Compressive Conjugate Directions: Linear Theory*

Musa Maharramov† and Stewart A. Levin†

Key words. L_1-regularization, total-variation regularization, regularized inversion, ADMM, method of multipliers, conjugate gradients, compressive conjugate directions

AMS subject classifications. 65K05, 90C06

Abstract. We present a powerful and easy-to-implement iterative algorithm for solving large-scale optimization problems that involve L_1/total-variation (TV) regularization. The method is based on combining the Alternating Directions Method of Multipliers (ADMM) with a Conjugate Directions technique in a way that allows reusing conjugate search directions constructed by the algorithm across multiple iterations of the ADMM. The new method achieves fast convergence by trading off multiple applications of the modeling operator for the increased memory requirement of storing previous conjugate directions. We illustrate the new method with a series of imaging and inversion applications.

1. Introduction.

We address a class of regularized least-squares fitting problems of the form

$$\begin{aligned} &\|\mathbf{Bu}\|_1 \;+\; \frac{\alpha}{2}\|\mathbf{Au}-\mathbf{d}\|_2^2 \;\to\; \min, \\ &\mathbf{u}\;\in\;\mathbb{R}^N,\; \mathbf{d}\;\in\;\mathbb{R}^M,\; \mathbf{A}:\mathbb{R}^N\to\mathbb{R}^M,\; \mathbf{B}:\mathbb{R}^N\to\mathbb{R}^K,\; K\le N, \end{aligned} \tag{1.1}$$

where $\mathbf{d}$ is a known vector (data), $\mathbf{u}$ a vector of unknowns[1], and $\mathbf{A},\mathbf{B}$ are linear operators. If $\mathbf{B}$ is the identity map, then problem (1.1) is a least-squares fitting with L_1 regularization,

$$\|\mathbf{u}\|_1 \;+\; \frac{\alpha}{2}\|\mathbf{Au}-\mathbf{d}\|_2^2 \;\to\; \min\,. \tag{1.2}$$

If the unknown vector $\mathbf{u}$ is the discretization of a function, and $\mathbf{B}$ is the first-order finite difference operator

$$(\mathbf{Bu})_i \;=\; u_{i+1}-u_i,\; i=1,2,\ldots,N-1,$$

then problem (1.1) turns into a least-squares fitting with a total-variation (TV) regularization

$$\|\nabla\mathbf{u}\|_1 \;+\; \frac{\alpha}{2}\|\mathbf{Au}-\mathbf{d}\|_2^2 \;\to\; \min\,. \tag{1.3}$$

On the one hand, in (1.2) we seek a model vector $\mathbf{u}$ such that forward-modeled data $\mathbf{Au}$ match observed data $\mathbf{d}$ in the least squares sense, while imposing sparsity-promoting L_1 regularization. In (1.3), on the other hand, we impose blockiness-promoting total-variation (TV) regularization. Note that rather than using a regularization parameter as a coefficient of the regularization term, we use a data-fitting weight α. TV regularization (also known as

*This work was supported by Stanford Exploration Project.
†Department of Geophysics, Stanford University, 397 Panama Mall, Stanford, CA 94305 (musa@sep.stanford.edu, stew@sep.stanford.edu).
[1]sometimes referred to as "model"

the Rudin-Osher-Fatemi, or ROF, model [36]) acts as a form of "model styling" that helps to preserve sharp contrasts and boundaries in the model even when spectral content of input data has a limited resolution.

L_1-TV regularized least-squares fitting, a key tool in imaging and de-noising applications (see, e.g. [36, 10, 42, 26]), is beginning to play an increasingly important role in applications where the modeling operator $\mathbf{A}$ in (1.1) is computationally challenging to apply. In particular, in seismic imaging problems of exploration geophysics such as full-waveform inversion [39, 16] modeling of seismic wave propagation in a three-dimensional medium from multiple seismic sources is by far the greatest contributor to the computational cost of inversion, and reduction of the number of applications of the operator $\mathbf{A}$ is key to success in practical applications.

L_1-regularized least-squares problems can be reduced to inequality-constrained quadratic programs and solved using interior-point methods based on, e.g., Newton [7] or nonlinear Conjugate Gradients [26] methods. Alternatively, the resulting bound-constrained quadratic programs can be solved using gradient projection [17] or projected Conjugate Gradients [33]. A conceptually different class of techniques for solving L_1-regularized least-squares problems is based on homotopy methods [23, 15, 31].

Another class of methods for solving (1.1) that merits a special mention applies splitting schemes for the sum of two operators. For example the *iterative shrinking-thresholding algorithm* (ISTA) is based on applying *forward-backward splitting* [8, 32] to solving the L_1-regularized problem (1.2) by gradient descent [4, 11, 12]:

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{u}_k \;-\; \gamma\alpha\mathbf{A}^T\left(\mathbf{A}\mathbf{u}_k - \mathbf{d}\right), \\ \mathbf{u}_{k+1} &= \operatorname{shrink}\left\{\mathbf{y}_{k+1}, \gamma\right\}, \end{aligned} \tag{1.4}$$

where $\gamma > 0$ is a sufficiently small step parameter, and the *soft thresholding* or *shrinkage* operator is the Moreau resolvent (see, e.g., [1]) of $\partial\gamma\|\mathbf{u}\|_1$,

$$\begin{aligned} \operatorname{shrink}\left\{\mathbf{y}, \gamma\right\} \;=\; \left(1 + \partial\gamma\|\mathbf{y}\|_1\right)^{-1} \;=\; \operatorname{argmin}_{\mathbf{x}}\left\{\gamma\|\mathbf{x}\|_1 + \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2\right\} \;= \\ \frac{\mathbf{y}}{|\mathbf{y}|}\max\left(|\mathbf{y}| - \gamma, 0\right), \end{aligned} \tag{1.5}$$

and $\partial = \partial_{\mathbf{u}}$ denotes the subgradient [34, 1], and the absolute value of a vector is computed component-wise. The typically slow convergence of the first-order method (1.4) can be accelerated by an over-relaxation step [29], resulting in the *Fast ISTA* algorithm (FISTA) [3]:

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{u}_k \;-\; \gamma\alpha\mathbf{A}^T\left(\mathbf{A}\mathbf{u}_k - \mathbf{d}\right), \\ \mathbf{z}_{k+1} &= \operatorname{shrink}\left\{\mathbf{y}_{k+1}, \gamma\right\}, \\ \zeta_{k+1} &= \left(1 + \sqrt{1 + 4\zeta_k^2}\right)/2, \\ \mathbf{u}_{k+1} &= \mathbf{y}_{k+1} + \frac{\zeta_k - 1}{\zeta_{k+1}}\left(\mathbf{y}_{k+1} - \mathbf{y}_k\right), \end{aligned} \tag{1.6}$$

where $\zeta_1 = 1$ and γ is sufficiently small.

It is important to note that algorithm (1.6) is applied to the L_1-regularized problem (1.2), not the TV-regularized problem (1.3). An accelerated algorithm for solving a TV-regularized *denoising problem*[2] was proposed in [2] and applied the Nesterov relaxation [29] to solving the dual of the TV-regularized denoising problem [9]. However, using a similar approach to solving (1.3) with a non-trivial operator $\mathbf{A}$ results in accelerated schemes that still require inversion of $\mathbf{A}$ [2, 21] and thus lack the primary appeal of the accelerated gradient descent methods—i.e., a single application of $\mathbf{A}$ and its transpose per iteration[3].

The advantage of (1.6) compared with simple gradient descent is that Nesterov's over-relaxation step requires storing two previous solution vectors and provides improved search direction for minimization. Note, however, that the step length γ is inversely proportional to the Lipschitz constant of $\alpha\mathbf{A}^T(\mathbf{Au}-\mathbf{d})$ [3] and may be small in practice.

A very general approach to solving problems (1.1) involving either L_1 or TV regularization is provided by primal-dual methods. For example, in TV-regularized least-squares problem (1.3), by substituting

$$\mathbf{z} = \mathbf{Bu} \tag{1.7}$$

and adding (1.7) as a constraint, we obtain an equivalent equality-constrained optimization problem

$$\begin{aligned} \|\mathbf{z}\|_1 \;+\; \frac{\alpha}{2}\|\mathbf{Au}-\mathbf{d}\|_2^2 \;&\rightarrow\; \min, \\ \mathbf{z} \;=\; \mathbf{Bu}&. \end{aligned} \tag{1.8}$$

The optimal solution of (1.8) corresponds to the saddle-point of its Lagrangian

$$L_0(\mathbf{u},\mathbf{z},\boldsymbol{\mu}) \;=\; \|\mathbf{z}\|_1 + \frac{\alpha}{2}\|\mathbf{Au}-\mathbf{d}\|_2^2 + \boldsymbol{\mu}^T(\mathbf{z}-\mathbf{Bu}), \tag{1.9}$$

that can be found by the *Uzawa method* [41]. The Uzawa method finds the saddle point by alternating a minimization with respect to the primal variables $\mathbf{u},\mathbf{z}$ and ascent over the dual variable $\boldsymbol{\mu}$ for the objective function equal to the standard Lagrangian (1.9), $L = L_0$,

$$\begin{aligned} (\mathbf{u}_{k+1},\mathbf{z}_{k+1}) \;&= \operatorname{argmin} L(\mathbf{u},\mathbf{z},\boldsymbol{\mu}_k), \\ \boldsymbol{\mu}_{k+1} \;&= \boldsymbol{\mu}_k \;+\; \lambda[\mathbf{z}_{k+1}-\mathbf{Bu}_{k+1}] \end{aligned} \tag{1.10}$$

for some positive step size λ. Approach (1.10), when applied to the Augmented Lagrangian [35], $L = L_+$,

$$L_+(\mathbf{u},\mathbf{z},\boldsymbol{\mu}) \;=\; \|\mathbf{z}\|_1 + \frac{\alpha}{2}\|\mathbf{Au}-\mathbf{d}\|_2^2 + \boldsymbol{\mu}^T(\mathbf{z}-\mathbf{Bu}) + \frac{\lambda}{2}\|\mathbf{z}-\mathbf{Bu}\|_2^2, \tag{1.11}$$

results in the *method of multipliers* [25]. For problems (1.1) all these methods still require joint minimization with respect to $\mathbf{u}$ and $\mathbf{z}$ of some objective function that includes both $\|\mathbf{z}\|_1$ and a

[2] with $\mathbf{A} = \mathbf{I}$ in (1.3)

[3] In [2] inversion of $\mathbf{A}$ is replaced by a single gradient descent, however, over-relaxation is applied to the dual variable.

smooth function of $\mathbf{u}$. Splitting the joint minimization into separate steps of minimization with respect $\mathbf{u}$, followed by minimization with respect to $\mathbf{z}$, results in the *Alternating-Directions Method of Multipliers* (ADMM) [20, 18, 19, 14, 6]. To establish a connection to the splitting techniques applied to the sum of two operators, we note that the ADMM is equivalent to applying the Douglas-Rachford splitting [13] to the problem

$$\partial \left[\|\mathbf{Bu}\|_1 \; + \; \frac{\alpha}{2} \|\mathbf{Au} - \mathbf{d}\|_2^2 \right] \ni \mathbf{0}, \tag{1.12}$$

where ∂ is the subgradient, and problem (1.12) is equivalent to (1.1). The ADMM is a particular case of a primal-dual iterative solution framework with splitting [43], where the minimization in (1.10) is split into two steps,

$$\begin{aligned} \mathbf{u}_{k+1} \; &= \operatorname{argmin} L\left(\mathbf{u}, \mathbf{z}_k, \boldsymbol{\mu}_k\right), \\ \mathbf{z}_{k+1} \; &= \operatorname{argmin} L\left(\mathbf{u}_{k+1}, \mathbf{z}, \boldsymbol{\mu}_k\right), \\ \boldsymbol{\mu}_{k+1} \; &= \boldsymbol{\mu}_k \; + \; \lambda \left[\mathbf{z}_{k+1} - \mathbf{Bu}_{k+1}\right] \end{aligned} \tag{1.13}$$

For the ADMM, we substitute $L = L_+$ in (1.13) but other choices of a modified Lagrange function L are possible that may produce convergent primal-dual algorithms [43]. Making the substitution $L = L_+$ from (1.11) into (1.13), and introducing a scaled vector of multipliers,

$$\mathbf{b}_k \; = \; \boldsymbol{\mu}_k / \lambda, \; k = 0, 1, 2, \ldots \tag{1.14}$$

we obtain

$$\begin{aligned} \mathbf{u}_{k+1} \; &= \; \operatorname{argmin} \frac{\alpha}{2} \|\mathbf{Au} - \mathbf{d}\|_2^2 \; + \; \frac{\lambda}{2} \|\mathbf{z}_k \; - \; \mathbf{Bu} + \mathbf{b}_k\|_2^2, \\ \mathbf{z}_{k+1} \; &= \; \operatorname{argmin} \|\mathbf{z}\|_1 \; + \; \frac{\lambda}{2} \|\mathbf{z} \; - \; \mathbf{Bu}_{k+1} + \mathbf{b}_k\|_2^2, \\ \mathbf{b}_{k+1} \; &= \; \mathbf{b}_k + \mathbf{z}_{k+1} - \mathbf{Bu}_{k+1}, \; k = 0, 1, 2, \ldots \end{aligned} \tag{1.15}$$

where we used the fact that adding a constant term $\lambda/2 \|\mathbf{b}_k\|_2^2$ to the objective function does not alter the solution. In the iterative process (1.15), we apply splitting, minimizing

$$\|\mathbf{z}\|_1 \; + \; \frac{\alpha}{2} \|\mathbf{Au} - \mathbf{d}\|_2^2 \; + \frac{\lambda}{2} \|\mathbf{z} \; - \; \mathbf{Bu} + \mathbf{b}_k\|_2^2 \tag{1.16}$$

alternately with respect to $\mathbf{u}$ and $\mathbf{z}$. Further we note that the minimization of (1.16) with respect to $\mathbf{z}$ (in a splitting step with $\mathbf{u}$ fixed) is given trivially by the shrinkage operator (1.5),

$$\mathbf{z}_{k+1} \; = \; \operatorname{shrink} \left\{ \mathbf{Bu} - \mathbf{b}_k, 1/\lambda \right\}. \tag{1.17}$$

Combining (1.15) and (1.17) we obtain Algorithm 1.

Minimization on the first line of (1.15) at each step of the ADMM requires inversion of the operator $\mathbf{A}$. In the first-order gradient-descent methods like (1.6) a similar requirement is obviated by replacing the minimization with respect to variable $\mathbf{u}$ by gradient descent. However, for ill-conditioned problems the gradient may be a poor approximation to the optimal search direction. One interpretation of Nesterov's over-relaxation step in (1.6) is that it

Algorithm 1 Alternating Direction Method of Multipliers (ADMM) for (1.1)

1: $\mathbf{u}_0 \leftarrow \mathbf{0}^N$, $\mathbf{z}_0^K \leftarrow \mathbf{0}$
2: $\mathbf{b}_0 \leftarrow \mathbf{0}^K$
3: **for** $k \leftarrow 0, 1, 2, 3, \ldots$ **do**
4: $\quad \mathbf{u}_{k+1} \leftarrow \text{argmin} \left\{ \frac{\lambda}{2} \|\mathbf{z}_k - \mathbf{B}\mathbf{u} + \mathbf{b}_k\|_2^2 + \frac{\alpha}{2} \|\mathbf{A}\mathbf{u} - \mathbf{d}\|_2^2 \right\}$
5: $\quad \mathbf{z}_{k+1} \leftarrow \text{shrink}\{\mathbf{B}\mathbf{u}_{k+1} - \mathbf{b}_k, 1/\lambda\}$
6: $\quad \mathbf{b}_{k+1} \leftarrow \mathbf{b}_k + \mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}$
7: $\quad$ Exit loop if $\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_2 / \|\mathbf{u}_k\|_2 \leq$ target accuracy
8: **end for**

provides a better search direction by perturbing the current solution update with a fraction of the previous update on the last line of (1.6). The intermediate least-squares problem in (1.15) can be solved approximately using, for example, a few iterations of conjugate gradients. However, repeating multiple iterations of Conjugate Gradients at each step of the ADMM may be unnecessary. Indeed, as we demonstrate in the following sections, conjugate directions constructed at earlier steps of the ADMM can be reused because the matrix of the system of normal equations associated with the minimization on the first line of (1.15) does not change between ADMM steps[4]. Therefore, we can trade the computational cost of applying the operator $\mathbf{A}$ and its transpose against the cost of storing a few solution and data-size vectors. As this approach is applied to the most general problem (1.1) with a non-trivial operator $\mathbf{B}$, in addition to the potential speed-up, this method has the advantage of working equally well for L_1 and TV-regularized problems.

We stress that our new approach does not improve the theoretical convergence properties of the classic ADMM method under the assumption of exact minimization in step 4 of Algorithm 1. The asymptotic convergence rate is still $O(1/k)$ as with exact minimization [24]. The new approach provides a numerically feasible way of implementing the ADMM for problems where a computationally expensive operator $\mathbf{A}$ precludes accurate minimization in step 4. However, the rate of convergence in the general method of multipliers (1.10) is sensitive to the choice of parameter λ, and an improved convergence rate for some values of λ can be accompanied with more ill-conditioned minimization problems at each step of (1.15) [19]. By employing increasingly more accurate conjugate-directions solution of the minimization problem at each iteration of (1.15) the new method offsets the deteriorating condition of the intermediate least-squares problems, and achieves a faster practical convergence at early iterations.

Practical utility of the ADMM in applications that involve sparsity-promoting (1.2) or edge-preserving (1.3) inversion is often determined by how quickly we can resolve sparse or blocky model components. These features can often be *qualitatively* resolved within relatively few initial iterations of the ADMM (see discussion in the Appendix of [22]). In our Section 4, fast recovery of such *local* features will be one of the key indicators for judging the efficiency of the proposed method.

In the next section we describe two new algorithms, *Steered* and *Compressive Conjugate*

[4]Only the right-hand sides of the system are updated as a result of thresholding.

Gradients based on the principle of reusing conjugate directions for multiple right-hand sides. In Section 3 we prove convergence and demonstrate that the new algorithm coincides with the exact ADMM in a finite number of iterations. Section 4 contains a practical implementation of the Compressive Conjugate Gradients method. We test the method on a series of problems from imaging and mechanics, and compare its performance against FISTA and ADMM with gradient descent and restarted conjugate gradients.

2. Steered and Compressive Conjugate Directions. Step 4 of Algorithm 1 is itself a least-squares optimization problem of the form

$$\|\mathbf{Fu} - \mathbf{v}_k\|_2^2 \to \min, \tag{2.1}$$

where

$$\mathbf{F} = \begin{bmatrix} \sqrt{\alpha}\mathbf{A} \\ \sqrt{\lambda}\mathbf{B} \end{bmatrix} \tag{2.2}$$

and

$$\mathbf{v}_k = \begin{bmatrix} \sqrt{\alpha}\mathbf{d} \\ \sqrt{\lambda}\,(\mathbf{z}_k + \mathbf{b}_k) \end{bmatrix} \tag{2.3}$$

Solving optimization problem (2.1) is mathematically equivalent to solving the following system of normal equations [40],

$$\mathbf{F}^T\mathbf{Fu} = \mathbf{F}^T\mathbf{v}_k, \tag{2.4}$$

as operator (2.2) has maximum rank. Solving (2.4) has the disadvantage of squaring the condition number of operator (2.2) [40]. When the operator $\mathbf{A}$ is available in a matrix form, and a factorization of operator $\mathbf{F}$ is numerically feasible, solving the normal equations (2.4) should be avoided and a technique based on a matrix factorization should be applied directly to solving (2.1) [5, 37]. However, when matrix $\mathbf{A}$ is not known explicitly or its size exceeds practical limitations of direct methods, as is the case in applications of greatest interest for us, an iterative algorithm, such as the Conjugate Gradients for Normal Equations (CGNE) [5, 37], can be used to solve (2.4). Solving (2.1) exactly may be unnecessary and we can expect that for large-scale problems only a few steps of an iterative method need be carried out. However, every iteration typically requires the application of operator $\mathbf{A}$ and its adjoint, and in large-scale optimization problems we are interested in minimizing the number of applications of these operations. For large-scale optimization problems we need an alternative to re-starting an iterative solver for each intermediate problem (2.1). We propose to minimize restarting iterations[5] by devising a conjugate-directions technique for solving (2.1) with a non-stationary right-hand side. At each iteration of the proposed algorithm we find a search direction that is conjugate to previous directions with respect to the operator $\mathbf{F}^T\mathbf{F}$. In the existing conjugate direction techniques, iteratively constructed conjugate directions span the Krylov subspaces [40],

$$\mathcal{K}_k = \operatorname{span}\left\{\mathbf{F}^T\mathbf{v}_0, \left(\mathbf{F}^T\mathbf{F}\right)\mathbf{F}^T\mathbf{v}_0, \ldots, \left(\mathbf{F}^T\mathbf{F}\right)^k\mathbf{F}^T\mathbf{v}_0\right\},\ k = 0, 1, \ldots\ . \tag{2.5}$$

[5] avoiding restarting altogether in the theoretical limit of infinite computer storage

However, in our approach we construct a sequence of vectors (search directions) that are conjugate with respect to operator $\mathbf{F}^T\mathbf{F}$ at the kth step but may not span the Krylov subspace $\mathcal{K}_k$. This complicates convergence analysis of our technique, but allows "steering" search directions by iteration-dependent right-hand sides. Since the right-hand side in (2.1) is the result of the shrinkage (1.17) at previous iterations that steer or compress the solution, we call our approach "steered" or "compressive" conjugate directions.

For the least-squares problem (2.1), we construct two sets of vectors for $k = 0, 1, 2, \ldots$

$$\begin{aligned} &\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k\}, \ \{\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k\}, \\ &\mathbf{q}_i \ = \ \mathbf{F}\mathbf{p}_i, \ i = 0, 1, 2, \ldots, k, \end{aligned} \tag{2.6}$$

such that

$$\mathbf{q}_i^T \mathbf{q}_j \ = \ \mathbf{p}_i^T \mathbf{F}^T \mathbf{F} \mathbf{p}_j \ = \ 0 \text{ if } i \ \neq \ j. \tag{2.7}$$

Equations (2.6) and (2.7) mean that the vectors $\mathbf{p}_i$ form *conjugate directions* [40, 37]. At each iteration we find an approximation $\mathbf{u}_k$ to the solution of (2.1) as a linear combination of vectors $\mathbf{p}_i, i = 0, 1, \ldots, k$, for which the residual

$$\mathbf{r}_{k+1} \ = \ \mathbf{v}_{k+1} \ - \ \mathbf{F}\mathbf{u}_{k+1}, \tag{2.8}$$

is orthogonal to vectors $\mathbf{q}_i$,

$$\mathbf{q}_i^T \mathbf{r}_{k+1} \ = \ \mathbf{q}_i^T \left(\mathbf{v}_{k+1} \ - \ \mathbf{F}\mathbf{u}_{k+1}\right) \ = \ 0, \ i = 0, 1, \ldots, k. \tag{2.9}$$

Vector $\mathbf{p}_k$ is constructed as a linear combination of *all* previous vectors $\mathbf{p}_i, i = 0, 1, \ldots, k$ and $\mathbf{F}^T\mathbf{r}_k$ so that the conjugacy condition in (2.6) is satisfied. The resulting algorithm for *arbitrary* $\mathbf{v}_k$ depending on k is given by Algorithm 2.

Note that the above algorithm is not specific to a particular sequence of right-hand-side vectors $\mathbf{v}_k$ and its applicability goes beyond solving the constrained optimization problems (1.8). The algorithm requires storing $2k + 2$ vectors (2.6), as well as one vector each for the current solution iterate $\mathbf{u}_k$, variable right-hand side $\mathbf{v}_k$, intermediate vectors $\mathbf{w}_k$ and $\mathbf{s}_k$. The requirement of storing a growing number of vectors makes the algorithm resemble the GMRES method [37] for solving linear systems with non-self-adjoint operators. However, in our case, this is a consequence of having a variable right-hand side, requiring re-computation of solution iterates as linear combinations of all of the previous search directions (2.6). This requirement can be relaxed in applications where vector $\mathbf{v}_k$ is updated, for example, by the modified Lagrangian technique for solving a constrained optimization problem, and converges to a limit. In Section 4 we describe practical applications of the algorithm achieving fast convergence while storing only a subset of vectors (2.6). The algorithm requires one application of $\mathbf{F}$ and its transpose at each iteration and $2k + 3$ dot-products of large vectors.

Combining Algorithms 1 and 2 we obtain the *Compressive Conjugate Directions* Algorithm 3.

Algorithm 2 Steered Conjugate Directions for solving (2.1)

$\mathbf{u}_0 \leftarrow \mathbf{0}^N$
$\mathbf{p}_0 \leftarrow \mathbf{F}^T\mathbf{v}_0,\ \mathbf{q}_0 \leftarrow \mathbf{F}\mathbf{p}_0,\ \delta_0 \leftarrow \mathbf{q}_0^T\mathbf{q}_0$
for $k = 0, 1, 2, 3, \ldots$ **do**
  for $i = 0, 1, \ldots, k$ **do**
    $\tau_i \leftarrow \mathbf{q}_i^T\mathbf{v}_k/\delta_i$
  end for
  $\mathbf{u}_{k+1} \leftarrow \sum_{i=0}^k \tau_i\mathbf{p}_i$
  $\mathbf{r}_{k+1} \leftarrow \mathbf{v}_{k+1} - \sum_{i=0}^k \tau_i\mathbf{q}_i$
  $\mathbf{w}_{k+1} \leftarrow \mathbf{F}^T\mathbf{r}_{k+1}$
  $\mathbf{s}_{k+1} \leftarrow \mathbf{F}\mathbf{w}_{k+1}$
  for $i = 0, 1, \ldots, k$ **do**
    $\beta_i \leftarrow -\mathbf{q}_i^T\mathbf{s}_{k+1}/\delta_i$
  end for
  $\mathbf{p}_{k+1} \leftarrow \sum_{i=0}^k \beta_i\mathbf{p}_i + \mathbf{w}_{k+1}$
  $\mathbf{q}_{k+1} \leftarrow \sum_{i=0}^k \beta_i\mathbf{q}_i + \mathbf{s}_{k+1}$
  $\delta_{k+1} \leftarrow \mathbf{q}_{k+1}^T\mathbf{q}_{k+1}$
  if $\delta_{k+1} = 0$ **then** ▷ Use condition "$\delta_{k+1} <$ tolerance" in practice
    $\delta_{k+1} \leftarrow 1,\ \mathbf{p}_{k+1} \leftarrow \mathbf{0}^N,\ \mathbf{q}_{k+1} \leftarrow \mathbf{0}^{M+K}$
  end if
  Exit loop if $\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_2/\|\mathbf{u}_k\|_2 \le$ target accuracy
end for

3. Convergence Analysis. Convergence properties of the ADMM were studied in many publications and are well known. However, here we provide a self-contained proof of convergence for Algorithm 1 that mostly follows the presentation of [6]. Later, we use this result to study the convergence of Algorithm 3.

Theorem 3.1. *Assume that $M \ge N$, operators $\mathbf{A}$, $\mathbf{B}$ are maximum rank, and*

$$\begin{aligned} \mathbf{u} &= \mathbf{u}^*, \\ \mathbf{z} &= \mathbf{z}^* = \mathbf{B}\mathbf{u}^*, \end{aligned} \tag{3.1}$$

is the unique solution of problem (1.8). Assume that a vector $\mathbf{b}^*$ is defined as

$$\mathbf{b}^* = \boldsymbol{\mu}^*/\lambda, \tag{3.2}$$

where $\boldsymbol{\mu}^*$ is the vector of Lagrange multipliers for the equality constraint in (1.8). Algorithm 1 then converges to this solution if $\lambda > 0$, that is,

$$\mathbf{u}_k \to \mathbf{u}^*,\ \mathbf{z}_k \to \mathbf{z}^*,\ \mathbf{b}_k \to \mathbf{b}^*,\ k \to \infty. \tag{3.3}$$

Proof. Problem (1.8) has a convex objective function and equality constraints, hence (3.1,3.2) is a saddle point of its Lagrangian (1.9) [7]. Substituting $\mathbf{z}_{k+1}, \mathbf{u}_{k+1}$ from Algorithm 1,

Algorithm 3 Compressive Conjugate Directions for (1.1)

1: $\mathbf{u}_0 \leftarrow \mathbf{0}^N$, $\mathbf{z}_0 \leftarrow \mathbf{0}^K$; $\mathbf{b}_0 \leftarrow \mathbf{0}^K$, $\mathbf{v}_0 \leftarrow \begin{bmatrix} \sqrt{\alpha}\mathbf{d} \\ \sqrt{\lambda}\,(\mathbf{z}_0 + \mathbf{b}_0) \end{bmatrix}$
2: $\mathbf{p}_0 \leftarrow \mathbf{F}^T\mathbf{v}_0$, $\mathbf{q}_0 \leftarrow \mathbf{F}\mathbf{p}_0$, $\delta_0 \leftarrow \mathbf{q}_0^T\mathbf{q}_0$
3: **for** $k = 0, 1, 2, 3, \ldots$ **do**
4: **for** $i = 0, 1, \ldots, k$ **do**
5: $\tau_i \leftarrow \mathbf{q}_i^T\mathbf{v}_k/\delta_i$
6: **end for**
7: $\mathbf{u}_{k+1} \leftarrow \sum_{i=0}^{k} \tau_i \mathbf{p}_i$
8: $\mathbf{z}_{k+1} \leftarrow \text{shrink}\,\{\mathbf{B}\mathbf{u}_{k+1} - \mathbf{b}_k, 1/\lambda\}$
9: $\mathbf{b}_{k+1} \leftarrow \mathbf{b}_k + \mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}$
10: $\mathbf{v}_{k+1} \leftarrow \begin{bmatrix} \sqrt{\alpha}\mathbf{d} \\ \sqrt{\lambda}\,(\mathbf{z}_{k+1} + \mathbf{b}_{k+1}) \end{bmatrix}$
11: $\mathbf{r}_{k+1} \leftarrow \mathbf{v}_{k+1} - \sum_{i=0}^{k} \tau_i \mathbf{q}_i$
12: $\mathbf{w}_{k+1} \leftarrow \mathbf{F}^T\mathbf{r}_{k+1}$
13: $\mathbf{s}_{k+1} \leftarrow \mathbf{F}\mathbf{w}_{k+1}$
14: **for** $i = 0, 1, \ldots, k$ **do**
15: $\beta_i \leftarrow -\mathbf{q}_i^T\mathbf{s}_{k+1}/\delta_i$
16: **end for**
17: $\mathbf{p}_{k+1} \leftarrow \sum_{i=0}^{k} \beta_i \mathbf{p}_i + \mathbf{w}_{k+1}$
18: $\mathbf{q}_{k+1} \leftarrow \sum_{i=0}^{k} \beta_i \mathbf{q}_i + \mathbf{s}_{k+1}$
19: $\delta_{k+1} \leftarrow \mathbf{q}_{k+1}^T\mathbf{q}_{k+1}$
20: **if** $\delta_{k+1} = 0$ **then** ▷ Use condition "$\delta_{k+1} <$ tolerance" in practice
21: $\delta_{k+1} \leftarrow 1$, $\mathbf{p}_{k+1} \leftarrow \mathbf{0}^N$, $\mathbf{q}_{k+1} \leftarrow \mathbf{0}^{M+K}$
22: **end if**
23: Exit loop if $\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_2/\|\mathbf{u}_k\|_2 \leq$ target accuracy
24: **end for**

we have

$$
\begin{aligned}
& L_0\left(\mathbf{z}^*, \mathbf{u}^*, \boldsymbol{\mu}^*\right) \leq L_0\left(\mathbf{z}_{k+1}, \mathbf{u}_{k+1}, \boldsymbol{\mu}^*\right) \iff \\
p^* = \|\mathbf{B}\mathbf{u}^*\|_1 + \frac{\alpha}{2}\|\mathbf{A}\mathbf{u}^* - \mathbf{d}\|_2^2 &= \|\mathbf{z}^*\|_1 + \frac{\alpha}{2}\|\mathbf{A}\mathbf{u}^* - \mathbf{d}\|_2^2 \leq \\
& \|\mathbf{z}_{k+1}\|_1 + \frac{\alpha}{2}\|\mathbf{A}\mathbf{u}_{k+1} - \mathbf{d}\|_2^2 + \boldsymbol{\mu}^{*T}\left(\mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}\right) = \\
& p_{k+1} + \boldsymbol{\mu}^{*T}\left(\mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}\right) = p_{k+1} + \lambda \mathbf{b}^{*T}\left(\mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}\right),
\end{aligned}
\tag{3.4}
$$

where p^* is the optimal value of the objective function and p_{k+1} is its approximation at iteration k of the algorithm. Inequality (3.4) provides a lower bound for the objective function estimate p_{k+1}. Step 4 of the algorithm is equivalent to

$$
\alpha\mathbf{A}^T\mathbf{A}\mathbf{u}_{k+1} + \lambda\mathbf{B}^T\mathbf{B}\mathbf{u}_{k+1} = \alpha\mathbf{A}^T\mathbf{d} + \lambda\mathbf{B}^T\left(\mathbf{z}_k + \mathbf{b}_k\right). \tag{3.5}
$$

Substituting the expression for $\mathbf{b}_k$ from steps 6 into (3.5), we obtain

$$
\alpha\mathbf{A}^T\mathbf{A}\mathbf{u}_{k+1} = \alpha\mathbf{A}^T\mathbf{d} + \lambda\mathbf{B}^T\left(\mathbf{z}_k - \mathbf{z}_{k+1} + \mathbf{b}_{k+1}\right). \tag{3.6}
$$

Equality (3.6) is equivalent to

$$\mathbf{u}_{k+1} \;=\; \operatorname{argmin} \frac{\alpha}{2}\|\mathbf{A}\mathbf{u}-\mathbf{d}\|_2^2 \;-\; \lambda\,(\mathbf{z}_k-\mathbf{z}_{k+1}+\mathbf{b}_{k+1})^T\,\mathbf{B}\mathbf{u}. \tag{3.7}$$

Substituting $\mathbf{u}_{k+1}$ and $\mathbf{u}^*$ into the right-hand side of (3.7), we obtain

$$\begin{aligned}\frac{\alpha}{2}\|\mathbf{A}\mathbf{u}_{k+1}-\mathbf{d}\|_2^2 \;\le\; &\frac{\alpha}{2}\|\mathbf{A}\mathbf{u}^*-\mathbf{d}\|_2^2 \;+\\ &\lambda\,(\mathbf{z}_k-\mathbf{z}_{k+1}+\mathbf{b}_{k+1})^T\,\mathbf{B}\,(\mathbf{u}_{k+1}-\mathbf{u}^*)\,.\end{aligned} \tag{3.8}$$

Step 5 is equivalent to

$$\begin{aligned}\mathbf{0} \;\in\; \partial_{\mathbf{z}}\|\mathbf{z}\|_1 \;+\; \lambda\,(\mathbf{z}_{k+1}-\mathbf{B}\mathbf{u}_{k+1}+\mathbf{b}_k) \;&=\; \partial_{\mathbf{z}}\|\mathbf{z}\|_1 \;+\; \lambda b_{k+1},\\ \mathbf{z}_{k+1} \;&=\operatorname{argmin}\left\{\|\mathbf{z}\|_1 \;+\; \lambda\mathbf{b}_{k+1}^T\mathbf{z}\right\},\end{aligned} \tag{3.9}$$

where we used the expression for $\mathbf{b}_k$ from step 6. Substituting $\mathbf{z}=\mathbf{z}_{k+1}$ and $\mathbf{z}=\mathbf{z}^*$ into the right-hand side of the second line of (3.9), we obtain

$$\|\mathbf{z}_{k+1}\|_1 \;\le\; \|\mathbf{z}^*\|_1 \;+\; \lambda\mathbf{b}_{k+1}^T\,(\mathbf{z}^*-\mathbf{z}_{k+1})\,. \tag{3.10}$$

Adding (3.8) and (3.10), we get

$$\begin{aligned}p_{k+1} \;\le p^* \;+\; \lambda\mathbf{b}_{k+1}^T\,(\mathbf{z}^*-\mathbf{z}_{k+1}) \,+\\ \lambda\,(\mathbf{z}_k-\mathbf{z}_{k+1}+\mathbf{b}_{k+1})^T\,\mathbf{B}\,(\mathbf{u}_{k+1}-\mathbf{u}^*)\,,\end{aligned} \tag{3.11}$$

an upper bound for p_{k+1}. Adding (3.4) and (3.11), we get

$$\begin{aligned}0 \;\le \lambda\mathbf{b}^{*T}\,(\mathbf{z}_{k+1}-\mathbf{B}\mathbf{u}_{k+1}) \;+\; \lambda\mathbf{b}_{k+1}^T\,(\mathbf{z}^*-\mathbf{z}_{k+1}) \,+\\ \lambda\,(\mathbf{z}_k-\mathbf{z}_{k+1}+\mathbf{b}_{k+1})^T\,\mathbf{B}\,(\mathbf{u}_{k+1}-\mathbf{u}^*)\,,\end{aligned} \tag{3.12}$$

or after rearranging,

$$\begin{aligned}0 \;\le \lambda\,(\mathbf{b}^*-\mathbf{b}_{k+1})^T\,(\mathbf{z}_{k+1}-\mathbf{B}\mathbf{u}_{k+1}) \;-\; \lambda\,(\mathbf{z}_k-\mathbf{z}_{k+1})^T\,(\mathbf{z}_{k+1}-\mathbf{B}\mathbf{u}_{k+1}) \;+\\ \lambda\,(\mathbf{z}_k-\mathbf{z}_{k+1})^T\,(\mathbf{z}_{k+1}-\mathbf{z}^*)\,.\end{aligned} \tag{3.13}$$

We will now use (3.13) to derive an upper estimate for

$$\|\mathbf{b}_k-\mathbf{b}^*\|_2^2 \;+\; \|\mathbf{z}_k-\mathbf{z}^*\|_2^2.$$

Using step 6 of Algorithm 1 for the first term in (3.13) and introducing $\boldsymbol{\rho}_{k+1} = \mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}$, we get

$$
\begin{aligned}
&\lambda\left(\mathbf{b}^* - \mathbf{b}_{k+1}\right)^T \boldsymbol{\rho}_{k+1} \;= \\
&\lambda\left(\mathbf{b}^* - \mathbf{b}_k - \boldsymbol{\rho}_{k+1}\right)^T \boldsymbol{\rho}_{k+1} \;=\; \lambda\left(\mathbf{b}^* - \mathbf{b}_k\right)^T \boldsymbol{\rho}_{k+1} - \lambda\|\boldsymbol{\rho}_{k+1}\|_2^2 \;= \\
&\lambda\left(\mathbf{b}^* - \mathbf{b}_k\right)^T \left(\mathbf{b}_{k+1} - \mathbf{b}_k\right) - \frac{\lambda}{2}\|\boldsymbol{\rho}_{k+1}\|_2^2 - \frac{\lambda}{2}\|\boldsymbol{\rho}_{k+1}\|_2^2 \;= \\
&\lambda\left(\mathbf{b}^* - \mathbf{b}_k\right)^T \left(\mathbf{b}_{k+1} - \mathbf{b}_k\right) - \frac{\lambda}{2}\|\boldsymbol{\rho}_{k+1}\|_2^2 - \frac{\lambda}{2}\left(\mathbf{b}_{k+1} - \mathbf{b}_k\right)^T \left(\mathbf{b}_{k+1} - \mathbf{b}_k\right) \;= \\
&-\lambda\left(\mathbf{b}_k - \mathbf{b}^*\right)^T \left[\left(\mathbf{b}_{k+1} - \mathbf{b}^*\right) - \left(\mathbf{b}_k - \mathbf{b}^*\right)\right] - \frac{\lambda}{2}\|\boldsymbol{\rho}_{k+1}\|_2^2 - \\
&\frac{\lambda}{2}\left[\left(\mathbf{b}_{k+1} - \mathbf{b}^*\right) - \left(\mathbf{b}_k - \mathbf{b}^*\right)\right]^T \left[\left(\mathbf{b}_{k+1} - \mathbf{b}^*\right) - \left(\mathbf{b}_k - \mathbf{b}^*\right)\right] \;= \\
&\frac{\lambda}{2}\|\mathbf{b}_k - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{b}_{k+1} - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\boldsymbol{\rho}_{k+1}\|_2^2.
\end{aligned}
\tag{3.14}
$$

Substituting (3.14) into (3.13), we obtain

$$
\begin{aligned}
0 \;\leq\; &\frac{\lambda}{2}\|\mathbf{b}_k - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{b}_{k+1} - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\boldsymbol{\rho}_{k+1}\|_2^2 \;-\; \lambda\left(\mathbf{z}_k - \mathbf{z}_{k+1}\right)^T \boldsymbol{\rho}_{k+1} + \\
&\lambda\left(\mathbf{z}_k - \mathbf{z}_{k+1}\right)^T \left(\mathbf{z}_{k+1} - \mathbf{z}^*\right) \;= \\
&\frac{\lambda}{2}\|\mathbf{b}_k - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{b}_{k+1} - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\boldsymbol{\rho}_{k+1}\|_2^2 \;-\; \lambda\left(\mathbf{z}_k - \mathbf{z}_{k+1}\right)^T \boldsymbol{\rho}_{k+1} + \\
&\lambda\left(\mathbf{z}_k - \mathbf{z}_{k+1}\right)^T \left[\left(\mathbf{z}_{k+1} - \mathbf{z}_k\right) + \left(\mathbf{z}_k - \mathbf{z}^*\right)\right] \;= \\
&\frac{\lambda}{2}\|\mathbf{b}_k - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{b}_{k+1} - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\boldsymbol{\rho}_{k+1}\|_2^2 \;-\; \lambda\left(\mathbf{z}_k - \mathbf{z}_{k+1}\right)^T \boldsymbol{\rho}_{k+1} - \\
&\lambda\left(\mathbf{z}_k - \mathbf{z}_{k+1}\right)^T \left(\mathbf{z}_k - \mathbf{z}_{k+1}\right) + \lambda\left(\mathbf{z}_k - \mathbf{z}_{k+1}\right)^T \left(\mathbf{z}_k - \mathbf{z}^*\right) \;= \\
&\frac{\lambda}{2}\|\mathbf{b}_k - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{b}_{k+1} - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\left(\mathbf{z}_k - \mathbf{z}_{k+1} + \boldsymbol{\rho}_{k+1}\right)^T \left(\mathbf{z}_k - \mathbf{z}_{k+1} + \boldsymbol{\rho}_{k+1}\right) \;- \\
&\frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k+1}\|_2^2 + \lambda\left(\mathbf{z}_k - \mathbf{z}_{k+1}\right)^T \left(\mathbf{z}_k - \mathbf{z}^*\right) \;= \\
&\frac{\lambda}{2}\|\mathbf{b}_k - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{b}_{k+1} - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k+1} + \boldsymbol{\rho}_{k+1}\|_2^2 - \frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k+1}\|_2^2 + \\
&\lambda\left[\left(\mathbf{z}_k - \mathbf{z}^*\right) - \left(\mathbf{z}_{k+1} - \mathbf{z}^*\right)\right]^T \left(\mathbf{z}_k - \mathbf{z}^*\right) \;= \\
&\frac{\lambda}{2}\|\mathbf{b}_k - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{b}_{k+1} - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k+1} + \boldsymbol{\rho}_{k+1}\|_2^2 \;- \\
&\frac{\lambda}{2}\|\left(\mathbf{z}_k - \mathbf{z}^*\right) - \left(\mathbf{z}_{k+1} - \mathbf{z}^*\right)\|_2^2 + \lambda\left[\left(\mathbf{z}_k - \mathbf{z}^*\right) - \left(\mathbf{z}_{k+1} - \mathbf{z}^*\right)\right]^T \left(\mathbf{z}_k - \mathbf{z}^*\right) \;= \\
&\frac{\lambda}{2}\|\mathbf{b}_k - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{b}_{k+1} - \mathbf{b}^*\|_2^2 - \frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k+1} + \boldsymbol{\rho}_{k+1}\|_2^2 \;- \\
&\frac{\lambda}{2}\|\mathbf{z}_{k+1} - \mathbf{z}^*\|_2^2 + \frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{z}^*\|_2^2,
\end{aligned}
\tag{3.15}
$$

yielding

$$\begin{aligned}&\frac{\lambda}{2}\|\mathbf{z}_k-\mathbf{z}_{k+1}+\boldsymbol{\rho}_{k+1}\|_2^2\ \le\\&\frac{\lambda}{2}\left(\|\mathbf{z}_k-\mathbf{z}^*\|_2^2+\|\mathbf{b}_k-\mathbf{b}^*\|_2^2\right)-\frac{\lambda}{2}\left(\|\mathbf{z}_{k+1}-\mathbf{z}^*\|_2^2+\|\mathbf{b}_{k+1}-\mathbf{b}^*\|_2^2\right).\end{aligned}\tag{3.16}$$

Expanding the left-hand side of (3.16), we obtain

$$\begin{aligned}&\frac{\lambda}{2}\left(\|\mathbf{z}_k-\mathbf{z}_{k+1}\|_2^2+2\left(\mathbf{z}_k-\mathbf{z}_{k+1}\right)^T\boldsymbol{\rho}_{k+1}+\|\boldsymbol{\rho}_{k+1}\|_2^2\right)\ \le\\&\frac{\lambda}{2}\left(\|\mathbf{z}_k-\mathbf{z}^*\|_2^2+\|\mathbf{b}_k-\mathbf{b}^*\|_2^2\right)-\frac{\lambda}{2}\left(\|\mathbf{z}_{k+1}-\mathbf{z}^*\|_2^2+\|\mathbf{b}_{k+1}-\mathbf{b}^*\|_2^2\right).\end{aligned}\tag{3.17}$$

Let us prove that the middle term in the left-hand side of (3.17) is non-negatve,

$$0\ \le\ \left(\mathbf{z}_k-\mathbf{z}_{k+1}\right)^T\boldsymbol{\rho}_{k+1}\ =\ \left(\mathbf{z}_k-\mathbf{z}_{k+1}\right)^T\left(\mathbf{b}_{k+1}-\mathbf{b}_k\right)$$

where we used step 6 of Algorithm 1. Indeed, since $\mathbf{z}_{k+1}$ minimizes (1.16) with $\mathbf{u}=\mathbf{u}_{k+1}$, using the convexity of L_1 norm, we have for $\mathbf{z}=\mathbf{z}_{k+1}$,

$$\begin{aligned}&\partial_z\frac{\lambda}{2}\|\mathbf{z}-\mathbf{B}\mathbf{u}_{k+1}+\mathbf{b}_k\|_2^2=\lambda\left(\mathbf{z}-\mathbf{B}\mathbf{u}_{k+1}+\mathbf{b}_k\right)\in-\partial\|\mathbf{z}\|_1\ \Rightarrow\\&\|\mathbf{z}_{k+1}\|_1-\|\mathbf{z}_k\|_1\ \le\ \left(\mathbf{z}_k-\mathbf{z}_{k+1}\right)^T\left(\mathbf{z}_{k+1}-\mathbf{B}\mathbf{u}_{k+1}+\mathbf{b}_k\right)=\left(\mathbf{z}_k-\mathbf{z}_{k+1}\right)^T\mathbf{b}_{k+1}.\end{aligned}\tag{3.18}$$

Similarly, since $\mathbf{z}_k$ minimizes (1.16) for $\mathbf{u}=\mathbf{u}_k$ and $\mathbf{b}=\mathbf{b}_{k-1}$, for $\mathbf{z}=\mathbf{z}_k$ we have

$$\begin{aligned}&\partial_z\frac{\lambda}{2}\|\mathbf{z}-\mathbf{B}\mathbf{u}_k+\mathbf{b}_{k-1}\|_2^2=\lambda\left(\mathbf{z}-\mathbf{B}\mathbf{u}_k+\mathbf{b}_{k-1}\right)\in-\partial\|\mathbf{z}\|_1\ \Rightarrow\\&\|\mathbf{z}_k\|_1-\|\mathbf{z}_{k+1}\|_1\ \le\ \left(\mathbf{z}_{k+1}-\mathbf{z}_k\right)^T\left(\mathbf{z}_k-\mathbf{B}\mathbf{u}_k+\mathbf{b}_{k-1}\right)=\left(\mathbf{z}_{k+1}-\mathbf{z}_k\right)^T\mathbf{b}_k.\end{aligned}\tag{3.19}$$

In both (3.18) and (3.19) we used step 6 of Algorithm 1 and the fact that for any convex function $f(\mathbf{x})$

$$f(\mathbf{x}_0)+\boldsymbol{\xi}^T\left(\mathbf{x}-\mathbf{x}_0\right)\ \le\ f(\mathbf{x})\ \Leftrightarrow\ f(\mathbf{x}_0)-f(\mathbf{x})\ \le\ -\boldsymbol{\xi}^T\left(\mathbf{x}-\mathbf{x}_0\right),\ \text{ if } \boldsymbol{\xi}\in\partial f(\mathbf{x}_0),$$

where ∂ is subgradient [34]. Summing (3.18) and (3.19) we get

$$0\ \le\ \left(\mathbf{z}_k-\mathbf{z}_{k+1}\right)^T\left(\mathbf{b}_{k+1}-\mathbf{b}_k\right).\tag{3.20}$$

From (3.20) and (3.17), we have

$$\begin{aligned}&\|\mathbf{z}_k-\mathbf{z}_{k+1}\|_2^2+\|\boldsymbol{\rho}_{k+1}\|_2^2\ \le\\&\left(\|\mathbf{z}_k-\mathbf{z}^*\|_2^2+\|\mathbf{b}_k-\mathbf{b}^*\|_2^2\right)-\left(\|\mathbf{z}_{k+1}-\mathbf{z}^*\|_2^2+\|\mathbf{b}_{k+1}-\mathbf{b}^*\|_2^2\right),\end{aligned}\tag{3.21}$$

or

$$\begin{aligned}&\|\mathbf{z}_{k+1}-\mathbf{z}^*\|_2^2+\|\mathbf{b}_{k+1}-\mathbf{b}^*\|_2^2\ \le\\&\|\mathbf{z}_k-\mathbf{z}^*\|_2^2+\|\mathbf{b}_k-\mathbf{b}^*\|_2^2-\|\mathbf{z}_{k+1}-\mathbf{z}_k\|_2^2-\|\boldsymbol{\rho}_{k+1}\|_2^2.\end{aligned}\tag{3.22}$$

From (3.22) we can see that the sequence $\|\mathbf{z}_k - \mathbf{z}^*\|_2^2 + \|\mathbf{b}_k - \mathbf{b}^*\|_2^2$ and consequently $\mathbf{z}_k$ and $\mathbf{b}_k$ are bounded. Summing (3.21) for $k = 0, 1, \ldots, \infty$, we obtain convergence of the series

$$\sum_{k=0}^{\infty} \left\{ \|\mathbf{z}_k - \mathbf{z}_{k+1}\|_2^2 + \|\boldsymbol{\rho}_{k+1}\|_2^2 \right\} \;\leq\; \|\mathbf{z}_0 - \mathbf{z}^*\|_2^2 + \|\mathbf{b}_0 - \mathbf{b}^*\|_2^2. \tag{3.23}$$

From (3.23) follows

$$\mathbf{z}_k - \mathbf{z}_{k+1} \to 0, \;\; \mathbf{z}_k - \mathbf{B}\mathbf{u}_k \to 0, \; k \to \infty. \tag{3.24}$$

Now using (3.11) we obtain

$$\begin{aligned} p_{k+1} \;-\; p^* \;\leq\; & \lambda \mathbf{b}_{k+1}^T (\mathbf{z}^* - \mathbf{z}_{k+1}) + \lambda (\mathbf{z}_k - \mathbf{z}_{k+1} + \mathbf{b}_{k+1})^T \mathbf{B} (\mathbf{u}_{k+1} - \mathbf{u}^*) \;= \\ & \lambda \mathbf{b}_{k+1}^T (\mathbf{z}_k - \mathbf{z}_{k+1}) + \lambda \mathbf{b}_{k+1}^T (\mathbf{z}^* - \mathbf{z}_k) \;+ \\ & \lambda (\mathbf{z}_k - \mathbf{z}_{k+1})^T \mathbf{B} (\mathbf{u}_{k+1} - \mathbf{u}^*) + \lambda \mathbf{b}_{k+1}^T \mathbf{B} (\mathbf{u}_{k+1} - \mathbf{u}^*) \;= \\ & \lambda \mathbf{b}_{k+1}^T (\mathbf{z}_k - \mathbf{z}_{k+1}) \;+\; \lambda (\mathbf{z}_k - \mathbf{z}_{k+1})^T \mathbf{B} (\mathbf{u}_{k+1} - \mathbf{u}^*) \;+ \\ & \lambda \mathbf{b}_{k+1}^T (\mathbf{z}^* - \mathbf{z}_k) \;+\; \lambda \mathbf{b}_{k+1}^T \mathbf{B} (\mathbf{u}_{k+1} - \mathbf{u}^*) \;= \\ & \lambda \mathbf{b}_{k+1}^T (\mathbf{z}_k - \mathbf{z}_{k+1}) \;+\; \lambda (\mathbf{z}_k - \mathbf{z}_{k+1})^T \mathbf{B} (\mathbf{u}_{k+1} - \mathbf{u}^*) \;+ \\ & \lambda \mathbf{b}_{k+1}^T (\mathbf{B}\mathbf{u}_{k+1} - \mathbf{z}_{k+1} + \mathbf{z}_{k+1} - \mathbf{z}_k + \mathbf{z}^* - \mathbf{B}\mathbf{u}^*) \;\to\; 0, \; k \to \infty, \end{aligned} \tag{3.25}$$

where the right-hand side of (3.25) converges to zero because of (3.24), boundedness of $\mathbf{z}_k$ and $\mathbf{b}_k$ and $\mathbf{z}^* = \mathbf{B}\mathbf{u}^*$. Likewise, from (3.4) we have

$$p^* - p_{k+1} \;\leq\; \lambda \mathbf{b}^{*T} (\mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}) \;\to\; 0, \; k \to \infty. \tag{3.26}$$

Combining (3.25) and (3.26) we obtain $p_k \to p^*$—i.e., value of the objective function estimate at iteration k converges to the true minimum as $k \to \infty$. From the bounded sequence $\mathbf{u}_k \in \mathbb{R}^N$ we can extract a convergent subsequence

$$\mathbf{u}_{k_i} \;\to\; \mathbf{u}^{**}. \tag{3.27}$$

Because our objective function is continuous, $\mathbf{u}^{**}$ is a solution of (1.1) and (1.8). However, if $\mathbf{A}$ is maximum rank the objective function of (1.1) is strictly convex, hence $\mathbf{u}^* = \mathbf{u}^{**}$. The sequence $\mathbf{u}_k$ must converge to $\mathbf{u}^*$ because otherwise we would be able to extract a subsequence convergent to a different limit and repeat the above analysis.

And finally, to prove that $\mathbf{b}_k \to \mathbf{b}^*$, we see that from the Karush-Kuhn-Tucker (KKT) conditions [7] for (1.8) we have

$$\alpha \mathbf{A}\mathbf{A}^T \mathbf{u}^* \;=\; \mathbf{A}^T \mathbf{d} \;+\; \lambda \mathbf{B}^T \mathbf{b}^*. \tag{3.28}$$

Passing (3.6) to limit as $k \to \infty$, using (3.24) and replacing $\mathbf{b}_{k+1}$ with a convergent subsequence as necessary, we get

$$\alpha \mathbf{A}\mathbf{A}^T \mathbf{u}^* \;=\; \mathbf{A}^T \mathbf{d} \;+\; \lambda \mathbf{B}^T \lim \mathbf{b}_k. \tag{3.29}$$

Since $\mathbf{B}$ is maximum rank, $\operatorname{rank}\mathbf{B} = K \leq N$, (3.29) means that $\lim \mathbf{b}_k = \mathbf{b}^*$. ■

Note that our proof does not depend on the selection of starting values for $\mathbf{u}_0$, $\mathbf{z}_0$ and $\mathbf{b}_0$, and this fact will be used later on in proving the convergence of Algorithm 3. Before we study convergence properties of Algorithm 3, we prove one auxiliary result.

Theorem 3.2. *Algorithm 3 constructs a sequence of subspaces of $\mathbb{R}^N$ spanning expanding sets of conjugate directions,*

$$\begin{aligned} S_k &= \operatorname{span}\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k\},\ k = 0, 1, 2, \ldots \\ S_0 &\subseteq S_1 \subseteq S_2 \subseteq \ldots \subseteq S_k \subseteq \ldots \end{aligned} \tag{3.30}$$

such that

$$\lim_{k\to\infty} S_k = S \subseteq \mathbb{R}^N. \tag{3.31}$$

Under the assumptions of Theorem 3.1, solution of the constrained optimization problem

$$\begin{aligned} &\|\mathbf{z}\|_1 + \frac{\alpha}{2}\|\mathbf{A}\mathbf{u} - \mathbf{d}\|_2^2 \to \min, \\ &\mathbf{z} = \mathbf{B}\mathbf{u}, \\ &\mathbf{u} \in S. \end{aligned} \tag{3.32}$$

matches the solution of (1.8).

Proof. If $S = \mathbb{R}^N$ statement of the theorem is trivial, so we assume that $\dim S < N$. Since our problem is finite-dimensional, the limit (3.31) is achieved at a finite iteration,

$$\exists k_1\ \forall k \geq k_1 :\ S_k \equiv S. \tag{3.33}$$

steps 4-7 of Algorithm 3 are equivalent to projecting the solution of the system of normal equations (2.4) onto the space S_k. If $p_{k+1} = 0$ in steps 20-22, then the right-hand side of (2.4) for any $k \geq k_1$ can be represented as a linear combination of vectors from $S_{k_1} \equiv S$. Steps 8 and 9 of Algorithm 3 are equivalent to steps 5 and 6 of Algorithm 1. Step 10 prepares the right-hand side of (2.4) for the minimization in step 4 of Algorithm 1 for iteration $k+1$. However, since the right-hand side of (2.4) is a linear combination of vectors $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k$ that span $S_k \equiv S$, steps 4-7 of Algorithm 3 are equivalent to the exact solution of the unconstrained minimization problem in step 4 of Algorithm 1. Hence, starting from iteration k_1 the two algorithms become equivalent. From Theorem 3.1 and

$$\forall k \geq k_1 :\ \mathbf{u}_{k+1} \in S$$

follows that the solution of (3.32) coincides with that of (1.8). ■

Convergence of Algorithm 3 now becomes a trivial corollary of theorems 3.1 and 3.2.

Theorem 3.3. *Under the assumptions of Theorem 3.1, Algorithm 3 converges to the unique solution (3.1) of problem (1.8), and (3.3) holds.*

Proof. In the proof of Theorem 3.2 we have demonstrated that starting from $k = k_1$ defined in (3.33) Algorithm 3 is mathematically equivalent to Algorithm 1 starting from an initial approximation $\mathbf{u}_{k_1-1}$, $\mathbf{z}_{k_1-1}$ and $\mathbf{b}_{k_1-1}$. Convergence of Algorithm 1 does not depend on

these starting values, hence Algorithm 3 converges to the same unique solution as Algorithm 1 and (3.3) holds. ■

The result of Theorem 3.3 indicates that our Compressive Conjugate Directions method matches the ADMM in exact arithmetic after a finite number of iterations, while avoiding direct inversion of operator $\mathbf{A}$. This obvously means that the (worst-case) asymptotic convergence rate of Algorithm 3 matches that of the ADMM and is $O(1/k)$ [24].

4. Limited-memory Compressive Conjugate Directions Method. Algorithm 3 (that we call "unlimited-memory" Compressive Conjugate Directions Method) requires storing all of the previous conjugate directions (2.6) because in step 7 the algorithm computes the expansion

$$\mathbf{u}_{k+1} \;=\; \sum_{i=0}^{k} \tau_i \mathbf{p}_i, \tag{4.1}$$

of these solution approximations with respect to all conjugate direction vectors (2.6) at each iteration. It is a consequence of changing right-hand sides of the normal equations system (2.1) that *all* of the coefficients of expansion (4.1) may require updating. However, in a practical implementation we may expect that only the last $m+1$ expansion coefficients (4.1) significantly change, and freeze the coefficients

$$\tau_i, \; i \;<\; k-m$$

at and after iteration k. This approach requires storing up to $2m+2$ latest vectors

$$\mathbf{p}_k, \mathbf{p}_{k-1}, \ldots, \mathbf{p}_{k-m}, \quad \mathbf{q}_k, \mathbf{q}_{k-1}, \ldots, \mathbf{q}_{k-m}. \tag{4.2}$$

A "limited-memory" variant of the method is implemented in Algorithm 4 that stores vectors (4.2) in a circular first-in-first-out buffer. An index variable j points to the latest updated element within the buffer. Once j exceed the buffer size for the first time and is reset to point to the head of the buffer, a flag variable *cycle* is set, indicating that a search direction is overwritten at each subsequent iteration of the algorithm. The projection of the current solution iterate onto the old vector $\tau_j \mathbf{p}_j$ (now to be overwritten in the buffer) is then accumulated in a vector $\tilde{\mathbf{u}}$; the corresponding contribution to the predicted data equals $\tau_j \mathbf{q}_j$ and is accumulated in a vector $\tilde{\mathbf{v}}$,

$$\tilde{\mathbf{u}} \;=\; \sum_{i=0}^{k-m-1} \tau_i \mathbf{p}_i, \quad \tilde{\mathbf{v}} \;=\; \sum_{i=0}^{k-m-1} \tau_i \mathbf{q}_i. \tag{4.3}$$

Contributions (4.3) to the solution and predicted data from the discarded vectors (2.6) are then added back to the approximate solution and residual in steps 8 and 12 of Algorithm 4.

4.1. Trade-off between the number of iterations and problem condition number. In practical implementations of the ADMM when the operator $\mathbf{A}$ does not lend itself to direct solution methods, an iterative method can be used to solve the minimization problem in step 4 of Algorithm 1 [22]. Algorithm 5, representing such an approach, uses a fixed number of iterations N_c of CGNE in step 4. At each iteration of the ADMM conjugate gradients are

Algorithm 4 Limited-Memory Compressive Conjugate Directions Method for (1.1)

$m \leftarrow$ memory size, $\tilde{\mathbf{u}} \leftarrow \mathbf{0}^N$, $\tilde{\mathbf{v}} \leftarrow \mathbf{0}^{N+K}$, $j \leftarrow 0$, $cycle \leftarrow .false.$
$\mathbf{u}_0 \leftarrow \mathbf{0}$, $\mathbf{z}_0 \leftarrow \mathbf{0}^K$; $\mathbf{b}_0 \leftarrow \mathbf{0}^K$, $\mathbf{v}_0 \leftarrow \begin{bmatrix} \sqrt{\alpha}\mathbf{d} \\ \sqrt{\lambda}\,(\mathbf{z}_0 + \mathbf{b}_0) \end{bmatrix}$
$\mathbf{p}_0 \leftarrow \mathbf{F}^T\mathbf{v}_0$, $\mathbf{q}_0 \leftarrow \mathbf{F}\mathbf{p}_0$, $\delta_0 \leftarrow \mathbf{q}_0^T\mathbf{q}_0$
for $k = 0, 1, 2, 3, \ldots$ **do**
    for $i = 0, 1, \ldots, \min(k, m)$ **do**
        $\tau_i \leftarrow \mathbf{q}_i^T(\mathbf{v}_k - \tilde{\mathbf{v}})/\delta_i$
    end for
    $\mathbf{u}_{k+1} \leftarrow \tilde{\mathbf{u}} + \sum_{i=0}^{\min(k,m)} \tau_i \mathbf{p}_i$
    $\mathbf{z}_{k+1} \leftarrow \operatorname{shrink}\{\mathbf{B}\mathbf{u}_{k+1} - \mathbf{b}_k, 1/\lambda\}$
    $\mathbf{b}_{k+1} \leftarrow \mathbf{b}_k + \mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}$
    $\mathbf{v}_{k+1} \leftarrow \begin{bmatrix} \sqrt{\alpha}\mathbf{d} \\ \sqrt{\lambda}\,(\mathbf{z}_{k+1} + \mathbf{b}_{k+1}) \end{bmatrix}$
    $\mathbf{r}_{k+1} \leftarrow \mathbf{v}_{k+1} - \sum_{i=0}^{\min(k,m)} \tau_i \mathbf{q}_i - \tilde{\mathbf{v}}$
    $\mathbf{w}_{k+1} \leftarrow \mathbf{F}^T\mathbf{r}_{k+1}$
    $\mathbf{s}_{k+1} \leftarrow \mathbf{F}\mathbf{w}_{k+1}$
    for $i = 0, 1, \ldots, \min(k, m)$ **do**
        $\beta_i \leftarrow -\mathbf{q}_i^T\mathbf{s}_{k+1}/\delta_i$
    end for
    $j \leftarrow j + 1$
    if $j = m + 1$ **then**
        $j \leftarrow 0$, $cycle \leftarrow .true.$
    end if
    if *cycle* **then**
        $\tilde{\mathbf{u}} \leftarrow \tilde{\mathbf{u}} + \tau_j \mathbf{p}_j$
        $\tilde{\mathbf{v}} \leftarrow \tilde{\mathbf{v}} + \tau_j \mathbf{q}_j$
    end if
    $\mathbf{p}_j \leftarrow \sum_{i=0}^{\min(k,m)} \beta_i \mathbf{p}_i + \mathbf{w}_{k+1}$
    $\mathbf{q}_j \leftarrow \sum_{i=0}^{\min(k,m)} \beta_i \mathbf{q}_i + \mathbf{s}_{k+1}$
    $\delta_j \leftarrow \mathbf{q}_j^T\mathbf{q}_j$
    if $\delta_j = 0$ **then** ▷ Use condition "$\delta_j <$ tolerance" in practice
        $\delta_j \leftarrow 1$, $\mathbf{p}_j \leftarrow \mathbf{0}^N$, $\mathbf{q}_j \leftarrow \mathbf{0}^{M+K}$
    end if
    Exit loop if $\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_2 / \|\mathbf{u}_k\|_2 \leq$ target accuracy
end for

hot-restarted from the previous solution approximation $\mathbf{u}_k$. For comparison purposes we will refer to this method as *restarted Conjugate Gradients* or *RCG*.

Note that Algorithm 5 with $N_c = 1$ performs a single step of gradient descent when solving

Algorithm 5 ADMM and hot-restarted CG (*RCG*)

1: $\mathbf{u}_0 \leftarrow \mathbf{0}^N$, $\mathbf{z}_0 \leftarrow \mathbf{0}^K$, $\mathbf{b}_0 \leftarrow \mathbf{0}^K$, $N_c \leftarrow$ prescribed number of CG iterations
2: $\mathbf{p}_0 \leftarrow \mathbf{F}^T\mathbf{v}_0$, $\mathbf{q}_0 \leftarrow \mathbf{F}\mathbf{p}_0$
3: **for** $k = 0, 1, 2, 3, \ldots$ **do**
4: Solve

$$\mathbf{u}_{k+1} \leftarrow \text{argmin} \left\{ \frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{B}\mathbf{u} + \mathbf{b}_k\|_2^2 + \frac{\alpha}{2}\|\mathbf{A}\mathbf{u} - \mathbf{d}\|_2^2 \right\},$$

starting from $\mathbf{u}_k$ and using N_c iterations of CGNE.
5: $\mathbf{z}_{k+1} \leftarrow \text{shrink}\{\mathbf{B}\mathbf{u}_{k+1} - \mathbf{b}_k, 1/\lambda\}$
6: $\mathbf{b}_{k+1} \leftarrow \mathbf{b}_k + \mathbf{z}_{k+1} - \mathbf{B}\mathbf{u}_{k+1}$
7: Exit loop if $\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_2 / \|\mathbf{u}_k\|_2 \leq$ target accuracy
8: **end for**

the following intermediate least-squares minimization problem in step 4,

$$\mathbf{u}_{k+1} = \text{argmin} \frac{\alpha}{2}\|\mathbf{A}\mathbf{u} - \mathbf{d}\|_2^2 + \frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{B}\mathbf{u} + \mathbf{b}_k\|_2^2. \tag{4.4}$$

The performance of Algorithm 5 depends on the condition number of the leasts-squares problem (4.4) [40]: for well-conditioned problems only a small number of conjugate gradients iterations N_c may achieve a sufficiently accurate approximation to $\mathbf{u}_{k+1}$. The condition number of (4.4) depends on properties of operators $\mathbf{A}$ and $\mathbf{B}$, as well as the value of λ. In applications with a simple modeling operator $\mathbf{A}$, such as is the case in denoising with $\mathbf{A} = \mathbf{I}$, a value of λ may be experimentally selected so as to reduce the condition number of (4.4). However, a trade-off may exist between the condition number of (4.4) and the number of ADMM iterations in the outer loop (Step 3) of Algorithm 1: well-conditioned interim least-squares problems may result in a significantly higher number of ADMM iterations. Such a trade-off is a well-known phenomenon in applications of the Augmented Lagrangian Method of Multipliers for smooth objective functions, see, e.g., [19]. For example, large values of λ in (1.15) more strongly penalize violations of the equality constraint, as in the Quadratic Penalty Function Method [30] with a larger penalty and a more ill-conditioned quadratic minimization. Of course, in the case of ADMM applied to (1.1), a non-smooth objective function, arbitrary and potentially ill-conditioned operator $\mathbf{A}$, and (most importantly) alternating splitting minimization of the modified Augmented Lagrangian (1.15)[6] complicate the picture. In fact, for an arbitrary $\mathbf{A}$, the condition number of (4.4) is not always an increasing function of λ. Some of the numerical examples described in the following subsections exhibit this trade-off between the condition number of the intermediate least-squares problem (4.4) and the number of ADMM iterations: the better the condition-number of (4.4), the more ADMM iterations are typically required. The main advantage of our Compressive Conjugate Directions approach implemented in Algorithms 3 and 4 is that information on the geometry of the objective function (4.4) accumulates through *external* ADMM iterations thus potentially reducing the amount of effort required to perform minimization of (4.4) at each step. Since our objective is a practical implementation of the ADMM for (1.1) with computationally expensive operators $\mathbf{A}$, the overall number

[6] "modified" because of the added constant term $\lambda/2\|\mathbf{b}_k\|_2^2$

of operator $\mathbf{A}$ and $\mathbf{A}^T$ applications required to achieve given accuracy will be the principal benchmark for measuring the performance of various algorithms.

5. Applications. In this section we apply the method of Compressive Conjugate Directions to solving L_1 and TV-regularized inversion problems for several practical examples.

5.1. Image Denoising. A popular image denoising technique for removing short-wavelength random Gaussian noise from an image is based on solving (1.3) with $\mathbf{A} = \mathbf{I}$. Vector $\mathbf{d}$ is populated with a noisy image, a denoised image is returned in $\mathbf{u}$,

$$\mathbf{u} = u_{i,j},\ i = 1, \ldots, N_y,\ j = 1, \ldots, N_x,$$

with an *anisotropic TV norm* in (1.3) defined by the linear gradient operator

$$\nabla \mathbf{u} = \begin{bmatrix} \nabla_x \mathbf{u} \\ \nabla_y \mathbf{u} \end{bmatrix} = \begin{bmatrix} u_{i,2} - u_{i,1} \\ \cdots \\ u_{i,N_x} - u_{i,N_x-1} \\ \cdots \\ u_{2,j} - u_{1,j} \\ \cdots \\ u_{N_y,j} - u_{N_y-1,j} \end{bmatrix},\ i = 1, \ldots, N_y,\ j = 1, \ldots, N_x. \tag{5.1}$$

Here, the dimension of the model space is $N = N_x \times N_y$ with $M = N$ and $K = N - N_x - N_y$. Since operator $\mathbf{A} = \mathbf{I}$ is trivial, minimization of the number of operator applications in this problem carries no practical advantage. The only reason for providing this example is to demonstrate the stability of the proposed Compressive Conjugate Directions method with respect to choosing a value of λ.

Figure 1(a) shows the true, noise-free 382×382 image used in this experiment. Random Gaussian noise with a standard deviation σ of 15% of maximum signal amplitude was added to the true image to produce the noisy image of Figure 1(b). All low-wavenumber or "blocky" components of the noise below a quarter of the Nyquist wavenumber were filtered out, leaving only high-wavenumber "salt-and-pepper" noise. Parameter $\alpha = 10$ was chosen experimentally based on the desired trade-off of fidelity and "blockiness" of the resulting denoised image. The result of solving (1.3) using Algorithm 5 with $\lambda = 1$, one hundred combined applications of $\mathbf{A}$ and $\mathbf{A}^T$, and $N_c = 1$ is shown in Figure 1(d). The result of applying our limited-memory Conjugate Directions Algorithm 4 for $m = 50$ is shown in Figure 1(c)[7]. Note that $N_c = 1$ means that only a single step of Conjugate Gradients, or a single gradient descent, is made in step 4 of Algorithm 5. For this choice of λ, problem (4.4) is very well conditioned, with a condition number of $\kappa = 1.8$. A single iteration of gradient descent achieves sufficient accuracy of minimization (4.4) and for $\lambda = 1$ there is no practical advantage in using our method as both methods perform equally well, see Figure 2(a). In fact, the overhead of storing and using conjugate directions from previous iterations may exceed the cost of operator $\mathbf{A}$ and its adjoint applications if the latter are computationally cheap. The approximation errors of applying the limited-memory Compressive Conjugate Directions Algorithm 4 with $m = 50$ versus Algorithm 5 with $N_c = 1, 5, 10$ for $\lambda = 10^2, 10^3, 10^4$ are shown in Figures 2(a),2(b),2(c),2(d).

[7]Here, this matches the results for *any* memory size $m > 0$ due to a well-conditioned problem (4.4).

Figure 1: (a) Clean image; (b) Noisy image contaminated with Gaussian noise with $\sigma = 15\%$ of maximum amplitude; (c) Image denoised using Algorithm 4 with $\alpha = 10,\ \lambda = 1$ and memory size $m = 50$; (d) Image denoised using Algorithm 5 with $\alpha = 10,\ \lambda = 1,\ N_c = 1$.

Note that larger values of λ result in increasingly larger condition numbers of (4.4) shown on top of the plots. The performance of Algorithm 5 here depends on a choice of N_c: increasing N_c as required to achieve a sufficiently accurate approximate solution of (4.4) results in fewer available ADMM iterations for a fixed "budget" of operator $\mathbf{A}$ and adjoint applications. However, Algorithm 4 accumulates conjugate directions (2.6) computed at earlier iterations and requires only one application of the operator and its adjoint per ADMM iteration. Note that

at iteration steps less than N_c, Algorithm 5 may still outperform Algorithm 4 as it conducts more Conjugate Gradient iterations per solution of each problem (4.4). However, once the ADMM iteration count exceeds the largest N_c, and sufficient information is accumulated by Algorithm 4 about the geometry of the objective function, the Compressive Conjugate Directions outperforms Algorithm 5. Note that this example does not demonstrate the trade-off

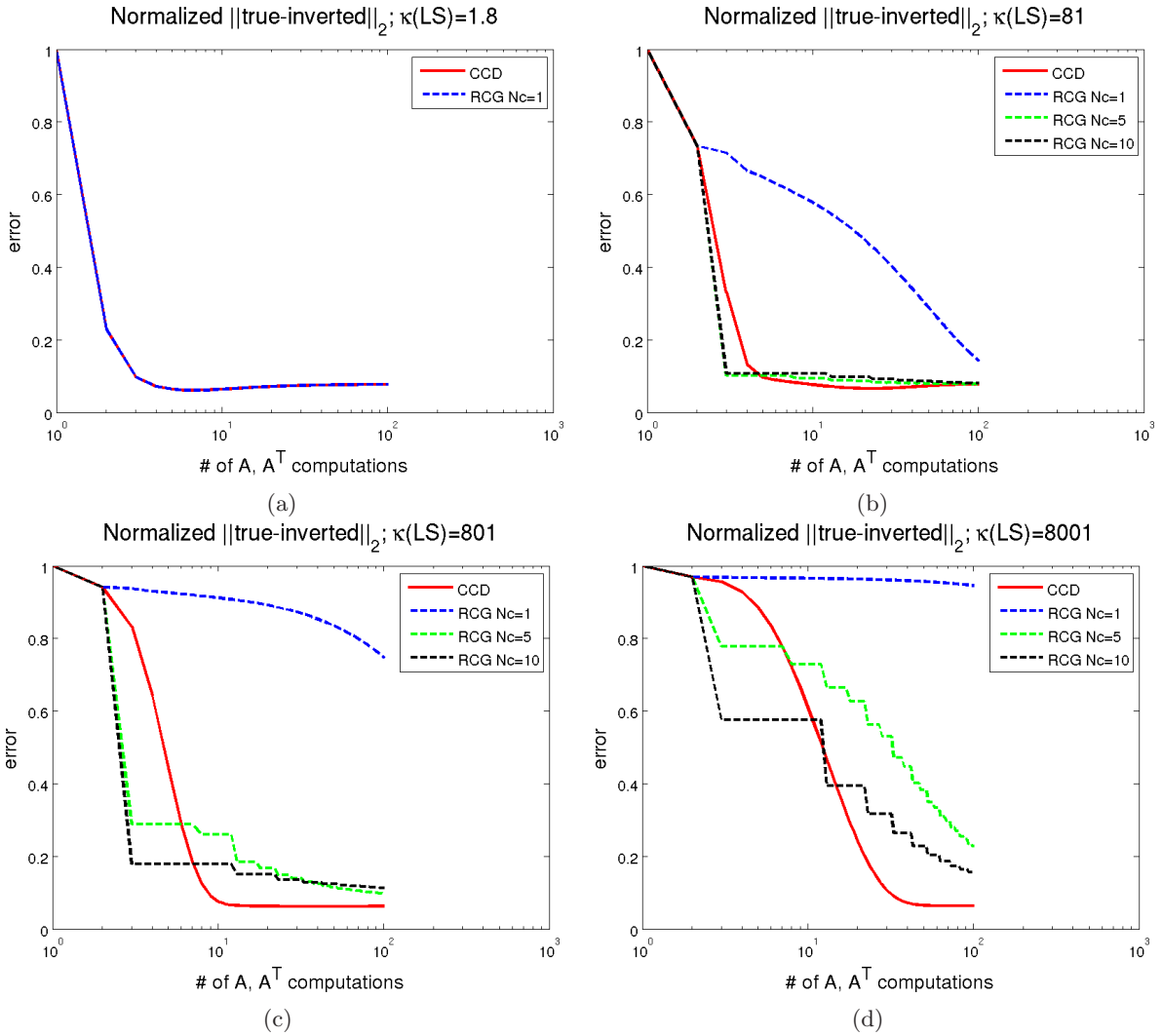


Figure 2: Performance of Algorithm 4 with $m = 20$ versus Algorithm 5 with varying N_c for (a) $\lambda = 1$; (b) $\lambda = 100$; (c) $\lambda = 1000$; (d) $\lambda = 10000$.

between the condition number of (4.4) and the number of ADMM iterations. The reason for this is that for large λ convergence is achieved relatively quickly within a number of iterations comparable to a number of Conjugate Gradients steps required to solve (4.4). However, this example demonstrate another feature of the proposed Compressive Conjugate Directions Method: compared with a technique based on a restarted iterative solution of (4.4), the

method may be less sensitive to a suboptimal choice of λ.

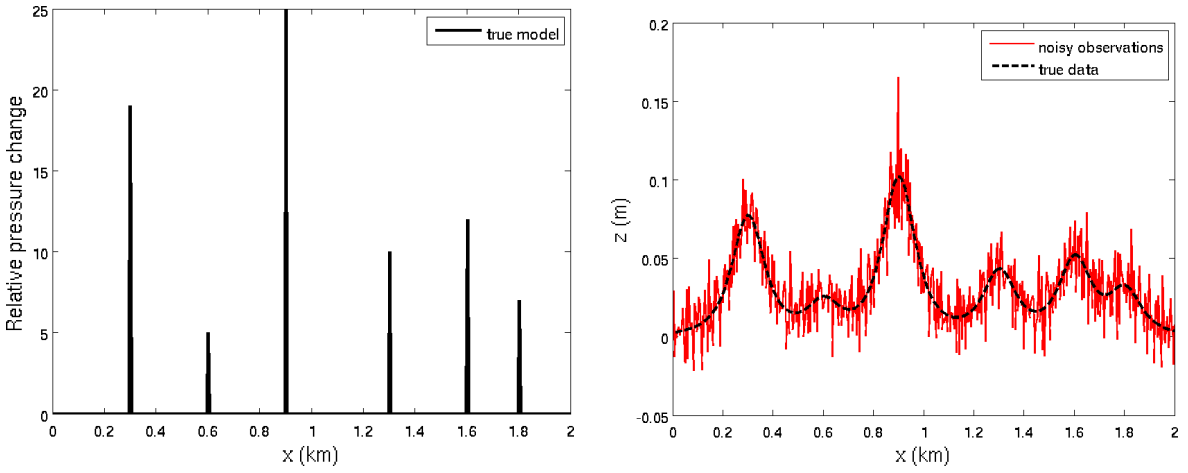
5.2. Inversion of Dilatational Point Pseudo-sources. In our second example, we demonstrate our method on a geomechanical inversion problem with a non-trivial forward-modeling operator $\mathbf{A}$. Here, we are interested in inverting subsurface sources of deformation from noisy measurements of surface displacements, such as GPS, tilt-meter and InSAR observations.

The forward modeling operator simulates vertical surface displacements in response to distributed dilatational (e.g. pressure change) sources [38]. Our modeling operator is defined as

$$\mathbf{Au} \;=\; d(z),\; d(z) \;=\; c\int_0^A \frac{Du(\xi)d\xi}{(D^2+(z-\xi)^2)^{3/2}}, \tag{5.2}$$

where we assume that $\mathbf{u} = u(\xi), \xi \in [0, A]$ is a relative pore pressure change along a horizontal segment $[0, A]$ of a reservoir at a constant depth D, $\mathbf{d} = d(x), x \in [0, A]$ is the induced vertical displacement on the surface, and a factor c is determined by the poroelastic medium properties, and reservoir dimensions. In this example, for demonstration purposes we consider a two-dimensional model, but a three-dimensional model is studied in subsection 5.3. Operator (5.2) is a smoothing integral operator that, after discretization and application of a simple quadrature, can be represented by a dense matrix. Analytical representation of the surface displacement modeling operator (5.2) is possible for simple homogeneous media; however, modeling surface displacements in highly heterogeneous media will involve computationally expensive numerical methods such as Finite Elements [27].

In this experiment we seek to recover a spiky model of subsurface sources shown in Figure 3(a) from noisy observations of the induced surface displacements shown in Figure 3(b). Such sparse dilatational pseudo-sources are mathematically equivalent to concentrated reser-



(a) (b)

Figure 3: (a) A spiky true pseudosources; (b) the resulting true (black) and noisy (red) surface displacements.

voir pressure changes in hydrogeology and exploration geophysics, as well as expanding spher-

ical lava chambers (the "Mogi model") in volcanology [38]. We forward-modeled surface displacements due to the sources of Figure 3(a) using operator (5.2), and, as in our denoising tests, added random Gaussian noise with $\sigma = 15\%$ of the maximum data amplitude. Prior to adding the noise, all low-wavenumber noise components below a fifth of the Nyquist wavenumber were muted, leaving only the high-wavenumber noise shown in Figure 3(b).

We set $D = .1$ km, $A = 2$ km, $c = 10^{-2}$ in (5.2), and discretized both the model and data space using a 500-point uniform grid, $N = M = 500$. We solve problem (1.2) with $\alpha = 10000$, and our objective is to accurately identify locations of the spikes in Figure 3(a) and their relative magnitudes, carrying out as few applications of operator (5.2) as possible. Inversion results of using the limited-memory Compressive Conjugate Directions Algorithm 4 with $m = 100$, ADMM with restarted Conjugate Gradients Algorithm 5 and FISTA of (1.6) are shown in Figures 4(a),4(b),4(c),4(d) for $\lambda = 0.05, 0.1, 1, 100$. In each case one hundred combined products of operators $\mathbf{A}$ and $\mathbf{A}^T$ with vectors were computed. We used the maximum FISTA step size of $\tau = 10^{-4}$ in (1.6) computed for operator (5.2). These results indicate that the Compressive Conjugate Directions method achieves qualitative recovery of the spiky model at early iterations. Superiority of the new method is especially pronounced when the intermediate least-squares minimization problem (4.4) is ill-conditioned (see plot tops). The method retains its advantage after 1000 operator and adjoint applications, as shown in Figures 5(a),5(b),5(c),5(d). Note that the error plots of the CCD in Figures 6(a),6(b),6(c),6(d) exhibit a trade-off between the convergence rate and condition number of problem (4.4) discussed earlier in this subsection 4.1: a more ill-conditioned (4.4) is associated with a faster convergence rate of the new method.

Figures 7(a),7(b),7(c),7(d) show error plots for the CCD, ADMM with *exact* minimization of (4.4), and FISTA. The said trade-off between the convergence rate and condition number of (4.4) is exhibited by the ADMM. The CCD curves approach the convergence rates of the ADMM once Algorithm 4 has accumulated enough information about the geometry of the objective function in vectors (4.2). Note that the advantage of a faster asymptotic convergence rate of FISTA kicks in only when the ADMM-based methods use values of λ that are not optimal for their convergence—see Figures 6(d) and 7(d). In this case (4.4) is very well conditioned, and its adequate solution requires only a single step of gradient descent at each iteration of the ADMM, depriving conjugate-gradients-based methods of their advantage. FISTA, being based on accelerating a gradient-descent method, now *asymptotically* beats the convergence rates of the other techniques but this happens too late through the iterations to be of practical significance. In other words, in this particular example FISTA can beat the ADMM (and CCD) only if the latter use badly selected values of λ. Generalizing this observation about FISTA and ADMM for problem (1.2) with a general operator $\mathbf{A}$ goes beyond the scope of our work.

5.3. Inversion of Pressure Contrasts. In this section we apply the Compressive Conjugate Gradients method to identify sharp subsurface pressure contrasts in a reservoir from observations of induced surface displacements. We use a 3-dimensional geomechanical poroelastostatic model of pressure-induced deformation based on Biot's theory [38].

We solve a TV-regularized inversion problem (1.3) with operator $\mathbf{B}$ given by (5.1), and

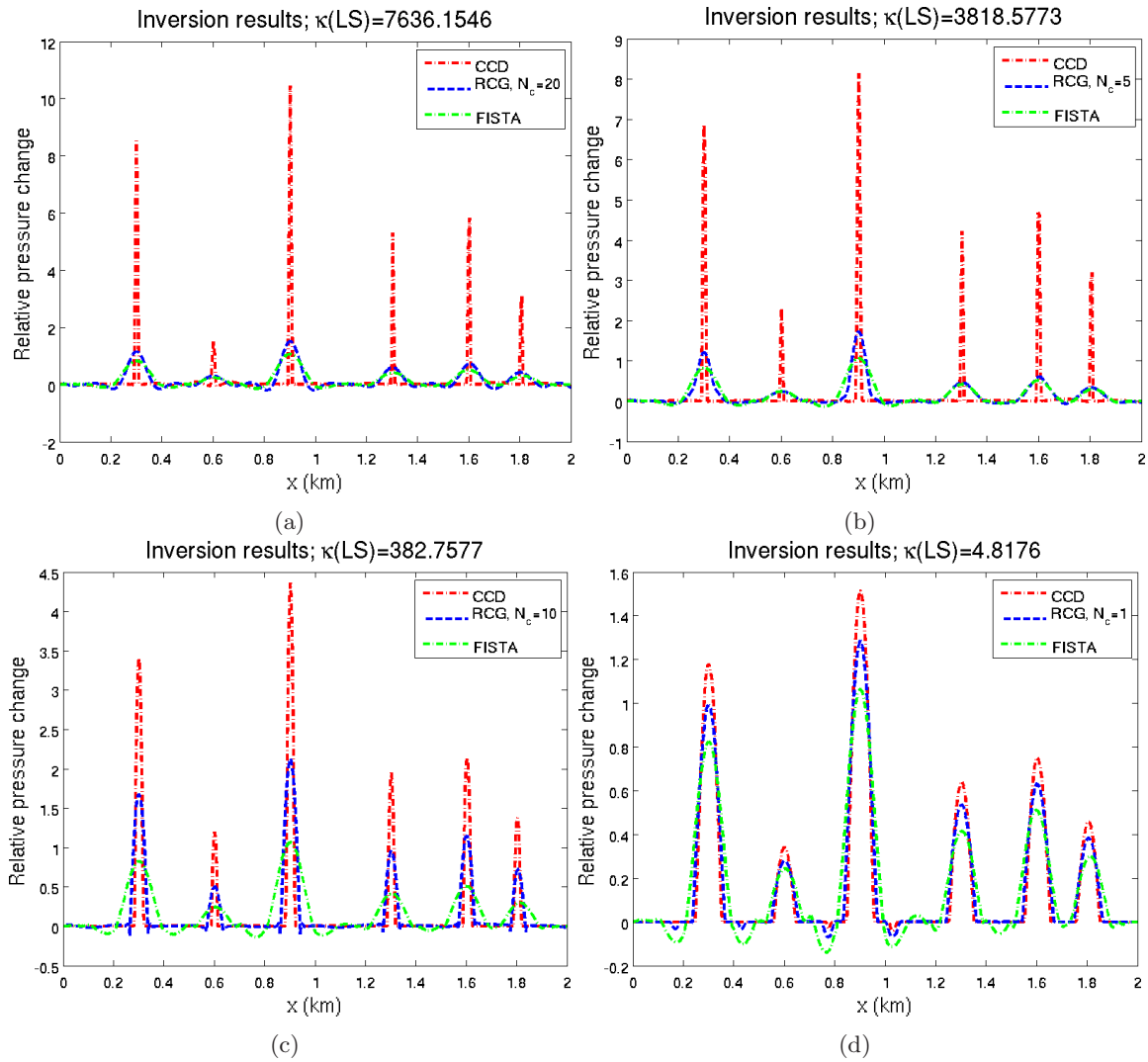


Figure 4: Inversion results for CCD (red), RCG (blue), FISTA (green) after 100 operator and adjoint applications for (a) $\lambda = .05$; (b) $\lambda = 0.1$; (c) $\lambda = 1$; (d) $\lambda = 100$. Note that FISTA does not use λ and the same FISTA results are shown in all plots but using different vertical scales. Improving condition number of (4.4) is accompanied by slower convergence. Compressive Conjugate Directions method most accurately resolves the spiky model at early iterations, and performs well when (4.4) is ill-conditioned.

operator $\mathbf{A}$ given by extension of (5.2)

$$\mathbf{Au} \;=\; d(x,y),\; d(x,y) \;=\; c\int_0^A \int_0^A \frac{Du(\xi,\eta)d\xi d\eta}{(D^2+(x-\xi)^2+(y-\eta)^2)^{3/2}}, \tag{5.3}$$

where we assume that $\mathbf{u} = u(\xi,\eta), (\xi,\eta) \in [-A,A]\times[-A,A]$ is a relative pore pressure change at a point (ξ,η) of the reservoir at a constant depth D, $2A$ is the reservoir length and breadth,

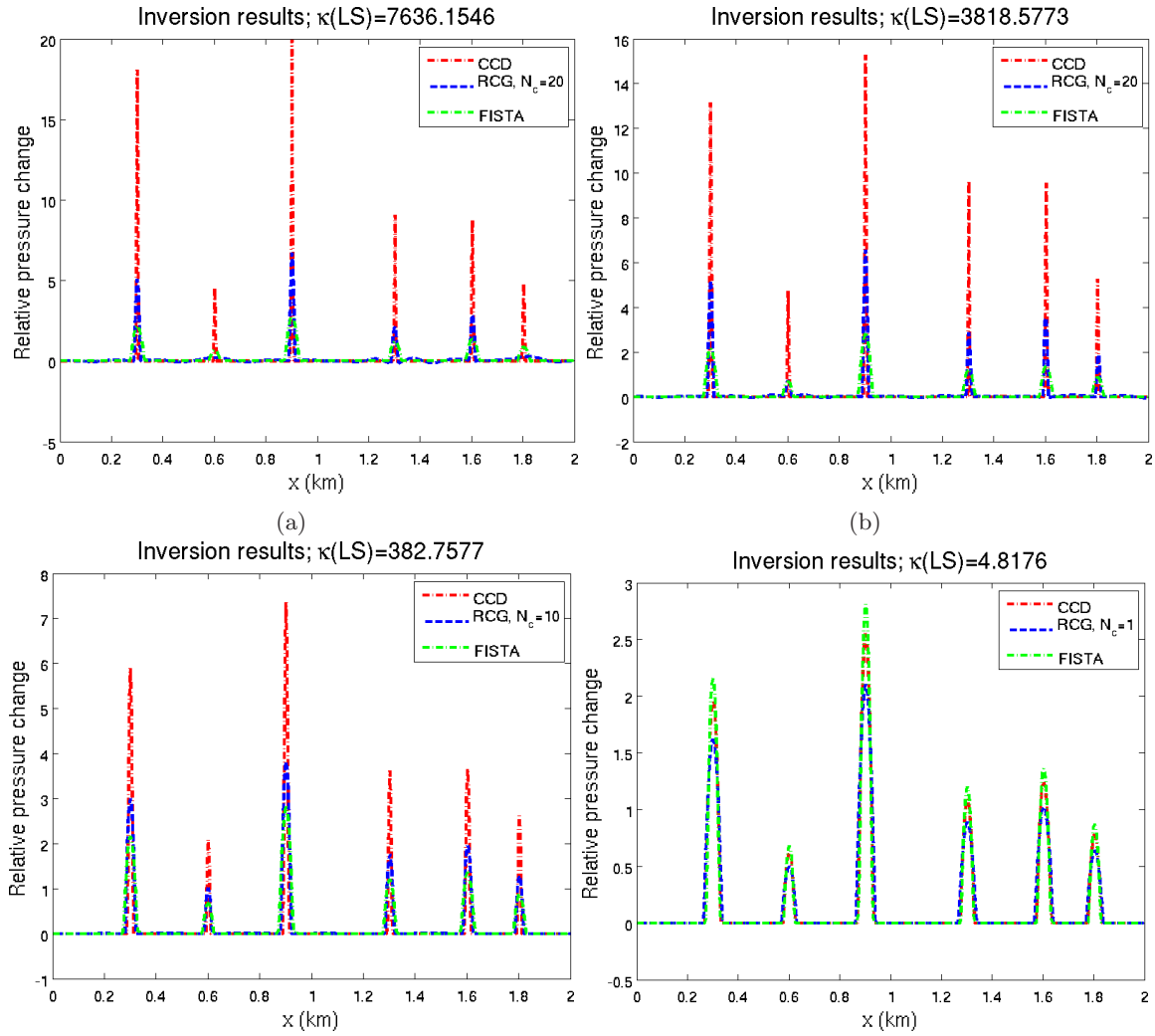


Figure 5: Inversion results for CCD (red), RCG (blue), FISTA (green) after 1000 operator and adjoint applications for (a) $\lambda = .05$; (b) $\lambda = 0.1$; (c) $\lambda = 1$; (d) $\lambda = 100$. Note that FISTA does not use λ and the same FISTA results are shown in all plots but using different vertical scales. Compressive Conjugate Directions method still retains its advantage in resolving the spiky model at earlier iterations. *Asymptotically* faster convergence of FISTA kicks in when $\lambda = 100$ with a well-conditioned (4.4), when the ADMM convergence is slowed—compare with Figure 7(d).

$\mathbf{d} = d(x, y), (x, y) \in [-A, A] \times [-A, A]$ is the induced vertical displacement at a point (x, y) on the surface, and a constant factor c is determined by the poroelastic medium properties and reservoir thickness.

In this experiment, we discretize the pressure and displacement using a 50×50 grid, with $A = 1.2$ km, $D = .455$ km and $c = 5.8515 \times 10^3$, based on a poroelastic model of a real-world

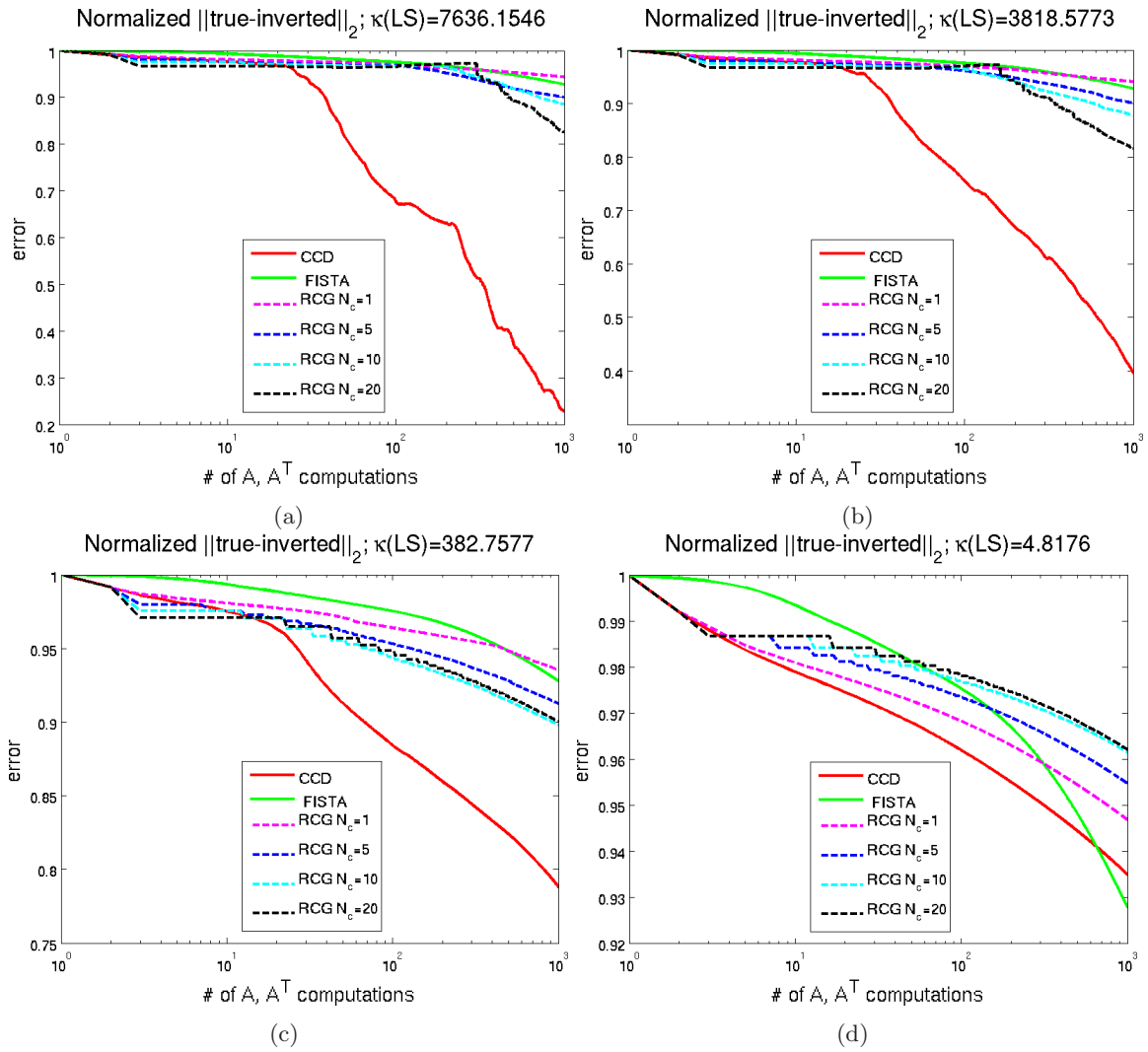


Figure 6: Convergence curves for CCD (solid red), RCG (dashed), FISTA (solid green) for (a) $\lambda = .05$; (b) $\lambda = 0.1$; (c) $\lambda = 1$; (d) $\lambda = 100$—compare with Figures 5(a),5(b),5(c),5(d).

unconventional hydrocarbon reservoir [28]. We use a least-squares fitting weight $\alpha = .1$ in (1.3) to achieve a desirable trade-off between fitting fidelity and blockiness of the inverted pressure change. The blocky model shown in Figure 8(a) was used to forward-model surface displacements using operator (5.3). Random Gaussian noise with $\sigma = 0.15\%$ of maximum data amplitude, muted below a quarter of the Nyquist wavenumber, was added to the clean data to produce the noisy displacement measurements of Figure 8(b).

Figure 9(a) shows the result of the limited-memory Compressive Conjugate Directions Algorithm 4 with $m = 100$, after a total of 100 combined applications of operator $\mathbf{A}$ and its adjoint. For the same number of operator applications, Figure 9(b) shows the best result of the ADMM with restarted Conjugate Gradients Algorithm 5. The corresponding results after

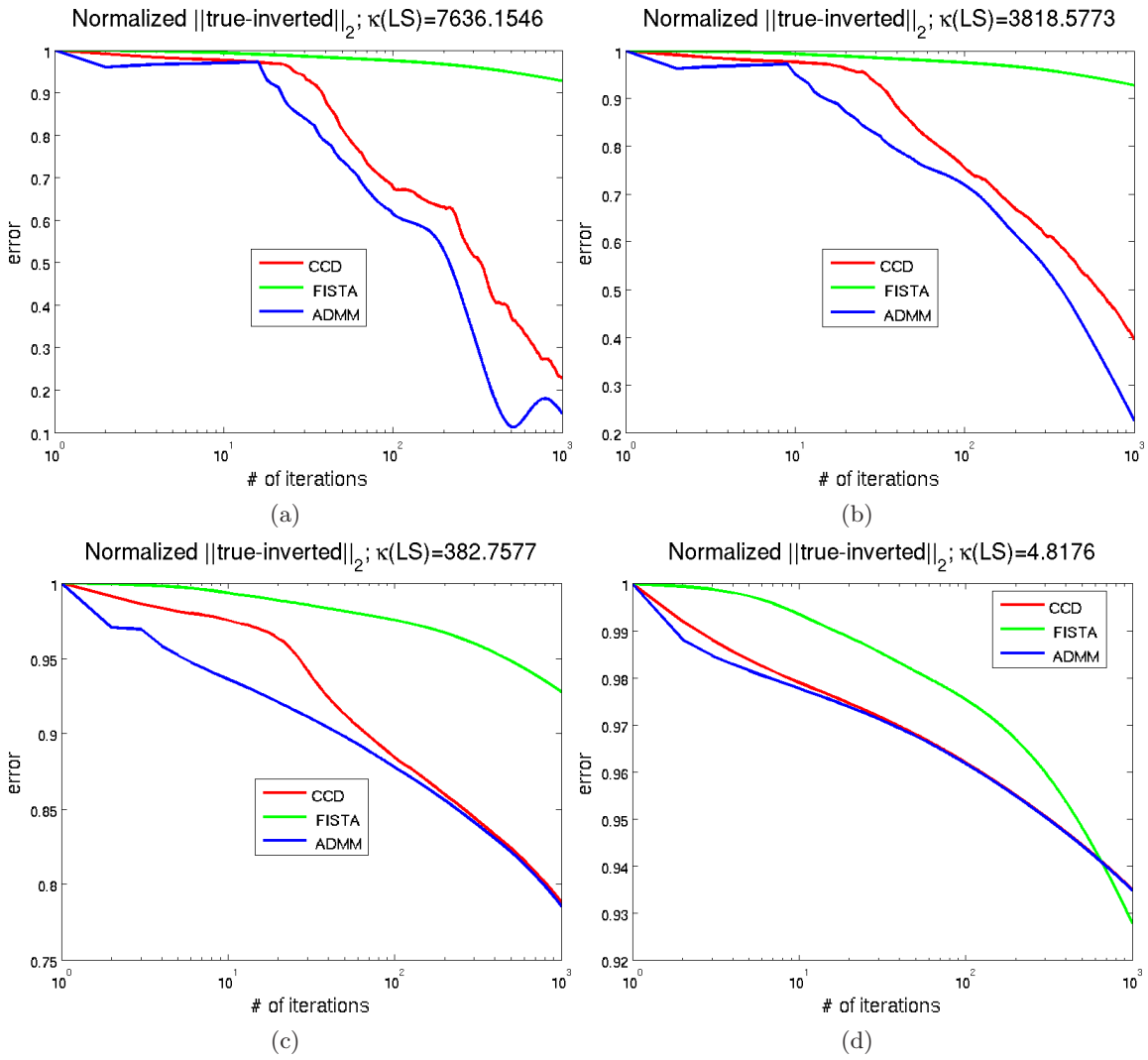
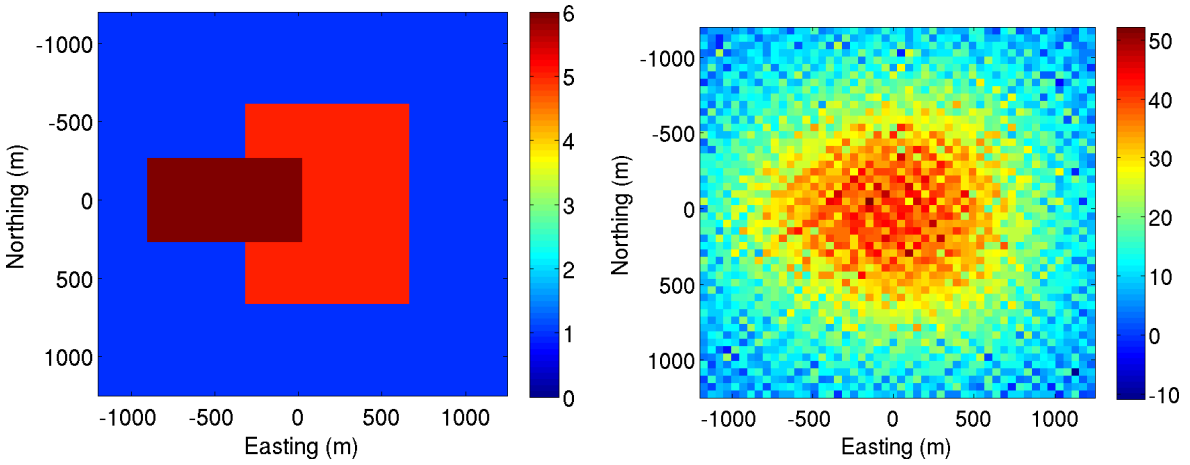


Figure 7: Convergence curves for CCD (solid red), ADMM with exact solver (blue), FISTA (green) for (a) $\lambda = .05$; (b) $\lambda = 0.1$; (c) $\lambda = 1$; (d) $\lambda = 100$. Limited-memory Compressive Conjugate Directions with $m = 100$ achieves convergence rate comparable to ADMM with exact minimization of (4.4).

1000 applications of $\mathbf{A}$ and $\mathbf{A}^T$ are shown in Figures 9(c) and 9(d), respectively.

The Compressive Conjugate Directions method resolves key model features faster than the ADMM using iterative solution of (4.4) restarted at each ADMM iteration. This advantage of our method is particularly pronounced when the intermediate least-squares problem (4.4) is ill-conditioned—compare Figures 10(a),10(b) with Figures 10(c),10(d). To accurately resolve the blocky pressure model of Figure 8(a), the Compressive Conjugate Directions technique requires about a tenth of operator $\mathbf{A}$ and adjoint applications compared with Algorithm 5



(a) (b)

Figure 8: (a) A blocky true pressure model (MPa); (b) the resulting surface displacements (mm) with added random Gaussian noise with $\sigma = 15\%$ of data amplitude.

when (4.4) is poorly conditioned. And again, as in the previous example, there is a trade-off between the convergence rate of the Compressive Conjugate Directions and the condition number of (4.4): values of λ that result in more poorly-conditioned (4.4) yield the fastest convergence.

6. Discussion. Compressive Conjugate Directions provides an efficient implementation of the Alternating Direction Method of Multipliers in $L_1 - TV$ regularized inversion problems (1.1) with computationally expensive operators $\mathbf{A}$. By accumulating and reusing information on the geometry of the intermediate quadratic objective function (4.4), the method requires only one application of the operator $\mathbf{A}$ and its adjoint per ADMM iteration while achieving accuracy comparable to that of the ADMM with exact minimization of (4.4). The method does not improve the worst-case asymptotic convergence rate of the ADMM. However, it can be used for fast recovery of spiky or blocky solution components. The method trades the computational cost of applying operator $\mathbf{A}$ and its adjoint for extra memory required to store previous conjugate direction vectors (4.2).

Our numerical experiments involving problems of geomechanical inversion demonstrated a trade-off between the number of ADMM iterations required to achieve a sufficiently accurate solution approximation, and condition number of the intermediate least-squares problem (4.4). Understanding the extent to which this phenomenon applies to solving (1.1) with other classes of modeling operators $\mathbf{A}$ requires further analysis.

6.1. Generalizations. The primary focus of this work are $L_1 - TV$ regularized inversion problems (1.1). However, the Steered Conjugate Directions Algorithm 2 can be combined with the Method of Multipliers to solve more general problems of large-scale equality-constrained optimization.

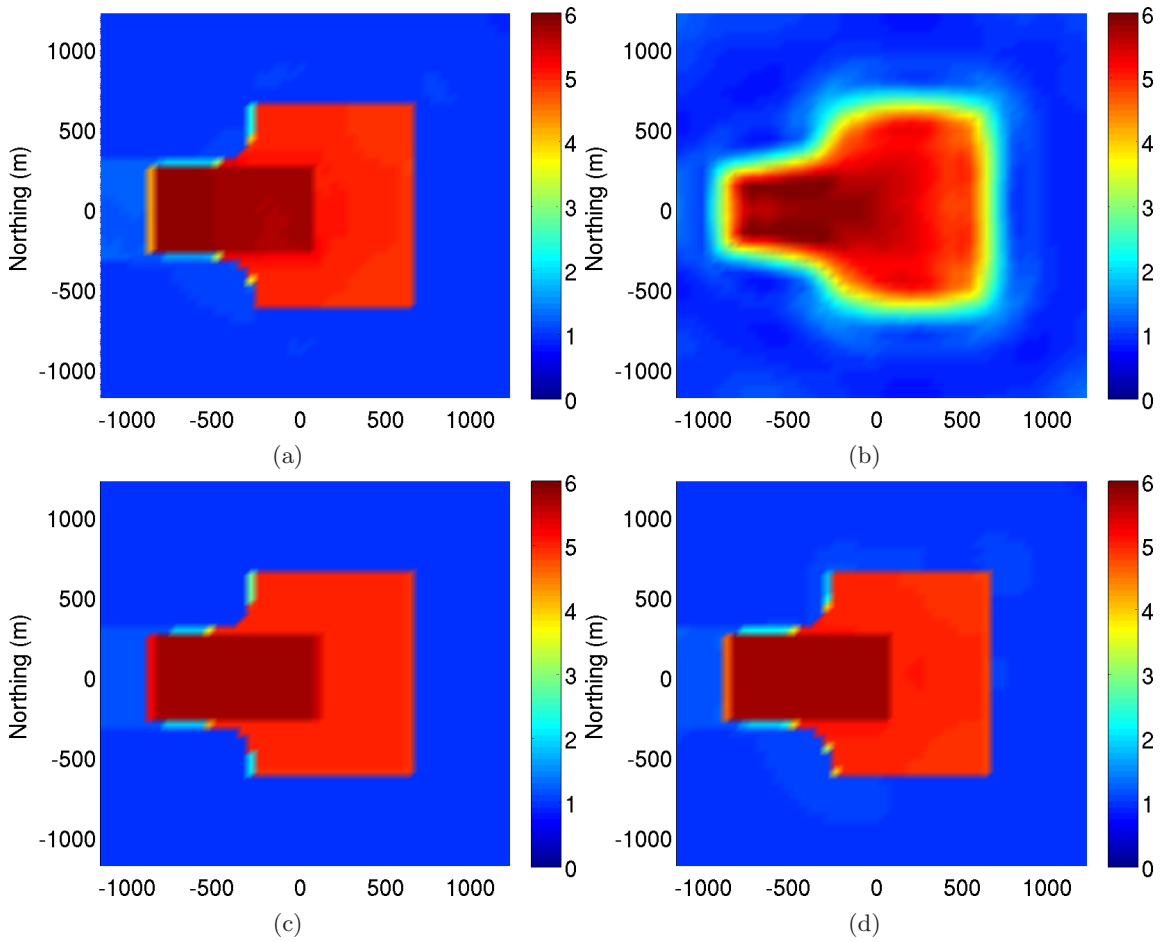


Figure 9: Inversion results after (a) 100 iterations (operator and adjoint applications) of CCD with $\lambda = 10$; (b) 100 iterations of RCG with $\lambda = 10$; (c) 1000 iterations of CCD with $\lambda = 10$; (d) 1000 iterations of RCG with $\lambda = 10$. In all tests, CCD is the limited-memory Compressive Conjugate Directions method of Algorithm 4; RCG is ADMM with restarted Conjugate Gradients of Algorithm 5 showing the most accurate model reconstruction among the outputs for different N_c–see Figures 10(a),10(b),10(c),10(d).

For example, consider the problem

$$\begin{aligned} &\|\mathbf{Au}-\mathbf{d}\|_2^2 \;\rightarrow\; \min, \\ &\mathbf{Bu}-\mathbf{c} \;=\; \mathbf{0}, \\ &\mathbf{u} \in \mathbb{R}^N,\ \mathbf{d} \in \mathbb{R}^M,\ \mathbf{A}:\mathbb{R}^N \rightarrow \mathbb{R}^M,\ \mathbf{B}:\mathbb{R}^N \rightarrow \mathbb{R}^K, \end{aligned} \tag{6.1}$$

where $\mathbf{A}$ is a computationally expensive operator. Many "coupled" systems governing two or more physical parameters can be described mathematically as a constrained problem (6.1). Of special interest are cases when $K \ll \min\{N, M\}$—e.g., large-scale optimization problems with a localized constraint. Applying the Augmented Lagrangian Method of Multipliers to

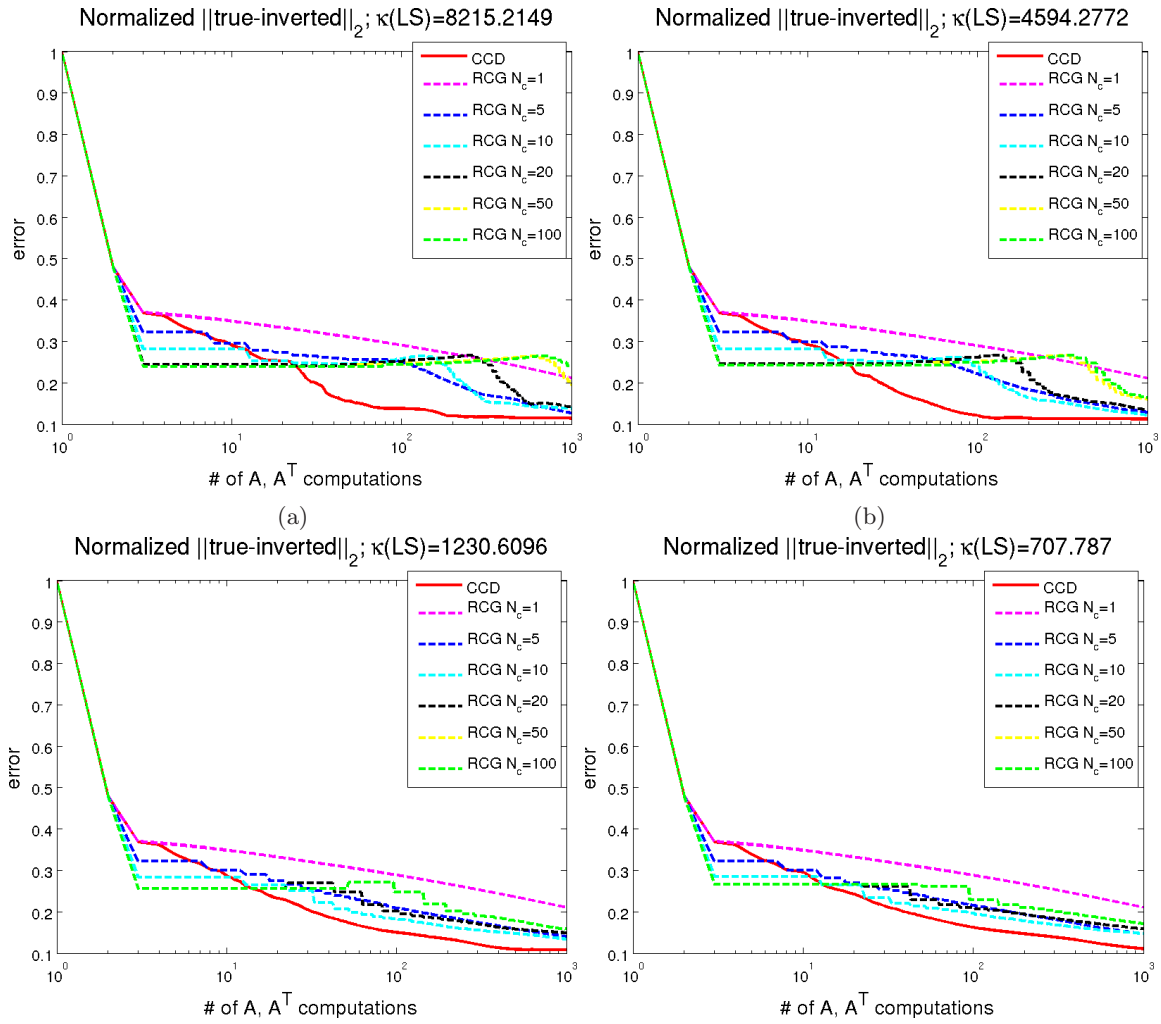


Figure 10: Convergence rates for CCD and RCG with various N_c for (a) $\lambda = 5$; (b) $\lambda = 10$; (c) $\lambda = 50$; (d) $\lambda = 100$.

(6.1), after re-scaling the multiplier vector, we get

$$\begin{aligned} \mathbf{u}_{k+1} &= \operatorname{argmin} \|\mathbf{A}\mathbf{u} - \mathbf{d}\|_2^2 + \frac{\lambda}{2}\|\mathbf{c} - \mathbf{B}\mathbf{u} + \mathbf{b}_k\|_2^2, \\ \mathbf{b}_{k+1} &= \mathbf{b}_k \; + \; \mathbf{c} - \mathbf{B}\mathbf{u}_{k+1}. \end{aligned} \tag{6.2}$$

As before, the minimization on the first line of (6.2) is equivalent to solving a system of normal equations with a fixed left-hand side and changing right-hand sides. Combining the dual-variable updates from (6.2) with Algorithm 2, we get Algorithm 6.

Operator $\mathbf{F}$ in Algorithm 6 is given by (2.2) with $\alpha = 1$. A limited-memory version of Algorithm 6 is obtained trivially by adapting Algorithm 4. We envisage potential utility of

Algorithm 6 Steered Conjugate Directions + Method of Multipliers for solving (6.1)

$\mathbf{u}_0 \leftarrow \mathbf{0}^N$, $\mathbf{b}_0 \leftarrow \mathbf{0}^K$, $\mathbf{v}_0 \leftarrow \begin{bmatrix} \mathbf{d} \\ \sqrt{\lambda}\,(\mathbf{c}+\mathbf{b}_0) \end{bmatrix}$
$\mathbf{p}_0 \leftarrow \mathbf{F}^T\mathbf{v}_0$, $\mathbf{q}_0 \leftarrow \mathbf{F}\mathbf{p}_0$, $\delta_0 \leftarrow \mathbf{q}_0^T\mathbf{q}_0$
for $k = 0, 1, 2, 3, \ldots$ **do**
    for $i = 0, 1, \ldots, k$ **do**
        $\tau_i \leftarrow \mathbf{q}_i^T\mathbf{v}_k/\delta_i$
    end for
    $\mathbf{u}_{k+1} \leftarrow \sum_{i=0}^{k} \tau_i\mathbf{p}_i$
    $\mathbf{b}_{k+1} \leftarrow \mathbf{b}_k + \mathbf{c} - \mathbf{B}\mathbf{u}_{k+1}$
    $\mathbf{v}_{k+1} \leftarrow \begin{bmatrix} \mathbf{d} \\ \sqrt{\lambda}\,(\mathbf{c}+\mathbf{b}_{k+1}) \end{bmatrix}$
    $\mathbf{r}_{k+1} \leftarrow \mathbf{v}_{k+1} - \sum_{i=0}^{k} \tau_i\mathbf{q}_i$
    $\mathbf{w}_{k+1} \leftarrow \mathbf{F}^T\mathbf{r}_{k+1}$
    $\mathbf{s}_{k+1} \leftarrow \mathbf{F}\mathbf{w}_{k+1}$
    for $i = 0, 1, \ldots, k$ **do**
        $\beta_i \leftarrow -\mathbf{q}_i^T\mathbf{s}_{k+1}/\delta_i$
    end for
    $\mathbf{p}_{k+1} \leftarrow \sum_{i=0}^{k} \beta_i\mathbf{p}_i + \mathbf{w}_{k+1}$
    $\mathbf{q}_{k+1} \leftarrow \sum_{i=0}^{k} \beta_i\mathbf{q}_i + \mathbf{s}_{k+1}$
    $\delta_{k+1} \leftarrow \mathbf{q}_{k+1}^T\mathbf{q}_{k+1}$
    if $\delta_{k+1} = 0$ **then** ▷ Use condition "$\delta_{k+1} <$ tolerance" in practice
        $\delta_{k+1} \leftarrow 1$, $\mathbf{p}_{k+1} \leftarrow \mathbf{0}^N$, $\mathbf{q}_{k+1} \leftarrow \mathbf{0}^{M+K}$
    end if
    Exit loop if $\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_2/\|\mathbf{u}_k\|_2 \le$ target accuracy
end for

Algorithm 6 in applications where storing a set of previous conjugate direction vectors (4.2) is computationally more efficient that iteratively solving the quadratic minimization problem in (6.2) from scratch at each iteration of the method of multipliers.

The Compressive Conjugate Directions Algorithm 4 can be extended for solving non-linear inversion problems with L_1 and *isotropic* total-variation regularization. Likewise, the Steered Conjugate Directions Algorithm 6 can be adapted to solving general equality-constrained non-linear optimization problems. A nonlinear theory and further applications of these techniques will be the subject of our next work.

REFERENCES

[1] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, 2011.

[2] A. Beck and M. Teboulle, *Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems*, IEEE Transactions on Image Processing, 18 (2009), pp. 2419–2434.

[3] ———, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on

Imaging Sciences, 2 (2009), pp. 183–202.
[4] J. M. Bioucas-Dias and M. A.T. Figueiredo, *A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration*, Trans. Img. Proc., 16 (2007), pp. 2992–3004.
[5] A. Björk, *Numerical Methods for Least Squares Problems*, SIAM, 1996.
[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, 3 (2010), pp. 1–122.
[7] S. P. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
[8] R. E. Bruck Jr., *On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in Hilbert space*, Journal of Mathematical Analysis and Applications, 61 (1977), pp. 159 – 164.
[9] A. Chambolle, *An algorithm for total variation minimization and applications*, Journal of Mathematical Imaging and Vision, 20 (2004), pp. 89–97.
[10] A. Chambolle and P. L. Lions, *Image recovery via total variational minimization and related problems*, Numerische Mathematik, 76 (1997), pp. 167–188.
[11] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, Multiscale Modeling & Simulation, 4 (2005), pp. 1168–1200.
[12] I. Daubechies, M. Defrise, and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Communications on Pure and Applied Mathematics, 57 (2004), pp. 1413–1457.
[13] J. Douglas and H. H. Rachford, *On the numerical solution of heat conduction problems in two and three space variables*, Transactions of the American mathematical Society, 82 (1956), pp. 421–439.
[14] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program., 55 (1992), pp. 293–318.
[15] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least angle regression*, The Annals of Statistics, 32 (2004), pp. 407–499.
[16] A. Fichtner, *Full Seismic Modeling and Inversion*, Springer, 2011.
[17] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, *Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems*, IEEE Journal of Selected Topics in Signal Processing, 1 (2007), pp. 586–597.
[18] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Computers & Mathematics with Applications, 2 (1976), pp. 17 – 40.
[19] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*, Society for Industrial and Applied Mathematics, 1989.
[20] R. Glowinski and A. Marroco, *Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires*, ESAIM: Mathematical Modelling and Numerical Analysis – Modélisation Mathématique et Analyse Numérique, 9 (1975), pp. 41–76.
[21] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, *Fast alternating direction optimization methods*, SIAM Journal on Imaging Sciences, 7 (2014), pp. 1588–1623.
[22] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 323–343.
[23] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, *The entire regularization path for the support vector machine*, J. Mach. Learn. Res., 5 (2004), pp. 1391–1415.
[24] B. He and X. Yuan, *On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method*, SIAM Journal on Numerical Analysis, 50 (2012), pp. 700–709.
[25] M. R. Hestenes, *Multiplier and gradient methods*, Journal of Optimization Theory and Applications, 4 (1969), pp. 303–320.
[26] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, *An interior-point method for large-scale ℓ_1-regularized least squares*, Selected Topics in Signal Processing, IEEE Journal of, 1 (2007), pp. 606–617.
[27] D. Kosloff, R.F. Scott, and J. Scranton, *Finite element simulation of Wilmington oil field subsidence: I. Linear modelling*, Tectonophysics, 65 (1980), pp. 339 – 368.
[28] M. Maharramov and M. Zoback, *Monitoring of cyclic steam stimulation by inversion of surface tilt*

measurements, AGU Fall Meeting, Session H23A-0859, (2014).
[29] Y. E. Nesterov, *A method for solving the convex programming problem with rate of convergence* $O(1/k^2)$, Dokl. Akad. Nauk SSSR, 269 (1983), pp. 543–547.
[30] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 2006.
[31] M. R. Osborne, B Presnell, and B. A. Turlach, *A new approach to variable selection in least squares problems*, IMA Journal of Numerical Analysis, 20 (2000), pp. 389–403.
[32] G. B. Passty, *Ergodic convergence to a zero of the sum of monotone operators in Hilbert space*, Journal of Mathematical Analysis and Applications, 72 (1979), pp. 383 – 390.
[33] Y. Qiu, W. Xue, and G. Yu, *Intelligent Science and Intelligent Data Engineering: Third Sino-foreign-interchange Workshop, IScIDE 2012, Nanjing, China, October 15-17, 2012. Revised Selected Papers*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, ch. A Projected Conjugate Gradient Method for Compressive Sensing, pp. 398–406.
[34] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1971.
[35] ———, *Augmented lagrangians and applications of the proximal point algorithm in convex programming*, Mathematics of Operations Research, 1 (1976), pp. 97–116.
[36] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: Nonlinear Phenomena, 60 (1992), pp. 259–268.
[37] Y. Saad, *Iterative Methods for Sparse Linear Systems, second edition*, SIAM, 2003.
[38] P. Segall, *Earth and Volcano Deformation*, Princeton University Press, 2010.
[39] A. Tarantola, *Inversion of seismic reflection data in the acoustic approximation*, Geophysics, 49 (1984), pp. 1259–1266.
[40] L. N. Trefethen and David Bau III, *Numerical Linear Algebra*, SIAM, 1997.
[41] H. Uzawa, *Studies in Linear and Non-Linear Programming*, Stanford University Press, 1958, ch. Iterative methods for concave programming.
[42] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.
[43] X. Zhang, M. Burger, and S. Osher, *A unified primal-dual algorithm framework based on Bregman iteration*, Journal of Scientific Computing, 46 (2010), pp. 20–46.