A MULTILEVEL APPROACH FOR L_1 PENALIZED LEAST-SQUARES MINIMIZATION

ERAN TREISTER AND IRAD YAVNEH

ABSTRACT. The area of sparse representation of signals is drawing tremendous attention in recent years. The idea behind the model is that a signal can be approximated as a linear combination of a few "atoms" from a prespecified and over-complete "dictionary". The sparse representation of a signal is 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, surpassing traditional optimization techniques. In this paper, we suggest a new simple multilevel approach that reduces the computational cost of existing solvers for these inverse problems. The new method takes advantage of the typically sparse representation of the signal and coarsens by reducing the dimension of the problem by gradually ignoring ostensibly irrelevant data from the over-complete dictionary. 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 dictionary is an explicit matrix.

1. INTRODUCTION

Sparse modelling of signals is an emerging area of research that is drawing vast interest and finding use in numerous applications. One of the key observations underlying this field is that natural signals, such as images, admit sparse decompositions over specific spatial transforms [4, 16]. The ability to reconstruct a sparse signal or object from only a few linear measurements is often called *compressive sensing* [11, 6, 5, 23].

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 referred to as 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

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

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 $\mathbf{x}$. However, this optimization problem is non-convex and generally very hard to solve, as its solution usually requires an intractable combinatorial search [7]. Nevertheless, the solution can be approximated using so-called "greedy algorithms" such as Orthogonal Matching Pursuit (OMP) [25, 29], Stagewise OMP (StOMP) [10], CoSAMP [26] and others.

An alternative approach is to relax (1) by replacing the l_0 quasi-norm with the well-known l_1 norm, which has somewhat similar "sparseness properties" [30]. The new problem,

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

Date: January 10, 2011.

Department of Computer Science, Technion—Israel Institute of Technology. eran@cs.technion.ac.il.

Research supported by the Israeli Science Foundation, grant number 795/08.

called Basis Pursuit [8], 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 [14]. A typical solution vector $\mathbf{x}^*$ of (2) is relatively sparse, and under some conditions it is in fact equal to the global minimizer of (1). Problem (2) can be formulated and solved as a linear programming problem.

Since 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. A common choice for this approach is 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 solution $\mathbf{x}^*$, but also a greater discrepancy $||A\mathbf{x}^* - \mathbf{y}||_2^2$. Problem (3) is popular in sparse modelling applications, partly because it is an unconstrained convex optimization problem. However, due to the discontinuity of its gradient, which arises from using the l_1 norm, traditional optimization methods such as gradient descent or quasi-Newton methods tend to be slow. Therefore, various so-called iterative "shrinkage" algorithms are commonly employed, often together with accelerations [12, 32, 9, 19, 31, 18, 17]. 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 sparse modelling problems such as (3), based on the main concept of classical multigrid methods [3]; that is, we accelerate the convergence of simple iterative methods for (3) using smaller (coarser) versions of the problem. The idea of introducing multilevel methods to sparse modelling has yet to be explored and it has tremendous potential. In this work we follow the general ideas of *adaptive* algebraic multigrid methods, which exploit a hierarchy of coarse approximate operators that evolve with the solution process, eventually becoming exact. Our algorithm requires the dictionary A to be given explicitly as a matrix, rather than a fast transform operator as in the Wavelet or DCT transforms. More precisely, the matrix A may take the form

$$A = H \cdot B, \tag{4}$$

where H is an operator that acts on the columns of the dictionary B [32]. For example, in the image deblurring problem the operator H is a low-pass filter while B is the underlying dictionary. In this work, we assume that either H or B are given as explicit matrices. In most of these cases, it is the matrix B which is explicit, as, for example, in trained dictionaries, where the matrices B may be either sparse or dense [27, 28, 15, 1, 24, 21, 20, 13].

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. Here, the difficulty is different. The gradient of (3) is given by

$$\nabla F(\mathbf{x}) = A^T A\mathbf{x} - A^T\mathbf{y} + \mu\,\mathrm{sign}(\mathbf{x}), \tag{5}$$

where $\mathrm{sign}(\mathbf{x})$ is the sign-pattern of $\mathbf{x}$, with the convention $\mathrm{sign}(0) = 0$. It can be seen that if the sign-pattern of the minimizer $\mathbf{x}^*$ of (3) is known, then the problem becomes quadratic and can normally be solved efficiently either by the Conjugate Gradient (CG) method or directly by solving the system that corresponds only to the non-zeros of the vector $\mathbf{x}^*$. Therefore, the main effort in solving (3) is invested in finding the non-zero elements of the minimizer $\mathbf{x}^*$ and their signs, after which the problem becomes linear.

2. Iterative shrinkage methods and accelerations

We briefly describe two *iterated shrinkage* methods for solving (3). The first is the Coordinate Descent (CD) method which is mostly relevant when the matrix is given explicitly, or else when the operator H in (4) is not explicit but is an easily-applied transform. The main drawback of this method is that it is sequential in nature and hard to parallelize efficiently. If the sign-pattern of the minimizer $\mathbf{x}^*$ is known, this method reduces to the Gauss-Seidel iterative linear solver.

In each iteration of CD, we update the elements of $\mathbf{x}$ one by one in some prescribed order. Updating the element x_i requires minimizing the following functional for the scalar variable z, which then replaces x_i:

$$g(z) = \frac{1}{2}\|A\mathbf{x} - a_i x_i + a_i z - \mathbf{y}\|_2^2 + \mu|z|, \tag{6}$$

where a_i is the i-th column of A. By setting $\mathbf{r} = \mathbf{y} - A\mathbf{x}$ we get the following equation for z:

$$\frac{dg(z)}{dz} = \|a_i\|_2^2(z - x_i) - a_i^T\mathbf{r} + \mu\text{sign}(z) = 0. \tag{7}$$

It is easy to see that if $|a_i^T\mathbf{r} + \|a_i\|_2^2 \cdot x_i| > \mu$ then the sign of z is given by the sign of $a_i^T\mathbf{r} + \|a_i\|_2^2 \cdot x_i$. Otherwise, the derivative (7) cannot be made to vanish, but the minimizer of (6) is then $z = 0$. Therefore, we can write the solution as follows:

$$z_{opt} = \mathcal{S}_{\frac{\mu}{\|a_i\|_2^2}}\left(\frac{a_i^T\mathbf{r}}{\|a_i\|_2^2} + x_i\right), \tag{8}$$

where

$$\mathcal{S}_q(t) \equiv \text{sign}(t) \cdot \max(0, |t| - q) \tag{9}$$

is the "shrinkage" function, so dubbed because the size of the argument t is reduced by q (or set to zero if $q > |t|$).

The next iterated shrinkage method that we describe is a parallel variant of CD, called Parallel Coordinate Descent (PCD), introduced in [12]. Just as CD is related to Gauss-Seidel, PCD is related to the well known Jacobi iteration. Let D be a diagonal matrix with diagonal equal to that of the matrix A^TA. Then the PCD direction is defined as

$$\mathbf{x}_{PCD} = \mathcal{S}_{D^{-1}\mu}\left(D^{-1}A^T(\mathbf{y} - A\mathbf{x}) + \mathbf{x}\right). \tag{10}$$

Here, unlike CD, a change in the vector $\mathbf{x}$ does not guarantee a descent in (3), only a descent *direction*. Therefore, as in [12], a line-search for minimizing the functional (3) with respect to the scalar α is performed for

$$\mathbf{x}_{opt} = \mathbf{x} + \alpha(\mathbf{x}_{PCD} - \mathbf{x}). \tag{11}$$

In the scope of (semi) explicit dictionaries, the PCD method is especially useful in cases of a relatively expensive fast transform H in (4), or for parallel computing. The convergence of PCD is usually slower than that of CD, but CD requires applying the transform H for each variable update, while PCD requires applying H and H^T only once per iteration. The advantage of PCD in parallel computing is obvious.

2.1. Acceleration.

The iterative shrinkage methods mentioned above are much faster than standard first-order minimization methods such as classical Steepest Descent, however, they may remain relatively slow, requiring acceleration. In [32, 18] an optimization method called "sequential subspace optimization" (SESOP) was considered together with the PCD iteration. Other acceleration methods include FISTA [2] and a version of nonlinear CG [32]. In [31], it was suggested to apply several tests to check whether the sign-pattern of the solution is achieved, and if so to apply CG until convergence. In this paper, we use SESOP (briefly described below) in order to accelerate

PCD. We also suggest and test using a line-search in the interval $[1, 2]$ for accelerating the result of CD, and denote this approach by over-relaxed CD (OR-CD).

2.1.1. *Sequential Subspace Optimization—SESOP-M.* In the process of iteratively minimizing (3), let $\mathbf{x}^k$ denote the approximation to the minimizer of (3) after the k-th iteration. Instead of searching along a single direction $\mathbf{r}^k$ at each iteration as in (11), a set of search directions is defined, $\{\mathbf{r}_i^k\}_{i=1}^{M+1}$. The next approximation is then obtained by minimizing (3) over the subspace spanned by these directions, i.e.,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + R^k\alpha^k \quad \text{s.t.} \quad \alpha^k = \underset{\alpha\in\mathbb{R}^{M+1}}{\arg\min} F(\mathbf{x}^k + R^k\alpha), \tag{12}$$

where the columns of the matrix R^k are the search directions $\{\mathbf{r}_i^k\}_{i=1}^{M+1}$. As in the configuration denoted as PCD-SESOP-M in [18], we choose (in MATLAB notation)

$$R^k = [\mathbf{x}^k_{PCD} - \mathbf{x}^k\,,\, \mathbf{x}^k - \mathbf{x}^{k-1}\,,\, ...\,,\, \mathbf{x}^{k-M+1} - \mathbf{x}^{k-M}], \tag{13}$$

where $\mathbf{x}^k_{PCD}$ is the PCD direction obtained from $\mathbf{x}^k$. The minimization (12) cannot be solved by a shrinkage method, so a quasi-Newton method is used [18]. In this work, we apply SESOP with two directions only ($M = 1$), which reduces to the CG method once the sign-pattern of the minimizer $\mathbf{x}^*$ is found.

3. A multilevel approach

We next describe our approach for solving (3), and note that similar approaches may be suitable for the solution of least-squares problems with other "sparsity promoting" penalty functions. Our goal is to define a coarse-level problem whose solution will improve the current approximation to the solution of (3). In fact, we aim for a coarse-level problem that is equivalent to the original fine-level problem, such that reducing the value of the coarse-level functional guarantees a reduction in the value of the fine-level functional. Furthermore, the coarse-level problem should be similar to the fine-level problem to allow recursion. Since the problem is nonlinear, we approximate the solution itself on the coarse level, rather than the error as in classical linear multigrid.

3.1. Definition of the coarse-level problem.

Definition of the coarse-level problem. Assume we have an approximation $\mathbf{x}$ for the minimizer of (3). Also, assume that we have defined a prolongation $P \in \mathbb{R}^{m\times m_c}$, with $m_c < m$, such that the current approximation $\mathbf{x}$ is in its range, that is, there exists a smaller vector $\mathbf{x}_c \in \mathbb{R}^{m_c}$ such that $\mathbf{x} = P\mathbf{x}_c$. Substituting $P\mathbf{x}_c$ for $\mathbf{x}$ in the objective (3), we get the new problem:

$$\min_{\mathbf{x}_c\in\mathbb{R}^{m_c}} F^c(\mathbf{x}_c) \equiv \min_{\mathbf{x}_c\in\mathbb{R}^{m_c}} F(P\mathbf{x}_c) = \min_{\mathbf{x}_c\in\mathbb{R}^{m_c}} \frac{1}{2}\|AP\mathbf{x}_c - \mathbf{y}\|_2^2 + \mu\|P\mathbf{x}_c\|_1, \tag{14}$$

which has only m_c degrees of freedom. In order to make this problem of the same form as the fine-level problem, we are require

$$\|P\mathbf{x}_c\|_1 = \|\mathbf{x}_c\|_1. \tag{15}$$

Our choice of P is very simple. Given the coarse-variable subset, $\mathcal{C} \subset \{1, ..., m\}$, we define the prolongation to simply zero-fill the elements of $\mathbf{x}$ that do not belong to $\mathcal{C}$, while leaving the elements that do belong to $\mathcal{C}$ unchanged; P^T is a simple injection operator. Thus, if a fine-level variable i corresponds to a coarse-level variable I then

$$P_{i,J} = \begin{cases} 1 & \text{if } i \in \mathcal{C} \text{ and } I = J, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Evidently, (15) holds for this simple choice of prolongation. Furthermore, let supp$(\mathbf{x})$ denote the support of $\mathbf{x}$,

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

then, to satisfy $\mathbf{x} = P\mathbf{x}_c$ we must choose the coarse variables $\mathcal{C}$ such that $\text{supp}(\mathbf{x}) \subseteq \mathcal{C}$.

With this definition of P, we define the coarse dictionary

$$A_c = AP, \tag{17}$$

which has fewer columns than the fine-level dictionary. In fact, A_c is a sub-matrix of the fine-level matrix A, with columns given by the columns of A corresponding to the indices in $\mathcal{C}$. The coarse-level problem (14) can now be written as

$$\min_{\mathbf{x}\in\mathbb{R}^{m_c}} F^c(\mathbf{x}) = \min_{\mathbf{x}_c\in\mathbb{R}^{m_c}} \frac{1}{2}\|A_c\mathbf{x}_c - \mathbf{y}\|_2^2 + \mu\|\mathbf{x}_c\|_1. \tag{18}$$

Since this problem is similar to (3), we can apply a multilevel algorithm by treating it recursively.

3.2. **Choosing the coarse-level variables.** The set of coarse variables $\mathcal{C}$ is based on the current approximation $\mathbf{x}$. First, we decide on the number of coarse variables m_c (in this work we choose $m_c = [m/2]$). Since we require $\text{supp}(\mathbf{x}) \subseteq \mathcal{C}$, then if $|\text{supp}(\mathbf{x})| \geq m_c$ then we simply choose $\mathcal{C} = \text{supp}(\mathbf{x})$. Otherwise, we need to add variable indices that are currently not in $\text{supp}(\mathbf{x})$, but have a relatively high chance of being in the support of the true solution $\mathbf{x}^*$. These correspond to atoms a_i with a relatively large value of $|a_i^T(A\mathbf{x} - \mathbf{y})|$, since including those in the support reduces the first term in $F(\mathbf{x})$ relatively more significantly per given increase in the second term (see also Proposition 4 below). This rationale is also used in the "greedy algorithms" mentioned earlier. Thus, we define

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

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

3.3. **Definition and properties of the multi-level V-cycle.** Algorithm 1 defines the multi-level V-cycle for solving (3).

Problem Input: Initial vector: $\mathbf{x} \in \mathbb{R}^m$, fine-level dictionary: $A \in \mathbb{R}^{n\times m}$, a signal $\mathbf{y} \in \mathbb{R}^n$, a penalty parameter μ.
Method Input: Iterative Shrinkage method (+ Acceleration): $Relax(A, \mathbf{x}, \mathbf{y})$.
Number of pre- and post-relaxations: ν_1, ν_2;
Minimal number of columns allowed: $\frac{1}{2}m_{min}$.
Algorithm: $\mathbf{V}(\nu_1, \nu_2, \nu_{coarse})$ **cycle**

```
(1) Apply $\nu_1$ pre-relaxations: $\mathbf{x} \leftarrow Relax(A, \mathbf{x}, \mathbf{y})$.
(2) Define the coarse-level variables $\mathcal{C}$.
(3) Define the prolongation $P$ and the coarse-level vector $\mathbf{x}_c = P^T\mathbf{x}$.
(4) Calculate the coarse-level dictionary: $A_c = AP$.
(5) If $\mathcal{C} = \text{supp}(\mathbf{x})$ or $|\mathcal{C}| < m_{min}$,
        (a) Solve the coarse-level problem (18).
    else Recursively apply the ML algorithm on (18), starting from $\mathbf{x}_c$ .
(6) Prolong solution: $\mathbf{x} \leftarrow P\mathbf{x}_c$ (Coarse-level correction).
(7) Apply $\nu_2$ post-relaxations: $\mathbf{x} \leftarrow Relax(A, \mathbf{x}, \mathbf{y})$.
```

Algorithm 1: V-cycle for solving l_1 penalized least squares

We next make some observations regarding the algorithm. Unless otherwise stated, we refer to the two-level version, where the minimization problem is solved exactly on the second-finest level. Note first that, by (14),

$$F^c(\mathbf{x}_c) = F(P\mathbf{x}_c), \;\forall \mathbf{x}_c \in \mathbb{R}^{m_c} . \tag{19}$$

Next, substitute $\mathbf{x}_c = P^T\mathbf{x}$ into (19), and observe that (16) and the fact that $\text{supp}(\mathbf{x}) \subseteq \mathcal{C}$ together imply $\mathbf{x} = PP^T\mathbf{x}$. Hence,

$$F^c(P^T\mathbf{x}) = F(\mathbf{x}),\ \forall \mathbf{x} \in \mathbb{R}^m\,. \tag{20}$$

The following two observations follow immediately from (19) and (20).

Proposition 1. (Monotonicity.) *Let $\mathbf{x}$ be the current approximation to the minimizer of F in* (3) *after Step 1 of Algorithm 1. Let $\mathbf{z}_c$ be a better approximation than $\mathbf{x}_c$ to the minimizer of the coarse-level problem* (18)*, such that $F^c(\mathbf{x}_c) > F^c(\mathbf{z}_c)$. Then $F(\mathbf{x}) > F(P\mathbf{z}_c)$.*

Proposition 2. (Direct Solution.) *If $\mathcal{C} \supseteq$ supp$(\mathbf{x}^*)$, then $\mathbf{x}^*_c = P^T\mathbf{x}^*$ is the solution of the coarse-level problem* (18)*, and the two-level Algorithm 1 solves problem* (3) *in one cycle.*

It follows that if $\text{supp}(\mathbf{x}) \supseteq \text{supp}(\mathbf{x}^*)$ then the two-level cycle solves the problem (3) in one cycle. Also, $\mathbf{x}^*$ is a stationary point of Algorithm 1.

For the next propositions we use the notion of sub-gradients [14]. $\partial F(\mathbf{x})$, the sub-differential of F, is a non-empty set of sub-gradients,

$$\partial F(\mathbf{x}) = \left\{A^T(A\mathbf{x} - \mathbf{y}) + \mu\mathbf{z}\right\}, \tag{21}$$

comprised of all vectors $\mathbf{z}$ whose elements satisfy

$$z_i = \text{sign}(x_i) \text{ if } |x_i| > 0\,,\ \ z_i \in [-1, 1] \text{ if } |x_i| = 0\,. \tag{22}$$

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

Proposition 3. (No Stagnation.) *If $\mathcal{C} \not\supseteq$ supp$(\mathbf{x}^*)$, and $\mathbf{x} = P\mathbf{x}_c$ is the new fine-level approximation after coarse-level correction, then a single iterated shrinkage post-relaxation must cause at least one atom to be added to supp$(\mathbf{x})$.*

Proof. Since $\mathbf{x}_c$ is a minimizer of the coarse-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 $A\mathbf{x} = A_c\mathbf{x}_c$, then $0 \in \partial F^c$ means that for all $k \in \mathcal{C}$

$$\begin{array}{ll} a_k^T(A\mathbf{x} - \mathbf{y}) + \mu\text{sign}(x_k) = 0 & \text{if } |x_k| > 0 \\ |a_k^T(A\mathbf{x} - \mathbf{y})| \le \mu & \text{if } |x_k| = 0 \end{array}. \tag{23}$$

Now, since $\mathcal{C} \not\supseteq \text{supp}(\mathbf{x}^*)$ then $\mathbf{x}*$ is not a minimizer of the fine-level functional, so $0 \notin \partial F(\mathbf{x})$. Therefore, there exists at least one variable $\ell \notin \mathcal{C}$ for which $|a_\ell^T(A\mathbf{x} - \mathbf{y})| > \mu$. Since $x_\ell = 0$, then, according to (8) or (10), after one post-relaxation sweep of either CD or PCD index ℓ will enter the support of $\mathbf{x}$ (for the case of CD, ℓ refers to the first atom in the update order of CD that violates $|a_\ell^T(A\mathbf{x} - \mathbf{y})| \le \mu$). □

¿From the last two propositions we can see the complementary roles of the relaxation and coarse-level correction in our multilevel approach. The relaxation is largely responsible for inserting the correct atoms into the support, while the coarse level is mainly responsible for finding the optimal values of the variables that are in the support.

The next proposition justifies our criterion for the choice of $\mathcal{C}$. Related statements exist for "greedy algorithms" that use the same rationale for choosing the support [14].

Proposition 4. ($\mathcal{C}$-Selection Guarantee.) *Assume for simplicity that the columns of A are normalized such that diag$(A^TA) = \mathbf{1}$, and let δ be the* mutual coherence *of the dictionary A, defined by*

$$\delta = \max_{i \ne j}\{|a_i^T a_j|\}.$$

Let $\mathbf{x}^*$ and $\mathbf{x}$ be the solution and the approximation at the time $\mathcal{C}$ is chosen, respectively, and let $\mathbf{e} = \mathbf{x}^* - \mathbf{x}$ be the error. Let i be an index satisfying $i \in supp(\mathbf{x}^*)$ and $i \notin supp(\mathbf{x})$. Then, so long as $|\mathcal{C}| \geq |supp(\mathbf{x}^*)|$, index i is guaranteed to be included in the coarse-level set $\mathcal{C}$ if

$$|x_i^*| \geq \frac{2\delta}{1+\delta}\|\mathbf{e}\|_1\,. \tag{24}$$

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 k not belonging to supp$(\mathbf{x}^*)$, i.e.,

$$|a_i^T(A\mathbf{x} - y)| > |a_k^T(A\mathbf{x} - y)| \tag{25}$$

for *any* $k \notin \text{supp}(\mathbf{x}) \cup \text{supp}(\mathbf{x}^*)$. To this end, we bound the left-hand side of (25) from below and the right-hand side from above, obtaining (24).

Since $i \in \text{supp}(\mathbf{x}^*)$, then $0 \in \partial F(\mathbf{x})$ and therefore $a_i^T(A\mathbf{x}^* - \mathbf{y}) + \mu\,\text{sign}(x_i^*)$ is equal to zero and may be subtracted from or added to the left-hand side of (25) yielding

$$|a_i^T(A\mathbf{x} - \mathbf{y})| = |a_i^T A\mathbf{e} + \mu\,\text{sign}(x_i^*)| = \left|\textstyle\sum_{j\neq i}(a_i^T a_j)e_j + e_i + \mu\,\text{sign}(x_i^*)\right|, \tag{26}$$

where we have used the fact that the dictionary columns are normalized, $a_i^T a_i = 1$. 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

$$|a_i^T(A\mathbf{x} - \mathbf{y})| \geq |x_i^*| + \mu - \delta\textstyle\sum_{j\neq i}|e_j| = x_i^*(1+\delta) + \mu - \delta\|\mathbf{e}\|_1. \tag{27}$$

Next, we bound from above the right hand side of (25). Since $k \notin \text{supp}(\mathbf{x}^*)$, there exists a scalar $z \in [-1, 1]$ for which $a_k^T(A\mathbf{x}^* - \mathbf{y}) + \mu\mathbf{z}$ is equal to zero and may be subtracted from or added to the right-hand side. Therefore,

$$|a_k^T(A\mathbf{x} - \mathbf{y})| = |a_k^T A\mathbf{e} + \mu z| \leq \delta\|\mathbf{e}\|_1 + \mu, \tag{28}$$

where the last inequality uses $e_k = 0$. Comparing the bounds (27) and (28), we obtain that condition (25) is guaranteed if (24) is satisfied. $\square$

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

3.4. **Convergence and choice of relaxation.**

For the relaxation we may use any of the iterated shrinkage solvers, optionally together with accelerations, as apposed to classical MG where simple iterative methods are mostly considered as relaxations, and acceleration methods usually apply MG as a preconditioner. Since the coarse-level correction cannot increase the functional value, convergence is guaranteed for any relaxation of a type that ensures a sufficient functional value reduction per iteration with any current approximation $\mathbf{x} \neq \mathbf{x}^*$.

3.5. **Treatment of the coarsest level.**

One big difference between our algorithm and classical AMG is in the treatment of the coarsest-level problem. Here, the dimension of the coarsest level is bounded by the size of the current support. Although the final support supp$(\mathbf{x}^*)$ is normally small compared to either m or n, we may not assume that an exact coarsest-level solution has negligible cost as in classical AMG. In this work, we limit the number of relaxations at the coarsest level to 40. Generally speaking, as suggested by proposition 4, dictionaries with low mutual coherence (small δ) yield better coarse-level selection, and therefore it is advisable to invest more work on coarser levels.

3.6. **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 coarse-level operators A_c. Instead, we use a single vector $\mathbf{x}$ and at coarser levels we address only the indices of $\mathbf{x}$ belonging to that level. This saves the need to store and extract the dynamic coarse-level matrices that usually change from one V-cycle to the next. This also lets us apply the SESOP acceleration on consecutive approximations from *different* levels (i.e., while traversing up and down the V-cycle).

4. Numerical Results

In this section we compare the performance of the previously described relaxations, with and without our multilevel acceleration. We run our iterations until the condition

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

is satisfied. This condition is taken from [22] and is related to the size of $\min\{||\partial F(\mathbf{x})||\}$. We do not require extreme accuracy because our aim is to discover the support and its sign-pattern, after which the problem can be dealt with by a linear systems solver. In practice, all solutions achieved by all algorithms correspond to functional values which are identical up to about 8 significant digits and have identical support size.

In our experiment, we first generate a random, normally distributed dictionary $A \in \mathbb{R}^{n\times m}$ ($a_{ij} \sim N(0,1)$), and then manipulate the singular values of the random matrix A to make it highly ill-conditioned, with a condition-number of about 10^{10}. A similar example is used in [22, 32], denoted by $K^{(4)}$. We finalize the generation of the dictionary by normalizing its columns as suggested in [14]. Figure 1 shows an example of the spectrum of the final dictionary A.

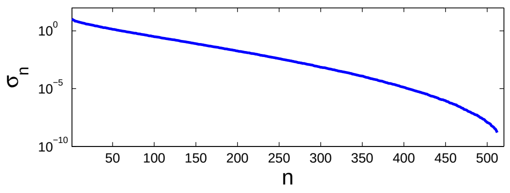

Figure 1. The singular values of the matrix A with $n = 512$

A support S_0 is chosen uniformly from all possible supports of size $[0.1n]$. Then, a vector $\mathbf{x}$ is generated with normally distributed values in the indices corresponding to S_0 and zeros elsewhere. Next, we add random Gaussian noise obtaining the noisy signal

$$\mathbf{y} = A\mathbf{x} + \mathbf{n}\,,$$

with $\mathbf{n} \sim N(0, \sigma_n I)$. In all our experiments we set the noise level to $\sigma_n = 0.1$. Since our experiments are randomly generated, each one is repeated 10 times, and the results are averaged and rounded.

We use V(0,1) multilevel cycles with only one post-relaxation applied in every level. The methods PCD, SESOP1 and CD are described above and the method OR-CD denotes the over-relaxed coordinate descent, which involves a CD iteration followed by a line-search as in (11). All tests are initialized with $\mathbf{x}_0 = 0$. For PCD and SESOP1, a warm-start strategy is performed, starting with a rather high value of $\mu = 0.9\|A^T\mathbf{y}\|_\infty$ and linearly reducing it to the chosen μ over the span of the first 100 iterations. Relaxations without multilevel acceleration are denoted by 1L (one-level).

Table 1 summarizes the performance for all the options that we tested. Each run is described by two numbers: the number of relaxations or V(0,1)-cycles, and (in brackets) the number of work-units, i.e., mn floating point multiplications. We consider only the operations $A\mathbf{x}$ and $A^T\mathbf{y}$ (the latter costing exactly one work-unit because A^T is dense and is applied to a dense vector). All line-searches and minimizations are not counted, as these are quite cheap if programmed properly.

Setting				PCD		SESOP-1		CD		OR-CD	
n	m	μ	$\|S^*\|$	$1L$	ML	$1L$	ML	$1L$	ML	$1L$	ML
128	$4n$	2σ	$0.1n$	751(822)	24(134)	132(146)	3.6(25)	341(352)	8.8(39)	182(222)	5.4(26)
256	$4n$	2σ	$0.09n$	1239(1316)	27(120)	234(254)	5.8(33)	328(336)	8(32)	184(217)	5.1(22)
512	$4n$	2σ	$0.09n$	1223(1394)	34(125)	247(265)	6.3(38)	301(312)	7.4(31)	174(209)	5.4(23)
1024	$4n$	2σ	$0.09n$	1494(1653)	35(145)	328(359)	8.2(49)	298(306)	7.5(30)	183(217)	5.5(24)
512	$1n$	2σ	$0.13n$	151(176)	4.8(63)	87(97)	3.4(38)	33(39)	3.5(35)	23(36)	3.4(29)
512	$2n$	2σ	$0.1n$	625(715)	15(100)	192(214)	5.1(42)	187(197)	6.2(38)	113(151)	4.4(29)
512	$4n$	2σ	$0.09n$	1223(1394)	34(125)	247(265)	6.3(38)	301(312)	7.4(31)	174(209)	5.4(23)
512	$8n$	2σ	$0.08n$	1691(1832)	35(112)	291(311)	7(30)	406(411)	9.3(29)	242(269)	6(20)
512	$4n$	16σ	$0.03n$	150(158)	4.4(16)	53(56)	1.8(8.5)	56(56)	2(8.7)	37(37)	1.7(7.7)
512	$4n$	4σ	$0.06n$	582(624)	15(55)	150(159)	4.3(19)	160(163)	5(20)	96(105)	4.2(17)
512	$4n$	1σ	$0.11n$	2344(2711)	62(285)	436(496)	8.7(80)	523(539)	13(59)	301(392)	8.4(41)
512	$4n$	$\sigma/4$	$0.15n$	4685(5441)	103(727)	963(1302)	14(252)	1525(1595)	31(175)	821(1352)	20(131)

Table 1. Number of iterations, and (in brackets) work-units of mn floating point multiplications each, for the single-level and multilevel algorithms.

The table contains 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.

Regarding the one-level methods, as expected, Table 1 shows that the CD relaxation is much more effective than PCD. (However, as mentioned before, CD has some disadvantages.) Accelerations improve the performance of both methods. For all the one-level results, the cost in work-units is comparable to the number of iterations, implying that most of the work in each iteration is used for computing $A^T\mathbf{y}$.

Table 1 also shows that the multilevel algorithm significantly reduces the cost of the solution, regardless of the relaxation method that is used. In most cases, the cost of the ML method is about 10%-25% of the cost of the 1L relaxations in terms of work-units. In the upper section (growing n), all methods, for both 1L and ML, show quite similar 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 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. As in [32], all 1L methods struggle, as do the ML methods (relatively to their high standards). The cost of the ML methods, however, remains quite reasonable.

5. Conclusions

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. The approach significantly accelerates the performance of existing iterated shrinkage relaxations as well as their accelerated versions.

In the scope of trained dictionaries, where the matrix is given explicitly, greedy algorithms are often preferred over the convex relaxations for solving (3). This is partly due to the lower cost of greedy algorithms compared to the shrinkage methods described above. (The cost of OMP, for example, is roughly $|S_0|$ work-units.) Greedy algorithms, however, may return a poor approximation, especially when the dictionary is ill-conditioned. When applied with an efficient relaxation, the multilevel algorithm presented in this work enjoys both worlds: global optimality using the l_1

norm together with a rather favorable cost. Being so, it is definitely worthy of consideration in this scope.

6. Acknowledgements

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

References

[1] M. Aharon, M. Elad, and A. Bruckstein, *K-svd: An algorithm for designing overcomplete dictionaries for sparse representation*, IEEE Transactions on Signal Processing, 54 (2006), pp. 4311–4322.

[2] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.

[3] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, SIAM, Philadelphia, second ed., 2000.

[4] 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, 51 (2009), pp. 34–81.

[5] E. Candes and T. Tao, *Near-optimal signal recovery from random projections: Universal encoding strategies?*, IEEE Trans. Inform. Theory, 52 (2006), pp. 5406–5425.

[6] E. Candes and M. B. Wakin, *An introduction to compressive sampling*, IEEE Signal Processing Magazine, (March 2008), pp. 21–30.

[7] E. Candes, M. B. Wakin, and S. P. Boyd, *Enhancing sparsity by reweighted l_1 minimization*, Fourier Anal. Appl., 14 (2007), pp. 877–905.

[8] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., 20 (1998), pp. 33–61.

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

[10] D. Donoho, Y. Tsaig, I. Drori, and J.-C. Starck, *Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit*, submitted to IEEE Trans. on Information theory, (2006).

[11] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inf. Theory, 52 (2006), pp. 1289–1306.

[12] M. Elad, *Why simple shrinkage is still relevant for redundant representations?*, IEEE Trans. Inform. Theory, 52 (2006), pp. 5559–5569.

[13] M. Elad, *Optimized projections for compressed-sensing*, IEEE Trans. on Signal Processing, 55 (2007), pp. 5695–5702.

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

[15] M. Elad and M. Aharon, *Image denoising via sparse and redundant representations over learned dictionaries*, IEEE Transactions on Image Processing, 15 (2006), pp. 3736–3745.

[16] M. Elad, M. A. Figueiredo, and Y. Ma, *On the role of sparse and redundant representations in image processing*, to appear in IEEE Proceedings - Special Issue on Applications of Compressive Sensing and Sparse Representation., (2010).

[17] M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky, *A wide angle view at iterated shrinkage algorithms*, SPIE (Wavelet XII), San Diego, CA, (2007).

[18] M. Elad, B. Matalon, and M. Zibulevsky, *Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization*, Appl. Comput. Harmon. Anal., 23 (2007), pp. 346–367.

[19] M. Figueiredo and R. Nowak, *A bound optimization approach to wavelet-based image deconvolution*, IEEE International Conference on Image Processing, (2005).

[20] 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., 15 (2003), pp. 349–396.

[21] M. S. Lewicki and T. J. Sejnowski, *Learning overcomplete representations*, Neural Comput., 12 (2000), pp. 337–365.

[22] I. Loris, *On the performance of algorithms for the minimization of ,l1-penalized functionals*, Inverse Probl., 25 (2009), pp. 1–16.

[23] M. Lustig, D. Donoho, and J. Pauly, *Sparse mri: The application of compressive sensing for rapid mr imaging*, Magnetic Resonance in Medicine, in Press, 58 (2007), pp. 1182–1195.

[24] J. Mairal, G. Sapiro, and M. Elad, *Learning multiscale sparse representations for image and video restoration*, SIAM Multiscale Modeling and Simulation, 7 (2008), pp. 214–241.

[25] S. G. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, IEEE Trans. Signal Processing, 41 (1993), pp. 3397–3415.

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

[27] R. Rubinstein, A. M. Bruckstein, and M. Elad, *Dictionaries for sparse representation modeling*, Proceedings of the IEEE, 98 (2010), pp. 1045–1057.

[28] R. Rubinstein, M. Zibulevsky, and M. Elad, *Double sparsity: Learning sparse dictionaries for sparse signal approximation*, IEEE Transactions on Signal Processing, 58 (2010), pp. 1553–1564.

[29] J. Tropp, *Greed is good: Algorithmic results for sparse approximation*, IEEE Trans. on Information Theory, 50 (2006), pp. 2231–2342.

[30] ———, *Just relax: Convex programming methods for identifying sparse signals*, IEEE Trans. on Information Theory, 51 (2006), pp. 1030–1051.

[31] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, *A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation*, SIAM Sci. Comp., (2009).

[32] M. Zibulevsky and M. Elad, *L1-l2 optimization in signal and image processing: Iterative shrikage and beyond*, IEEE Signal Processing Magazine, (May 2010), pp. 76–88.