A Multilevel Iterated-Shrinkage Approach to l_1 Penalized Least-Squares Minimization

Eran Treister and Irad Yavneh

***Abstract*—The area of sparse approximation of signals is drawing tremendous attention in recent years. Typically, sparse solutions of underdetermined linear systems of equations are required. Such solutions are often achieved by minimizing an l_1 penalized least squares functional. Various iterative-shrinkage algorithms have recently been developed and are quite effective for handling these problems, often surpassing traditional optimization techniques. In this paper, we suggest a new iterative multilevel approach that reduces the computational cost of existing solvers for these inverse problems. Our method takes advantage of the typically sparse representation of the signal, and at each iteration it adaptively creates and processes a hierarchy of lower-dimensional problems employing well-known iterated shrinkage methods. Analytical observations suggest, and numerical results confirm, that this new approach may significantly enhance the performance of existing iterative shrinkage algorithms in cases where the matrix is given explicitly.**

I. Introduction

Sparse approximation of signals is an emerging area of research that is drawing vast interest and finding use in numerous applications. One popular application is sparse representation of signals and images, where the key underlying observation is that natural signals, such as images, admit sparse decompositions over specific spatial transforms [1], [2]. Another popular application is known as *compressive sensing* [3], [4], [5] where signals are reconstructed from only a few linear measurements. Other applications include statistical analysis, machine learning, and coding theory [6].

There has been an enormous effort in recent years to develop mathematical formulations and computational methods for applying such reconstructions. The simplest way to mathematically formulate this idea is to assume that the sought signal $\mathbf{y} \in \mathbb{R}^n$ can be approximately represented by only a few columns of a matrix $A \in \mathbb{R}^{n\times m}$. That is, $\mathbf{y} = A\mathbf{x}$, where the *representation vector* $\mathbf{x} \in \mathbb{R}^m$ is sparse, containing few non-zero elements. The matrix A, often called the *dictionary*, is usually over-complete, having more columns than rows, $m > n$. This means that the underdetermined system $A\mathbf{x} = \mathbf{y}$ has infinitely many solutions, and we seek the sparsest one by solving the problem

$$\min_{\mathbf{x}\in\mathbb{R}^m} \|\mathbf{x}\|_0 \quad \text{subject to} \quad A\mathbf{x} = \mathbf{y}, \tag{1}$$

The authors are with the Department of Computer Science, The Technion—Israel Institute of Technology, Haifa 32000, Israel.
Contact email: eran@cs.technion.ac.il, irad@cs.technion.ac.il.

Research supported by the Israeli Science Foundation, grant number 795/08. Eran Treister is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

where the sparseness measure $\|\mathbf{x}\|_0 = |\{i : x_i \neq 0\}|$ is called the l_0 quasi-norm, defined as the number of non-zero elements in the vector $\mathbf{x}$. There are alternative formulations based on the l_0 quasi-norm. However, these optimization problem are non-convex and generally very hard to solve, as their solution usually requires an intractable combinatorial search [7]. Nevertheless, the solution for such problems can be approximated using so-called "greedy algorithms" such as (Orthogonal) Matching Pursuit (OMP/MP) [8], [9], [10], [11], [12], Stagewise OMP (StOMP) [13], CoSAMP [14], Subspace Pursuit (SP) [15], iterative hard thresholding [16], [17], [18], [19], [20], [21], and others.

A common alternative approach is to relax (1) by replacing the l_0 quasi-norm with the well-known l_1 norm, which has somewhat similar "sparsity properties" [3], [5], [22], [24]. The new problem,

$$\min_{\mathbf{x}\in\mathbb{R}^m} \|\mathbf{x}\|_1 \quad \text{subject to} \quad A\mathbf{x} = \mathbf{y}\,, \tag{2}$$

called Basis Pursuit [23], is convex. Its solution may not be unique, but if more than one solution exists then all the solutions belong to a convex and compact set; that is, any convex combination of the basic solutions is itself a solution [24]. A typical solution vector $\mathbf{x}^*$ of (2) is relatively sparse, and under certain conditions it is in fact equal to a global minimizer of (1). Problem (2) can be formulated and solved as a linear programming problem [23].

Because observed signals typically contain some noise, which has no sparse representation, the constraint $A\mathbf{x} = \mathbf{y}$ is usually relaxed in both (1) and (2), with approximate equality measured using the quadratic penalty function $\|A\mathbf{x} - \mathbf{y}\|_2$, where $\mathbf{y}$ henceforth denotes the observed noisy signal. Two other approaches to treat this case are LASSO [25], and the common Basis Pursuit denoising (BPDN) [23]. The latter features an l_1 penalized least-squares functional minimization:

$$\min_{\mathbf{x}\in\mathbb{R}^m} F(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^m} \frac{1}{2}||A\mathbf{x} - \mathbf{y}||_2^2 + \mu\|\mathbf{x}\|_1, \tag{3}$$

with μ a scalar parameter that balances between sparsity and adherence to the data. Generally, a larger parameter μ yields a sparser minimizer $\mathbf{x}^*$, but also a greater discrepancy $||A\mathbf{x}^* - \mathbf{y}||_2^2$. Although this problem is an unconstrained convex optimization problem, traditional optimization methods, such as gradient descent or quasi-Newton methods, tend to be slow due to the discontinuity of the gradient, which arises from using the l_1 norm. Therefore, various computational optimization methods were developed for the task. The most common methods are the so-called "iterative shrinkage" or "iterative soft thresholding (IST)" methods that are often used together with some accelerations [26], [27], [28], [29],

[30], [31], [32], [33], [34], [35]. Other related approaches for solving (3) include [36], [37], [38], [39], [40], [41], [42]. Similarly to (2), problem (3) may also have more than one globally optimal solution. In this work we adopt the common practice of seeking any one of those solutions and refer to it as "the minimizer" of (3), denoted by $\mathbf{x}^*$.

This paper introduces a straightforward multilevel method for l_1 penalized least-squares problems like (3), based on the main concept of classical algebraic multigrid methods [43]; that is, we accelerate the convergence of simple iterative methods for (3) using a nested hierarchy of smaller versions of the problem. Multigrid methods are commonly applied to linear systems arising from discretization of partial differential equations as well as other ill-conditioned systems. In many cases algebraic multigrid methods enable us to treat such problems effectively regardless of their condition number. This is done by projecting the original problem to a lower-dimensional subspace that contains the error components that are not treated effectively by standard iterative methods, such as Jacobi and Gauss-Seidel. Then, these error components are corrected by solving a lower-dimensional problem. This correction together with the standard iterative methods serve as two complementary processes which combine to yield an effective solver. The idea of introducing multigrid-like methods for (3) has yet to be explored and it has great potential. In this work we follow the idea of "multiplicative correction" multigrid methods, which exploit a hierarchy of approximate operators that evolve with the solution process, eventually becoming exact—see [44], [45] and references therein.

In classical multigrid methods, the aim is to define a multilevel solver with optimal *asymptotic* convergence behavior, because the asymptotic convergence rates of simpler iterative solvers tend to be slow for problems of interest. However, the present problem is different as the main challenge is finding the non-zero elements of the minimizer $\mathbf{x}^*$ and its sign-pattern. Therefore, our algorithm is different from the classical multigrid approach. At each iteration (called a "multilevel V-cycle") it reduces the dimension of the problem and creates a multilevel hierarchy of smaller and smaller problems, involving a lower dimensional dictionary at each "level". We take advantage of the typical sparsity of $\mathbf{x}$ and reduce the dimension of the problem (3) by ignoring ostensibly irrelevant columns from A. That is, each low-level problem is defined by (3), restricted to a specially chosen subset of the columns of A, resulting in a nested hierarchy of sub-dictionaries. It then performs sub-space correcting shrinkage sweeps over each of the low dimensional problems in turn, that aim to activate the atoms that comprise the support of a true minimizer. Under suitable conditions, our algorithm converges to the global minimizer of (3)—we do not compromise solution quality in return for improved performance.

II. Iterated shrinkage methods and accelerations

Many of the simple iterated shrinkage methods for solving (3) are of the form

$$\mathbf{z} = \mathcal{S}_{\mu/c}\left(\frac{1}{c}A^T(\mathbf{y} - A\mathbf{x}^k) + \mathbf{x}^k\right), \tag{4}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha(\mathbf{z} - \mathbf{x}^k), \tag{5}$$

where $\mathbf{x}^k$ is the approximate solution at the k-th iteration, $\alpha > 0$ is a line-search scalar, $c > 0$ is method dependent, and

$$\mathcal{S}_q(t) = \text{sign}(t) \cdot \max(0, |t| - q) \tag{6}$$

is the "soft shrinkage" function, so dubbed because the size of the argument t is reduced by q (or set to zero if $q > |t|$). Some methods, such as [28], [38], simply choose $\alpha = 1$ in (5), i.e., $\mathbf{x}^{k+1} = \mathbf{z}$. Others, such as [31], [34], apply a line-search by choosing α that minimizes the functional (3) of $\mathbf{x}^{k+1}$ according to the search direction (5).

The constant $c > 0$ in (4) can be chosen in different ways and is the main difference between the various shrinkage methods. For example, in SSF [28], c is chosen such that $c \geq \rho(A^TA)$; in SpaRSA [38] c is recalculated at each iteration via a linesearch over $F(\mathbf{z})$ defined in (3); in [34] c is adapted throughout the iterations, and in PCD [31] $1/c$ is replaced by the inverse of a diagonal matrix D which is equal to the diagonal of A^TA.

The iterative shrinkage methods mentioned above are much faster than standard first-order minimization methods such as classical Steepest Descent, however, some of them may remain relatively slow, requiring acceleration. In [31], [35] an optimization method called "sequential subspace optimization" (SESOP) was considered, together with the PCD and SSF iterations. Other acceleration methods include FISTA [46], TwIST [32], and a version of nonlinear Conjugate Gradients (CG) [35]. In this paper, we use a non-linear CG approach that is slightly different from that of [35], and is also based on the Polak-Ribiere CG method [47]. Given a search direction $\mathbf{r}^k = \mathbf{z} - \mathbf{x}^k$ obtained in (5) at the k-th iteration, we compute $\mathbf{x}^{k+1}$ via a line-search along the direction $\mathbf{r}^k = \mathbf{r}^k + \beta\mathbf{r}^{k-1}$, where

$$\beta = \max\left\{\frac{(\mathbf{r}^k)^T(\mathbf{r}^k - \mathbf{r}^{k-1})}{\|\mathbf{r}^{k-1}\|_2^2}, 0\right\}.$$

Another type of acceleration technique that can be used with the above methods is the "continuation" strategy, also known as "warm-start", that is applied in [33], [34], [36], [38]. In this approach, a large parameter μ is first chosen for the problem (3) (lower than but proportional to $\|A^T\mathbf{y}\|_\infty$). Once (3) is solved approximately for this value of μ, it is gradually decreased, and at each stage the new initial guess is given by the approximate solution to (3) obtained with the previous, bigger, parameter μ. This way, a sequence of problems is solved corresponding to the sequence of the decreasing μ's, until some convergence criterion is satisfied.

Asymptotically, once the sign-pattern of the minimizer $\mathbf{x}^*$ is recovered and fixed by the iterations, the functional in (3) becomes quadratic and can normally be solved efficiently either by the CG method or directly by solving the system that corresponds only to the non-zeros of the vector $\mathbf{x}^*$. Furthermore, in most applications the l_1 norm is only used as a regularizer for promoting sparsity of $\mathbf{x}$, and so, once we determine the support of $\mathbf{x}^*$, we may ignore the regularization term in (3) and only consider the quadratic term [36], [38] (an approach known as "debiasing"). Overall, the main effort in

solving (3) is invested in determining the non-zero elements of the minimizer $\mathbf{x}^*$ and their signs.

III. A MULTILEVEL ITERATED SHRINKAGE APPROACH

We next describe our new multilevel approach for solving (3). At each iteration, called a "V-cycle", we define a hierarchy of reduced problems, referred to as *low-level problems*. Each low-level problem is defined by (3), restricted to a specially chosen subset of the columns of A, and in each V-cycle we traverse the entire hierarchy of levels. We iteratively repeat these V-cycles, reducing the functional of (3) at each one, until some convergence criterion is satisfied. A precise description is given in the following sections in a two-level framework, with the extension to the multi-level framework obtained by recursion. In this description, all elements that are related to the low-level problem are denoted by a subscript c. Keeping with common practice, we use the terms "matrix" and "dictionary" interchangeably, and similarly the terms "column" and "atom".

A. Definition of the low-level problem

In this subsection we define the reduced problem given its designated subset of atoms, $\mathcal{C} \subset \{1, ..., m\}$, while the choice of $\mathcal{C}$ will be discussed later. Given $\mathcal{C}$, we define a so-called prolongation matrix $P \in \mathbb{R}^{m \times |\mathcal{C}|}$, that transfers a low-level vector $\mathbf{x}_c \in \mathbb{R}^{|\mathcal{C}|}$ into an upper-level vector $\mathbf{x} \in \mathbb{R}^m$ by the relation $\mathbf{x} = P\mathbf{x}_c$. We choose P to be a zero-filling operator, which zeros the elements of $\mathbf{x}$ that do not belong to $\mathcal{C}$, while retaining the values of $\mathbf{x}_c$ in the elements that do belong to $\mathcal{C}$. We have found this simple approach to be effective, but more sophisticated choices of P may be worthy of investigation for more general problems.

Next, we restrict (3) onto the atoms in $\mathcal{C}$, or more generally, onto the range of P. That is, we substitute $P\mathbf{x}_c$ for $\mathbf{x}$ in the objective (3), and get the new problem:

$$\begin{aligned} &\min_{\mathbf{x}_c \in \mathbb{R}^{|\mathcal{C}|}} \; F_c(\mathbf{x}_c) \equiv \min_{\mathbf{x}_c \in \mathbb{R}^{|\mathcal{C}|}} \; F(P\mathbf{x}_c) = \\ &\min_{\mathbf{x}_c \in \mathbb{R}^{|\mathcal{C}|}} \; \frac{1}{2}\|AP\mathbf{x}_c - \mathbf{y}\|_2^2 + \mu\|P\mathbf{x}_c\|_1, \end{aligned} \tag{7}$$

which has only $|\mathcal{C}|$ degrees of freedom. Since our P is zero-filling, we have that $\|P\mathbf{x}_c\|_1 = \|\mathbf{x}_c\|_1$ holds for all $\mathbf{x}_c$, and therefore we can write

$$\min_{\mathbf{x} \in \mathbb{R}^{|\mathcal{C}|}} \; F_c(\mathbf{x}_c) = \min_{\mathbf{x}_c \in \mathbb{R}^{|\mathcal{C}|}} \; \frac{1}{2}\|A_c\mathbf{x}_c - \mathbf{y}\|_2^2 + \mu\|\mathbf{x}_c\|_1, \tag{8}$$

where $A_c = AP$ is the reduced sub-dictionary of the upper-level dictionary A, with columns given by the columns of A corresponding to the indices in $\mathcal{C}$. Note that if $\mathcal{C}$ contain the support of the true minimizer of (3), and (8) is solved exactly, then $P\mathbf{x}_c$ is in fact a solution of (3). Furthermore, because this problem is similar to (3), we can recursively extend this two-level framework to multi levels.

B. Choosing the low-level variables

Our low-level definition above suggests that we need to select a subset of low-level variables, $\mathcal{C}$, that is as likely as possible to contain the support of the true minimizer. Therefore, for choosing $\mathcal{C}$ we use the approximate solution at the k-th iteration, $\mathbf{x}^k$, which is the best one currently available. Let

$$\text{supp}(\mathbf{x}) = \{i : x_i \neq 0\},$$

denote the support of any vector $\mathbf{x}$. Then evidently, if $\text{supp}(\mathbf{x}^k) \subseteq \mathcal{C}$, then $\mathbf{x}^k$ is in the range of P. Indeed, $\mathbf{x}^k = P\mathbf{x}_c$ where $\mathbf{x}_c$ is the vector $\mathbf{x}^k$ restricted to the indices in $\mathcal{C}$. Therefore, we start by requiring $\text{supp}(\mathbf{x}^k) \subseteq \mathcal{C}$, so that by (7)-(8) we have that $F(\mathbf{x}^k) = F_c(\mathbf{x}_c)$ holds. This implies that the prolongation matrix P changes during the iterations, depending on $\mathbf{x}^k$.

Next, we decide on the additional atoms in $\mathcal{C}$, besides those in $\text{supp}(\mathbf{x}^k)$, aiming to limit its size to $|\mathcal{C}| = \lceil m/2 \rceil$ (this choice will be discussed later). If $|\text{supp}(\mathbf{x}^k)| \geq \lceil m/2 \rceil$, then we choose $\mathcal{C} = \text{supp}(\mathbf{x}^k)$. Otherwise (the common case) we add $\lceil m/2 \rceil - |\text{supp}(\mathbf{x}^k)|$ atoms that are currently not in $\text{supp}(\mathbf{x}^k)$, and yet have a relatively good chance of being in the support of the true solution $\mathbf{x}^*$. These correspond to atoms i with a relatively large value of $|\mathbf{a}_i^T(A\mathbf{x}^k - \mathbf{y})|$, since including them in the support reduces the first term in the functional of (3) more significantly per given increase in the second term (see also Proposition 5 below). This rationale is commonly employed in existing "greedy algorithms" mentioned earlier. This leads to the following definition of $\mathcal{C}$ at the k-th iteration:

$$\mathcal{C} = \text{supp}(\mathbf{x}^k) \cup \text{likely}(\kappa),$$

where $\kappa = \max\{\lceil m/2 \rceil - |\text{supp}(\mathbf{x}^k)|, 0\}$, $\text{likely}(0) = \emptyset$, and $\text{likely}(\kappa)$ is the set of indices of the κ largest elements in the likelihood vector $|A^T(A\mathbf{x}^k - \mathbf{y})|$.

C. Definition of the multi-level V-cycle

For solving (3), we repeat

$$\mathbf{x}^{k+1} = \textbf{V-cycle}(A, \mathbf{x}^k, \mathbf{y}, \nu) \tag{9}$$

iteratively, until some convergence criterion is satisfied. The multilevel V-cycle() procedure, along with its parameters, is defined in Algorithm 1. The algorithm creates a reduced version of the problem (3) as described above, and then treats it recursively, yielding a hierarchy of smaller and smaller problems. The recursion is terminated (Step 3a) when one of the following happens. The common base case is when $|\text{supp}(\mathbf{x})| \geq \lceil m/2 \rceil$— then we cannot reduce the problem further. In this case we choose $\mathcal{C} = \text{supp}(\mathbf{x})$, and solve the problem (8) directly. The second base case is when the problem becomes sufficiently small and can be solved easily. In practice, we choose a minimal number of allowable columns m_{min}, and if $|\mathcal{C}| < 2m_{min}$ then we process (8) directly rather than continuing recursively. (In our tests we use $m_{min} = 10$.).

The algorithm uses iterated shrinkage methods as so-called "relaxations"—the usual name for the iterations employed within multilevel algorithms—carrying out ν such relaxations at each level ($\nu = 1$ in our tests). All shrinkage methods of the form (4), as well as most other shrinkage methods, can be incorporated into this multilevel approach.

In terms of cost, if we reduce the number of unknowns by a factor of two at each level ($|\mathcal{C}| = \lceil m/2 \rceil$), then the total cost of

Algorithm: $\mathbf{x} \leftarrow$ **V-cycle**$(A, \mathbf{x}, \mathbf{y}, \nu)$
%Iterative Shrinkage method: $Relax(A, \mathbf{x}, \mathbf{y})$.
%Number of relaxations at each level: ν.
%Minimal number of columns allowed: m_{min}.

1) Choose the low-level variables $\mathcal{C}$ and define the prolongation P.
2) Define the low-level dictionary A_c and approximation $\mathbf{x}_c = P^T\mathbf{x}$.
3) **If** $\mathcal{C} = \text{supp}(\mathbf{x})$ or $|\mathcal{C}| < 2m_{min}$,
   a) Solve the lowest-level problem (8).
   Else $\mathbf{x}_c \leftarrow$ **V-cycle**$(A_c, \mathbf{x}_c, \mathbf{y}, \nu)$ *% Recursive call*
4) Prolong solution: $\mathbf{x} \leftarrow P\mathbf{x}_c$ *% Solution update*.
5) Apply ν relaxations: $\mathbf{x} \leftarrow Relax(A, \mathbf{x}, \mathbf{y})$.

Algorithm 1: V-cycle for l_1 penalized LS minimization

a V-cycle (excluding the treatment of the lowest level) is only about twice the cost of ν iterated shrinkage relaxations on the highest level. This means that, although we include a relatively large fraction of the atoms in the next low level, the cost of the entire V-cycle remains relatively small, possibly excluding the lowest level solution. The latter is a key component of our algorithm, and it will be discussed later.

Remark 1: As one can see, our algorithm can be readily applied when the columns of A can be easily extracted. This requires A to be given explicitly as a matrix, rather than a fast transform operator as in the Wavelet or DCT transforms. Such explicit dictionaries are mostly used in the context of "trained dictionaries", where they may be either sparse or dense; see [48], [49], [50], [51], [52], [53] and references therein. Furthermore, in many applications the dictionary A takes the form

$$A = H \cdot B, \tag{10}$$

where H is an operator that acts on the columns of the matrix B [35]. For example, in the image deblurring problem the operator H is a low-pass filter while B is the underlying dictionary. In this scenario, our algorithm is applicable if either H or B are given as explicit matrices, and the other is cheap to apply. In most cases, it is the matrix B which is explicit.

Remark 2: Algorithm 1 is a multilevel framework wrapped around a chosen iterative shrinkage (soft thresholding) method. In spirit, it bears some relation to accelerated iterative hard thresholding [19], [21] and subspace pursuit [14], [15] methods. However, the objectives, the techniques, and the available theoretical observations, are quite different. At the k-th iteration, these algorithms update the solution by minimizing $||A\mathbf{x}^k - \mathbf{y}||_2$ with respect to a small subspace restricting the support of $\mathbf{x}^k$. This is similar to the lowest-level solution in Step 3a of our V-cycle, but with a different objective. In [14], [15], a sparse approximation with a small, fixed, support size is sought, and the selection of the atoms is done according to the largest elements in $A^T(A\mathbf{x}^k - \mathbf{y})$. In [19], [21], the selection of atoms is done by iterative hard thresholding.

In our approach we target l_1 penalized problems, as opposed to l_0 penalized problems in [19], [21] and support-size constrained problems in [14], [15]. We use the likelihood criterion $A^T(A\mathbf{x}^k - \mathbf{y})$ only to create a hierarchy of subspaces (low-level dictionaries), while the iterated shrinkage (relaxation) method is responsible for selecting the atoms assumed to belong to the support. We elaborate on this mechanism in the next section. Unlike these methods, the subspaces in our hierarchy are very large (until we get to the lowest level) and the problem we target at each level is non-quadratic, and is analogous to the fine-level problem.

D. Theoretical properties of the V-cycle

This section is devoted to theoretical performance observations regarding Algorithm 1. In some cases we refer to its two-level version, which differs from the original algorithm only in Step 3. There, the recursive call is replaced with an exact solution of the problem (8) on the second level, that is, the condition in Step 3 is removed and Step 3a is always performed.

In the following discussions we use the iteration specific notation $\mathbf{x}^k$ and the generic notation $\mathbf{x}$, depending on the context. Both relate to an *approximate* solution to (3). Also, as before, we denote a solution to the problem (3) by $\mathbf{x}^*$, and assume that it is a stationary point of any relaxation method.

By (7), together with $\text{supp}(\mathbf{x}^k) \subseteq \mathcal{C}$, we can write

$$F_c(\mathbf{x}_c) = F(P\mathbf{x}_c) = F_c(P^T\mathbf{x}^k) = F(\mathbf{x}^k), \tag{11}$$

from which the next two observations follow immediately.

Proposition 1: (Inter-level Correspondence.) Let $\mathbf{x}$ be an approximation to the minimizer of F in (3). Let $\mathbf{z}_c$ be a better approximation than $\mathbf{x}_c$ to the minimizer of the low-level problem (8), such that $F_c(\mathbf{z}_c) < F_c(\mathbf{x}_c)$. Then $F(P\mathbf{z}_c) < F(\mathbf{x})$.

Proposition 2: (Direct Solution.) If $\mathcal{C} \supseteq \text{supp}(\mathbf{x}^*)$, then $\mathbf{x}_c^* = P^T\mathbf{x}^*$ is a solution of the low-level problem (8), and the two-level Algorithm 1 solves problem (3) in one V-cycle.

Equation (11) relates the original problem (3) to any of the problems (8) on any level of the V-cycle. Let the original problem (3) define level 0, and let P_i be a prolongation from level i to level $i-1$. Then (11) holds for $F_c(\mathbf{x}_c)$ on any level i together with the so-called "composite prolongation",

$$P_i^c = P_1 \cdot P_2 \cdots P_i, \tag{12}$$

that transfers from level i to level 0. Using P_i^c, Proposition 1 holds for any level i of the V-cycle. The next corollary is an extension of Proposition 2 to the multilevel case, and its proof follows immediately from Proposition 2 using the composite prolongation (12).

Corollary 1: If $\text{supp}(\mathbf{x}) \supseteq \text{supp}(\mathbf{x}^*)$ or $\mathcal{C} \supseteq \text{supp}(\mathbf{x}^*)$ on the lowest level, then Algorithm 1 solves problem (3) in one V-cycle. Furthermore, $\mathbf{x}^*$ is a stationary point of Algorithm 1.

For the following propositions we use the notion of sub-gradients [24]. $\partial F(\mathbf{x})$, the sub-differential of F, is a non-empty set of sub-gradients,

$$\partial F(\mathbf{x}) = \left\{ A^T(A\mathbf{x} - \mathbf{y}) + \mu\mathbf{z} : \mathbf{z} \in \mathcal{A}(\mathbf{x}) \right\}, \tag{13}$$

where $\mathcal{A}(\mathbf{x})$ is a set of all vectors $\mathbf{z} \in \mathbb{R}^m$ whose elements satisfy

$$\begin{aligned} z_i &= \text{sign}(x_i) \quad \text{if } x_i \neq 0\,, \\ z_i &\in [-1,1] \quad \text{if } x_i = 0\,. \end{aligned} \tag{14}$$

A vector $\mathbf{x}^*$ is a minimizer of (3), if and only if $0 \in \partial F(\mathbf{x}^*)$. That is, for all elements of $\mathbf{x}^*$, if $x_i^* \neq 0$ then $\mathbf{a}_i^T(A\mathbf{x}^* - \mathbf{y}) + \mu\text{sign}(x_i^*) = 0$, and otherwise $|\mathbf{a}_i^T(A\mathbf{x}^* - \mathbf{y})| \leq \mu$. Using this, we next show that the two-level algorithm does not stagnate.

Proposition 3: (No Stagnation.) If $\mathcal{C} \not\supseteq \text{supp}(\mathbf{x}^*)$, and $\mathbf{x} = P\mathbf{x}_c$ is the updated upper-level solution after Step 4 of the two-level Algorithm 1, then a single Iterated Shrinkage relaxation as in (4) on $\mathbf{x}$ must cause at least one atom to be added to $\text{supp}(\mathbf{x})$.

Proof: Since $\mathbf{x}_c$ is a minimizer of the low-level functional, then $0 \in \partial F_c(\mathbf{x}_c)$, and since A_c is comprised of a subset of the columns of A, and $A\mathbf{x} = AP\mathbf{x}_c = A_c\mathbf{x}_c$, then $0 \in \partial F_c$ means that for all $j \in \mathcal{C}$

$$\begin{aligned} &\mathbf{a}_j^T(A\mathbf{x} - \mathbf{y}) + \mu\text{sign}(x_j) = 0 \quad \text{if } x_j \neq 0\,, \\ &|\mathbf{a}_j^T(A\mathbf{x} - \mathbf{y})| \leq \mu \qquad\qquad\quad \text{if } x_j = 0\,. \end{aligned} \tag{15}$$

Now, since $\mathcal{C} \not\supseteq \text{supp}(\mathbf{x}^*)$, then $\mathbf{x}$ is not a minimizer of the upper-level functional, so $0 \notin \partial F(\mathbf{x})$. Therefore, there exists at least one variable $\ell \notin \mathcal{C}$ for which $|\mathbf{a}_\ell^T(A\mathbf{x} - \mathbf{y})| > \mu$. Since $x_\ell = 0$, then, according to (4), after one Iterated Shrinkage relaxation, index ℓ will enter the support of $\mathbf{x}$. ∎

From the last two propositions we can see the complementary roles of the relaxation and solution update in Algorithm 1. The relaxation is largely responsible for inserting the correct atoms into the support, while the solution update is mainly responsible for finding the optimal values of the variables that are in the support.

For the next proposition we define the term "memory-less monotonic iteration" (MLMI). An iterated shrinkage relaxation $T(\mathbf{x})$ is called MLMI if

$$F(\mathbf{x}) - F(T(\mathbf{x})) \geq K \cdot \min_{\mathbf{z} \in \mathcal{A}(\mathbf{x})} (\|\partial F(\mathbf{x})\|^2) \quad \forall \mathbf{x} \in \mathbb{R}^m, \tag{16}$$

where K is a positive constant and $\partial F(\mathbf{x})$ is the sub-differential of F defined in (13) and (14). This definition implies that every iteration of $T(\mathbf{x})$ reduces the functional $F(\mathbf{x})$ in (3) at worst proportionally to the size of $\min_{\mathbf{z} \in \mathcal{A}(\mathbf{x})}(\|\partial F(\mathbf{x})\|^2)$. For example, one such method is SSF [28], which was shown to be both MLMI and convergent under some conditions (see Apendix B in [31]). We next prove an auxiliary Lemma that will help us to show that Algorithm 1 is convergent under suitable conditions.

Lemma 1: (Monotonicity.) Assume that Algorithm 1 is used together with $\nu > 0$ MLMI relaxations T, and is applied on $\mathbf{x}$. Let $Q(\mathbf{x})$ be the solution update in Step 4 of Algorithm 1 on level 0. Then $F(\mathbf{x}) \geq F(Q(\mathbf{x}))$.

Proof: The proof follows from Equation (11), which holds for any level i via the corresponding composite prolongation (12). Since an exact minimization is performed on the lowest level, and since the relaxations that are used on each level are MLMI, the values of consecutive low-level functionals F_c cannot increase as we traverse the V-cycle hierarchy. ∎

We now show that Algorithm 1 is convergent under suitable conditions, following the idea of the convergence proof of SSF in [31].

Proposition 4: (Convergence.) Assume that the level set $\mathcal{R} = \{\mathbf{x} : F(\mathbf{x}) \leq F(\mathbf{x}^0)\}$ is compact. Also, assume that Algorithm 1 is applied with $\nu > 0$ MLMI relaxations T. Let $\{\mathbf{x}^k\}$ be a series of points produced by Algorithm 1, i.e., $\mathbf{x}^{k+1} = \text{V-cycle}(A, \mathbf{x}^k, \mathbf{y}, \nu)$, starting from an initial guess $\mathbf{x}^0$. Then any limit point $\mathbf{x}^*$ of the sequence $\{\mathbf{x}^k\}$ is a stationary point of F in (3), i.e., $0 \in \partial F(\mathbf{x}^*)$, and $F(\mathbf{x}^k)$ converges to $F(\mathbf{x}^*)$.

Proof: We start by showing that the series $\{F(\mathbf{x}^k)\}$ is monotonically decreasing. By Lemma 1 we have that $F(\mathbf{x}^k) \geq F(Q(\mathbf{x}^k))$ where $Q(\mathbf{x}^k)$ is the solution update on level 0. Then, following the algorithm, we apply ν relaxations on $Q(\mathbf{x}^k)$. By (16), and $\nu > 0$, we can bound

$$\begin{aligned} F(\mathbf{x}^k) - F(\mathbf{x}^{k+1}) &\geq F(\mathbf{x}^k) - F(T(Q(\mathbf{x}^k))) \\ &\geq F(Q(\mathbf{x}^k)) - F(T(Q(\mathbf{x}^k))) \\ &\geq K \min_{\mathbf{z} \in \mathcal{A}(Q(\mathbf{x}^k))} \|\partial F(Q(\mathbf{x}^k))\|^2, \end{aligned} \tag{17}$$

which implies that $\{F(\mathbf{x}^k)\}$ is monotonically decreasing.

Since the functional F in (3) is non-negative, then it is bounded from below, and hence the series $\{F(\mathbf{x}^k)\}$ converges to a limit. Because the level set $\mathcal{R}$ is compact by assumption, we have that $\{\mathbf{x}^k\}$ is bounded in $\mathcal{R}$, and therefore there exists a sub-series $\{\mathbf{x}^{k_n}\}$ converging to a limit point $\mathbf{x}^*$.

Assume to the contrary that $\mathbf{x}^*$ is not stationary, i.e., $0 \notin \partial F(\mathbf{x}^*)$. It is known that $\partial F(\mathbf{x})$ is always a non-empty convex compact set, which means that $\min_{\mathbf{z} \in \mathcal{A}(\mathbf{x})} \|\partial F(\mathbf{x})\|$ always exists. That, together with $0 \notin \partial F(\mathbf{x}^*)$ ensures that $\min_{\mathbf{z} \in \mathcal{A}(\mathbf{x}^*)} \|\partial F(\mathbf{x}^*)\| > 0$. Denote $\min_{\mathbf{z} \in \mathcal{A}(\mathbf{x}^*)} \|\partial F(\mathbf{x}^*)\| = \epsilon$. Since $\mathbf{x}^{k_n} \to \mathbf{x}^*$, then there are infinitely many k_n's satisfying $\min_{\mathbf{z} \in \mathcal{A}(\mathbf{x}^{k_n})} \|\partial F(\mathbf{x}^{k_n})\| = \epsilon$. By (17) we have infinitely many k_n's satisfying

$$F(\mathbf{x}^{k_n}) - F(\mathbf{x}^{k_{n+1}}) \geq K \cdot \epsilon^2, \tag{18}$$

which contradicts the fact that F is bounded from below. This shows that the point $\mathbf{x}^*$ is stationary. Since F is continuous, $\mathbf{x}^{k_n} \longrightarrow \mathbf{x}^*$ yields $F(\mathbf{x}^{k_n}) \longrightarrow F(\mathbf{x}^*)$. The limit of $\{F(\mathbf{x}^k)\}$ equals to that of any of its sub-series, specifically $\{F(\mathbf{x}^{k_n})\}$, and thus $F(\mathbf{x}^k) \longrightarrow F(\mathbf{x}^*)$. ∎

The next proposition justifies our criterion for the choice of the set $\mathcal{C}$ of atoms comprising the low level. This result is in the spirit of thresholding algorithms, as well as other greedy algorithms, which apply the same approach for selecting the support [24].

Proposition 5: ($\mathcal{C}$-Selection Guarantee.) Without loss of generality, assume that the columns of A are normalized such that $diag(A^TA) = \mathbf{1}$, and then let δ be the so-called *mutual coherence* of the dictionary A, defined by

$$\delta = \max_{i \neq j} \{|\mathbf{a}_i^T \mathbf{a}_j|\}.$$

Let $\mathbf{x}^*$ and $\mathbf{x}$ be the solution and the approximate solution at the time $\mathcal{C}$ is chosen, respectively, and let $\mathbf{e} = \mathbf{x}^* - \mathbf{x}$ be the current error. Let i be an index satisfying $i \in \text{supp}(\mathbf{x}^*)$ and $i \notin \text{supp}(\mathbf{x})$. Then, so long as $|\mathcal{C}| > |\text{supp}(\mathbf{x})|$, index i

is guaranteed to be included in the set $\mathcal{C}$ prior to any index $\ell \notin \text{supp}(\mathbf{x}) \cup \text{supp}(\mathbf{x}^*)$ if

$$|x_i^*| \geq \frac{2\delta}{1+\delta}\|\mathbf{e}\|_1 \,. \tag{19}$$

Proof: The choice of $\mathcal{C}$ is based on the likelihood measure. We derive a condition that guarantees that index i be included in $\mathcal{C}$ prior to *any* index ℓ not belonging to $\text{supp}(\mathbf{x}^*)$, i.e.,

$$|\mathbf{a}_i^T(A\mathbf{x}-y)| > |\mathbf{a}_\ell^T(A\mathbf{x}-y)| \tag{20}$$

for *any* $\ell \notin \text{supp}(\mathbf{x}) \cup \text{supp}(\mathbf{x}^*)$. To this end, we bound the left-hand side of (20) from below and the right-hand side from above, obtaining the condition (19).

Since $i \in \text{supp}(\mathbf{x}^*)$ and $0 \in \partial F(\mathbf{x}^*)$, then $\mathbf{a}_i^T(A\mathbf{x}^*-\mathbf{y}) + \mu\,\text{sign}(x_i^*)$ is equal to zero and may be subtracted from or added to the left-hand side of (20) yielding

$$\begin{aligned} |\mathbf{a}_i^T(A\mathbf{x}-\mathbf{y})| = |\mathbf{a}_i^T A\mathbf{e} + \mu\,\text{sign}(x_i^*)| = \\ \left|\textstyle\sum_{j\neq i}(\mathbf{a}_i^T\mathbf{a}_j)e_j + e_i + \mu\,\text{sign}(x_i^*)\right|, \end{aligned} \tag{21}$$

where we have used the fact that the dictionary columns are normalized, $\mathbf{a}_i^T\mathbf{a}_i = 1$. Observe that $e_i = x_i^*$ because $i \notin \text{supp}(\mathbf{x})$. Using this, the triangle inequality $|a+b| \geq |a| - |b|$, the fact that x_i^* and $\mu\,\text{sign}(x_i^*)$ have the same sign, and the definition of mutual coherence, we obtain

$$\begin{aligned} |\mathbf{a}_i^T(A\mathbf{x}-\mathbf{y})| &\geq |x_i^*| + \mu - \delta\textstyle\sum_{j\neq i}|e_j| \\ &= |x_i^*|(1+\delta) + \mu - \delta\|\mathbf{e}\|_1. \end{aligned} \tag{22}$$

Next, we bound from above the right hand side of (20). Since $\ell \notin \text{supp}(\mathbf{x}^*)$, there exists a scalar $z \in [-1,1]$ for which $\mathbf{a}_\ell^T(A\mathbf{x}^*-\mathbf{y}) + \mu\mathbf{z}$ is equal to zero and may be subtracted from or added to the right-hand side. Therefore,

$$|\mathbf{a}_\ell^T(A\mathbf{x}-\mathbf{y})| = |\mathbf{a}_\ell^T A\mathbf{e} + \mu z| \leq \delta\|\mathbf{e}\|_1 + \mu, \tag{23}$$

where the last inequality uses $e_\ell = 0$. Comparing the bounds (22) and (23), we obtain that μ drops out and the condition (20) is guaranteed if (19) is satisfied. ∎

This proposition implies that if the dictionary A is far from being degenerate, i.e, $\delta \ll 1$, then, as our approximate solution gets better, any atom that contributes significantly to the solution is guaranteed to be chosen to the low-level set $\mathcal{C}$.

E. Treatment of the lowest level

One key component of our algorithm is the treatment of the lowest-level problem, whose dimension is typically the size of the current support. Assuming that in the lowest level of a V-cycle, $\mathbf{x}_c$ is dense and of length $|\mathcal{C}|$, the cost of the shrinkage iterations (relaxations) discussed above is $O(n|\mathcal{C}|)$ operations each. Therefore, applying many relaxations for solving the lowest-level problem may be costly. Moreover, although the solution $\mathbf{x}^*$ is normally sparse, there is no such guarantee for $\mathbf{x}$ throughout the entire solution process. An exact solution might be wasteful if $\mathbf{x}$ is too dense or if $\mathcal{C} \not\supseteq \text{supp}(\mathbf{x}^*)$.

Following the above reasoning, we limit the number of relaxations done on the lowest level to balance between cost and efficiency of the V-cycle. We aim to apply relaxations only until we increase the accuracy of $\mathbf{x}_c$ by some order of magnitude compared to the initial $\mathbf{x}_c$ on that lowest level. More specifically, we apply relaxations until the value of the expression $\|\mathbf{x}_c - \mathcal{S}_\mu(\mathbf{x}_c + A_c^T(\mathbf{y} - A\mathbf{x}_c))\|$ becomes 10 times smaller than it is initially on the lowest level. We found that this ratio balances well between the cost and efficiency of the V-cycles.

In addition, we explicitly limit the number of iterations proportionally to the cost ratio between a highest level relaxation and the lowest level relaxation, i.e., $\frac{m}{|\mathcal{C}|}$. Thus, the cost of the lowest-level solution will not exceed the cost of several high-level relaxations. In our tests, we apply at most $5\lceil\frac{m}{|\mathcal{C}|}\rceil$ relaxations which cost roughly the same as 10 high level relaxations.

F. A gradual initialization—"full multilevel cycle"

Our strategy aims at limiting the support of $\mathbf{x}$ throughout the solution process, thus saving unnecessary computations. The key problem is that if we initialize the solution process with $\mathbf{x} = 0$, and do a shrinkage iteration of the form (4), then many atoms will enter the support because of the rather large residual. This might happen with most shrinkage methods. Our way to prevent this from happening is to start our solution process from the bottom, instead of the top. That is, initialize $\mathbf{x} = 0$, choose a small set of atoms that are most likely to be in $\text{supp}(\mathbf{x}^*)$ (as in section III-B), and solve the reduced problem (8). Then, gradually enlarge the relevant set of columns, and apply a V-cycle for each level. This strategy, together with the exact solution of the lowest-level problem, is expected to maintain a sparse $\mathbf{x}$ throughout the solution process. Figure 1 and Algorithm 2 describe this approach. A similar strategy is used in multigrid methods and is called a "full multigrid algorithm" (FMG) [43]. The input parameters for Algorithm 2 are identical to those of Algorithm 1.

Algorithm: F-cycle$(A, \mathbf{x}, \mathbf{y}, \nu)$

1) Choose $\mathcal{C} = \text{likely}([m/2])$ and define P.
2) Define $A_c = AP$ and restrict $\mathbf{x}_c = P^T\mathbf{x}$.
3) **If** $|\mathcal{C}| < 2m_{min}$, solve the problem (8).
   Else $\mathbf{x}_c \leftarrow$ **F-cycle**$(A_c, \mathbf{x}_c, \mathbf{y}, \nu)$ *% Recursive call*
   $\mathbf{x}_c \leftarrow$ **V-cycle**$(A_c, \mathbf{x}_c, \mathbf{y}, \nu)$ *% Algorithm 1*
4) Prolong solution: $\mathbf{x} \leftarrow P\mathbf{x}_c$.
5) Apply ν relaxations: $\mathbf{x} \leftarrow Relax(A, \mathbf{x}, \mathbf{y})$.

Algorithm 2: Full multilevel cycle initialization.

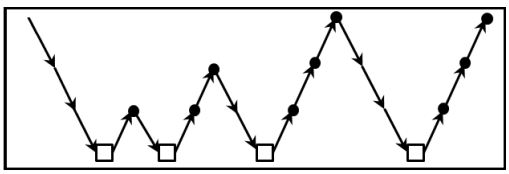

Fig. 1. Full multilevel cycle initialization.
'↘' refers to choosing $\mathcal{C}$ and reducing the problem, '□' refers to performing a lowest-level solve, '↗' refers to prolonging the solution, and '•' refers to applying ν relaxations.

G. A note on implementation

Algorithm 1 is presented in a rather symbolic way. In practice, we do not explicitly construct either the prolongations P or the low-level operators A_c. Instead, we use a single vector $\mathbf{x}$ and at low levels we address only the indices of $\mathbf{x}$ belonging to that level. This saves the need to store and extract the dynamic hierarchy of low-level matrices which usually changes from one V-cycle to the next. This also lets us apply multiple-point methods (such as SESOP, CG and TwIST) on consecutive approximations from *different* levels (i.e., while traversing up the V-cycle). This approach, however, requires the ability to efficiently multiply a vector by A_c^T using A and the subset $\mathcal{C}$ (in $n|\mathcal{C}|$ operations). Similarly, since $\mathbf{x}$ is sparse, for all the methods we compute $A\mathbf{x}$ while ignoring the zero entries of $\mathbf{x}$ (i.e., in $n\|\mathbf{x}\|_0$ operations). In MATLAB, for example, the latter can be achieved by storing $\mathbf{x}$ in a `sparse` data structure, but the former needs to be programmed manually.

In our multilevel algorithm, we do not perform any significant computations besides the relaxations and lowest-level solution. Most importantly, we do not calculate the likelihood vector $A^T(A\mathbf{x}-\mathbf{y})$ for choosing $\mathcal{C}$. Instead, we use the likelihood vector that corresponds to $\mathbf{x}$ before the last relaxation, which is calculated inside that relaxation as a by-product. We assume that the slight change in the likelihood vector as a result of the relaxations does not justify the extra computational cost.

IV. Numerical Results

In this section we compare the performance of several known shrinkage methods to our multilevel framework. We run the shrinkage iterations until the method-independent condition,

$$\|\mathbf{x}-\mathcal{S}_\mu(A^T(\mathbf{y}-A\mathbf{x}))\|/\|\mathbf{x}\| < 10^{-5}, \tag{24}$$

is satisfied. This condition is taken from [54] and is related to the size of $\min_{\mathbf{z}\in\mathcal{A}(\mathbf{x})}\{||\partial F(\mathbf{x})||\}$. We do not require extreme accuracy because our aim is only to discover the support of the minimizer for the reason described in Section II. In practice, all solutions achieved by all algorithms in our tests correspond to functional values which are identical up to several significant digits and have an essentially identical support size, which justifies this convergence criterion. Once this convergence criterion is satisfied, a debiasing phase is performed, where the norm $\|A\mathbf{x}-\mathbf{y}\|_2$ is minimized with respect only to the non-zeros in the minimizer of (3).

The "one-level" methods that we show in our comparison include Coordinate Descent (CD) [39], [40]; Parallel Coordinate Descent (PCD) [30], [31], [35]; acceleration of PCD by SESOP [31], [35] and non-linear CG as described above (denoted by SESOP1 and CG, respectively); we also test GPSR-Basic (GPSR) and GPSR-BB (non-monotone) [36], TwIST [32] and SpaRSA (monotone, BB variant)[38].

For PCD, CG, and SESOP, we used the exact linesearch procedure of [34]. We also accelerated CD by a linesearch, seeking a parameter $\alpha \geq 1$ as in (5). We denote this method by CD$^+$.

Within our multilevel framework we use CD$^+$ and CG. These are used both as relaxations and lowest-level solvers, and are denoted by ML-CD and ML-CG respectively. In some cases, we show our multilevel framework with CD$^+$ as a relaxation and CG-PCD as a low-level solver, and denote this option by ML-CD/CG.

For the methods of GPSR, SpaRSA, and TwIST, we used the default parameters suggested in the original papers and adapted the authors' MATLAB software for our tests. All the rest of the methods are implemented in MATLAB with certain procedures in the `mex` framework, including the linesearch, CD iteration, $A\mathbf{x}$ and $A_c^T\mathbf{y}$ multiplications. The matrix-vector multiplications in `mex` were also parallelized using the OpenMP library, which enhanced their calculation timings so that they are comparable to MATLAB's internal parallelization. Our parallel implementation of CD was not as fruitful because CD is sequential.

We perform four synthetic experiments using dense explicit dictionaries. In each of these we first generate a random, normally distributed dictionary,

$$A \in \mathbb{R}^{n\times m} \text{ s.t } a_{ij} \sim N(0,1), \tag{25}$$

and normalize its columns. Following this, we uniformly generate a support S_0 from all possible supports of size $[0.1n]$. Then, a vector $\mathbf{s}_0$ is generated with normally distributed values in the indices corresponding to S_0 and zeros elsewhere and a clean signal is generated by $A\mathbf{s}_0$, normalized such that $\|A\mathbf{s}_0\|_\infty = 1$. Next, we add random Gaussian noise obtaining the noisy signal,

$$\mathbf{y} = A\mathbf{s}_0 + \mathbf{n}\,, \tag{26}$$

with $\mathbf{n} \sim N(0,\sigma^2 I)$. In all our experiments we set the noise level to $\sigma = 0.02$. Because our experiments are randomly generated, each one is repeated 15 times, and the results are averaged and rounded. As in [54], our aim is only to examine the cost of computing the solution of (3), and not to measure the relation of its minimizer to the original sparse vector $\mathbf{s}_0$ used to generate $\mathbf{y}$.

The description above relates to our first experiment. A similar matrix is used in [54], [35], denoted by $K^{(2)}$. In the second experiment, taken from [55], the elements of A in (25) are replaced by ± 1 according to their signs, and the columns are normalized.

In the third experiment, we manipulate the singular values of A in (25) to make it highly ill-conditioned, with a condition-number of about 10^{10}. As before, we finalize the generation of the dictionary by normalizing its columns. A similar example is used in [54], [35], denoted by $K^{(4)}$. Figure 2 shows an example of the singular values of a dictionary A.

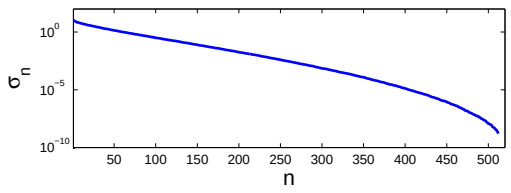

Fig. 2. The singular values of the matrix A with $n = 512$

In the fourth experiment we use a matrix that is similar to the $K^{(3)}$ matrix in [54]. The aim is to construct a matrix that is problematic for our solvers, even though it is well-conditioned. The construction is as follows: Let $U_1\Sigma_1 V_1^T$ be the SVD decomposition of A in (25) (after normalizing its columns). Let A_1 be the sub-matrix of A comprised of its first $\lceil 0.55m \rceil$ columns. Let $\mathbf{v}$ be the last column of A_1. Define: $A_2 = [A_1 | \mathbf{v}\mathbf{1}^T + 0.05B]$, where $\mathbf{1}$ is a vector of all ones, and B is a random matrix of size $n \times (m - \lceil 0.55m \rceil)$ like (25), with normalized columns. Let $U_2\Sigma_2 V_2^T$ be the SVD decomposition of A_2, and then define the final matrix A for this experiment as $U_2\Sigma_1 V_2^T$. This matrix has the favorable spectrum of (25) and the problematic singular vectors of A_2.

In the tables below, each test is described by two numbers: the mean number of iterations—relaxations or V-cycles—and (in brackets) the number of work-units. Each work-unit stands for mn floating point multiplications. We show these work-units in order to compare the multilevel method, that requires only a few albeit expensive cycles, with the shrinkage methods, that require many cheap iterations. We count all the operations preformed in all the algorithms. We note that the two most significant operations are

1) Multiplying $A^T\mathbf{y}$—costs mn operations, (a single work-unit). Multiplying $A_c^T\mathbf{y}$ costs $n|\mathcal{C}|$ operations.
2) Multiplying $A\mathbf{x}$—costs $n\|\mathbf{x}\|_0$ with $\|\mathbf{x}\|_0$ as in (1). Similarly, multiplying $A_c\mathbf{x}_c$ costs $n\|\mathbf{x}_c\|_0$ operations.

For some of the runs (with the largest $n = 2048$) we also show the average solution time measured by MATLAB's `tic` and `toc` mechanism. We note, however, that such timings depend on many variables that are not algorithmic (MATLAB compilation and parallel procedures, cache usage, memory usage, indexing etc.). The experiments were performed using MATLAB R2011b on a machine with an Intel core i7 quad-core CPU with 8 GB of RAM memory, running Windows 7.

The tables contain three parts. The upper section shows the influence of n, the middle part shows the influence of m, and the lower part shows the influence of μ. We use multilevel V-cycles with only one relaxation applied in each level ($\nu = 1$). The multilevel process is initialized with a single full multilevel cycle (Algorithm 2) starting from $\mathbf{x} = 0$. The one-level methods (excluding CD$^+$) are initially accelerated by the warm-start (or continuation) strategy of [38] with parameter $\xi = 0.3$. The tables also show the support size of the minimizer, $\|\mathbf{x}^*\|_0$, and improvement in signal-to-noise ratio defined as

$$\text{ISNR} = 10\log_{10}\left(\frac{\|A\mathbf{s}_0 - \mathbf{y}\|_2^2}{\|A\mathbf{s}_0 - A\mathbf{x}^*\|_2^2}\right),$$

that was achieved *after* the debiasing phase. Here, $A\mathbf{s}_0$ is the *clean signal* used in (26) to generate $\mathbf{y}$. The higher this value is, the better is the denoised reconstruction. As noted above, the $\|\mathbf{x}^*\|_0$ and ISNR measures are both almost identical for all the convergent algorithms.

A. Experiment 1: Well-conditioned random A

Table I summarizes the results for the first problem. Because A is such that every one of its sub-matrices is well-conditioned, we find that all methods perform relatively well.

For all the one-level results, the cost in work-units is similar to the number of iterations, implying that most of the work in each iteration is spent on computing $A^T\mathbf{r}$ (which costs 1 work-unit). The second significant operation in terms of cost is the multiplication $A\mathbf{x}$, which is performed in $n\|\mathbf{x}\|_0$ operations (approximately $\frac{\|\mathbf{x}\|_0}{m}$ work units). Most one-level methods are comparable in their performance (iterations and timings), with an edge to CD$^+$ (CD + linesearch). However, CD requires each column of A separately, which is problematic in terms of parallelization and handling composite dictionaries as in (10). SESOP1, which does not appear in the table, performs similarly to CG.

The multilevel acceleration for CG reduces the solution work-units by up to 50% or so compared to CG, while with CD$^+$ the multilevel speedup is only about 20%. In terms of timings, our implementation of ML has some overhead, therefore, the timings show a slightly smaller advantage than the work-units. In that respect, the cost per iteration of CD$^+$ is higher than that of the others due to the lack of efficient parallelization.

The values of the ISNR measures show that the noise level is indeed reduced in these experiments. The ISNR values in the lower section of the table show that if we choose μ too low, then the efficiency of the reconstruction decreases.

B. Experiment 2: Random ± 1 entries A

Table II summarizes the results for the second problem, which involves a matrix with random ± 1 entries. Although A is random as in the previous problem, it is clear that this problem is more challenging, as all the methods require more iterations to converge.

Unlike the previous case, the CD$^+$ method does not have an advantage over the rest of the one-level methods in terms of iterations and work-units. The PCD and SESOP1 methods (omitted from the table) exhibit performance similar to SpaRSA and CG, respectively; GPSR-BB was outperformed by GPSR and TwIST in this test case.

The multilevel acceleration significantly improves the performance of CD$^+$, while it only slightly accelerates CG. This is partly a result of an increased support of $\mathbf{x}$ during the course of the iterations. Applying ML-CD/CG (which uses CD instead of CG as a relaxation) overcomes this and improves the performance of ML. In terms of timings ML-CD/CG is clearly the favorite. The ISNR measures for this test case are similar to those of the previous one.

C. Experiment 3: ill-conditioned A

In this experiment we use the matrix A with the manipulated singular values. Most sub-matrices of A are also ill-conditioned, but not necessarily with a similar condition number.

Regarding the one-level methods, again, Table III shows that the CD$^+$ method is more effective than the rest of the one-level methods. The PCD and GPSR methods (not shown) did not converge after 4000 iteration and were outperformed by the rest of the methods. The '—' sign indicates that convergence was not reached after 3000 iterations in most of the tests.

n	m	μ	$\|\mathbf{x}^*\|_0$	ISNR	CD$^+$	ML-CD	PCD	CG	ML-CG	SpaRSA	GPSR-BB	TwIST	GPSR
256	$4n$	4σ	35.2	5.2	17.1(23)	**2.5(17)**	44.3(49)	36.8(43)	2.9(21)	36.8(43)	41.5(48)	56.6(67)	38.8(43)
512	$4n$	4σ	70.3	5.4	16.8(23)	**2.6(17)**	43.5(49)	34.6(42)	2.9(21)	37.3(43)	40.7(47)	57.3(69)	39.1(44)
1024	$4n$	4σ	139.8	5.4	16.8(25)	**2.7(20)**	43.5(51)	36.7(46)	3.0(24)	37.0(45)	40.2(49)	59.4(73)	38.8(46)
2048	$4n$	4σ	279	5.2	16.5(29) 0.38sec	**2.9(24)** 0.32sec	43.3(56) 0.60sec	35.7(49) 0.50sec	3.0(28) **0.31sec**	37.4(50) 0.50sec	40.5(54) 0.55sec	60.7(79) 0.79sec	39.7(52) 0.55sec
1024	$2n$	4σ	117.9	6.3	13.6(24)	**2.8(23)**	36.9(47)	32.3(44)	3.0(26)	31.7(42)	32.9(43)	53.5(72)	33.7(43)
1024	$6n$	5σ	125.5	4.8	16.2(20)	**2.3(14)**	40.7(45)	33.8(40)	3.0(18)	35.7(41)	38.7(44)	67.3(76)	37.6(42)
1024	$8n$	5σ	144.9	4.6	18.4(22)	**2.6(15)**	46.9(51)	40.3(46)	3.0(19)	41.6(47)	46.9(54)	76.7(87)	43.5(48)
1024	$4n$	5σ	118.5	5.2	15.1(20)	**2.6(17)**	39.7(45)	32.3(39)	2.9(20)	34.3(41)	36.5(43)	57.4(68)	35.6(41)
1024	$4n$	3σ	182.7	4.0	19.7(34)	**2.9(26)**	51.6(64)	43.9(58)	3.2(32)	44.5(57)	47.7(61)	65.3(85)	46.9(58)
1024	$4n$	2σ	298.0	2.0	25.7(58)	**3.1(44)**	65.3(93)	55.2(84)	3.9(56)	52.3(81)	54.9(83)	80.9(122)	57.7(85)

TABLE I

EXPERIMENT 1: WELL-CONDITIONED A. MEAN NUMBERS OF ITERATIONS (WORK-UNITS IN BRACKETS). THE AVERAGE TIMINGS ARE FOR $n = 2048$.

n	m	μ	$\|\mathbf{x}^*\|_0$	ISNR	CD$^+$	ML-CD	CG	ML-CG	ML-CD/CG	SpaRSA	TwIST	GPSR
256	$4n$	4σ	37.0	5.8	55.4(66)	**3.5(29)**	60.0(85)	3.7(60)	3.5(42)	101(133)	173(238)	136(156)
512	$4n$	4σ	68.7	5.2	103(117)	**4.6(42)**	68.5(99)	4.4(90)	3.9(57)	135(193)	234(314)	235(266)
1024	$4n$	4σ	132.3	5.3	238(258)	**4.8(47)**	86.5(132)	4.8(101)	3.8(57)	201(304)	326(421)	438(493)
2048	$4n$	4σ	268.9	5.2	711(754) 13.7sec	5.7(86) 1.34sec	106(175) 1.70sec	5.9(150) 1.66sec	**4.1(75)** **0.82sec**	338(557) 5.88sec	519(746) 6.91sec	951(1082) 12.9sec
1024	$2n$	4σ	117.8	6.5	203(231)	4.9(57)	70.6(112)	4.3(85)	**3.7(53)**	152(238)	247(401)	329(382)
1024	$6n$	5σ	126.3	5.0	214(226)	**4.9(37)**	88.2(131)	4.4(88)	3.8(47)	207(304)	377(482)	427(473)
1024	$8n$	5σ	143.1	4.3	263(275)	**5.1(34)**	95.4(135)	4.9(83)	3.7(40)	225(325)	475(588)	536(585)
1024	$4n$	5σ	113.8	4.7	187(200)	**4.9(39)**	73.9(110)	4.5(90)	3.7(45)	172(252)	326(440)	374(417)
1024	$4n$	3σ	170.5	4.5	375(416)	**5.5(76)**	98.9(157)	5.9(154)	4.1(80)	243(393)	366(514)	569(645)
1024	$4n$	2σ	268.3	2.3	684(773)	6.0(133)	123(197)	8.1(204)	**4.7(106)**	308(513)	448(732)	983(1128)

TABLE II

EXPERIMENT 2: RANDOM ± 1 ENTRIES A. MEAN NUMBERS OF ITERATIONS (WORK-UNITS IN BRACKETS). THE AVERAGE TIMINGS ARE FOR $n = 2048$.

n	m	μ	$\|\mathbf{x}^*\|_0$	ISNR	CD$^+$	ML-CD	CG	ML-CG	ML-CD/CG	SESOP1	SpaRSA	TwIST
256	$4n$	1σ	39.7	3.8	573(630)	**7.6(101)**	710(850)	6.7(184)	5.5(154)	713(900)	1941(2279)	2980(3348)
512	$4n$	1σ	78.1	3.7	591(646)	**7.2(115)**	936(1103)	8.0(240)	5.4(175)	901(1096)	2416(2813)	3643(4122)
1024	$4n$	1σ	153.3	4.0	569(624)	**7.7(107)**	1049(1218)	7.6(222)	6.0(185)	1012(1168)	2542(2886)	—
2048	$4n$	1σ	293	4.4	635(701) 12.5sec	**7.5(111)** 1.7sec	1276(1431) 16.0sec	8.6(252) 2.4sec	6.1(202) **1.6sec**	1251(1428) 17.3sec	2995(3373) 37.5sec	— —
1024	$2n$	1σ	169.3	3.8	406(491)	**5.7(143)**	779(986)	10.0(298)	6.9(221)	761(968)	1701(2116)	1383(1788)
1024	$6n$	1σ	142.6	4.3	659(700)	**5.7(76)**	1164(1292)	6.6(177)	5.1(154)	1136(1279)	2897(3193)	—
1024	$8n$	1σ	139.1	4.5	774(809)	**6.1(68)**	1347(1496)	5.9(164)	5.3(144)	1316(1457)	3340(3633)	—
1024	$4n$	4σ	107.3	-0.4	213(228)	**4.8(34)**	398(446)	5.2(76)	4.7(66)	385(432)	784(857)	1267(1388)
1024	$4n$	3σ	118.9	0.7	276(297)	**5.2(43)**	482(542)	5.2(101)	4.9(89)	485(551)	931(1027)	1906(2092)
1024	$4n$	2σ	132.1	1.7	363(395)	**6.2(63)**	692(796)	5.6(145)	5.3(134)	696(797)	1427(1589)	3039(3359)

TABLE III

EXPERIMENT 3: ILL-CONDITIONED A. MEAN NUMBERS OF ITERATIONS (WORK-UNITS IN BRACKETS). THE AVERAGE TIMINGS ARE FOR $n = 2048$.

Table III also shows that now the multilevel algorithm significantly reduces the cost of the solution, compared to the one-level methods. In most cases, the cost of the ML method is about 15%-25% of the cost of the 1L methods in terms of work-units and timings. The Results of ML-CD/CG are slightly better than those of ML-CG.

In the upper section (growing n), all methods show quite scalable performance in the work-unit measure. In the second section (growing m), it is seen that the problem becomes harder as m grows, since there are more possible supports and more room for error. The ML versions, however, appear quite scalable in their work-unit cost with respect to the growing redundancy (m) of the dictionary. In the third section, where different values of μ are used, one can see the loss of scalability of the methods with respect to the support size. Asymptotically, bigger supports yield bigger effective matrices and also a bigger condition number. Furthermore, finding the true support becomes a much harder task.

The values of ISNR for this test case are lower than those in the previous two cases, indicating that the noise reduction using the l_1 regularization is somewhat less effective for this ill-conditioned dictionary.

D. Experiment 4: well-conditioned A with similar columns

Our last experiment involves a matrix A that consists of two types of columns: those that belong to the random part (A_1), and those that belong to the other, replicated part. If the support of the true solution has only columns from A_1, the effective matrix A_S is well-conditioned and the solvers handle it well. However, if the support has columns from both sections, then A_S is most likely ill-conditioned even though A is not.

Table IV shows the results for this problem. Again, as in the second test case, CD$^+$ has no advantage over the rest of the one-level methods. Because the support sub-matrix A_S is ill-conditioned, the CG and SESOP methods outperform the rest of the one-level methods. This corresponds to CG outperforming all one-point iterations for quadratic problems. The results of PCD (omitted) are similar to those of GPSR. GPSR-BB and TwIST were outperformed by all the other methods.

n	m	μ	$\|\mathbf{x}^*\|_0$	ISNR	CD$^+$	ML-CD	CG	ML-CG	ML-CD/CG	SESOP1	SpaRSA	GPSR
256	$4n$	4σ	26.8	2.5	1323(1392)	8.5(123)	462(575)	7.3(117)	**4.1(47)**	426(556)	1552(2097)	2025(2222)
512	$4n$	4σ	44.9	3.6	2411(2497)	10.9(202)	635(768)	9.7(167)	**5.0(50)**	586(731)	2382(3144)	2653(2913)
1024	$4n$	4σ	76.0	3.6	2460(2526)	12.0(204)	695(839)	9.1(154)	**4.9(48)**	719(903)	3029(4057)	2921(3320)
2048	$4n$	4σ	148	3.7	2738(2800) 51.6sec	13.9(263) 4.4sec	817(969) 11.3sec	11.6(206) 2.4sec	**5.3(49)** **0.58sec**	944(1177) 14.2sec	3268(4528) 48.4sec	3412(3974) 48.9
1024	$2n$	4σ	66.7	2.4	1651(1727)	14.5(221)	353(437)	6.7(101)	**4.2(38)**	345(436)	1441(1980)	1918(2134)
1024	$6n$	4σ	84.5	3.4	2216(2263)	15.7(166)	765(906)	9.4(160)	**5.3(39)**	827(1023)	3066(4140)	3226(3652)
1024	$8n$	4σ	88.9	3.1	2210(2248)	11.3(98)	692(804)	8.6(127)	**4.7(35)**	776(963)	3098(4298)	3032(3491)
1024	$4n$	5σ	67.6	2.8	2320(2377)	8.3(123)	623(739)	7.9(124)	**4.8(37)**	645(797)	2621(3443)	2731(3058)
1024	$4n$	3σ	106.5	2.5	2154(2227)	15.0(293)	787(955)	10.1(182)	**5.5(61)**	823(1034)	3125(4341)	3345(3840)
1024	$4n$	2σ	210.3	1.3	1886(2006)	22.1(389)	801(997)	12.4(274)	**5.6(91)**	832(1068)	2889(4292)	3352(4011)

TABLE IV
EXPERIMENT 4: WELL-CONDITIONED A WITH SIMILAR COLUMNS. MEAN NUMBERS OF ITERATIONS (WORK-UNITS IN BRACKETS). THE AVERAGE TIMINGS ARE FOR $n = 2048$.

Here we have an interesting case for the multilevel approach. As the CD$^+$ iteration is not very efficient, neither is the lowest-level solution of ML-CD. Because we limit the number of iterations on the lowest levels, the ML-CD method requires a few more cycles to converge. Compared to that, we see that ML-CD/CG outperforms the other ML options, showing that the CD iteration is better than CG as a relaxation even though it is far inferior as a solver.

Again, the values of ISNR for this test case are relatively low, but other choices of μ lead to lower values of ISNR. Thats is, the noise is reduced as effectively as possible via the solution of (3).

E. Discussion

The multilevel approach enhances the performance of one-level shrinkage iterations in almost all our tests. We note that any iterated shrinkage method can be incorporated in our ML framework. Generally, the best option that we found is ML-CD/CG (ML with CD$^+$ as relaxation and CG as lowest-level solver). That is because the CD iteration updates the entries x_i one by one and updates the residual accordingly, so it does not have a tendency to fill $\mathbf{x}$ with a large number of non-zeros which would harm the efficiency of ML. CG, on the other hand, is generally better at handling ill-conditioned matrices and therefore it is a more robust lowest-level solver.

The one-level methods that we tested behaved differently in the different problems. Overall, the best results were obtained by CD$^+$, CG, SESOP1 and SpaRSA. On top of the disadvantages of CD mentioned earlier (evident in its timings), it may struggle when the effective sub-matrix A_S is ill-conditioned.

V. CONCLUSIONS AND FUTURE RESEARCH

A multilevel approach is introduced for the solution of (3) when the matrix A is given explicitly. The new method takes advantage of the typically sparse representation of the signal by gradually ignoring ostensibly irrelevant data from the over-complete dictionary. This approach significantly accelerates the performance of existing iterated shrinkage methods as well as their accelerated versions. The biggest advantage is gained when the solution is indeed sparse and the problem is effectively ill-conditioned (A_S ill-conditioned). Also, in most of our numerical tests, the multilevel approach reduced the required number of iterations dramatically. We expect significant further gains as more efficient methods are developed for the dense lowest-level problem.

A challenging future research direction may target a multilevel approach for fast operator dictionaries, where the low-level dictionary is chosen as a smaller version of the upper-level fast operator. Such an approach is used in the so-called *geometric* multigrid [43]. The key question in this research is how to define the transfer operators so that the solution updates significantly enhance the convergence of the relaxation methods.

VI. ACKNOWLEDGEMENTS

The authors would like to thank Prof. Michael Elad and Javier Turek for their valuable advice.

REFERENCES

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM review*, vol. 51, no. 1, pp. 34–81, 2009.
[2] M. Elad, M. A. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *IEEE Proceedings - Special Issue on Applications of Compressive Sensing and Sparse Representation.*, vol. 98, no. 6, pp. 972–982, 2010.
[3] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
[4] E. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, pp. 21–30, March 2008.
[5] E. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Trans. Inform. Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.
[6] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proc. of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.
[7] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *J. Constr. Approx.*, vol. 13, pp. 57–98, 1997.
[8] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.
[9] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," *Proc. Asilomar Conf. Signals, Syst., Comput.*, vol. 1, pp. 40–44, 1993.
[10] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. on Information Theory*, vol. 50, no. 10, pp. 2231–2342, 2004.
[11] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.
[12] T. Blumensath and M. E. Davies, "Gradient pursuits," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2370–2382, 2008.
[13] D. Donoho, I. Drori, Y. Tsaig, and J. Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," *Stanford Statistics Dept. Tech. Report*, 2006.

[14] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, 2008.

[15] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Info. Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[16] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *J. Fourier Analysis Applications*, vol. 14, no. 5, pp. 629–654, 2008.

[17] ——, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, pp. 265–274, 2009.

[18] ——, "Normalized iterative hard thresholding: guaranteed stability and performance," *IEEE J. Selected Topics Signal Process.*, vol. 4, pp. 298–309, 2010.

[19] T. Blumensath, "Accelerated iterative hard thresholding," *IEEE Sig. Proc. Letts.*, vol. 92, no. 3, pp. 752–756, 2012.

[20] K. K. Herrity, A. C. Gilbert, and J. A. Tropp, "Sparse approximation via iterative thresholding," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 3, pp. 624–627, 2006.

[21] V. Cevher, "On accelerated hard thresholding methods for sparse approximation," *Technical Report, EPFL, Idiap Research Ins.*, 2011.

[22] J. A. Tropp, "Just relax: Convex programming methods for identifying sparse signals," *IEEE Trans. on Information Theory*, vol. 52, no. 3, pp. 1030–1051, 2006.

[23] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[24] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, 1st ed. Springer, 2010.

[25] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Statist. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.

[26] M. Figueiredo and R. Nowak, "An em algorithm for wavelet-based image restoration," *IEEE Trans. Signal Process.*, vol. 12, no. 8, pp. 906–916, 2003.

[27] F. M. Jean-Luc Starck, Mai K. Nguyen, "Wavelets and curvelets for image deconvolution: a combined approach," *Signal Process.*, vol. 83, no. 10, pp. 2279–2283, 2003.

[28] I. Daubechies, M. Defrise, and C. De-Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. LVII, pp. 1413–1457, 2004.

[29] M. Figueiredo and R. Nowak, "A bound optimization approach to wavelet-based image deconvolution," *IEEE International Conference on Image Processing - ICIP 2005*, vol. 2, pp. 782–785, 2005.

[30] M. Elad, "Why simple shrinkage is still relevant for redundant re tations?" *IEEE Trans. Inform. Theory*, vol. 52, no. 12, pp. 555 2006.

[31] M. Elad, B. Matalon, and M. Zibulevsky, "Coordinate and s optimization methods for linear least squares with non-quadrat larization," *Appl. Comput. Harmon. Anal.*, vol. 23, pp. 346–367

[32] J. Bioucas-Dias and M. Figueiredo, "Two-step iterative sh thresholding algorithm for linear inverse problems," *IEEE Tran. Process.*, vol. 16, no. 12, pp. 2980–2991, 2007.

[33] E. T. Hale, W. Yin, and Y. Zhang, "Fixed-point continuation minimization: Methodology and convergence," *SIAM Journal o mization*, vol. 19, pp. 1107–1130, 2008.

[34] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, "A fast algorit..... sparse reconstruction based on shrinkage, subspace optimization a continuation," *SIAM Sci. Comp.*, vol. 32, no. 4, pp. 1832–1857, 201

[35] M. Zibulevsky and M. Elad, "L1-l2 optimization in signal and ima processing: Iterative shrinkage and beyond," *IEEE Signal Process Magazine*, vol. 27, no. 3, pp. 76–88, May 2010.

[36] M. Figueiredo, R. Nowak, and S. J. Wright, "Gradient projection sparse reconstruction: Application to compressed sensing and ot inverse problems," *IEEE J. Selected Topics Signal Process.*, vol. 1, no pp. 586–597, 2007.

[37] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, "Bregman iterative al rithms for ell1-minimization with applications to compressed sensing, *SIAM J. Imaging Sciences*, vol. 1, no. 1, p. 143168, 2008.

[38] S. J. Wright, R. Nowak, and M. Figueiredo, "Sparse reconstruction by seperable approximation," *IEEE Trans. Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[39] S. Bourguignon, H. Carfantan, and J. Idier, "A sparsity-based method for the estimation of spectral lines from irregularly sampled data," *IEEE J. of selected topics in signal processing*, vol. 1, no. 4, pp. 575–585, 2007.

[40] J. Friedman, T. Hastie, H. Hofling, and R. Tibshirani, "Pathwise coordinate optimization," *Ann. Appl. Stat.*, vol. 1, no. 2, pp. 302–332, 2007.

[41] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, 2001.

[42] S. Sardy, A. G. Bruce, and P. Tseng, "Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries," *J. Comput. Graph. Statist.*, vol. 9, no. 2, pp. 361–379, 2000.

[43] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, 2nd ed. Philadelphia: SIAM, 2000.

[44] A. Brandt and D. Ron, "Multigrid solvers and multilevel optimization strategies," in *Multilevel Optimization and VLSICAD*. Kluwer (Boston), 2003, pp. 1–69.

[45] E. Treister and I. Yavneh, "Square and stretch multigrid for stochastic matrix eigenproblems," *Numerical Linear Algebra with Application*, vol. 17, pp. 229–251, 2010.

[46] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[47] J. Nocedal and S. Wright, *Numerical Optimization*. New-York: Springer, 1999.

[48] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[49] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural Comput.*, vol. 12, no. 2, pp. 337–365, 2000.

[50] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Comput.*, vol. 15, no. 2, pp. 349–396, 2003.

[51] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[52] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[53] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[54] I. Loris, "On the performance of algorithms for the minimization of ,l1-penalized functionals," *Inverse Probl.*, vol. 25, no. 3, pp. 1–16, 2009.

[55] A. Maleki and D. L. Donoho, "Optimally tuned iterative reconstruction algorithms for compressed sensing," *IEEE J. Sel. Topics Signal Process.*, vol. 4, pp. 330–341, 2010.

Eran Treister: is currently a PhD candidate in the department of computer science at the Technion—Israel Institute of Technology, Haifa, Israel. His research interests include numerical methods in linear algebra, multigrid methods, image and signal processing, optimization, and solution of partial differential equations. His research focusses on algebraic multilevel iterative methods for large scale problems.

Irad Yavneh: received the Ph.D. degree in applied mathematics from the Weizmann Institute of Science, Rehovot, Israel, in 1991. He is currently a Professor in the faculty of computer science at the Technion—Israel Institute of Technology, Haifa, Israel. His research interests include multiscale computational techniques, scientific computing and computational physics, image processing and analysis, and geophysical fluid dynamics.