Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

Reducing Effects of Bad Data Using Variance Based Joint Sparsity Recovery

Anne Gelb · Theresa Scarnati

Received: date / Accepted: date

Abstract Much research has recently been devoted to jointly sparse (JS) signal recovery from multiple measurement vectors (MMV) using $\ell_{2,1}$ regularization, which is often more effective than performing separate recoveries using standard sparse recovery techniques. However, JS methods are difficult to parallelize due to their inherent coupling. The *variance* based joint sparsity (VBJS) algorithm was recently introduced in [1]. VBJS is based on the observation that the pixel-wise variance across signals convey information about their shared support, motivating the use of a *weighted* ℓ_1 JS algorithm, where the weights depend on the information learned from calculated variance. Specifically, the ℓ_1 minimization should be more heavily penalized in regions where the corresponding variance is small, since it is likely there is no signal there. This paper expands on the original method, notably by introducing weights that ensure accurate, robust, and cost efficient recovery using both ℓ_1 and ℓ_2 regularization. Moreover, this paper shows that the VBJS method can be applied in situations where some of the measurement vectors may misrepresent the unknown signals or images of interest, which is illustrated in several numerical examples.

Keywords multiple measurement vectors · joint sparsity · image reconstruction · false data injections;

Mathematics Subject Classification (2000) 65F22 · 65K10 · 68U10

1 Introduction

Recovering sparse signals and piecewise smooth functions from under-sampled and noisy data has been a heavily investigated topic over the past decade. Typical algorithms minimize the ℓ_1 norm of an approximation of a sparse feature (e.g. wavelets, gradients, or edges) of the solution so that the reconstructed solution will preserve sparsity in its corresponding sparse domain. A weighted ℓ_1 reconstruction algorithm was introduced in [8] to reconcile the difference between the "true"

Anne Gelb's work is supported in part by the grants NSF-DMS 1502640, NSF-DMS 1732434, and AFOSR FA9550-15-1-0152. Approved for public release. PA Approval #:[88AWB-2017-6162].

Anne Gelb
Department of Mathematics, Dartmouth College, USA.
E-mail: annegelb@math.dartmouth.edu

Theresa Scarnati
School of Mathematical and Statistical Sciences, Arizona State University, USA,
Air Force Research Laboratory, Wright-Patterson AFB, USA.
E-mail: theresa.scarnati.1@us.af.mil

sparsity ℓ_0 norm and the surrogate ℓ_1. Sparse signal recovery was accomplished by a solving a sequence of weighted ℓ_1 minimization problems, with the weights iteratively updated at each step. As was demonstrated there, updating the weights yielded successively improved estimations of the non-zero coefficient locations, and consequently relaxes standard sampling rate requirements for sparse signal recovery. An adjustment for the weight calculation was proposed in [10] resulting in an improvement to the iterative reweighting algorithm. An adaptively weighted total variation (TV) regularization algorithm, where the spatially adaptive weights were based on the difference of values between neighboring pixels, was introduced in [25]. A different weighting technique was developed in [9] to reduce the staircase effect of TV regularization. An adaptive function was used along with new parameters to balance the trade off between penalizing discontinuities and recovering sharp edges. While the method accomplishes the goal of allowing smooth transitions without reducing sharp edges, the mathematical formulation is challenging and uniqueness is not guaranteed. Further weighted ℓ_1 literature can be found at [8–10, 25, 39, 40] and references therein.

In many inverse problems, it may be possible to acquire multiple measurement vectors (MMVs) of the unknown signal or image, [2, 11, 12, 16–18, 24, 30]. MMV collection is especially useful when trying to recover solutions of an underdetermined system when the MMVs have the same, but unknown, sparsity structure. Techniques exploiting this type of commonality, referred to as *joint sparsity* (JS) methods, can be developed by extending the commonly used single measurement vector (SMV) algorithms for sparse solutions, [12]. Additional examples of this can be found in [11, 21, 33, 35, 37] and references therein. In particular, jointly sparse vectors are often recovered using the popular $\ell_{2,1}$ minimization, [11, 14, 31, 42, 44], which was thoroughly analyzed in [16, 17]. Conditions for guaranteeing improvements over SMV were determined for a class of MMV techniques in [16] and moreover, it was shown in [17] that under mild conditions the probability of *not* recovering a sparse vector with high probability (based on a chosen threshold) using $\ell_{2,1}$ regularization decays exponentially with the increase of measurements. Various algorithms are used to implement $\ell_{2,1}$ regularization, including the alternating direction method of multipliers (ADMM), split Bregman, joint-OMP, and "reduce-and-boost", [26, 34, 42]. An algorithm is typically chosen to yield the most efficiency for the particular problem at hand (for example, based on problem complexity). In this paper we use ADMM, and note that while other methods may yield faster convergence for our chosen examples, in general $\ell_{2,1}$ regularization techniques are inherently coupled, making them difficult to parallelize.

While much work has been done on designing weighted ℓ_p (specifically ℓ_1) reconstruction methods for SMV, and in constructing joint sparsity MMV methods using the $\ell_{2,1}$ norm, there has been less work devoted to improving MMV through weighted ℓ_p minimization. Three notable investigations include: (i) [31], where the SMV weights were adapted from those in [8] to $\ell_{2,1}$ minimization for the problem of multi-channel electrocardiogram signal recovery. Although the technique enhances the sparseness of the solution and reduce the number of measurements required for accurate recovery, it requires hand tuning of parameters. (ii) [44], where a weighted $\ell_{2,1}$ minimization algorithm is used for direction of arrival estimation, high resolution radar imaging and other sparse recovery related problems using random measurement matrices. The singular value decomposition is used to exploit the relationship between the signal subspace and the noise subspace for designing the weights. (iii) [18], where a shape-adaptive jointly sparse classification method for hyperspectral imaging was developed. We note that all of these developments were problem specific, and not easily adapted for general sparse signal recovery.

In this investigation we propose using the *variance* based joint sparsity (VBJS) method for MMV, introduced in [1]. The VBJS technique exploits the idea that the variance across multi-measurement vectors that are jointly sparse should be sparse in the sparsity domain of the underlying signal or image, an idea first proposed in [15] for the purpose of edge detection and localization. The *weights* used for the weighted ℓ_1 regularization term are essentially reciprocals of this variance (with a threshold built in to ensure no division by zero), with the idea being that the ℓ_1 term should be heavily penalized when the variance is small, but should not influence the solution as much when the variance is large. Presumably, the large variance indicates support of the image or signal in the sparse domain. One of the main advantages of VBJS is that it is easily parallelized. In particular, it was shown in [1] that VBJS is consistently more computationally efficient than $\ell_{2,1}$ regularization algorithms when using standard black box solvers. In this investigation we improve on the VBJS algorithm by designing weights that reduce the parametric dependence on the reconstruction, making it more amenable to a variety of other applications not considered in [1]. Specifically, the VBJS can now be used in situations where some measurement vectors may misrepresent the unknown function of interest. In contrast, such "rogue" data may wield undue influence on the reconstruction of piecewise smooth solutions when using the standard $\ell_{2,1}$ approach. The original VBJS approach does not adequately account for false data in the weight design, so much more parameter tuning would be needed. False data problems appear in applications including state estimation of electrical power grids, [23], large scale sensor network estimation, [41], synthetic aperture radar (SAR) automated target recognition (ATR), [20], and many others, [43, 45]. False data may be purposefully injected into these systems to decrease the performance of automated detection algorithms. In other situations, misrepresentations of data occur due to human error or environmental issues effecting the measurements. For example, in SAR ATR it is often the case that targets are obscured by their surroundings (trees) or by enemies (meshes placed over the targets). Also, additional parts may be taken off or added to targets, corrupting measurement data, [20]. As part of our reconstruction algorithm, we include a numerically efficient comparative measurement of the measurement vectors, which allow us to appropriately disregard rogue data and improve our overall reconstruction.

Our proposed VBJS technique offers several advantages: (i) Our method is (essentially) non-parametric so that regularization parameters need not be hand tuned; (ii) We take advantage of the joint sparsity information available in the MMV setup, thus improving reconstruction accuracy while decreasing sampling rates, independent of application; (iii) With some sharpness reduction, our weights allow us to use the ℓ_2 norm, which is much more cost efficient; (iv) Our method mitigates the effects of rogue data. Finally, as noted above, the VBJS algorithm is easily parallelizable, so even when using the weighted ℓ_1 norm, it is much more efficient than when the $\ell_{2,1}$ norm is used.

The rest of the paper is organized as follows: In Section 2 we define joint sparsity for multi-measurement vectors and provide details for the standard $\ell_{2,1}$ regularization approach used to recover sparse signals. In Section 3 we describe the variance based joint sparsity (VBJS) approach, initially developed in [1], and demonstrate how weights should be constructed to reduce the impact of false information. We also propose a technique to choose the "best" solution from the set of possible solution vectors that can be recovered from the VBJS method, so that we do not have to compute each vector in the solution space. In Section 4 we prove that the alternating direction method of multipliers (ADMM) can be applied to the weighted ℓ_1 minimization. We also show how the VBJS method can be efficiently computed for the weighted ℓ_2 norm. Section 5 provides some numerical results for sparse signal recovery and one and two dimensional images. Some concluding remarks are given in Section 6.

2 Preliminaries

Consider a piecewise smooth function $f(x)$ on $[a, b]$. We seek to recover $\boldsymbol{f} \in \mathbb{R}^N$, where each element of $\boldsymbol{f}$ is given as $f_i = f(x_i), i = 1, ..., N$, with

$$x_i = a + \Delta x(i-1), \tag{2.1}$$

and $\Delta x = \frac{b-a}{N}$. We note that x_i are chosen to be uniform for simplicity of numerical experiments and is not required for our algorithm.

Since the underlying function f is piecewise smooth, it is sparse in its corresponding edge domain. Formally we have:

Definition 1 [11,13] A vector $\boldsymbol{p} \in \mathbb{R}^N$ is s-sparse for some $1 \leq s \leq N$ if

$$||\boldsymbol{p}||_0 = |\text{supp}(\boldsymbol{p})| \leq s.$$

In our case, $\boldsymbol{p}$ corresponds to the edge vector of f at the set of grid points in (2.1).

Suppose we acquire J data vectors, $\boldsymbol{y}^j \in \mathbb{C}^M$, as

$$\boldsymbol{y}^j = \boldsymbol{A}^j(\boldsymbol{f}) + \boldsymbol{\eta}^j, \quad j = 1, ..., J. \tag{2.2}$$

Here $\boldsymbol{A}^j : \mathbb{R}^N \to \mathbb{C}^M$ is a forward operator (often defined as a square ($N = M$), orthogonal matrix for simplicity) and

$$\boldsymbol{\eta}^j \in \mathbb{C}^M, \quad j = 1, ..., J, \tag{2.3}$$

model J Gaussian noise vectors.

Due to the sparsity in the edge domain, ℓ_1 regularization provides an effective means for reconstructing $\boldsymbol{f}$ given any of the J noisy data vectors. Specifically, we compute the unconstrained optimization problem

$$\check{\boldsymbol{f}}^j = \underset{\boldsymbol{g}}{\text{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{A}^j \boldsymbol{g} - \boldsymbol{y}^j||_2^2 + \mu ||\mathcal{L}\boldsymbol{g}||_1 \right\}, \quad j = 1, ..., J, \tag{2.4}$$

where μ is the ℓ_1 regularization parameter. In our experiments we often sample μ from a uniform distribution for all calculations of $\check{\boldsymbol{f}}^j$ to simulate the ad-hoc proceedure for selecting typical regularization parameters. The sparsifying operator, $\mathcal{L}$, is designed so that the chosen solution is sparse in the edge domain. In this investigation we choose $\mathcal{L}$ to be the mth order polynomial annihilation (PA), [3,4], and note that when $m = 1$ the method is equivalent to using total variation (TV).[1] To solve (2.4) we use the traditional alternating direction method of multipliers (ADMM) algorithm [19,22,36].

As shown in Figure 1(left), assuming that the model in (2.2) is correct, *any* of the reconstructed $\check{\boldsymbol{f}}^j$ (which we will refer to as the single measurement vector (SMV) reconstruction) should adequately approximate the underlying function f or any desired features of it. However, this may be impossible due to undersampling, noise, or bad information. Intuitively, using the redundant data from part of or all of the available data sets in (3.1) should lead to a better reconstruction algorithm. Indeed, many techniques have been developed to recover images from such multiple measurement vectors (MMV), [2,11,12,16–18,24,30]. In our case the underlying function f is sparse in the edge domain, and so the collected set of recovered vectors is *jointly sparse* in the edge domain. The formal definition of joint sparsity is given by

Definition 2 We say that

$$\boldsymbol{P} = \left[\boldsymbol{p}^1 \; \boldsymbol{p}^2 \; \cdots \; \boldsymbol{p}^J\right] \in \mathbb{R}^{N \times J}$$

is s-joint sparse if

$$||\boldsymbol{P}||_{2,0} = \left| \bigcup_{j=1}^{J} \text{supp}\left(\boldsymbol{p}^j\right) \right| \leq s,$$

where each $\boldsymbol{p}^j$ is s-sparse according to Definition 1.

[1] Although there are subtle differences in the derivations and normalizations, the PA transform can be thought of as higher order total variation (HOTV). Because part of our investigation discusses parameter selection, which depends explicitly on $||\mathcal{L}f||$, we will exclusively use the PA transform as it appears in [3] so as to avoid any confusion. Explicit formulations for the PA transform matrix can be found in [3]. We also note that the method can be easily adapted for other sparsifying transformations.

For the variance based joint sparsity method in Algorithm 1, we also will assume that

$$\mathrm{supp}(\boldsymbol{p}_1) \approx \mathrm{supp}(\boldsymbol{p}_2) \approx \cdots \approx \mathrm{supp}(\boldsymbol{p}_J), \tag{2.5}$$

that is, the joint sparsity of the vectors does not greatly exceed the sparsity of each individual vector.

To exploit the joint sparsity of the system, $\ell_{2,1}$ regularization is often applied, [11, 32, 42, 44]. Essentially, each vector is assumed to be sparse in its sparsity domain (e.g. edge domain), which motivates minimizing the ℓ_1 norm of each column. The "jointness" is accomplished by minimizing the ℓ_2 norm of each row (spatial elements). The general joint sparsity technique using $\ell_{2,1}$ regularization is [32]

$$\hat{\boldsymbol{f}} = \left\{ \underset{\boldsymbol{z}\in\mathbb{R}^{N\times J}}{\mathrm{argmin}} \quad ||\mathcal{L}\boldsymbol{z}||_{2,1} \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{z} = \boldsymbol{Y} \right\}, \tag{2.6}$$

where $\mathcal{L}$ is the sparsifying transform matrix (here the PA transform of order m), $\boldsymbol{Y} = [\boldsymbol{y}^1 \quad \boldsymbol{y}^2 \quad \cdots \quad \boldsymbol{y}^J] \in \mathbb{R}^{M\times J}$ and $\boldsymbol{A} = \boldsymbol{A}^1 = \cdots = \boldsymbol{A}^J$. The solution $\hat{\boldsymbol{f}} = [\hat{\boldsymbol{f}}^1 \quad \hat{\boldsymbol{f}}^2 \quad \cdots \quad \hat{\boldsymbol{f}}^J] \in \mathbb{R}^{N\times J}$ contains estimates for each measurement $\boldsymbol{y}^j$, $j = 1, ..., J$. It has been shown, both theoretically and in practice, that (2.6) yields improved approximations to *each* reconstruction in (2.4), [11, 31, 44].

Note that (2.6) is typically solved using optimization techniques such as the ADMM, focal underdetermined system solvers (FOCUSS) and matching pursuit algorithms, [12].[2] As demonstrated in Figure 1(middle), the joint sparsity approach using $\ell_{2,1}$ regularization is effective in cases where the data vectors are somewhat predictable, that is, when each measurement vector is determined from (2.2), and $\boldsymbol{A}^j$ is known. However, it is often the case when some of the acquired data do not have known sources. Worse, the information can be deliberately misleading, so that we assume we are acquiring $\boldsymbol{y}^j$ but in fact a completely different data set is obtained. We will refer to such a data set as a "rogue" vector. Figure 1(right) illustrates that in these situations, using (2.6) may be heavily influenced by the false measurements.[3] Hence we are motivated to find a technique that is able to discern "good" from "bad" information in the context of joint sparsity.

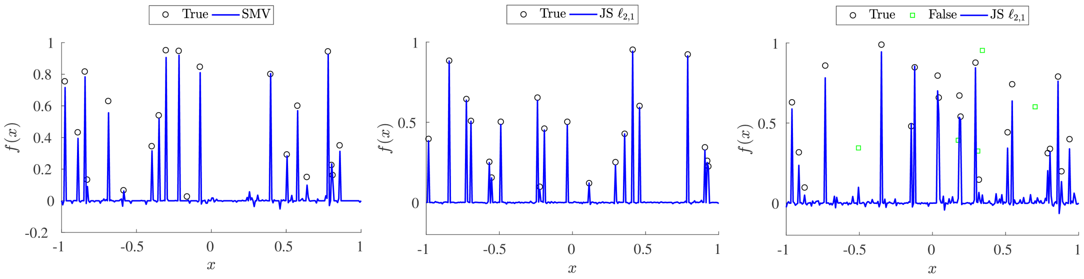

Fig. 1: Sparse vector of uniformly distributed values on $[0, 1]$ reconstructed using (left) ℓ_1 regularization with a single measurement vector (SMV), (middle) (2.6) applied on $J = 10$ true measurement vectors, and (right) (2.6) applied to $J = 10$ measurement vectors, with 5 containing false data. In each case $N = 256$, $M = 100$ and $||\boldsymbol{f}||_0 = 20$ with $\boldsymbol{A}$ having i.i.d. Gaussian entries and $\mu = .25$ in (2.4). Plotted here is the average of the final 10 joint sparsity (JS) $\ell_{2,1}$ reconstructions.

[2] We used the Matlab code provided in [14, 42] when implementing (2.6).

[3] For this simple example, each of the $K = 5$ false measurement vectors was formed by adding a single false data point, with height sampled from the corresponding distribution, (binary, uniform or Gaussian).

3 Variance based joint sparsity

Minimizing the effect of rogue measurement vectors consists of two parts. First, we must develop a technique to recognize points in the spatial domain where the measured data are inconsistent, and ensure that these regions of uncertainty do not have undue influence on the rest of the approximation. Second, we must have a way to identify the best reconstruction from the set of J solutions. With regard to the first, the *variance based weighted joint sparsity* (VBSJ) algorithm, developed in [1], can be adapted for the rogue measurement problem. The idea is described below.

We begin by gathering the (processed) measurements from (2.4) into a measurement matrix given by

$$\check{F} = \left[\check{\boldsymbol{f}}^1 \; \check{\boldsymbol{f}}^2 \; \cdots \; \check{\boldsymbol{f}}^J\right] \in \mathbb{R}^{N\times J}. \tag{3.1}$$

We note that in most applications the initial data sets will come from (2.2), so it will be necessary to construct $\check{\boldsymbol{f}}^j$, $j = 1, \cdots, J$. Techniques other than (2.4) may be used for this purpose, however, and it might be sufficient to use a more cost efficient algorithm. Moreover, in some cases only one data vector is acquired, but is then processed in multiple (i.e. J) ways, with each processing providing different information. Indeed this was the case for one example discussed in [1], where one vector of Fourier data was collected but then several edge detection algorithms were used to construct jump function vectors (e.g. $\boldsymbol{y}^j$ in (2.4)). For ease of presentation, in this paper we use the traditional interpretation of (2.2) followed by the computation of (2.4) for a given set of J measurement vectors to obtain (3.1), and leave these other cases to future work.

Next we define

$$\mathcal{P} = \left[\mathcal{L}\check{\boldsymbol{f}}^1 \; \mathcal{L}\check{\boldsymbol{f}}^2 \; \cdots \; \mathcal{L}\check{\boldsymbol{f}}^J\right] \in \mathbb{R}^{N\times J} \tag{3.2}$$

as the matrix of J vectors approximating some sparse feature of the underlying function f. For example, here $\mathcal{L}$ is the PA transform operator so that $\mathcal{L}\check{\boldsymbol{f}}^j$ is an approximation of the edges of piecewise smooth f on the set of grid points given in (2.1).[4] Note that even if f is known explicitly, $\mathcal{L}\check{\boldsymbol{f}}^j$ will only be *approximately* zero in smooth regions, and hence is not truly sparse. However, the *behavior* of $\mathcal{L}\check{\boldsymbol{f}}^j$ should be consistent across all data sets, $j = 1, \cdots, J$, especially in smooth regions where $|\mathcal{L}\check{\boldsymbol{f}}^j|$ is small. This behavior should be confirmed in the variance vector $\check{\boldsymbol{v}} = (\check{v}_i)_{i=1}^N$, where each component is given by

$$\check{v}_i = \frac{1}{J}\sum_{j=1}^J \mathcal{P}_{i,j}^2 - \left(\frac{1}{J}\sum_{j=1}^J \mathcal{P}_{i,j}\right)^2, \quad i = 1, ..., N. \tag{3.3}$$

That is, (3.3) should yield small values in smooth regions when the data measurements are consistent. Note that $\text{supp}(\check{\boldsymbol{v}}) \approx \bigcup_{j=1}^J \text{supp}(\mathcal{L}\check{\boldsymbol{f}}_j)$.

We will exploit (3.3) in determining how the joint sparsity algorithm should be regularized. Figure 2 demonstrates how this may be useful. Five measurement vectors of the function in Example 1, where $\boldsymbol{A}$ has i.i.d. entries sampled from a uniform distribution on $[0, 1]$ and the noise is Gaussian with mean zero and variance 0.1, is shown in the top left. The bottom left displays the corresponding sparsity vectors, $\mathcal{L}\check{\boldsymbol{f}}^j$. Observe that the variance of the sparsity vectors, provided in the top right, is spatially variant, with the larger values occuring near the jump discontinuities as well as where more noise is apparent in the data measurements. This suggests that a spatially variant (weighted) ℓ_1 norm might work better than the uniform $\ell_{2,1}$ norm in regularizing the joint sparsity approximation. Algorithm 1 describes this process.

Remark 1 Observe that in contrast to (2.6), any $p \in [1, 2]$ can be used in Step 5 of Algorithm 1. While $p = 1$ is consistent with compressive sensing techniques, a spatially variant weighting vector may relax the requirements on p while still achieving the goal of sparsity. Intuitively, using ℓ_1 effectively promotes sparsity because of the higher penalty placed on small values in the

[4] Specifically it approximates the jump function $[f](x) = f(x^+) - f(x^-)$ on a set of N grid points.

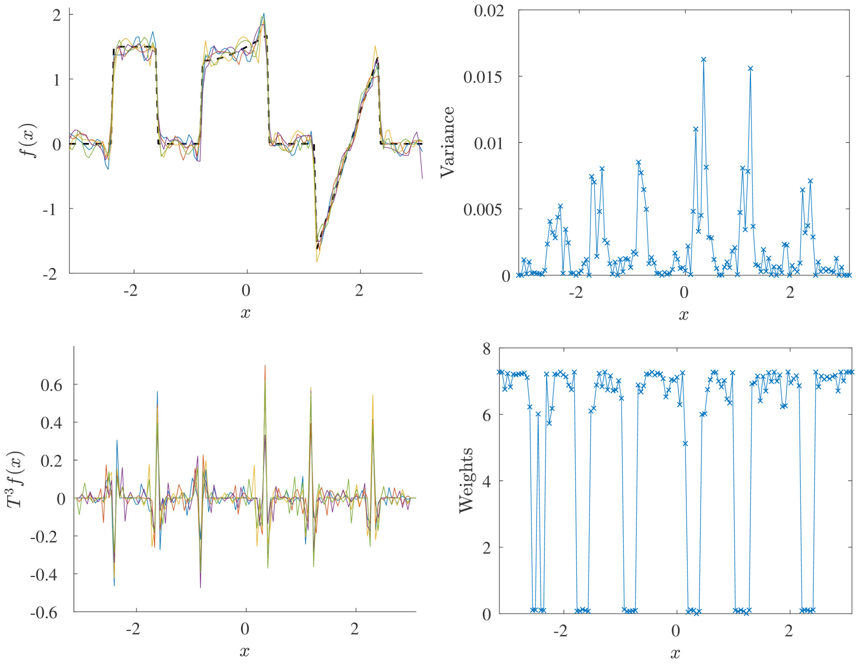

Fig. 2: (top-left) Five measurements of the underlying function in Example 1, acquired using (2.4). (bottom-left) Corresponding five sparsity vectors (3.2) with order $m = 3$. (top-right) The variance of the sparsity vectors calculated using (3.3). (bottom-right) The corresponding weights calculated as in (3.6).

Algorithm 1 Variance-Based Joint Sparsity algorithm

```
Recover the vectors $\check{\boldsymbol{f}}^j$, $j = 1, \dots, J$, separately using (2.4) to obtain (3.1).
Compute the variance of $\mathcal{L}\check{\boldsymbol{f}}^j$, $j = 1, \cdots, J$, using (3.3).
Use the results from (3.3) to determine the weights for the weighted $\ell_p$ norm, $1 \leq p \leq 2$, in the joint sparsity
    reconstruction. In particular, $\check{v}_i$ should be large when the index $i$ belongs to the support of $\check{\boldsymbol{v}}$, while $\check{v}_i \approx 0$
    otherwise. Hence we compute a vector of nonnegative weights $\boldsymbol{w} = (w_i)_{i=1}^N$, $0 \leq w_i \leq C$, $C \in \mathbb{R}$ based on
    this information. In general, $w_i \approx 0$ when $\check{v}_i$ is large and $w_i \approx C$ when $\check{v}_i \approx 0$. The weights we design for this
    purpose are provided in (3.6).
Determine data vector $\hat{\boldsymbol{y}} \in \boldsymbol{y}^j$, $j = 1, \cdots, J$, and corresponding matrix $\hat{\boldsymbol{A}}$ that will be used as the "best"
    initial vector approximation. This is done according to (3.9) and (3.10).
Solve the weighted $\ell_p$ minimization problem to get the final reconstruction of the vector $\boldsymbol{f}$:

    $\hat{\boldsymbol{g}} = \underset{\boldsymbol{g} \in \mathbb{R}^N}{\text{argmin}} \; \frac{1}{p} ||\mathcal{L}\boldsymbol{g}||_{p,\boldsymbol{w}}^p + \frac{\mu}{2} ||\hat{\boldsymbol{A}}\boldsymbol{g} - \hat{\boldsymbol{y}}||_2^2, \qquad (3.4)$

    for $\mu > 0$ a constant parameter.
```

reconstruction of what is presumably sparse (e.g. edges of piecewise smooth f), as compared to the standard ℓ_2 minimization, which imposes a penalty proportional to the square of each value in the reconstructed edge vector. Employing a (spatially variant) weighted ℓ_2 minimization designed to more strongly enforce small values in sparse regions should yield the same desired property for promoting sparsity. Moreover, using $|| \cdot ||_{2,\boldsymbol{w}}$ will be much more efficient numerically, since a closed form gradient of the objective function is available. A complete characterization of ℓ_1 and weighted ℓ_2 minimizers can be found in [13].

3.1 Weight design

In contrast to (2.6), where each grid point in the sparsity domain is equally weighted in the regularization term, Algorithm 1 uses a spatially variant regularization, with the weights $(w_i)_{i=1}^N$ being inherently linked to (3.3). In particular, since small variance values strongly suggest joint sparsity in the sparsity domain, the associated values $|\mathcal{L}\boldsymbol{f}_i|$, where $\boldsymbol{f}_i \approx f(x_i)$ of the underlying function and $\mathcal{L}$ is the sparsifying transform operator, should be heavily penalized in the regularization term. On the other hand, large variance values may indicate that the the corresponding indices belong to the support of the function (or image) in the sparsity domain. Large variance values may also indicate unreliable information at that particular spatial grid point. Hence $|\mathcal{L}\boldsymbol{f}_i|$ should be penalized less at those indices when minimizing the regularization term. Figure 2 (bottom right) depicts the weights chosen by (3.6) to minimize the weighted ℓ_p norm in (3.4).

From the discussion above and illustrated in Figure 2, we see that the weights for the regularization term should not depend on how the measurements in (2.4) are constructed, but rather only the expectation that they be jointly sparse in the same domain, as defined in Definition 2. In our examples, we assume that this joint sparsity occurs in the edge domain. The variance calculated in (3.3) provides a means of determining the actual joint sparsity, and moreover provides us a way to reduce the effects of bad data.

As described in Algorithm 1, the PA transformation is used to approximate the edges of the underlying function or image from which the weighting vector $\boldsymbol{w}$ is scaled according to the spatially variant jump height, (3.3), of our solutions. To specifically determine $\boldsymbol{w}$ we first define

$$\tilde{\mathcal{P}} = \left[\tilde{\mathcal{P}}_1\ \tilde{\mathcal{P}}_2 \cdots \tilde{\mathcal{P}}_J\right] \in \mathbb{R}^{N\times J}$$

as the normalized PA transform matrix from (3.2), where

$$\tilde{\mathcal{P}}_{i,j} = \frac{|\mathcal{P}_{i,j}|}{\max\limits_i |\mathcal{P}_{i,j}|}, \quad j = 1, ..., J.$$

We then define a weighting scalar C as the average ℓ_1 norm across all measurements of the normalized sparsifying transform of our measurements,

$$C = \frac{1}{J}\sum_{j=1}^{J}\sum_{i=1}^{N}\tilde{\mathcal{P}}_{i,j}, \tag{3.5}$$

which will enable us to further scale the weights according to the magnitude of the values in the sparsity domain. This will ultimately reduce the need for fine tuning regularization parameters in the numerical implementation. Finally, $\boldsymbol{w}$ is constructed element-wise as

$$w_i = \begin{cases} C\left(1 - \frac{v_i}{\max_i v_i}\right), & i \notin I \\ \frac{1}{C}\left(1 - \frac{v_i}{\max_i v_i}\right), & i \in I \end{cases} \tag{3.6}$$

where I consists of the indices i such that

$$\frac{1}{J}\sum_{j=1}^{J}\tilde{\mathcal{P}}_{i,j} > \tau. \tag{3.7}$$

Here τ is a threshold chosen so that when (3.7) is satisfied, we assume there is a corresponding edge at x_i, and that the index i is part of the support in the sparse domain of $\boldsymbol{f}$. Since the jumps are normalized, it is reasonable for $\tau = \mathcal{O}(\frac{1}{N})$, that is, τ is resolution dependent. Because noise in the system, we choose $\tau > \frac{1}{N}$, and in our examples $\tau = .1$, and note that if more is known apriori about the size of the noise, then τ can be chosen accordingly. In general as τ increases, more noise is assumed to be in the system, which corresponds to a more uniform weighting scheme. Choosing weights based on information about system noise and nuisance parameters will be addressed more in future investigations.

Observe that $w_i \in [0, C]$, $i = 1, \cdots, N$ and $C > 1$. The weighting scalar C defined in (3.5) allows the regularization to better account for functions that contain multiple edges with different magnitudes. Specifically, the weights in (3.6) are designed to scale the penalty of the regularization according to the size of the jump, with the largest weights being reserved for regions where the function is presumably smooth. The intuition used for determining the weights formula in (3.6) is illustrated in Figure 2. In this case we have $J = 5$ measurements for Example 1. We use the PA transform in (3.2) with order $m = 3$, and $\mu = .25$ in (2.4).[5]

For comparative purposes, we will also consider weights that were used in [1]

$$w_i = \frac{1}{v_i + \epsilon}, \tag{3.8}$$

where ϵ is a small parameter chosen to avoid dividing by zero. In [1] it was demonstrated that this weighting strategy was robust in sparse signal recovery (in the noiseless case) for $\epsilon = 10^{-2}$.

3.2 Determining the optimal solution vector

The traditional method that exploits the joint sparsity of J multi-measurement vectors (MMV) in (2.6) can recover J solution vectors. This is also the case in Algorithm 1, however we are only interested in one "best" solution. Moreover, we want to avoid using any bad information or rogue vectors as the base of our solution. Therefore, we choose the final data vector $\hat{\boldsymbol{y}}$ in Step 4 of Algorithm 1 to be one whose corresponding measurements are closest to most of the other measurement vectors in the set of J vectors. Thus we define the distance matrix $\mathcal{D}$ with entries

$$\mathcal{D}_{i,j} = ||\check{\boldsymbol{f}}^i - \check{\boldsymbol{f}}^j||_2, \tag{3.9}$$

where each $\check{\boldsymbol{f}}$ is defined in (2.4). The data vector $\hat{\boldsymbol{y}} = \boldsymbol{y}^{j^*}$ and forward operator $\hat{\boldsymbol{A}} = \boldsymbol{A}^{j^*}$ correspond to the j^*th index that solves

$$(i^*, j^*) = \underset{\substack{1 \le i,j, \le J \\ i \ne j}}{\operatorname{argmin}} \mathcal{D}_{i,j}. \tag{3.10}$$

WLOG, we choose the optimal *column* index j^* for the final reconstruction. Note that because D is symmetric, the optimal row index can similarly be used as an indicator of good data. An example of this process is depicted in Figure 3. On the left we see ten measurements of Example 2 where the first five measurements are false measurements. Displayed on the right is the matrix $\mathcal{D}$ given in (3.9). Assuming that the number of true measurement vectors $J - K$ is greater than 2, it is reasonable to use (3.10) to determine the "best" data vector for the final reconstruction. It must also be true that rogue data vectors are not similar to one another, that is, for all $i, j = 1, \cdots, K$, $||\check{\boldsymbol{f}}^i - \check{\boldsymbol{f}}^j|| > \sigma$ where $\sigma > 0$ is a chosen distance threshold. The quality of the solution is clearly dependent on the number of rogue measurements in the collection set. More analysis is needed to determine the relationship between the ratio of false and true measurements and the success of Algorithm 1, and will be the subject of future work.

[5] It was observed in [8] that multiple scales in jump heights can be handled by iteratively redefining a weighted $\ell_{2,1}$ norm in the MMV case (2.6). This method proved to be computationally expensive, as the optimization problem must be resolved at each iteration, however.

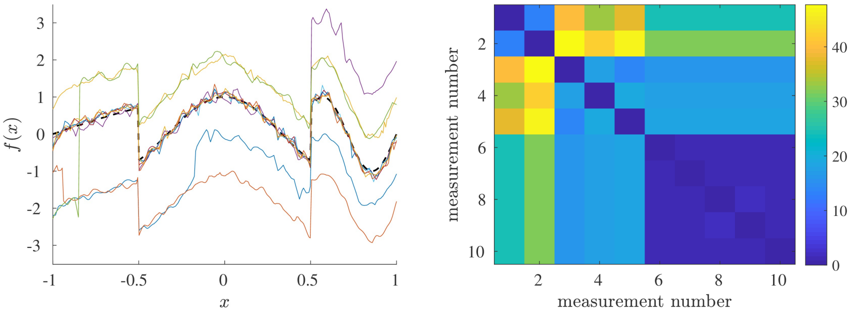

Fig. 3: (left) Five false measurements and five true measurements of Example 2. The true underlying function is displayed as the bold dashed line. (right) The corresponding construction of the distance matrix $\mathcal{D}$ in (3.9).

4 Efficient implementation of Algorithm 1

Once we determine the initial solutions, (2.4), the weighting vector, (3.6), and the most suitable vector for reconstruction, (3.10), we can now approximate the solution to (3.4) in Algorithm 1. When using the weights designed in (3.6), we eliminate the need to tune the parameter μ to ensure convergence, and thus we set $\mu = 1$ in (3.4) for our experiments.

For $x \in \mathbb{R}^N$, the weighted ℓ_p norm is defined as

$$||x||_{p,\boldsymbol{w}} = \left(\sum_{i=1}^{N} w_i |x_i|^p \right)^{1/p} = ||Wx||_p, \tag{4.1}$$

where $W = \text{diag}(\boldsymbol{w}) \in \mathbb{R}^{N \times N}$. With this definition we can now solve (3.4) using stardard ℓ_p minimization techniques, see e.g. [19, 22, 36, 38].

In two dimensions, especially as the number of data points increase, it quickly becomes computationally expensive to write the weights as a diagonal matrix. That is, even though “stacking” the columns (noted by the *vec* function) holds intuitive appeal for solving (3.4), since $W = \text{diag}(vec(\boldsymbol{w})) \in \mathbb{R}^{N^2 \times N^2}$, the problem becomes computationally prohibitive.

Fortunately, however, we are able to show that the ADMM algorithm can also be applied in this case, as will be described below. For this purpose we first define the weighted ℓ_p norm as

$$||x||_{p,\boldsymbol{w}}^p = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,j} |x_{i,j}|^p, \tag{4.2}$$

where $w_{i,j}$ are elements of $\boldsymbol{w} \in \mathbb{R}^{N \times N}$ and $x \in \mathbb{R}^{N \times N}$.

4.1 The ADMM algorithm for weighted ℓ_1

We now demonstrate how the ADMM can be applied to solve (3.4) when $p = 1$. While the algorithm can be used for either the one or two dimensional case, for computational efficiency, such an approach is critical for two dimensional problems.

To start, we write (3.4) with $\mu = 1$ as the equivalent non-parametric weighted ℓ_1 problem

$$(\hat{\boldsymbol{g}}, \hat{\boldsymbol{z}}) = \left\{ \underset{\boldsymbol{g},\boldsymbol{z}}{\text{argmin}} \;\; ||\boldsymbol{z}||_{1,\boldsymbol{w}} + \frac{1}{2} ||\hat{\boldsymbol{A}}\boldsymbol{g} - \hat{\boldsymbol{y}}||_2^2 \quad \text{subject to} \quad \boldsymbol{\mathcal{L}}\boldsymbol{g} = \boldsymbol{z} \right\}. \tag{4.3}$$

Here we assume $\hat{\boldsymbol{A}}, \boldsymbol{g}, \boldsymbol{z}$ and $\hat{\boldsymbol{y}}$ are all in $\mathbb{R}^{N\times N}$. Because of the non-differentiability in the $\ell_{1,\boldsymbol{w}}$ norm, we introduce slack variables $\boldsymbol{z} \in \mathbb{R}^{N\times N}$ and the Lagrangian multiplier $\boldsymbol{\nu} \in \mathbb{R}^{N^2}$ to minimize

$$\underset{\boldsymbol{g},\boldsymbol{z}}{\operatorname{argmin}}\left\{||\boldsymbol{z}||_{1,\boldsymbol{w}} - \boldsymbol{\nu}^T vec\left(\mathcal{L}\boldsymbol{g} - \boldsymbol{z}\right) + \frac{\beta}{2}||\mathcal{L}\boldsymbol{g} - \boldsymbol{z}||_2^2 + \frac{1}{2}||\hat{\boldsymbol{A}}\boldsymbol{g} - \hat{\boldsymbol{y}}||_2^2\right\}. \tag{4.4}$$

Remark 2 Two parameters, μ from (3.4) and β in (4.4), typically must be prescribed in ADMM. In (4.3) we observe that we can use $\mu = 1$ since the weighting of this term is considered in the construction of the weighting vector (3.6). We also note that although we have not formally analyzed the impact of using the weighted ℓ_1 norm on the overall rate of convergence, our numerical experiments demonstrate that choosing $\beta = 1$ yields reasonably fast convergence. A study of how the weighting vector affects the convergence rate for different choices of β will be the subject of future investigations. Thus we see that the ADMM method for VBJS is robust, as no fine tuning of parameters is needed at the optimization stage.

The problem is now split into two sub-problems, known as the *z-subproblem* and the *g-subproblem.*

The *z-subproblem*

To analyze the *z-subproblem*, we assume that the value of $\boldsymbol{g}$ is known and fixed and set $\beta = 1$ in (4.4), so that

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\operatorname{argmin}}\left\{||\boldsymbol{z}||_{1,\boldsymbol{w}} - \boldsymbol{\nu}^T vec\left(\mathcal{L}\boldsymbol{g} - \boldsymbol{z}\right) + \frac{1}{2}||\mathcal{L}\boldsymbol{g} - \boldsymbol{z}||_2^2\right\}. \tag{4.5}$$

Lemma 1 demonstrates that a closed form solution exists in general for the z-subproblem for any $\beta > 0$.

Lemma 1 *For a given $\beta > 0$, $x, y \in \mathbb{R}^{N\times N}$ and $\nu \in \mathbb{R}^{N^2}$, the minimizer of the proximal operator, [29],*

$$Prox_{f/\beta}(x) = \underset{x}{\operatorname{argmin}}\left\{f(x) + \frac{\beta}{2}||y - x||_2^2\right\} \tag{4.6}$$

where $f(x) = ||x||_{1,w} - \nu^T vec(y - x)$, is given by the shrinkage-like formula

$$\hat{x} = \max\left\{\left|y - \frac{\nu}{\beta}\right| - \frac{w}{\beta}, 0\right\} sign\left(y - \frac{\nu}{\beta}\right). \tag{4.7}$$

The proof of Lemma 1 can be found in the Appendix. In light of Lemma 1, the closed form solution to (4.5) is given as

$$\hat{\boldsymbol{z}} = \max\left\{|\mathcal{L}\boldsymbol{g} - \boldsymbol{\nu}| - \boldsymbol{w}, 0\right\} \operatorname{sign}\left(\mathcal{L}\boldsymbol{g} - \boldsymbol{\nu}\right). \tag{4.8}$$

The *g-subproblem*

Once the *z-subproblem* is solved, we can proceed using standard ADMM. Specifically, $\boldsymbol{z}$ is held fixed, $\beta = 1$ in (4.4), and we construct *g-subproblem* from (4.4) as

$$\hat{\boldsymbol{g}} = \underset{\boldsymbol{g}}{\operatorname{argmin}}\ \ J(\boldsymbol{g}) := \left\{\frac{1}{2}||\mathcal{L}\boldsymbol{g} - \boldsymbol{z}||_2^2 + \frac{1}{2}||\hat{\boldsymbol{A}}\boldsymbol{g} - \hat{\boldsymbol{y}}||_2^2 - \boldsymbol{\nu}^T vec\left(\mathcal{L}\boldsymbol{g} - \boldsymbol{z}\right)\right\}. \tag{4.9}$$

Since $\hat{\boldsymbol{A}}$ is ill-conditioned in many applications of interest we solve (4.9) using gradient descent, [19, 22, 36],

$$\boldsymbol{g}_{k+1} = \boldsymbol{g}_k - \alpha_k \nabla_{\boldsymbol{g}} J(\boldsymbol{g}_k), \tag{4.10}$$

where

$$\nabla_{\boldsymbol{g}} J(\boldsymbol{g}_k) = -\boldsymbol{\nu}^T \mathcal{L} + (\mathcal{L})^T (\mathcal{L}\boldsymbol{g} - \boldsymbol{z}) + \hat{\boldsymbol{A}}^T (\hat{\boldsymbol{A}}\boldsymbol{g} - \hat{\boldsymbol{y}}). \tag{4.11}$$

Note that for ease of presentation we have again dropped the *vec* notation, although it is of course needed for implementation. The step length is chosen as the Barzilai-Borwein (BB) step (see [5]),

$$\alpha_k = \frac{\boldsymbol{s}_k^T \boldsymbol{s}_k}{\boldsymbol{s}_k^T \boldsymbol{u}_k}, \tag{4.12}$$

with

$$\begin{aligned} \boldsymbol{s}_k &= \boldsymbol{g}_k - \boldsymbol{g}_{k-1} \\ \boldsymbol{u}_k &= \nabla_{\boldsymbol{g}} J(\boldsymbol{g}_k) - \nabla_{\boldsymbol{g}} J(\boldsymbol{g}_{k-1}). \end{aligned}$$

A backtracking algorithm is performed to ensure α_k is not chosen to be too large. This requires checking what is known as the Armijo condition, [38], which guarantees that using (4.12) sufficiently reduces the magnitude of the objective function. Algorithmically, the Armijo condition is given by

$$J(\boldsymbol{g}_k - \alpha \nabla_{\boldsymbol{g}} J(\boldsymbol{g}_k)) \leq J(\boldsymbol{g}_k) - \delta \alpha_k \nabla_{\boldsymbol{g}}^T J(\boldsymbol{g}_k) \nabla_{\boldsymbol{g}} J(\boldsymbol{g}_k), \tag{4.13}$$

where $\delta \in (0, 1)$. If the Armijo condition (4.13) is not satisfied, we backtrack and decrease the step length according to

$$\alpha_k = \rho \alpha_k,$$

where $\rho \in (0, 1)$ is the backtracking parameter. At the kth iteration of the algorithm, after the new $\boldsymbol{z}$ and $\boldsymbol{g}$ values are found using (4.8) and (4.9), the Lagrange multiplier is updated according to

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k - vec(\mathcal{L}\boldsymbol{g}_{k+1} - \boldsymbol{z}_{k+1}) \tag{4.14}$$

Algorithm 2 provides the weighted version of the ADMM. The technique involves alternating solving the z-subproblem (4.5) and g-subproblem (4.9) at each iteration. Typical parameter choices are $\rho = .4$ and $\delta = 10^{-4}$, [22, 38].

Algorithm 2 Weighted ADMM

```
Initialize $\boldsymbol{\nu}_0$. Determine weights $\boldsymbol{w}$, starting points $\boldsymbol{g}_0$ and $\boldsymbol{z}_0$ and maximum number of iterations $K$.
for $i = 0$ to $K$ do
  Set $0 < \rho, \delta < 1$ and tolerance $tol$.
  while $||\boldsymbol{g}_{k+1} - \boldsymbol{g}_k|| > tol$ do
    Compute $\boldsymbol{z}_{k+1}$ using (4.8).
    Set $\alpha_k$ using (4.12).
    while Armijo condition (4.13) unsatisfied do
      Backtrack: $\alpha_k = \rho\alpha_k$.
    end while
    Compute $\boldsymbol{g}_{k+1}$ using (4.10) and (4.11).
  end while
  Update Lagrange multiplier according to (4.14).
end for
```

4.2 Efficient implementation for the ℓ_2 case

When $p = 2$ in (3.4) we solve

$$\hat{\boldsymbol{g}} = \underset{\boldsymbol{g}}{\operatorname{argmin}} \; J(\boldsymbol{g}) := \left\{ \frac{1}{2} ||\mathcal{L}\boldsymbol{g}||_{2,\boldsymbol{w}}^2 + \frac{1}{2} ||\hat{\boldsymbol{A}}\boldsymbol{g} - \hat{\boldsymbol{y}}||_2^2 \right\} \tag{4.15}$$

using the gradient descent method defined in (4.10). However, some care must be taken to derive the gradient of the first term of (4.15). According to (4.2), for $\mathcal{L}, \boldsymbol{g}, \boldsymbol{w} \in \mathbb{R}^{N \times N}$,

$$||\mathcal{L}\boldsymbol{g}||_{2,w}^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}\left(\sum_{k=1}^{N}\mathcal{L}_{i,k}g_{k,j}\right)^2. \tag{4.16}$$

Taking the derivative of (4.16) with respect to an element of $\boldsymbol{g}$ yields

$$\frac{\partial}{\partial g_{k,j}}||\mathcal{L}\boldsymbol{g}||_{2,\boldsymbol{w}}^2 = 2\sum_{i=1}^{N} w_{i,j}\mathcal{L}_{i,k}\left(\sum_{l=1}^{N}\mathcal{L}_{i,l}g_{l,j}\right), \quad k,j = 1,...,N.$$

Performing this operation over all $k, j = 1, ..., N$, produces

$$\nabla_{\boldsymbol{g}}||\mathcal{L}\boldsymbol{g}||_{2,\boldsymbol{w}}^2 = 2\mathcal{L}^T\left[\boldsymbol{w}\odot(\mathcal{L}\boldsymbol{g})\right], \tag{4.17}$$

where $\odot$ denotes the pointwise Hadamard product. Thus, the gradient of the objective function J in (4.15) is given by

$$\nabla_{\boldsymbol{g}}J(\boldsymbol{g}) = \mathcal{L}^T\left[\boldsymbol{w}\odot(\mathcal{L}\boldsymbol{g})\right] + \hat{\boldsymbol{A}}^T(\hat{\boldsymbol{A}}\boldsymbol{g} - \hat{\boldsymbol{y}}). \tag{4.18}$$

Using (4.18) in (4.10) with the BB step length (4.12), we can now solve (4.15) for $\hat{\boldsymbol{g}}$. The weighted ℓ_2 gradient descent process is described in Algorithm 3. Typical parameter choices again are $\rho = .4$ and $\delta = 10^{-4}$ and a starting step length of $\alpha_0 = 1$ is chosen to initiate the algorithm [38].

Algorithm 3 Weighted Gradient Descent

```
1: Initialize starting points $\boldsymbol{g}_0$ and $\alpha_0$, parameters $\delta, \rho \in (0,1)$ and tolerance $tol$.
2: Determine weights $\boldsymbol{w}$.
3: while $||\boldsymbol{g}_{k+1} - \boldsymbol{g}_k|| > tol$ do
4:    Set $\alpha_k$ using (4.12).
5:    while Armijo condition (4.13) unsatisfied do
6:       Backtrack: $\alpha_k = \rho\alpha_k$.
7:    end while
8:    Compute $\boldsymbol{g}_{k+1}$ using (4.10) and (4.18).
9: end while
```

5 Numerical Results

We test the variance based joint sparsity (VBJS) technique in three different situations and compare our method in Algorithm 2 to the typical $\ell_{2,1}$ minimization algorithm in (2.6), the SMV case, and the VBJS method with the weights given in (3.8). In our experiments we employ both ℓ_1 and ℓ_2 regularization in (3.4) with $\mu = 1$, demonstrating the accuracy and robustness of our methods in each case. As was shown in [1], the VBJS method is consistently more cost efficient than $\ell_{2,1}$ regularization. Moreover, using weighted ℓ_2 regularization is clearly less costly than using weighted ℓ_1.

First we consider recovering sparse signals. A similar experiment was performed for VBSJ in [1] on noiseless data. In our example the measurement vectors contain noise, and there are also measurements that contain false information. In this regard it is important to note that the weights in (3.6) are designed so that *no* additional parameters are needed in (4.4). That is, $\beta = 1$ in the *z-subproblem* and regularization parameters normally included in the ADMM *g-subproblem* are not needed [22]. However, this is not the case when using (3.8), where we will see that regularization parameters are needed to obtain any meaningful results. As noted previously, to obtain the first measurements in each algorithm, we use (2.4) with μ sampled from a uniform distribution for each $j = 1, \cdots, J$, thus simulating the ad-hoc procedure for selecting typical regularization parameters.

For the second experiment we consider two one-dimensional signals that exhibit sparsity in the edge domain. We apply the VBJS technique for both $p = 1$ and 2 in the weighted ℓ_p regularization, and again compare our method to techniques in [1] with (3.8). In our third test we reconstruct two-dimensional images with sparse edges.

5.1 Case 1: Sparse Signal Recovery

We seek to recover the sparse signal $\boldsymbol{f}$ from a set of measurment vectors. This problem has been widely studied within the context of MMV, [12, 16, 17]. An adaptively weighted ℓ_1 reconstruction method was developed in [8] for the single measurement vector (SMV) case, and the VBJS method using the weights in (3.8) was developed for MMV in [1]. In this case each data vector $\{\boldsymbol{y}^j\}_{j=1}^J$ in (2.2) is acquired using a measurement matrix $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ where each element of $\boldsymbol{A}$ is sampled independently from a zero mean unit variance Gaussian distribution. The corresponding noise vectors $\{\boldsymbol{\eta}^j\}_{j=1}^J$ are i.i.d. Gaussian with zero mean and unit variance. Of the J measurements, K contain false information and in some cases are complete misrepresentations of the underlying signal. To recover the sparse signal $\mathbf{f}$ we used (4.1) with $p = 1$ in Algorithm 2. Since the $J - K$ true measurements have overlapping support, we use the PA transform with order $m = 0$, that is $\mathcal{L} = \boldsymbol{I}$ in the sparsity regularization term.

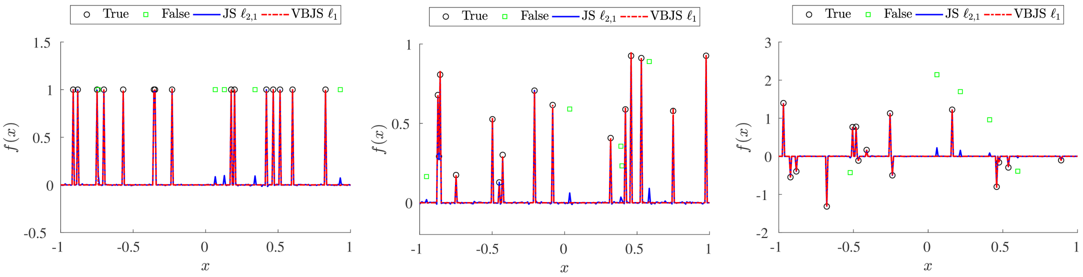

Fig. 4: Sparse signal recovery employing the usual $\ell_{2,1}$ joint sparsity method in (2.6) and our proposed VBJS ℓ_1 technique. Here there are $J = 10$ measurements of which $K = 5$ contain false data. (left) Binary data values. (middle) Data values sampled from a uniform distribution on $[0, 1]$. (right) Data values sampled from a zero-mean unit-variance Gaussian distribution.

Figure 4 compares the signal recovery results for three sparse signals using the VBJS ℓ_1 technique (dot-dashed) and the more classical $\ell_{2,1}$ JS regularization in (2.6), implemented using techniques in [14, 42]. In this case the final JS reconstruction is the pointwise average of the recovered vectors, $\{\hat{\boldsymbol{f}}^j\}_{j=1}^J$. In Figure 4(left), the signal consists of a sparse number of binary values, while the signals in Figure 4(middle) and (right) contain a sparse number of values sampled from a uniform distribution on $[0, 1]$ and a Gaussian distribution with zero-mean and unit-variance respectively. In each case there are a total of $J = 10$ measurements vectors where each of the $K = 5$ measurement vectors are corrupted by adding a single false data point with height sampled from the corresponding distribution. Based on parameters used in other studies, [1, 27], we choose $N = 256$, $M = 100$ and sparsity $s = ||\boldsymbol{f}||_0 = 15$ for all three experiments. As is evident in Figure 4, the VBJS method successfully recovers each of the three sparse signals with limited influence from the false data. Conversely, the classic $\ell_{2,1}$ JS method is indeed influenced by the bad data. Similar behavior (not reported here) can be observed for different choices of N, M J, K and s.

Table 1: Relative reconstruction errors (5.1) for the traditional $\ell_{2,1}$ JS method and the VBJS method with $p = 1$ using the weights defined in (3.6) and (3.8).

False	Binary			Uniform			Gaussian		
Data	JS $\ell_{2,1}$	(3.6)	(3.8)	JS $\ell_{2,1}$	(3.6)	(3.8)	JS $\ell_{2,1}$	(3.6)	(3.8)
0%	.0127	.0104	.0383	.0254	.0244	.0579	.0142	.0153	.0388
20%	.1724	.0094	.0288	.1068	.0234	.0594	.1304	.0144	.0314
50%	.2961	.0108	.0499	.2943	.0243	.0621	.1249	.0168	.0262
90%	.1543	.0083	.0358	.3089	.0196	.0775	.1233	.0128	.0397

Table 1 displays the relative error,

$$E = \frac{||\hat{\boldsymbol{g}} - \boldsymbol{f}||_2}{||\boldsymbol{f}||_2}, \tag{5.1}$$

for the recovery vector $\hat{\boldsymbol{g}}$. In each case we use $J = 10$ measurements where K, the number of false data measurements, is based on the given percentage in the first column. For consistent comparison we use $N = 256$, $M = 100$ and sparsity $s = ||\boldsymbol{f}||_0 = 20$ in all cases. It is evident that the VBJS ℓ_1 technique yields small error even as the percentage of false data increases. Conversely, the traditional $\ell_{2,1}$ JS method is more susceptible to false data. For comparison we included results using the weights given in (3.8). We note that to handle the noise and different jump heights in the problem, when using the weights in (3.8), we must solve (3.4) by tuning the parameter μ to $\mu = .1$. Regardless, it is evident that the weights designed in (3.6) outperform the weights in (3.8) in all cases, and in the former case, *no* additional parameter tuning is needed.

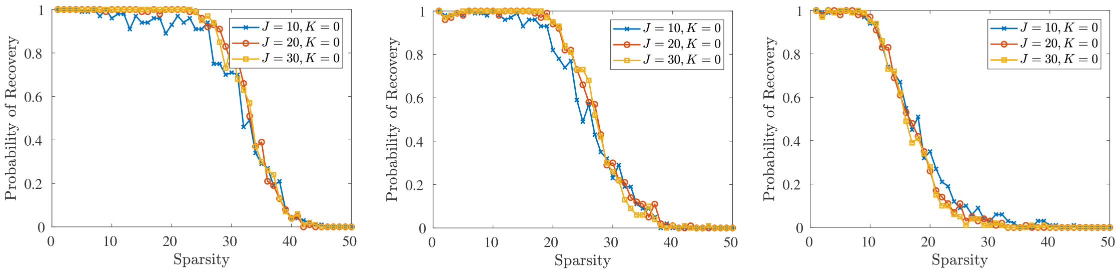

Fig. 5: Probability of successful recovery of the sparse (left) binary signal, (middle) uniform signal, and (right) Gaussian signal with $J = 10, 20$ and 30 measurements, none of which contain false data. Recovery is deemed a success if $||\hat{\boldsymbol{g}} - \boldsymbol{f}||_\infty \leq 5 \times 10^{-3}$, where $\hat{\boldsymbol{g}}$ is the recovery vector and $\boldsymbol{f}$ is the true solution vector.

To further demonstrate the success of our method, at varying levels of sparsity for different numbers of measurements and false data, we calculate the probability that the sparse signal is successfully recovered. Similar analysis was done in [8,12,16,17,24,44]. Specifically, the probability of recovery is calculated over 100 trials at the specified configuration (J, K, and sparsity level) with $N = 256$, $M = 100$ and no additive noise. Recovery is deemed a success if $||\hat{\boldsymbol{g}} - \boldsymbol{f}||_\infty \leq 5 \times 10^{-3}$, that is, when the VBJS method can successfully distinguish signals larger than the resolution size, $\mathcal{O}(\frac{1}{N})$.

In Figure 5 we see the recovery plots for each of the three signals considered with $J = 10, 20$ and 30 measurements, none of which contain false data. In this case, additional measurements do not improve the already high recovery rates. However, in Figure 6 we see that as the percentage of

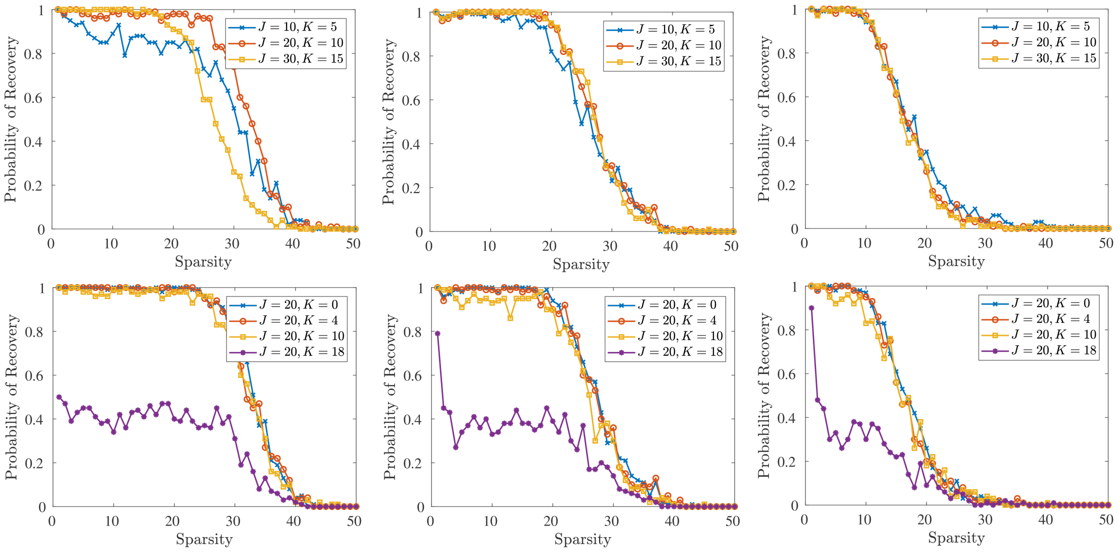

Fig. 6: The probability of recovery of the sparse signal with values sampled from a Gaussian distribution with zero-mean and unit-variance for various combinations of J, K and $||\boldsymbol{f}||_0$. Here $N = 256$ and $M = 100$. (left) Binary sparse vectors, (middle) uniform sparse vectors and (right) Gaussian sparse vectors.

measurements that are false increases, it becomes more advantageous to have more measurements. Across top row of Figure 6 the percentage of false data increases to 50% while the number of measurements changes from $J = 10, 20$ to 30 for each type of sparse vector (binary, uniform, and Gaussian). Across the bottom row of Figures 6 the number of measurements $J = 20$ remains fixed, while the percentage of false data included increases from 20% to 50% to 90%. We see that when 50% of the measurements are false, the probability of recovery remains high for large sparsity values. When the percentage of false data increases to 90%, most probability of recovery values fall below .5.

5.2 Case 2: Reconstructing one dimensional piecewise smooth functions

We now consider the reconstruction of two piecewise smooth functions, given by

Example 1 Define $f(x)$ on $[-\pi, \pi]$ as

$$f(x) = \begin{cases} \frac{3}{2}, & -\frac{3\pi}{4} \leq x < -\frac{\pi}{2} \\ \frac{7}{4} - \frac{x}{2} + \sin\left(x - \frac{1}{4}\right), & -\frac{\pi}{4} \leq x < \frac{\pi}{8} \\ \frac{11}{4}x - 5, & \frac{3\pi}{8} \leq x < \frac{3\pi}{4} \\ 0, & \text{otherwise.} \end{cases}$$

Example 2 Define $f(x)$ on $[-1, 1]$ as

$$f(x) = \begin{cases} \cos\left(\frac{\pi}{2}x\right), & -1 \leq x < -\frac{1}{2} \\ \cos\left(\frac{3\pi}{2}x\right), & -\frac{1}{2} \leq x < \frac{1}{2} \\ \cos\left(\frac{7\pi}{2}x\right), & \frac{1}{2} \leq x \leq 1 \end{cases}$$

Each function exhibits sparsity in the jump function domain, that is $\boldsymbol{f}$ is not sparse, but $||[\boldsymbol{f}]||_0 = s$, with $s << N$, and $[\boldsymbol{f}] = \{[f](x_j)\}_{j=1}^N$ is the corresponding vector of edges. We consider the proposed weights (3.6) and the weights given by (3.8) in [1] for the weighted ℓ_p reconstructions (3.4) with $p = 1$ and 2.

For both examples we seek to approximate f by constructing a solution $\boldsymbol{f}$ on N uniform points given by (2.1) from J vectors of $M \leq N$ acquired measurements. We acquire $J - K$ data vectors according to (2.2). The acquisition process for the K rogue vectors, as described below, considers situations where there is false information about the underlying solution as well as in the measurement matrix. In both examples we initialize the VBJS algorithm by constructing data vectors $\tilde{\boldsymbol{f}}^j$ for $j = 1, ..., J$ via (2.4) with μ sampled from a uniform distribution on $[0, 1]$. The sparsifying transform operator $\mathcal{L}$ is chosen to be the polynomial annihilation (PA) transform matrix of order $m = 2$ in (3.2).

In Example 1, the K false data vectors are formed by adding random shifts at random locations to the initial underlying function $\boldsymbol{f}$ in (2.2). That is, the data vectors (2.2) are modified such that

$$\boldsymbol{y}^j = \begin{cases} \boldsymbol{A}^j(\tilde{\boldsymbol{f}}^j) + \boldsymbol{\eta}^j, & j = 1, ..., K \\ \boldsymbol{A}^j(\boldsymbol{f}) + \boldsymbol{\eta}^j, & j = K+1, ..., J, \end{cases} \tag{5.2}$$

where each element $\tilde{f}_i^j$ of $\tilde{\boldsymbol{f}}^j$ is given as

$$\tilde{f}^j(x_i) = \begin{cases} f(x_i) + \alpha_j, & x \leq -1 + 2\gamma_j \\ f(x_i) + \beta_j, & x > 1 + 2\gamma_j. \end{cases}$$

Here α_j and β_j are random integers in $[-2, 2]$ for $j = 1, ..., K$ and each γ_j is i.i.d. sampled from a uniform distribution on $[0, 1]$. The forward model $\boldsymbol{A}^j \in \mathbb{R}^{M \times N}$ in (5.2) is defined as a matrix with i.i.d., zero-mean, unit-variance, Gaussian entries for all $j = 1, ..., J$, and the additive noise $\boldsymbol{\eta}^j$ is assumed to be i.i.d. Gaussian with zero-mean and variance equal to .16.

For Example 2, we choose $\boldsymbol{A}^j \in \mathbb{R}^{N \times N}$ ($M = N$) to be a subsampled discrete Fourier transform (DFT) matrix for $j = 1, ..., K$, and the standard DFT matrix for $j = K+1, ..., J$, so that

$$\boldsymbol{A}^j = \begin{cases} \frac{1}{\sqrt{N}} P_{\Omega^j} \boldsymbol{F}, & j = 1, ..., K \\ \frac{1}{\sqrt{N}} \boldsymbol{F}, & j = K+1, ..., J. \end{cases} \tag{5.3}$$

Here $\boldsymbol{F} \in \mathbb{C}^{N \times N}$ is the DFT matrix and $P_{\Omega^j} \in \mathbb{R}^{N \times N}$ is a row selector matrix where each $\Omega^j \subseteq \{1, ..., N\}$ randomly selects and zeros out $N/2$ rows of $\boldsymbol{F}$. We choose to replace 75% of the selected rows with a random vector $\gamma \sin(x)$, where γ is repeatedly sampled from the normal distribution. In this way, we can simulate K false and $J - K$ true data vectors according to (2.2) where $\boldsymbol{\eta}^j$ is chosen as complex Gaussian noise with zero mean and variance equal to .75 for all $j = 1, ..., J$.

Figures 7 and 8 display the results of reconstructing Examples 1 and 2 respectively using VBJS with weights defined in (3.6) and (3.8) for $p = 1$ and 2 in (3.4). It is evident that using our proposed weights yields improved accuracy as well as prevents the influence of misleading/false data. We repeat these experiments, without adding Gaussian noise to the data ($\boldsymbol{\eta}^j = 0$ for all $j = 1, ..., J$), with our proposed weights for $N = M = 32, 64, 128$ and 256, each time calculating the pointwise error in the reconstruction. That is, for each $\hat{\boldsymbol{g}}$ we calculate

$$\log_{10} |\hat{\boldsymbol{g}} - \boldsymbol{f}|. \tag{5.4}$$

The pointwise error plots corresponding to the reconstruction of Examples 1 and 2 are then displayed in Figure 9(top) and Figure 9(bottom), respectively, for $p = 1$ and 2 in (3.4). In Figure 9, the left two columns were calculated using our proposed weights (3.6) and the right two columns were calculated using the weights (3.8) given in [1]. The results shown here are consistent with

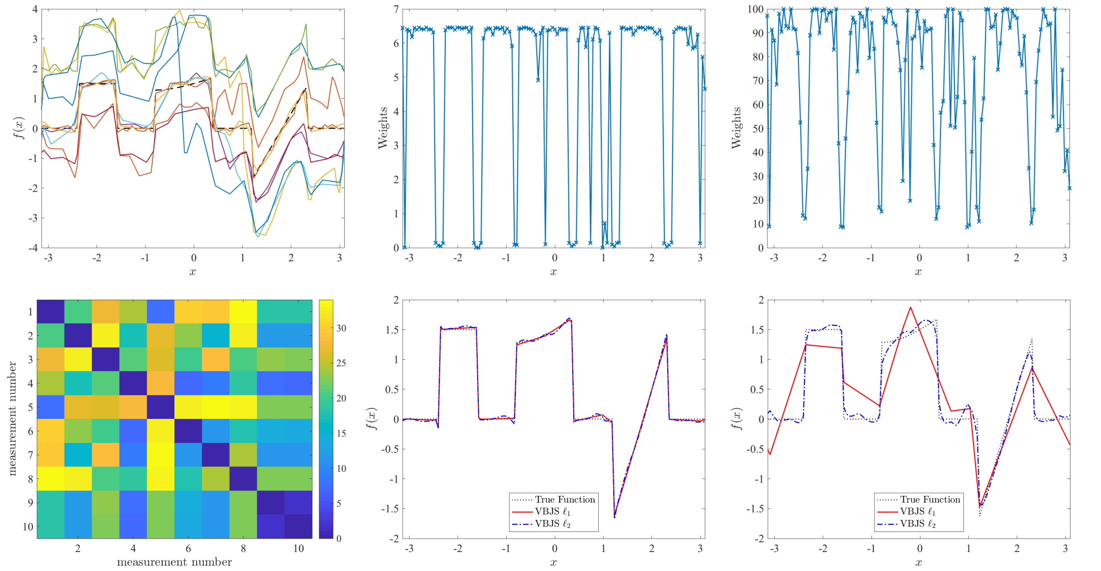

Fig. 7: (top-left) $J = 10$ measurement vectors with $K = 8$ false data acquired using (2.4) with (5.2). Weights proposed in (3.6) (top-middle) and (3.8) from [1] (top-right). (bottom-left) Corresponding distance matrix $\mathcal{D}$ in (3.9). VBJS ℓ_1 and ℓ_2 reconstructions with weights in (bottom-middle) (3.6) and (bottom-right) (3.8). Here $N = 128$ and $M = 64$.

those displayed in Figure 6. It is also evident that the weights provided by (3.6) yield better results than those given by (3.8). Finally, we see that the VBJS weighted ℓ_2 solutions also maintain a high level of accuracy, indicating that accurate solutions can be obtained using the less computationally intensive ℓ_2 regularization. For multi-dimensional problems with many measurement vectors, using ℓ_2 instead of ℓ_1 would provide an enormous reduction in computational cost.

Table 2: Relative reconstruction errors (5.1) for the VBJS method (3.4) with $p = 1$ and 2 using the weights defined in (3.6) and (3.8). Here $N = M = 128$.

	SMV	ℓ_1 (3.6)	ℓ_2 (3.6)	ℓ_1 (3.8)	ℓ_2 (3.8)
Example 1	.0844	.0335	.0393	.4173	.1694
Example 2	.0692	.0536	.0716	.3142	.0959

For further comparison, Table 2 displays the relative error (5.1) for each example, while Table 3 measures the performance at a neighboring grid point to a jump discontinuity, given by

$$|\boldsymbol{f}(x_*) - \hat{\boldsymbol{g}}(x_*)|.$$

For the SMV approximation we choose $\hat{\boldsymbol{y}}$ using (3.10), that is, we consider the best possible solution. In each case we use $J = 10$ measurements where $K = 5$ vectors contain false infromation. Observe that using the VBJS algorithm with the weights in (3.6) with either ℓ_1 or ℓ_2 regularization yields better accuracy than the weights in (3.8), proposed in [1]. These results occur *without* any additional parameter tuning, which is required for both the SMV and VBJS using (3.8). Our method also shows general improvement over the SMV approximation, (2.4), which does not contain any false information.

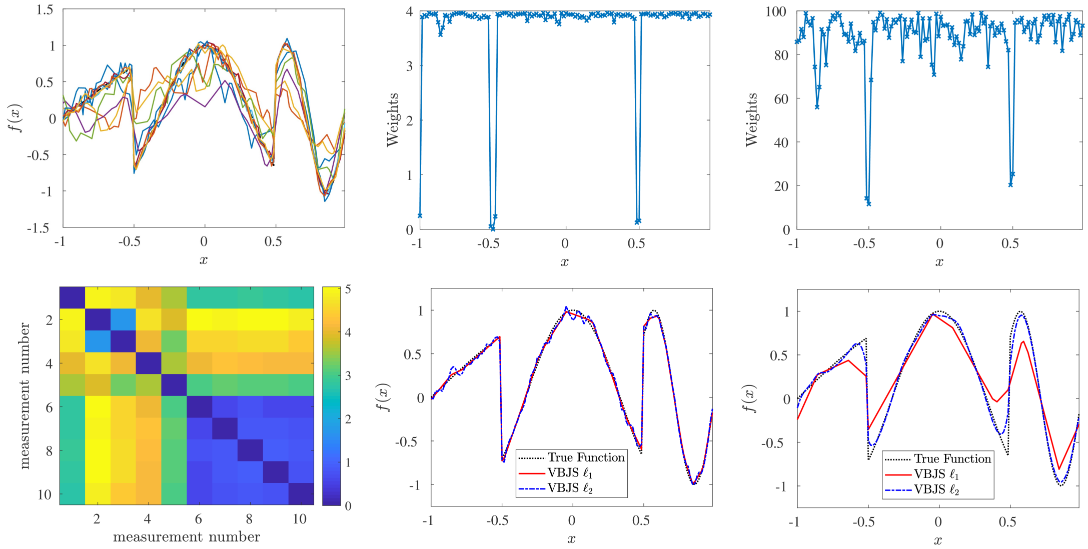

Fig. 8: (top-left) $J = 10$ measurement vectors with $K = 5$ false data acquired using (2.4) with (5.3). Weights proposed in (3.6) (top-middle) and (3.8) from [1] (top-right). (bottom-left) Corresponding distance matrix $\mathcal{D}$ in (3.9). VBJS ℓ_1 and ℓ_2 reconstructions with weights in (bottom-middle) (3.6) and (bottom-right) (3.8). Here $N = M = 128$.

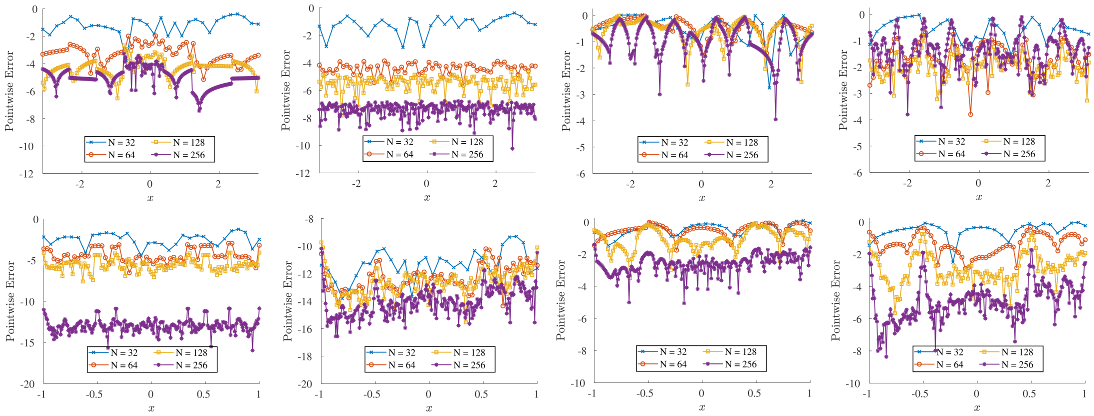

Fig. 9: The pointwise error of the VBJS reconstructions of (top) Example 1 and (bottom) Example 2 for $N = M = 32, 64, 128$ and 256 with $J = 20$ measurements $K = 4$ of which are false with $p = 1$ (left,right-middle) and $p = 2$ (left-middle,right). In (left,left-middle) we use the weights given in (3.6) and in (right,right-middle) we use the weights given in (3.8).

5.3 Case 3: Reconstructing two dimensional images

We now consider reconstructing two dimensional images using the VBJS approach. We note that the original polynomial annihilation edge detection method constructed in [4] was, by design, multi-dimensional. However, as was discussed in [3], for optimization algorithms using ℓ_1 regularization, applying the PA transform dimension by dimension was both more efficient and more accurate

Table 3: Absolute error near a discontinuity for the VBJS method (3.4) with $p = 1$ and 2 using the weights defined in (3.6) and (3.8). In Example 1, $x_* = 1.23$ and in Example 2, $x_* = -.55$. Here $N = M = 128$.

	SMV	ℓ_1 (3.6)	ℓ_2 (3.6)	ℓ_1 (3.8)	ℓ_2 (3.8)
Example 1	.1417	.0048	.0285	.3440	.4276
Example 2	.0131	.0206	.0132	.3738	.1325

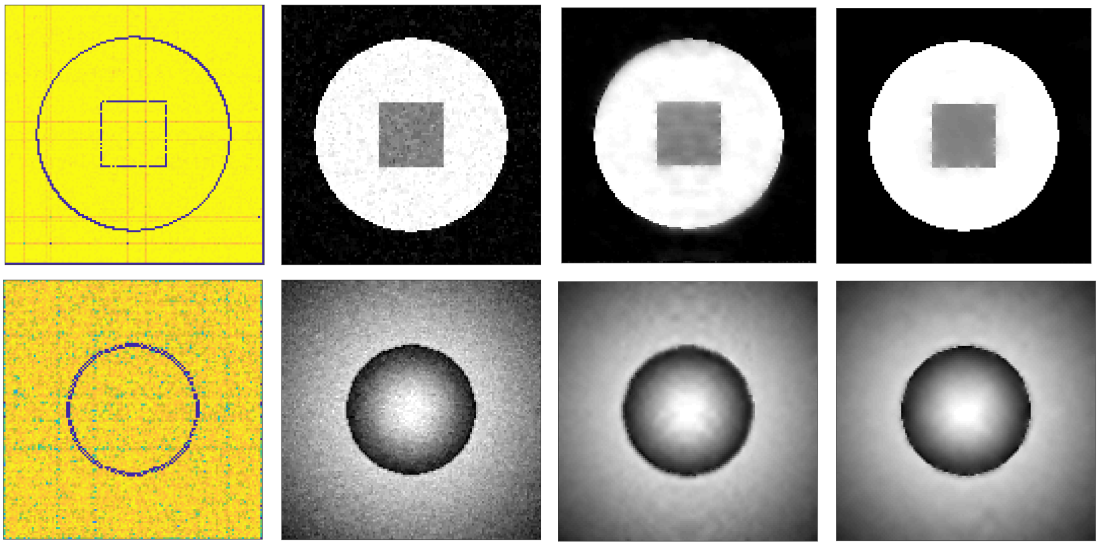

Fig. 10: (left) Weights calculated using (3.6), where the darker shades indicate $w_{i,j} \approx 0$. (middle-left) Reconstruction of a single measurement vector using (3.9) and (2.4). (middle-right) VBJS with $p = 1$. (right) VBJS with $p = 2$. (top) Example 3 reconstruction performed with PA transform of order $m = 1$ in (3.2). (bottom) Example 4 reconstruction performed with PA transform of order $m = 2$ in (3.2).

when on a uniform grid. Therefore, to calculate the weights (3.6) in the two dimensional case, we first calculate the two dimensional edge map for each $j = 1, \cdots, J$ as

$$\mathcal{E}^j = \mathcal{L}\check{\boldsymbol{f}}^j + \check{\boldsymbol{f}}^j\mathcal{L}^T.$$

The columns of each $\mathcal{E}^j$, $j = 1, \cdots, J$, are then stacked on top of each other to form the matrix of J vectors of approximations of some sparse feature of the underlying image, i.e. the two dimensional analogue of (3.2). Continuing as in one dimension, the weights are now calculated according to (3.6) and then reshaped into a matrix $W \in \mathbb{R}^{N\times N}$. The non-zero entries $w_{i,j}$ correspond to the sparse regions of the image, while the entries are approximately zero whenever an edge is assumed to be present. Observe that W is not sparse, so the implementation methods developed in Section 4 is critical for numerical efficiency.

As in the one dimensional case, we consider two examples:

Example 3 Define $f(x, y)$ on $[-1, 1]^2$ as

$$f(x,y) = \begin{cases} 15, & |x|, |y| \leq \frac{1}{4} \\ 20, & |x|, |y| > \frac{1}{4}, \quad \sqrt{x^2+y^2} \leq \frac{3}{4} \\ 10, & \text{else} \end{cases}$$

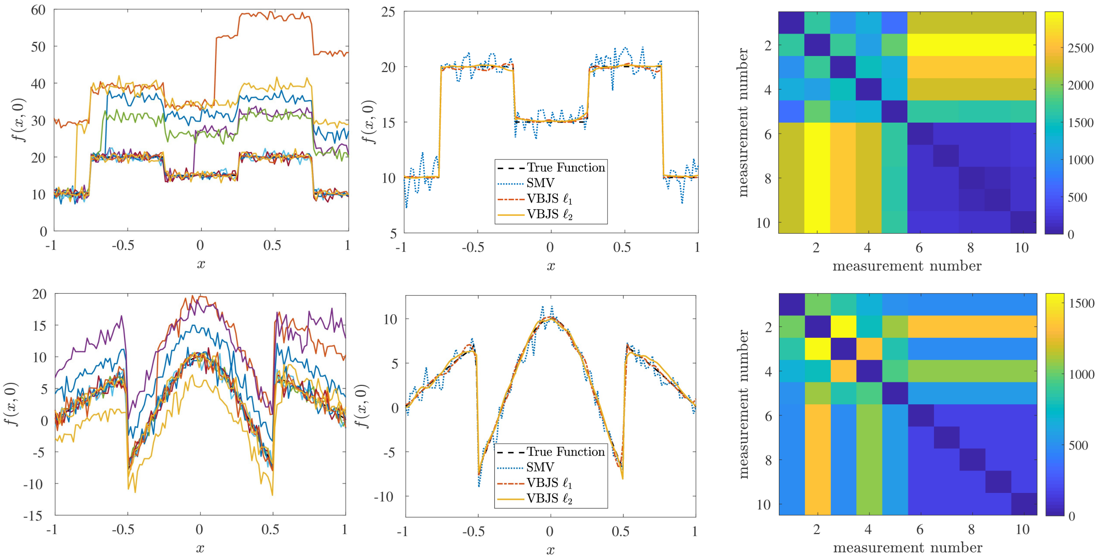

Fig. 11: (top) Results corresponding to Example 3. (bottom) results corresponding to Example 4. (left) Cross sections ($y = 0$) of $J = 10$ measurement vectors with $K = 5$ false data representations. (middle) Cross sections ($y = 0$) of VBJS reconstructions for $p = 1$ and 2 in (3.4) compared to the SMV constructed using (2.4). (right) Data selection matrices $\mathcal{D}$.

Example 4 Define $f(x, y)$ on $[-1, 1]^2$ as

$$f(x,y) = \begin{cases} 10\cos\left(\frac{3\pi}{2}\sqrt{x^2+y^2}\right), & \sqrt{x^2+y^2} \leq \frac{1}{2} \\ 10\cos\left(\frac{\pi}{2}\sqrt{x^2+y^2}\right), & \sqrt{x^2+y^2} > \frac{1}{2} \end{cases}$$

We sample each function $\boldsymbol{f} : \mathbb{R}^{N\times N} \to \mathbb{R}$ on a uniform grid as $f_{i,l} = f(x_i, y_l)$, where

$$x_i = -1 + \frac{2}{N}(i-1), \qquad y_l = -1 + \frac{2}{N}(l-1),$$

for each $i, l = 1, \cdots, N$. In (2.2), $\boldsymbol{A} : \mathbb{R}^{N\times N} \to \mathbb{C}^{N\times N}$ is defined to be the normalized, two dimensional discrete Fourier transform operator so that $\boldsymbol{A}^* = \boldsymbol{A}^{-1}$, and $\boldsymbol{\eta}^j$ is zero mean complex Gaussian noise with .5 variance for all $j = 1, ..., J$. As in the one dimensional case we use (2.4) to construct each $\check{\boldsymbol{f}}^j$. Because of the piecewise constant nature of Example 3 we apply the PA transform with order $m = 1$. Similarly, for Example 4 we use $m = 2$. We note that it is possible to use $m > 2$, but in this case, because of the noise, the higher order polynomial approximation leads to overfitting. For each data vector the regularization parameter μ is sampled from a uniform distribution on $[0, 10]$.

Figure 10 displays the result of applying VBJS with ℓ_1 (middle-right) and ℓ_2 (right) to Examples 3 and 4. For both examples we use $J = 10$ measurement vectors where $K = 5$ falsely represent the underlying function. (The corresponding measurement selection matrices (3.9) are shown in Figure 11(right).) Figure 10(middle-left) shows the the SMV results on the measurement vector selected by (3.9) calculated using (2.4). It is evident in both examples that the VBJS technique with either $p = 1$ or 2 in (3.4) leads to improved visualization over the standard SMV reconstruction, even when the standard SMV uses the “best” initialization as determined by (3.9). This result is confirmed in Figure 11, where we compare the corresponding one-dimensional cross sections at $y = 0$.

6 Concluding Remarks

In this investigation we proposed a modification to the variance based joint sparsity technique (VBJS), introduced in [1], in both the weighting vector and in the choice of reconstruction vector. Our adaptation is especially critical when some data vectors contain false measurements. We additionally proved that the ADMM algorithm could be successfully used for the weighted ℓ_1 case, and moreover, that for our choice of weights in (3.6), *no* extra parameter tuning is needed to achieve high accuracy and fast convergence. Hence our method is robust and suitable to a wide range of problems. We also presented a corresponding gradient descent method for the weighted ℓ_2 case.

Our numerical results demonstrate that the VBJS method with the weights designed in (3.6) yields improved accuracy and robustness over the single measurement vector case, the classical $\ell_{2,1}$ JS method, and the original VBJS method proposed in [1]. By including an optimal data vector selection step, we are able to obtain high accuracy and good sparse signal recovery even when a subset of the given measurement data misrepresents the underlying function. Furthermore, using the weighted ℓ_2 norm also yields good results and is much more cost effective than the weighted ℓ_1 reconstructions.

In future investigations we will conduct a thorough convergence analysis of the VBJS method, in particular to establish rigorous results for the weighted ℓ_2 case. We will also parallelize our algorithm so that we may test it on synthetic aperture radar automatic target recognition problems, where current algorithms fail when obstructions are added to (or taken out of) imaging scenes. Because our method is non-parametric, autonomy will be maintained. This framework also lends itself to data fusion problems, where measurements of a scene are obtained through multiple imaging techniques and must be combined to yield optimal results. Finally, the VBJS can potentially be used in numerical partial differential equation solvers, in particular to develop predictor-corrector methods for equations that exhibit singularities or for which shock discontinuities evolve.

References

1. Adcock, B., Gelb, A., Song, G., Sui, Y.: Joint sparse recovery based on variances. SIAM Journal on Scientific Computing (submitted)
2. Ao, D., Wang, R., Hu, C., Li, Y.: A sparse SAR imaging method based on multiple measurement vectors model. Remote Sensing **9**(3), 297 (2017)
3. Archibald, R., Gelb, A., Platte, R.B.: Image reconstruction from undersampled Fourier data using the polynomial annihilation transform. Journal of Scientific Computing **67**(2), 432–452 (2016)
4. Archibald, R., Gelb, A., Yoon, J.: Polynomial fitting for edge detection in irregularly sampled signals and images. SIAM Journal on Numerical Analysis **43**(1), 259–279 (2005)
5. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA journal of numerical analysis **8**(1), 141–148 (1988)
6. Bauschke, H.H., Combettes, P.L.: Convex analysis and monotone operator theory in Hilbert spaces, vol. 408. Springer (2011)
7. Beck, A.: Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB. SIAM (2014)
8. Candes, E.J., Wakin, M.B., Boyd, S.P.: Enhancing sparsity by reweighted ℓ_1 minimization. Journal of Fourier analysis and applications **14**(5), 877–905 (2008)
9. Chan, T., Marquina, A., Mulet, P.: High-order total variation-based image restoration. SIAM Journal on Scientific Computing **22**(2), 503–516 (2000)
10. Chartrand, R., Yin, W.: Iteratively reweighted algorithms for compressive sensing. In: Acoustics, speech and signal processing, 2008. ICASSP 2008. IEEE international conference on, pp. 3869–3872. IEEE (2008)
11. Chen, J., Huo, X.: Theoretical results on sparse representations of multiple-measurement vectors. IEEE Transactions on Signal Processing **54**(12), 4634–4643 (2006)
12. Cotter, S.F., Rao, B.D., Engan, K., Kreutz-Delgado, K.: Sparse solutions to linear inverse problems with multiple measurement vectors. IEEE Transactions on Signal Processing **53**(7), 2477–2488 (2005)
13. Daubechies, I., DeVore, R., Fornasier, M., Güntürk, C.S.: Iteratively reweighted least squares minimization for sparse recovery. Communications on Pure and Applied Mathematics **63**(1), 1–38 (2010)
14. Deng, W., Yin, W., Zhang, Y.: Group sparse optimization by alternating direction method. Tech. rep., Rice University, Houston, TX, Department of Computationa and Applied Mathematics (2012)

15. Denker, D., Gelb, A.: Edge detection of piecewise smooth functions from undersampled Fourier data using variance signatures. SIAM Journal on Scientific Computing **39**(2), A559–A592 (2017)
16. Eldar, Y.C., Mishali, M.: Robust recovery of signals from a structured union of subspaces. IEEE Transactions on Information Theory **55**(11), 5302–5316 (2009)
17. Eldar, Y.C., Rauhut, H.: Average case analysis of multichannel sparse recovery using convex relaxation. IEEE Transactions on Information Theory **56**(1), 505–519 (2010)
18. Fu, W., Li, S., Fang, L., Kang, X., Benediktsson, J.A.: Hyperspectral image classification via shape-adaptive joint sparse representation. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing **9**(2), 556–567 (2016)
19. Glowinski, R., Le Tallec, P.: Augmented Lagrangian and operator-splitting methods in nonlinear mechanics. SIAM (1989)
20. Keydel, E.R., Lee, S.W., Moore, J.T.: MSTAR extended operating conditions: A tutorial. In: Aerospace/Defense Sensing and Controls, pp. 228–242. International Society for Optics and Photonics (1996)
21. Leviatan, D., Temlyakov, V.N.: Simultaneous approximation by greedy algorithms. Advances in Computational Mathematics **25**(1), 73–90 (2006)
22. Li, C.: An efficient algorithm for total variation regularization with applications to the single pixel camera and compressive sensing. Ph.D. thesis, Citeseer (2009)
23. Liu, L., Esmalifalak, M., Ding, Q., Emesih, V.A., Han, Z.: Detecting false data injection attacks on power grid by sparse optimization. IEEE Transactions on Smart Grid **5**(2), 612–621 (2014)
24. Liu, Q.y., Zhang, Q., Gu, F.f., Chen, Y.c., Kang, L., Qu, X.y.: Downward-looking linear array 3D SAR imaging based on multiple measurement vectors model and continuous compressive sensing. Journal of Sensors **2017** (2017)
25. Liu, Y., Ma, J., Fan, Y., Liang, Z.: Adaptive-weighted total variation minimization for sparse data toward low-dose x-ray computed tomography image reconstruction. Physics in medicine and biology **57**(23), 7923 (2012)
26. Mishali, M., Eldar, Y.C.: Reduce and boost: Recovering arbitrary sets of jointly sparse vectors. IEEE Transactions on Signal Processing **56**(10), 4692–4702 (2008)
27. Monajemi, H., Jafarpour, S., Gavish, M., Donoho, D.L., Ambikasaran, S., Bacallado, S., Bharadia, D., Chen, Y., Choi, Y., Chowdhury, M., et al.: Deterministic matrices matching the compressed sensing phase transitions of Gaussian random matrices. Proceedings of the National Academy of Sciences **110**(4), 1181–1186 (2013)
28. Niculescu, C., Persson, L.E.: Convex functions and their applications: a contemporary approach. Springer Science & Business Media (2006)
29. Parikh, N., Boyd, S., et al.: Proximal algorithms. Foundations and Trends in Optimization **1**(3), 127–239 (2014)
30. Sanders, T., Gelb, A., Platte, R.B.: Composite SAR imaging using sequential joint sparsity. Journal of Computational Physics **338**, 357–370 (2017)
31. Singh, A., Dandapat, S.: Weighted mixed-norm minimization based joint compressed sensing recovery of multi-channel electrocardiogram signals. Computers & Electrical Engineering **53**, 203–218 (2016)
32. Steffens, C., Pesavento, M., Pfetsch, M.E.: A compact formulation for the l21 mixed-norm minimization problem. In: Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on, pp. 4730–4734. IEEE (2017)
33. Tropp, J.A.: Algorithms for simultaneous sparse approximation. Part II: Convex relaxation. Signal Processing **86**(3), 589–602 (2006)
34. Tropp, J.A., Gilbert, A.C., Strauss, M.J.: Simultaneous sparse approximation via greedy pursuit. In: Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on, vol. 5, pp. v–721. IEEE (2005)
35. Tropp, J.A., Gilbert, A.C., Strauss, M.J.: Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit. Signal Processing **86**(3), 572–588 (2006)
36. Wang, Y., Yang, J., Yin, W., Zhang, Y.: A new alternating minimization algorithm for total variation image reconstruction. SIAM Journal on Imaging Sciences **1**(3), 248–272 (2008)
37. Wipf, D.P., Rao, B.D.: An empirical bayesian strategy for solving the simultaneous sparse approximation problem. IEEE Transactions on Signal Processing **55**(7), 3704–3716 (2007)
38. Wright, S., Nocedal, J.: Numerical optimization. Springer Science **35**, 67–68 (1999)
39. Xie, W., Deng, Y., Wang, K., Yang, X., Luo, Q.: Reweighted l1 regularization for restraining artifacts in fmt reconstruction images with limited measurements. Optics letters **39**(14), 4148–4151 (2014)
40. Yang, Z., Xie, L.: Enhancing sparsity and resolution via reweighted atomic norm minimization. IEEE Transactions on Signal Processing **64**(4), 995–1006 (2016)
41. Ye, F., Luo, H., Lu, S., Zhang, L.: Statistical en-route filtering of injected false data in sensor networks. IEEE Journal on Selected Areas in Communications **23**(4), 839–850 (2005)
42. Zhang, Y.: Users guide for YALL1: Your algorithms for l1 optimization. Technique report pp. 09–17 (2009)
43. Zhao, B., Fei-Fei, L., Xing, E.P.: Online detection of unusual events in videos via dynamic sparse coding. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 3313–3320. IEEE (2011)
44. Zheng, C., Li, G., Liu, Y., Wang, X.: Subspace weighted l21 minimization for sparse signal recovery. EURASIP Journal on Advances in Signal Processing **2012**(1), 98 (2012)
45. Zhou, F., Wu, R., Xing, M., Bao, Z.: Approach for single channel SAR ground moving target imaging and motion parameter estimation. IET Radar, Sonar & Navigation **1**(1), 59–66 (2007)

A Proof of Lemma 1

Proof [Lemma 1] Following the technique described in [22] for the non-weighted, one-dimensional case, let $x \in \mathbb{R}^{N \times N}$ and $w_{i,j} \geq 0$ for all $i, j = 1, ..., N$. We drop the *vec* notation for simplicity.

Define the objective function $H : \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times N}$ as

$$H(x) := ||x||_{1,w} - \nu^T(y - x) + \frac{\beta}{2}||y - x||_2^2. \tag{A.1}$$

To show $H(x)$ is convex, we first observe that for $\alpha \in (0, 1)$ and $p, q \in \mathbb{R}^{N \times N}$, we have

$$\begin{aligned}
&||y - \alpha p - (1-\alpha)q||_2^2 - \left(\alpha||y-p||_2^2 + (1-\alpha)||y-q||_2^2\right) \\
&= (y - \alpha p - (1-\alpha)q)^T\left(y - \alpha p - (1-\alpha)q\right) - \Big(\alpha(y-p)^T(y-p) + (1-\alpha)(y-q)^T(y-q)\Big) \\
&= \alpha(\alpha - 1)\left(p^Tp - p^Tq - q^Tp + q^Tq\right) \\
&= \alpha(\alpha-1)||p-q||_2^2 \\
&\leq 0.
\end{aligned} \tag{A.2}$$

Applying (A.2) to H yields

$$\begin{aligned}
&H(\alpha p + (1-\alpha)q) - (\alpha H(p) + (1-\alpha)H(q)) \\
&= ||\alpha p + (1-\alpha)q||_{1,w} - \nu^T\left(y - (\alpha p + (1-\alpha)q)\right) + \frac{\beta}{2}||y - (\alpha p + (1-\alpha)q)||_2^2 \\
&- \alpha||p||_{1,w} - (1-\alpha)||q||_{1,w} + \alpha\nu^T(y-p) + (1-\alpha)\nu^T(y-q) - \frac{\beta\alpha}{2}||y-p||_2^2 - \frac{\beta(1-\alpha)}{2}||y-q||_2^2 \\
&\leq \frac{\beta}{2}||y - (\alpha p + (1-\alpha)q)||_2^2 - \frac{\beta\alpha}{2}||y-p||_2^2 - \frac{\beta(1-\alpha)}{2}||y-q||_2^2 \\
&= \frac{\beta}{2}\alpha(\alpha-1)||p-q||_2^2 \\
&\leq 0.
\end{aligned} \tag{A.3}$$

Therefore H is convex. For $p \neq q$, H is *strictly/strongly convex* and thus coercive [6, 7, 28]. Hence there exists at least one solution $\hat{x}$ of (4.6), [38].

The subdifferential of $f(x) = ||x||_{1,w}$ is given element-wise as

$$(\partial_x f(x))_{i,j} = \begin{cases} \text{sign}(x_{i,j})w_{i,j}, & x_{i,j} \neq 0 \\ \{h; |h| \leq w_{i,j},\ h \in \mathbb{R}\}, & \text{otherwise,} \end{cases} \tag{A.4}$$

where the origin is required to be included according to the optimality condition for convex problems. According to (A.4), to minimize (A.1), each component $\hat{x}_{i,j}$, $i, j = 1, ..., N$, must satisfy

$$\begin{cases} \text{sign}(\hat{x}_{i,j})w_{i,j} + \beta(\hat{x}_{i,j} - y_{i,j}) + \nu_{i,j} = 0, & x_{i,j} \neq 0 \\ |v_{i,j} - \beta y_{i,j}| \leq w_{i,j}, & \text{otherwise.} \end{cases} \tag{A.5}$$

If $\hat{x}_{i,j} \neq 0$, (A.5) yields

$$\frac{w_{i,j}}{\beta}\text{sign}(\hat{x}_{i,j}) + \hat{x}_{i,j} = y_{i,j} - \frac{\nu_{i,j}}{\beta}. \tag{A.6}$$

Since $w_{i,j}/\beta > 0$, (A.6) implies

$$\frac{w_{i,j}}{\beta} + |\hat{x}_{i,j}| = |y_{i,j} - \frac{\nu_{i,j}}{\beta}|. \tag{A.7}$$

Combining (A.6) and (A.7) gives

$$\begin{aligned}
\text{sign}(\hat{x}_{i,j}) &= \frac{\text{sign}(\hat{x}_{i,j})|\hat{x}_{i,j}| + \text{sign}(\hat{x}_{i,j})w_{i,j}/\beta}{|\hat{x}_{i,j}| + w_{i,j}/\beta} = \frac{\hat{x}_{i,j} + \text{sign}(\hat{x}_{i,j})w_{i,j}/\beta}{|\hat{x}_{i,j}| + w_{i,j}/\beta} \\
&= \frac{y_{i,j} - \nu_{i,j}/\beta}{|y_{i,j} - \nu_{i,j}/\beta|} = \text{sign}\left(y_{i,j} - \frac{\nu_{i,j}}{\beta}\right)
\end{aligned} \tag{A.8}$$

Thus, for $\hat{x}_{i,j} \neq 0$, we have

$$\hat{x}_{i,j} = |\hat{x}_{i,j}|\text{sign}(\hat{x}_{i,j}) = \left(|y_{i,j} - \frac{\nu_{i,j}}{\beta}| - \frac{w_{i,j}}{\beta}\right) \text{sign}\left(y_{i,j} - \frac{\nu_{i,j}}{\beta}\right), \tag{A.9}$$

where we have used (A.7) and (A.8) in the result.

Conversely, we now show that $\hat{x}_{i,j} = 0$ if and only if

$$|y_{i,j} - \frac{\nu_{i,j}}{\beta}| \leq \frac{w_{i,j}}{\beta}. \tag{A.10}$$

First assume that $\hat{x}_{i,j} = 0$. Then (A.10) follows from (A.5) since $\beta > 0$.

Now assume (A.10) holds for some $\hat{x}_{i,j} \neq 0$. By (A.5), $\hat{x}_{i,j}$ satisfies (A.7). Hence

$$|\hat{x}_{i,j}| = |y_{i,j} - \frac{\nu_{i,j}}{\beta}| - \frac{w_{i,j}}{\beta} \leq 0$$

which only holds for $\hat{x}_{i,j} = 0$. Hence by contradiction, $\hat{x}_{i,j} = 0$. Combining (A.10) with (A.9) yields

$$\hat{x}_{i,j} = \max\left\{|y_{i,j} - \frac{\nu_{i,j}}{\beta}| - \frac{w_{i,j}}{\beta}, 0\right\} \text{sign}\left(y_{i,j} - \frac{\nu_{i,j}}{\beta}\right).$$

which is equivalent to (4.7) in matrix form.