Why Simple Shrinkage Is Still Relevant for Redundant Representations?

Michael Elad, *Member, IEEE*

***Abstract*—Shrinkage is a well known and appealing denoising technique, introduced originally by Donoho and Johnstone in 1994. The use of shrinkage for denoising is known to be optimal for Gaussian white noise, provided that the sparsity on the signal's representation is enforced using a unitary transform. Still, shrinkage is also practiced with nonunitary, and even redundant representations, typically leading to very satisfactory results. In this correspondence we shed some light on this behavior. The main argument in this work is that such simple shrinkage could be interpreted as the first iteration of an algorithm that solves the basis pursuit denoising (BPDN) problem. While the desired solution of BPDN is hard to obtain in general, we develop a simple iterative procedure for the BPDN minimization that amounts to stepwise shrinkage. We demonstrate how the simple shrinkage emerges as the first iteration of this novel algorithm. Furthermore, we show how shrinkage can be iterated, turning into an effective algorithm that minimizes the BPDN via simple shrinkage steps, in order to further strengthen the denoising effect.**

***Index Terms*—Basis pursuit, denoising, frame, overcomplete, redundant, sparse representation, shrinkage, thresholding.**

I. Introduction

A commonly practiced approach toward the removal of additive Gaussian white noise from a signal $\boldsymbol{y}$ is the minimization of the function

$$f(\boldsymbol{x}) = \frac{1}{2} \cdot \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \cdot P_r\{\boldsymbol{x}\}. \tag{1}$$

This expression can be shown to emerge from a Baysian point of view, when deploying the maximum *a posteriori* probability (MAP) estimation [4]. The first term is commonly referred to as the log-likelihood, describing the relation between the desired (clean) signal $\boldsymbol{x} \in \mathbb{R}^N$ and a noisy version of it $\boldsymbol{y} \in \mathbb{R}^N$. We assume the model $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{v}$, with $\boldsymbol{v} \in \mathbb{R}^N$ a Gaussian zero-mean white noise.[1] The term $P_r\{\boldsymbol{x}\}$ stands for a prior posed on the unknown signal $\boldsymbol{x}$; numerous such expressions have been used for various signal types, as can be found in the literature. Among the popular methods that are now considered as classic methods in signal processing, we mention the following.

1. Energy minimization $P_r\{\boldsymbol{x}\} = \|\boldsymbol{x}\|_2^2$—the simplest to handle, leading to $\hat{\boldsymbol{x}} = \boldsymbol{y}/(1+\lambda)$, which could be interpreted as a trivial shrinkage operation, as the signal is attenuated as part of the restoration.
2. Smoothness penalty $P_r\{\boldsymbol{x}\} = \|\boldsymbol{L}\boldsymbol{x}\|_2^2$ using the Laplacian operator, leading to the Wiener filter $\hat{\boldsymbol{x}} = (\boldsymbol{I} + \lambda \boldsymbol{L}^T\boldsymbol{L})^{-1}\boldsymbol{y}$.
3. Total variation (TV) handles smoothness while allowing for sharp edges$P_r\{\boldsymbol{x}\} = \||\nabla \boldsymbol{x}\|\|_1$. This is done by an iterative update rule of the form $\hat{\boldsymbol{x}}_{k+1} = (1-\mu)\hat{\boldsymbol{x}}_k + \mu\boldsymbol{y} - \mu\lambda\nabla^T(\nabla\boldsymbol{x}_k/|\nabla\boldsymbol{x}_k|)$ [1].
4. Scalar entropy of $\boldsymbol{x}$, defined by $P_r\{\boldsymbol{x}\} = -\boldsymbol{x}^T\log(\boldsymbol{x})$. This promotes nonuniformity in $\boldsymbol{x}$, while assuming nonnegative values only. Here again one cannot have a closed-form solution, and an iterative procedure of the form $\hat{\boldsymbol{x}}_{k+1} = \exp\{\boldsymbol{1} - (\boldsymbol{x}_k - \boldsymbol{y})/\lambda\}$ can be used [2].
5. Sparsity of the unknown signal with respect to its transformed representation $P_r\{\boldsymbol{x}\} = \|\boldsymbol{T}\boldsymbol{x}\|_1$. Again, iterative solver should be applied in general, giving an update of the form $\hat{\boldsymbol{x}}_{k+1} = (1-\mu)\hat{\boldsymbol{x}}_k + \mu\boldsymbol{y} - \mu\lambda\cdot\boldsymbol{T}^T\mathrm{sign}(\boldsymbol{T}\boldsymbol{x}_k)$ [25].

Formulations of the denoising problem as in (1) are commonly used, and are the basis for many more general inverse problems. Using priors such as TV and other robust methods, the above opens an opportunity to the study of nonlinear filtering, typically formulated as partial differential equations (PDE). As we have seen in the above list (items 3, 4, and 5), those are typically handled by iterative numerical solvers, that are characterized as slow and cumbersome.[2]

In parallel to the progress made in the PDE directions, Donoho and Johnstone pioneered a wavelet-based signal denoising algorithm in line with the above list of priors (item 5). In a sequence of publications, they advocated the use of sparsity of the wavelet coefficients as a driving force in recovering the desired signal [5]–[10]. Later work in [11]–[13] simplified these ideas and related them to the MAP formulation as presented above, using the prior $P_r\{\boldsymbol{x}\} = \|\boldsymbol{W}\boldsymbol{x}\|_p$, with a unitary wavelet transform matrix[3] $\boldsymbol{W} \in \mathbb{R}^{N\times N}$, and $0 \le p \le 1$.

Interestingly, using such a prior in (1) leads to a *simple closed-form solution, known as shrinkage* (for the sake of completeness of the discussion, Section II establishes that). This solution amounts to a wavelet transform on the noisy signal, a lookup table (LUT)[4] function on the coefficients (that depends on p), $\mathcal{S}\{\boldsymbol{W}\boldsymbol{y}\}$, and an inverse wavelet transform to produce the outcome $\hat{\boldsymbol{x}}$. Fig. 1 presents this appealing and simple algorithm. Such a direct solution stands as a refreshing alternative to the iterative and slow methods mentioned above.

In Section II we follow [5]–[10], [12], [13] and present the analysis that shows how the shrinkage algorithm becomes indeed the optimal solver of (1). This optimality depends strongly on the ℓ^2-norm used in evaluating the distance $\boldsymbol{x} - \boldsymbol{y}$, and this has direct roots in the white Gaussianity assumptions on the noise. Also, crucial to the optimality of this method is the orthogonality of $\boldsymbol{W}$.

A new trend in recent years has been the use of overcomplete transforms, replacing the traditional unitary ones—see [14]–[24], [28]–[32] for representative works. This trend was partly motivated by the growing realization that orthogonal wavelets are weak in describing the singularities found in images. The sources of this weakness are the loss of shift-invariance due to mandatory decimation, and the separability forced, which implies a lack of directional treatment. Another driving force in the introduction of redundant representations is the sparsity it can provide, which many applications find desirable.

Manuscript received January 24, 2005; revised December 25, 2005. This work was supported by the Technion's VPR funds, and the Gabriel and Matilda Brent Trust.

The authors is with the Department of Computer Science, Technion–Israel Institute of Technology, Technion City, Haifa 32000 Israel (e-mail: elad@cs.technion.ac.il).

Communicated by V. A. Vaishampayan, Associate Editor At-Large.

Color versions of Figs. 1–8 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2006.885522

[1]In most papers, when the noise characteristics are introduced, an accompanying "no-restriction" statement is added, suggesting that other models can be easily handled similarly. In this work, this is not the case. The assumptions about the noise being Gaussian and white are crucial and cannot be replaced with alternatives such as colored noise or different distributions. In other words, shrinkage, as will be discussed here, is tightly coupled with these assumptions.

[2]Speedup algorithms have been proposed (e.g., the AOS method by Weickert [3] or the bilateral filter [4]), but those are still quite involved compared to simplicity of shrinkage—see its description next.

[3]As will be shown hereafter, it is the unitarity of $\boldsymbol{W}$ that makes this choice so appealing, whereas the fact that this is specifically chosen as the wavelet transform has to do with the type of signals handled.

[4]LUT means a one-dimensional memoryless function that operates on each wavelet coefficient in the same way. Assuming that the input is discretized to a finite number of bits, such operation can be done by considering the incoming value as an address to a pre-organized table of output values, and thus the name lookup table.

0018-9448/$20.00 © 2006 IEEE

In these methods, the transform is defined via a nonsquare full-rank matrix $\boldsymbol{T} \in \mathbb{R}^{L\times N}$, with $L > N$. Such redundant methods, like the un-decimated wavelet transform, curvelet, contourlet, steerable-wavelet, and more, were shown to be more effective in representing images, and other signal types. It is often assumed that $\boldsymbol{T}$ is a tight frame, implying that $\alpha\boldsymbol{T}^T\boldsymbol{T} = \boldsymbol{I}$. In such a case, the adjoint $\boldsymbol{T}^T$ stands for the Moore–Penrose pseudoinverse transform, up to the constant α.

Given a noisy signal $\boldsymbol{y}$, one can still follow the shrinkage procedure, by computing the forward transform $\boldsymbol{T}\boldsymbol{y}$, putting the coefficients through a shrinkage LUT operation $\mathcal{S}\{\boldsymbol{T}\boldsymbol{y}\}$, and finally applying the inverse transform[5] to obtain the denoised outcome, $\boldsymbol{T}^+\mathcal{S}\{\boldsymbol{T}\boldsymbol{y}\}$. Will this be the solver of (1) when using the prior $P_r\{\boldsymbol{x}\} = \|\boldsymbol{T}\boldsymbol{x}\|_p$? The answer is negative. As we have said before, the orthogonality of the transform plays a crucial role in the construction of the shrinkage as an optimal procedure. Still, shrinkage is practiced quite often with nonunitary, and even redundant representations, typically leading to results better than in the nonredundant cases—see [15]–[24] for representative examples. Naturally, we should wonder why this is the case.

In this correspondence, we shed some light on this behavior. Our main argument, as will be composed in Section III, is that such a shrinkage could be interpreted as the first iteration of a converging algorithm that solves the basis pursuit denoising (BPDN) problem [25]. The BPDN forms a similar problem to the one posed in (1), replacing the analysis prior with a generative one. While the desired solution of BPDN is hard to obtain in general, a simple iterative procedure that amounts to stepwise shrinkage can be employed with quite successful performance. Thus, beyond showing that shrinkage has justified roots in solid denoising methodology, we show how shrinkage can be iterated in a simple form, to further strengthen the denoising effect. As a by-product, we get an effective algorithm that minimizes the BPDN functional via simple shrinkage steps.

This correspondence is organized as follows: In Section II, we show how shrinkage emerges as an optimal solution for the prior $P_r\{\boldsymbol{x}\} = \|\boldsymbol{W}\boldsymbol{x}\|_p$ with *any* unitary matrix $\boldsymbol{W}$. This is a well-known result, belonging now to the classics of signal processing. We bring it here for completeness, and to set the stage for the redundant representation case, whose analysis follows. In Section III, we generalize the prior to use redundant transforms. We show first the BPDN formulation as the desired denoising method, and then show how shrinkage could approximate it. In Section IV, we present some experimental results to support the claims made in Section III.

II. Shrinkage for Unitary Transforms

In this section, we consider the denoising problem with a general additive prior that utilizes an orthonormal matrix $\boldsymbol{W}$. The ideas presented in this section can be traced back to [5], and also found in [6]–[13], although they may appear different. We intend to minimize

$$f(\boldsymbol{x}) = \frac{1}{2}\cdot\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \lambda\cdot\boldsymbol{1}^T\cdot\rho\{\boldsymbol{W}\boldsymbol{x}\}. \tag{2}$$

In our notations, $\rho(\cdot)$ is an arbitrary scalar function. When applied to a vector, it is producing an output vector obtained by operating on each entry independently. The vector $\boldsymbol{1}\in\mathbb{R}^N$ is a vector of ones. Thus, the prior term amounts to

$$\boldsymbol{1}^T\cdot\rho\{\boldsymbol{W}\boldsymbol{x}\} = \sum_{n=1}^{N}\rho\{[\boldsymbol{W}\boldsymbol{x}]_n\} \tag{3}$$

and thus the name "additive prior."

[5]There are infinitely many ways to define the inverse, and in most cases the Moore–Penrose pseudoinverse is practiced.

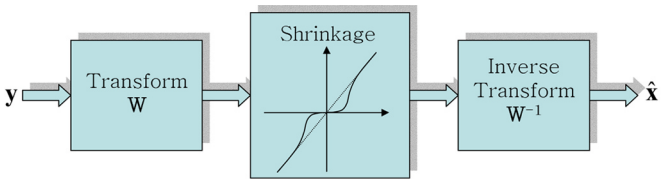

Fig. 1. A block diagram of the shrinkage method for denoising.

We typically assume (for convenience, and those assumptions can be relaxed) that $\rho(z)$ is symmetric ($\rho(z) = \rho(-z)$) and monotonic nondecreasing in the range $z > 0$, implying $\rho'(z) \geq 0$. As examples, choosing $\rho(z) = |z|^2$ leads to $P_r\{\boldsymbol{x}\} = \|\boldsymbol{W}\boldsymbol{x}\|_2^2$, choosing $\rho(z) = |z|$ gives the ℓ^1 alternative—$P_r\{\boldsymbol{x}\} = \|\boldsymbol{W}\boldsymbol{x}\|_1$, and $\rho(z) = |z|^p$ leads to the ℓ^p option—$P_r\{\boldsymbol{x}\} = \|\boldsymbol{W}\boldsymbol{x}\|_p^p$, all being special cases of this general additive form.

Defining $\boldsymbol{x}_w = \boldsymbol{W}\boldsymbol{x}$ and similarly $\boldsymbol{y}_w = \boldsymbol{W}\boldsymbol{y}$, the function $f(\boldsymbol{x})$ in (2) can be rearranged to become a function of $\boldsymbol{x}_w$

$$f(\boldsymbol{x}_w) = \frac{1}{2}\cdot\|\boldsymbol{W}^{-1}(\boldsymbol{x}_w-\boldsymbol{y}_w)\|_2^2 + \lambda\cdot\boldsymbol{1}^T\cdot\rho\{\boldsymbol{x}_w\}. \tag{4}$$

Exploiting the unitary invariance property of the ℓ^2-norm,[6] we can discard of the multiplication by $\boldsymbol{W}^{-1}$ in the first term, and obtain

$$\begin{aligned} f(\boldsymbol{x}_w) &= \frac{1}{2}\cdot\|(\boldsymbol{x}_w-\boldsymbol{y}_w)\|_2^2 + \lambda\cdot\boldsymbol{1}^T\cdot\rho\{\boldsymbol{x}_w\} \\ &= \sum_{n=1}^{N}\left[\frac{1}{2}(x_w(n)-y_w(n))^2 + \lambda\rho(x_w(n))\right] \\ &= f(x_w(1), x_w(2), \ldots, x_w(N)). \end{aligned} \tag{5}$$

A consequence of the above simple manipulation is the fact that the original problem with respect to the unknown $\boldsymbol{x}_w$ is now decoupled. It can be solved independently for each unknown entry $x_w(n)$ as a scalar optimization procedure, and this is far easier than the N-dimensional optimization task we embarked from.

Note that the first stage to be done in solving (5) is to transform the input signal $\boldsymbol{y}$ to obtain $\boldsymbol{y}_w = \boldsymbol{W}\boldsymbol{y}$. This aligns with the block diagram described in Fig. 1. Once this is done, we face N one-dimensional (1-D) optimization problems of the general form

$$z_{\text{opt}} = \arg\min_z g(z,a) = \arg\min_z \frac{1}{2}(z-a)^2 + \lambda\rho(z) \tag{6}$$

with a and λ assumed known. The solution is an antisymmetric LUT curve of the form $z_{\text{opt}} = \psi(a)$. Indeed, for an arbitrary a we have

$$\begin{aligned} g(z,-a) &= \frac{1}{2}(z-(-a))^2 + \lambda\rho(z) \\ &= \frac{1}{2}((-z)-a)^2 + \lambda\rho(-z) = g(-z,a) \end{aligned}$$

where we have exploited the fact that ρ is symmetric. Thus, if for some $a > 0$ we have that $z_{\text{opt}} = \psi(a)$, then necessarily, $\psi(-a) = -z_{\text{opt}} =$

[6]This is where we exploit both the orthogonality of $\boldsymbol{W}$ and the white Gaussianity of the noise, as promised.

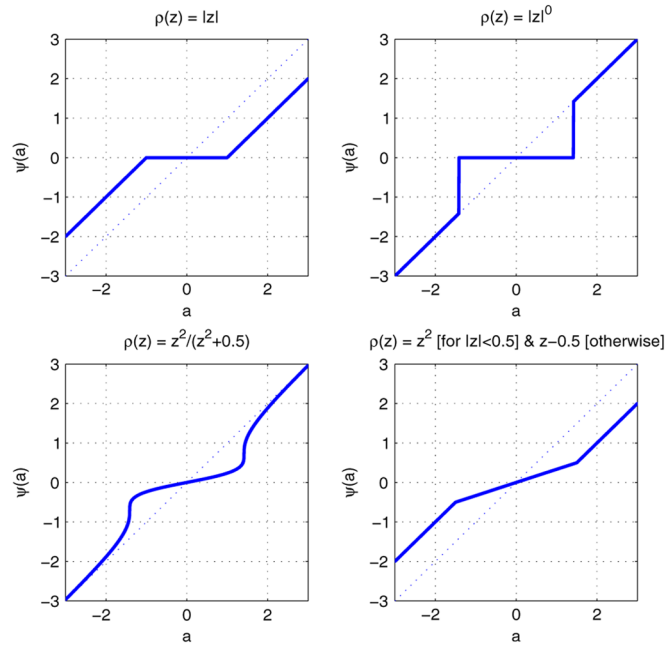

Fig. 2. Several examples of the shrinkage LUT curves $\psi_\rho(a)$. In all these cases, $\lambda = 1$.

$-\psi(a)$, being antisymmetric as claimed. Thus, we can restrict our analysis to positive inputs $a \geq 0$. Assuming that ρ is continuously differentiable,[7] $\rho(\cdot) \in \mathbb{C}^1$, the solution should satisfy

$$z_{\text{opt}} = a - \lambda \rho'(z_{\text{opt}}). \tag{7}$$

This implicit equation can give the curve $\psi(a)$. An alternative approach to obtain $\psi(a)$ is via a direct numerical minimization of $g(z, a)$, and this can be done *even if* $\rho(\cdot)$ *is nonconvex*, still leading to the global minimum solution of the penalty function given in (2). Note that addressing this nonconvex penalty term in (2) using classical iterative optimizers (steepest descent, conjugate gradient, Newton methods, etc.) is susceptible to local minima traps in general, and depend on the initialization chosen. Here, not only iterations have been avoided, but finding of the best solution is guaranteed.

As can be seen, if we assume that $\rho(\cdot)$ is monotonic nondecreasing for $a \geq 0$, (7) implies that the curve is sublinear, giving a shrinkage of the input coefficient a, and thus the name given to this procedure. Fig. 2 presents several such curves. Those where obtained by numerically finding the minimum of (6) for varying values of a.

After taking every coefficient $y_w(n)$ and passing it through the curve $\psi(y_w(n))$, the outputs are the representation coefficients of $\boldsymbol{x}$. A final stage of inverse transform provides the desired solution $\boldsymbol{x}$, as Fig. 1 shows.

III. Treating Redundant Transforms

We turn now to the case where the prior term is given by $P_r\{\boldsymbol{x}\} = \mathbf{1}^T \cdot \rho(\boldsymbol{T}\boldsymbol{x})$, with $\boldsymbol{T}$ being a nonsquare full rank matrix $\boldsymbol{T} \in \mathbb{R}^{L \times N}$, where $L > N$. The following discussion will be mostly restricted to the choice $\rho(z) = |z|$, although all the discussion can be easily generalized to other choices of $\rho(\cdot)$.

[7]Or can be presented as the limit of a sequence of such functions, which is a more reasonable assumption.

Let us first describe the heuristic shrinkage algorithm that could be done for denoising a signal under these circumstances. This algorithm is described as *Algorithm A*.

> *Algorithm A*—Heuristic Shrinkage
> **Task:** Denoise $\boldsymbol{y}$ using a heuristic shrinkage, based on the transform $\boldsymbol{T}$.
>
> **Data and Parameters:** $\lambda, \boldsymbol{T}$ and $\boldsymbol{y}$.
>
> **Step 1:** Compute $\boldsymbol{y}_T = \boldsymbol{T}\boldsymbol{y}$.
>
> **Step 2:** Apply shrinkage $\mathcal{S}\{\boldsymbol{y}_T\}$ with threshold λ.
>
> **Step 3:** Compute the inverse transform $\hat{\boldsymbol{x}} = \boldsymbol{T}^+ \mathcal{S}\{\boldsymbol{y}_T\}$.
>
> **Finalize:** The denoised output is $\hat{\boldsymbol{x}}$.

Note that in the above-described algorithm, it is unclear whether using a fixed and equal threshold to all coefficients is the proper way to go. We will see that this matter resolves as we relate this algorithm to a solid Baysian objective. We thus turn now to define the Bayesian objective for our denoising task.

A. The Generative Bayesian Objective

Following the preceding discussion, we intend to minimize a penalty similar to the one posed in (2), with an updated prior term

$$f_1(\boldsymbol{x}) = \frac{1}{2} \cdot \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho\{\boldsymbol{T}\boldsymbol{x}\}. \tag{8}$$

Defining $\boldsymbol{x}_T = \boldsymbol{T}\boldsymbol{x}$, multiplying both sides by $\boldsymbol{T}^T$ we get $\boldsymbol{T}^T\boldsymbol{x}_T = \boldsymbol{T}^T\boldsymbol{T}\boldsymbol{x}$. In the general case, $\boldsymbol{T}^T\boldsymbol{T}$ is invertible (since $\boldsymbol{T}$ is full-rank), and thus[8]

$$\boldsymbol{x} = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{x}_T = \boldsymbol{T}^+\boldsymbol{x}_T.$$

We can use these relations to rearrange (8) and obtain a new function of the representation vector $\boldsymbol{x}_T$

$$\begin{aligned} f_2(\boldsymbol{x}_T) &= \frac{1}{2} \cdot \|\boldsymbol{T}^+\boldsymbol{x}_T - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho\{\boldsymbol{x}_T\} \\ &= \frac{1}{2} \cdot \|\boldsymbol{D}\boldsymbol{x}_T - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho\{\boldsymbol{x}_T\} \end{aligned} \tag{9}$$

where we denote $\boldsymbol{D} = \boldsymbol{T}^+$.

Denoising can be done by minimizing f_1 and obtaining a solution $\hat{\boldsymbol{x}}_1$. Alternatively, we can minimize f_2 with respect to $\boldsymbol{x}_T$ and deduce the denoised outcome by $\hat{\boldsymbol{x}}_2 = \boldsymbol{D}\hat{\boldsymbol{x}}_T$. Interestingly, *these two results are not expected to be the same in the general case*, since in the conversion from f_1 to f_2 we have expanded the set of feasible solutions by allowing $\boldsymbol{x}_T$ to be an arbitrary vector in $\mathbb{R}^L$, whereas the original definition $\boldsymbol{x}_T = \boldsymbol{T}\boldsymbol{x}$ implies that it must be confined to the column space of $\boldsymbol{T}$. Notice that this difference between the two formulations disappears when $\boldsymbol{T}$ is a full-rank square matrix, which explains why this dichotomy of methods did not bother us in the previous section.

Still, the formulation posed in (9) is a feasible alternative Bayesian method that uses a generative prior. Indeed, for the choice $\rho\{z\} = |z|$, this formulation is known as the basis pursuit denoising (BPDN). Referring to $\boldsymbol{D}$ as a dictionary of signal prototypes (atoms) as its columns, we assume that the desired signal $\boldsymbol{x}$ is a linear construction of these atoms, with coefficients drawn independently from a probability density function proportional to $\exp\{-\text{Const} \cdot \rho\{x_T(j)\}\}$. In the case of $\rho(z) = |z|$ this is the Laplace distribution, and we effectively promote sparsity in the representation.

[8]If $\boldsymbol{T}$ is a tight frame ($\alpha\boldsymbol{T}^T\boldsymbol{T} = \boldsymbol{I}$), then the above leads to $\boldsymbol{x} = \alpha\boldsymbol{T}^T\boldsymbol{x}_T$.

B. Our Objective

Our objective is now the optimization problem

$$\min_{\boldsymbol{z}} \frac{1}{2} \cdot \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho\{\boldsymbol{z}\} \tag{10}$$

and the denoised output is $\hat{\boldsymbol{x}} = \boldsymbol{D}\hat{\boldsymbol{z}}$. Although BPDN has been defined for the specific choice $\rho(z) = |z|$, we shall refer hereafter to this more general objective as BPDN as well.

The numerical method we shall use in later experiments to compare against shrinkage is based on the iterative reweighed least-squares (IRLS) as practiced by the FOCUSS algorithm [26], [27]. We outline it here very briefly: Define a new scalar function $\rho_0(z)$ to satisfy $\rho(z) = 0.5|z|^2 \cdot \rho_0(z)$. For example, for $\rho(z) = |z|$, we have $\rho_0(z) = 2/|z|$ (or better yet, $\rho_0(z) = 2/(|z| + \epsilon)$ with $0 < \epsilon \ll 1$ so as to avoid divisions by zero). Now, the above formulation can be rewritten as

$$\min_{\boldsymbol{z}} \frac{1}{2} \cdot \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{y}\|_2^2 + \frac{\lambda}{2} \cdot \boldsymbol{z}^T \cdot \mathrm{diag}\{\rho_0\{\boldsymbol{z}\}\}\boldsymbol{z}. \tag{11}$$

The term $\mathrm{diag}\{\rho_0\{\boldsymbol{z}\}\}$ is a diagonal weight matrix, and thus the second term has a quadratic structure, if we assume this diagonal matrix to be fixed. Minimization can be done iteratively by assuming that the term $\rho_0\{\boldsymbol{z}\}$ is fixed, being computed with the current solution $\boldsymbol{z}_k$. Then the problem has a simple quadratic form, leading to the update equation

$$\boldsymbol{z}_{k+1} = [\boldsymbol{D}^T\boldsymbol{D} + \lambda \mathrm{diag}\{\rho_0\{\boldsymbol{z}_k\}\}]^{-1}\boldsymbol{D}^T\boldsymbol{y}. \tag{12}$$

The IRLS algorithm is described as *Algorithm B*.

Algorithm B—The Iterative Reweighed Least Squares (IRLS)

Task: Denoise the signal $\boldsymbol{y}$ by

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{z}} \frac{1}{2} \cdot \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \|\boldsymbol{z}\|_1.$$

Data and Parameters: λ, $\boldsymbol{D}$, and $\boldsymbol{y}$ are given, K_0 iterations.

Initialization: Set $k = 0$ and choose $\boldsymbol{z}_k = \mathbf{1}$.

Main Iteration: Set $k = 1$ and apply:

- **Weights:** Compute
$$\boldsymbol{W} = \mathrm{diag}\{\rho_0\{\boldsymbol{z}_{k-1}\}\} = 2\mathrm{diag}\{1/|\boldsymbol{z}_{k-1}|\}.$$
- **Update:** Compute $\boldsymbol{z}_k = [\boldsymbol{D}^T\boldsymbol{D} + \lambda\boldsymbol{W}]^{-1}\boldsymbol{D}^T\boldsymbol{y}$.
- **Return:** Set $k = k + 1$. If $k \leq K_0$ return to "Weights."

Finalize: The denoised output is $\hat{\boldsymbol{x}} = \boldsymbol{D}\boldsymbol{z}_{K_0}$.

Notice that we have recommended an initialization with ones. Using a zero initialization causes a slow start because then $\boldsymbol{D}^T\boldsymbol{D}$ is negligible compared to $\boldsymbol{W}$. The choice of ones parallels a regularized pseu-doinverse start.

In the experiments to follow, we shall implement this algorithm to minimize the function in (10), but it should be clear that in general this is a daunting task for typical sizes used in image processing ($N \approx 10e+4$, $L \approx 10e+6$, and beyond). The need to invert a matrix of size $L \times L$, as the above update formula requires, is prohibitive, and should be replaced with an iterative solver. In fact, this is why a shrinkage method as in *Algorithm A* would be of interest in the first place.

C. Shrinkage Again? A Sequential Method

For a dictionary built as a union of J ortho-matrices, there is yet another, more efficient, numerical solver, based on a block-coordinate-relaxation (BCR) process. This algorithm, as introduced by Bruce *et al.* [33] is using shrinkage in the spirit of Section II. The representation vector $\boldsymbol{z}$ is broken into J parts, each referring to a unitary matrix in $\boldsymbol{D}$. The BCR algorithm addresses one set of representation coefficients at a time, assuming all the others as fixed. We will imitate this idea here, but consider a general dictionary, and treat scalar entries in $\boldsymbol{z}$.

Assume that in an iterative process used to solve the above problem, we hold the kth solution $\hat{\boldsymbol{z}}_k$. We are interested in updating its jth entry $z(j)$, assuming all the others as fixed. Thus, we obtain a 1-D optimization problem of the form

$$\min_{w} \frac{1}{2} \cdot \|[\boldsymbol{D}\boldsymbol{z}_k - \boldsymbol{d}_j z_k(j)] + \boldsymbol{d}_j w - \boldsymbol{y}\|_2^2 + \lambda \cdot \rho\{w\}. \tag{13}$$

In the above expression, $\boldsymbol{d}_j$ is the jth column in $\boldsymbol{D}$. The term $\boldsymbol{D}\boldsymbol{z}_k - \boldsymbol{d}_j z_k(j)$ uses the current solution for all the coefficients, but discards of the jth one, assumed to be replaced with a new value, w.

Since this is a 1-D optimization task, it is relatively easy to solve. Assume, for example, that $\rho(w) = |w|$ as was done in the previous section. Taking a derivative with respect to w we get the equation

$$0 = \boldsymbol{d}_j^T([\boldsymbol{D}\boldsymbol{z}_k - \boldsymbol{d}_j z_k(j)] + \boldsymbol{d}_j w - \boldsymbol{y}) + \lambda \cdot \mathrm{sign}\{w\} \tag{14}$$

leading to

$$\begin{aligned} w &= \frac{\boldsymbol{d}_j^T(\boldsymbol{y} - [\boldsymbol{D}\boldsymbol{z}_k - \boldsymbol{d}_j z_k(j)])}{\|\boldsymbol{d}_j\|_2^2} - \frac{\lambda \cdot \mathrm{sign}\{w\}}{\|\boldsymbol{d}_j\|_2^2} \\ &= z_k(j) + \frac{\boldsymbol{d}_j^T(\boldsymbol{y} - \boldsymbol{D}\boldsymbol{z}_k)}{\|\boldsymbol{d}_j\|_2^2} - \frac{\lambda \cdot \mathrm{sign}\{w\}}{\|\boldsymbol{d}_j\|_2^2} \\ &= v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j) - \hat{\lambda}(j) \cdot \mathrm{sign}\{w\}. \end{aligned} \tag{15}$$

Here we have defined

$$\begin{aligned} v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j) &= \frac{\boldsymbol{d}_j^T(\boldsymbol{y} - \boldsymbol{D}\boldsymbol{z}_k)}{\|\boldsymbol{d}_j\|_2^2} + z_k(j) \quad \text{and} \\ \hat{\lambda}(j) &= \frac{\lambda}{\|\boldsymbol{d}_j\|_2^2}. \end{aligned} \tag{16}$$

Both $v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j)$ and $\hat{\lambda}(j)$ are computable using the known dictionary, noisy signal, current solution, the value of λ, and the index in question. Thus, the same reasoning as the one practiced in Section II leads to a closed-form formula for the optimal solution for w, being a shrinkage operation on $v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j)$

$$\begin{aligned} w_{\mathrm{opt}} &= \mathcal{S}\{v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j)\} \\ &= \begin{cases} v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j) - \hat{\lambda}(j), & \text{for } v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j) > \hat{\lambda}(j) \\ 0, & \text{for } |v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j)| \leq \hat{\lambda}(j) \\ v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j) + \hat{\lambda}(j), & \text{for } v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j) < -\hat{\lambda}(j). \end{cases} \end{aligned} \tag{17}$$

A similar LUT result can be developed for any other choice of the function $\rho(\cdot)$.

It is tempting to suggest an algorithm that applies the above procedure for $j = 1, 2, \ldots, L$, updating one coefficient at a time in a sequential coordinate descent algorithm, and cycle such process several times. Such algorithm is described as *Algorithm C*.

Algorithm C—Sequential Shrinkage

Task: Denoise the signal $\boldsymbol{y}$ by

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{z}} \frac{1}{2} \cdot \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot |\boldsymbol{z}|_1.$$

Data and Parameters: λ, $\boldsymbol{D}$, and $\boldsymbol{y}$ are given, K_0 iterations.

Initialization: Set $k = 0$ and choose $\boldsymbol{z}_k = \mathbf{0}$.

Main Iteration: Set $k = 1$ and apply:

- **Sweep:** Set $j = 1$.
  1. 1. Compute $v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j) = \frac{\boldsymbol{d}_j^T(\boldsymbol{y} - \boldsymbol{D}\boldsymbol{z}_k)}{\|\boldsymbol{d}_j\|_2^2} + z_k(j)$.
  2. Compute $w_{\text{opt}} = \mathcal{S}\{v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j)\}$ using threshold $\lambda/\|\boldsymbol{d}_j\|_2^2$.
  3. Update the solution at the jth location to be $z_k(j) = w_{\text{opt}}$.
  4. Set $j = j + 1$. If $j \leq L$, return to step 1.
- **Return:** Set $k = k + 1$. If $k \leq K_0$ return to "Sweep."
- **Finalize:** The denoised output is $\hat{\boldsymbol{x}} = \boldsymbol{D}\boldsymbol{z}_{K_0}$.

While such algorithm necessarily converges, and could be effective in minimizing the objective function using scalar shrinkage operations only, it is impractical in most cases. The reason is the necessity to draw one column at a time from $\boldsymbol{D}$ to perform this computation. Consider, for example, the curvelet dictionary. While the transform and its inverse can be interpreted as multiplications by the dictionary and its transpose (because it is a tight frame), this matrix is never explicitly constructed, and an attempt to draw basis functions from it or store them could be devastating. Thus, we take a different route.

D. Shrinkage in a Parallel Method

Given the current solution $\boldsymbol{z}_k$, let us assume that we use the above update formulation to update *all the coefficients* in parallel, rather than doing this sequentially. Obviously, this process must be slower in minimizing the objective function, but with this slowness comes a much desired simplicity that will be evident shortly.

First, let us convert the terms $v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, j)$ in (16) to a vector form that accounts for all the updates at once. Gathering these terms for all $j \in [1, L]$, this reads

$$\begin{aligned}\boldsymbol{v}(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k) &= \begin{bmatrix} v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, 1) \\ v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, 2) \\ \vdots \\ v(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k, L) \end{bmatrix} \\ &= \text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\}\boldsymbol{D}^T(\boldsymbol{y} - \boldsymbol{D}\boldsymbol{z}_k) + \boldsymbol{z}_k. \end{aligned} \quad (18)$$

Notice that in the computation of $\boldsymbol{v}$ in the above equation, we do not need to extract some columns from the dictionary, and need not use these matrices explicitly in any other way. If the transform we use is such that multiplication by $\boldsymbol{D}$ and its adjoint $\boldsymbol{D}^T$ are fast, then computing the above term is easy and efficient. The normalization by the norms of the columns is simple to obtain and can be kept as fixed parameters of the transform, computed once off-line.[9]

In the case of tight frames, applying multiplications by $\boldsymbol{D}^T$ and $\boldsymbol{D}$ are the forward and the inverse transforms, up to a constant. For a non-tight frame, the above formula says that we need to be able to apply the adjoint *and not the pseudoinverse* of $\boldsymbol{D}$.

There is also a natural weakness to the above strategy. One cannot take a shrinkage of the above vector with respect to the threshold vector $\lambda \cdot \text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\} \cdot \mathbf{1}$, and expect the objective function to be minimized well. While updating every scalar entry w_j using the above shrinkage formula is necessarily decreasing the function's value, applying all those at once is likely to diverge, and cause an ascent in the objective. Thus, instead of applying a complete shrinkage as (17) suggests, we consider a relaxed step of the form

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \mu[\mathcal{S}\{\boldsymbol{v}(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{z}_k)\} - \boldsymbol{z}_k] = \boldsymbol{z}_k + \mu\boldsymbol{h}_k. \quad (19)$$

This way, we compute the shrinkage vector as the formula suggests, and use it to define a descent direction. The solution is starting from the current solution $\boldsymbol{z}_k$ and updates it by "walking" toward the shrinkage result. For $\mu = 1$, the shrinkage is adopted in full, and for $\mu < 1$ the effect is a relaxed step. For a sufficiently small $\mu > 0$, this step *must* lead to a feasible descent in the penalty function, because this direction is a nonnegative combination of L descent directions.

We can apply a line search to find the proper choice for the value of μ. In general, line search seeks the best step size as a 1-D optimization procedure that solves

$$\min_{\mu} \frac{1}{2} \cdot \|\boldsymbol{D}[\boldsymbol{z}_k + \mu\boldsymbol{h}_k] - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho\{\boldsymbol{z}_k + \mu\boldsymbol{h}_k\} \quad (20)$$

where $\boldsymbol{h}_k$ is a computable vector. The solution is given by solving the equation

$$0 = \boldsymbol{h}_k^T\boldsymbol{D}^T(\boldsymbol{D}[\boldsymbol{z}_k + \mu\boldsymbol{h}_k] - \boldsymbol{y}) + \lambda \cdot \boldsymbol{h}_k^T \cdot \rho'\{\boldsymbol{z}_k + \mu\boldsymbol{h}_k\}. \quad (21)$$

This again has a shrinkage-like structure. Finding an appropriate μ amounts to a multiplication by $\boldsymbol{D}$ and its adjoint to compute the first term, and then seek optimal solution for μ by a zero-crossing iterative solver.

To summarize our findings so far, we desire a solution to the optimization problem posed in (10). We do this iteratively, and update the result by performing a shrinkage. Defining the solution at the kth iteration by $\boldsymbol{z}_k$, it is updated by computing $\boldsymbol{z}_k + \text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\}\boldsymbol{D}^T(\boldsymbol{y} - \boldsymbol{D}\boldsymbol{z}_k)$, applying shrinkage to it using the threshold vector $\lambda \cdot \text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\} \cdot \mathbf{1}$, and finally applying a line search on the line that connect $\boldsymbol{z}_k$ and the new result to get the best descent. This algorithm is described as *Algorithm D*.

Algorithm D—Parallel Shrinkage

Task: Denoise the signal $\boldsymbol{y}$ by

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{z}} \frac{1}{2} \cdot \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot |\boldsymbol{z}|_1.$$

Data and Parameters: λ, $\boldsymbol{D}$, $\boldsymbol{y}$ and K_0 (iterations) are given. $\boldsymbol{W} = \text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\}$ is computed off-line.

Initialization: Set $k = 0$ and choose $\boldsymbol{z}_k = \mathbf{0}$.

Main Iteration: Set $k = 1$ and apply:

- **Error:** Compute $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{D}\boldsymbol{z}_{k-1}$.
- **Project:** Compute $\boldsymbol{e}_T = \boldsymbol{W}\boldsymbol{D}^T\boldsymbol{e}$.
- **Shrinkage:** Compute $\boldsymbol{e}_T^S = \mathcal{S}\{\boldsymbol{e}_T + \boldsymbol{z}_{k-1}\}$ with threshold $\lambda \cdot \boldsymbol{W} \cdot \mathbf{1}$.
- **Line-Search:** Find μ_0 to obtains a descent with $\boldsymbol{z}_{k-1} + \mu(\boldsymbol{e}_T^S - \boldsymbol{z}_{k-1})$.
- **Relax:** Update $\boldsymbol{z}_k = \boldsymbol{z}_{k-1} + \mu_0(\boldsymbol{e}_T^S - \boldsymbol{z}_{k-1})$.
- **Return:** Set $k = k + 1$. If $k \leq K_0$ return to "Error."

Finalize: The denoised output is $\hat{\boldsymbol{x}} = \boldsymbol{D}\boldsymbol{z}_{K_0}$.

[9]While storing the columns of $\boldsymbol{D}$ may require huge memory volume, storing a scalar per column is reasonable.

E. The First Iteration—A Closer Look

Let us now concentrate on the first iteration, assuming that the algorithm is initialized with $\boldsymbol{z}_0 = \boldsymbol{0}$. The term in (18) becomes

$$\boldsymbol{v}(\boldsymbol{D}, \boldsymbol{y}, \boldsymbol{0}) = \text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\}\boldsymbol{D}^T\boldsymbol{y}. \tag{22}$$

The solution $\boldsymbol{z}_1$ is obtained by first applying shrinkage to the above vector, using $\lambda\,\text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\}\boldsymbol{1}$ as the threshold vector, and then relaxing it, as in (19), giving

$$\boldsymbol{z}_1 = \mu\mathcal{S}\{\text{diag}^{-1}\{\bar{\boldsymbol{D}}^T\bar{\boldsymbol{D}}\}\bar{\boldsymbol{D}}^T\boldsymbol{y}\}. \tag{23}$$

As to the line search for choosing μ, it is found by solving (19). For the case of $\rho(z) = |z|$, this gives

$$\begin{aligned} 0 &= \boldsymbol{h}_1^T\boldsymbol{D}^T(\mu\boldsymbol{D}\boldsymbol{h}_1 - \boldsymbol{y}) + \lambda\cdot\boldsymbol{h}_1^T\cdot\rho'\{\mu\boldsymbol{h}_1\} \\ &= \boldsymbol{h}_1^T\boldsymbol{D}^T(\mu\boldsymbol{D}\boldsymbol{h}_1 - \boldsymbol{y}) + \lambda\mu\boldsymbol{h}_1^T\cdot\text{sign}\{\boldsymbol{h}_1\} \\ &= \mu\|\boldsymbol{D}\boldsymbol{h}_1\|_2^2 - \boldsymbol{y}^T\boldsymbol{D}\boldsymbol{h}_1 + \lambda\mu\|\boldsymbol{h}_1\|_1. \end{aligned} \tag{24}$$

Thus, we choose

$$\mu = \frac{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{h}_1}{\|\boldsymbol{D}\boldsymbol{h}_1\|_2^2 + \lambda\|\boldsymbol{h}_1\|_1} \tag{25}$$

and this can be computed by applying the multiplication by $\boldsymbol{D}$ only once more. To conclude, the denoised result is obtained by computing

$$\hat{\boldsymbol{x}} = \mu\cdot\boldsymbol{D}\mathcal{S}\{\text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\}\boldsymbol{D}^T\boldsymbol{y}\} \tag{26}$$

where the threshold to use in the shrinkage is given by $\lambda\cdot\text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\}\boldsymbol{1}$.

As a side note, we mention that we can give a rough estimate to the value of μ under several simplifying assumptions. We assume that i) $\bar{\boldsymbol{D}}$ represents a tight frame, i.e., $\boldsymbol{D}\boldsymbol{D}^T = \alpha\boldsymbol{I}$; ii) its columns are ℓ^2-normalized, i.e., $\text{diag}^{-1}\{\boldsymbol{D}^T\boldsymbol{D}\} = \boldsymbol{I}$; and iii) the shrinkage operation is nearly transparent, i.e., $\mathcal{S}\{\boldsymbol{D}^T\boldsymbol{y}\} \approx \boldsymbol{D}^T\boldsymbol{y}$, meaning that the shrinkage almost does not change the data fed to it. Such is the case, for example, for a dictionary built as a union of ortho-matrices. Under these assumptions we obtain

$$\begin{aligned} \mu &\approx \frac{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{D}^T\boldsymbol{y}}{\|\boldsymbol{D}\boldsymbol{D}^T\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{D}^T\boldsymbol{y}\|_1} \\ &= \frac{\alpha\cdot\|\boldsymbol{y}\|_2^2}{\alpha^2\cdot\|\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{D}^T\boldsymbol{y}\|_1} \approx \frac{1}{\alpha}. \end{aligned} \tag{27}$$

F. Relation To Heuristic Shrinkage

How all this compares with the heuristic shrinkage we described in *Algorithm A*? Recall that we have defined $\boldsymbol{D} = \boldsymbol{T}^+ = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T$, and thus $\boldsymbol{D}^T = \boldsymbol{T}\cdot(\boldsymbol{T}^T\boldsymbol{T})^{-1}$. Going back to the final formula given in (26) and using the relation between $\boldsymbol{T}$ and $\boldsymbol{D}$, we have that the denoising obtained here amounts to

$$\begin{aligned} \hat{\boldsymbol{x}} &= \mu\cdot\boldsymbol{T}^+\mathcal{S}\{\text{diag}^{-1}\{[\boldsymbol{T}^+]^T\boldsymbol{T}^+\}[\boldsymbol{T}^+]^T\boldsymbol{y}\} \\ &= \mu\cdot(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\mathcal{S}\{\text{diag}^{-1} \\ &\quad\times\{\boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-2}\boldsymbol{T}^T\}\boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{y}\}. \end{aligned} \tag{28}$$

For a general redundant transform, this formulation seems different from the one proposed in *Algorithm A* is several ways.

- Instead of starting by applying the transform on the signal, $\boldsymbol{T}\boldsymbol{y}$, the transposed inverse $[\boldsymbol{T}^+]^T\boldsymbol{y}$ is applied.
- The outcome is scaled element-by-element, by the matrix $\text{diag}^{-1}\{[\boldsymbol{T}^+]^T\boldsymbol{T}^+\}$, which does not appear in the heuristic algorithm.
- The shrinkage is applied with respect to a varying threshold, depending on the same diagonal matrix mentioned above. Thus, the threshold comparison is independent of this scale.
- In the return to the signal domain, both methods employ a multiplication by $\boldsymbol{T}^+$, however, here we have also a scale μ to take into account.

Are these algorithms truly so different? Let us consider several special cases.

Case 1: $\boldsymbol{T}$ is a tight frame, built with normalized columns of the dictionary. In this case, $\alpha\boldsymbol{T}^T\boldsymbol{T} = \boldsymbol{I}$ and $\text{diag}^{-1}\{\boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-2}\boldsymbol{T}^T\} = \boldsymbol{I}$. Thus, (28) becomes

$$\begin{aligned} \hat{\boldsymbol{x}} &= \mu\cdot(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\mathcal{S}\{\text{diag}^{-1}\{\boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-2}\boldsymbol{T}^T\}\boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{y}\} \\ &= \mu\boldsymbol{T}^+\mathcal{S}\{\alpha\boldsymbol{T}\boldsymbol{y}\}. \end{aligned} \tag{29}$$

The shrinkage in this case is done with a threshold λ. Assuming $\rho(z) = |z|$, we have

$$\mathcal{S}\{\alpha[\boldsymbol{T}\boldsymbol{y}]_j\} = \begin{cases} \alpha[\boldsymbol{T}\boldsymbol{y}]_j - \lambda, & \text{for } [\boldsymbol{T}\boldsymbol{y}]_j > \lambda/\alpha \\ 0, & \text{for } |[\boldsymbol{T}\boldsymbol{y}]_j| \le \lambda/\alpha \\ \alpha[\boldsymbol{T}\boldsymbol{y}]_j + \lambda, & \text{for } [\boldsymbol{T}\boldsymbol{y}]_j < -\lambda/\alpha. \end{cases} \tag{30}$$

If we choose $\mu = 1/\alpha$ as previously suggested, this scales the above equation to give

$$\begin{aligned} \mu\mathcal{S}\{\alpha[\boldsymbol{T}\boldsymbol{y}]_j\} &= \frac{1}{\alpha}\mathcal{S}\{\alpha[\boldsymbol{T}\boldsymbol{y}]_j\} \\ &= \begin{cases} [\boldsymbol{T}\boldsymbol{y}]_j - \lambda/\alpha, & \text{for } [\boldsymbol{T}\boldsymbol{y}]_j > \lambda/\alpha \\ 0, & \text{for } |[\boldsymbol{T}\boldsymbol{y}]_j| \le \lambda/\alpha \\ [\boldsymbol{T}\boldsymbol{y}]_j + \lambda/\alpha, & \text{for } [\boldsymbol{T}\boldsymbol{y}]_j < -\lambda/\alpha \end{cases} \end{aligned} \tag{31}$$

and this is the very same algorithm we presented in *Algorithm A* with a threshold chosen as λ/α.

Case 2: $\boldsymbol{T}$ is a general frame with normalized columns of the dictionary. In this case, $\text{diag}^{-1}\{\boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-2}\boldsymbol{T}^T\} = \boldsymbol{I}$. Thus, (28) becomes

$$\hat{\boldsymbol{x}} = \mu\boldsymbol{T}^+\mathcal{S}\{[\boldsymbol{T}^+]^T\boldsymbol{y}\}. \tag{32}$$

Given the matrix $\boldsymbol{T}$, we can define a new forward transform as $\tilde{\boldsymbol{T}}\boldsymbol{y} = [\boldsymbol{T}^+]^T\boldsymbol{y}$. With this definition, the above operation amounts to an application of the newly defined transform, application of shrinkage with a constant threshold λ, and then using the transform's adjoint to return to the signal domain. This stands as an interesting alternative to *Algorithm A*, being similar in some respects, and different in others. Specifically, we need to redefine how the transform operates, and we do not use its inverse but adjoint.

Case 3: $\boldsymbol{T}$ is a tight frame, with nonnormalized columns. In this case, $\alpha\boldsymbol{T}^T\boldsymbol{T} = \boldsymbol{I}$, and thus (28) becomes

$$\begin{aligned} \hat{\boldsymbol{x}} &= \mu\cdot(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\mathcal{S}\{\text{diag}^{-1}\{\boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-2}\boldsymbol{T}^T\}\boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{y}\} \\ &= \mu\alpha\cdot\boldsymbol{T}^T\mathcal{S}\{\frac{1}{\alpha}\text{diag}^{-1}\{\boldsymbol{T}\boldsymbol{T}^T\}\boldsymbol{T}\boldsymbol{y}\}. \end{aligned} \tag{33}$$

The shrinkage in this case is done for the jth entry with a threshold $\frac{\lambda}{\alpha^2}\|\boldsymbol{t}_j\|_2^{-2}$, where $\boldsymbol{t}_i$ is the jth row in $\boldsymbol{T}$. Assuming $\rho(z) = |z|$, we have

$$\mathcal{S}\left\{\frac{[\boldsymbol{T}\boldsymbol{y}]_j}{\alpha\|\boldsymbol{t}_j\|_2^2}\right\} = \begin{cases} \frac{[\boldsymbol{T}\boldsymbol{y}]_j - \lambda/\alpha}{\alpha\|\boldsymbol{t}_j\|_2^2}, & \text{for } [\boldsymbol{T}\boldsymbol{y}]_j > \lambda/\alpha \\ 0, & \text{for } |[\boldsymbol{T}\boldsymbol{y}]_j| \le \lambda/\alpha \\ \frac{[\boldsymbol{T}\boldsymbol{y}]_j + \lambda/\alpha}{\alpha\|\boldsymbol{t}_j\|_2^2}, & \text{for } [\boldsymbol{T}\boldsymbol{y}]_j < -\lambda/\alpha. \end{cases} \tag{34}$$

The multiplication after the shrinkage by $\alpha\boldsymbol{T}^T$ stands for a regular inverse transform, and thus the multiplication by μ should be absorbed in the shrinkage. This way we obtained a very similar algorithm to the one presented in *Algorithm A*, with a regular forward transform, regular shrinkage with a constant threshold λ/α, a scale of each entry by $\frac{\mu}{\alpha\|\boldsymbol{t}_j\|_2^2}$, and finally an inverse transform. Thus, the only difference is the element-wise scale of the transformed coefficients, prior to the return to the signal domain.

G. Discussion

The fact that our algorithm differs from *Algorithm A* for nontight frames or nonnormalized dictionaries should not be interpreted as a statement that *Algorithm A* is wrong. It may well be that an alternative justification could be developed, leading to *Algorithm A* as a first iteration. As an example, one could consider $\boldsymbol{D}^+\boldsymbol{y}$ as an initialization to our denoising process, based on the replacement of the ℓ^1-norm with an ℓ^2 one. Using this initialization, followed by appropriate adjustments after one iteration of some algorithm, it may be possible to give rise to a different shrinkage algorithm than the one we have developed. In this work we have not followed this line of reasoning.

An immediate benefit that can be drawn from the above results is the ability to operate the heuristic shrinkage in a better setting. Also, if we are willing to invest more computations, the above results give us a way to further minimize the objective by more iterations that are shrinkage-like, and this way possibly get stronger noise removal.

Also, from a different perspective, since BPDN is considered as an important objective functional (e.g., as a nonlinear transform that promotes sparsity), what we have obtained here (as described in *Algorithm D*) could be an effective solver that uses only simple and fast operations. Thus, when applying a complicated transform such as curvelet or contourlet, instead of using the ℓ^2-based linear method for obtaining the forward redundant transform, one can use the basis pursuit (or BPDN) and get a sparser outcome (see [44] for such experiments on image denoising using the BPDN with contourlet). *Algorithm D* can perform this task by applying the regular forward and adjoint transforms, coupled with simple shrinkage steps. As such, this algorithm can be perceived as a novel and effective pursuit algorithm.

IV. Experimental Results

In this section, we present four experiments—the first three match the cases discussed in Section III-F, illustrating the performance of the various algorithms discussed on some signal examples. The fourth experiment present the average denoising behavior of *Algorithm D* as a function of the noise power and the cardinality of the representation.

Experiment 1—A tight frame with normalized columns: We build $\boldsymbol{D}$ as a union of 10 random unitary matrices of size 100×100. We synthesize a sparse representation $\boldsymbol{z}_0$ with 15 nonzeros in random locations and Gaussian independent and identically distributed (i.i.d.) entries, so as to match the sparsity prior we use. Thus, the clean signal is defined as $\boldsymbol{x}_0 = \boldsymbol{D}\boldsymbol{z}_0$. This signal is contaminated by a Gaussian i.i.d. noise $\sigma = 0.3$ (parallels a signal-to-noise ratio (SNR) of ≈ 1.3 dB). We apply Algorithms A-D with $\rho(z) = |z|$, and the results are reported in Figs. 3–5.

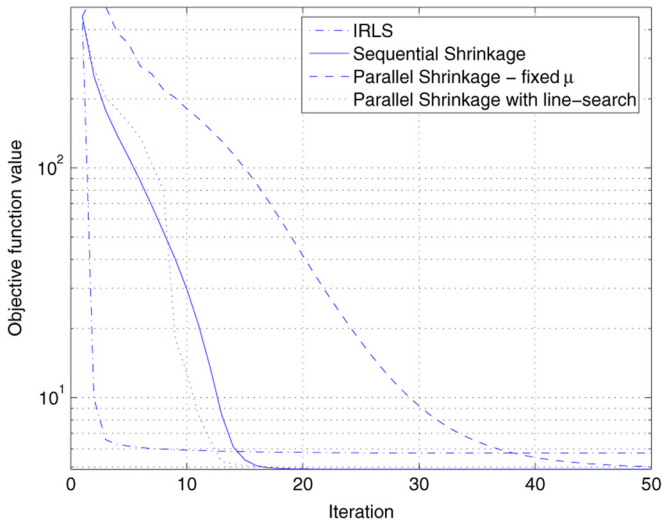

Fig. 3. Experiment 1: The objective in (10) as a function of the iteration—Algorithms B–D.

First, we show how effective are Algorithms B–D in minimizing the objective in (10). Fig. 3 presents the value of the objective as a function of the iteration number. Here we have implemented *Algorithm D* both with a fixed $\mu = 1/\alpha$ and with a line search. The IRLS (*Algorithm B*) performs the best in terms of convergence speed at the first five iterations, but then slows dramatically. The sequential and the parallel (with line search) coordinate descent are comparable to each other, being somewhat inferior to the IRLS at the first five iterations, but show a consistent decent in the function value afterwards.

When implementing Algorithms A–D, we sweep through the possible values of λ to find the best choice. In this correspondence, we have not treated the question of how to automatically find it. Also, in assessing the denoising effect, we use the noise decay factor measure, $r(\hat{\boldsymbol{x}}, \boldsymbol{x}_0, \boldsymbol{y}) = \|\hat{\boldsymbol{x}} - \boldsymbol{x}_0\|_2^2 / \|\boldsymbol{y} - \boldsymbol{x}_0\|_2^2$, which gives the ratio between the final reconstruction error and the error with $\boldsymbol{y}$ as our estimate. Thus, a value smaller than 1 implies a decay in the noise, and the closer it is to zero the better the result.

Using the IRLS (with few iterations due to its fast convergence) can give us an evaluation of the denoising potential that exists in the objective function we use. We compare the IRLS results (after the first and the fifth iterations) to the simple shrinkage *Algorithm A*. The simple shrinkage in this case uses a threshold being $\lambda/\alpha = 10\lambda$, based on (30), so as to match to the objective function that uses λ in its formulation. Fig. 4 (top left) presents this comparison, showing the noise decay factor versus λ. Interestingly, it appears than the simple shrinkage manages to utilize most of the denoising potential, and five iterations of the IRLS give only slightly better results.

Fig. 4 also presents similar comparisons of the simple shrinkage with the sequential coordinate descent (*Algorithm C*), and the parallel coordinate descent with line search or with a fixed μ chosen as $\mu = 1/\alpha$. First, we see that five iterations of the sequential coordinate descent are as effective as five IRLS iterations, giving better results than the simple shrinkage. Second, we see that the first iteration of the parallel shrinkage aligns perfectly with the simple shrinkage when $\mu = 1/\alpha$, as predicted, and having five iterations gives a slight improvement. Finally, as line search is introduced in *Algorithm D*, the results hardly change, implying that the choice $\mu = 1/\alpha$ is near-optimal.

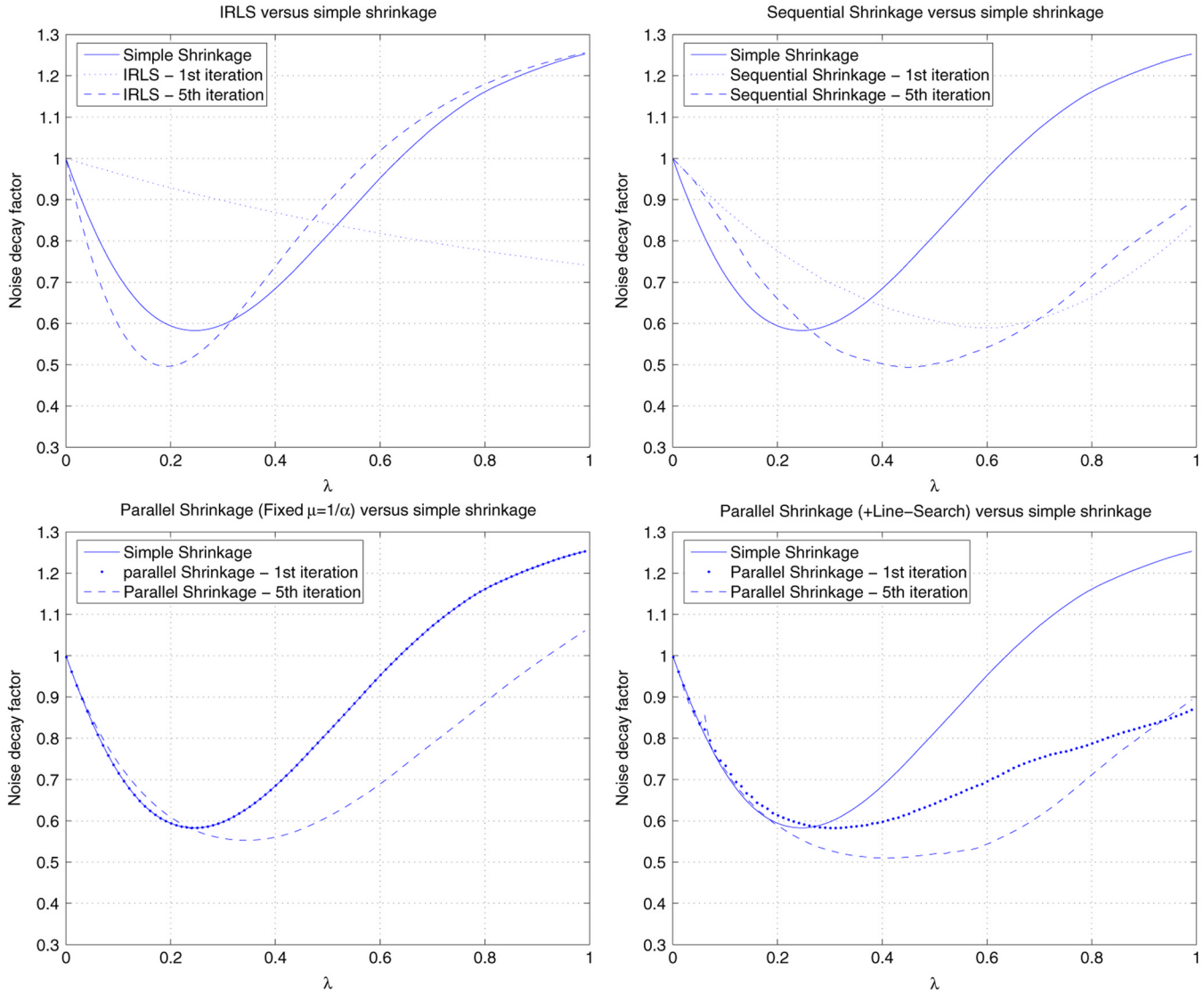

Fig. 4 Experiment 1 results—comparing denoising effects of the various algorithms. Top left: the IRLS versus simple shrinkage. Top right: the sequential coordinate descent algorithm versus simple shrinkage. Bottom left: the parallel coordinate descent algorithm with fixed $\mu = 1/\alpha$ versus simple shrinkage. Bottom right: the parallel coordinate descent algorithm with line-search versus simple shrinkage.

Fig. 5 presents the actual μ found by the line search in *Algorithm D* in the first iteration, as a function of the varying λ. We see that for small λ (where our assumptions in Section III-F hold true), the value found is close to $1/\alpha$ as expected.

Experiment 2—A nontight frame with normalized columns: We build $\boldsymbol{D}$ as a random matrix of size 100×1000 with entries drawn as Gaussian i.i.d., and then normalize each column. The rest of the data generation follows the same procedure described for Experiment 1.

Generally speaking, the results of this experiment are similar to those in Experiment 1, assuggested by Fig. 6. Here we cannot align the choice of threshold in the simple shrinkage to the choice of λ in the objective function, simply because those two have not been related. We see that five iterations of the IRLS or the sequential CD can give a substantial improvement in denoising. Also, we see that the first iteration of *Algorithm D* that parallels in complexity to the simple shrinkage perform as good, and adding several iterations give further noise decay. Here we have not tried a fixed μ since the $1/\alpha$ rule does not apply.

Experiment 3—A general frame: We build $\boldsymbol{D}$ as a random matrix of size 100×1000 with entries drawn as Gaussian i.i.d. We deliberately change the scale of the columns to range linearly between 0.5 and 1.

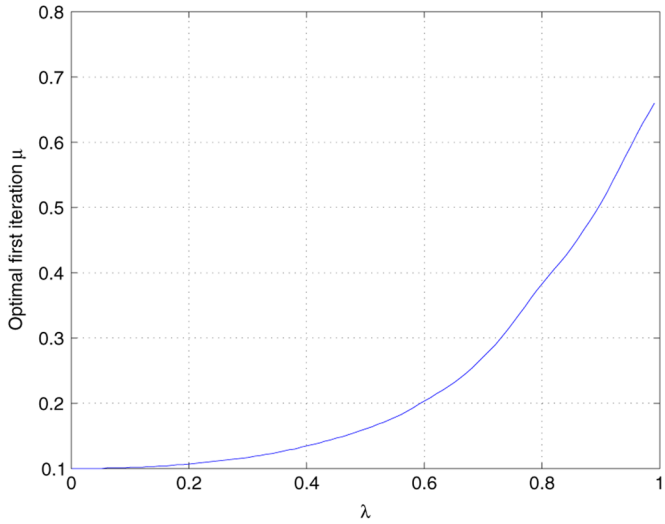

Fig. 5. Experiment 1: The line-search results for μ in the first iteration.

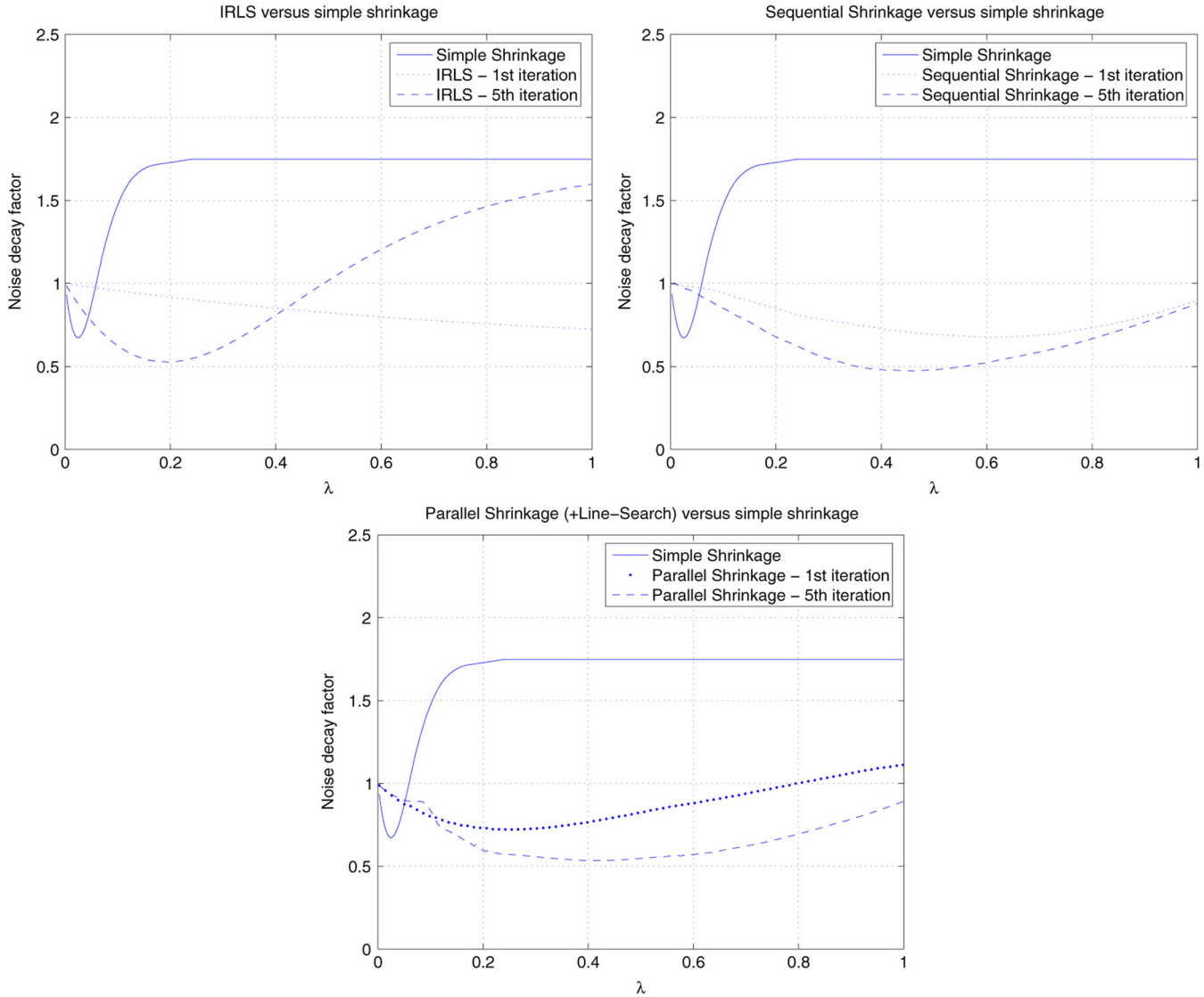

Fig. 6. Experiment 2 results—comparing denoising effects of the various algorithms. Top left: the IRLS versus simple shrinkage. Top right: the sequential coordinate descent algorithm versus simple shrinkage. Bottom: the parallel coordinate descent algorithm with line-search versus simple shrinkage.

The rest of the data generation follows the same procedure described for Experiment 1. Fig. 7 presents how *Algorithm D* compares to the simple shrinkage. As can be seen again, while the alternative shrinkage formulation we get is different from the heuristic method, it has the same complexity and a comparable (and slightly better) noise decay. Performing five iterations further improves the performance slightly more.

Experiment 4—Average performance: The results reported above correspond to one specific signal and the denoising obtained for it, so as to illustrate the relation between the various methods. We now introduce a wider experiment, where a corpus of signals is generated, contaminated by additive noise, and then denoised. Our objective here is to show the *average* amount of better denoising that can be expected when turning from the heuristic shrinkage (i.e., the first iteration of *Algorithm D*) to several iterations of *Algorithm D*.

Using the same dictionary as in Experiment 1, the signals in this experiment are generated by synthesizing a sparse representation z_0 with L nonzeros, where $1 \leq L \leq 20$. Each such signal is normalized, and then contaminated by additive Gaussian noise with varying power in the range $\sigma = [0.03, 0.96]$ (i.e., SNR in the range $[0, 30]$ dB). Per each L and σ, we generate 50 random signals, and apply denoising based on *Algorithm D* with 1 to 10 iterations. Per each experiment we

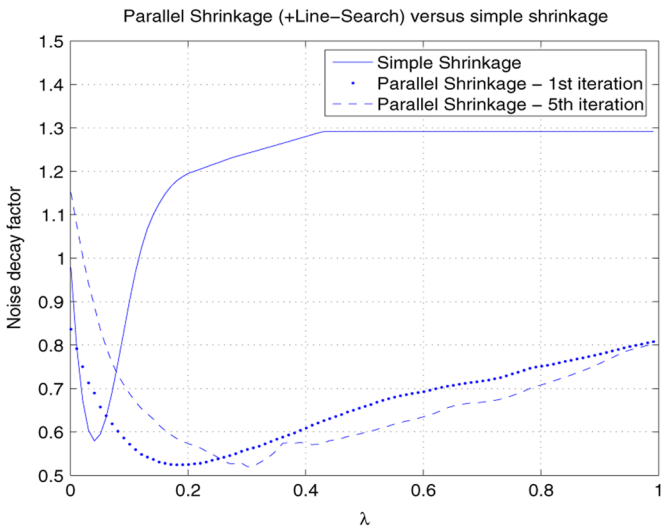

Fig. 7. Experiment 3: The denoising effect of the parallel coordinate descent algorithm with line-search versus simple shrinkage.

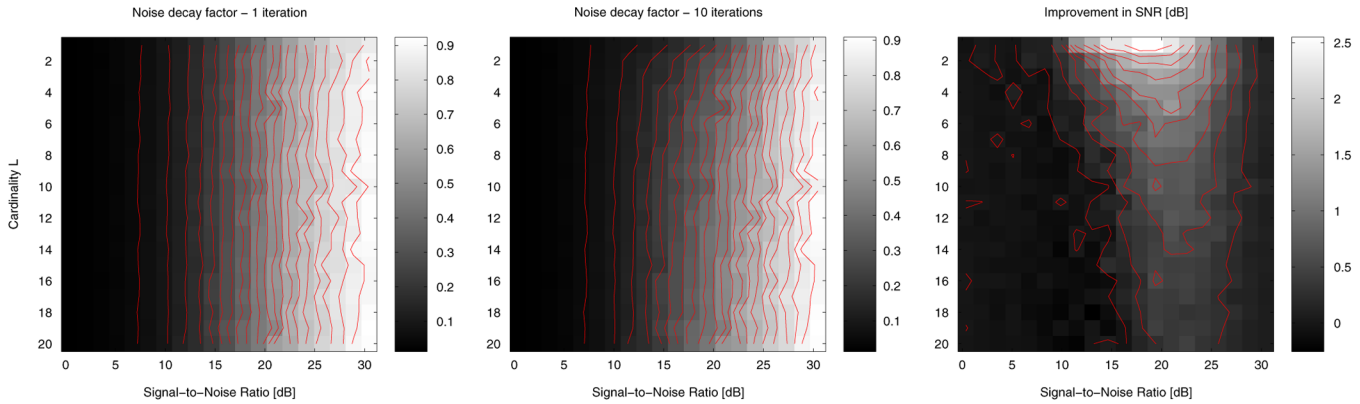

Fig. 8. Experiment 4: Average denoising obtained by *Algorithm D* with one iteration (left), 10 iterations (middle), and their difference in decibels (right). The results are shown as images with gray values being proportional to the results. The bar on the right of each image shows the relation between brightness and resulting values. For convenience, the equi-height contours of the results are overlayed.

choose the optimal λ, as done in Experiment 1, so as to exclude its influence.

Fig. 8 shows the average noise decay factors obtained per each L and input SNR. Several conclusions can be drawn from the results: i) The denoising results are roughly the same for cardinalities in the range $[1, 20]$, implying that all these signals are sparse enough and thus handled similarly; ii) the denoising effect depends on the SNR, showing better denoising results for a higher SNR; and iii) using more than one iteration in *Algorithm D* we typically get better performance with up to 2.5-dB improvement. In some cases, more iterations may cause a deterioration in the denoising performance, but when this happens, it is a very mild loss (less than 0.05 dB on average).

V. Related Work

Interestingly, a sequence of recent contributions proposed a similar sequential shrinkage algorithm. First, the work reported in [40], [41] uses such an algorithm for finding the sparsest representation over redundant dictionaries (such as the curvelet, or combination of dictionaries). These papers motivated such algorithm heuristically, relying on the resemblance to the unitary case, on one hand, and the block-coordinate-relaxation method, on the other [33].

Figueiredo and Nowak suggested a constructive method for image deblurring, based on iterated shrinkage [42]. Their algorithm aims at minimizing the penalty function

$$f_B(\boldsymbol{x}) = \frac{1}{2} \cdot \|\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho\{\boldsymbol{W}\boldsymbol{x}\} \tag{35}$$

where $\boldsymbol{K}$ is a (square) matrix representing the blur, and $\boldsymbol{W}$ is a unitary wavelet transform. Their sequential shrinkage method is derived via expectation–maximization (EM), and its structure is very similar to the method proposed in this work. It turns out that their algorithm is not restricted to the case of square matrix $\boldsymbol{K}$, and as such can be generalized to handle the minimization of the objective posed in (9) by defining $\boldsymbol{D} = \boldsymbol{K}\boldsymbol{W}^H$.

Similarly, the paper by Daubechies, Defrise, and De-Mol [43] addresses the same objective as posed above, leaning on the definition of a sequence of surrogate functions, each minimized via shrinkage. This leads to yet another iterated shrinkage algorithm, very much like the one in [42].

While these two algorithms (EM-based and surrogate-based) are similar to ours, they are not the same. The norms of the atoms play different roles in these algorithms; the thresholds chosen in the shrinkage are somewhat different; and the choice of μ is done entirely different. Further work is required to clarify the relation between these methods.

VI. Conclusion

In this correspondence, we studied the heuristic shrinkage as is commonly practiced with redundant transforms. We have shown that such method has origins in Bayesian denoising, being the first iteration of an iterative denoising algorithm. This leads to several consequences: i) we are now able to extend the heuristic shrinkage and get better denoising if more computations are allowed; ii) we obtain alternative shrinkage algorithms that use the transform and its adjoint, rather than its pseudoinverse; iii) the new interpretation may help in addressing the question of choosing the threshold in shrinkage, and how to adapt it to the various coefficients, and iv) the obtained algorithm can be used as an effective approximate solver for the BPDN for other applications, such as a nonlinear transform that promotes sparsity.

We should emphasize that these findings are not to be confused as a recommendation to use shrinkage for denoising in its simple form. Treating each transform coefficient alone is appealing because it is simple. However, recent work has shown that by treating clusters of coefficients, or exploiting the coefficients' interdependencies in other ways (e.g., hidden Markov models), could give a substantial improvement in the denoising effect [28]–[32], [34]–[39].

Acknowledgment

The author would like to thank Dr. Michael Zibulevsky, Dr. Doron Shaked, and Prof. Yaacov Hel-Or for insightful discussions that helped in making this a better manuscript.

References

[1] J. L. Rudin, S. Osher, and C. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.

[2] A. S. Carasso, "Linear and nonlinear image deblurring: A documented study," *SIAM J. Num. Anal.*, vol. 36, pp. 1659–1689, 1999.

[3] J. Weickert, B. M. Romeny, and M. A. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 398–410, Mar. 1998.

[4] M. Elad, "On the bilateral filter and ways to improve it," *IEEE Trans. Image Process.*, vol. 11, no. 10, pp. 1141–1151, Oct. 2002.

[5] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, Sept. 1994.

[6] D. L. Donoho, "Denoising by soft thresholding," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 613–627, May 1995.

[7] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard, "Wavelet shrinkage-asymptopia," *J. Roy. Statist. Soc. Ser. B—Methodological*, vol. 57, no. 2, pp. 301–337, 1995.

[8] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *J. Amer. Statist. Assoc.*, vol. 90, no. 432, pp. 1200–1224, Dec. 1995.

[9] D. L. Donoho, "Wedgelets: Nearly minimax estimation of edges," *Ann. Statist.*, vol. 27, no. 3, pp. 859–897, Jun. 1998.
[10] D. L. Donoho and I. M. Johnstone, "Minimax estimation via wavelet shrinkage," *Ann. Stat.*, vol. 26, no. 3, pp. 879–921, Jun. 1998.
[11] E. P. Simoncelli and E. H. Adelson, "Noise removal via Bayesian wavelet coring," in *Proc. Int. Conf. Image Processing*, Laussanne, Switzerland, Sep. 1996, pp. 379–382.
[12] P. Moulin and J. Liu, "Analysis of multiresolution image denoising schemes using generalized Gaussian and complexity priors," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 909–919, Apr. 1999.
[13] M. Jansen, *Noise Reduction by Wavelet Thresholding*. New York: Springer-Verlag, 2001.
[14] E. J. Candes and D. L. Donoho, "Recovering edges in ill-posed inverse problems: Optimality of curvelet frames," *Ann. Statist.*, vol. 30, no. 3, pp. 784–842, Jun. 2002.
[15] J.-L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Process.*, vol. 11, no. 6, pp. 670–684, Jun. 2002.
[16] J.-L. Starck, M. Elad, and D. L. Donoho, "Redundant multiscale transforms and their application for morphological component separation," *Adv. Imaging Electron Phys.*, vol. 132, pp. 287–348, 2004.
[17] M. N. Do and M. Vetterli, "Rotation invariant texture characterization and retrieval using steerable wavelet-domain hidden markov models," *IEEE Trans. Multimedia*, vol. 4, no. 4, pp. 517–527, Dec. 2002.
[18] M. N. Do and M. Vetterli, "Framing pyramids," *IEEE Trans. Signal Process.*, vol. 51, no. 9, pp. 2329–2342, Sep. 2003.
[19] P. Ishwar and P. Moulin, "Shift invariant restoration—An overcomplete maxent MAP framework," in *Proc. Int. Conf. Image Processing*, Vancouver, BC, Canada, 2000, vol. 3, pp. 270–272.
[20] M. N. Do and M. Vetterli, "The finite ridgelet transform for image representation," *IEEE Trans. Image Process.*, vol. 12, no. 1, pp. 16–28, Jan. 2003.
[21] P. Carre and E. Andres, "Discrete analytical ridgelet transform," *Signal Process.*, vol. 84, no. 11, pp. 2165–2173, Nov. 2004.
[22] M. Lang, H. Guo, and J. E. Odegard, "Noise reduction using undecimated discrete wavelet transform," *IEEE Signal Process. Lett.*, vol. 3, no. 1, pp. 10–12, Jan. 1996.
[23] R. Eslami and H. Radha, "The contourlet transform for image de-noising using cycle spinning," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2003, pp. 1982–1986.
[24] R. Eslami and H. Radha, "Translation-invariant contourlet transform and its application to image denoising," *IEEE Trans. Image Process.*, submitted for publication.
[25] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.
[26] L. A. Karlovitz, "Construction of nearest points in the ℓ^p, p even and ℓ^1 norms," *J. Approx. Theory*, vol. 3, pp. 123–127, 1970.
[27] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A re-weighted norm minimization algorithm," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 600–616, Mar. 1997.
[28] J. Scharcanski, C. R. Jung, and R. T. Clarke, "Adaptive image denoising using scale and space consistency," *IEEE Trans. Image Process.*, vol. 11, no. 9, pp. 1092–1101, Sep. 2002.
[29] C. R. Jung and J. Scharcanski, "Adaptive image denoising and edge enhancement in scale-space using the wavelet transform," *Pattern Recogn. Lett.*, vol. 24, no. 7, pp. 965–971, Apr. 2003.
[30] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Adaptive Wiener denoising using a gaussian scale mixture model in the wavelet domain," in *Proc. 8th Int. Conf. Image Process.*, Thessaloniki, Greece, Oct. 2001, vol. 2, pp. 37–40.
[31] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1338–1351, Nov. 2003.
[32] O. G. Guleryuz, "Weighted overcomplete denoising," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2003, vol. 2, pp. 1992–1996.
[33] A. G. Bruce, S. Sardy, and P. Tseng, "Block coordinate relaxation methods for nonparametric signal de-noising," *Proc. SPIE*, vol. 3391, pp. 75–86, 1998.
[34] S. G. Chang, B. Yu, and M. Vettereli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1532–1546, Sep. 2000.
[35] ——, "Wavelet thresholding for multiple noisy image copies," *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1631–1635, Sep. 2000.
[36] ——, "Spatially adaptive wavelet thresholding with context modeling for image denoising," *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1522–1531, Sep. 2000.
[37] P. Mrázek and J. Weickert, "Rotationally invariant wavelet shrinkage," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2003, vol. 2781, pp. 156–163.
[38] G. Steidl, J. Weickert, T. Brox, P. Mrázek, and M. Welk, "On the equivalence of soft wavelet shrinkage, total variation diffusion, total variation regularization, and SIDEs," *SIAM J. Numer. Anal.*, vol. 42, no. 2, pp. 686–713, 2004.
[39] J. Weickert, G. Steidl, P. Mrázek, M. Welk, and T. Brox, "Diffusion filters and wavelets: What can they learn from each other?," in *Mathematical Models of Computer Vision: The Handbook*, N. Paragios, Y. Chen, and O. Faugeras, Eds. Berlin, Germany: Springer-Verlag, 2005.
[40] J.-L. Starck, E. Candes, and D. L. Donoho, "Astronomical image representation by the curvelet transform," *Astron. Astrophys.*, vol. 398, pp. 785–800, 2003.
[41] J.-L. Starck, M. Elad, and D. L. Donoho, "Redundant multiscale transforms and their application for morphological component analysis," *J. Adv. Imag. Electron Phys.*, vol. 132, pp. 287–348, 2004.
[42] M. A. Figueiredo and R. D. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 906–916, Aug. 2003.
[43] I. Daubechies, M. Defrise, and C. De-Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure and Applied Math.*, vol. LVII, pp. 1413–1457, 2004.
[44] B. Matalon, M. Elad, and M. Zibulevsky, "Improved denoising of images using modeling of the redundant contourlet transform," in *Proc. SPIE Conf. Wavelets*, San Diego, CA, Jul. 2005, vol. 5914.