

Efficient Implementations of Orthogonal Matching Pursuit Based on Inverse Cholesky Factorization

Hufei Zhu, Ganghua Yang

Communications Technology Laboratory

Huawei Technologies Co., Ltd., P. R. China

Email: zhuhufei@huawei.com, yang.yangganghua@huawei.com

Wen Chen

the Department of Electronic Engineering

Shanghai Jiao Tong University, Shanghai, P. R. China

Email: wenchen@sjtu.edu.cn

Abstract—Based on the recently proposed efficient inverse Cholesky factorization, we propose three efficient implementations of Orthogonal Matching Pursuit (OMP), and compare them with the existing implementations of OMP by theoretical and empirical analysis. The proposed implementation 1 theoretically needs the least computational complexity, and is the fastest in the simulations for almost all problem sizes. Among the implementations that store the Gram matrix of the dictionary, the proposed implementation 1 needs the least memories in the k -th iteration ($k > 1$). As the memory-saving variants of the proposed implementation 1, the proposed implementations 2 and 3 save the memories for the Gram matrix at the expense of higher computational complexity, and in the simulations they are faster than the existing implementations for most problem sizes. With respect to the existing efficient implementation that does not store the Gram matrix, the proposed implementation 2 needs less computational complexity and a little more memories, while the proposed implementation 3 needs the same complexity and less memories.

I. INTRODUCTION

Let $\mathbf{z} \in \mathbb{C}^N$ be a K -sparse signal vector with no more than K non-zero entries. Compressive sensing (CS) theory [1] asserts that \mathbf{z} can be exactly recovered from a measurement vector $\mathbf{y} \in \mathbb{C}^M$ with $M \ll N$. The measurement vector

$$\mathbf{y} = \mathbf{T}\mathbf{z} \quad (1)$$

where \mathbf{T} is an $M \times N$ dictionary matrix. Since $M \ll N$, it is generally an undetermined problem to recover \mathbf{z} from \mathbf{y} and \mathbf{T} in (1). However, we can exactly recover the $k \leq K$ non-zero entries of \mathbf{z} by CS signal recovery algorithms, which include Basis pursuit (BP) [2], matching pursuit (MP) [3], and Bayesian framework [4].

Orthogonal Matching Pursuit (OMP) [5]–[13] is a popular MP algorithm that possesses the merits of low complexity and good recovery quality [6], [14]. For OMP, some computationally efficient implementations have been proposed [5]–[13], and have been compared in [15]. Based on the recently proposed efficient inverse Cholesky factorization [16], in this paper we develop computationally efficient implementations of OMP, which are then compared with the existing efficient implementations of OMP. To focus on efficient implementations of OMP, we do not consider extensions, variants, or approximate implementations of OMP, such as Stagewise OMP (StOMP) [17], Backtracking-based Adaptive OMP (BAOMP) [18], gradient pursuit [19], or cyclic matching pursuit [20].

In this paper, $(\bullet)^T$, $(\bullet)^*$, $(\bullet)^H$ and $(\bullet)^{-1}$ denote matrix transposition, matrix conjugate, matrix conjugate transposition, and matrix inverse, respectively. $\mathbf{0}_k$ is the zero column vector with k elements.

II. EXISTING IMPLEMENTATIONS OF OMP

The dictionary \mathbf{T} in (1) can be written as

$$\mathbf{T} = (\mathbf{t}_{:1} \quad \mathbf{t}_{:2} \quad \cdots \quad \mathbf{t}_{:N}), \quad (2)$$

where $\mathbf{t}_{:i} \in \mathbb{C}^M$ denotes the i -th column of \mathbf{T} , and is called as an atom. The signal recovery problem is to find the sparse solution to the underdetermined linear system (1), i.e., to represent \mathbf{y} as a weighted sum of the least number of atoms. OMP is a greedy algorithm that selects atoms from \mathbf{T} iteratively to approximate \mathbf{y} gradually. In each iteration, OMP selects the atom best matching the current residual (i.e. the approximation error), renews the weights for all the already selected atoms (to obtain the least squares approximation of \mathbf{y}), and then updates the residual accordingly.

In what follows, we summarize the naive implementation of OMP [7], [15]. As the OMP implementation described in [7, left lines 27–34 on page 2], we also follow these conventions: Γ^k is a set containing the indices of the selected k atoms, and \mathbf{T}_{Γ^k} is a sub-matrix of \mathbf{T} containing only those columns of \mathbf{T} with indices in Γ^k .

The Naive Implementation of OMP

1. Initialise $\Gamma^0 = \emptyset$, the residual $\mathbf{r}^0 = \mathbf{y}$, and the iteration counter $k = 1$.

2. Projection:

$$\nabla^{k-1} = \mathbf{T}^H \mathbf{r}^{k-1}. \quad (3)$$

3. Decide $i^k = \operatorname{argmax}_{i=1,2,\dots,N} (|\nabla_i^{k-1}| / \|\mathbf{t}_{:i}\|_2)$, where ∇_i^{k-1} is the i -th entry of ∇^{k-1} , and $\|\cdot\|_2$ denotes the l_2 -norm of a vector. Then let $\Gamma^k = \Gamma^{k-1} \cup i^k$, i.e.,

$$\gamma_k = i^k, \quad (4)$$

where γ_k is the k -th entry of the set Γ .

4. Renew the weights for the selected k atoms by

$$\mathbf{s}_k = (\mathbf{T}_{\Gamma^k}^H \mathbf{T}_{\Gamma^k})^{-1} \mathbf{T}_{\Gamma^k}^H \mathbf{y}. \quad (5)$$

5. Update the residual by

$$\mathbf{r}^k = \mathbf{y} - \mathbf{T}_{\Gamma^k} \mathbf{s}_k. \quad (6)$$

6. $k := k + 1$. Then return to Step 2 if $k < K$ and the residual energy [15]

$$|\mathbf{r}^k|^2 > \xi |\mathbf{y}|^2, \quad (7)$$

where ξ is a small positive real number, e.g., $\xi = 10^{-10}$.

7. Output: $\mathbf{r}^{K'}$, $\mathbf{\Gamma}^{K'}$ and $\mathbf{s}_{K'}$, where $K' \leq K$.

The computational complexity of OMP can be reduced by utilizing the Matrix Inversion Lemma [5]–[7], the QR Factorization [7]–[9], or the Cholesky Factorization [10]–[12], [21]. Recently in [15], the authors summarized the existing various implementations of OMP, studied their numerical and computational performances, and empirically compared their computational times. According to the conclusions in [15], the accumulation of error is insignificant in each existing OMP implementation. On the other hand, any of the existing implementations compared in [15] can be the fastest for some specific problem sizes, while the OMP implementation by the QR Factorization [9] appears to be the fastest for particularly large problem sizes.

III. PROPOSED IMPLEMENTATIONS OF OMP

In this section, we develop computational efficient implementations of OMP, which are based on the efficient inverse Cholesky factorization recently proposed in [16].

The Gram matrix [7] of all the N atoms $\mathbf{t}_{:,1}, \dots, \mathbf{t}_{:,N}$ is

$$\mathbf{G} = \mathbf{T}^H \mathbf{T}, \quad (8)$$

and the Gram matrix of the k atoms selected in iteration k is

$$\mathbf{G}_{\Gamma^k} = \mathbf{T}_{\Gamma^k}^H \mathbf{T}_{\Gamma^k}, \quad (9)$$

where \mathbf{T}_{Γ^k} can be written as

$$\mathbf{T}_{\Gamma^k} = [\mathbf{T}_{\Gamma^{k-1}} \quad \mathbf{t}_{:\gamma_k}]. \quad (10)$$

The inverse Cholesky factor [16] of \mathbf{G}_{Γ^k} is \mathbf{F}_k that satisfies

$$\mathbf{F}_k \mathbf{F}_k^H = \mathbf{G}_{\Gamma^k}^{-1} = (\mathbf{T}_{\Gamma^k}^H \mathbf{T}_{\Gamma^k})^{-1}. \quad (11)$$

Substitute (11) into (5) to obtain

$$\mathbf{s}_k = \mathbf{F}_k \mathbf{F}_k^H \mathbf{T}_{\Gamma^k}^H \mathbf{y}. \quad (12)$$

Substitute (12) into (6), and substitute (6) into (3) to obtain

$$\nabla^{k-1} = \mathbf{T}^H \mathbf{y} - \mathbf{T}^H \mathbf{T}_{\Gamma^{k-1}} \mathbf{F}_{k-1} \mathbf{F}_{k-1}^H \mathbf{T}_{\Gamma^{k-1}}^H \mathbf{y}. \quad (13)$$

Let us define

$$\mathbf{E}_k = \mathbf{T}_{\Gamma^k} \mathbf{F}_k, \quad (14)$$

and

$$\left\{ \begin{array}{l} \mathbf{D}_k = \mathbf{T}^H \mathbf{T}_{\Gamma^k} \mathbf{F}_k = \mathbf{T}^H \mathbf{E}_k, \\ \mathbf{a}_k = \mathbf{F}_k^H \mathbf{T}_{\Gamma^k}^H \mathbf{y} = \mathbf{E}_k^H \mathbf{y}. \end{array} \right. \quad (15a)$$

$$(15b)$$

Then we can write (13) as

$$\nabla^{k-1} = \mathbf{T}^H \mathbf{y} - \mathbf{D}_{k-1} \mathbf{a}_{k-1}. \quad (16)$$

In the next section we will show that actually

$$\mathbf{F}_k = \mathbf{L}_k^{-H}, \quad (17)$$

where \mathbf{L}_k is the unique lower triangular Cholesky factor [24] of \mathbf{G}_{Γ^k} , which satisfies

$$\mathbf{L}_k \mathbf{L}_k^H = \mathbf{G}_{\Gamma^k}. \quad (18)$$

On the other hand, let us introduce the efficient algorithm proposed in [16] that computes \mathbf{F}_k from \mathbf{F}_{k-1} iteratively. Substitute (10) into (9) to obtain

$$\mathbf{G}_{\Gamma^k} = \begin{bmatrix} \mathbf{T}_{\Gamma^{k-1}}^H \mathbf{T}_{\Gamma^{k-1}} & \mathbf{T}_{\Gamma^{k-1}}^H \mathbf{t}_{:\gamma_k} \\ \mathbf{t}_{:\gamma_k}^H \mathbf{t}_{:\gamma_k} & \mathbf{t}_{:\gamma_k}^H \mathbf{t}_{:\gamma_k} \end{bmatrix}, \quad (19)$$

which can be written as

$$\mathbf{G}_{\Gamma^k} = \begin{bmatrix} \mathbf{G}_{\Gamma^{k-1}} & \mathbf{v}_{k-1} \\ \mathbf{v}_{k-1}^H & g_{\gamma_k, \gamma_k} \end{bmatrix}, \quad (20)$$

where

$$\mathbf{v}_{k-1} = \mathbf{T}_{\Gamma^{k-1}}^H \mathbf{t}_{:\gamma_k}, \quad (21)$$

and $g_{\gamma_k, \gamma_k} = \mathbf{t}_{:\gamma_k}^H \mathbf{t}_{:\gamma_k}$ is the entry in the γ_k -th row and column of \mathbf{G} . In [16], \mathbf{F}_k is computed by

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{F}_{k-1} & \mathbf{u}_{k-1} \\ \mathbf{0}_{k-1}^T & \lambda_k \end{bmatrix}, \quad (22)$$

where λ_k and \mathbf{u}_{k-1} can be computed by [16, equation (24)]

$$\left\{ \begin{array}{l} \lambda_k = 1/\sqrt{g_{\gamma_k, \gamma_k} - \mathbf{w}_{k-1}^H \mathbf{w}_{k-1}}, \\ \mathbf{u}_{k-1} = -\lambda_k \mathbf{F}_{k-1} \mathbf{w}_{k-1}, \end{array} \right. \quad (23a)$$

$$(23b)$$

and

$$\mathbf{w}_{k-1} = \mathbf{F}_{k-1}^H \mathbf{v}_{k-1}. \quad (24)$$

In what follows, we utilize the above-described \mathbf{F}_k , \mathbf{D}_k and \mathbf{a}_k to develop an efficient implementation of OMP, where step 3 is identical to step 3 in the naive implementation of OMP (described in Section II). The following equations (25)–(29) are verified in [22].

The Proposed Implementation 1 of OMP

1. Initialise $\mathbf{\Gamma}^0 = \emptyset$, the residual energy $|\mathbf{r}^0|^2 = \mathbf{y}^H \mathbf{y}$, and the iteration counter $k = 1$.

2. Projection:

if $k = 1$, compute ∇^0 by (3); else compute ∇^{k-1} by

$$\nabla^{k-1} = \nabla^{k-2} - \mathbf{d}_{:(k-1)} \cdot a_{k-1}, \quad (25)$$

where $\mathbf{d}_{:(k-1)}$ is the $(k-1)$ -th column of \mathbf{D}_{k-1} , and a_{k-1} is the $(k-1)$ -th entry of \mathbf{a}_{k-1} .

3. Decide $i^k = \text{argmax}_{i=1,2,\dots,N} (|\nabla_i^{k-1}| / \|\mathbf{t}_{:,i}\|_2)$. Then let $\mathbf{\Gamma}^k = \mathbf{\Gamma}^{k-1} \cup i^k$.

4. Obtain \mathbf{w}_{k-1} from $\mathbf{d}_{i^k,:}^H$ (i.e. the i^k -th row of \mathbf{D}_{k-1}) by

$$\mathbf{w}_{k-1} = (\mathbf{d}_{i^k,:}^H)^H, \quad (26)$$

compute λ_k from \mathbf{w}_{k-1} by (23a), compute

$$\left\{ \begin{array}{l} a_k = \lambda_k \nabla_{i^k}^{k-1}, \\ \mathbf{a}_k = [\mathbf{a}_{k-1}^T \quad a_k]^T, \end{array} \right. \quad (27a)$$

$$(27b)$$

and

$$\begin{cases} \mathbf{d}_{:k} = \lambda_k (\mathbf{g}_{:i^k} - \mathbf{D}_{k-1} \mathbf{w}_{k-1}), \\ \mathbf{D}_k = [\mathbf{D}_{k-1} \quad \mathbf{d}_{:k}], \end{cases} \quad (28a)$$

$$(28b)$$

where $\nabla_{i^k}^{k-1}$ denotes the i^k -th entry of ∇^{k-1} , and $\mathbf{g}_{:i^k}$ is the i^k -th column of \mathbf{G} . When $k = 1$, $\mathbf{D}_0 = \mathbf{a}_0 = \emptyset$ is assumed in (26), (27) and (28).

5. Update the residual energy by

$$|\mathbf{r}^k|^2 = |\mathbf{r}^{k-1}|^2 - |a_k|^2. \quad (29)$$

6. $k := k+1$. Return to Step 2 if $k < K$ and $|\mathbf{r}^k|^2 > \xi |\mathbf{y}|^2$.

7. Output: $|\mathbf{r}^{K'}|^2$, $\Gamma^{K'}$ and

$$\mathbf{s}_{K'} = \mathbf{F}_{K'} \mathbf{a}_{K'}, \quad (30)$$

where $K' \leq K$. When required, the residual $\mathbf{r}^{K'}$ can be computed by (6). Notice that (30) can be obtained by substituting (15b) into (12), and $\mathbf{F}_{K'}$ in (30) can be computed by (23) and (22), or by (18) and (17).

In the k -th iteration, the proposed implementation 1 of OMP requires $N^2 + Nk + k$ memories (to store \mathbf{G} , \mathbf{D}_k and \mathbf{a}_k). The proposed implementation 1 has two memory-saving variants that both can save the N^2 memories to store \mathbf{G} , at the expense of higher computational complexity. These two variants are called as the proposed implementations 2 and 3, respectively.

The proposed implementation 2 of OMP only needs to modify step 4 of the proposed implementation 1 into step 4', as follows, where (31) and (32) are verified in [22].

4'. Obtain \mathbf{w}_{k-1} in \mathbf{D}_{k-1} by (26), compute λ_k from \mathbf{w}_{k-1} by (23a), compute a_k by (27a) to obtain \mathbf{a}_k by (27b), compute \mathbf{E}_k iteratively by

$$\begin{cases} \mathbf{e}_{:k} = \lambda_k (\mathbf{t}_{:\gamma_k} - \mathbf{E}_{k-1} \mathbf{w}_{k-1}), \\ \mathbf{E}_k = [\mathbf{E}_{k-1} \quad \mathbf{e}_{:k}], \end{cases} \quad (31a)$$

$$(31b)$$

and compute $\mathbf{d}_{:k}$ by

$$\mathbf{d}_{:k} = \mathbf{T}^H \mathbf{e}_{:k} \quad (32)$$

to obtain \mathbf{D}_k by (28b). Here we compute $\mathbf{d}_{:k}$ by (32) instead of (28a) to avoid using \mathbf{G} . When $k = 1$, $\mathbf{E}_0 = \mathbf{D}_0 = \mathbf{a}_0 = \emptyset$ is assumed in (26), (27), (31) and (28b).

The proposed implementation 3 of OMP also saves the Nk memories to store \mathbf{D}_k . It needs to modify the steps 2 and 4 of the proposed implementation 1 into the steps 2'' and 4'', respectively, as follows, where (33) and (34) are verified in [22].

2''. Projection:

if $k = 1$, compute ∇^0 by (3); else compute ∇^{k-1} by

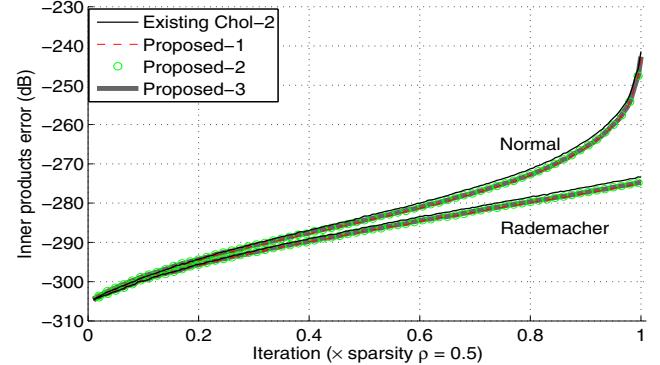
$$\nabla^{k-1} = \nabla^{k-2} - \mathbf{T}^H \cdot (\mathbf{e}_{:(k-1)} a_{k-1}). \quad (33)$$

4''. Compute \mathbf{w}_{k-1} from \mathbf{E}_{k-1} by

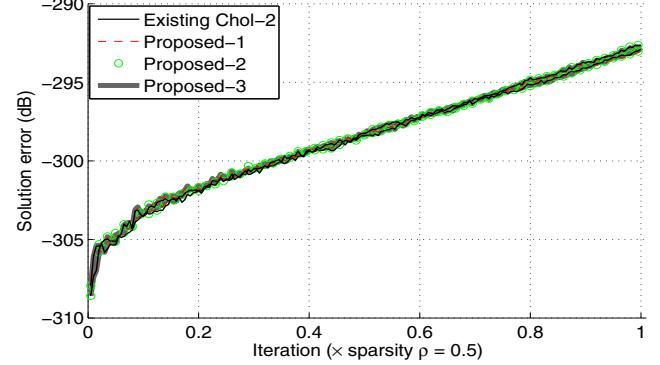
$$\mathbf{w}_{k-1} = \mathbf{E}_{k-1}^H \mathbf{t}_{:i^k}, \quad (34)$$

TABLE I
COMPARISON OF COMPLEXITIES AND MEMORY REQUIREMENTS AMONG
OMP IMPLEMENTATIONS

Algorithm	Complexity	Memory
Naive	$NM + 2Mk + 2Mk^2 + O(k^3)$	NM
Chol-1	$NM + Mk + \frac{3}{2}k^2$	$N^2 + NM + k^2$
Chol-2	$Nk + \frac{3}{2}k^2$	$N^2 + NM + k^2$
QR-1	$NM + 2Mk$	$NM + Mk + k^2$
QR-2	$Nk + 2Mk + k^2$	$N^2 + NM + Mk + k^2$
MIL	$Nk + 3Mk$	$N^2 + NM + 3Mk$
Proposed-1	Nk	$N^2 + Nk$
Proposed-2	$NM + Mk$	$NM + Nk + Mk$
Proposed-3	$NM + 2Mk$	$NM + Mk$



(a) Mean error in inner products



(b) Mean error in solution

Fig. 1. Numerical Errors of Chol-2 and the proposed three implementations with respect to the naive one, as a function of algorithm iteration.

compute λ_k from \mathbf{w}_{k-1} by (23a), compute a_k by (27a) to obtain \mathbf{a}_k by (27b), and compute $\mathbf{e}_{:k}$ by (31a) to obtain \mathbf{E}_k by (31b). When $k = 1$, $\mathbf{E}_0 = \mathbf{a}_0 = \emptyset$ is assumed in (34), (27) and (31).

IV. THEORETICAL AND EMPIRICAL ANALYSIS

We compare the proposed OMP implementations with the existing ones by theoretical and empirical analysis.

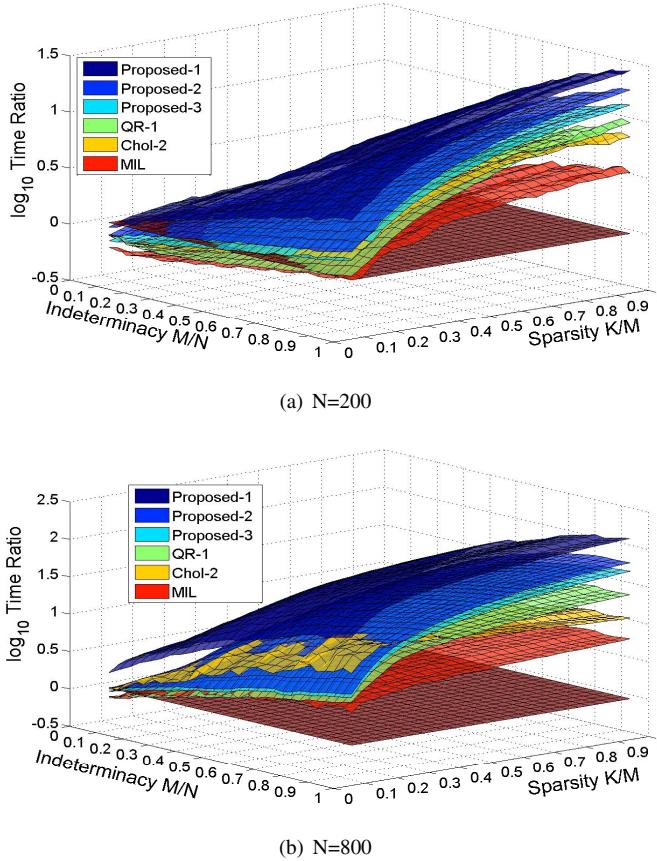


Fig. 2. Computation times of the OMP implementations as a function of problem sizes for two different dimensions N .

A. Theoretical Analysis of Computational Complexities and Memory Requirements

Table I compares the expected computational complexities and memory requirements of the existing and proposed OMP implementations in the k -th iteration ($k > 1$), where the entries for the existing OMP implementations are deduced from Table I and the shared simulation code [23] of [15]. As in [15], Naive, MIL, Chol-1, Chol-2, QR-1 and QR-2 in Table I denote the naive OMP implementation [5], [15], the OMP implementation by the Matrix Inversion Lemma [5], two OMP implementations by the Cholesky Factorization [10]–[12], and two OMP implementations by the QR Factorization [9], respectively. On the other hand, Proposed-1, Proposed-2 and Proposed-3 in Table I denote the proposed implementations 1, 2 and 3, respectively. To evaluate the computational complexities, we list the numbers of multiplications and additions¹, which include both dominating terms and coefficients. When evaluating the memory requirements, we neglect the fact that we may only store about half elements of the Hermitian Gram matrix \mathbf{G} , the triangular \mathbf{R} factor (of the QR factorization), and the triangular Cholesky factor. In Table I, we do not include the terms that are $O(N)$, $O(M)$ or $O(k)$ for simplicity.

¹It can be seen that each OMP implementation needs nearly the same number of multiplications and additions

Table I shows that Proposed-1 needs the least computational complexity, and needs less memories than the existing implementations (i.e., Chol-1, Chol-2, QR-2 and MIL) that spend N^2 memories to store \mathbf{G} . On the other hand, with respect to the existing efficient implementation (i.e. QR-1) that does not spend N^2 memories to store \mathbf{G} , Proposed-2 needs less computational complexity and a little more memories, while Proposed-3 needs the same complexity and less memories.

B. Empirical Comparison of Numerical Performance and Computational Time

We compare the proposed implementations with the existing ones, by MATLAB simulations² on a 64-bit 2.4-GHz Microsoft-Windows Xeon workstation with 15.9 GB of RAM.

Firstly we utilize the shared simulation code [23] of [15] that gives the numerical performance and computational times of four existing implementations (i.e., Naive, QR-1, Chol-2 and MIL). As in [15], \mathbf{T} is sampled from the uniform spherical ensemble, while sparse vectors are sampled from a Rademacher or Normal distribution. Moreover, usually the simulation parameters are the same as those in [15].

As Fig. 1 in [15], Fig. 1 here shows the accumulation of numerical errors in the recursive computation of the inner products and solutions, compared to the values given by the naive implementation. We perform 50 independent trials with $N = M = 400$ and $K = 200$. For each iteration, we compute the mean relative errors of the inner products and those of the solutions, which are plotted in Fig. 1(a) and Fig. 1(b), respectively. Since Fig. 1 in [15] shows that QR-1, Chol-2 and MIL have nearly the same numerical errors, here we only compare the numerical errors of the proposed implementations with those of Chol-2. From Fig. 1, it can be seen that the differences between implementations are inconsequential.

Similar to Fig. 2 in [15], Fig. 2 here shows the computation times of 100 independent runs of K iterations (sparsity) for \mathbf{T} of size $M \times N$. We measure time by tic/toc in MATLAB. To be fair in the comparisons, the MATLAB code for the proposed implementations is similar to the shared simulation code of [15], and has counted the times to compute $\mathbf{F}_{K'}$ in (30) by (18) and (17). Fig. 2 shows $\log_{10}(T_{imp}(K, M, N)/T_{naive}(K, M, N))$ for two different N . According to Fig. 2, Proposed-1 is the fastest for almost all N , M and K , while Proposed-2 and Proposed-3 are faster than the existing implementations for most N , M and K .

When considering the computation time, notice that the proposed three implementations do not need any back-substitution in each iteration, and only include parallelizable matrix-vector products [25]. Contrarily, the existing Chol-1, Chol-2 and QR-2 usually solve triangular systems by back-substitutions [24], which are inherent serial processes unsuitable for parallel implementations [25]. In detail, Chol-1 and Chol-2 both need $\frac{3}{2}k^2$ multiplications and additions for the back-substitutions to solve $3k \times k$ triangular systems [24] (i.e., equations (12), (13)

²The MATLAB code to generate Fig. 1, Fig. 2 and Table 2 of this paper is shared in <http://omp2vtc2013.blogspot.com/2013/05/omp.html>. If this site is not available, please visit <http://blog.sina.com.cn/u/3266279821>.

TABLE II
TO COMPARE COMPUTATION TIMES (IN MILLISECONDS) OF OMP IMPLEMENTATIONS, CFAR AND CFDR, WHERE CFAR AND CFDR ARE STOMP EQUIPPED WITH THRESHOLDING CFAR (CONSTANT FALSE ALARM RATE) AND CFDR (CONSTANT FALSE DISCOVERY RATE), RESPECTIVELY.

Problem Suite (K,M,N)	OMP in [17]	CFAR	CFDR	Proposed-1	Proposed-2	Proposed-3
(10,100,1000)	0.04	0.01	0.08	0.02	0.02	0.02
(100,500,1000)	0.06	0.12	0.08	0.03	0.04	0.05
(100,1000,10000)	1.07	0.85	0.70	0.27	1.07	1.05
(500,2500,10000)	15.30	11.09	6.25	7.27	13.51	14.76

of [15], and the equation in the 2nd line behind equation (9) of [15]), while QR-2 needs k^2 multiplications and additions for the back-substitutions to solve $2 k \times k$ triangular systems (i.e. equations (20) and (21) of [15]).

Moreover, we also utilize the shared MATLAB code [10] for Table I of [17], which compares the computation times of StOMP with those of the OMP implementation by the Cholesky Factorization. We simply add the computation times of the proposed OMP implementations, to obtain the following Table II. As shown in Table II, the proposed three OMP implementations are usually faster than the OMP implementation (by the Cholesky Factorization) employed in [17], while the computation times of Proposed-1 are even comparable to those of StOMP (i.e. CFAR and CFDR in Table II).

V. CONCLUSION

Based on the recently proposed efficient inverse Cholesky factorization [16], we propose three efficient implementations of OMP, and compare them with the existing implementations of OMP. Our simulations show that the error accumulation in the proposed implementations is insignificant, as that in the existing ones. Among all the implementations, the proposed implementation 1 theoretically needs the least computational complexity in the k -th iteration ($k > 1$), and is the fastest in the simulations for almost all problem sizes. Among the implementations that spend N^2 memories to store the Gram matrix of the dictionary, the proposed implementation 1 needs the least memories. As the memory-saving variants of the proposed implementation 1, the proposed implementations 2 and 3 save the N^2 memories for the Gram matrix at the expense of higher computational complexity, and in the simulations they are faster than the existing implementations for most problem sizes. With respect to the existing efficient implementation (by the QR Factorization) that does not spend N^2 memories to store the Gram matrix, the proposed implementation 2 needs less computational complexity and a little more memories, while the proposed implementation 3 needs the same complexity and less memories.

REFERENCES

- [1] D. Donoho, "Compressed sensing", *IEEE Trans. on Information Theory*, vol. 52, no. 4, pp. 1289-1306, April 2006.
- [2] Chen S B, Donoho D L, Saunders M A., "Atomic decomposition by basis pursuit", *SIAM Journal on Scientific Computing*, 1998, 20(1): 33-61.
- [3] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries", *IEEE Trans. on SP*, vol. 41, no. 12, pp. 3397-3415, 1993.
- [4] S. Ji, Y. Xue and L. Carin, "Bayesian compressive sensing", *IEEE Trans. Signal Processing*, vol. 56, no. 6, pp. 2346-2356, June 2008.
- [5] Y. Pati, R. Rezaifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition", in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Nov. 1993.
- [6] Y. Fang, L. Chen, J. Wu and B. Huang, "GPU Implementation of Orthogonal Matching Pursuit for Compressive Sensing", *ICPADS*, 2011.
- [7] T. Blumensath and M. E. Davies, "In Greedy Pursuit of New Directions: (Nearly) Orthogonal Matching Pursuit by Directional Optimisation", *Proc. EUSIPCO*, Poznan, Poland, 2007.
- [8] J.A. Tropp and A.C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit", *IEEE Trans. on Information Theory*, vol. 53, no. 12, pp. 4655-4666, December 2007.
- [9] S. F. Cotter, J. Adler, B. D. Rao, and K. Kreutz-Delgado, "Forward sequential algorithms for best basis selection", *IEEE Proc. Vision, Image, Signal Process.*, vol. 146, no. 5, pp. 235-244, 1999.
- [10] D. Donoho, V. Stodden, and Y. Tsaig, "Sparselab", <http://sparselab.stanford.edu/>, 2007.
- [11] I. Damnjanovic, M. E. P. Davies, and M. D. Plumley, "Smallbox - an evaluation framework for sparse representations and dictionary learning algorithms", *Proc. Latent Variable Analysis/Independent Component Analysis*, St. Malo, France, Sep. 2010, pp. 418-425.
- [12] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit", *CS Technion*, 2008.
- [13] D. Yang and G. D. Peterson, "Compressed sensing and Cholesky decomposition on FPGAs and GPUs", *Parallel Computing*, vol. 38, no. 8, pp. 421-37, August 2012.
- [14] J.A. Tropp, "Greed is good: algorithmic results for sparse approximation", *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231-2242, Oct. 2004.
- [15] B.L. Sturm and M.G. Christensen, "Comparison of orthogonal matching pursuit implementations", *EUSIPCO*, 2012.
- [16] H. Zhu, W. Chen, B. Li and F. Gao, "An Improved Square-root Algorithm for V-BLAST Based on Efficient Inverse Cholesky Factorization", *IEEE Trans. on Wireless Communications*, vol. 10, no. 1, pp. 43-48, 2011.
- [17] D. Donoho, Y. Tsaig, I. Drori, and J.L. Starck, "Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit", *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1094-1121, Feb. 2012.
- [18] H. Huang, A. Makur, "Backtracking-Based Matching Pursuit Method for Sparse Signal Reconstruction", *IEEE Signal Processing Letters*, vol. 18, no. 7, pp. 391-394, July 2011.
- [19] T. Blumensath and M. E. Davies, "Gradient pursuits", *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2370-2382, June 2008.
- [20] B. L. Sturm, M. G. Christensen, and R. Gribonval, "Cyclic pure greedy algorithms for recovering compressively sampled sparse signals", *Proc. Asilomar Conf. Signals, Systems, and Computers*, Nov. 2011.
- [21] D. Yang, J. Sun, J. Lee, G. Liang, D. D. Jenkins, G. D. Peterson, and H. Li, "Performance Comparison of Cholesky Decomposition on GPUs and FPGAs", *SAAHPC*, knoxville, TN, 2010.
- [22] H. Zhu, W. Chen, G. Yang, "An Efficient Implementation for OMP Based on Inverse Cholesky Factorization and its Application to Simultaneous OMP in Blind Sub-Nyquist Sampling System", submitted to *IEEE Trans. on Signal Processing*.
- [23] B.L. Sturm and M.G. Christensen, <http://imi.aau.dk/~bst>.
- [24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.
- [25] E. J. Baranowski, "Triangular factorization of inverse data covariance matrices", *ICASSP*, pp. 2245 - 2247, vol. 3, Apr. 1991.