Comparison of Orthogonal Matching Pursuit Implementations

Sturm, Bob L.; Christensen, Mads Græsbøll

Published in:
Proceedings of the European Signal Processing Conference

Publication date:
2012

Document Version
Accepted author manuscript, peer reviewed version

COMPARISON OF ORTHOGONAL MATCHING PURSUIT IMPLEMENTATIONS

Bob L. Sturm and Mads Græsbøll Christensen

Dept. of Architecture, Design and Media Technology, Aalborg University, Denmark
E-mail: bst,mgc@create.aau.dk

ABSTRACT

We study the numerical and computational performance of three implementations of orthogonal matching pursuit: one using the QR matrix decomposition, one using the Cholesky matrix decomposition, and one using the matrix inversion lemma. We find that none of these implementations suffer from numerical error accumulation in the inner products or the solution. Furthermore, we empirically compare the computational times of each algorithm over the phase plane.

Index Terms— Orthogonal matching pursuit, algorithms

1. INTRODUCTION

We wish to efficiently model a signal $\mathbf{u}$ by atoms defined in a dictionary $\mathcal{D} := \{\boldsymbol{\varphi}_\omega \in \mathbb{C}^M : \omega \in \Omega := \{1, 2, \ldots, N\}\}$ by

$$\mathbf{u} = \boldsymbol{\Phi}\mathbf{x} + \mathbf{n} \tag{1}$$

where $\boldsymbol{\Phi} := [\boldsymbol{\varphi}_1|\boldsymbol{\varphi}_2|\cdots|\boldsymbol{\varphi}_N]$, and $\mathbf{n}$ is noise. We wish our model $\hat{\mathbf{x}}$ to have $s < M \ll N$ non-zero elements (sparsity), such that $\|\mathbf{u} - \boldsymbol{\Phi}\hat{\mathbf{x}}\|$ is on the order of $\|\mathbf{n}\|$. This problem is sparse approximation or representation [1, 2], and many algorithms have been designed to solve it [1–3]. In this work, we are concerned with only one: the orthogonal greedy algorithm, orthogonal matching pursuit (OMP) [4]. OMP allows one to directly tune the sparsity or order of the approximation, and its performance is competitive with other more complex algorithms, e.g., convex optimization [5, 6].

OMP is simple and straightforward to implement in a naive manner, and its computational complexity can be reduced for a region of problem dimensions using matrix decomposition, e.g., the QR or Cholesky decomposition [4, 7, 8]. The numerical behavior of OMP in its different implementations, however, have not been studied. In this paper, we review three computationally efficient implementations of OMP. We are not concerned with guarantees of sparse solutions for OMP, but only with the numerical behavior and computational complexity of each implementation as a function of problem size. Furthermore, we do not consider approximate implementations of OMP, such as gradient pursuit [9], or cyclic matching pursuit [10].

B. L. Sturm is supported in part by Independent Postdoc Grant 11-105218 from Det Frie Forskningsråd.

Let $\Omega_k \subset \Omega$ denote an ordered set of k indices into $\mathcal{D}$. We denote the ith element of a set $\mathcal{I}$ by $\mathcal{I}(i)$. The matrix $\boldsymbol{\Phi}_k$ is composed of the ordered atoms indexed by Ω_k. We denote $\mathbf{P}_k := \boldsymbol{\Phi}_k\boldsymbol{\Phi}_k^\dagger$, where we assume $\mathrm{rank}(\boldsymbol{\Phi}_k) = k$, and thus $\boldsymbol{\Phi}_k^\dagger := (\boldsymbol{\Phi}_k^H\boldsymbol{\Phi}_k)^{-1}\boldsymbol{\Phi}_k^H$. The projection matrix $\mathbf{P}_k^\perp := \mathbf{I} - \mathbf{P}_k$ is the orthogonal projection onto space orthogonal to the span of $\mathcal{D}_{\Omega_k}$, or equivalently, the left null space of $\boldsymbol{\Phi}_k$. We denote the inner product between two vectors in $\mathbb{C}^M$ as $\langle \mathbf{u}_1, \mathbf{u}_2 \rangle := \mathbf{u}_2^H\mathbf{u}_1$, where ^{H} denotes the complex conjugate transpose. The vector $\mathbf{e}_k$ is the kth element of the standard basis of $\mathbb{R}^N$, i.e., all zeros with a 1 in its kth row. Finally, unless explicitly noted, all norms are the ℓ_2-norm, i.e., $\|\mathbf{x}\|^2 := \langle \mathbf{x}, \mathbf{x} \rangle$.

2. COMPUTATIONAL VARIETIES OF OMP

If in (1) $\|\mathbf{n}\| = 0$, OMP attempts to solve

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject to } \mathbf{u} = \boldsymbol{\Phi}\mathbf{x} \tag{2}$$

where $\|\mathbf{x}\|_0$ counts the number of non-zeros in $\mathbf{x}$. When there is noise, OMP attempts to solve

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject to } \|\mathbf{u} - \boldsymbol{\Phi}\mathbf{x}\|^2 \leq \epsilon^2 \tag{3}$$

where $\epsilon^2 > 0$. In its kth iteration, OMP augments Ω_{k-1} by $\Omega_k := \Omega_{k-1} \cup \{n_k\}$ using the selection criterion

$$n_k := \arg\min_{\beta \in \mathbb{C}, n \in \Omega} \|\mathbf{r}_{k-1} - \beta\boldsymbol{\varphi}_n\| = \arg\max_{n \in \Omega} \frac{|\langle \mathbf{r}_{k-1}, \boldsymbol{\varphi}_n \rangle|}{\|\boldsymbol{\varphi}_n\|}. \tag{4}$$

OMP then orthogonalizes the residual, $\mathbf{r}_{k-1} := \mathbf{P}_{k-1}^\perp\mathbf{u}$, and updates the solution, initialized as $\hat{\mathbf{x}}_k := \mathbf{0}$, becomes

$$[\hat{\mathbf{x}}_k]_{\Omega_k} := \boldsymbol{\Phi}_{k-1}^\dagger\mathbf{u} \tag{5}$$

where $[\hat{\mathbf{x}}_k]_{\Omega_k}$ are those elements of $\hat{\mathbf{x}}_k$ at indices Ω_k. OMP is initialized $\Omega_0 := \emptyset$ and $\hat{\mathbf{x}}_0 := \mathbf{0}$.

As has been shown before, e.g., [4, 7, 8], one can make OMP efficient through matrix decomposition. After we review the naive implementation of OMP, we review two approaches using matrix decomposition, and one approach employing the matrix inversion lemma. For our evaluations of algorithmic complexity, we assume that multiplies and adds have the same cost, as done in, e.g., [1, 9, 11]; but we keep only dominating terms, and do not carry coefficients. Table

Algorithm	Complexity	Memory
Naïve	$NM + Mk + Mk^2 + k^3$	MN
Chol-1	$NM + Mk + k^2$	$N^2 + NM + k + k^2$
Chol-2	$Nk + k^2$	$N^2 + NM + k + k^2$
QR-1	$NM + Mk$; solve: K^2	$NM + Mk + k^2$
QR-2	$Nk + Mk + k^2$	$N^2 + NM + Mk + k^2$
MIL	$Nk + Mk$	$N^2 + NM + Mk$

Table 1. Summary of complexities and memory requirements for each computational approach to OMP.

1 summarizes the computational complexities and memory requirements of each implementation in terms of the parameters. Since the complexity of these implementations are multivariate, big-O analysis of complexity is troublesome [12]; however, we find support from regression analysis for how the parameters affect the computation times we observe. Since we are concerned with practical implementations, we only consider $M < N < \infty$, and assume we can compute and store $\mathbf{\Phi}$, and its Gramian $\mathbf{\Phi}^H\mathbf{\Phi}$.

2.1. OMP the Naive Way

In its kth iteration, a naive implementation of OMP creates and searches through the set

$$\mathcal{I}_{k-1} := \{\langle \mathbf{r}_{k-1}, \boldsymbol{\varphi}_n\rangle / \|\boldsymbol{\varphi}_n\|\}_{n\in\Omega} \tag{6}$$

which has a complexity of $\mathcal{O}(MN)$. After selecting a new atom, OMP orthogonalizes the residual $\mathbf{r}_k := \mathbf{P}_k^{\perp}\mathbf{u} = \mathbf{u} - \mathbf{\Phi}_k(\mathbf{\Phi}_k^H\mathbf{\Phi}_k)_1^{-1}(\mathbf{\Phi}_k^H\mathbf{u})$. Assuming that the cost of inverting a complex $k \times k$ matrix is at least $\mathcal{O}(k^3)$, this procedure has a complexity of $\mathcal{O}(Mk + Mk^2 + k^3)$. We find this from evaluating $[\mathbf{u} - (\mathbf{\Phi}_k[(\mathbf{\Phi}_k^H\mathbf{\Phi}_k)_1^{-1}(\mathbf{\Phi}_k^H\mathbf{u})_2]_3)_4]_5$, with subscripts denoting the order of operations. Thus, the kth iteration of the naive OMP implementation has complexity $\mathcal{O}(NM + Mk + Mk^2 + k^3)$, which shows that orthogonalization dominates complexity. The memory required for the naive approach is $\mathcal{O}(NM)$ for the dictionary and inner products. While this implementation has a higher computational complexity than the approaches below, it has the lowest memory requirement.

2.2. OMP by the Cholesky Decomposition

This implementation is used in several software implementations of OMP, e.g., [13, 14].[1] Consider that in its kth step, OMP has the set of inner products in (6), the set of atom norms $\{\|\boldsymbol{\varphi}_n\|\}_{n\in\Omega}$, the set of initial inner products

$$\mathcal{I}_0 := \{\langle \mathbf{u}, \boldsymbol{\varphi}_n\rangle / \|\boldsymbol{\varphi}_n\|\}_{n\in\Omega} \tag{7}$$

the set of Gramian products for $l \in \Omega$

$$\Gamma_l := \{\langle \boldsymbol{\varphi}_l, \boldsymbol{\varphi}_n\rangle / \|\boldsymbol{\varphi}_n\|\}_{n\in\Omega} \tag{8}$$

and, finally, the Cholesky decomposition of the Gramian $\mathbf{\Phi}_{k-1}^H\mathbf{\Phi}_{k-1} = \mathbf{L}_{k-1}\mathbf{L}_{k-1}^H$. Being positive semidefinite, any Gramian has a Cholesky decomposition.

[1] http://code.soundsoftware.ac.uk/projects/smallbox

OMP selects the new atom index n_k from $\mathcal{I}_{k-1}$ (6), and updates the Cholesky factorization as follows. Notice

$$\begin{aligned}\mathbf{L}_k\mathbf{L}_k^H = \mathbf{\Phi}_k^H\mathbf{\Phi}_k &= \begin{bmatrix} \mathbf{\Phi}_{k-1}^H\mathbf{\Phi}_{k-1} & \mathbf{\Phi}_{k-1}^H\boldsymbol{\varphi}_{n_k} \\ \boldsymbol{\varphi}_{n_k}^H\mathbf{\Phi}_{k-1} & \|\boldsymbol{\varphi}_{n_k}\|^2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{L}_{k-1} & 0 \\ \mathbf{v}^H & b \end{bmatrix}\begin{bmatrix} \mathbf{L}_{k-1}^H & \mathbf{v} \\ 0 & b^* \end{bmatrix} \end{aligned} \tag{9}$$

where $\|\mathbf{v}\|^2 + |b|^2 = \|\boldsymbol{\varphi}_{n_k}\|^2$. Thus, we first need to solve for $\mathbf{v}$ in the triangular system $\mathbf{L}_{k-1}\mathbf{v} = \mathbf{\Phi}_{\Omega_{k-1}}^H\boldsymbol{\varphi}_{n_k}$ — the right-hand side coming from Γ_{n_k} in (8) — and then update the Cholesky factor by adding a row and column, i.e.,

$$\mathbf{L}_k := \begin{bmatrix} \mathbf{L}_{k-1} & 0 \\ \mathbf{v}^H & \sqrt{\|\boldsymbol{\varphi}_{n_k}\|^2 - \|\mathbf{v}\|^2} \end{bmatrix}. \tag{10}$$

Since solving a triangular system of size k has complexity $\mathcal{O}(k^2)$, the complexity of the search and update is $\mathcal{O}(N + k^2 + k)$. Defining the least squares solution $\mathbf{x}_k := \mathbf{\Phi}_k^{\dagger}\mathbf{u}$, then

$$\mathbf{\Phi}_k^H\mathbf{\Phi}_k\mathbf{x}_k = \mathbf{L}_k\mathbf{L}_k^H\mathbf{x}_k = \mathbf{\Phi}_k^H\mathbf{u} \tag{11}$$

where the right-hand side comes from $\mathcal{I}_0$. OMP now only needs to solve two triangular systems by back-substitution

$$\mathbf{L}_k\mathbf{y} = \mathbf{\Phi}_k^H\mathbf{u} \tag{12}$$

$$\mathbf{L}_k^H\mathbf{x}_k = \mathbf{y} \tag{13}$$

with a complexity $\mathcal{O}(k^2)$. The new residual energy is then

$$\|\mathbf{r}_k\|^2 = \|\mathbf{u}\|^2 - \|\mathbf{y}\|^2. \tag{14}$$

Finally, OMP can update the inner products (6) in one of two ways. For the first (Chol-1), OMP explicitly computes the residual signal $\mathbf{r}_k := \mathbf{u} - \mathbf{\Phi}_k\mathbf{x}_k$ with a cost of $\mathcal{O}(Mk)$, and then directly computes the inner products $\mathcal{I}_k := \{\langle \mathbf{r}_k, \boldsymbol{\varphi}_n\rangle / \|\boldsymbol{\varphi}_n\|\}_{n\in\Omega}$ with a cost of $\mathcal{O}(NM)$. Thus, the total complexity of the kth iteration is $\mathcal{O}(NM + Mk + k^2)$. In the second case (Chol-2), OMP updates the initial products (7)

$$\mathcal{I}_k := \left\{\frac{\langle \mathbf{r}_k, \boldsymbol{\varphi}_n\rangle}{\|\boldsymbol{\varphi}_n\|}\right\}_{n\in\Omega} = \left\{\mathcal{I}_0(n) - \sum_{l=1}^{k} x_l\Gamma_{n_l}(n)\right\}_{n\in\Omega} \tag{15}$$

where x_l is the lth element of $\mathbf{x}_k$. This entire update has a complexity of $\mathcal{O}(Nk)$, and thus the total complexity of the kth iteration in the second case is $\mathcal{O}(Nk + k^2)$. The iteration when the second approach becomes more complex than the first one is $k \approx MN/(N - M)$. In both cases, the memory requirements of OMP is $\mathcal{O}(N^2 + NM + k^2 + k)$.

2.3. OMP by the QR Decomposition

This implementation is suggested in [8]. Consider that in the kth step of OMP we have the set of current inner products (6), initial inner products (7), as well as the QR decomposition

$\mathbf{\Phi}_{k-1} = \mathbf{Q}_{k-1}\mathbf{R}_{k-1}$. Any matrix can be decomposed into a QR decomposition, with $\mathbf{Q}$ a unitary but not unique matrix. Once OMP searches through $\mathcal{I}_{k-1}$ (6) to find n_k, it defines $\mathbf{w} := \mathbf{Q}_{k-1}^H \boldsymbol{\varphi}_{n_k}$. Thus, $\mathbf{Q}_{k-1}\mathbf{w}$ is the least squares projection of $\boldsymbol{\varphi}_{n_k}$ onto the span of the atoms indexed by Ω_{k-1}, and so $\boldsymbol{\varphi}_{n_k} - \mathbf{Q}_{k-1}\mathbf{w}$ is the new direction contributed by the atom. OMP updates $\mathbf{Q}_{k-1}$ by adding this unit-norm direction

$$\mathbf{Q}_k := \begin{bmatrix} \mathbf{Q}_{k-1} & \frac{\boldsymbol{\varphi}_{n_k} - \mathbf{Q}_{k-1}\mathbf{w}}{\sqrt{\|\boldsymbol{\varphi}_{n_k}\|^2 - \|\mathbf{w}\|^2}} \end{bmatrix}. \tag{16}$$

Since $\mathbf{\Phi}_k = \mathbf{Q}_k\mathbf{R}_k = [\mathbf{\Phi}_{k-1}|\boldsymbol{\varphi}_{n_k}]$, we see that OMP can update the other factor by

$$\mathbf{R}_k := \begin{bmatrix} \mathbf{R}_{k-1} & \mathbf{w} \\ \mathbf{0}^T & \sqrt{\|\boldsymbol{\varphi}_{n_k}\|^2 - \|\mathbf{w}\|^2} \end{bmatrix}. \tag{17}$$

With the search, updating these matrices has complexity $\mathcal{O}(N + Mk)$. OMP computes the residual energy recursively

$$\begin{aligned} \|\mathbf{r}_k\|^2 &= \|\mathbf{u} - \mathbf{Q}_{k-1}\mathbf{Q}_{k-1}^H\mathbf{u} - \mathbf{q}_k\mathbf{q}_k^H\mathbf{u}\|^2 \\ &= \|\mathbf{r}_{k-1}\|^2 - |\langle \mathbf{u}, \mathbf{q}_k\rangle|^2 \end{aligned} \tag{18}$$

where $\mathbf{q}_k$ is the last column of $\mathbf{Q}_k$.

OMP then updates the projections (6) in one of two ways. First (QR-1), notice OMP need not compute the residual since $\mathbf{r}_k = \mathbf{r}_{k-1} - \mathbf{q}_k\mathbf{q}_k^H\mathbf{u}$. Thus, the updated projections are

$$\mathcal{I}_k := \left\{ \frac{\langle \mathbf{r}_k, \boldsymbol{\varphi}_n\rangle}{\|\boldsymbol{\varphi}_n\|} \right\} = \left\{ \mathcal{I}_{k-1}(n) - \langle \mathbf{u}, \mathbf{q}_k\rangle \frac{\langle \mathbf{q}_k, \boldsymbol{\varphi}_n\rangle}{\|\boldsymbol{\varphi}_n\|} \right\} \tag{19}$$

for $\{n \in \Omega\}$. This has complexity $\mathcal{O}(NM)$. OMP thus avoids explicitly computing the solution and residual during decomposition. Only after the final iteration K does OMP find the solution. Since the least squares solution satisfies $\mathbf{\Phi}_K^H\mathbf{\Phi}_K\mathbf{x}_K = \mathbf{\Phi}_K^H\mathbf{u}$, we see that by substituting the QR decomposition on the left side, OMP finds $\mathbf{x}_K$ by solving

$$\mathbf{R}_K^H\mathbf{y} = \mathbf{\Phi}_K^H\mathbf{u} \tag{20}$$

$$\mathbf{R}_K\mathbf{x}_K = \mathbf{y} \tag{21}$$

where the right hand side of the first system comes from $\mathcal{I}_0$ (7). Solving these with back-substitution has complexity $\mathcal{O}(K^2)$. In this way, the total complexity of the kth iteration of QR-1 is $\mathcal{O}(NM + Mk)$. Its memory requirements are $\mathcal{O}(NM + Mk + k^2)$ since it need not store the Gramian.

In the second approach (QR-2), OMP updates the projections (6) using (15) after solving (20) and (21). It can also update the residual energy using (14) with the solution of (20). This reduces the complexity at the cost of computing the solution at each iteration and storing the dictionary Gramian (8). Thus, the total complexity of the kth of QR-2 is $\mathcal{O}(Nk + Mk + k^2)$. The cross-over point where the complexity of this approach exceeds QR-1 is $k(k + N) \approx NM$. Its memory requirements are $\mathcal{O}(N^2 + NM + Mk + k^2)$.

2.4. OMP by the Matrix Inversion Lemma

This implementation is suggested in [4]. Consider that in its kth iteration, OMP has the set of current inner products (6), initial inner products (7), the set of Gramian products (8), the current solution $\mathbf{x}_{k-1}$, and the biorthogonal basis of $\mathbf{\Phi}_{k-1}$, i.e., the columns of the matrix

$$\mathbf{\Psi}_{k-1} := \mathbf{\Phi}_{k-1}(\mathbf{\Phi}_{k-1}^H\mathbf{\Phi}_{k-1})^{-1}. \tag{22}$$

This means that $\mathbf{\Psi}_{k-1}^H\mathbf{\Phi}_{k-1} = \mathbf{\Phi}_{k-1}^H\mathbf{\Psi}_{k-1} = \mathbf{I}$. OMP guarantees that $\mathbf{\Phi}_k$ has full column rank when $k \leq M$, and so $\mathbf{\Psi}_k$ always exists when $k \leq M$. When it selects the new atom index n_k from $\mathcal{I}_{k-1}$ (6) OMP must update the biorthogonal basis and solution. Using the matrix inversion lemma, we see

$$(\mathbf{\Phi}_k^H\mathbf{\Phi}_k)^{-1} = \begin{bmatrix} (\mathbf{\Phi}_{k-1}^H\mathbf{\Phi}_{k-1})^{-1} + \lambda\mathbf{h}\mathbf{h}^H & -\lambda\mathbf{h} \\ -\lambda\mathbf{h}^H & \lambda \end{bmatrix} \tag{23}$$

where $\mathbf{h} := \mathbf{\Psi}_{k-1}^H\boldsymbol{\varphi}_{n_k}$, and $\lambda^{-1} := \|\boldsymbol{\varphi}_{n_k}\|^2 - \|\mathbf{\Phi}_{k-1}\mathbf{h}\|^2$. OMP thus computes the new biorthogonal basis by

$$\mathbf{\Psi}_k := \mathbf{\Phi}_k(\mathbf{\Phi}_k^H\mathbf{\Phi}_k)^{-1} = \begin{bmatrix} \mathbf{\Psi}_{k-1} - (\lambda\mathbf{v})\mathbf{h}^H | \lambda\mathbf{v} \end{bmatrix} \tag{24}$$

where $\mathbf{v} := \boldsymbol{\varphi}_{n_k} - \mathbf{\Phi}_{k-1}\mathbf{h}$. To update the biorthogonal basis then, OMP need only compute $\mathbf{h}$ and λ at a complexity of $\mathcal{O}(N + Mk)$, including the search. It is not necessary until the last step, but OMP can compute the new solution by

$$\mathbf{x}_k = \mathbf{\Psi}_k^H\mathbf{u} = \begin{bmatrix} \mathbf{x}_{k-1} \\ 0 \end{bmatrix} - \Delta \begin{bmatrix} \mathbf{h} \\ -1 \end{bmatrix} \tag{25}$$

where

$$\begin{aligned} \Delta := \lambda\mathbf{v}^H\mathbf{u} &= \lambda(\boldsymbol{\varphi}_{n_k} - \mathbf{\Phi}_{k-1}\mathbf{h})^H\mathbf{u} \\ &= \lambda\left[\mathcal{I}_0(n_k) - \sum_{l=1}^{k-1} h_l^*\mathcal{I}_0(n_l)\right]. \end{aligned} \tag{26}$$

OMP can then update the set of projections for $\{n \in \Omega\}$ by

$$\mathcal{I}_k := \left\{ \frac{\langle \mathbf{r}_k, \boldsymbol{\varphi}_n\rangle}{\|\boldsymbol{\varphi}_n\|} \right\} = \{\langle \mathbf{u} - \mathbf{\Phi}_k\mathbf{x}_k, \boldsymbol{\varphi}_n/\|\boldsymbol{\varphi}_n\|\rangle\} \tag{27}$$

which becomes

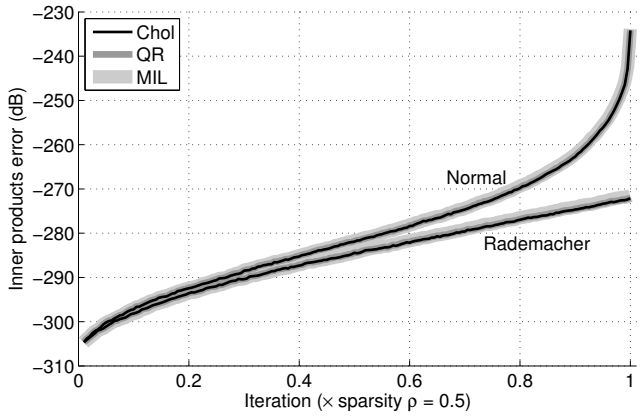
$$\mathcal{I}_k = \left\{ \mathcal{I}_{k-1}(n) - \Delta\Gamma_{n_k}(n) + \Delta\sum_{l=1}^{k-1} h_l\Gamma_{n_l}(n) \right\}. \tag{28}$$

OMP can also use $\mathbf{x}_k$ in (15) at the same computational cost.

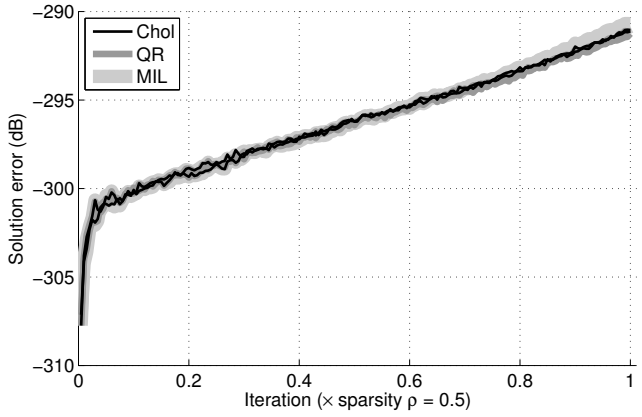
Updating the residual energy is simply done by

$$\begin{aligned} \|\mathbf{r}_k\|^2 = \|\mathbf{u} - \mathbf{\Phi}_k\mathbf{x}_k\|^2 &= \left\| \mathbf{u} - \mathbf{\Phi}_k \left(\begin{bmatrix} \mathbf{x}_{k-1} \\ 0 \end{bmatrix} - \Delta \begin{bmatrix} \mathbf{h} \\ -1 \end{bmatrix} \right) \right\|^2 \\ &= \|\mathbf{r}_{k-1}\|^2 + \frac{|\Delta|^2}{\lambda} - 2\mathrm{Re}\{\Delta\mathcal{I}_{k-1}(n_k)\} \end{aligned} \tag{29}$$

since $\mathbf{r}_{k-1}$ is orthogonal to all selected atoms except the last. In total, the kth iteration of OMP with the matrix inversion lemma has a complexity of $\mathcal{O}(Nk + Mk)$, and memory requirements of $\mathcal{O}(N^2 + MN + Mk)$.



(a) Mean error in inner products



(b) Mean error in solution

Fig. 1. Errors (30) and (31) of the three approaches with respect to the naive one, for signals distributed Rademacher and Normal, as a function of algorithm iteration ($N = M = 400$).

3. NUMERICAL EXPERIMENTS

We now test the numerical performance and computational time of the three implementations (QR-1, Chol-2, MIL), and compare with the naive implementation.[2] In every experiment, we sample $\mathbf{\Phi}$ from the uniform spherical ensemble, meaning we sample each entry independently from a Normal distribution, and then normalize each column. We test sparse vectors with non-zero entries sampled from either a Rademacher distribution, i.e., equiprobable in $\{-1, 1\}$, or a Normal distribution. We use MATLAB on a 64-bit machine.

In our first experiment, we investigate the accumulation of numerical errors in the recursive computation of the inner products and solution, compared to the values given by the naive implementation. We perform 50 independent trials for signals with sparsity $K = 200$ in ambient dimension $N = 400$, and projected into a space of dimension $M = 400$. In this experiment, $M = N = 400$, but smaller values of M produce the same results. For each iteration, we compute the mean relative error of the inner products, e.g., we compute for the ith trial of the Cholesky approach the normalized squared kth difference

$$h_i^{(\text{Chol})}(k) := \frac{\sum_{n=1}^{N} |\mathcal{I}_k^{(\text{Chol})}(n) - \mathcal{I}_k^{(\text{Naive})}(n)|^2}{\sum_{n=1}^{N} |\mathcal{I}_k^{(\text{Naive})}(n)|^2} \tag{30}$$

and find the mean of $\{h_i^{\text{Chol}}(k)\}$ over several trials.

Figure 1(a) shows how the mean relative error of the inner products slightly increases as a function of iteration in terms of the sparsity of the signal. At the iteration $k/K = 1$, the algorithm has performed as many iterations as there are non-zero elements in the true solution. It is clear that the differences between implementations are inconsequential, but we see a small difference between signal distributions. In Fig. 1(b) we plot the mean relative error in the solutions (between the implementations and not between the true and the solution built by OMP), e.g., for the ith trial of the Cholesky approach the normalized squared kth difference

$$e_i^{(\text{Chol})}(k) := \|\mathbf{x}_k^{(\text{Chol})} - \mathbf{x}_k^{(\text{Naive})}\|^2 / \|\mathbf{x}_k^{(\text{Naive})}\|^2 \tag{31}$$

We see little difference between the approaches and the distributions. Furthermore, we see little differences between different sparsities ρ or signal dimension N.

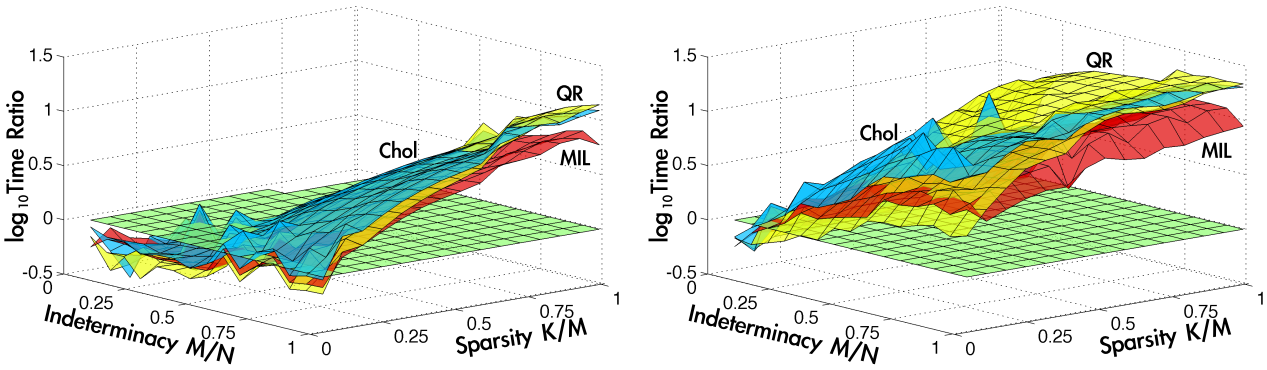
In our second experiment, we measure for each implementation the computation time of 10 independent runs of K iterations (sparsity) for $\mathbf{\Phi}$ of size $M \times N$. We use as similar code as possible to be fair in the comparisons, and measure times using `tic/toc` in MATLAB. For two different N, Fig. 2 shows $\log_{10}(T_{\text{imp}}(K, M, N)/T_{\text{naive}}(K, M, N))$. For small N, M and K the naive approach is the fastest; but as N increases, the other implementations become faster. For small numbers of iterations, the Cholesky implementation appears the fastest; but at larger iterations, the QR approach is faster.

Finally, to evaluate the results in Table 1, we find predictors for each implementation of the mean computation times for $N \in \{100, 200, \ldots, 800\}$, and 15 different M and K (as in Fig. 2). We perform least squares regression with a non-negativity constraint solving

$$\min_{\mathbf{w}} \|\mathbf{T} - [\mathbf{1}|\mathbf{X}]\mathbf{w}\|_2 \text{ subject to } \mathbf{w} \succeq 0 \tag{32}$$

where $\mathbf{T}$ is a matrix of measurements $T_{\text{imp}}(K, M, N)$, and $\mathbf{X}$ is a predictor matrix with values from each test, such as M, Mk, and Nk^2, with each column made to have unit variance. Figure 3 shows the weights of the various terms for Naive, QR-1, Chol-2, and MIL. We see the naive implementation is dominated by $k^2, MNk^2, (kM)^2$ and MNk^3. For Chol-2, we see only $Nk^2, (Nk)^2$ and k^3 are the strong predictors. QR-1 is dominated by Mk^2, MNk^2 and $(kM)^2$. And MIL appears dominated by k^2 and $(kM)^2$. The differences between these complexities and those in Table 1 are possibly a result of assuming big-O complexity applies to multivariable processes [12].

[2] Our experiments and the figures in this paper can be reproduced with the code at `http://imi.aau.dk/~bst`.



(a) $N = 200$ (b) $N = 800$

Fig. 2. Computation times of each implementation as a function of problem sizes for two ambient dimensions N.

4. CONCLUSION

We have investigated three different matrix factorization implementations of OMP, and compared them to the naive implementation. We find that, though their variables are updated recursively, the accumulation of error is insignificant in each implementation. Depending on the problem size, any of the four implementations can be the fastest. However, as the ambient dimension N increases, the computation time of the naive approach becomes much longer than for the other three. For particularly large problem sizes, as the number of iterations increases, the QR implementation appears to be the fastest. Future work will explore to what extents the QR implementation decreases the computational complexity of the Cholesky implementation.

5. REFERENCES

[1] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, Elsevier, Amsterdam, 3rd edition, 2009.

[2] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, New York, NY, USA, 2010.

[3] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proc. IEEE*, vol. 98, no. 6, pp. 948–958, June 2010.

[4] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 1993, pp. 40–44.

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

[6] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Info. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[7] M. Gharavi-Alkhansari and T. Huang, "A fast orthogonal matching pursuit algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1998.

[8] 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.

[9] T. Blumensath and M. E. Davies, "Gradient pursuits," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2370–2382, June 2008.

[10] B. L. Sturm, M. G. Christensen, and R. Gribonval, "Cyclic pure greedy algorithms for recovering compressively sampled sparse signals," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2011.

[11] S. Krstulovic and R. Gribonval, "MPTK: Matching pursuit made tractable," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Toulouse, France, Apr. 2006, vol. 3, pp. 496–499.

[12] R. R. Howell, "On asymptotic notation with multiple variables," Tech. Rep., Kansas State University, Manhattan, KS, USA, Jan 2008.

[13] D. Donoho, V. Stodden, and Y. Tsaig, "Sparselab," http://sparselab.stanford.edu/, 2007.

[14] I. Damnjanovic, M. E. P. Davies, and M. D. Plumbley, "Smallbox - an evaluation framework for sparse representations and dictionary learning algorithms," in *Proc. Latent Variable Analysis/Independent Component Analysis*, St. Malo, France, Sep. 2010, pp. 418–425.

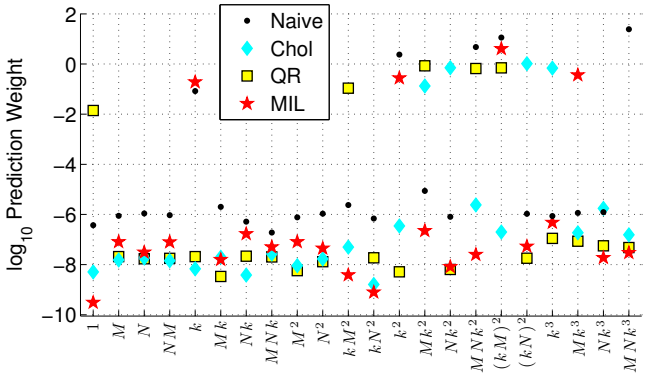


Fig. 3. Prediction weights for the regression on the mean computation times for $N \in \{100, 200, \ldots, 800\}$.