# Basis Pursuit Denoise with Nonsmooth Constraints

Robert Baraldi[1], Rajiv Kumar[2], and Aleksandr Aravkin[1].

[1] Department of Applied Mathematics, University of Washington

[2] Formerly School of Earth and Atmospheric Sciences, Georgia Institute of Technology, USA; Currently DownUnder GeoSolutions, Perth, Australia

*Abstract*—Level-set optimization formulations with data-driven constraints minimize a regularization functional subject to matching observations to a given error level. These formulations are widely used, particularly for matrix completion and sparsity promotion in data interpolation and denoising. The misfit level is typically measured in the $\ell_2$ norm, or other smooth metrics.

In this paper, we present a new flexible algorithmic framework that targets *nonsmooth* level-set constraints, including $\ell_1$, $\ell_\infty$, and even $\ell_0$ norms. These constraints give greater flexibility for modeling deviations in observation and denoising, and have significant impact on the solution. Measuring error in the $\ell_1$ and $\ell_0$ norms makes the result more robust to large outliers, while matching many observations exactly.

We demonstrate the approach for basis pursuit denoise (BPDN) problems as well as for extensions of BPDN to matrix factorization, with applications to interpolation and denoising of 5D seismic data. The new methods are particularly promising for seismic applications, where the amplitude in the data varies significantly, and measurement noise in low-amplitude regions can wreak havoc for standard Gaussian error models.

*Index Terms*—Nonconvex nonsmooth optimization, level-set formulations, basis pursuit denoise, interpolation, seismic data.

## I. INTRODUCTION

Basis Pursuit Denoise (BPDN) seeks a sparse solution to an under-determined system of equations that have been corrupted by noise. The classic level-set formulation [22], [2] is given by

$$\min_x \|x\|_1 \quad \text{s.t.} \quad \|\mathcal{A}(x) - b\|_2 \leq \sigma \tag{1}$$

where $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^d$ is a linear functional taking unknown parameters $x \in \mathbb{R}^{m \times n}$ to observations $b \in \mathbb{R}^d$. Problem (1) is also known as a Morozov formulation (in contrast to Ivanov or Tikhonov [17]). The functional $\mathcal{A}$ can include a transformation to another domain, including Wavelets, Fourier, or Curvelet coefficients [7], as well as compositions of these transforms with other linear operators such as restriction in interpolation problems. The parameter $\sigma$ controls the error budget, and is based on an estimate of noise level in the data.

Theoretical recovery guarantees for classes of operators $\mathcal{A}$ are developed in [6] and [20]. BPDN and the closely related LASSO formulation have applications to compressed sensing [18], [6] and machine learning [11], [10], as well as to applied domains including MRI [16]. Seismic data is a key use case [3], [15], [9], where acquisition is prohibitively expensive and interpolation techniques are used to fill in data volumes by promoting parsimonious representations in the Fourier [19] or Curvelet [12] domains. Matricization of the data leads to low-rank interpolation schemes [3], [15], [9], [24].

While BPDN uses nonsmooth regularizers (including the $\ell_1$ norm, nuclear norm, and elastic net), the inequality constraint is ubiquitously smooth, and often taken to be the $\ell_2$ norm as in (1). Prior work, including [23], [3], [9], [2], exploits the smoothness of the inequality constraint in developing algorithms for the problem class. Smooth constraints work well when errors are Gaussian, but this assumption fails for seismic data and is often violated in general.

**Contributions.** The main contribution of this paper is to provide a fast, easily adaptable algorithm to solve non-smooth and nonconvex data constraints in general level-set formulations including BPDN, and illustrate the efficacy of the approach using large-scale interpolation and denoising problems. To do this, we extend the universal regularization framework of [26] to level-set formulations with nonsmooth/nonconvex constraints. We develop a convergence theory for the optimization approach, and illustrate the practical performance of the new formulations for data interpolation and denoising in both sparse recovery and low-rank matrix factorization.

**Roadmap.** The paper proceeds as follows. Section II develops the general relaxation framework and approach. Section III specifies this framework to the BPDN setting with nonsmooth, nonconvex constraints. In Section IV we apply the approach to sparse signal recovery problem and sparse Curvelet reconstruction. In Section V, we extend the approach to a low-rank interpolation framework, which embeds matrix factorization within the BPDN constraint. In Section VI we test the low-rank extension using synthetic examples and data extracted from a full 5D dataset simulated on complex SEG/EAGE overthrust model.

## II. NONSMOOTH, NONCONVEX LEVEL-SET FORMULATIONS.

We consider the following problem class:

$$\min_x \phi(\mathcal{C}(x)) \quad \text{s.t.} \quad \psi(\mathcal{A}(x) - b) \leq \sigma, \tag{2}$$

where $\phi$ and $\psi$ may be nonsmooth, nonconvex, but have well-defined proximity and projection operators:

$$\text{prox}_{\alpha\phi}(y) = \arg\min_x \frac{1}{2\alpha}\|x - y\|^2 + \phi(x)$$
$$\text{proj}_{\psi(\cdot) \leq \sigma} = \arg\min_{\psi(x) \leq \sigma} \frac{1}{2\alpha}\|x - y\|^2. \tag{3}$$

Here, $\mathcal{C} : \mathbb{C}^{m \times n} \to \mathbb{R}^c$ is typically a linear operator that converts $x$ to some transform domain, while $\mathcal{A} : \mathbb{C}^{m \times n} \to \mathbb{R}^d$ is a linear observation operator also acting on $x$. In the context of interpolation, $\mathcal{A}$ is often a restriction operator.

**Algorithm 1** Prox-gradient for (4).

1: **Input:** $x^0, w_1^0, w_2^0$
2: Initialize: $k = 0$
3: **while** not converged **do**
4: $\quad x^{k+1} \leftarrow x^k - \alpha \left( \frac{1}{\eta_1} \mathcal{C}^T (\mathcal{C}(x) - w_1) \right.$
$\quad\quad\quad\quad\quad\quad \left. + \frac{1}{\eta_2} \mathcal{A}^T (\mathcal{A}(x) - w_2 - b) \right)$
5: $\quad w_1^{k+1} \leftarrow \text{prox}_{\frac{\eta_1}{\alpha} \phi} \left( w_1^k - \frac{\alpha}{\eta_1} (w_1^k - \mathcal{C}(x^{k+1})) \right)$
6: $\quad w_2^{k+1} \leftarrow \text{proj}_{\sigma \mathbb{B}_\psi} \left( w_2^k - \frac{\alpha}{\eta_2} (w_2 - (\mathcal{A}(x^{k+1}) - b)) \right)$
7: $\quad k \leftarrow k + 1$
8: **end while**
9: **Output:** $w_1^k, w_2^k, x^k$

**Algorithm 2** Value-function optimization for (4).

1: **Input:** $x^0, w_1^0, w_2^0$
2: Initialize: $k = 0$
3: Define: $\mathcal{H} = \frac{1}{\eta_1} \mathcal{C}^T \mathcal{C} + \frac{1}{\eta_2} \mathcal{A}^T \mathcal{A}$
4: **while** not converged **do**
5: $\quad x^{k+1} \leftarrow \mathcal{H}^{-1} \left( \frac{1}{\eta_1} \mathcal{C}^T w_1^k + \frac{1}{\eta_2} \mathcal{A}^T (b + w_2^k) \right)$
6: $\quad w_1^{k+1} \leftarrow \text{prox}_{\frac{\eta_1}{\beta} \phi} \left( w_1^k - \frac{\beta}{\eta_1} (w_1^k - \mathcal{C}(x^{k+1})) \right)$
7: $\quad w_2^{k+1} \leftarrow \text{proj}_{\sigma \mathbb{B}_\psi} \left( w_2^k - \frac{\beta}{\eta_2} (w_2 - (\mathcal{A}(x^{k+1}) - b)) \right)$
8: $\quad k \leftarrow k + 1$
9: **end while**
10: **Output:** $w_1^k, w_2^k, x^k$

This setting significantly extends that of [2], who assume $\psi$ and $\phi$ are convex, $\mathcal{C} = I$, and use the *value function*

$$v(\tau) = \min_x \psi(\mathcal{A}(x) - b) \quad \text{s.t.} \quad \phi(x) \leq \tau$$

to solve (2) using root-finding to solve $v(\tau) = \sigma$. Variational properties of $v$ are fully only understood in the convex setting, and efficient evaluation of $v(\tau)$ requires $\psi$ to be smooth, so that efficient first-order methods are applicable.

Here, we develop an approach to solve any problem of type (2), including problems with nonsmooth and nonconvex $\psi, \phi$, using only matrix vector products with $\mathcal{A}, \mathcal{A}^T, \mathcal{C}, \mathcal{C}^T$ and simple nonlinear operators. In special cases, the approach can also use equation solves to gain significant speedup.

The general approach uses the relaxation formulation proposed in [26], [25]. We use relaxation to split $\phi, \psi$ from the linear map $\mathcal{A}$ and transformation map $\mathcal{C}$, extending (2) to

$$\min_{x, w_1, w_2} \phi(w_1) + \frac{1}{2\eta_1} \|\mathcal{C}(x) - w_1\|^2 + \frac{1}{2\eta_2} \|w_2 - \mathcal{A}(x) + b\|_2^2$$
$$\text{s.t.} \quad \psi(w_2) \leq \sigma.$$
(4)

with $w_1 \in \mathbb{R}^c$ and $w_2 \in \mathbb{R}^d$. In contrast to [26], we use a continuation scheme to force $\eta_i \to 0$, in order to solve the original formulation (2). Thus the only external algorithmic parameter the scheme requires is $\sigma$, which controls the error budget for $\psi$.

There are two algorithms readily available to solve (4). The first is prox-gradient descent, detailed in Algorithm 1. We let $z = (x, w_1, w_2)$, and define

$$\Phi(z) = \phi(w_1) + \delta_{\psi(\cdot) \leq \sigma}(w_2),$$

where the *indicator function* $\delta_{\psi(\cdot) \leq \sigma}$ takes the value 0 if $\psi(w_2) \leq \sigma$, and infinity otherwise. Problem (4) can now be written as

$$\min_z \frac{1}{2} \underbrace{\left\| \begin{bmatrix} \frac{1}{\sqrt{\eta_1}} \mathcal{C} & -\frac{1}{\sqrt{\eta_1}} I & 0 \\ \frac{1}{\sqrt{\eta_2}} \mathcal{A} & 0 & -\frac{1}{\sqrt{\eta_2}} I \end{bmatrix} z - \begin{bmatrix} 0 \\ b \\ 0 \end{bmatrix} \right\|^2}_{f(z)} + \Phi(z).$$
(5)

Applying the prox-gradient descent iteration with step-size $\alpha$

$$z^{k+1} = \text{prox}_{\alpha\Phi}(z^k - \alpha \nabla f(z^k)) \tag{6}$$

gives the coordinate updates in Algorithm 1.

Prox-gradient has been analyzed in the general nonconvex setting by [4]. However, Problem (5) is the sum of a convex quadratic and a nonconvex regularizer. The rate of convergence for this problem class can be quantified, and [26, Theorem 2], reproduced below, will be very useful here.

**Theorem II.1** (Prox-gradient for Regularized Least Squares). *Consider the least squares objective*

$$\min_z p(z) := \frac{1}{2} \|Az - a\|^2 + \Phi(z).$$

*with $p$ bounded below, and $\Phi$ potentially nonsmooth, nonconvex, and non-finite valued. With step $\alpha = \frac{1}{\sigma_{\max}}$, the iterates (6) satisfy*

$$\min_{k=0,\dots,N} \|v_{k+1}\|^2 \leq \frac{\|A\|^2}{N} (p(z_0) - \inf p)$$

*where*

$$v_k = (\|A\|_2^2 I - A^T A)(x_k - x_{k+1})$$

*is a subgradient (generalized gradient) of $p$ at $z^k$.*

We can specialize Theorem II.1 to our case by computing the norm of the least squares system in (5).

**Corollary II.2** (Rate for Algorithm 1). *Theorem II.1 applied to problem 4 gives*

$$\min_{k=0,\dots,N} \|v_{k+1}\|^2 \leq C(\eta_1, \eta_2, \mathcal{C}, \mathcal{A}) \frac{1}{N} (p(z_0) - \inf p)$$

*with*

$$C(\eta_1, \eta_2, \mathcal{C}, \mathcal{A}) = \frac{1}{\eta_1} (c + \|\mathcal{C}\|_F^2) + \frac{1}{\eta_2} (d + \|\mathcal{A}\|_F^2).$$

Problem (4) also admits a different optimization strategy, summarized in Algorithm 2. We can formally minimize the objective in $x$ directly via the gradient, with the minimizer given by

$$x(w) = \mathcal{H}^{-1} \left( \frac{1}{\eta_1} \mathcal{C}^T w_1 + \frac{1}{\eta_2} \mathcal{A}^T (w_2 + b) \right)$$
$$\mathcal{H} = \frac{1}{\eta_1} \mathcal{C}^T \mathcal{C} + \frac{1}{\eta_2} \mathcal{A}^T \mathcal{A}$$

with $w = (w_1, w_2)$. Plugging this expression back in gives a regularized least squares problem in $w$ alone:

$$\min_{w_1, w_2} p(w) := \phi(w_1) + \left\| \mathcal{F} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \tilde{b} \right\|^2 \quad \text{s.t.} \quad \psi(w_2) \leq \sigma$$

$$\mathcal{F} = \begin{bmatrix} \frac{1}{\sqrt{\eta_1}} \left( \frac{1}{\eta_1} \mathcal{C} \mathcal{H}^{-1} \mathcal{C}^T - I \right) & \frac{1}{\sqrt{\eta_1 \eta_2}} \mathcal{C} \mathcal{H}^{-1} \mathcal{A}^T \\ \frac{-1}{\sqrt{\eta_2 \eta_1}} \mathcal{A} \mathcal{H}^{-1} \mathcal{C}^T & \frac{1}{\sqrt{\eta_2}} \left( I - \frac{1}{\eta_1} \mathcal{A} \mathcal{H}^{-1} \mathcal{A}^T \right) \end{bmatrix}$$

$$\tilde{b} = \begin{bmatrix} \frac{-1}{\sqrt{\eta_1 \eta_2}} \mathcal{C} \mathcal{H}^{-1} \mathcal{A}^T b \\ \frac{1}{\sqrt{\eta_2}} \left( \frac{1}{\eta_1} \mathcal{A} \mathcal{H}^{-1} \mathcal{A}^T - I \right) b \end{bmatrix}. \tag{7}$$

Prox-gradient applied to the *value function* $p(w)$ in (7) with step $\beta$ gives the iteration

$$w^+ = \text{prox}_{\frac{1}{\beta} \Phi} (w^k - \beta \mathcal{F}^T (\mathcal{F} w - \tilde{b})) \tag{8}$$

This iteration, as formally written, requires forming and applying the system $\mathcal{F}$ in (7) at each iteration. In practice we compute the $x(w)$ update on the fly, as detailed in Algorithm 2. The equivalence of Algorithm 2 to iteration (8) comes from the following derivative formula for value functions [5]:

$$\mathcal{F}^T (\mathcal{F} w - \tilde{b})) = \frac{1}{\eta_1} \mathcal{C}^T (\mathcal{C}(x(w)) - w_1)$$
$$+ \frac{1}{\eta_2} \mathcal{A}^T (\mathcal{A}(x(w)) - (w_2 + b)).$$

In order to compute $\beta$, and apply Theorem II.1, we first prove the following lemma:

**Lemma II.3** (Bound on $\|\mathcal{F}^T \mathcal{F}\|_2$). *The operator norm $\|\mathcal{F}^T \mathcal{F}\|_2$ is bounded above by* $\max \left( \frac{1}{\eta_1}, \frac{1}{\eta_2} \right)$.

*Proof.* Considering the function

$$\|\mathcal{F} w - \tilde{b}\|^2 = \min_x \underbrace{\frac{1}{2\eta_1} \|\mathcal{C}(x) - w_1\|^2 + \frac{1}{2\eta_2} \|w_2 - \mathcal{A}(x) + b\|_2^2}_{Q(x,w)},$$

we know that the gradient is given by $\mathcal{F}^T (\mathcal{F} w - \tilde{b})$, and any Lipschitz bound $L$ gives

$$\|\mathcal{F}^T \mathcal{F} w_1 - \mathcal{F}^T \mathcal{F} w_2\| \leq L \|w_1 - w_2\|,$$

which means $\|\mathcal{F}^T \mathcal{F}\|_2 \leq L$. On the other hand, we can write the right hand side as

$$Q(w, x) = q(Dw, x)$$

where

$$q(z, x) = \frac{1}{2} \left\| z - \begin{bmatrix} \frac{1}{2\sqrt{\eta_1}} \mathcal{C}(x) \\ \frac{1}{2\sqrt{\eta_2}} \mathcal{A}(x) \end{bmatrix} - \begin{bmatrix} 0 \\ b \end{bmatrix} \right\|^2$$

and

$$D = \begin{bmatrix} \frac{1}{\sqrt{\eta_1}} & 0 \\ 0 & \frac{1}{\sqrt{\eta_1}} \end{bmatrix}.$$

Using Theorem 1 of [25] with $g(z) = 0$, we have that the value function

$$\tilde{q}(z) = \min_x q(z, x)$$

is differentiable, with $\text{lip}(\nabla \tilde{q}) \leq 1$. Therefore

$$\tilde{Q}(w) = \min_x Q(w, x)$$

---

**Algorithm 3** Block-coordinate descent for (4).

1: **Input:** $x^0, w_1^0, w_2^0$
2: Initialize: $k = 0$
3: Define: $\mathcal{H} = \frac{1}{\eta_1} \mathcal{C}^T \mathcal{C} + \frac{1}{\eta_2} \mathcal{A}^T \mathcal{A}$
4: **while** not converged **do**
5: $\quad x^{k+1} \leftarrow \mathcal{H}^{-1} \left( \frac{1}{\eta_1} \mathcal{C}^T w_1^k + \frac{1}{\eta_2} \mathcal{A}^T (b + w_2^k) \right)$
6: $\quad w_1^{k+1} \leftarrow \text{prox}_\phi \left( \mathcal{C}(x^{k+1}) \right)$
7: $\quad w_2^{k+1} \leftarrow \text{proj}_{\sigma \mathbb{B}_\psi} \left( \mathcal{A}(x^{k+1}) - b \right)$
8: $\quad k \leftarrow k + 1$
9: **end while**
10: **Output:** $w_1^k, w_2^k, x^k$

---

is also differentiable, with

$$\nabla \tilde{Q}(w) = D' \nabla \tilde{q}(Dw),$$

and hence

$$\text{lip}(\nabla \tilde{Q}) \leq \|D^T D\|_2 = \max \left( \frac{1}{\eta_1}, \frac{1}{\eta_2} \right).$$

This immediately gives the result. $\square$

Now we can combine iteration (8) with Theorem II.1 to get a rate of convergence for Algorithm 2.

**Corollary II.4** (Convergence of Algorithm 2). *When $\beta$ satisfies*

$$\beta \leq \min(\eta_1, \eta_2),$$

*the iterates of Algorithm 2 satisfy*

$$\min_{k=0,...,N} \|v_{k+1}\|^2 \leq \frac{1}{N} \max \left( \frac{1}{\eta_1}, \frac{1}{\eta_2} \right) (p(w_0) - \inf p))$$

*where $v_k$ is in the subdifferential (generalized gradient) of objective (7) at $w^k$. Moreover, if $\eta_1 = \eta_2$, then Algorithm (2) is equivalent to block-coordinate descent, as detailed in Algorithm 3.*

*Proof.* The convergence statement comes directly from plugging the estimate of iteration 8 into Theorem II.1. The equivalence of Algorithm 3 with Algorithm 2 is obtained by plugging in step size $\beta = \eta_1 = \eta_2$ into each line of Algorithm 2. $\square$

An important consequence of Corollary II.4 is that the convergence rate of Algorithm 2 does not depend on $\mathcal{C}$ or $\mathcal{A}$, in contrast to Algorithm 1, whose rate depends on both matrices (Corollary II.2). The rates of both algorithms are affected by $(\eta_1, \eta_2)$. We use continuation in $\eta$, driving $(\eta_1, \eta_2)$ to $(0, 0)$ at the same rate, and warm-starting each problem at the previous solution. A convergence theory that takes this continuation into account is left to future work.

TABLE I
SNR VALUES AGAINS THE TRUE $x$ FOR DIFFERENT $\ell_p$ NORMS WITH
ALGORITHM 3.

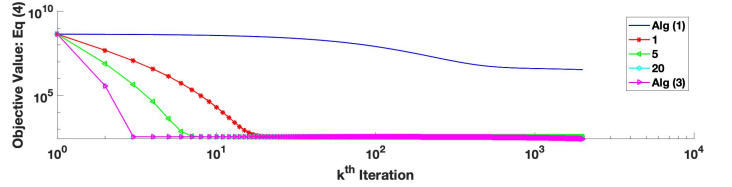| BPDN with Random Linear Operator | |
|---|---|
| Method/Norm | SNR |
| $\ell_2$ with SPGL1 | 0.2007 |
| $\ell_2$ with Alg.3 | 0.2032 |
| $\ell_1$ with Alg.3 | 33.7281 |
| $\ell_\infty$ with Alg.3 | -0.6708 |
| $\ell_0$ with Alg.3 | 45.0601 |



Fig. 1. Objective function decay for Equation 4 with proximal-gradient descent (Algorithm 1), Direct solving (Algorithm 3), and several steps in between where we only partially solve for $\mathcal{H}^{-1}(\ldots)$ with Algorithm 2.

### A. Inexact Least-Squares Solves.

Algorithm 3 has a provably faster rate of convergence than Algorithm 1. The practical performance of these algorithms is compared in Figure 1, which is solving a problem with both a $\ell_1$ norm regularizer and $\ell_1$ norm BPDN constraint, with $\alpha = \|\mathcal{A}\|_F^{-2}$, $\mathcal{C} = I$, and $\eta_1 = \eta_2 = 10^{-4}$. We see a huge performance difference in practice as well as in theory: the proximal gradient descent from Algorithm 1 yields a slower cost function decay than solving exactly for $x(w)$ as in Algorithm 3. Indeed, Algorithm 3 admits the fastest cost function decay as shown in Corollary II.4, albeit at the expense of more operations per iteration. This is due to the fact that fully solving the least squares problem in Line 5 is not tractable for large-scale problems. Hence, we implement Algorithm 3 inexactly, using the Conjugate Gradient (CG) algorithm. Figure 1 shows the results when we use 1, 5, and 20 CG iterations. Each CG iteration is implemented using matrix-vector products, and at 20 iterations the results are indistinguishable from those of Algorithm 3 with full solves. Even at 5 iterations, the performance is remarkably close to that of of Algorithm 3 with full solves. Algorithm 3 has a natural warm-start strategy, with the $x$ from each previous iteration used in the subsequent LS solve using CG. Using a CG method with a bounded number of iterates gives fast convergence and saves computational time. This approach is used in the subsequent experiments.

### III. APPLICATION TO BASIS PURSUIT DE-NOISE MODELS

The Basis Pursuit De-noise problem can be formulated as

$$\min_x \|x\|_1 \quad \text{s.t.} \quad \rho\left(\mathcal{A}(x) - b\right) \le \sigma \tag{9}$$

where $\rho(\cdot)$ is classically taken to be the $\ell_2$-norm. In this problem, $x$ represents unknown coefficients that are sparse in a transform domain, while $\mathcal{A}$ is a composition of the observation operator with a transform matrix; popular examples of transform domains include discrete cosine transforms, wavelets, and curvelets. The observed and noisy data $b$ resides in the temporal/spatial domain, and $\sigma$ is the misfit tolerance. This problem was famously solved with the SPGL1 [23] algorithm for $\rho(\cdot) = \|\cdot\|_2$. When the observed data is affected by large sparse noise, a smooth constraint is ineffective. A nonsmooth variant of (9) is very difficult for approaches such as SPGL1, which solves subproblems of the form

$$\min_x \rho\left(\mathcal{A}(x) - b\right) \quad \text{s.t.} \quad \|x\|_1 \le \tau.$$

However, the proposed Algorithm 2 is easily adaptable to different norms. We apply Algorithm 3 with $\phi(x) = \|x\|_1$,

taking $(\eta_1, \eta_2) \to (0, 0)$ so that $(w_1, w_2) \to (x, \mathcal{A}(x) - b)$. We can take many different $\psi$, including $\ell_2, \ell_1, \ell_\infty$, and $\ell_0$.

Algorithm 3 is simple to implement. The least squares update in step 4 can be computed efficiently using either factorization with Woodbury, or an iterative method in cases where $\mathcal{A}$ is too large to store. For the Woodbury approach, we have

$$\left(\eta_2 + \eta_1 \mathcal{A}^T \mathcal{A}\right)^{-1} = \frac{1}{\eta_2} I - \frac{1}{\eta_2^2} \mathcal{A}^T \left(\frac{1}{\eta_1} I + \frac{1}{\eta_2} \mathcal{A} \mathcal{A}^T\right)^{-1} \mathcal{A}. \tag{10}$$

For moderate size systems, we can store Cholesky factor

$$LL^T = \frac{1}{\eta_1} I + \frac{1}{\eta_2} \mathcal{A} \mathcal{A}^T,$$

with $L \in \mathbb{R}^{m \times m}$, and use $L$ with (10) to implement step 4. However, in the seismic/curvelet experiment described below, the left-hand side of Equation 10 is too large to store in memory, but is positive definite. Hence, we solve the resulting linear system in step 4 of Algorithm 3 with CG, using matrix-vector products. The $w_1$ update is implemented via the $\ell_1$-proximal operator (soft thresholding), while the $w_2$ update requires a projection onto the $\ell_p$ ball. The projectors used in our experiments are collected in Table II.

The least squares solve for $x$ is when $\mathcal{C}^T$ is an orthogonal matrix or tight frame, so that $\mathcal{C}^T \mathcal{C} = I$; this is the case for Fourier transforms, wavelets, and curvelets. When $\mathcal{A}$ is a restriction operator, as for many data interpolation problems, $\mathcal{A}^T \mathcal{A}$ is a diagonal matrix with zeros and ones, and hence

$$\mathcal{H} = \frac{1}{\eta_1} \mathcal{C}^T \mathcal{C} + \frac{1}{\eta_2} \mathcal{A}^T \mathcal{A}$$

is a diagonal matrix with entries either $\frac{1}{\eta_1}$ or $\frac{1}{\eta_1} + \frac{1}{\eta_2}$; the least squares problem for the $x$ update is then trivial.

### IV. BASIS PURSUIT DE-NOISE EXPERIMENTS

In this application, we consider two examples: the first is a small-scale BPDN to illustrate the proof of concept of our technique, while the second is an application to de-noising a common source gather extracted from a seismic line simulated using a 2D BG Compass model. The data set contains time samples with a temporal-interval of 4ms, and the spatial sampling is 10m. For this example, we use curvelets as a

TABLE II
PROJECTORS FOR $\ell_p$ BALLS.

| Norm | $\ell(x)$ | $\mathrm{proj}_{\tau\mathbb{B}_\ell}(z)$ | Solution |
|---|---|---|---|
| $\ell_2$ | $\sqrt{\sum_i x_i^2}$ | $\begin{cases} z, & \|z\| < \tau \\ \tau z/\|z\|_2, & \|z\| > \tau \end{cases}$ | Analytic |
| $\ell_\infty$ | $\max_i |x_i|$ | $\max(\min(x,1),-1)$ | Analytic |
| $\ell_1$ | $\sum_i |x_i|$ | See e.g. [22] | $O(n \ln n)$ routine |
| $\ell_0$ | $\sum_i \mathbf{1}_{x_i \neq 0}$ | $\begin{cases} z_i, & i \text{ one of the } \tau \text{ largest indices} \\ 0 & \text{otherwise.} \end{cases}$ | Analytic |



(a) True
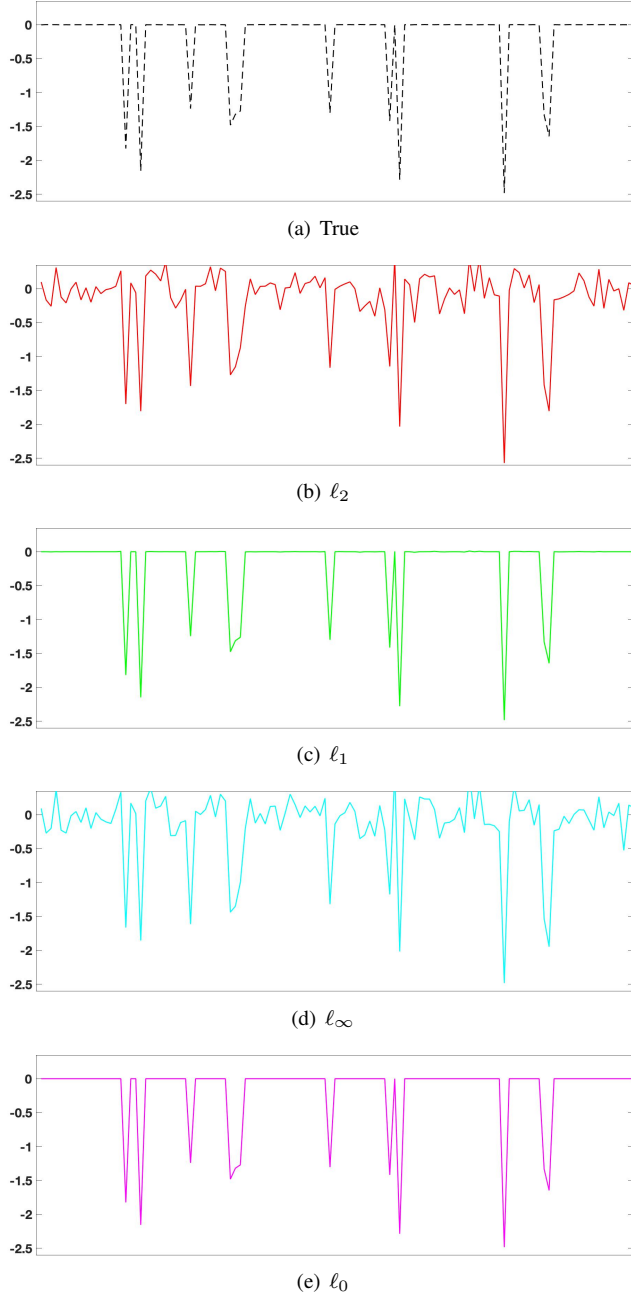
(b) $\ell_2$

(c) $\ell_1$

(d) $\ell_\infty$

(e) $\ell_0$

Fig. 2. Residuals for different $\ell_p$-norms after algorithm termination. Note how the $\ell_1$- and $\ell_0$-norms can capture the outliers only.



(a) True

(b) $\ell_2$

(c) $\ell_1$

(d) $\ell_\infty$

(e) $\ell_0$

Fig. 3. Basis Pursuit De-noising results for a randomly generated linear model with large, sparse noise.

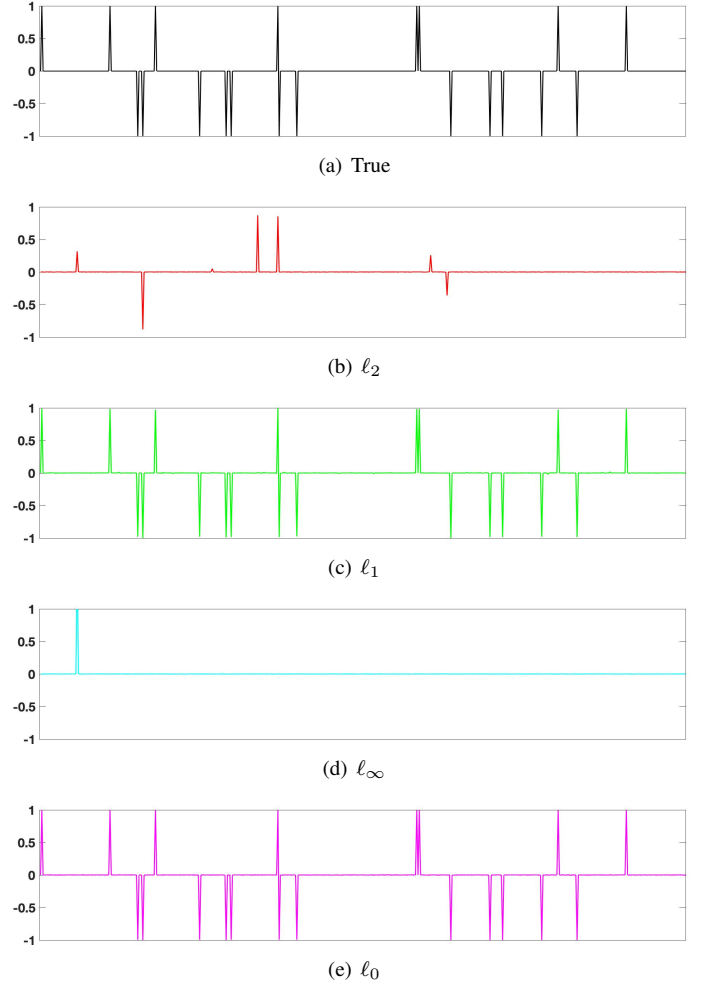sparsfying transform domain. The first example considers the same model as in (9) where we want to enforce sparsity on $x$ while constraining the data misfit. The variable $x$ is a vector of length $n$ that has values $\{-1, 1\}$ on a random $4\%$ of its entries and zeros everywhere else; represents a spike train that we observe using a linear operator, $A \in \mathbb{R}^{n,m}$. $A$ was generated with independent standard Gaussian entries, and $b \in \mathbb{R}^m$ is observed data with large, sparse noise. We take $m = 120$ and $n = 512$. The noise is generated by placing large values on $10\%$ of the observations and assuming everything else was observed cleanly (ie no noise). Here, we test the efficacy of using different $\ell_p$ norms on the residual constraint. With the addition of large, sparse noise to the data, smooth norms on

TABLE III
CURVELET INTERPOLATION AND DENOISING RESULTS FOR SPGL1 AND
ALGORITHM 4 FOR SELECTED $\ell_p$-NORMS FOR BPDN.

| 4D Monochromatic Interpolation | | | |
|---|---|---|---|
| Method/Norm | SNR | SNR $w_1$ | Time (s) |
| $\ell_2$ with SPGL1 | 1.4594 | - | 52.5 (early stoppage) |
| $l_2$ with Alg.4 | 0.1420 | 0.0851 | 1348 |
| $l_1$ with Alg.4 | 13.0193 | 12.5768 | 1335 |
| $l_\infty$ with Alg.4 | 0.0000 | 0 | 776 |
| $l_0$ with Alg.4 | 13.9019 | 13.4294 | 1120 |



(a) True Data
(b) Added Noise (binary)

(c) Noisy Data with Missing Sources
(d) SPGL1
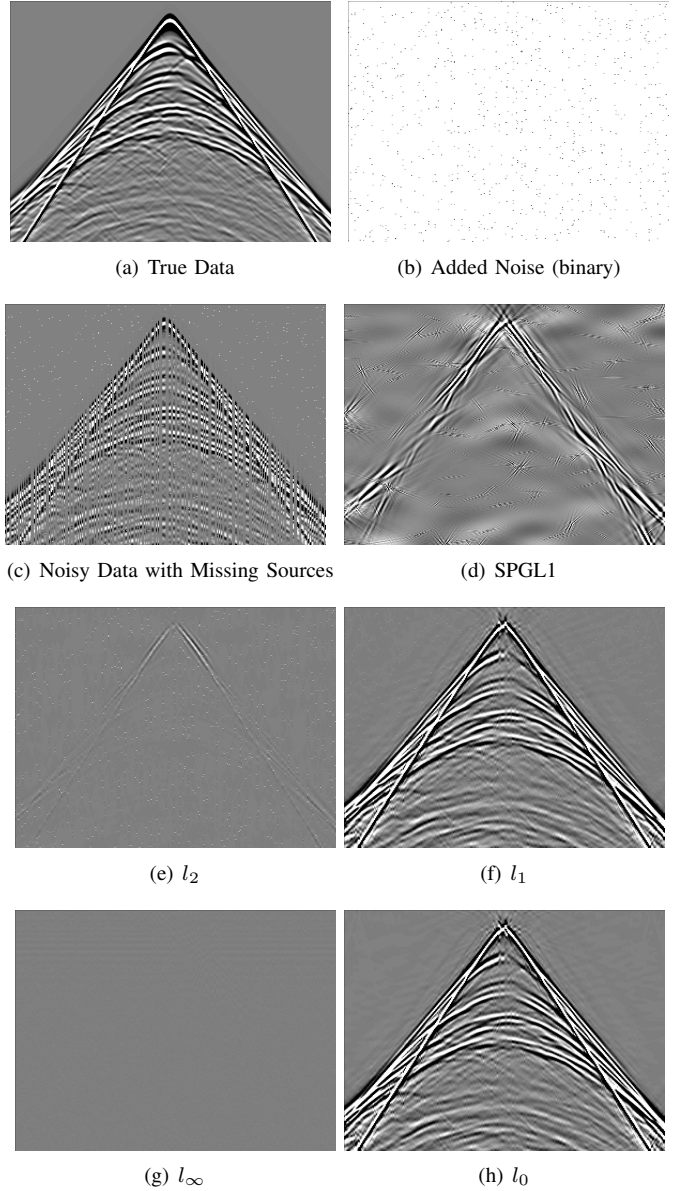
(e) $l_2$
(f) $l_1$

(g) $l_\infty$
(h) $l_0$

Fig. 4. Interpolation and de-noising results for BPDN in the curvelet domain. Observe the complete inaccuracy of smooth norms with large, sparse noise.

the residual constraint should not be able to effectively deal with such outlier residuals. With our adaptable formulation, it should be easy to enforce both sparsity in the $x$ domain as well as the residuals. Other formulations, such as SPGL1, do not have this capability.

This noise is depicted in as the bottom black dashed line in Figure 2. The results are shown in Figure 3 and in Table I. From these, we can clearly see that the $\ell_2$ norm is not effective for sparse noise, even at the correct error budget $\sigma$. Our approach is resilient to different types of noise since we can easily change the residual ball projection. This is seen by the almost exact accuracy of the $\ell_1$ and $\ell_0$ norms, with SNR's of 33 and 45 respectively.

The next test of the BPDN formulation is for a common source gather where entries are both omitted and corrupted with synthetic noise. Here, the objective function looks for sparsity in the curvelet domain, while the residual constraint seeks to match observed data within a certain tolerance $\sigma$. First, we note that doing interpolation only without added noise yields an SNR of approximately 13 for all formulations and algorithms; that is, all $\ell_p$ norms for Algorithm 4 and SPGL1. Here, we again want to enforce sparsity both in the curvelet domain ($x$) and the data residual ($\|\mathcal{A}(x) - b\|$), which SPGL1 and other algorithms lack the capacity to do.

Following the first experiment, we add large sparse noise to a handful of data points; in this case, we added large values to a random 1% of observations (this does not include omitted entries). The noise added is approximately 120, while the observed data can range from 0 to 30. The interpolated and denoising results are shown in Figure 4 and Table III. Large, sparse noise cannot be filtered effectively by a smooth norm constraint, using either Algorithm 4 or SPGL1. However, $\ell_1$ and $\ell_0$ norms effectively handle such noise, and can be optimized using our approach. The SNR's for these implementations are approximately 10 and 11 respectively, approaching that of the noiseless data mentioned above.

## V. EXTENSION TO LOW-RANK MODELS

Treating the data as having a matrix structure gives additional regularization tools — in particular low-rank structure in particular domains. The BPDN formulation for residual-constrained low-rank interpolation is given by

$$\min_{X} \|X\|_* \quad \text{s.t.} \quad \rho\left(\mathcal{A}(X) - b\right) \leq \sigma \qquad (11)$$

for $X \in \mathbb{C}^{m \times n}$, $\mathcal{A} : \mathbb{C}^{n \times m} \to \mathbb{C}^p$ is a linear masking operator from full to observed (noisy) data $b$, and $\sigma$ is the

misfit tolerance. The nuclear norm $\|X\|_*$ is the $\ell_1$ norm of the singular values of $X$. Solving the problem (11) requires using a decision variable that is the size of the data, as well as updates to this variable that require SVDs at each iteration. It is much more efficient to model $X$ is a product of two matrices $L$ and $R$, given by

$$\min_{L,R} \frac{1}{2}(\|L\|_F^2 + \|R\|_F^2) \quad \text{s.t.} \quad \rho\left(\mathcal{A}(LR^T) - b\right) \leq \sigma \qquad (12)$$

where $L \in \mathbb{C}^{n \times k}$, $R \in \mathbb{C}^{m \times k}$, and $LR^T$ is the low-rank representation of the data. The solution is guaranteed to be at most rank $k$, and in addition, the regularizer $\frac{1}{2}(\|L\|_F^2 + \|R\|_F^2)$ is an upper bound for $\|LR^T\|_*$, the sum of singular values of $LR^T$, further penalizing rank by proxy. The decision variables then have combined dimension $k(m \times n)$, which is much smaller than the $nm$ variables required by convex formulations. When

$\rho$ is smooth, the problems are solved using a continuation that interchanges the roles of the objective and constraints, solving a sequence of problems where $\rho\left(\mathcal{A}(LR^T) - b\right)$ is minimized over the $\ell_2$ ball [3] using projected gradient; an approach we call SPGLR below.

When $\rho$ is not smooth, SPGLR does not work and there are no available implementations for (12). Nonsmooth $\rho$ arise when we want the residual to be in the $\ell_1$ norm ball, so we are robust to outliers in the data, and can exactly fit inliers.

We now extend Algorithm 3 to this case. For any $\rho$ (smooth or nonsmooth), we introduce a latent variable $W$ for the data matrix, and solve

$$\min_{L,R,W} \left\| \begin{matrix} L \\ R \end{matrix} \right\|_F^2 + \frac{1}{2\eta}\|W - LR^T\|_2^2, \quad \text{s.t. } \|\mathcal{A}(W) - b\|_p \leq \sigma \tag{13}$$

with $\eta$ a parameter that controls the degree of relaxation; as $\eta \downarrow 0$ we have $W \to LR^T$. The relaxation allows a simple block-coordinate descent detailed in Algorithm 4.

---

**Algorithm 4** Block-Coordinate Descent for (13).

1: **Input:** $w_0, L_0, R_0$
2: Initialize: $k = 0$
3: **while** not converged **do**
4: $\quad L_{k+1} \leftarrow \left(I + \eta R_k^T R_k\right)^{-1}\left(\eta W_k R_k\right)$
5: $\quad R_{k+1} \leftarrow \left(\eta W_k^T L_{k+1}\right)\left(I + \eta L_{k+1}^T L_{k+1}\right)^{-1}$
6: $\quad W_{k+1} \leftarrow \begin{cases} (L_{k+1}R_{k+1}^T)_{ij}, & (i,j) \in X_{obs} \\ \text{proj}_{\mathbb{B}_{\rho,\sigma}}\left(\mathcal{A}(L_{k+1}R_{k+1}^T) - b\right), & \text{o.w.} \end{cases}$
7: $\quad k \leftarrow k + 1$
8: **end while**
9: **Output:** $w_k, L_k, R_k$

---

Algorithm 4 is also simple to implement. It requires two least squares solves (for $L$ and $R$), which are inherently parallelizable. It also requires a projection of the updated data matrix estimates $LR^T$ onto the $\sigma$-level set of the misfit penalty $\rho$. This step is detailed below.

For unobserved data $(i,j) \notin X_{obs}$, we have $W_{ij} = (LR^T)_{ij}$. For observed data, let $v$ denote $\mathcal{A}(LR^T)$. Then the $W$ update step is given by solving

$$\min_w \|w - v\|_2^2, \quad \text{s.t.} \|w - b\|_p \leq \sigma.$$

Using the simple substitution $z = w - b$, the we get

$$\min_z \|z - (v - b)\|_2^2, \quad \text{s.t.} \quad \|z\|_p \leq \sigma$$

which is precisely the projection of $\mathcal{A}(LR^T) - b$ onto $\mathbb{B}_{p,\sigma}$, the $\sigma$-level set of $\rho$. We use the same projectors for $\rho \in \{l_0, l_1, l_2, l_\infty\}$ as in Section IV, see Table II. The convergence criteria for Algorithm 4 is based on the optimality of the quadratic subproblems in $L, R$ and feasibility measure of $W - LR^T$, though in practice we compare performance of algorithms based on a computational budget. This block-coordinate descent scheme converges to a stationary point of Equation 13 by [21, Theorem 4.1].

Implementing block-coordinate descent on these forms until convergence produces the completed low-rank matrix. Setting $\nu = \|LR^T - w\|_2^2$, we iterate until $\nu < 1e - 5$ or a maximum number of iterations is reached. In the next section, we develop an application of this method to seismic interpolation and denoising.

## VI. 4D MATRIX COMPLETION WITH DE-NOISING

There are two main requirements when using the rank-minimization based framework for seismic data interpolation and denoising: *(i)* underlying seismic data should exhibit low-rank structure (singular values should decay fast) in some transform domain, and, *(ii)* subsampling and noise destroy the low-rank structure (singular values decay slow) in that domain. For exploiting the low-rank structure during interpolation and denoising, we follow the matricization strategy proposed by [8]. The matricization (source-x, source-y), i.e., placing both the source coordinates along the columns (Figure 6(a)), gives slow-decay of singular values (Figure 5(a)), while the matricization (source-x, receiver-x) (Figure 6(c)) gives fast decay of the singular values (Figure 5(b)). To understand the effect of subsampling on the low-rank structure, we remove the 50% of the sources. Subsampling destroys the fast singular value decay in the (source-x, receiver-x) matricization, but not in the (source-x, receiver-y) matricization. This is because missing sources are missing columns in the (source-x, source-y) matricization, and missing sub-blocks in the (source-x, receiver-x) matricization (Figure 6(b)). The latter is more effective for low-rank interpolation.

Similar to the BPDN experiments, we want to show that nonsmooth constraints on the data residual can be effective for dealing with large, sparse noise. The smooth $\ell_2$ norm that is most common in BPDN problem will fail in such examples, thereby leading to better data estimation with the implementation of non-smooth norms on the residuals. Thus, the goal of the below experiments is to show that enforcing sparsity in the singular values (ie low-rank) and sparsity in the residual constraint can be more effective with large, sparse noise than smooth residual constraints solved by most contemporary algorithms.

### A. Experiment Description

This example demonstrates the efficacy of the proposed approach using data created by a 5D dataset based on a complex SEG/EAGE overthrust model simulation [1]. The dimension of the model is $5\,\text{km} \times 10\,\text{km} \times 10\,\text{km}$ and is discretized on a $25\,\text{m} \times 25\,\text{m} \times 25\,\text{m}$ grid. The simulated data contains $201 \times 201$ receivers sampled at 50 m and $101 \times 101$ sources sampled at 100 m. We apply the Fourier transform along the time domain and extract a frequency slice at 10 Hz as shown in Figure 7(a), which is a 4D object (source-x, source-y, receiver-x and receiver-y). We eliminate 80% of the sources and add large sparse outliers from the random gaussian distribution $\mathcal{N}(0, a_i \max(X_{s_i}))$ (mean zero and variance on the order of the largest value in that particular source). The 10 generated values with the highest magnitudes are kept, and these are randomly added to observations in the remaining sources (Figure 7(f)). The largest value of our dataset is approximately 40, while the smallest is close to zero. Thus, we are essentially increasing/decreasing 1% of the

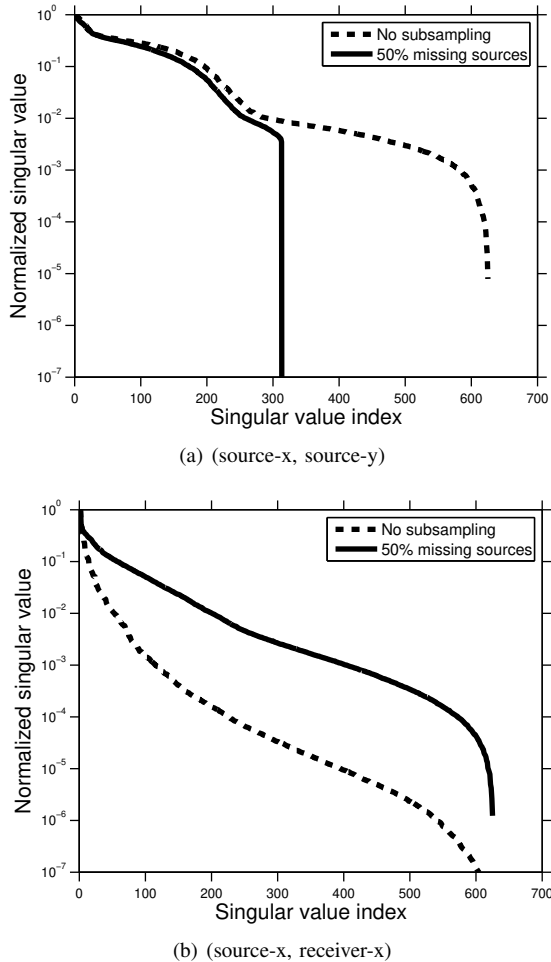(a) (source-x, source-y)



(b) (source-x, receiver-x)

Fig. 5. Normalized Singular value decay for full data and 50% missing sources with two different matricizations. (Source: [13]).

entries by several orders of magnitude, which contaminates the data significantly, especially if the original entry was nearly 0. For all low-rank completion and denoising, we let $a_i = 10^{-1}$. The objective is to recover missing sources and eliminate noise from observed data. We use a rank of $k = 75$ for the formulation (that is, $L \in \mathbb{C}^{n \times 75}$ and similarly for $R$), and run all algorithms for 150 iterations, using a fixed computational budget. We perform three experiments on the same dataset: 1) De-noising only (Figure 7(c)); 2) Interpolation only (Figure 7(d)); and 3) Combined Interpolation and De-noising (Figure 7(f)). Since we have ground truth, we pick $\sigma$ to be the exact difference between generated noisy data and the true data; $\sigma$ for the $l_0$ norm is a cardinality measure, so it is set to number of noisy points added.

## B. Results

Tables IV-VI display SNR values for different algorithms and formulations for the three types of experiments, and Figures 8-10 display the results for a randomly selected number of sources for the three experiments. Even a small number of outliers can greatly impact the quality of the low-rank de-noising and interpolation for the standard, smoothly residual-constrained algorithms. The de-noising only results (Figure 8,



(a) Full (src-x, src-y)



(b) Subsampled (src-x, src-y)



(c) Full (src-x, rec-x)
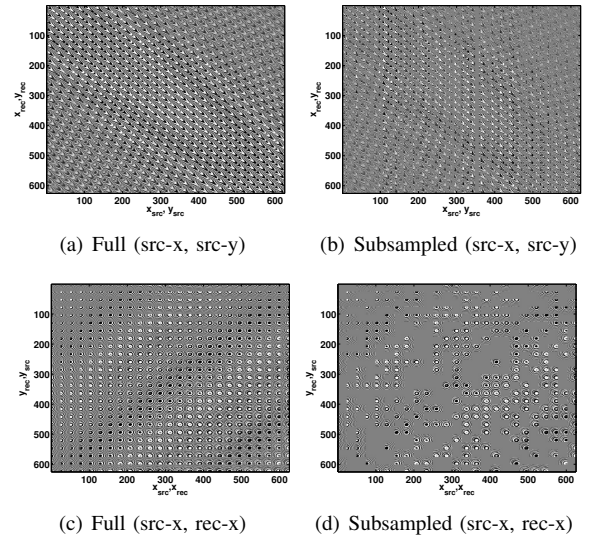


(d) Subsampled (src-x, rec-x)

Fig. 6. Full and subsampled matricizations used in low-rank completion (Source: [14]).

TABLE IV
4D DE-NOISING RESULTS FOR SPGLR AND ALGORITHM 4 FOR SELECTED $\ell_p$ NORMS.

| 4D Monochromatic De-noising | | | |
|---|---|---|---|
| Method/Norm | SNR | SNR-W | Time (s) |
| $\ell_2$ with SPGLR | 11.7489 | - | 16530 |
| $l_2$ with Alg.4 | 11.7463 | -2.3338 | 9430 |
| $l_1$ with Alg.4 | 11.7638 | -2.3063 | 11546 |
| $l_\infty$ with Alg.4 | 11.7456 | -2.3338 | 12108 |
| $l_0$ with Alg.4 | 17.9595 | 48.8607 | 11569 |

Table IV) show that all methods perform well when all sources are available. The interpolation only results (Figure 9, Table V) show that all constraints perform well in interpolating the missing data. This makes sense, as all algorithms will simply favor the low-rank nature of the data. However, the combined de-noising and interpolation dataset shows that the $\ell_0$ norm approach does far better than any smooth norm in comparable time. Table VI shows that when data for similar sources is absent/not observed, the smoothly-constrained formulations fail completely. When noise is added to the low-amplitude section of the observed data, the smoothly-constrained norms fail drastically, while the $\ell_0$ norm can effectively remove the errors. This is starkly evident in Figures 10(a)-10(e), where all except Figure 10(e) are essentially noise; the result is supported by the SNR values in Table VI. While Figures 10(a)-10(e) can mostly capture the structure of the data where there were nonzero values (ie where the seismic wave is observed in the upper left corner of each source), only the $\ell_0$ norm can capture the areas of lower energy data.

## VII. CONCLUSIONS

We proposed a new approach for level-set formulations, including basis pursuit denoise and residual-constrained low-rank formulations. The approach is easily adapted to a variety of nonsmooth and nonconvex data constraints. The resulting problems are solved using Algorithm 2 and 4; which require

TABLE V
4D INTERPOLATION RESULTS FOR SPGLR AND ALGORITHM 4 FOR
SELECTED $\ell_p$ NORMS.

| 4D Monochromatic Interpolation | | | |
|---|---|---|---|
| Method/Norm | SNR | SNR-W | Time (s) |
| $\ell_2$ with SPGLR | 16.3976 | - | 5817 |
| $l_2$ with Alg.4 | 16.0629 | 16.5424 | 7526 |
| $l_1$ with Alg.4 | 16.0692 | 16.5491 | 7996 |
| $l_\infty$ with Alg.4 | 16.0627 | 16.5423 | 8119 |
| $l_0$ with Alg.4 | 16.0096 | 16.4728 | 6848 |

TABLE VI
4D COMBINED DE-NOISING AND INTERPOLATION RESULTS FOR SPGLR
AND ALGORITHM 4 FOR SELECTED $\ell_p$ NORMS.

| 4D Monochromatic De-noising & Interpolation | | | |
|---|---|---|---|
| Method/Norm | SNR | SNR-W | Time (s) |
| $\ell_2$ with SPGLR | -3.2906 | - | 8712 |
| $l_2$ with Alg.4 | 0.9185 | -0.3321 | 6802 |
| $l_1$ with Alg.4 | 0.9193 | -0.3235 | 8068 |
| $l_\infty$ with Alg.4 | 0.9185 | -0.3321 | 8117 |
| $l_0$ with Alg.4 | 16.0655 | 16.5445 | 6893 |

only that the penalty $\rho$ has an efficient projector. The algorithms are simple, scalable, and efficient. Sparse curvelet denoising and low-rank interpolation of a monochromatic slice from the 4D seismic data volumes demonstrate the potential of the approach.
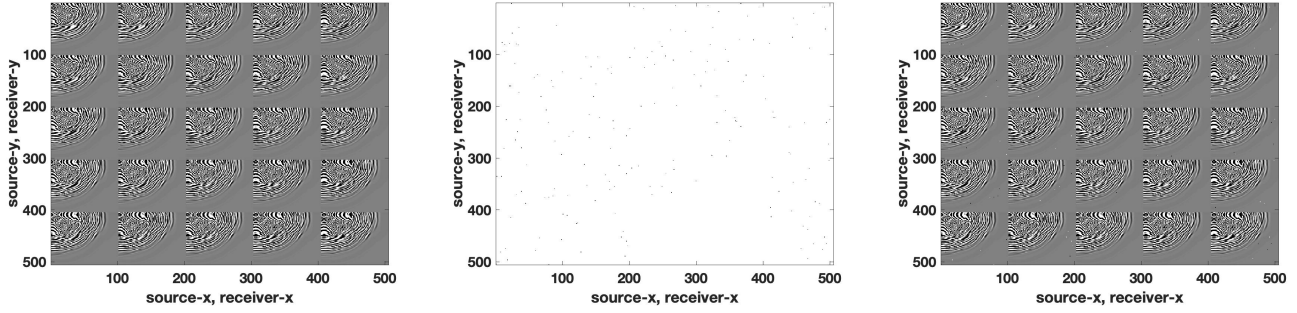
A particular quality of the seismic denoising and interpolation problem is that the amplitudes of the signal have significant spatial variation. The error in the data is a much larger problem for low-amplitude data. This quality makes it very difficult to obtain reasonable results using Gaussian misfits and constraints. Nonsmooth exact formulations (including $\ell_1$ and particularly $\ell_0$) appear to be extremely well-suited for this magnified heteroscedastic issue.
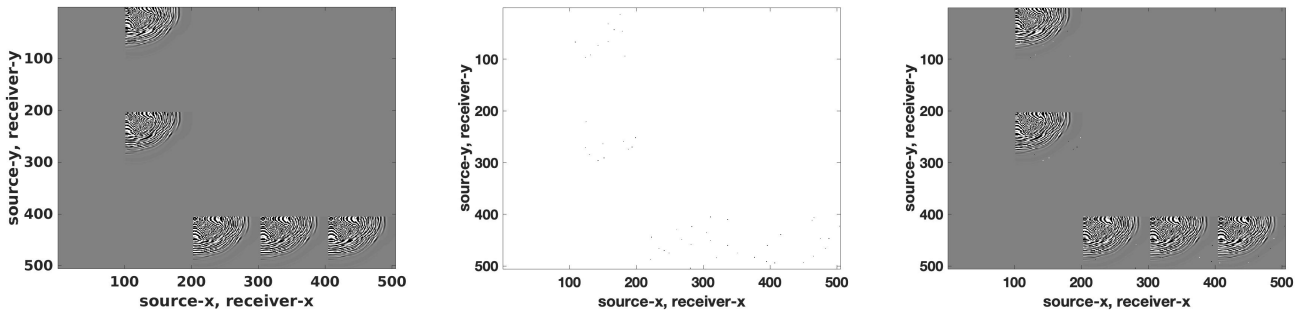
## VIII. ACKNOWLEDGEMENTS

## REFERENCES

[1] F. Aminzadeh, N. Burkhard, L. Nicoletis, F. Rocca, and K. Wyatt. Seg/eaeg 3-d modeling project: 2nd update. *The Leading Edge*, 13(9):949–952, 1994.

[2] A. Y. Aravkin, J. V. Burke, D. Drusvyatskiy, M. P. Friedlander, and S. Roy. Level-set methods for convex optimization. *To appear in Mathematical Programming, Series B.*, 2018.

[3] A. Y. Aravkin, R. Kumar, H. E. Mansour, B. Recht, and F. J. Herrmann. Fast methods for denoising matrix completion formulations, with applications to robust seismic data interpolation. *SIAM J. Scientific Computing*, 36, 2014.

[4] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

[5] B. M. Bell and J. V. Burke. Algorithmic differentiation of implicit functions and optimal values. In *Advances in Automatic Differentiation*, pages 67–77. Springer, 2008.

[6] E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: universal encoding strategies. *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[8] C. Da Silva and F. J. Herrmann. Optimization on the hierarchical tucker manifold–applications to tensor completion. *Linear Algebra and its Applications*, 481:131–173, 2015.

[9] D. Davis and W. Yin. Convergence rate analysis of several splitting schemes. In *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 115–163. Springer, 2016.

[10] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–99, 2004.

[11] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Comp.*, 10(6):1455–1480, 1998.

[12] F. J. Herrmann and G. Hennenfent. Non-parametric seismic data recovery with curvelet frames. *Geophysical Journal International*, 173(1):233–248, 2008.

[13] A. Kadu and R. Kumar. Decentralized full-waveform inversion. Submitted to EAGE on January 15, 2018, 2018.

[14] R. Kumar, C. Da Silva, O. Akalin, A. Y. Aravkin, H. Mansour, B. Recht, and F. J. Herrmann. Efficient matrix completion for seismic data reconstruction. *Geophysics*, 80(5):V97–V114, 2015.

[15] R. Kumar, O. López, D. Davis, A. Y. Aravkin, and F. J. Herrmann. Beating level-set methods for 5-d seismic data interpolation: A primal-dual alternating approach. *IEEE Transactions on Computational Imaging*, 3(2):264–274, June 2017.

[16] M. Lustig, D. Donoho, and J. Pauly. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magnetic Resonance in Medicine*, 58:1182–95, 2007.

[17] L. Oneto, S. Ridella, and D. Anguita. Tikhonov, Ivanov and Morozov regularization for support vector machine learning. *Machine Learning*, 103(1):103–136, 2016.

[18] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52(3):471–501, Aug. 2010.

[19] M. D. Sacchi, T. J. Ulrych, and C. J. Walker. Interpolation and extrapolation using a high-resolution discrete fourier transform. *IEEE Transactions on Signal Processing*, 46(1):31–38, Jan 1998.

[20] J. A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1050, March 2006.

[21] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[22] E. Van Den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

[23] E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.*, 31(2):890–912, Nov. 2008.

[24] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher. Sketchy Decisions: Convex Low-Rank Matrix Optimization with Optimal Storage. *ArXiv e-prints*, Feb. 2017.

[25] P. Zheng and A. Aravkin. Fast methods for nonsmooth nonconvex minimization. *ArXiv e-prints*, Feb. 2018.

[26] P. Zheng, T. Askham, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin. A Unified Framework for Sparse Relaxed Regularized Regression: SR3. *ArXiv e-prints*, July 2018.

(a) Fully sampled monochromatic slize at 10 Hz.

(b) Noisy data alone (binary). Sparse noise was added by keeping the top 10 entries generated from a normal distribution with mean zero and variance $0.1 \max(X_{s_i})$

(c) Observed noisy data.

(d) Subsampled noiseless data. We omitted 80% of sources.

(e) Subsampled and noise, with noise only present (binary).

(f) Subsampled and noisy data. We again omitted 80% of sources and added the noise described above to the rest of the sources.

Fig. 7. True data and three different experiments for testing our completeness algorithm.



(a) SPGLR
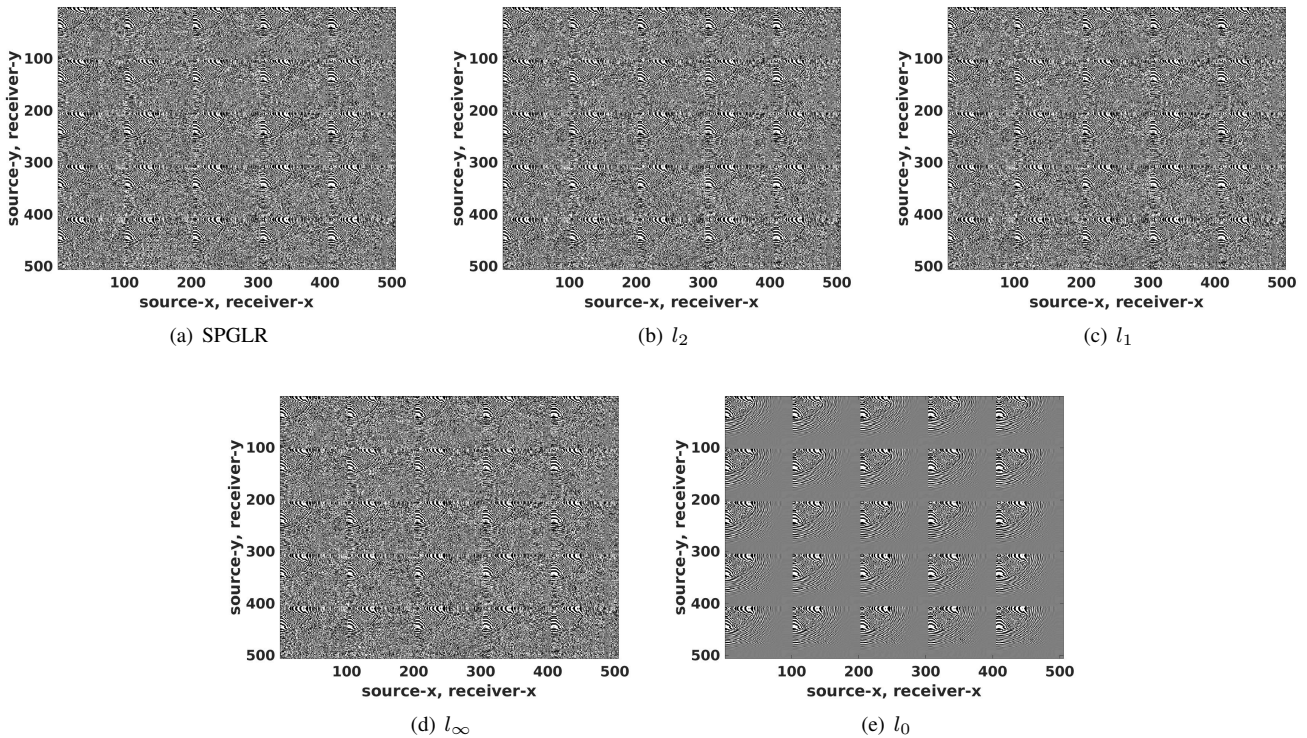
(b) $l_2$

(c) $l_1$

(d) $l_\infty$

(e) $l_0$

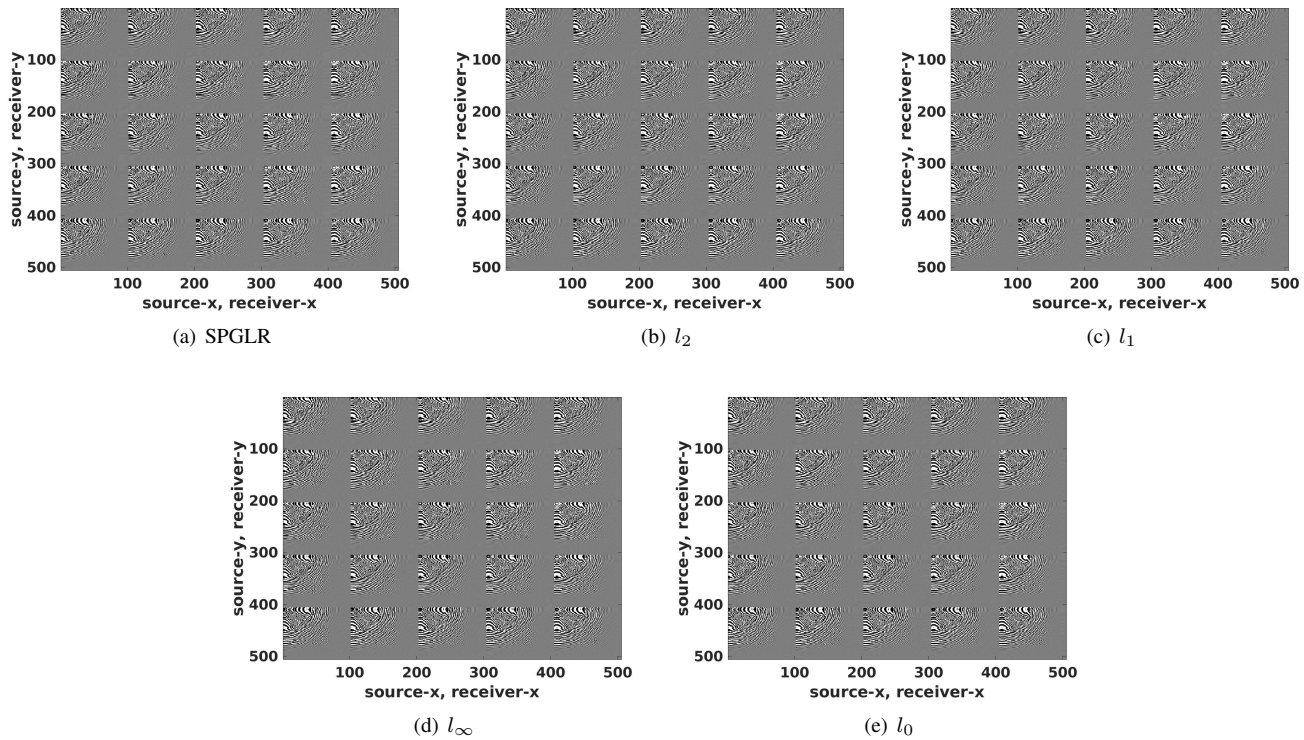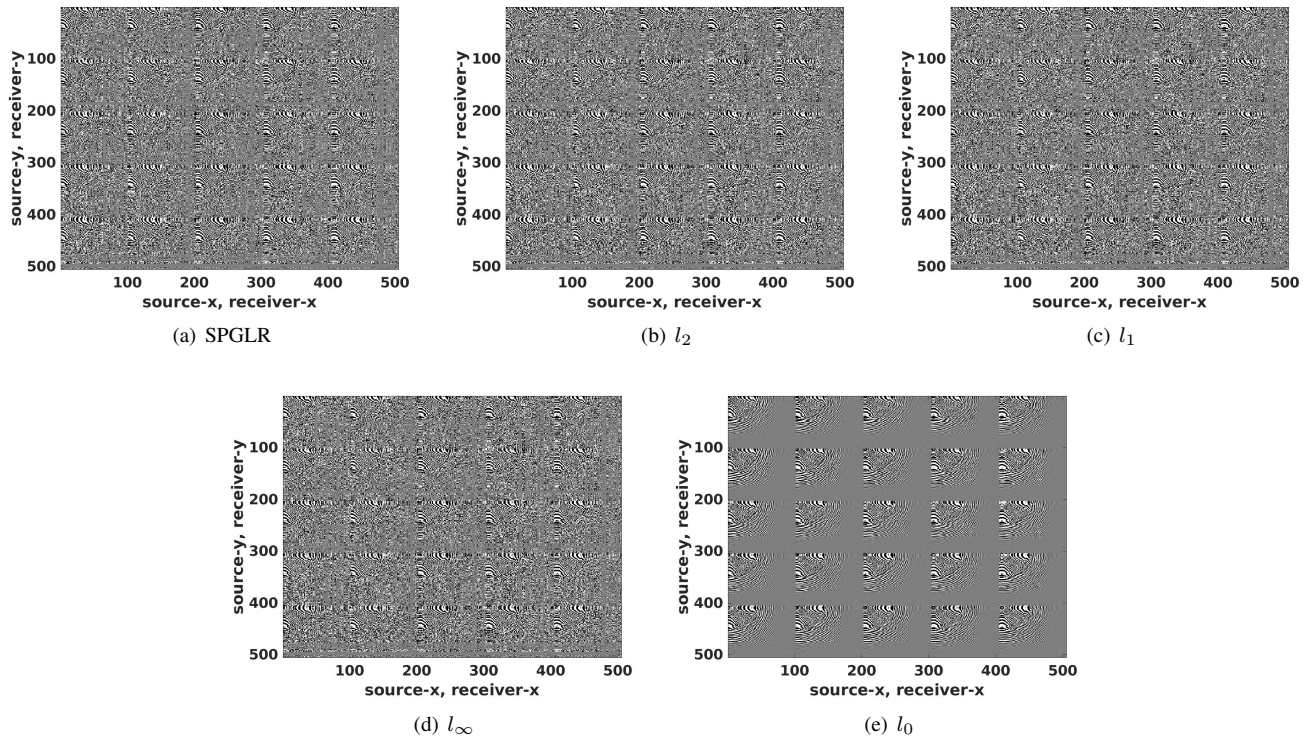Fig. 8. Denoising-only results.

Fig. 9. Interpolation-only results.



Fig. 10. Interpolation and Denoising results.