_k + Tu_{k+1} - d_{k+1}. \end{aligned} \tag{4.6}$$

As we mentioned in section 2.1 already, the above minimization for u and d is often done in an alternating fashion, which turns the split Bregman (or augmented Lagrangian) method, into the alternating split Bregman method (or alternating directions of multipliers method (ADMM)). Our convergence analysis will be concerned with solving the optimization for u and d simultaneously without alternating minimization. Despite the fact that this leads to a slightly different iteration, we believe the two approaches to be closely related and have similar convergence properties. Particularly, factors that influence the convergence speed of the split Bregman method, will also influence the convergence speed of the alternating split Bregman method, which is why we believe that the convergence analysis below is very useful, even when using the alternating minimization approach. After the first submission of this manuscript, He and Yuan proved an ergodic convergence of the Douglas-Rachford operator splitting (in a metric similar to the Bregman distance) in [16] as well as its non-ergodic convergence (in a metric based on how close the optimality condition is to being met) in [17]. By duality, Douglas-Rachford splitting is equivalent to the alternating split Bregman method (c.f. Figure 3.1 in [13]), such that their convergence analysis provide estimates for the alternating minimization case.

Continuing our analysis, notice that the above minimization problem (4.5) can be written as

$$\min_{u,d} H(u) + R(d) \text{ such that } (T \ \ -I)\begin{pmatrix} u \\ d \end{pmatrix} = 0, \tag{4.7}$$

which has exactly the same form as (4.1) and the split Bregman algorithm is just Bregman iteration on the above problem with the operator $\tilde{A} = (T \ \ -I)$ and the right hand side $f = 0$. Therefore, the convergence estimate of Theorem 4.2 holds, if we can show that the true solution to the above problem meets a source condition. We can use the lemma from [20] which we have slightly generalized here:

Lemma 4.3. *(generalization of Lemma 3.3 in* [20]*)*
The split Bregman algorithm for Problem (4.5) always satisfies a source condition.

Proof. To prove this lemma we have to show that there exists a source element q such that $\tilde{A}^T q \in \partial_{(u,d)}(H(\tilde{u}) + R(\tilde{d}))$ for the true solution $(\tilde{u}, \tilde{d})$ to (4.5). With

$$\tilde{A}^T = \begin{pmatrix} T^T \\ -I \end{pmatrix}$$

we can write the SC as the existence of a q such that

$$T^T q \ \in \ \partial_u H(\tilde{u}), \tag{4.8}$$

$$-q \ \in \ \partial_d R(\tilde{d}). \tag{4.9}$$

Notice that since $(\tilde{u}, \tilde{d})$ is the true solution to (4.5), we have $\tilde{d} = T\tilde{u}$ and $\tilde{u}$ being the minimizer of (4.4). The optimality condition to (4.4) tells us that

$$p_1 + p_2 = 0$$

for a $p_1 \in \partial_u H(\tilde{u})$ and $p_2 \in \partial_u E(\tilde{u})$ for $E(u) = R(Tu)$. By subdifferential calculus (see e.g. [12]) the latter condition can be written as $p_2 \in T^T \partial_{Tu} R(T\tilde{u})$. Thus, there exists a $\tilde{q} \in \partial_{Tu} R(T\tilde{u})$ such that

$$\begin{aligned} & p_1 + T^T \tilde{q} = 0, \\ \Rightarrow \quad & p_1 = T^T(-\tilde{q}) \in \partial_u H(\tilde{u}). \end{aligned}$$

Knowing that the true solutions satisfies $\tilde{d} = T\tilde{u}$ together with the two implications

$$\begin{aligned} \tilde{q} \ &\in \ \partial_{Tu} R(T\tilde{u}), \\ T^T(-\tilde{q}) \ &\in \ \partial_u H(\tilde{u}). \end{aligned}$$

and comparing to the necessary conditions (4.8), (4.9) for a SC, we can see that by choosing $q = -\tilde{q}$ a SC is always satisfied for the split Bregman algorithm. □

With Theorem 4.2 and Lemma 4.3 we have seen, that the split Bregman algorithm (4.6) always converges with

$$D_J^{(p_u^k, p_d^k)}\big((\tilde{u}, \tilde{d}), (u_k, d_k)\big) \le \frac{\|q\|^2}{2\lambda k}, \tag{4.10}$$

where q is the source element such that (4.8) and (4.9) hold. Furthermore, the Bregman distance to (u_k, d_k) decomposes into the sum of the two Bregman distances with respect to R and H.

$$\begin{aligned}
& D_J^{(p_u^k,p_d^k)}\big((\tilde{u},\tilde{d}),(u_k,d_k)\big) \\
= \; & H(\tilde{u}) + R(\tilde{d}) - (H(u_k) + R(d_k)) - \left\langle \begin{pmatrix} \tilde{u} \\ \tilde{d} \end{pmatrix} - \begin{pmatrix} u_k \\ d_k \end{pmatrix}, \begin{pmatrix} p_u^k \\ p_d^k \end{pmatrix} \right\rangle \\
= \; & D_H^{p_u^k}(\tilde{u},u_k) + D_R^{p_d^k}(\tilde{u},d_k).
\end{aligned}$$

Knowing the estimate for minimizing very general energies with the split Bregman method, we can now give a recipe for how to obtain more precise estimates for specific energy minimization problems:

(1) Determine the operator T and the two parts $H(u)$ and $R(d)$ of the energy after the splitting.
(2) Characterize the subdifferential of H at u_k as well as the Bregman distance $D_H^{p_u^k}(\tilde{u},u_k)$.
(3) Characterize the subdifferential of R at d_k as well as the Bregman distance $D_H^{p_d^k}(T\tilde{u},d_k)$.
(4) Try to determine the source element q by the conditions

$$T^T q \in \partial_u H(\tilde{u}), \tag{4.11}$$

$$-q \in \partial_d R(\tilde{d}). \tag{4.12}$$

(5) We can conclude the convergence speed

$$\boxed{D_H^{p_u^k}(\tilde{u},u_k) + D_R^{p_d^k}(T\tilde{u},d_k) \leq \frac{\|q\|^2}{2\lambda k},} \tag{4.13}$$

to the true solution and by Theorem 2.2 in [4] have the relative convergence of the two variables

$$\boxed{\frac{\lambda}{2}\|Tu_k - d_k\|^2 \leq \frac{H(\tilde{u}) + R(T\tilde{u})}{k}.} \tag{4.14}$$

In the next subsection we will apply this concept to the three split Bregman methods relevant for solving our problem: The primal split Bregman method for solving (1.2) as described in Section 2.1, our proposed method, and the split Bregman method applied to the primal Problem when also using the elastic net $J(u) = \|u\|_1 + \frac{1}{2\alpha}\|u\|^2$.

4.3. **Convergence analysis for split Bregman methods determining ℓ^1 minimizing solutions.**

4.3.1. *Constrained model - primal ℓ^1 minimizing solutions.* Let us first look at the primal Bregman method for solving (1.2) as described in Section 2.1. Following our recipe we can see that

(1) $T = I$ is the identity and we split according to the two parts

$$R(d) = \begin{cases} 0 & \text{if } Ad = f \leq \infty \\ \infty & \text{if } Ad \neq f \end{cases}, \tag{4.15}$$

$$H(u) = \|u\|_1. \tag{4.16}$$

(2) The Bregman distance with respect to H, i.e. the ℓ^1 norm, measures how close the subgradient p_u^k is to being a subgradient of $\tilde{u}$.
(3) The Bregman distance with respect to R tells us that every d_k satisfies $Ad_k = f$, but does not have much more expressiveness because $D_R^{p_d^k}(T\tilde{u}, d_k) = \langle p_d^k, d_k - \tilde{u}\rangle$ and p_d^k can be arbitrarily close to zero. We can say that $p_d^k = \lambda \sum_{i=1}^k (u_i - d_i)$ but do not know a-priori what this quantity will be.
(4) The source element q is difficult to determine more precisely. For the same reason the Bregman distance with respect to R is hard to classify further the condition $q \in \partial R(\tilde{u})$ does not reveal much about q. The second condition tells us that $q \in \partial \|\tilde{u}\|_1$, which bounds q by 1 in the infinity norm. We can hope that the entries in q corresponding to zero components in $\tilde{u}$ are much smaller in magnitude, but there in no guarantee. Thus, for a vector $\tilde{u} \in \mathbf{R}^n$ the best estimate we can give is $\|q\|^2 \le n$.
(5) We obtain the convergence speeds

$$\|\tilde{u}\|_1 - \langle p_u^k, \tilde{u}\rangle \le \frac{n}{2\lambda k}, \tag{4.17}$$

$$\|Au_k - f\|^2 \le \frac{2}{\lambda k}\|A\|^2\|\tilde{u}\|_1. \tag{4.18}$$

4.3.2. *The proposed model - dual ℓ^1 minimizing solutions with elastic net.* Let us apply the convergence analysis to our proposed method.

(1) Since we now enforce $A^t y = d$ by Bregman iteration we have $T = A^t$ and we obtain the two parts

$$R(d) = \frac{\alpha}{2}\|\text{shrink}(d,1)\|^2, \tag{4.19}$$

$$H(y) = -\langle f, y\rangle, \tag{4.20}$$

using Bregman iteration to obtain $\tilde{d} = A^t\tilde{y}$.
(2) The Bregman distance with respect to a fully linear functional, in our case $H(u)$, is always zero.
(3) A short calculation shows that $\partial_d R(d) = \alpha\text{shrink}(d,1)$, and the Bregman distance with respect to R becomes

$$D_R^{p_d^k}(\tilde{u}, d_k) = \frac{\alpha}{2}\|\text{shrink}(d_k,1) - \text{shrink}(A^t\tilde{y},1)\|^2, \tag{4.21}$$

$$= \frac{1}{2\alpha}\|u_k - \tilde{u}\|^2, \tag{4.22}$$

where we used relation (3.2) and denoted the reconstruction at the k-th iteration by $u_k = \alpha\text{shrink}(d_k, 1)$.
(4) By verifying that $\partial_{A^t\tilde{y}} R(A^t\tilde{y}) = \{\tilde{u}\}$ and $\partial H(\tilde{y}) = \{-f\}$ it is easy to see that $q = \tilde{u}$ is our source element satisfying (4.11) and (4.12).
(5) Thus, we obtain the very simple estimate

$$\|u_k - \tilde{u}\|^2 \le \frac{\alpha}{\lambda k}\|\tilde{u}\|^2. \tag{4.23}$$

As we can see we obtain a stronger result here than we did for the primal split Bregman case, namely the convergence of our iterates to the true solution in the ℓ^2 norm. Notice that by the results of [24] α has to be chosen large enough for the method to converge to the ℓ^1 minimizing solution. However, the above estimate shows that a smaller α immediately leads to faster convergence and we obtain results faster the more accuracy in the computation of the ℓ^1 minimizing solution we are willing to sacrifice. This is of course reasonable because in the limit $\alpha \to 0$ we are determining an ℓ^2 minimizing solution, which can be determined in one step.

It is remarkable that the error estimate (4.23) has a pure ℓ^2 nature and does not contain any ℓ^1 term, although for α large enough, we determine an ℓ^1 minimizer. Since the ℓ^2 norm is rather small for entries close to zero we however have to be aware that the convergence might not immediately lead to a very sparse signal but may contain some small oscillations close to zero. If we are willing to sacrifice this type of accuracy the estimate (4.23) indicates fast convergence.

4.3.3. *Constrained model - primal ℓ^1 minimizing solutions with elastic net.* It is interesting to see that even the estimate for the primal Bregman method with elastic net is different from the dual method. For the primal elastic net model we would have

(1) $T = I$

$$R(d) = \|d\|_1, \tag{4.24}$$

$$H(u) = \begin{cases} \frac{1}{2\alpha}\|u\|^2 & \text{if } Au = f \\ \infty & \text{else} \end{cases}. \tag{4.25}$$

(2) The Bregman distance with respect to a $H(u)$ tell us that for each u_k we have $Au_k = f$. The subgradient at u_k is $\frac{1}{\alpha}u_k + r_k$ for some $r_k \in \text{range}(A^t)$, such that

$$D_H^{p_u^k}(\tilde{u}, u_k) = \frac{1}{\alpha}\|\tilde{u} - u_k\|^2 \tag{4.26}$$

(3) As in the case of the primal split Bregman method without the elastic net, the Bregman distance with respect to R, i.e. the ℓ^1 norm, measures how close the subgradient p_d^k is to being a subgradient of $\tilde{u}$.

(4) The negative source element q has to be in the subdifferential of $\|\tilde{u}\|_1$, which - as discussed before - is difficult to interpret any further. The second condition for q tells us that $q = \frac{1}{\alpha}\tilde{u} + r$ for some $r \in \text{range}(A^t)$. The problem is that these two conditions do not allow to calculate the norm $\|q\|^2$ any further. The elastic net leads to the sum of a quadratic convergence term and the Bregman distance with respect to the ℓ^1 norm, but as for the other primal method the best estimate for the convergence constant is the dimension n of $\tilde{u}$, although we can hope to have much better convergence for sparse $\tilde{u}$.

(5) The final estimate we can give is

$$\frac{1}{\alpha}\|u_k - \tilde{u}\|^2 + D_{\|\cdot\|_1}^{p_d^k}(\tilde{u}, d_k) \leq \frac{n}{2\lambda k}. \tag{4.27}$$

Notice that the estimates of the primal and dual method are similar in their asymptotic behavior. They share the term with quadratic convergence in the ℓ^2 norm and the dependency on α. However, the constant on the right hand side is quite different. While the constant is difficult to determine in the primal case it is the ℓ^2 norm of the true solution for the dual case. For the primal method we expect the constant to improve as $\tilde{u}$ becomes more and more sparse while the dual formulation seems to be entirely independent of the sparsity of $\tilde{u}$. The price the dual method pays for this independence is that the error estimate does not contain a term that measures the distance to the solution $\tilde{u}$ in a metric that is related to the sparsity of u_k as it is the case for the primal method with the ℓ^1 Bregman distance $D^{p_d^k}_{\|\cdot\|_1}(\tilde{u}, d_k)$. From the estimates, we would expect that if an approximate solution (with the approximation being close to the true solution in an ℓ^2 sense - not necessarily in terms of their support) is sufficient, the dual method might converge faster to a desired solution particularly if the solution to the solution is not very sparse.

Remarks:

- The above estimates are based on solving (4.6) exactly. In practice one often uses alternating minimization. For the error estimates to hold one has to do several of such alternating minimizations before updating b_{k+1}.
- The error estimates above are estimates based on worst case scenarios. Hence, most of the times the convergence will be faster than predicted by the estimates. However, we believe that the above estimates do reveal something about the general behavior of each method. For any minimization problem that is intended to be solved by split Bregman it might be worthwhile to analyze the convergence of both, a primal and a dual approach, and choose the method which seems to give the more desirable estimate.

5. Numerical results

In this section, we compare our algorithm LB-SB with the methods described in Section 2, which are primal split Bregman (pSB), aISS, FPC with Bregman iteration (FPC for short), FISTA with Bregman iteration (FISTA for short), LB-BBLS and LB-BFGS. As examined in [24], the original linearized Bregman method, even with "kicking", is much slower than LB-BBLS and LB-BFGS. Hence the comparison between our algorithm and these two will suffice.

Here in the first subsection all the sensing matrices A are chosen to be the following type: Gaussian matrices whose elements are generated from independent and identically distributed (i.i.d.) normal distributions. In the 2nd subsection, the rows of A are orthogonalized by QR decompositions. Although different data types tend to give different algorithm performances, the relative speed and robustness of these implementations remains roughly the same. We generate the data f via $f = Au_{real}$, where only a small proportion of the original signal u_{real} is nonzero and the positions of the nonzero elements of u_{real} is selected randomly.

For pSB we have two output variables u and d, where u minimizes $\|Au - Au_{real}\|$ and d gives us a smaller ℓ^0 and ℓ^1 norm. Both outputs are included in the comparison, labeled as pSB-u and pSB-d. Denoting the reconstruction result of each algorithm by u we also compare the relative error defined as $\|u - u_{real}\|/\|u_{real}\|$. The threshold is chosen to be $5 \cdot 10^{-4}\|f\|$. For each setting, 10 different tests are conducted and we recorded the average time, $\|Au - Au_{real}\|$, the ℓ^0 and ℓ^1 norm of the recovered signal as well as the relative error.

5.1. General Case.

In this subsection we focus on solving (1.2), i.e. determining ℓ^1 minimizing solution without a non-negativity constraint and generate the nonzero entries of u_{real} by drawing from a standard Gaussian distribution. For all experiments we used the parameter $\alpha = 5$ for LB-BBLS, LB-BFGS and LB-SB algorithms. Since sometimes FPC and FISTA with Bregman may take a very long time for large problems, we stopped these methods after 400 iterations for matrices A of the size 900×3000 and 1000×5000.

TABLE 1. Noise-free case. Size $A = 100 \times 1000$, 1% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	8.01e-02	8.13e-15	1.00e+01	1.29e+01	5.97e-04
pSB-d	8.01e-02	5.87e-04	1.00e+01	1.29e+01	5.77e-04
aISS	4.94e-02	3.78e-16	1.00e+01	1.29e+01	2.96e-16
FPC	2.75e+00	4.21e-04	1.00e+01	1.29e+01	4.10e-04
FISTA	2.81e+00	5.32e-04	1.00e+01	1.28e+01	4.97e-04
LB-BBLS	2.68e-01	2.96e-04	1.03e+01	1.29e+01	2.91e-04
LB-BFGS	2.23e-01	1.64e-04	1.00e+01	1.29e+01	1.10e-04
LB-LBSB	3.95e-02	5.63e-04	1.00e+01	1.28e+01	5.77e-04

TABLE 2. Noise-free case. Size $A = 100 \times 1000$, 3% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	1.42e-01	1.69e-14	9.37e+01	4.04e+01	4.10e-01
pSB-d	1.42e-01	4.75e-03	9.37e+01	4.03e+01	4.10e-01
aISS	3.01e-01	3.46e-04	9.73e+01	4.03e+01	4.10e-01
FPC	7.17e+00	2.58e-03	9.47e+01	4.03e+01	4.12e-01
FISTA	7.01e+00	2.56e-03	9.43e+01	4.03e+01	4.11e-01
LB-BBLS	4.54e-01	1.14e-03	1.19e+02	4.12e+01	5.72e-01
LB-BFGS	4.19e-01	1.35e-03	1.20e+02	4.12e+01	5.73e-01
LB-LBSB	6.45e-02	1.20e-03	1.20e+02	4.12e+01	5.73e-01

TABLE 3. Noise-free case. Size $A = 900 \times 3000$, 0.9% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	1.59e+00	1.62e-14	2.70e+01	4.30e+01	5.51e-04
pSB-d	1.59e+00	1.61e-03	2.70e+01	4.29e+01	4.70e-04
aISS	3.85e-01	3.47e-15	2.70e+01	4.29e+01	9.13e-16
FPC	3.56e+01	1.04e-03	2.70e+01	4.29e+01	3.21e-04
FISTA	3.62e+01	7.59e-04	2.70e+01	4.29e+01	2.35e-04
LB-BBLS	3.62e+00	9.23e-04	2.70e+01	4.29e+01	2.53e-04
LB-BFGS	3.28e+00	9.01e-04	2.70e+01	4.29e+01	2.40e-04
LB-LBSB	1.22e+00	1.56e-03	2.70e+01	4.29e+01	4.73e-04

Tables 1 to 5 show the results for clean data, while for the results shown in Tables 6 to 10 we consider the noise-added case. The noise is generated from a standard Gaussian distribution and then normalized to have norm ϵ. ϵ is set to be 10^{-3}. For these cases the stopping criteria is changed to $\|Au - f\| \leq \epsilon$. There are other possible ways to conduct the noise-added experiments, and we refer the readers to [21, 22].

As general conclusions we can say, that if moderate accuracy in the data fidelity term is sufficient (which is generic for noisy data), LB-SB beats all the competing methods when the original signal is not quite sparse. Actually, we can see that for a fixed size of A, the speed of LB-SB stays almost the same whenever we increase the density of u_{real} or add noise to f while for other methods we may expect a sharp increase in time cost. The speed of pSB is acceptable, but it is not as efficient as our algorithm. aISS acts pretty well when u_{real} is quite sparse. But even with a slightly increase in the density we may expect a sharp increase in time cost. The FPC with Bregman and FISTA with Bregman always converge quite slowly. Although FISTA

Table 4. Noise-free case. Size $A = 900 \times 3000$, 3% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	3.40e+00	2.99e-14	8.97e+01	1.49e+02	3.48e-04
pSB-d	3.40e+00	1.82e-03	8.97e+01	1.48e+02	3.10e-04
aISS	2.17e+00	1.32e-04	8.97e+01	1.48e+02	2.06e-05
FPC	1.15e+02	3.58e-03	9.03e+01	1.48e+02	6.28e-04
FISTA	1.15e+02	4.35e-03	9.07e+01	1.48e+02	7.60e-04
LB-BBLS	4.14e+00	2.28e-03	8.97e+01	1.48e+02	3.62e-04
LB-BFGS	4.07e+00	1.31e-03	8.97e+01	1.48e+02	2.07e-04
LB-LBSB	1.47e+00	3.04e-03	8.97e+01	1.48e+02	5.44e-04

Table 5. Noise-free case. Size $A = 1000 \times 5000$, 3% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	1.26e+01	5.32e-14	1.53e+02	2.32e+02	5.68e-04
pSB-d	1.26e+01	3.38e-03	1.52e+02	2.31e+02	5.87e-04
aISS	1.53e+01	9.60e-15	1.50e+02	2.31e+02	1.33e-15
FPC	2.44e+02	8.39e-03	1.73e+02	2.31e+02	1.50e-03
FISTA	2.49e+02	8.26e-03	1.75e+02	2.31e+02	1.50e-03
LB-BBLS	2.24e+01	3.44e-03	4.89e+02	2.32e+02	5.93e-03
LB-BFGS	1.35e+01	1.89e-02	3.53e+02	2.32e+02	7.40e-03
LB-LBSB	5.59e+00	3.59e-03	3.89e+02	2.32e+02	4.29e-03

Table 6. Noise-added case. Size $A = 100 \times 1000$, 1% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	9.88e-02	1.00e-03	1.02e+01	7.45e+00	5.75e-03
pSB-d	9.88e-02	2.32e-03	9.80e+00	7.36e+00	5.82e-03
aISS	5.08e-02	3.61e-04	1.00e+01	7.37e+00	6.42e-04
FPC	2.24e+00	3.63e-03	9.40e+00	7.36e+00	8.53e-03
FISTA	2.23e+00	3.64e-03	9.40e+00	7.36e+00	8.51e-03
LB-BBLS	2.57e-01	4.46e-04	1.12e+01	7.37e+00	7.70e-04
LB-BFGS	3.77e-01	3.67e-03	1.62e+01	7.39e+00	7.02e-03
LB-LBSB	4.66e-02	5.70e-04	1.02e+01	7.38e+00	1.10e-03

Table 7. Noise-added case. Size $A = 100 \times 1000$, 3% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	1.40e-01	1.00e-03	9.02e+01	2.01e+01	4.89e-01
pSB-d	1.40e-01	4.30e-03	9.02e+01	2.00e+01	4.89e-01
aISS	2.53e-01	1.30e-03	9.58e+01	2.00e+01	4.86e-01
FPC	3.02e+00	1.17e-02	8.20e+01	1.99e+01	4.75e-01
FISTA	2.80e+00	1.20e-02	8.20e+01	1.99e+01	4.78e-01
LB-BBLS	3.17e-01	1.36e-03	1.07e+02	2.01e+01	5.48e-01
LB-BFGS	3.65e-01	1.88e-03	1.07e+02	2.01e+01	5.48e-01
LB-LBSB	6.80e-02	1.46e-03	1.08e+02	2.01e+01	5.49e-01

is much faster than FPC in solving the unconstrained problem (2.12), we do not see much difference in the time cost comparison for these two methods. This is because in our code we only take 5 inner iterations of FPC and FISTA, and the convergence speed of them can not be reflected by this small number. LB-BBLS

TABLE 8. Noise-added case. Size $A = 900 \times 3000$, 0.9% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	1.98e+00	1.00e-03	2.70e+01	4.30e+01	1.92e-04
pSB-d	1.98e+00	2.55e-04	2.70e+01	4.29e+01	7.35e-05
aISS	3.87e-01	1.61e-04	2.70e+01	4.29e+01	2.62e-05
FPC	3.69e+01	1.94e-04	2.70e+01	4.29e+01	5.49e-05
FISTA	3.92e+01	2.44e-04	2.70e+01	4.29e+01	7.32e-05
LB-BBLS	3.56e+00	2.22e-04	2.70e+01	4.29e+01	6.21e-05
LB-BFGS	5.01e+00	1.10e-02	3.20e+01	4.29e+01	3.04e-03
LB-LBSB	1.53e+00	2.39e-04	2.70e+01	4.29e+01	6.76e-05

TABLE 9. Noise-added case. Size $A = 900 \times 3000$, 3% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	2.46e+00	1.00e-02	9.70e+01	1.49e+02	1.12e-03
pSB-d	2.46e+00	4.23e-03	9.00e+01	1.48e+02	6.48e-04
aISS	2.07e+00	3.18e-03	9.20e+01	1.48e+02	4.81e-04
FPC	1.25e+02	3.01e-03	9.20e+01	1.48e+02	5.18e-04
FISTA	1.25e+02	2.99e-03	9.20e+01	1.48e+02	5.14e-04
LB-BBLS	4.57e+00	3.55e-03	9.20e+01	1.48e+02	5.46e-04
LB-BFGS	5.39e+00	4.78e-02	1.28e+02	1.49e+02	7.57e-03
LB-LBSB	1.43e+00	5.76e-03	1.00e+02	1.48e+02	9.38e-04

TABLE 10. Noise-added case. Size $A = 1000 \times 5000$, 2% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error
pSB-u	8.66e+00	1.00e-02	1.01e+02	1.54e+02	1.15e-03
pSB-d	8.66e+00	4.23e-03	1.01e+02	1.53e+02	7.98e-04
aISS	3.95e+00	3.30e-03	1.04e+02	1.53e+02	5.99e-04
FPC	2.09e+02	6.06e-03	1.07e+01	1.53e+02	7.54e-04
FISTA	2.06e+02	6.06e-03	1.07e+01	1.53e+02	7.52e-04
LB-BBLS	1.33e+01	4.15e-03	1.04e+02	1.53e+02	7.47e-04
LB-BFGS	1.40e+01	3.88e-02	1.79e+02	1.53e+02	7.52e-03
LB-LBSB	3.29e+00	6.45e-03	1.32e+02	1.53e+02	1.27e-03

and LB-BFGS are comparable to our algorithm when A is close to a square matrix. For the case where $m/n < 0.3$, LB-SB is significantly faster than LB-BBLS and LB-BFGS. All the LB based methods tend to give results with acceptable ℓ^1 norm but relative large ℓ^0 norm, since there are small oscillations in these results caused by the ℓ^2 term $\frac{1}{2\alpha}\|u\|^2$ in (2.22).

Let us comment on each single experiment in a little more detail:

(1) In the first experiment (Table 1), all methods were able to reconstruct u_{real} up to small deviations, the largest deviation occurring for pSB with a relative error of 0.06%. The LB-SB algorithm has the fastest runtime, and is around 2 times faster than pSB. The aISS takes a little longer than LB-SB, however, it is (for this low sparsity level of only 10 non-zero entries) quite fast and by far the most accurate method. The other two LB methods are slower than pSB. FPC and FISTA based methods follow with 1-2 orders of magnitude slower runtime.

(2) As we decreased the sparsity level of u_{real} in our second experiment (Table 2), none of the methods was able to reconstruct u_{real} anymore, although they do seem to share some peaks of u_{real}, since the relative error of the ℓ^1 minimizing techniques is around 41% and the error of the elastic net minimizing techniques is around 57%. Although an experiment with a relative error of 41% or more does not seem to have much expressiveness at first glance, we believe that it is still interesting to consider this case. There are many practical applications for which the coincidence between the ℓ^1 and ℓ^0 minimizing solution can not be guaranteed. Therefore, it makes sense to investigate how the ℓ^1 minimization techniques behave in such a case.

    We can see a clear advantage of the LB-SB algorithm in terms of the runtime, outperforming the other LB based methods by more than a factor of 7. The second fastest method, pSB, is still more than 1.5 times slower than LB-SB however, leads to a result which has a smaller ℓ^1 and ℓ^0 norm. Opposed to the first experiment, the elastic net minimizer did not coincide with the ℓ^1 minimizer, leading to small oscillations in the solutions of the LB methods and a higher ℓ^1 norm as well as a lower sparsity of the solution.

(3) Changing the matrix size to 900×3000 and having 27 non-zero entries in u_{real} all methods were able to reconstruct u_{real} exactly again (Table 3). Due to the very high sparsity of the true solution, aISS outperformed all other methods in accuracy as well as in algorithm speed. Nevertheless, LB-SB is the second fastest algorithm being three times as fast as the other LB based methods and leading to a slightly higher accuracy than the pSB method.

(4) Increasing the number of non-zero components of u_{real} to 90, the results change significantly (Table 4). While all methods still reconstruct u_{real} exactly, aISS now takes about 5 times longer than in the previous example while the runtime of LB-SB increases only slightly, making it the fastest algorithm in this case. Still the primal Bregman method takes a little bit longer and is slightly more accurate than LB-SB. The advantage in runtime of LB-SB over LB-BFGS and LB-BBLS is still more than a factor of 3. The FPC and FISTA algorithms are much slower than all other methods and lead to runtimes that are two orders of magnitude slower.

(5) In the last noise free experiment we consider the case where LB based algorithms give relatively dense results (Table 5). pSB and aISS give very good recovery results. FPC with Bregman and FISTA with Bregman took more than 200 seconds to reconstruct the results. In Figure 1 we plot the original u_{real} as well as the results by aISS and LB-SB and the error of LB-SB, which is determined as the absolute value of $(u_{real} - u_{LBSB})$. According to the graph, it is really hard to tell the difference between the original u_{real} and LB-SB result. The relative error between the LB-SB result and u_{real} is only about 0.4% and as we can see in the bottom right part of Figure 1, the peak error is on the order of $\mathbb{O}(10^{-3})$ while the signal is in $\mathbb{O}(1)$. Therefore, we can conclude that the oscillations do not affect the final result much. As for the time cost, we can see that the small sacrifies in accuracy gave us an LB-SB algorithm speed that is about 3 times faster than aISS.

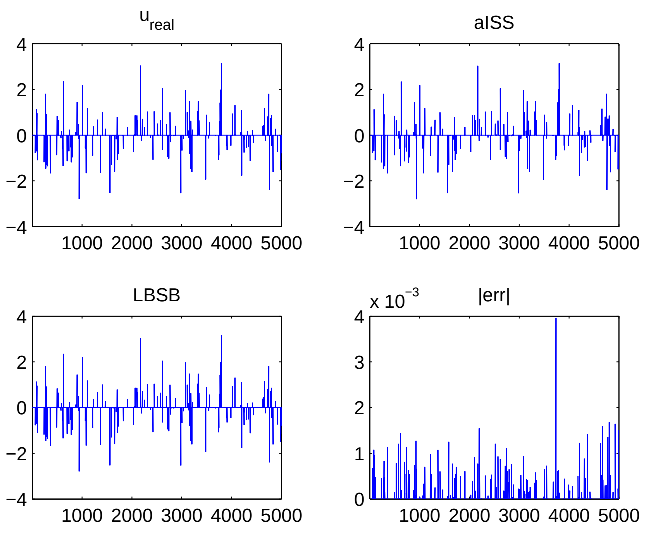

FIGURE 1. Result comparison for Table 5.

(6) In the noise added case, we have a slightly different situation than before: While in the noise free case all methods converge either to the ℓ^1 minimizing or the elastic net minimizing solution, there can now be a difference between the reconstruction results due to the different paths the algorithms take in approaching a solution to $Au = f$.

In the first experiment with noisy data (Table 6), we can see that for high sparsity and the relative low noise level we chose, the reconstruction of u_{real} worked well for all methods. All methods stopped at a solution with 10 non-zero entries having a relative error of less than 1%. In terms of the runtime, LB-SB was the fastest method closely followed by aISS. Among the LB methods, LB-SB was about 5 times faster than LB-BBLS and 7 times faster than LB-BFGS.

(7) Similar to the noise free case the reconstruction of u_{real} gets much more difficult as the sparsity level decreases to 90 non-zero entries (Table 7). Here the ℓ^1 minimizing methods seem to have an advantage in accuracy over the LB methods. Still LB-SB remains the fastest method with almost no increase in runtime in comparison to the previous test case.

(8) For a 900×3000 matrix, 27 non-zero entries in u_{real} and small noise all methods could recover a very good approximation of the true solution (Table 8). In this case aISS outperformed all other methods. Still LB-SB is the second fastest and recovered the support of the true solution exactly.

(9) Again, increasing the number of non-zeros of u_{real} strongly influences the runtime of aISS, such that for 90 non-zeros LB-SB is almost 0.6 second faster than aISS and more than 2.5 seconds faster than the other LB methods (Table 9). We can see that the support of the LB-SB reconstruction

TABLE 11. The α study.

size	$\|x\|_\infty$	$\alpha = 5$		$\alpha = 4\|x\|_\infty$		$\alpha = 7\|x\|_\infty$		$\alpha = 11\|x\|_\infty$	
		Time	rel.err	Time	rel.err	Time	rel.err	Time	rel.err
600×3000	4.16	1.52	5.61e-5	1.58	6.37e-5	1.63	5.78e-5	1.66	6.13e-5
800×4000	5.86	2.61	6.05e-5	2.88	6.25e-5	2.88	6.17e-5	2.93	5.68e-5
1000×5000	4.18	4.37	6.00e-5	4.57	6.29e-5	4.58	6.52e-5	4.64	6.11e-5
1200×6000	6.21	8.07	7.67e-5	8.75	7.55e-5	8.98	7.56e-5	9.59	7.71e-5

has 100 non zero entries, which however seem to be due to very small oscillations, because the relative error shows a reconstruction of u_{real} with less than 0.1% deviation.

(10) In the more computationally expensive case of a 1000×5000 matrix and u_{real} consisting of 100 non-zero entries (Table 10), the run times vary from 3.29 seconds for LB-SB up to 209 seconds for FPC with Bregman iteration. The reconstructed support varies between the almost correct 101 entries for pSB to 179 entries for LB-BFGS. The support of the LB-SB method again is too large, but the relative error still indicates that the additional non-zero components are in a negligible order of magnitude.

According to the above comparison, although $\alpha = 5$ usually is much smaller than the "equivalence bound" $10\|u_{real}\|_\infty$ in [19], the results seem acceptable. In order to study how α affects the performance of our algorithm, we tested LB-SB with different α on different matrices and recorded the runtime and relative error. In all these tests only 2% of u_{real} is nonzero and the nonzero terms are generated by Gaussian distribution with standard deviation 2. The stopping criteria is set as $\|Au - f\| < 5 \cdot 10^{-5}\|f\|$. The results are shown in Table 11.

We can see that the runtime increases as α increases, which agrees with our analysis in Section 4 that smaller α tends to give faster convergence. We can also see that the relative errors always stay in the same level for different α, and this convinces us that in practice an α much smaller than the theoretical bound often yields satisfactory results. Usually, $\alpha = 5$ works well both in our tests and in [24].

5.2. **Nonnegative Case.**

In this section we investigate the algorithms speed on the non-negative ℓ^1 minimization problem (1.3). Furthermore, we will study the influence of α on our algorithm and on LB-BBLS, LB-BFGS. This time the nonzero entries of u_{real} are generated by the absolute value of Gaussian distribution with variance 2. Noise is generated the same way as in the previous subsection with $\epsilon = 10^{-2}$. The stopping criteria is set to be $\|Au - f\| \leq \epsilon$. This time the Matlab function for solving nonnegative least squares (Matlab NNLS), which is based on an active set method, is also included in the comparison. FPC with Bregman iteration and FISTA with Bregman iteration are still stopped after 400 iterations if they have not converged then.

In detail we can conclude the following:

(1) In the first non-negative reconstruction experiment (Table 12), all methods determined u_{real} to a good accuracy with typically less than 1% deviation. The fastest reconstruction result is achieved by LB-SB, where difference between the choice of the parameter $\alpha = 10$ and $\alpha = 3$ is small in terms of reconstruction accuracy as well as run time. Matlabs NNLS algorithm does

TABLE 12. Size $A = 100 \times 1000$, 1% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error	α
pSB-u	9.52e-02	6.27e-03	1.10e+01	1.30e+01	3.51e-03	-
pSB-d	9.52e-02	4.84e-03	1.03e+01	1.29e+01	3.22e-03	-
aISS	4.03e-02	3.17e-03	1.07e+01	1.29e+01	2.00e-03	-
FPC	4.24e-01	3.74e-03	1.13e+01	1.28e+01	2.67e-03	-
FISTA	3.50e-01	3.43e-03	1.13e+01	1.28e+01	2.62e-03	-
LB-BBLS	1.74e-01	7.13e-03	1.80e+01	1.29e+01	6.15e-03	3
LB-BFGS	1.27e+00	1.00e-02	7.93e+01	1.30e+01	1.38e-02	3
LB-LBSB	2.41e-02	8.55e-03	2.40e+01	1.29e+01	7.97e-03	3
LB-BBLS	1.30e-01	4.62e-03	1.17e+01	1.28e+01	3.16e-03	10
LB-BFGS	1.26e+00	1.00e-02	7.33e+01	1.30e+01	8.98e-03	10
LB-LBSB	3.76e-02	8.54e-03	1.73e+01	1.29e+01	6.74e-03	10
Matlab NNLS	3.85e-02	5.30e-03	9.67e+00	1.28e+01	3.36e-03	-

TABLE 13. Size $A = 900 \times 3000$ case, 0.9% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error	α
pSB-u	4.62e+00	5.95e-03	2.70e+01	2.16e+01	1.55e-03	-
pSB-d	4.62e+00	2.04e-03	2.70e+01	2.14e+01	7.42e-04	-
aISS	3.94e-01	1.66e-03	2.70e+01	2.14e+01	5.84e-04	-
FPC	1.50e+01	2.37e-03	2.68e+01	2.14e+01	8.44e-04	-
FISTA	1.44e+01	2.56e-03	2.68e+01	2.14e+01	9.13e-04	-
LB-BBLS	2.22e+00	1.99e-03	2.70e+01	2.14e+01	6.98e-04	3
LB-BFGS	1.63e+01	1.05e-02	2.13e+02	2.19e+01	4.65e-03	3
LB-LBSB	1.46e+00	2.21e-03	2.76e+01	2.14e+01	8.08e-04	3
LB-BBLS	3.28e+00	1.89e-03	2.70e+01	2.14e+01	6.62e-04	10
LB-BFGS	1.63e+01	1.52e-02	2.44e+02	2.19e+01	6.52e-03	10
LB-LBSB	1.40e+00	2.24e-03	2.72e+01	2.14e+01	8.20e-04	10
Matlab NNLS	3.46e-01	3.44e-03	2.66e+01	2.14e+01	1.20e-03	-

TABLE 14. Size $A = 900 \times 3000$ case, 3% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error	α
pSB-u	2.60e+00	6.60e-03	9.00e+01	1.41e+02	6.21e-04	-
pSB-d	2.60e+00	4.32e-03	9.00e+01	1.41e+02	4.82e-04	-
aISS	1.56e+00	3.14e-03	9.00e+01	1.41e+02	3.21e-04	-
FPC	8.82e+00	3.50e-03	9.02e+01	1.41e+02	3.76e-04	-
FISTA	8.64e+00	3.59e-03	9.00e+01	1.41e+02	3.85e-04	-
LB-BBLS	2.12e+00	4.00e-03	9.16e+01	1.41e+02	3.97e-04	3
LB-BFGS	1.44e+01	1.04e-02	2.84e+02	1.41e+02	1.50e-03	3
LB-LBSB	1.37e+00	4.64e-03	9.18e+01	1.41e+02	5.37e-04	3
LB-BBLS	2.65e+00	3.56e-03	9.00e+01	1.41e+02	3.60e-04	10
LB-BFGS	1.54e+01	1.43e-02	2.95e+02	1.41e+02	1.84e-03	10
LB-LBSB	1.45e+00	4.63e-03	9.00e+01	1.41e+02	4.33e-04	10
Matlab NNLS	1.55e+00	4.22e-03	8.98e+01	1.41e+02	4.28e-04	-

not determine the ℓ^1 minimizing solution, but still was able to determine u_{real} fast an accurate.

(2) In our second test (Table 13), all methods gave good approximations for u_{real}, however, with significantly higher computational afford. We can see that aISS is the fastest method for determining non-negative solutions. As

TABLE 15. Size $A = 1000 \times 5000$ case, 2% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error	α
pSB-u	7.73e+00	6.29e-03	1.00e+02	1.53e+02	6.51e-04	-
pSB-d	7.73e+00	4.13e-03	1.00e+02	1.53e+02	5.37e-04	-
aISS	4.00e+00	3.18e-03	1.01e+02	1.53e+02	3.84e-04	-
FPC	2.91e+01	3.89e-03	1.00e+02	1.53e+02	4.89e-04	-
FISTA	2.90e+01	3.79e-03	1.01e+02	1.53e+02	4.81e-04	-
LB-BBLS	6.47e+00	4.19e-03	1.06e+02	1.53e+02	5.21e-04	3
LB-BFGS	2.62e+01	1.01e-02	3.38e+02	1.53e+02	1.94e-03	3
LB-LBSB	3.10e+00	5.57e-03	1.17e+02	1.53e+02	7.74e-04	3
LB-BBLS	6.95e+00	3.67e-03	1.01e+02	1.53e+02	4.27e-04	10
LB-BFGS	2.78e+01	1.17e-02	3.55e+02	1.53e+02	1.83e-03	10
LB-LBSB	3.21e+00	4.96e-03	1.05e+02	1.53e+02	6.74e-04	10
Matlab NNLS	3.67e+00	4.91e-03	9.97e+01	1.53e+02	6.12e-04	-

TABLE 16. Size $A = 1000 \times 5000$ case, 3% of u_{real} is nonzero.

Algorithm	Time	$\|Au - Au_{real}\|$	$\|u\|_0$	$\|u\|_1$	relative error	α
pSB-u	1.57e+01	6.71e-03	1.55e+02	2.32e+02	6.75e-04	-
pSB-d	1.57e+01	5.37e-03	1.55e+02	2.31e+02	6.07e-04	-
aISS	8.42e+00	3.95e-03	1.54e+02	2.31e+02	4.11e-04	-
FPC	5.58e+01	8.24e-03	1.55e+02	2.31e+02	8.98e-04	-
FISTA	5.38e+01	8.13e-03	1.55e+02	2.31e+02	8.87e-04	-
LB-BBLS	9.10e+00	8.72e-03	2.87e+02	2.32e+02	1.51e-03	3
LB-BFGS	2.78e+01	1.00e-02	5.30e+02	2.32e+02	2.57e-03	3
LB-LBSB	3.12e+00	9.14e-03	2.45e+02	2.32e+02	1.31e-03	3
LB-BBLS	1.17e+01	5.83e-03	1.63e+02	2.31e+02	7.15e-04	10
LB-BFGS	2.96e+01	1.10e-02	4.34e+02	2.32e+02	1.67e-03	10
LB-LBSB	3.70e+00	6.38e-03	1.68e+02	2.31e+02	7.71e-04	10
Matlab NNLS	5.24e+00	9.46e-03	1.48e+02	2.31e+02	1.03e-03	-

for the influence of α it is interesting to see that this time the smaller choice of α required more running time for LB-SB.

(3) Even for 180 positive values in u_{real} the reconstruction worked well (Table 14). In this experiment LB-SB is significantly faster than any other method. The results we get for the LB methods also illustrate the behavior for changing the parameter α nicely: While the relative errors to u_{real} increase only a little bit for the smaller value of α the support of the solutions increases significantly. For $\alpha = 10$ LB-BBLS gives a result similar to the ℓ^1 minimizing methods, while for $\alpha = 3$ the solutions support is much larger. However, due to the good relative errors we can conclude that again the increased support is based on very small, negligible oscillations.

(4) For a matrix size of 1000×5000 and 100 positive entries in u_{real} LB-SB beats the other LB methods by almost a factor of 3 in run time (Table 15). Otherwise we can observe similar effects as in the previous cases about the support of the LB solutions in comparison to the ℓ^1 minimizing methods.

(5) Finally, for an increased number of non-zeros in u_{real}, LB-SB remains the fastest method beating all other elastic net or ℓ^1 minimizing methods by at least a factor of 3 (Table 16). In this case the choice of α had very little influence on the runtime of LB-SB, while the larger α gave a sparser result.

Once more LB based methods have a larger support, but only a slightly increased relative error.

In conclusion we can say that on average a larger α tends to give sparser results with the sacrifice in speed, which is to be expected considering the error estimates we derived in Section 4. Considering the little gain in relative error and sparsity and great loss in speed, we need to choose a reasonable α. Usually in application, $\alpha = 5$ will be a simple and good choice.

In comparison to the ℓ^1 minimizing methods, the LB methods result in a larger support, but only very small differences in relative error. Thus, for applications that do not require very high precision solutions, the LB-SB method will be a very good choice due to its advantages in run time and its independence of the sparsity level of the reconstructed solution.

6. Conclusion

We proposed a new method for finding a $J(u) = \|u\|_1 + \frac{1}{2\alpha}\|u\|^2$ minimizing solution to $Au = f$, which for large α coincide with ℓ^1 minimizing solutions. Our idea is based off of the dual formulation to the constrained problem similar to the linearized Bregman approach, but solves the resulting problem using the split Bregman method instead of gradient descent.

We presented a framework for the convergence analysis for split Bregman methods in general and derived particular estimates for the split Bregman methods applied to the primal and dual formulation of our problem. Our analysis leads us to the conclusion, that the split Bregman algorithm can yield different convergence properties on the primal and dual problem in general. Thus, for any minimization problem the choice of the formulation could and should depend on the question which formulation has the more desirable convergence properties. For our specific example, we were able to derive an ℓ^2-type error estimate although we are determining the ℓ^1 minimizing solution.

In an extensive numerical experiment we compared our proposed model to state-of-the-art methods like primal split Bregman, aISS, FPC with Bregman iteration, FISTA with Bregman iteration, LB-BBLS and LB-BFGS. The comparison included ℓ^1 minimization on data with and without noise as well as non-negative ℓ^1 minimization. We can conclude from our experiments that our algorithm is faster than the gradient descent based algorithms LB-BBLS and LB-BFGS, particularly for very under determined matrices. aISS is generally more accurate (particularly in the noise free case) and very fast if the solution is very sparse, but its runtime heavily depends on the sparsity of the true solution while LB-SB does not. The primal split Bregman algorithm pSB is slightly more accurate than LB-SB, but also converges slower. In our experiments, LB-SB on average was by far the fastest method. While there might be some small oscillations in the final solution they are small enough to almost not affect the relative error at all.

7. Acknowledgement

The authors would like to thank Wotao Yin for providing the Matlab code for the LB-BBLS and LB-BFGS algorithms, and Martin Burger for his helpful comments and discussions. The authors would also like to thank the anonymous reviewers for their comments. YY and SO acknowledge the support of ONR grants N00014-11-1-0749 and N00014081119 and NSF grants DMS0835863 and DMS0914561.

References

1. A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci **2** (2009), 183–202.
2. M. Burger, G. Gilboa, S. Osher, and J. Xu, *Nonlinear inverse scale space methods*, Comm. Math. Sci. **4** (2006), 179–212.
3. M. Burger, M. Moeller, M. Benning, and S. Osher, *An adaptive inverse scale space method for compressed sensing*, To appear in Math. Comp.
4. M. Burger, E. Resmerita, and L. He, *Error estimation for Bregman iterations and inverse scale space methods in image restoration*, Computing **81** (2007), 109–135.
5. J. Cai, S. Osher, and Z. Shen, *Linearized Bregman iterations for compressed sensing*, Math. Comp. **78** (2008), no. 267, 1515–1536.
6. ———, *Convergence of the linearized Bregman iteration for ℓ^1-norm minimization*, Math. Comp. **78** (2009), no. 268, 2127–2136.
7. ———, *Split Bregman methods and frame based image restoration*, Multiscale Modeling Simulation **8** (2010), no. 2, 337.
8. E.J. Candes and T. Tao, *Decoding by linear programming*, IEEE Trans. Inform. Theory **51** (2004), 4203–4215.
9. ———, *Near-optimal signal recovery from random projections: universal encoding strategies*, IEEE Trans. Inform. Theory **52** (2004), 5406–5425.
10. D.L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory **52** (2006), 1289–1306.
11. D.L. Donoho and M. Elad, *Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization*, Proc. Natl. Acad. Sci. USA **100** (2003), 2197–2202.
12. I. Ekeland and R. Temam, *Convex analysis and variational problems*, corrected reprint edition ed., SIAM, Philadelphia, 1999.
13. E. Esser, *Primal dual algorithms for convex models and applications to image restoration, registration and nonlocal inpainting*, Ph.D. thesis, UCLA CAM (10-31), June 2010.
14. T. Goldstein and S. Osher, *The split Bregman method for ℓ^1 regularized problems*, SIAM J. Imaging Sci **2** (2009), 323–343.
15. E. T. Hale, W. Yin, and Y. Zhang, *A fixed-point continuation method for ℓ^1-regularized minimization with applications to compressed sensing*, Rice CAM Report TR07-07, July 2007.
16. B. S. He and X. M. Yuan, *On convergence rate of the douglas-rachford operator splitting method*, Preprint, December 2011, online at `http://www.math.hkbu.edu.hk/~xmyuan/Paper/DRSM-Con-Dec13.pdf`.
17. ———, *On nonergodic convergence rate of Douglas-Rachford alternating direction method of multipliers*, Preprint, January 2012, online at `http://www.math.hkbu.edu.hk/~xmyuan/Paper/ADM-HY-Jan16.pdf`.
18. B. Huang, S. Ma, and D. Goldfarb, *Accelerated linearized Bregman method*, Preprint, June 2011, online at `http://www.optimization-online.org/DB_HTML/2011/06/3079.html`.
19. Ming-Jun Lai and Wotao Yin, *Augmented ℓ^1 and nuclear-norm models with a globally linearly convergent algorithm*, Submitted, 2012.
20. M. Moeller, T. Wittman, A.L. Bertozzi, and M. Burger, *A variational approach for sharpening high dimensional images*, J. Imaging Sci. **5** (2012), 150–178.
21. S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, SIAM Multiscale Model. Simul. **4** (2005), 460–489.
22. 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.
23. S. Setzer, *Split Bregman algorithm, douglas-rachford splitting and frame shrinkage*, Proceedings of the Second International Conference on Scale Space and Variational Methods in Computer Vision, SSVM '09, Springer-Verlag, 2009, pp. 464–476.
24. W. Yin, *Analysis and generalizations of the linearized Bregman method*, Preprint, 2009.
25. W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ^1-minimization with applications to compressed sensing*, SIAM J. Imaging Sci. **1** (2008), 143–168.

Department of Mathematics, University of California Los Angeles, Portola Plaza, Los Angeles CA 90095, USA.
E-mail address: yyang@math.ucla.edu

Westfälische Wilhelms-Universität Münster, Institut für Numerische und Angewandte Mathematik, Einsteinstr, 62, D 48149 Münster, Germany
E-mail address: m.moeller@gmx.net

Department of Mathematics, University of California Los Angeles, Portola Plaza, Los Angeles CA 90095, USA.
E-mail address: sjo@math.ucla.edu