MULTILEVEL DECOMPOSITIONS FOR SPARSE SIGNAL RECOVERY IN AN ORTHOGONAL BASIS

TOM GOLDSTEIN AND JUDAH JACOBSON

ABSTRACT. We introduce a multi-level decomposition scheme for solving basis pursuit problems in a Fourier or Hadamard basis. The iterates generated by this scheme are equivalent to the coordinate-descent (CD) method for basis pursuit. However, unlike standard CD methods, the new algorithm computes each iterate using only $O(n \log n)$ operations. The decomposition method is numerically compared to two other common algorithms for basis pursuit. For the problems considered here, runtimes for the CD algorithm are approximately 5-10 times faster than conventional methods.

CONTENTS

1. INTRODUCTION

In this manuscript, we consider a novel method for solving problems of the form

$$\operatorname*{argmin}_{u \in \mathbb{C}^N} \sum_{j=0}^{N-1} \phi(u_j) + \frac{1}{2}\|RTu - s\|^2, \tag{1}$$

where T is a fast transform of the Fourier or Hadamard type, $\phi : \mathbb{C} \to \bar{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$ is closed, proper, and convex, $s \in \mathbb{C}^N$, and R is a diagonal $N \times N$ matrix with nonnegative real entries.

A prominent special case of (14) is the ℓ_1 regularized problem

$$\underset{u\in\mathbb{C}^N}{\operatorname{argmin}} |u| + \frac{1}{2}\|R\mathcal{F}_N u - s\|^2, \tag{2}$$

where $\mathcal{F}_N$ is the N-point discrete Fourier transform defined as

$$[\mathcal{F}_N u]_k := \sum_{j=0}^{N-1} e^{-2\pi i j k/N} u_j. \tag{3}$$

In Section 2.1, we will list several applications which lead to problems of the form (1). These include *compressed sensing, spectral estimation* and *deblurring* problems, among others. These problems are traditionally solved using techniques of the gradient-descent type, which involve repeated application of the sample matrix RT and its conjugate. These operations are computed efficiently in $O(N \log N)$ time using an algorithm such as the Fast Fourier Transform (FFT).

Coordinate descent is another method for solving problems of the form (1). Such methods minimize a function over each coordinate in turn. Coordinate descent is very effective for problems involving dense matrices, but is not practical for problems of the Fourier or Hadamard type – because it cannot directly utilize fast transforms, it requires $O(N^2)$ operations for each coordinate sweep. Sardy et al. [40, 50] proposed a block-level decomposition which optimizes over whole subsets of coordinates at once using a block transform such as the FFT. But, as we will demonstrate, that method does not extend to the general framework of (1).

In this manuscript, we present a new coordinate descent based method for solving (1). We perform an efficient coordinate descent sweep in "bit-reversed" order for the Fourier transform and in the standard order for the Hadamard transform. Unlike traditional coordinate descent, each sweep takes only $O(N \log N)$ operations, and does not require any explicit orthogonal transforms.

The organization of this paper is as follows: We first review some common techniques for solving regression problems of the form (1). We also review prior work on coordinate descent for basis pursuit problems, and discuss the limitations of this method for (1). We then show how the *Cooley-Tukey* and *Sylvester* decompositions lead to efficient multilevel schemes for problems involving the Fourier and Hadamard transforms. Finally, we present numerical results demonstrating the efficiency of our method for these problems.

1.1. Notation and Definitions.

We first define a few bits of notation. To avoid cumbersome summations, we shall denote the discrete 2-norm and 1-norm as follows:

$$\|u\| = \sqrt{u^* u} = \sqrt{\sum_{i=1}^{n} |u_i|^2}, \qquad |u| = \sum_{i=1}^{n} |u_i|.$$

Suppose that W is a real diagonal $N \times N$ matrix with nonnegative entries. Then we define the weighted seminorm $\|\cdot\|_W$ by

$$\|u\|_W^2 = u^* W u. \tag{4}$$

In general, $\|\cdot\|_W$ is not a norm unless each $W_{i,i} > 0$. Additionally, recall that the Moore-Penrose pseudoinverse W^+ satisfies $WW^+W = W$. Since W is diagonal,

the pseudoinverse can be written explicitly as

$$W^+_{i,i} = \begin{cases} 1/W_{i,i} & \text{if } W_{i,i} \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

As a consequence,

$$[WW^+]_{i,i} = \begin{cases} 1 & \text{if } W_{i,i} \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

2. Background

2.1. **Applications.** We begin by listing several applications for which our method can be used. They each lead to a minimization problem which involves an orthogonal transform in some way. We will demonstrate how those problems can be rewritten in the form (1). Note in particular for the Fourier transform that a change of variables $v_j = u_{(N-j)(\mathrm{mod}N)}$ yields $\overline{\mathcal{F}_N(u)} = \mathcal{F}^*_N(v)$, and thus the equivalent problem

$$\operatorname*{argmin}_{v\in\mathbb{C}^N} \sum_{j=0}^{N-1} \phi(v_j) + \frac{1}{2}\|R\mathcal{F}^*_N v - s\|^2, \tag{7}$$

which is more appropriate for some applications.

A prominent example is that of *compressed sensing* (CS), where we wish to reconstruct u from a small subset of its Fourier coefficients [7, 8, 9, 14, 39, 47]. We choose $\Phi(u) = \sum_i \phi(u_i)$ to be some regularizer which promotes sparsity of the solution, such as the scaled complex ℓ_1 norm $\phi(u_i) = \lambda|u_i|$. If the kth Fourier coefficient is known, we let s_k be that coefficient and set $R_{k,k} = 1$. If the kth mode is unknown, we set $s_k = R_{k,k} = 0$. Problems of this form also arise in analog-to-digital conversion [48, 25] and statistical regression [44].

A related problem arises from the *basis pursuit denoising* method [11]. Suppose that an observed signal $f \in \mathbb{C}^N$ is known to be a linear sum of a small number of sinusoids. Such signals can arise in many fields, including speech processing, radar and sonar systems, astronomy, and seismology [42]. We let M be a positive integer, and consider a "dictionary" of complex exponentials $g_k \in \mathbb{C}^N$:

$$[g_k]_j = e^{2\pi ijk/(MN)}, \quad 0 \leq k < MN, \quad 0 \leq j < N. \tag{8}$$

When $M = 1$, this dictionary is just the standard Fourier basis. When $M > 1$, however, the dictionary contains extra sinusoids at the fractional frequencies $k/M = m_1 + m_2/M$, and is thus over-complete. We aim to find a linear combination of a small number of g_k which closely matches the observed signal.

The 'basis pursuit' scheme solves

$$\operatorname*{argmin}_{u\in\mathbb{C}^{MN}} \lambda|u| + \frac{1}{2}\|Au - f\|^2, \tag{9}$$

where the columns of A are the elements of our dictionary. In particular,

$$[Au]_k = \sum_{j=0}^{N-1} e^{2\pi ijk/(MN)} u_j, \quad k = 0, \ldots, N-1. \tag{10}$$

Thus (9) is equivalent to the problem

$$\underset{u\in\mathbb{C}^{MN}}{\operatorname{argmin}} \lambda|u| + \frac{1}{2}\|R\mathcal{F}_{MN}^{*}u - s\|^2, \tag{11}$$

where $s \in \mathbb{C}^{MN}$ and R is an $MN \times MN$ diagonal matrix with $R_{i,i} = 1$, $s_i = f_i$ when $i < N$, and $R_{i,i} = 0$, $s_i = 0$ otherwise.

Another class of problems that can be represented in the form (1) are *sparse deconvolution problems*, which have the form

$$\underset{u\in\mathbb{C}^{N}}{\operatorname{argmin}} \lambda|u| + \frac{1}{2}\|Ku - s\|^2 \tag{12}$$

where K is a convolution matrix. These problems arise, for example, in heat-source identification, seismology, and medical imaging applications [29, 33, 34, 35, 43, 20, 17]. Often $K = \mathcal{F}_N^{*}R\mathcal{F}_N$ for some diagonal matrix R. Then, because of the unitary nature of the Fourier transform,

$$\|Ku - s\|^2 = \|\mathcal{F}_N^{*}R\mathcal{F}_N u - s\|^2 = N\,\|R\mathcal{F}_N u - \tfrac{1}{N}\mathcal{F}_N s\|^2$$

where the rightmost form of this energy is the same as that appearing in (1). Thus problems of the form (12) can also be written in the form (1).

Stern et al. [41] have applied a similar approach to the computation of spectra from NMR data. Spectrometer measurements suffer from a decay w; that is, if the physical signal produced time samples f_j, they would be measured as $s_j = w_j f_j$. The raw observed signal s has the blurred spectrum $\mathcal{F}(s) = \mathcal{F}(w) * \mathcal{F}(f)$. To "deconvolve" the spectrum, we can instead solve

$$\underset{u\in\mathbb{C}^{N}}{\operatorname{argmin}} \lambda|u| + \frac{1}{2}\|WR\mathcal{F}_{MN}^{*}u - s\|^2. \tag{13}$$

where W is a diagonal matrix representing the decay factor, and R is as in (11).

All of the above examples have used the ℓ^1 norm as a regularizer. An alternate technique for enforcing sparsity is *non-negative least squares*, which forces the recovered signal to take on only positive values. Non-negative least squares corresponds to problems of the form (1) where we choose

$$\phi^{+}(u_j) = \begin{cases} 0 & \text{if } u_j \in \mathbb{R} \text{ and } u_j \geq 0, \\ \infty & \text{otherwise.} \end{cases}$$

The use of non-negative least squares for compressed sensing, and the uniqueness of solutions to this problem, in studied in [6, 15].

2.2. **Iterative Methods for Sparse Signal Recovery.** In this section we review commonly used methods for finding sparse solutions to systems of equations. These algorithms apply to problems of the form

$$\min_{u\in\mathbb{C}^{N}} \Phi(u) + \frac{1}{2}\|Au - s\|^2 \tag{14}$$

where $A : \mathbb{C}^N \to \mathbb{C}^M$ is an arbitrary linear operator and $\Phi : \mathbb{C}^N \to \bar{\mathbb{R}}$ is closed, proper and convex.

FBS. A very common approach to solving problems of the form (14) is to use a gradient-descent-based method such as forward-backward splitting. Algorithms of this form were originally proposed for differentiable problems by Lions and Mercier [28] and Passty [37], and later studied extensively by others [10, 32, 51]. Rigorous results for ℓ_1-regularized problems were first proposed by Hale, Yin, and Zheng in [23].

Forward-Backward splitting is a two stage algorithm that operates on some initial guess u^k. During the first stage, we obtain $\bar{u}^k$ using a gradient descent step on the differentiable term in (14).

$$\bar{u}^k = u^k - tA^T(Au^k - s).$$

During the second stage, we update the value of $\bar{u}^k$ by solving the "proximal" problem

$$u_{k+1} = \operatorname*{argmin}_{u\in\mathbb{C}^N} \Phi(u) + \frac{1}{2t}\|u - \bar{u}^k\|$$

We can write this equation as $u_{k+1} = \mathrm{Prox}_{t\Phi}(\bar{u}^k)$, where we define

$$\mathrm{Prox}_{\Phi}(u) = \operatorname*{argmin}_{v\in\mathbb{C}^N} \Phi(v) + \frac{1}{2}\|v - u\|^2. \tag{15}$$

The map Prox_Φ is uniquely defined for all closed, proper, convex Φ; see for example [32].

The overall algorithm can be written

Algorithm 1 Forward-Backward Splitting (FBS)

1: Initialize: $u^0 \in \mathbb{C}^N$, $t < 2/\rho(A^TA)$
2: **for** $k = 0, 1, 2, \cdots$ **do**
3: $\quad \bar{u}^k = u^k - tA^T(Au^k - s)$
4: $\quad u_{k+1} = \mathrm{Prox}_{t\Phi}(\bar{u}^k)$
5: **end for**

For the problems considered here, the minimization in step 4 of the above algorithm has a simple closed form solution. For ℓ_1 regularized problems, we have $\Phi(u) = \lambda|u|$, and

$$\left[\mathrm{Prox}_{t\lambda|\cdot|}(u)\right]_j = \frac{u_j}{|u_j|}\max\{|u_j| - t\lambda, 0\}. \tag{16}$$

In the case of non-negative least squares, we have

$$\left[\mathrm{Prox}_{t\Phi^+}(u)\right]_j = \max\{u_j, 0\}. \tag{17}$$

Note that the FBS algorithm (1) only requires that we be able to evaluate the linear operator A and its adjoint. In case the operator A involves a Fourier transform, step 3 of the FBS algorithm can be evaluated quickly using the fast Fourier transform (FFT). This makes FBS advantageous for problems involving the Fourier transform (and other fast transforms). Another advantage of the FBS method is its extremely simple implementation.

FISTA. The Fast Iterative Shrinkage and Thresholding (FISTA) method of Beck and Teboulle [2] is an accelerated variant of forward-backward splitting. The scheme can be thought of as a predictor-corrector method. A 'prediction' is made by moving in the direction indicated by the difference between the last two iterates. This predicted value is then 'corrected' by applying the forward-backward iteration. This is described as algorithm 2 below:

Algorithm 2 FISTA

Require: $v_1 = u_0 \in \mathbb{C}^N$, $\alpha_1 = 1$, $\tau < 1/\rho(A^T A)$

1: **for** $k = 1, 2, 3, \cdots$ **do**
2: $\quad u_k = \text{Prox}_{t\Phi}(v^k - tA^T(Av^k - s))$
3: $\quad \alpha_{k+1} = (1 + \sqrt{1 + 4\alpha_k^2})/2$
4: $\quad v_{k+1} = u_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(u_k - u_{k-1})$
5: **end for**

Note that the over-relaxation parameter α_k increases with time. By changing the level of over-relaxation, the authors of [2] show that FISTA achieves the best possible worst-case convergence rate for splitting problems involving one smooth term. For this reason, FISTA is generally considered to be a more efficient solver than simple FBS.

Subspace Optimization / CGIST. While the signals we wish to recovery using problems of the form (14) often contain many coefficients, the search for these signals often takes place within very low-dimensional subspaces of $\mathbb{C}^N$. This is because most coefficients in the solution are zero (i.e. solutions are sparse), and only a small number of entries lie in the active set.

Most first-order schemes (e.g. FBS and FISTA) rely on fixed time steps that are stable for every possible active set. However, when the active set is small, the maximum allowable timestep within a low-dimensional subspace is often much larger than the globally stable timestep. For this reason, an adaptive timestep often outperforms a fixed timestep by a large margin. Furthermore, by using 'subspace optimization,' which minimizes directly within these low dimensional spaces, it is possible to moved quickly towards a minimizer even in cases where FBS or FISTA are slow because of poorly conditioned active sets.

Many of the most modern ℓ_1 solvers incorporate adaptive stepsizes and subspace optimization. The FPC_AS and FPC_BB solvers [53] rely on a Barzilai-Borwein timestep [1], and call a dedicated quadratic program solver to perform subspace minimization. Another approach is the semi-smooth Newton method of Griesse and Lorenz [22]. This method formulates and directly inverts the reduced Hessian within the subspace corresponding to the current active set.

One particularly efficient method is Conjugate-Gradient Iterative Shrinkage / Threshold method (CGIST) [21]. This method exploits the quadratic nature of (14) to compute the optimal step size at each iteration. The method also incorporates a simple acceleration step that allows the method to perform conjugate-gradient acceleration within the active subspace. This acceleration step allows CGIST to perform subspace optimization without the overhead of making a call to dedicated quadratic program solver. For this reason, CGIST tends to be faster than many other accelerated schemes in cases where the active set evolves quickly. The CGIST

code provided by the authors of [21] is also capable of solving problems involving a variety of different constraints, including constraints involving the ℓ_2 norm.

NESTA. Recently, the code NESTA [3] has become a popular choice for solving a variety of ℓ_1 regularized problems. The code is capable of solving ℓ_2 norm constrained problems, however the authors also provide the code NESTA_UP which solves problems of the form (14). This method works by regularizing the dual of (14). The chosen regularization is equivalent to the popular Huber regularization of ℓ_1, which replaces the non-differentiable region of the ℓ_1 norm with a smooth quadratic term. The resulting formulation is then minimized using Nesterov's optimal first-order method [31]. Rigorous convergence results for this regularized scheme have been proved by Nesterov [30]. In particular, this scheme achieves the same worst-case asymptotic error bounds as FISTA.

Coordinate Descent. Forward-Backward Splitting is a gradient descent based minimization technique. For general basis pursuit problems (e.g. problems not involving the Fourier transform or other fast transforms), faster methods are available. For unconstrained problems, coordinate descent (CD) is frequently the most efficient approach. These techniques work by minimizing (14) with respect to each individual element u_i in sequence.

Algorithm 3 Cyclic Coordinate Descent (CD)

1: Initialize: $u^0 \in R^N$
2: **for** $k = 1, 2, \cdots$ **do**
3: **for** $i = 1, 2, \cdots, N$ **do**
4: $u_i^k \leftarrow \text{argmin}_{u_i \in \mathbb{C}} \Phi(u) + \frac{1}{2}\|Au - s\|^2$
 such that $u = (u_1^k, \ldots, u_{i-1}^k, u_i, u_{i+1}^{k-1}, \ldots, u_N^{k-1})$
5: **end for**
6: **end for**

Methods of this type were proposed very early by Fu [19], and were later popularized by Daubechies et al. [13]. Variants of this algorithm have been studied extensively for various applications including the elastic net [55, 52], non-negative least squares [5], grouped regression [54], and many other applications [26, 50, 49]. Variants of this algorithm have also been proposed for TV regularized problems (also called the "fused lasso") by Tibshirani and others [45, 46, 4, 36]. A detailed review of many of these techniques can be found in [18].

Not only does CD converge more quickly than FBS, but for dense A the cost of a CD iteration is lower then the cost of an FBS iteration. These two factors make CD a superior solver when speed is the primary consideration.

Unfortunately, CD is generally not effective for basis pursuit problems involving the Fourier or Hadamard transforms (see, for example, [27]). Each execution of line 4 of the above algorithm turns out to require $O(N)$ operations, since it computes inner products against individual columns of the sample matrix. As a result, each iteration of CD requires $O(N^2)$ computations. In contrast, each iteration of FBS involves a single application of the matrix A and its transpose, which can generally be performed in $O(N \log N)$ using a fast transform.

Block Coordinate Descent. Sardy et al. [40, 50] proposed an improvement for CD which can use fast transforms for certain problems. Consider the basis pursuit problem (14); suppose that $A \in \mathbb{C}^{N\times(NM)}$ can be written in the block form $(A_1 \cdots A_M)$ where each A_m is unitary, and furthermore that $\Phi : \mathbb{C}^{MN} \to \bar{\mathbb{R}}$ is separable: $\Phi(u_1, \ldots, u_M) = \sum_m \Phi_m(u_m)$. Then (14) can be written in the block form

$$\operatorname*{argmin}_{u_j\in\mathbb{C}^N} \sum_{m=1}^{M} \Phi_m(u_m) + \frac{1}{2}\|\sum_{m=1}^{M} A_m u_m - s\|^2. \tag{18}$$

BCD proceeds by minimizing over an entire block of coordinates at once, rather than over each coordinate separately as in Algorithm 3. Minimizing (18) over the block u_j results in:

$$\operatorname*{argmin}_{u_j\in\mathbb{C}^N} \Phi_j(u_j) + \frac{1}{2}\Big\|u_j + A_j^*\Big(\sum_{m\neq j} A_m u_m - s\Big)\Big\|^2 \tag{19}$$

$$= \operatorname{Prox}_{\Phi_j}\Big[A_j^*\big(s - \sum_{m\neq j} A_m u_m - s\big)\Big], \tag{20}$$

which as mentioned before has a unique closed-form solution for many choices of Φ_j, including the complex ℓ_1 norm. This results in the following algorithm:

Algorithm 4 Block Coordinate Descent (BCD)

1: Initialize: $u_k^0 \in \mathbb{C}^N$ for $k = 1, \ldots, M$
2: **for** $k = 1, 2, \cdots$ **do**
3: **for** $j = 1, 2, \cdots, M$ **do**
4: $u_j^k \leftarrow \operatorname{Prox}_{\Phi_j}\big[A_j^*(s - \sum_{m<j} A_m u_m^k - \sum_{m>j} A_m u_m^{k-1})\big]$
5: **end for**
6: **end for**

We can make BCD more efficient by observing that line 4 applies $M-1$ unitary block matrices A_m to the input variables u_m, even though only one of them changes at each step. We can "cache" those values by operating instead on the new variables $v_m = A_m u_m$. Then the method becomes:

Algorithm 5 Block Coordinate Descent, fast version

1: Initialize: $v_k^0 \in \mathbb{C}^N$ for $k = 1, \ldots, M$
2: **for** $k = 1, 2, \cdots$ **do**
3: **for** $j = 1, 2, \cdots, M$ **do**
4: $v_j^k \leftarrow A_j \operatorname{Prox}_{\Phi_j}\big[A_j^*(s - \sum_{m<j} v_m^k - \sum_{m>j} v_m^{k-1})\big]$
5: **end for**
6: **end for**

Now the update step for each v_j^k requires only a single application of A_j, A_j^* and $\operatorname{Prox}_{\Phi_j}$. Thus we can expect Algorithm 5 to be faster than Algorithm 4 by approximately a factor of M.

BCD turns out to apply directly to the basis pursuit problem (9) in the *unweighted* case, i.e., when $W = I$. Suppose that the columns of $A \in \mathbb{C}^{N\times(MN)}$ make

up a dictionary of complex exponentials with fractional frequencies:

$$[A]_{k,j} = \tfrac{1}{\sqrt{N}} e^{2\pi ijk/(MN)} \quad 0 \le j \le N-1, \;\; 0 \le k \le MN-1. \tag{21}$$

We reorder the coordinates of u by letting $v_0, \dots, v_{M-1} \in \mathbb{C}^N$ be defined by $[v_m]_k = u_{Mk+m}$. Then

$$[Au]_j = \tfrac{1}{\sqrt{N}} \sum_{k=0}^{MN-1} e^{2\pi ijk/(MN)} u_k \tag{22}$$

$$= \tfrac{1}{\sqrt{N}} \sum_{m=0}^{M-1} \sum_{k=0}^{N-1} e^{2\pi ij(Mk+m)/(MN)} [v_m]_k \tag{23}$$

$$= \sum_{m=0}^{M-1} e^{2\pi ijm/(MN)} \tfrac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} [v_m]_k \tag{24}$$

$$= \sum_{m=0}^{M-1} [\Psi^m \tfrac{1}{\sqrt{N}} \mathcal{F}_N^* v_m]_j, \tag{25}$$

where we let $\Psi \in \mathbb{C}^{N \times N}$ be a diagonal matrix defined by

$$[\Psi]_{j,j} = e^{2\pi ij/(MN)}. \tag{26}$$

Thus (9) is equivalent to

$$\operatorname*{argmin}_{v_0,\dots,v_{M-1} \in \mathbb{C}^N} \sum_m |v_m| + \frac{1}{2} \| \sum_m A_m v_m - s \|^2, \tag{27}$$

where each $A_m = \Psi^m \frac{1}{\sqrt{N}} \mathcal{F}_N^*$ is unitary. As a result, it can be directly solved with the BCD method.

Unfortunately, the block-unitary assumption does not hold for the more general problems under discussion. For example, in compressed sensing problems the observed Fourier modes are picked at random and thus do not form unitary blocks. As another example, consider the NMR problem (13) which involves a nontrivial weighing factor. Following the above steps in that case will produce the blocks $W\Psi^m \frac{1}{\sqrt{N}} \mathcal{F}_N^*$, which in general are not unitary.

2.3. **The Cooley-Tukey FFT.** One of the most efficient schemes for computing the DFT is the Cooley-Tukey factorization [12, 16, 38]. Although this type of procedure can be performed on vectors of arbitrary composite size, we will assume for simplicity that the signal length is a power of 2. In this case, we get a radix 2 decimation-in-time algorithm in which the Fourier transform of a length N signal is represented as a linear combination of smaller Fourier transforms of size $n = N/2$.

The Cooley-Tukey factorization proceeds by splitting the input into even and odd coefficients. Let us define $\mathrm{Ev}_n, \mathrm{Od}_n : \mathbb{C}^(2n) \to \mathbb{C}^n$ to be the linear operators which select the even and odd coefficients of their inputs, respectively:

$$[\mathrm{Ev}_n u]_j = u_{2j}, \quad [\mathrm{Od}_n u]_j = u_{2j+1}. \tag{28}$$

The operator $\begin{pmatrix} \mathrm{Ev}_n \\ \mathrm{Od}_n \end{pmatrix} : \mathbb{C}^{2n} \to \mathbb{C}^{2n}$ is orthogonal and permutes its input by separating the even and odd coefficients. Its inverse is $\begin{pmatrix} \mathrm{Ev}_n^T & \mathrm{Od}_n^T \end{pmatrix}$, which interleaves

the first half of its inputs' coefficients with the second half:

$$\left[\begin{pmatrix}\mathrm{Ev}_n^T & \mathrm{Od}_n^T\end{pmatrix}\begin{pmatrix}u_e\\u_o\end{pmatrix}\right] = \begin{cases}[u_e]_j & \text{if } k=2j,\\ [u_o]_j & \text{if } k=2j+1.\end{cases} \tag{29}$$

Theorem 1. *Suppose that N is even and $n = N/2$. Let D_n be the diagonal matrix defined by*

$$[D_n]_{k,k} = e^{-2\pi ik/N}. \tag{30}$$

Then

$$\mathcal{F}_N = \begin{pmatrix}I_n & D_n\\ I_n & -D_n\end{pmatrix}\begin{pmatrix}\mathcal{F}_n\mathrm{Ev}_n\\ \mathcal{F}_n\mathrm{Od}_n\end{pmatrix}. \tag{31}$$

Proof. We can write the Fourier transform component-wise as

$$[\mathcal{F}_n u]_k = \sum_{j=0}^{N-1} e^{-2\pi ijk/N}u_j \tag{32}$$

$$= \sum_{j=0}^{N-1} e^{-2\pi i(2j)k/N}u_{2j} + \sum_{j=0}^{N-1} e^{-2\pi i(2j+1)k/N}u_{2j+1} \tag{33}$$

$$= [\mathcal{F}_n\mathrm{Ev}_n u]_k + e^{-2piik/N}[\mathcal{F}_n\mathrm{Od}_n u]_k, \tag{34}$$

from which the result follows. □

Using the Cooley-Tukey decomposition, we have written the Fourier matrix as a product of a block-diagonal 'butterfly matrix' and a set of smaller Fourier matrices. Because of the block-diagonal structure of the butterfly matrix, multiplication by this matrix can be performed in $O(N)$ operations. The Fourier transforms of the smaller matrices are performed by recursively applying the Cooley-Tukey formula.

Algorithm 6 : **FFT**(u, N) (Cooley-Tukey)

1: **if** $N = 1$ **then**
2: **return** u
3: **else**
4: $n \leftarrow N/2$
5: $v_e \leftarrow$ **FFT**$(\mathrm{Ev}_n u, n)$
6: $v_o \leftarrow$ **FFT**$(\mathrm{Od}_n u, n)$
7: **return** $\begin{pmatrix}v_e + D_n v_o\\ v_e - D_n v_o\end{pmatrix}$
8: **end if**

A simple induction argument shows that the total cost of computing the N-point FFT using this scheme is $O(N \log N)$.

Another way to describe the Cooley-Tukey FFT's structure is as a bit-reversal permutation of the inputs. For an input $u \in \mathbb{C}^N$, the method first considers $\mathrm{Ev}_n u$ and then $\mathrm{Od}_n u$; that is, first those indices whose binary representation ends in 0, and then those whose representation ends in 1. Proceeding inductively, we conclude that the method considers indices in an order determined by reversing their binary representations.

As an illustration, we give pseudo-code for a bit-reversed enumeration procedure:

Algorithm 7 : **bitreversed**$(k_0, ..., k_{N-1})$ Bit-reversed enumeration

1: **if** $N = 1$ **then**
2: output k_0
3: **else**
4: **bitreversed**$(\mathrm{Ev}_n(k_0, ...k_{N-1}))$
5: **bitreversed**$(\mathrm{Od}_n(k_0, ..., k_{N-1}))$
6: **end if**

Calling **bitreversed**$(0, 1, ...N-1)$ will output the indices between 0 and $N-1$ in bit-reversed order.

2.4. **The Fast Hadamard Transform.** A Hadamard matrix is an orthogonal matrix, whose entries are either 1 or -1. While it is conjectured that Hadamard Matrices exist of every order divisible by 4, constructions for these matrices only exist in limited cases [24]. The simplest construction occurs for matrices whose order is a power of 2. In this case, the Sylvester construction builds Hadamard matrices using the recursive formula

$$H_N = \begin{pmatrix} H_{N/2} & H_{N/2} \\ H_{N/2} & -H_{N/2} \end{pmatrix} \tag{35}$$

where H_N denotes the $N \times N$ Hadamard matrix, and $H_1 = 1$.

Like the Fourier Transform, the N-point Hadamard transform $H_N u$ can be performed in $O(N \log N)$ operations. This is done by computing the $N/2$-point Hadamard transform of the upper and lower halves of the vector u, and then summing and differencing the results to form the Sylvester composition (35). The Hadamard transform of each half of u is performed by recursively applying the decomposition (35).

3. CD Minimization in a Fourier Basis

In this section, we will demonstrate how to perform fast CD minimization for the general problem (1). We will assume for simplicity that the signal length is a power of 2. Our method will work by subdividing the original problem into two smaller subproblems, solving those problems, and then recombining the solutions. In this sense, the method is a divide-and-conquer algorithm much like the FFT itself.

Our method will minimize functionals of the form

$$E_N(u) = \Phi_N(u) + \frac{1}{2}\|\mathcal{F}_N u - s\|_W^2, \tag{36}$$

where, as before,

$$\Phi_N(u) = \sum_{i=0}^{N-1} \phi(u_i). \tag{37}$$

Any problem of our original form (1) may be converted to a problem in form (36), as follows. Suppose that we have $f \in \mathbb{C}^N$ and $R \in \mathbb{R}^N$. Let $W_{i,i} = |R_{i,i}|^2$,

and let $s = R^+ f$. Then

$$\|R\mathcal{F}_N u - f\|^2 = \|R\mathcal{F}u - RR^+ f\|^2 + \sum_{\substack{0\le i<N,\\ \bar{R}_i=0}} |f_i|^2 \tag{38}$$

$$= \|\mathcal{F}_N u - s\|_W^2 + C, \tag{39}$$

where C does not depend on u. As a result, minimization of (36) is equivalent to the problem (1).

3.1. **Bit-reversed Coordinate Descent.** For any generic function $E : \mathbb{C}^N \to \bar{\mathbb{R}}$, we can perform a bit-reversed CD sweep recursively, as follows:

Algorithm 8 : $u^{k+1} \leftarrow$ **bitreversedCD**$(u^k, E(\cdot), N)$

1: **if** $N = 1$ **then**
2: **return** $\operatorname{argmin}_{u\in\mathbb{C}} E(u)$
3: **else**
4: $\begin{pmatrix} u_e^k \\ u_o^k \end{pmatrix} \leftarrow \begin{pmatrix} \mathrm{Ev}_n \\ \mathrm{Od}_n \end{pmatrix} u^k$
5: Define $E(u_e, u_o) = E\left(\begin{pmatrix} \mathrm{Ev}_n^T & \mathrm{Od}_n^T \end{pmatrix} \begin{pmatrix} u_e \\ u_o \end{pmatrix} \right)$
6: $u_e^{k+1} \leftarrow$ **bitreversedCD**$(u_e^k, E(\cdot, u_o^k), N/2)$
7: $u_o^{k+1} \leftarrow$ **bitreversedCD**$(u_o^k, E(u_e^{k+1}, \cdot), N/2)$
8: **return** $\begin{pmatrix} \mathrm{Ev}_n^T & \mathrm{Od}_n^T \end{pmatrix} \begin{pmatrix} u_e^{k+1} \\ u_o^{k+1} \end{pmatrix}$
9: **end if**

In the base case (line 2), we solve a one-dimensional subproblem. For every other level, we split the coefficients into even and odd subsets. We first sweep recursively over the even coefficients while keeping the odd coefficients fixed. Then, we sweep recursively over the odd coefficients while fixing the even coefficients. Finally, we merge the resulting even and odd coefficients and return the result.

3.2. **Splitting the weighted Fourier objective.** We now rewrite the energy E_N in terms of the even- and odd-indexed components of u: $u_e = \mathrm{Ev}_n u$ and $u_o = \mathrm{Od}_n u$. The regularizer $\Phi_N(u) = \sum_{i=0}^{N-1} \phi(u_i)$ can be easily decomposed in this way:

$$E_N(u) = E_N(u_e, u_o) = \Phi_{N/2}(u_e) + \Phi_{N/2}(u_o) + \frac{1}{2}\|\mathcal{F}_N u - s\|_W^2. \tag{40}$$

However, we must also find a way to decouple the even- and odd-indexed components of u in the fidelity term $\|\mathcal{F}_N u - s\|_W^2$. This is provided by the following theorem.

Theorem 2. *Let $W_1, W_2 \in \mathbb{R}^{n\times n}$ be nonnegative diagonal matrices, and let $s_1, s_2 \in \mathbb{C}$. Let*

$$W = \begin{pmatrix} W_1 & 0 \\ 0 & W_2 \end{pmatrix}, \tag{41}$$

$$s = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}, \tag{42}$$

$$F(u) = \|\mathcal{F}_{2n}u - s\|_W^2. \tag{43}$$

Then F can be rewritten in the following two equivalent forms:

$$F(u_e, u_o) = \|\mathcal{F}_n u_e - s_e\|_{W_1+W_2}^2 + C_e, \tag{44}$$

$$F(u_e, u_o) = \|\mathcal{F}_n u_o - s_o\|_{W_1+W_2}^2 + C_o, \tag{45}$$

where C_e and C_o are constants not depending on u_e and u_o, respectively, and

$$s_e = (W_1 + W_2)^+ \left[W_1 s_1 + W_2 s_2 + (W_2 - W_1) D_n \mathcal{F}_n u_o\right], \tag{46}$$

$$s_o = (W_1 + W_2)^+ D_n^* \left[W_1 s_1 - W_2 s_2 + (W_2 - W_1) \mathcal{F}_n u_e\right]. \tag{47}$$

Proof. We first note that for any $a, b \in \mathbb{C}^n$ we have

$$[(W_1 + W_2)(W_1 + W_2)^+](W_1 a + W_2 b) = W_1 a + W_2 b. \tag{48}$$

This follows from (6), since $[W_1 + W_2]_{i,i} = 0$ exactly when both $[W_1]_{i,i} = 0$ and $[W_2]_{i,i} = 0$.

Now, using the Cooley-Tukey decomposition, we can rewrite

$$F(u) = \|\mathcal{F}_n u_e + D_n \mathcal{F}_n u_o - s_1\|_{W_1}^2 + \|\mathcal{F}_n u_e - D_n \mathcal{F}_n u_o - s_2\|_{W_2}^2 \tag{49}$$

$$= \|\mathcal{F}_n u_e\|_{W_1}^2 - 2\big\langle \mathcal{F}_n u_e, W_1(s_1 - D_n \mathcal{F}_n u_o)\big\rangle \tag{50}$$

$$+ \|\mathcal{F}_n u_e\|_{W_2}^2 - 2\big\langle \mathcal{F}_n u_e, W_2(s_2 + D_n \mathcal{F}_n u_o)\big\rangle + C_e' \tag{51}$$

$$= \|\mathcal{F}_n u_e\|_{W_1+W_2}^2 - 2\big\langle \mathcal{F}_n u_e, (W_1 + W_2) s_e\big\rangle + C_e', \tag{52}$$

which is equivalent to (44). Note that equation (52) follows from (48).

Similarly,

$$F(u) = \|\mathcal{F}_n u_o + D_n^*(\mathcal{F}_n u_e - s_1)\|_{W_1}^2 + \|\mathcal{F}_n u_o - D_n^*(\mathcal{F}_n u_e - s_2)\|_{W_2}^2 \tag{53}$$

$$= \|\mathcal{F}_n u_o\|_{W_1+W_2} - 2\big\langle \mathcal{F}_n u_o, W_1 D_n^*(s_1 - \mathcal{F}_n u_e)\big\rangle \tag{54}$$

$$- 2\big\langle \mathcal{F}_n u_o, W_2 D_n^*(-s_2 + \mathcal{F}_n u_e)\big\rangle + C_o' \tag{55}$$

$$= ||\mathcal{F}_n u_o||_{W_1+W_2} - 2\big\langle \mathcal{F}_n u_o, (W_1 + W_2) s_o\big\rangle + C_o', \tag{56}$$

which is equivalent to (45). □

We now describe how the two forms from the previous theorem can be used to perform coordinate descent. Suppose first that we want to perform coordinate descent on the even-indexed elements of u, while fixing the odd-indexed elements. For this step, the energy E_N can be written as

$$\Phi_n(u_e) + \frac{1}{2}\|\mathcal{F}_n u_e - s_e\|_{W_1+W_2}^2, \tag{57}$$

where we omit terms not depending on u_e. Similarly, to perform element-wise minimization of the odd-indexed components of u, we need only perform a coordinate

sweep on the following functional:

$$\Phi_n(u_o) + \frac{1}{2}\|\mathcal{F}_n u_o - s_o\|^2_{W_1+W_2}. \tag{58}$$

The sweep of each small (size $n = N/2$) problem is performed recursively using the same decomposition that we used for problems of size N. On each stage of the recursion, the problem size gets reduced by a factor of 2. The recursion terminates when the problem size has been reduced to 1, and we must then solve the resulting problem analytically. The length-1 problem has the form

$$\operatorname*{argmin}_{u\in\mathbb{C}^n} \phi(u) + \frac{w}{2}\|\mathcal{F}_1 u - s\|^2 \tag{59}$$

$$= \text{Prox}_{(w^{-1}\phi)}(s), \tag{60}$$

since $\mathcal{F}_1 = I_1$. As noted before, (60) is uniquely defined for any proper, lower-semicontinuous, convex ϕ, and is easily solvable for many choices of ϕ. In particular, for ℓ_1 optimization we have

$$\operatorname*{argmin}_{u\in\mathbb{C}} \lambda|u| + \frac{w}{2}|u-s|^2 = \frac{s}{|s|}\max\{s - \lambda/w, 0\}. \tag{61}$$

Remark: Note that our formula (60) requires a nonzero w. Fortunately, it turns out that $w > 0$ whenever the original $N \times N$ weight matrix W_N is nonzero. (If $W = 0$, the problem (36) becomes trivial.) To show that $w > 0$, we first observe that for any diagonal matrix $W = \left(\begin{smallmatrix} W_1 & 0 \\ 0 & W_2 \end{smallmatrix}\right)$, its trace $\text{tr}(W) = \text{tr}(W_1 + W_2)$. Then, proceeding by induction, we find that that the trace of the weight matrix is the same in all subproblems. In particular, for the size-1 problem (59), $w = \text{tr}(W_N)$. But since each diagonal entry of W_N is nonnegative, $\text{tr}(W_N) > 0$ unless $W_N = 0$.

3.3. **Implementation of the Coordinate-Descent method.** Following the arguments above, coordinate descent on (1) can thus be achieved by the following recursive algorithm:

Algorithm 9 : $u^{k+1} \leftarrow \textbf{cdfft}(u^k, W, s, N)$ (slow version)

if $N = 1$ **then**
  return $\text{Prox}_{(W^{-1}\phi)}(s)$
else
  $n \leftarrow N/2$
  $\begin{pmatrix} u_e^k \\ u_o^k \end{pmatrix} \leftarrow \begin{pmatrix} \text{Ev}_n \\ \text{Od}_n \end{pmatrix} u^k, \quad \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \leftarrow s$
  $s_e \leftarrow (W_1 + W_2)^+ \left[W_1 s_1 + W_2 s_2 + (W_2 - W_1) D_n \mathcal{F}_n u_o^k\right]$
  $u_e^{k+1} \leftarrow \textbf{cdfft}(u_e^k, W_1 + W_2, s_e, n)$
  $s_o \leftarrow (W_1 + W_2)^+ D_n^* \left[W_1 s_1 - W_2 s_2 + (W_2 - W_1)\mathcal{F}_n u_e^{k+1}\right]$
  $u_o^{k+1} \leftarrow \textbf{cdfft}(u_o^k, W_1 + W_2, s_o, n)$
  return $\begin{pmatrix} \text{Ev}_n^T & \text{Od}_n^T \end{pmatrix} \begin{pmatrix} u_e^{k+1} \\ u_o^{k+1} \end{pmatrix}$
end if

This algorithm is slow, however, because it requires the computation of FFT's at each level (in the definitions of s_e and s_o). A close analysis of Algorithm 9 shows

that these FFT's can be eliminated by operating on the Fourier transform of u rather than on u itself. This trick is related to the speedup between Algorithm 4 and 5 for the BCD.

We first make the substitution $v = \mathcal{F}_N u$. Our goal is to rewrite Algorithm 9 in terms of the variable v so that all computations can be done in the Fourier domain. We define new variables $v_e = \mathcal{F}_n \mathrm{Ev}_n u$ and $v_o = \mathcal{F}_n \mathrm{Od}_n u$. Then by the Cooley-Tukey decomposition (31), we can transform between v and (v_e, v_o) with

$$v = \begin{pmatrix} I & D_n \\ I & -D_n \end{pmatrix} \begin{pmatrix} v_e \\ v_o \end{pmatrix}, \tag{62}$$

$$\begin{pmatrix} v_e \\ v_o \end{pmatrix} = \frac{1}{2} \begin{pmatrix} I & I \\ D_n^* & -D_n^* \end{pmatrix} v. \tag{63}$$

Using these identities, we can rewrite Algorithm 9 so that we only operate in the Fourier domain:

Algorithm 10 : $v^{k+1} \leftarrow \mathbf{cdfft}(v^k, W, s, N)$

1: **if** $N = 1$ **then**
2: **return** $\mathrm{Prox}_{(W^{-1}\phi)}(s)$
3: **else**
4: $n \leftarrow N/2$
5: $\begin{pmatrix} v_e^k \\ v_o^k \end{pmatrix} \leftarrow \frac{1}{2} \begin{pmatrix} I & I \\ D_n^* & -D_n^* \end{pmatrix} v^k, \quad \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = s$
6: $s_e \leftarrow (W_1 + W_2)^+ \left[W_1 s_1 + W_2 s_2 + (W_2 - W_1) D_n v_o^k \right]$
7: $v_e^{k+1} \leftarrow \mathbf{cdfft}(v_e^k, W_1 + W_2, s_e, n)$
8: $s_o \leftarrow (W_1 + W_2)^+ D_n^* \left[W_1 s_1 - W_2 s_2 + (W_2 - W_1) v_e^{k+1} \right]$
9: $v_o^{k+1} \leftarrow \mathbf{cdfft}(v_o^k, W_1 + W_2, s_o, n)$
10: **return** $\begin{pmatrix} I & D_n \\ I & -D_n \end{pmatrix} \begin{pmatrix} v_e^{k+1} \\ v_o^{k+1} \end{pmatrix}$
11: **end if**

The above algorithm requires no explicit FFT's, and runs in $O(N \log N)$ operations. Almost all of the computation takes place in lines 6 and 8, where we build the vectors s_e and s_o. Furthermore, those formulas individually represent only a small amount of computation. Each of s_e and s_o are formed using a simple linear combination of 3 vectors, and the coefficients of this linear combination may be precomputed only once.

4. Block Decomposition in a Hadamard Basis

In this section we consider block decompositions for problems of the form

$$\operatorname*{argmin}_{u \in \mathbb{C}^N} \sum_{j=0}^{N-1} \phi(u_j) + \frac{1}{2} \|R H_N u - s\|^2. \tag{64}$$

Note that the Sylvester decomposition for Hadamard matrices (35) is structurally almost identical to the Cooley Tukey decomposition for Fourier Matrices (31). For this reason, Hadamard matrices admit a block decomposition very similar to that for the Fourier basis. This decomposition is elaborated in the theorem below:

Theorem 3. *Let $W_1, W_2 \in \mathbb{R}^{n\times n}$ be nonnegative diagonal matrices, and let $s_1, s_2 \in \mathbb{C}$. Let*

$$W = \begin{pmatrix} W_1 & 0 \\ 0 & W_2 \end{pmatrix},$$

$$s = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix},\ u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

$$F(u) = \|H_{2n}u - s\|_W^2.$$

Then F can be rewritten in the following two equivalent forms:

$$F(u_1, u_2) = \|Hu_1 - \hat{s}_1||_{W_1+W_2}^2 + C_1, \tag{65}$$

$$F(u_1, u_2) = \|Hu_2 - \hat{s}_2||_{W_1+W_2}^2 + C_2, \tag{66}$$

where C_1 and C_2 are constants not depending on u_1 and u_1, respectively, and

$$\hat{s}_1 = (W_1 + W_2)^+ \left[W_1 s_1 + W_2 s_2 + (W_2 - W_1)H_n u_2\right], \tag{67}$$

$$\hat{s}_2 = (W_1 + W_2)^+ \left[W_1 s_1 - W_2 s_2 + (W_2 - W_1)H_n u_1\right]. \tag{68}$$

Proof. This proof proceeds similarly to that of theorem 2. We first note that for any $a, b \in \mathbb{C}^n$ we have

$$(W_1 + W_2)(W_1 + W_2)^+(W_1 a + W_2 b) = W_1 a + W_2 b. \tag{69}$$

Now, using the Sylvester decomposition, we can rewrite

$$F(u) = \|H_n u_1 + H_n u_2 - s_1\|_{W_1}^2 + \|H_n u_1 - H_n u_2 - s_2\|_{W_2}^2 \tag{70}$$

$$= \|H_n u_1\|_{W_1}^2 - 2\big\langle H_n u_1, W_1(s_1 - H_n u_2)\big\rangle \tag{71}$$

$$+ \|H_n u_1||_{W_2}^2 - 2\big\langle H_n u_1, W_2(s_2 + H_n u_2)\big\rangle + C_1' \tag{72}$$

$$= \|H_n u_1||_{W_1+W_2}^2 - 2\big\langle H_n u_1, (W_1 + W_2)\hat{s}_1\big\rangle + C_1', \tag{73}$$

which is equivalent to (65). Note that equation (73) follows from (69).

Similarly,

$$F(u) = \|H_n u_2 + (H_n u_1 - s_1)\|_{W_1}^2 + \|H_n u_2 - (H_n u_1 - s_2)\|_{W_2}^2 \tag{74}$$

$$= \|H_n u_2\|_{W_1+W_2} - 2\big\langle H_n u_2, W_1(s_1 - H_n u_1)\big\rangle \tag{75}$$

$$- 2\big\langle H_n u_2, W_2(-s_2 + H_n u_1)\big\rangle + C_2' \tag{76}$$

$$= ||H_n u_2||_{W_1+W_2} - 2\big\langle H_n u_2, (W_1 + W_2)\hat{s}_2\big\rangle + C_2', \tag{77}$$

which is equivalent to (66). □

Using this block decomposition, one may immediately apply the recursive approach of section 3.3 and arrive at an algorithm nearly identical to algorithm 9. However, as discussed above, this approach is slow because each recursive sub-step would require a call to the fast Hadamard transform. This problem can be solved by operating on the Hadamard transform of u, rather than u itself.

Following the approach of section 3.3, we make the substitution $v = H_n u$. We decompose v into its upper and lower halves

$$v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}.$$

The following trivial identities follow immediately from the Sylvester factorization (35):

$$\hat{v}_1 := H_n u_1 = \frac{1}{2}(v_1 + v_2)$$
$$\hat{v}_2 := H_n u_2 = \frac{1}{2}(v_1 - v_2)$$
$$v = \begin{pmatrix} \hat{v}_1 + \hat{v}_2 \\ \hat{v}_1 - \hat{v}_2 \end{pmatrix}$$

Using these identities, we can derive the Hadamard analog of algorithm 10:

Algorithm 11 : cdhadamard(v, W, s, N)

1: **if** $N = 1$ **then**
2: **return** $\text{Prox}_{(W^{-1}\phi)}(s)$
3: **end if**
4: $n = N/2$
5: $\hat{v}_1 = \frac{1}{2}(v_1 + v_2)$
6: $\hat{v}_2 = \frac{1}{2}(v_1 - v_2)$
7: $\hat{s}_1 = (W_1 + W_2)^+ [W_1 s_1 + W_2 s_2 + (W_2 - W_1)\hat{v}_2]$
8: Recursive call: $\hat{v}_1 = \textbf{cdhadamard}(\hat{v}_1, W_1 + W_2, \hat{s}_1, n)$
9: $\hat{s}_2 = (W_1 + W_2)^+ [W_1 s_1 - W_2 s_2 + (W_2 - W_1)\hat{v}_1]$
10: Recursive call: $\hat{v}_2 = \textbf{cdhadamard}(\hat{v}_2, W_1 + W_2, \hat{s}_2, n)$
11: $v = \begin{pmatrix} \hat{v}_1 + \hat{v}_2 \\ \hat{v}_1 - \hat{v}_2 \end{pmatrix}$
12: **return** v

The above algorithm requires no explicit Hadamard transforms, and runs in $O(N \log N)$ operations. Almost all of the computation takes place in steps 7 and 9, where we build the vectors $\hat{s}_1$ and $\hat{s}_2$, each of which is formed using a simple linear combination of 3 vectors, and the coefficients of this linear combination may be precomputed.

Note that algorithm 11 returns the vector $v = H_N u$, rather then the vector u itself. The above algorithm is iterated until convergence, and then the solution is recovered by computing

$$u^* = H_N^{-1} v^* = \frac{1}{N} H_N v^*.$$

5. Numerical Experiments

To demonstrate the performance of the CD algorithm, we compare it to FBS, FISTA, CGIST, and NESTA. We use two types of test problems: compressed sensing problems, and deconvolution-type problems.

The first set of problems are conventional compressed sensing problems. We wish to recover a K-sparse signal of length N from M measurements in an orthogonal basis. For each trial, a sparse signal of length N is generated by randomly choosing K non-zero elements of unit intensity. Signals are recovered using an ℓ_1-regularized problem of the form (14), with $\lambda = 5N/M$ for the Fourier basis, and $\lambda = 5/M$ for the Hadamard basis. Note the difference in scaling here is because the Fourier transform

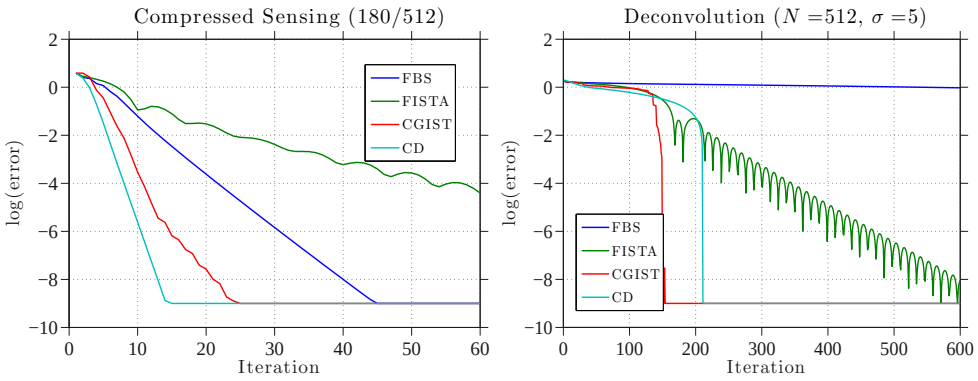

FIGURE 1. Sample convergence curves for (left) sparse reconstruction of a signal from 180 of 512 Fourier modes, and (right) deconvolution of a 5-sparse signal that has been blurred with a Gaussian of radius 5.

is unitary, while the Hadamard transform has norm $\sqrt{N}$. The measurements are contaminated with random Gaussian noise before signal recovery.

The second test problem we consider is a sparse deconvolution problem. In this case, the vector u^* represents the summation of K delta-function "sources" with unknown locations. The locations of these sources is chosen at random on each trial. The observed data, s, represents the summation of these delta functions after blurring with a Gaussian kernel. The goal of this problem is to 'reverse the heat equation,' and find the location of the unknown sources. The vector R is computed using the formula $R = \mathcal{F}_N(g_\sigma)$ where g represents a discrete Gaussian kernel with variance σ^2 pixels. When a Fourier basis is used, the matrix $R\mathcal{F}_N$ represents the blurring operator. This deconvolution problem is regularized with an ℓ_1 penalty to ensure sparse results. The regularization parameter was chosen to be $\lambda = 5/\sigma_{blur}$.

Efficiency results averaged over 100 trials are reported in Table 1 and 2. For each problem instance and algorithm, we report both runtime (in milliseconds) and the number of Fourier/Hadamard transforms needed. For purposes of counting fast transforms, each iteration of the CD method is counted as one transform. Each of the FBS, FISTA, CGIST, and CD methods were run until the current iterate satisfied the condition $\|u_k - u^*\|/K < 10^{-3}$.

Unlike the other algorithms, NESTA_UP solves a regularized version of (14), rather than the original non-differentiable problem. Consequently, we found that we could not identify a choice of parameters for NESTA_UP that reliably achieved the desired level of precision. For this reason, NESTA_UP was run with its default parameter settings, and the final value of the regularization parameter was 10^{-3}. With these parameter settings, NESTA_UP generated somewhat less accurate results than the other methods; the average 2-norm error of the final NESTA solution for the first Hadamard problem in table 2 was $\|u_k - u^*\| < 0.0248$. This code did not converge reliably for deconvolution problems, and so results for this algorithm are only recorded for compressed sensing.

TABLE 1. Fourier Compressed Sensing Results. For each algorithm we report runtime (ms) / transform count averaged over 100 random trials.

N	M	K	σ_{noise}	*FBS*	*FISTA*	*CGIST*	*NESTA*	*CD*
512	90	10	1e-3	8.58/90	17.3/171	6.07/36.9	49.2/305	0.745/7.9
512	180	10	1e-3	3.58/30.4	7.63/73.8	5.24/27.3	42.6/238	0.522/5.3
512	90	20	1e-3	24.1/270	34.1/361	15.3/99	68.2/440	1.85/22
512	180	20	1e-3	4.44/43.8	10.0/103	6.00/33.9	40.6/248	0.731/7.7
512	90	10	1e-2	9.91/107	18.3/183	6.61/40.5	53.3/337	0.845/8.43
512	180	10	1e-2	3.20/30.2	7.27/73.4	4.90/27.1	40.4/237	0.516/5.12
512	90	20	1e-2	24.2/267	33.5/353	14.2/95.7	66.0/432	1.84/21.7
512	180	20	1e-2	4.39/43.7	10.1/102	5.86/33.9	40.5/248	0.738/7.56
2048	110	10	1e-3	67.3/344	103/503	18.0/56.6	128.2/460	3.67/8.94
2048	220	10	1e-3	24.9/122	48.4/232	11.9/35.6	96.2/339	2.41/5.82
2048	110	20	1e-3	233/1236	219/1096	59.1/183	185/682	12.3/30.9
2048	220	20	1e-3	35.7/177	67.1/330	14.3/42.9	107/383	3.47/8.76
2048	110	10	1e-2	75.1/390	103/512	18.1/57.3	118/427	3.96/9.88
2048	220	10	1e-2	25.2/123	49.0/234	12.0/35.9	99.8/351	2.42/5.76
2048	110	20	1e-2	289/1548	263/1324	80.0/248	180/663	14.7/38.0
2048	220	20	1e-2	36.7/183	68.2/331	14.4/43.6	106/376	3.62/9.06

TABLE 2. Hadamard Compressed Sensing Results. For each algorithm we report runtime (ms) / transform count averaged over 100 random trials.

N	M	K	σ_{noise}	*FBS*	*FISTA*	*CGIST*	*NESTA*	*CD*
512	90	10	1e-3	9.15/150	13.3/209	7.08/51.2	47.3/371	0.471/12.4
512	180	10	1e-3	2.23/32.5	4.83/76.8	4.07/24.4	33.5/240	0.237/5.63
512	90	20	1e-3	75.4/1339	64.6/1076	23.3/169	74.3/594	3.51/107
512	180	20	1e-3	4.04/53.5	8.22/115	5.89/34.7	41.9/270	0.447/9.05
512	90	10	1e-2	8.78/126	10.3/144	7.14/47.1	56.5/375	0.503/10.7
512	180	10	1e-2	2.26/26	4.26/53.0	3.92/22.4	39.2/242	0.242/4.75
512	90	20	1e-2	64.8/1099	43.9/680	22.0/157	82.6/621	3.17/90.7
512	180	20	1e-2	3.53/44.5	6.46/84.5	5.35/32.3	43.4/272	0.397/7.85
2048	110	10	1e-3	70.3/656	67.4/600	15.4/80.6	101/584	2.41/16.5
2048	220	10	1e-3	15.4/137	28.5/240	8.76/40.7	64.1/357	1.08/6.6
2048	110	20	1e-3	850/8139	589/5284	84.3/374	136/794	27.5/198
2048	220	20	1e-3	29.7/263	44.3/380	12.3/58.9	79.0/441	1.94/12.8
2048	110	10	1e-2	62.1/538	48.1/391	15.5/75.7	108/575	2.31/14.3
2048	220	10	1e-2	13.2/108	21.1/166	8.89/39.2	72.3/361	1.06/5.59
2048	110	20	1e-2	698/6445	343/2951	73.0/327	140/791	23.0/160
2048	220	20	1e-2	24.9/220	31.8/271	11.3/55.8	79.5/443	1.72/11.3

From the results in Tables 1 and 2, it is clear that the primary advantage of the coordinate descent (CD) method is speed. For the compressed sensing problems, it was observed that the CD method was approximately one order of magnitude faster than the second fastest method. For small, well-conditioned problems both

TABLE 3. Fourier Deconvolution Results. For each algorithm we report runtime (ms) / transform count averaged over 100 random trials.

N	σ_{blur}	*FBS*	*FISTA*	*CGIST*	*CD*
128	2	33.9/714	8.81/137	13.6/129	2.21/127
128	5	1863/39902	117/1990	148/1381	85/5173
128	10	39114/717164	1443/24945	748/7205	1108/66207
128	20	51843/999998	21250/388882	1406/14783	9270/584209
512	2	22.6/307	8.23/90.5	12.3/86.8	3.86/48.5
512	5	527/6777	49.2/523	96.3/611	55.5/644
512	10	15738/201612	410/4925	397/2720	1718/22328
512	20	62425/803883	3617/43781	1226/8626	7091/90727
2048	2	35.6/193	17.3/82.1	25.8/82.4	10.5/28.1
2048	5	1144/6521	92/462	160/511	296/808
2048	10	10724/62242	298/1517	482/1512	1980/5428
2048	20	71404/417653	1236/6387	1197/3768	6781/18827

CGIST and FBS were competitive for second fastest method. However, for less well conditioned problems (i.e. problems with relatively small M) or larger test problems CGIST was reliably the second fastest method. Interestingly, for the compressed problems, FISTA is not always faster than FBS. The advantages of FISTA seem to become more apparent on larger problems, and problems with low M/N ratio.

Another significant advantage of the CD method is that, unlike the FBS algorithm, it does not require the user to choose a time step and it has no stability restriction.

One notable disadvantage of the CD scheme is that its implementation is relatively complex when compared to FBS or FISTA. The FBS-based methods are extremely easy to implement in many languages (for example, Matlab, C or C++) using well-optimized implementations of the fast Fourier transform such as FFTW. In contrast, our CD methods cannot directly use such components. In order to compete with such efficient codes, the decomposition schemes must be implemented in a language such as C or C++ which enables fast low-level array access.

6. Conclusion

We introduce a multi-level decomposition scheme for solving basis pursuit problems in a Fourier and Hadamard basis. The iterates generated by this scheme are equivalent to the coordinate-descent (CD) method for basis pursuit. Using four different test problems, the CD algorithm is compared to 4 other common algorithms for sparse signal recovery. For most of the problems considered here, runtimes for the CD algorithm are faster than conventional methods.

References

[1] Jonathan Barzilai and Jonathan M. Borwein. Two-point step size gradient methods. *IMA J Numer Anal*, 8(1):141–148, January 1988.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Sciences*, 2:183–202, 2009.

[3] Stephen Becker, Jerome Bobin, and Emmanuel Candes. Nesta: A fast and accurate first-order method for sparse recovery. *Technical Report of Stanford University*, Apr 2009.
[4] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, September 1999.
[5] Leo Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384, 1995.
[6] A.M. Bruckstein, M. Elad, and M. Zibulevsky. On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. *Information Theory, IEEE Transactions on*, 54:4813–4820, 2008.
[7] E. J. Candes, J. Romberg, and T.Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52:489 – 509, 2006.
[8] E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.
[9] Emmanuel J. Candes and Justin Romberg. Quantitative robust uncertainty principles and optimally sparse decompositions. *Found. Comput. Math.*, 6(2):227–254, 2006.
[10] George H.-G. Chen and R. T. Rockafellar. Convergence rates in forward–backward splitting. *SIAM J. on Optimization*, 7(2):421–444, 1997.
[11] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1998.
[12] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.
[13] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.
[14] D Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52:1289–1306, 2006.
[15] D.L. Donoho and J Tanner. Sparse nonnegative solutions of underdetermined linear equations by linear programming. *Proceedings of the National Academy of Sciences*, 102(27):9446–9451, 2005.
[16] P. Duhamel and M. Vetterli. Fast fourier transforms: a tutorial review and a state of the art. *Signal Process.*, 19(4):259–299, April 1990.
[17] Willard S. Ellis, Susan J. Eisenberg, David M. Auslander, Michael W. Dae, Avideh Zakhor, and Michael D. Lesh. Deconvolution: A novel signal processing approach for determining activation time from fractionated electrograms and detecting infarcted tissue. *Circulation*, 94:2633–2640, 1996.
[18] Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *Technical report of Dept. of Statistics, Stanford University*, Dec 2007.
[19] W. Fu. Penalized regressions: The bridge vs the lasso. *JCGS*, 7:397–416, 1998.
[20] N.P. Galatsanos, A.K. Katsaggelos, R.T. Chin, and A.D. Hillery. Least squares restoration of multichannel images. *IEEE Transactions on Signal Processing*, 39:2222–2236, 1991.
[21] Tom Goldstein and Simon Setzer. High-order methods for basis pursuit. *UCLA CAM technical report, 10-41*, 2010.
[22] R. Griesse and D.A. Lorenz. Convergence of a block coordinate descent method for nondifferentiable minimization. *Inverse Problems*, 24, 2008.
[23] Elaine Hale, Wotao Yin, and Yin Zhang. A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. *CAAM Technical Report*, TR07, 2007.
[24] K. J. Horadam. *Hadamard Matrices and Their Applications*. Princeton University Press, Princeton, NJ, 2007.
[25] Sami Kirolos, Jason Laska, Michael Wakin, Marco Duarte, Dror Baron, Tamer Ragheb, Yehia Massoud, and Richard Baraniuk. Analog-to-information conversion via random demodulation. In *IEEE Dallas/CAS Workshop on Design, Applications, Integration and Software*, 2006.
[26] Yinbo Li and Gonzalo R. Arce. A maximum likelihood approach to least absolute deviation regression. *EURASIP J. Appl. Signal Process.*, 2004:1762–1769, 2004.
[27] Yingying Li and Stanley Osher. Coordinate descent optimization for ℓ^1 minimization with application to compressed sensing; a greedy algorithm. *Inverse Problems and Imaging*, 3(3):487–503, August 2009.

[28] P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.
[29] J.M. Mendel and C.S. Currus. *Maximum-likelihood deconvolution: a journey into model-based signal processing*. Springer-Verlag, 1990.
[30] Yu. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005. 10.1007/s10107-004-0552-5.
[31] Yurri Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Math. Dokl.*, 27:372–376, 1983.
[32] Muhammad Aslam Noor. Splitting algorithms for general pseudomonotone mixed variational inequalities. *J. of Global Optimization*, 18(1):75–89, 2000.
[33] M.S. O'Brien, A.N. Sinclair, and S.M. Kramer. Recovery of a sparse spike time series by l1 norm deconvolution. *IEEE Transactions on Signal Processing*, 42:3353–3365, 1994.
[34] T. Olofsson and E. Wennerstrom. Sparse deconvolution of b-scan images. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 54:1634–1641, 2007.
[35] Tomas Olofsson. Semi-sparse deconvolution robust to uncertainties in the impulse responses. *Ultrasonics*, 37:423–432, 1999.
[36] Art B. Owen. A robust hybrid of lasso and ridge regression. Technical report, Stanford University, 2006.
[37] B. G Passty. Ergodic convergence to a zero of the sum of monotone operators in hilbert space. *Journal of Mathematical Analysis and Applications*, 72:386–390, 1979.
[38] K. R. Rao and P. Yip. *Discrete cosine transform: algorithms, advantages, applications*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.
[39] Mark Rudelson and Roman Vershynin. Geometric approach to error correcting codes and reconstruction of signals. *Int. Math. Res. Not*, 64:4019–4041, 2005.
[40] S. Sardy, A.G. Bruce, and P. Tseng. Block coordinate relaxation methods for nonparametric wavelet denoising. *Journal of Computational and Graphical Statistics*, 9(2):361–379, 2000.
[41] Alan S. Stern, David L. Donoho, and Jeffrey C. Hoch. Nmr data processing using iterative thresholding and minimum l1-norm reconstruction. *Journal of Magnetic Resonance*, 188(2):295 – 300, 2007.
[42] Petre Stoica and Randolph L. Moses. *Introduction to Spectral Analysis*. Prentice Hall, Upper Saddle River, NJ, 1997.
[43] Oloffson T. and Stepinksi T. Maximum a posteriori deconvolution of sparse ultrasonic signals using genetic optimization. *Ultrasonics*, 37:423–432, 1999.
[44] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.
[45] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, 67(1):91–108, 2005.
[46] Robert Tibshirani and Pei Wang. Spatial smoothing and hot spot detection for cgh data using the fused lasso. *Biostatistics (Oxford, England)*, 9(1):18–29, January 2008.
[47] J.A. Tropp. Just relax: convex programming methods for identifying sparse signals in noise. *Information Theory, IEEE Transactions on*, 52:1030–1051, 2006.
[48] J.A. Tropp, M.B. Wakin, M.F. Duarte, D. Baron, and R.G. Baraniuk. Random filters for compressive sampling and reconstruction. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 3, 2006.
[49] P. Tseng. Prediction accuracy and stability of regression with optimal scaling transformations. *Technical report LIDS-P;1840, Massachusetts Institute of Technology. Laboratory for Information and Decision Systems*, 1988.
[50] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494, 2001.
[51] Paul Tseng. A modified forward-backward splitting method for maximal monotone mappings. *SIAM J. Control Optim*, 38:431–446, 1998.
[52] A. Van der Kooij. Prediction accuracy and stability of regression with optimal scaling transformations. *Technical report, Dept. of Data Theory, Leiden University*, 2007.
[53] Zaiwen Wen, Wotao Yin, Donald Goldfarb, and Yin Zheng. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation. Technical report, Rice University, 2009.

[54] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.
[55] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67:301–320, 2005.