Non-Negative Orthogonal Greedy Algorithms

Thi Thanh Nguyen, Jérôme Idier, Charles Soussen, El-Hadi Djermoune

Non-Negative Orthogonal Greedy Algorithms

Thanh T. Nguyen, Jérôme Idier, *Member, IEEE,* Charles Soussen, *Member, IEEE,*
and El-Hadi Djermoune, *Member, IEEE*

Abstract

Orthogonal greedy algorithms are popular sparse signal reconstruction algorithms. Their principle is to select atoms one by one. A series of unconstrained least-squares subproblems of gradually increasing size is solved to compute the approximation coefficients, which is efficiently performed using a fast recursive update scheme. When dealing with non-negative sparse signal reconstruction, a series of *non-negative* least-squares subproblems have to be solved. Fast implementation becomes tricky since each subproblem does not have a closed-form solution anymore. Recently, non-negative extensions of the classical orthogonal matching pursuit and orthogonal least squares algorithms were proposed, using slow (*i.e.*, non-recursive) or recursive but inexact implementations. In this paper, we revisit these algorithms in a unified way. We define a class of non-negative orthogonal algorithms and exhibit their structural properties. We propose a fast and exact implementation based on the active-set resolution of non-negative least-squares and exploiting warm start initializations. The algorithms are assessed in terms of accuracy and computational complexity for a sparse spike deconvolution problem. We also present an application to near-infrared spectra decomposition.

Index Terms

Orthogonal greedy algorithms; sparse reconstruction; non-negativity; non-negative least-squares; active-set algorithms.

This work was supported by the project ANR 15-CE23-0021 BECOSE.

It was carried out in part while T. T. Nguyen was visiting L2S during the academic years 2017-2019.

T. T. Nguyen and E.-H. Djermoune are with the Université de Lorraine and CNRS at the Centre de Recherche en Automatique de Nancy (CRAN), F-54506 Vandœuvre-lès-Nancy, France (e-mail: {thi-thanh.nguyen,el-hadi.djermoune}@univ-lorraine.fr).

C. Soussen was with CRAN. He is now with CentraleSupélec at the Laboratoire des Signaux et Systèmes (CentraleSupélec-CNRS-Université Paris-Sud, Université Paris-Saclay), F-91192 Gif-sur-Yvette, France (e-mail: charles.soussen@centralesupelec.fr).

J. Idier is with CNRS at the Laboratoire des Sciences du Numérique de Nantes (UMR 6004), F-44321 Nantes, France (e-mail: jerome.idier@ls2n.fr).

I. INTRODUCTION

In the last decade, sparse approximation has received considerable attention in the signal and image processing community, in connection with compressive sensing. Greedy algorithms for sparse signal reconstruction are very popular iterative schemes. Their principle is to repeatedly *(i)* enrich the sparse support by selecting a new dictionary atom, and then *(ii)* update the sparse approximation coefficients. In orthogonal greedy algorithms, the sparse approximation signal is computed as the orthogonal projection of the data vector onto the subspace spanned by the selected atoms. Therefore, the coefficients can be estimated by solving an Unconstrained Least Squares (ULS) problem. Popular orthogonal greedy algorithms include Orthogonal Matching Pursuit (OMP) [1] and Orthogonal Least Squares (OLS) [2], also known as forward selection [3], Order Recursive Matching Pursuit (ORMP) [4], Optimized Orthogonal Matching Pursuit (OOMP) [5] and Pure Orthogonal Matching Pursuit [6]. OMP and OLS differ in the way the new atom is selected. In both cases, the atom inducing the largest decrease of the norm of the residual is selected. However, all nonzero atom weights are optimally tuned in OLS whereas only the new atom weight is considered in OMP, which amounts to selecting the atom having the largest inner product with the current residual. The computational complexity of OLS is obviously higher, since the selection rule requires to solve as many ULS problems as the number of candidate atoms. Fortunately, the ULS solutions have a closed-form expression, which can be recursively (fastly) updated when the support is enriched by a new element, see *e.g.,* [3]. Specifically, both OMP and OLS implementations are recursive and make use of matrix factorization, such as Gram-Schmidt orthogonalization, the Cholesky factorization or techniques utilizing the matrix inversion lemma [7].

Many application fields such as geoscience and remote sensing [8], [9], audio [10], chemometrics [11], bioinformatics [12], astrophysics [13] and computed tomography [14], give rise to inverse problems where the signal or image of interest is sparse, but also non-negative. In such contexts, a common practice is to regularize the inverse problem in order to favor both sparsity and non-negativity of the sought signal, see, *e.g.,* [12], [14]–[16]. Some classical sparse algorithms can be straightforwardly adapted to deal with non-negativity constraints. This is the case of proximal splitting algorithms and the Alternating Direction Method of Multipliers (ADMM) for convex optimization [17], [18], and of the DC algorithm (Difference of Convex functions) for nonconvex optimization [19], [20]. On the contrary, the non-negative extension of greedy algorithms is a challenging issue since the unconstrained least-squares subproblems are replaced by non-negative least-squares (NNLS) subproblems which do not have closed-form solutions anymore, so a subroutine solving NNLS is needed. There are different methods for NNLS solving [21] such as active-set [22], interior-point [23], and gradient-projection [24]. The latter two families typically require

to tune some stopping criteria in an empirical manner, resulting in *approximate* resolution of the NNLS problem. Here, we are focusing on active-set methods for NNLS solving since such methods have a greedy structure and exactly solve NNLS problems after a finite number of iterations. Although our focus will be on extensions of orthogonal greedy schemes to the non-negative case, let us mention that several other non-negative sparse methods have been elaborated on the basis of the active-set NNLS algorithm, *e.g.,* hard-thresholded NNLS [25], [26] and Sparse NNLS or Reverse Sparse NNLS [27].

Several existing contributions deal with orthogonal greedy algorithms in the non-negative case. Non-Negative OMP (NNOMP) was first proposed by Bruckstein *et al.* [28] as a direct generalization of OMP. At each iteration, the atom having the maximum *positive* inner product with the current residual is selected. Contrary to OMP, negative inner products are discarded. Then, the sparse approximation coefficients are updated by solving the NNLS problem related to the augmented subset. The canonical (*i.e.*, non-recursive) NNOMP implementation of Bruckstein *et al.* [28] solves NNLS subproblems independently. Later, Yaghoobi *et al.* proposed an accelerated version named Fast Non-Negative OMP (FNNOMP), which avoids solving NNLS subproblems but rather recursively approximates the sought solution using QR matrix factorization [29]. Although FNNOMP is much faster than canonical NNOMP, it is an approximate version likely to deliver a different output. In [30], Yaghoobi *et al.* introduced a canonical version of Non-Negative OLS (NNOLS), defined as a direct non-negative extension of OLS. The principle of NNOLS is to select the atom for which the positive residual (the residual corresponding to the NNLS solution) is minimum. This selection rule appears to be time-consuming since one needs to compute as many positive residuals as the number of candidate atoms, *i.e.*, $n-k$ NNLS problems have to be solved at iteration k, with n the size of the dictionary.

Yaghoobi *et al.* [30] further proposed two accelerated versions of NNOLS named Suboptimal NNOLS (SNNOLS) and Fast NNOLS (FNNOLS). These proposals can be understood from a dual interpretation of OLS in the unconstrained setting. It is well-known than the OLS selection rule can be formulated in two equivalent ways [5], [31]: OLS selects the atom *(i)* inducing the minimum value of the data residual in ℓ_2-norm; *(ii)* whose orthogonal projection onto the subspace orthogonal to the active atoms has a maximum angle with the current data residual. When dealing with non-negativity, SNNOLS [30] selects the atom that is positively correlated with the current residual whose projection forms a maximum angle with the residual. However, the equivalence that holds for OLS is not true anymore since maximizing the angle for positive inner products is not equivalent to minimizing the ℓ_2-norm of the positive residual. Therefore, SNNOLS iterates do not generally identify with NNOLS iterates, hence the *suboptimal* algorithm denomination. In SNNOLS, the projected atoms have to be recomputed whenever the support is updated. Moreover, an NNLS problem must be solved at each iteration to update the sparse approximation

coefficients. FNNOLS is a recursive implementation in the same spirit as FNNOMP, where no NNLS problem needs to be solved anymore. It shall be noticed that FNNOLS and SNNOLS do not necessarily deliver the same iterates, and that both can be viewed as approximate versions of NNOLS.

Generally speaking, the "orthogonal" denomination of NNOMP and NNOLS is somewhat abusive since when the support set S is updated, the related NNLS solution may not identify with the orthogonal projection of the data vector onto the span of atoms indexed by S, but rather with its projection onto their positive span. Both projected vectors differ as soon as the NNLS solution has zero entries, *i.e.*, when some non-negativity constraints become active. Therefore, in NNOMP, NNOLS and their derived versions [28]–[30], the support of the sparse vector at the current iteration is a subset of the current support set S (which is expanded at each iteration) and may not identify with it. In turn, more than K iterations may be necessary to reach a truly K-sparse representation.

Our contributions are twofold. First, non-negative orthogonal greedy algorithms are revisited in a unified way. The algorithms under study share four desirable properties:

1) The norm of the data residual is always decreasing when a new atom enters the solution support.
2) The algorithm does not stop while additional atom selections would make it decrease, unless an explicit stopping condition is reached.
3) A so-called compression step is included to shrink the support set by removing the atoms having zero coefficients, so the support set identifies to the support of the current NNLS solution.
4) The residual vector is orthogonal to the selected atoms. In other words, the sparse approximation vector identifies with the orthogonal projection of the data vector onto the span of the selected atoms.

These structural properties are exhibited and compared to those of existing non-negative greedy algorithms. The second contribution is a fast and exact implementation of non-negative orthogonal greedy algorithms exploiting the active-set algorithm [22] for NNLS solving, and based on warm start initialization. Moreover, we elaborate on recursive implementations and we design further types of accelerations.

The paper is organized as follows. Section II introduces the family of so-called *Non-negative Orthogonal Greedy* (NNOG) algorithms. The different members of the family differ by the selection rule to pick a new atom at each iteration. It includes NNOLS, SNNOLS and NNOLS up to a modification of their structure, namely the compression step mentioned above. In Section III, we propose a fast implementation based on recursivity and on the use of warm starts for solving the NNLS subproblems. Section IV is devoted to NNOG acceleration. Sections V and VI propose numerical validations on a simulated example of sparse deconvolution and on the decomposition of real-world near-infrared (NIR) spectra. Finally, discussion and perspectives will be found in Section VII.

II. NON-NEGATIVE GREEDY ALGORITHMS

A. Basic definitions and notations

Given a data vector $\boldsymbol{y} \in \mathbb{R}^m$ and a dictionary $H \in \mathbb{R}^{m \times n}$, we are interested in finding a K-sparse non-negative weight vector $\boldsymbol{x} \in \mathbb{R}^n_+$ yielding an accurate approximation $\boldsymbol{y} \approx H\boldsymbol{x}$. This can be formulated as the constrained minimization program:

$$\min_{\boldsymbol{x} \geq \boldsymbol{0}} \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_0 \leq K. \tag{ℓ_0+}$$

where $\boldsymbol{x} \geq \boldsymbol{0}$ means that each entry of $\boldsymbol{x}$ is nonnegative, $\|\boldsymbol{x}\|_0$ is the ℓ_0-"norm" counting the number of nonzero entries in $\boldsymbol{x}$, and the quadratic fidelity-to-data term $\|\boldsymbol{y} - H\boldsymbol{x}\|_2^2$ measures the quality of approximation. The ℓ_2-norm $\|\cdot\|_2$ will be also denoted $\|\cdot\|$. Without loss of generality, we assume that H is column-normalized. Each dictionary column $\boldsymbol{h}_i$, $i = 1, \ldots, n$ is called an atom. We have the following useful identity for any two vectors $\boldsymbol{r}$, $\boldsymbol{h}$ of same length, $\boldsymbol{h}$ being normalized:

$$\min_{v \geq 0} \|\boldsymbol{r} - \boldsymbol{h}v\|^2 = \|\boldsymbol{r}\|^2 - \left(\max\{\boldsymbol{h}^{\mathrm{t}}\boldsymbol{r}, 0\}\right)^2 \tag{1}$$

where $.^{\mathrm{t}}$ stands for the transpose operator.

We denote by $S = \mathrm{supp}(\boldsymbol{x}) = \{i : \boldsymbol{x}(i) \neq 0\}$ the support of $\boldsymbol{x}$ ($\boldsymbol{x}(i)$ being the i-th entry of $\boldsymbol{x}$), $\bar{S}$ the complement of S, $|S|$ the cardinality of S, H_S and $\boldsymbol{x}(S)$ the subdictionary and subvector indexed by S, respectively. Finally, $H^\dagger$ and $\mathrm{span}(H)$ are the pseudo-inverse and the column space of H, respectively. Let $\tilde{\boldsymbol{h}}_{i,S} = \boldsymbol{h}_i - H_S H_S^\dagger \boldsymbol{h}_i$ stand for the orthogonal projection of $\boldsymbol{h}_i$ onto the orthogonal complement $(\mathrm{span}(H_S))^\perp$, which will be simply denoted $\tilde{\boldsymbol{h}}_i$ whenever unambiguous, and $\tilde{\boldsymbol{g}}_i = \tilde{\boldsymbol{h}}_i / \|\tilde{\boldsymbol{h}}_i\|$ denote the normalized projected atom if $\boldsymbol{h}_i \notin \mathrm{span}(H_S)$, *i.e.*, $\tilde{\boldsymbol{h}}_i \neq \boldsymbol{0}$. If $\boldsymbol{h}_i \in \mathrm{span}(H_S)$, it will be convenient to set $\tilde{\boldsymbol{g}}_i = \boldsymbol{0}$.

For any support S, let us call an unconstrained least-squares (ULS) and a nonnegative least-squares (NNLS) solution corresponding to S, any vector $\boldsymbol{x}$ in $\mathbb{R}^n$ or $\mathbb{R}^n_+$, respectively, that minimizes $\|\boldsymbol{y} - H\boldsymbol{x}\|^2$ with the constraint that $\mathrm{supp}(\boldsymbol{x}) \subset S$. Such vectors will be denoted $\hat{\boldsymbol{x}}_S$ and $\hat{\boldsymbol{x}}_S^+$, respectively. The following notations will be also useful:

$$\boldsymbol{r}_S = \boldsymbol{y} - H\hat{\boldsymbol{x}}_S,$$

$$\boldsymbol{r}_S^+ = \boldsymbol{y} - H\hat{\boldsymbol{x}}_S^+.$$

When H_S is full column rank, $\|\boldsymbol{y} - H_S\boldsymbol{z}\|^2$ is a strictly convex function of $\boldsymbol{z} \in \mathbb{R}^{|S|}$, so $\hat{\boldsymbol{x}}_S^+$ and $\hat{\boldsymbol{x}}_S$ are then uniquely defined. Throughout the paper, we will denote by $C \subset S$ the so-called compressed support, defined as the support of $\hat{\boldsymbol{x}}_S^+$. The NNLS optimal solutions can be characterized using the Karush-Kuhn-Tucker (KKT) conditions [22, Chap. 3], which are recalled next for completeness.

Lemma 1 *Consider the NNLS problem related to support S:*

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \|\boldsymbol{y}-H\boldsymbol{x}\|^2 \quad \textit{s.t.} \quad \mathrm{supp}(\boldsymbol{x})\subset S. \tag{2}$$

$\hat{\boldsymbol{x}}_S^+$ is a solution to (2) *if and only if the KKT conditions are satisfied:*

$$\begin{cases} H_C^{\mathrm{t}}(\boldsymbol{y}-H\hat{\boldsymbol{x}}_S^+) &= \mathbf{0} \\ H_{S\backslash C}^{\mathrm{t}}(\boldsymbol{y}-H\hat{\boldsymbol{x}}_S^+) &\leq \mathbf{0} \end{cases} \tag{3}$$

where $C := \mathrm{supp}(\hat{\boldsymbol{x}}_S^+) \subset S$.

Proof: From the definition of C, it is clear that $\hat{\boldsymbol{x}}_S^+(C) > \mathbf{0}$, so the active constraints in (2) are indexed by $\bar{C}$. Let $\boldsymbol{\lambda} \in \mathbb{R}^n$ gather the Lagrange multipliers related to both equality and inequality constraints. The Lagrangian function induced by (2) is defined as $\mathcal{L}(\boldsymbol{x};\boldsymbol{\lambda}) = \|\boldsymbol{y}-H\boldsymbol{x}\|^2 - \boldsymbol{\lambda}^{\mathrm{t}}\boldsymbol{x}$ and the KKT conditions for optimal variables $(\hat{\boldsymbol{x}}_S^+, \hat{\boldsymbol{\lambda}})$ read:

$$\begin{cases} \nabla_{\boldsymbol{x}}\mathcal{L}(\hat{\boldsymbol{x}}_S^+;\hat{\boldsymbol{\lambda}}) = 2H^{\mathrm{t}}(H\hat{\boldsymbol{x}}_S^+ - \boldsymbol{y}) - \hat{\boldsymbol{\lambda}} = \mathbf{0}, \\ \forall i \in S,\ \hat{\boldsymbol{\lambda}}(i)\hat{\boldsymbol{x}}_S^+(i) = 0 \text{ with } \hat{\boldsymbol{x}}_S^+(i) \geq 0,\ \hat{\boldsymbol{\lambda}}(i) \geq 0, \\ \forall i \notin S,\ \hat{\boldsymbol{x}}_S^+(i) = 0. \end{cases} \tag{4}$$

For quadratic programming problems involving positive semidefinite matrices, the KKT conditions are necessary and sufficient conditions of optimality [32, Chap. 16], so $\hat{\boldsymbol{x}}_S^+$ is a solution to (2) *if and only if*

$$\begin{cases} \hat{\boldsymbol{\lambda}} = -2H^{\mathrm{t}}(\boldsymbol{y}-H\hat{\boldsymbol{x}}_S^+) \\ \hat{\boldsymbol{\lambda}}(C) = \mathbf{0} \\ \hat{\boldsymbol{\lambda}}(S\backslash C) \geq \mathbf{0} \end{cases}$$

that is, when (3) is satisfied. ■

Definition 1 *Let us call a positive support related to the full-size NNLS problem*

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \|\boldsymbol{y}-H\boldsymbol{x}\|^2, \tag{5}$$

any index set S such that H_S is full column rank and $\hat{\boldsymbol{x}}_S^+(S) > \mathbf{0}$. By extension, the empty support $S = \emptyset$ will also be considered as a positive support.

Lemma 2 *S is a positive support if and only if H_S is full rank and $\hat{\boldsymbol{x}}_S(S) > \mathbf{0}$. Moreover, when S is a positive support, $\hat{\boldsymbol{x}}_S^+ = \hat{\boldsymbol{x}}_S$, $\boldsymbol{r}_S^+ = \boldsymbol{r}_S$ and $H_S^{\mathrm{t}}\boldsymbol{r}_S = \mathbf{0}$.*

Proof: When S is a positive support, ULS and NNLS solutions $\hat{\boldsymbol{x}}_S$ and $\hat{\boldsymbol{x}}_S^+$ are uniquely defined and coincide. The orthogonality property $H_S^{\mathrm{t}}\boldsymbol{r}_S = \mathbf{0}$ follows from (3). ■

Positive supports will play an important role in our specification of fast non-negative orthogonal greedy algorithms.

B. Non-negative orthogonal greedy algorithms

Let us define the class of *non-negative orthogonal greedy* (NNOG) algorithms, sharing the following general structure. We start from the empty support $S = \emptyset$. At each iteration, an atom is moved from $\bar{S}$ to S. A new NNLS solution $\hat{\boldsymbol{x}}_S^+$ is then computed to optimally adapt the weights to the newly extended support. The algorithm stops when the desired cardinality K is reached or when the norm of the residual cannot decrease anymore. The general structure of NNOG algorithms is given by Algorithm 1. Some aspects will be made clear later, such as the role of the test $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S > 0$ with respect to the decrease of the norm of the residual.

Algorithm 1: General structure of a non-negative orthogonal greedy algorithm to solve (ℓ_0+).

input : $\boldsymbol{y}, H, K$

output: $\boldsymbol{x}$

$\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r}_S \leftarrow \boldsymbol{y}$;

while $|S| < K$ *and* $\max_{i \in \bar{S}} \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S > 0$ **do**

  Select an index $\ell \in \bar{S}$ by a selection rule $\mathcal{S}(\boldsymbol{y}, H, S)$;

  $S \leftarrow S \cup \{\ell\}$;

  $\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}_S^+$;

  $S \leftarrow \mathrm{supp}(\boldsymbol{x})$;

  $\boldsymbol{r}_S = \boldsymbol{y} - H\boldsymbol{x}$;

end

The NNOG class is a direct adaptation of the family of orthogonal greedy algorithms from the unconstrained case to the nonnegative one. At first glance, the two families only differ by the fact that an NNLS solution is computed rather than a ULS one to update the weights at each iteration. However, some important features differ between the two cases, which require non trivial adaptations.

In both cases, the greedy character corresponds to the fact that a unique atom is added to the current support per iteration. However, a distinct feature of NNOG algorithms is that the support size may be smaller than the current iteration index, because some components of $\hat{\boldsymbol{x}}_S^+(S)$ may vanish at each iteration due to the activation of the corresponding nonnegativity constraints. In the unconstrained case, some components of $\hat{\boldsymbol{x}}_S(S)$ may also vanish, but such events are fortuitous and do not need any specific consideration.

In the unconstrained case, $\boldsymbol{r}_S$ is orthogonal to $\mathrm{span}(H_S)$. This geometrical property does not necessarily hold for $\boldsymbol{r}_S^+$ because NNLS is an inequality constrained problem. Fortunately, provided that the indices of

zero components of $\hat{\boldsymbol{x}}_S^+$ are moved to $\bar{S}$, it remains true that $\boldsymbol{y} - H\hat{\boldsymbol{x}}_S^+$ is orthogonal to $\mathrm{span}(H_S)$. This is a direct consequence of the following lemma, which states that $\hat{\boldsymbol{x}}_S^+$ reads as a ULS solution related to the compressed version of support S.

Lemma 3 *For any S, let $C = \mathrm{supp}(\hat{\boldsymbol{x}}_S^+)$ (where neither $\hat{\boldsymbol{x}}_S^+$ nor C are necessarily unique if H_S is not full column rank). Then we have*

$$\hat{\boldsymbol{x}}_S^+ = \hat{\boldsymbol{x}}_C = \hat{\boldsymbol{x}}_C^+. \tag{6}$$

Proof: Using Lemma 1, we have $H_C^{\mathrm{t}}\boldsymbol{r}_S^+ = \boldsymbol{0}$. Since $\boldsymbol{r}_S^+ = \boldsymbol{y} - H\hat{\boldsymbol{x}}_S^+$, we get $H_C^{\mathrm{t}}\boldsymbol{y} = H_C^{\mathrm{t}}H\hat{\boldsymbol{x}}_S^+$. Thus, $\hat{\boldsymbol{x}}_S^+$ is a ULS solution (denoted by $\hat{\boldsymbol{x}}_C$) associated to support C. We have also $\hat{\boldsymbol{x}}_C = \hat{\boldsymbol{x}}_C^+$ since $\hat{\boldsymbol{x}}_C \geq \boldsymbol{0}$. ■

According to the above definition of NNOG algorithms, distinct algorithms can only differ by the selection rule used to select an atom at each iteration. The design of a selection rule corresponds to the definition of a function $\mathcal{S}(\boldsymbol{y}, H, S)$, taking values in $\bar{S}$. It is clear that some indices $\ell \in \bar{S}$ correspond to inappropriate choices, in the sense that their selection would produce $\hat{\boldsymbol{x}}_{S\cup\{\ell\}}^+(\ell) = 0$, and hence a useless iteration, and possibly an early stopping of the algorithm. In contrast, in the unconstrained case, any selection $\ell \in \bar{S}$ yields a decrease of $\|\boldsymbol{y} - H\boldsymbol{x}\|^2$ unless $\boldsymbol{h}_\ell \in \mathrm{span}(H_S)$. The capacity of some selection rules to avoid inappropriate selections is examined in the next two subsections.

Finally, a practically important aspect is the computing cost of NNOG algorithms. It is computationally more demanding to solve an NNLS problem than the corresponding ULS problem, so one must expect a larger computing cost for NNOG algorithms compared to their unconstrained counterparts. However, NNOG algorithms lend themselves to recursive implementations akin to usual orthogonal greedy schemes, as detailed in Section III.

C. Descending atoms and valid selection rules

Definition 2 *For a given support S, let us define the set of indices corresponding to descending atoms as follows:*

$$\mathcal{D}_S = \left\{ i \in \{1, \ldots, n\}, \|\boldsymbol{r}_{S\cup\{i\}}^+\| < \|\boldsymbol{r}_S^+\| \right\}.$$

Obviously, we have $\mathcal{D}_S \subset \bar{S}$. As already mentioned, rules ensuring the selection of a descending atom at any iteration are a natural choice. In what follows, we focus on this family of rules, referred to as *valid* selection rules.

Definition 3 *A valid selection rule is a function $\mathcal{S}(\boldsymbol{y}, H, S)$ that takes its values in $\mathcal{D}_S$.*

The following proposition allows one to check whether an atom is descending.

Proposition 1 *The descending atom condition $i \in \mathcal{D}_S$ is equivalent to*

$$0 < \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S^+. \tag{7}$$

When S is a positive support, it is also equivalent to each condition

$$0 < \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S, \tag{8}$$

$$0 < \tilde{\boldsymbol{g}}_i^{\mathrm{t}} \boldsymbol{r}_S^+, \tag{9}$$

$$0 < \tilde{\boldsymbol{g}}_i^{\mathrm{t}} \boldsymbol{r}_S. \tag{10}$$

When $H_{S\cup\{i\}}$ is full column rank, it is also equivalent to

$$\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(i) > 0. \tag{11}$$

Proof: See Appendix A-A. ■

Let us remark that from the first item of Proposition 1, the selection rule is invoked at Line 3 of Algorithm 1 only if $\mathcal{D}_S \neq \emptyset$, otherwise the stopping condition of Line 2 is activated. Hence, we do not need to define $\mathcal{S}(\boldsymbol{y}, H, S)$ when $\mathcal{D}_S = \emptyset$.

The following three lemmas have interesting consequences for the practical specification of valid NNOG algorithms.

Lemma 4 *If S is a positive support and $i \in \mathcal{D}_S$, then matrix $H_{S\cup\{i\}}$ is full column rank.*

Proof: Assume that S is a positive support, so H_S is full column rank, and $\hat{\boldsymbol{x}}_S^+ = \hat{\boldsymbol{x}}_S$. Let us also assume that $H_{S\cup\{i\}}$ is not full column rank. Then $\boldsymbol{h}_i \in \mathrm{span}(H_S)$, so $\hat{\boldsymbol{x}}_S$ is a ULS solution corresponding to $S \cup \{i\}$, and also an NNLS solution corresponding to $S \cup \{i\}$ since $\hat{\boldsymbol{x}}_S \geq \boldsymbol{0}$. This implies that $i \notin \mathcal{D}_S$. ■

Lemma 5 *After each iteration of an NNOG algorithm relying on a valid selection rule, it holds that the support of the current solution is positive.*

Proof: The proof is immediate by recursive application of Lemmas 3 and 4, starting with the empty support. ■

Let us stress that Lemma 5 refers to NNOG algorithms strictly conforming to the scheme of Algorithm 1 (with an additional restriction to valid selection rules at Line 3). In particular, the *support compression* step performed at Line 6 is necessary to make Lemma 4 applicable. To our best knowledge, such a compression step has not been proposed in any previous contribution about nonnegative greedy schemes[1].

[1]However, this kind of operation was introduced in [33].

According to Lemma 5, the restriction to a valid selection rule implies $\boldsymbol{r}_S^+ = \boldsymbol{r}_S$ at any iteration, which justifies that we have dropped the '+' sign in Algorithm 1. This simplification is adopted in the rest of the paper. Moreover, the termination rule $\max_i \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S \leq 0$ is used at Line 2 since in this case, there are no descending atoms anymore, so the residual cannot decrease by selection of a new atom.

Lemma 6 *NNOG algorithms relying on a valid selection rule terminate after a finite number of iterations.*

Proof: The error norm decreases at each iteration, and there is a finite number of supports with a cardinality not exceeding K, and thus a finite number of solutions to visit. ■

D. Examples of valid selection rules

In what follows, selection rules are denoted $\mathcal{S}(S)$, the dependence on $\boldsymbol{y}$ and H being implicit. Let us introduce three important selection rules by their distinct ways of picking an index in $\mathcal{D}_S$ when $\mathcal{D}_S \neq \emptyset$.

- NNOMP rule [28], [29], [34], [35]:

$$\mathcal{S}_1(S) \in \underset{i \notin S}{\arg\max}\, \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S \tag{12}$$

- Suboptimal NNOLS (SNNOLS, [30]) rule:

$$\mathcal{S}_2(S) \in \underset{i \notin S}{\arg\max}\, \tilde{\boldsymbol{g}}_i^{\mathrm{t}} \boldsymbol{r}_S \tag{13}$$

- NNOLS rule [30]:

$$\mathcal{S}_3(S) \in \underset{i \notin S}{\arg\min}\, \|\boldsymbol{r}_{S \cup \{i\}}^+\|^2 \tag{14}$$

Selection rule (14) is valid by definition. The fact that (12) and (13) are valid is deduced from recursive application of Proposition 1 and Lemma 5. As regards the latter rules, $\boldsymbol{r}_S$ is the current residual vector, *i.e.*, a readily available quantity. On the other hand, projected atoms $\tilde{\boldsymbol{g}}_i$ enter rule (13), so we can expect the computing cost of (13) to be larger than that of (12). Rule (14) needs the solution of NNLS problems on supports $S \cup \{i\}$, which is even more demanding.

Note that [36] introduced another version of non-negative OMP named NN-OMP in which the selection rule is that of OMP. Clearly, this version does not rely on a valid selection rule. On the other hand, it is unclear whether the FNNOMP and FNNOLS algorithms proposed in [29], [30] rely on a valid selection rule. They will therefore not be further analyzed.

The following proposition makes it possible to compare the three rules (12)-(14) by relating them to the minimization of a residual norm.

Proposition 2 *Rules* (12)-(14) *are equivalent to*

$$\mathcal{S}_j(S) \in \underset{i \in \mathcal{D}_S}{\arg\min}\, \mu_j(S,i), \tag{15}$$

where μ_j *are specific to each rule:*

$$\mu_1(S,i) = \min_{v} \|\boldsymbol{y} - H_S \boldsymbol{x}_S - \boldsymbol{h}_i v\|^2, \tag{16}$$

$$\mu_2(S,i) = \min_{\boldsymbol{u},v} \|\boldsymbol{y} - H_S \boldsymbol{u} - \boldsymbol{h}_i v\|^2, \tag{17}$$

$$\mu_3(S,i) = \min_{\boldsymbol{u}\geq \boldsymbol{0}, v\geq 0} \|\boldsymbol{y} - H_S \boldsymbol{u} - \boldsymbol{h}_i v\|^2. \tag{18}$$

Proof: Let us first emphasize that because (12)-(14) are valid selection rules, $i \notin S$ in (12)-(14) was replaced by $i \in \mathcal{D}_S$ in (15). Clearly, (15) with (18) simply duplicate (14). The link between

$$\min_{v\geq 0} \|\boldsymbol{y} - H_S \boldsymbol{x}_S - \boldsymbol{h}_i v\|^2 \tag{19}$$

and (12) is obtained using identity (1) with $\boldsymbol{r} = \boldsymbol{r}_S$ and $\boldsymbol{h} = \boldsymbol{h}_i$. Likewise, the link between (13) and

$$\min_{\boldsymbol{u},v\geq 0} \|\boldsymbol{y} - H_S \boldsymbol{u} - \boldsymbol{h}_i v\|^2 = \min_{v\geq 0} \|\boldsymbol{r}_S - \tilde{\boldsymbol{h}}_i v\|^2 \tag{20}$$

can be shown by applying (1) with $\boldsymbol{r} = \boldsymbol{r}_S$ and $\boldsymbol{h} = \tilde{\boldsymbol{g}}_i$, since $\tilde{\boldsymbol{g}}_i$ is the normalized version of $\tilde{\boldsymbol{h}}_i$. (20) follows from explicit minimization with respect to $\boldsymbol{u}$ and from the fact that $\boldsymbol{r}_S$ and $\tilde{\boldsymbol{h}}_i$ read as the orthogonal projections of $\boldsymbol{y}$ and $\boldsymbol{h}_i$ onto $(\mathrm{span}(H_S))^\perp$. Finally, the positivity constraint on v in (19)-(20) turns out to be inactive: the minimum error norm in (19)-(20) cannot be equal to $\|\boldsymbol{r}_S\|$ because $i \in \mathcal{D}_S$, see Proposition 1. Thus, (19)-(20) identify with (16)-(17). ■

For $j \in \{1,2,3\}$, an alternate way of viewing the descending character of each rule consists in noticing that $\|\boldsymbol{r}_{S\cup\{i\}}\|^2 \leq \mu_j(S,i) < \|\boldsymbol{r}_S\|^2$ for all $i \in \mathcal{D}_S$. It is interesting to see that, by restricting the selection to the set of descending atoms, the selection rules of NNOMP and SNNOLS rely on the same criteria (16)-(17) as those of OMP and OLS, respectively.

E. Valid non-negative orthogonal greedy algorithms

We are naturally led to formally define the class of *valid* NNOG algorithms, with the general structure given in Algorithm 1, when the selection rule $\mathcal{S}$ is valid.

As already mentioned, rules (12)-(14) have already been introduced in existing works devoted to problem (ℓ_0+) up to the support compression step. The fact that the notion of validity is restricted to algorithms performing support compression deserves to be commented. Given Proposition 1, condition $\boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S^+ > 0$ still characterizes descending atoms without support compression. However, Lemma 4 is no more valid then, so H_S may become rank deficient at any subsequent iteration, $\hat{\boldsymbol{x}}_S^+$ not being well

defined anymore. Support compression is also necessary to ensure that condition $\tilde{\boldsymbol{g}}_i^{\mathrm{t}} \boldsymbol{r}_S^+ > 0$ corresponds to descending atoms (according to Proposition 1). As a consequence, the SNNOLS algorithm of [30] might not necessarily pick descending atoms without support compression. Finally, support compression is also a mandatory step to maintain the current support S in coherence with its definition.

III. Active-set NNLS algorithms and recursivity

Let us consider the NNLS problem related to support S in (2). When matrix H_S is full column rank, it is a special case of a strictly convex quadratic program. The successive resolution of possibly many NNLS problems for nested supports S is a basic ingredient of NNOG algorithms. NNLS problems do not admit closed-form solutions in general. However, *active-set* NNLS (AS-NNLS) algorithms are well-known schemes that solve NNLS problems in a finite number of iterations [22], [37]. Moreover, the support of the solution is positive in the sense of Definition 1, and its computation can be efficiently accelerated using an appropriate recursive scheme.

This section first contains a short reminder on active-set NNLS algorithms and on their efficient implementation. Then, we show how to preserve computational efficiency when NNLS subproblems are solved within an NNOG scheme. We also analyze the similarity between the structures of AS-NNLS algorithm and NNOMP, already pointed out in [27], [38].

A. Fast active-set algorithms

Among many numerical methods for solving (2), AS-NNLS algorithms correspond to greedy schemes, since the solution is found by incremental modifications of its support $V \subset S$, the *active set* being defined as the complementary set $S \backslash V$ (by reference to the active constraints). Such an incremental structure is an essential element to obtain practically fast implementations [21], [22], [32]. Whenever V is modified, the corresponding ULS solution $\hat{\boldsymbol{x}}_V$ is updated. Each iteration requires at least one ULS solution of the *selection* type, *i.e.*, $H_{V\cup\{\ell\}}^{\dagger}\boldsymbol{y}$. Some iterations also need to compute *deselection* type ULS solutions $H_{V\backslash\{\ell\}}^{\dagger}\boldsymbol{y}$. For the sake of computational efficiency, it is crucial to compute both types of solutions recursively, given that $\hat{\boldsymbol{x}}_V$ is already available. In this respect, the situation is identical to that of bi-directional greedy algorithms in the unconstrained case (*e.g.,* Bayesian OMP [39], SBR [40] or FoBa [41]), also called stepwise algorithms [3, Chapter 3], for which selections and deselections are implemented recursively. Fast recursive implementations require specific computations and storage of quantities related to the Gram matrix $G_V = H_V^{\mathrm{t}} H_V$. Efficient selection steps can be obtained using QR or Cholesky matrix factorizations applied to G_V, or the Matrix Inversion Lemma (MIL), with roughly

the same cost [7]. MIL consists in storing and updating the inverse of G_V. It appears to be the cheapest concerning deselections, so our default choice here is based on the MIL.

B. Warm start and full recursivity

Since NNOG algorithms are based on iterated calls to an active-set scheme, we must carefully consider the way we initialize the latter. The starting point of AS-NNLS is usually defined as the zero solution (associated with the empty support). This is the case in the Lawson-Hanson algorithm, which is the reference AS-NNLS scheme [22]. In [32, Chap. 16], Nocedal and Wright proposed an AS-NNLS algorithm with any feasible vector as a possible starting point. Algorithm 2 is a generalized version of the Lawson-Hanson algorithm to address the NNLS problem related to an augmented target set $T \supset S$, where the initial point is not restricted to be the zero vector (the rest of the scheme being unaltered). Specifically, the initial point is set as the ULS solution $\hat{\boldsymbol{x}}_S \geq \mathbf{0}$, S being a positive support. For this specific initial point, it can easily be checked that Algorithm 2 identifies with Nocedal and Wright's scheme.

At any iteration of a valid NNOG algorithm, the current solution $\hat{\boldsymbol{x}}_S^+$ can be used as the initial point to compute $\hat{\boldsymbol{x}}_{S\cup\{\ell\}}^+$ using Algorithm 2 with $T = S \cup \{\ell\}$. In this way, the initial support is $V = S$, so $T\backslash V = \{\ell\}$ and the first iteration of AS-NNLS begins by selecting ℓ. In practice, the NNLS algorithm is expected to terminate after a single iteration (if no deselection is performed in the second part of it), or otherwise after very few iterations. Algorithm 3 is a global view of the active-set implementation of NNOG obtained by integrating the calls to the AS-NNLS solver (Algorithm 2) in the NNOG framework (Algorithm 1). Whenever a new atom ℓ is selected, AS-NNLS starts by computing $\hat{\boldsymbol{x}}_{S\cup\{\ell\}}$. If $\hat{\boldsymbol{x}}_{S\cup\{\ell\}} \geq \mathbf{0}$, then $\hat{\boldsymbol{x}}_{S\cup\{\ell\}}^+ = \hat{\boldsymbol{x}}_{S\cup\{\ell\}}$ and AS-NNLS stops after one support change. Otherwise, AS-NNLS deselects at least one atom from $S \cup \{\ell\}$ (Algorithm 2, Line 7) and then alternates between atom selections and series of deselections. This mechanism is illustrated by a simple example in the next subsection.

A reduced number of iterations is obtained because we use Algorithm 2 with a warm start. To further improve the overall numerical efficiency, we also need to reduce the computing cost of the ULS solution at Lines 5 and 11. These are selection and deselection-type ULS problems, respectively, that can be solved recursively provided that the inverse of the Gram matrix G_V be both an input and an output quantity of the NNLS algorithm. In Appendix B, Algorithm 4 is a pseudo-code to implement ULS updates in the forward ($V \leftarrow V \cup \{\ell\}$) and backward scenarios ($V \leftarrow V\backslash\{\ell\}$). This implementation enables us to obtain a fully recursive implementation of AS-NNLS as well by updating the Gram matrix inverse at each call to ULS in Algorithm 2 (Lines 4-5 and 10-11).

Algorithm 2: Active-set algorithm to solve the NNLS problem related to T, starting from a positive support S.

input : $\boldsymbol{y}$, H, target set T, initial support $S \subset T$, $\hat{\boldsymbol{x}}_S$

output: $V := \mathrm{supp}(\hat{\boldsymbol{x}}_T^+)$, $\hat{\boldsymbol{x}}_T^+ := \hat{\boldsymbol{x}}_V$

$\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}_S$; $V \leftarrow S$;

while $\max\{\boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_V,\, i \in T \backslash V\} > 0$ **do**

  $\ell^+ \leftarrow \arg\max\{\boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_V,\, i \in T \backslash V\}$;

  $V \leftarrow V \cup \{\ell^+\}$;

  Update $\hat{\boldsymbol{x}}_V$ (call Algorithm 4);

  while $\min(\hat{\boldsymbol{x}}_V) < 0$ **do**

    $\ell^- \in \underset{\{i \in V : \hat{\boldsymbol{x}}_V(i) < 0\}}{\arg\min} \boldsymbol{x}(i)/(\boldsymbol{x}(i) - \hat{\boldsymbol{x}}_V(i))$;

    $\alpha \leftarrow \boldsymbol{x}(\ell^-)/(\boldsymbol{x}(\ell^-) - \hat{\boldsymbol{x}}_V(\ell^-))$;

    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha(\hat{\boldsymbol{x}}_V - \boldsymbol{x})$;

    $V \leftarrow V \setminus \{\ell^-\}$;

    Update $\hat{\boldsymbol{x}}_V$ (call Algorithm 4);

  end

  $\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}_V$;

  $\boldsymbol{r}_V \leftarrow \boldsymbol{y} - H\boldsymbol{x}$;

end

Algorithm 3: NNOG with active-set implementation.

input : $\boldsymbol{y}, H, K$

output: $\boldsymbol{x} := \hat{\boldsymbol{x}}_S^+$ (with S a positive support)

$\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r}_S \leftarrow \boldsymbol{y}$;

while $|S| < K$ *and* $\max_{i \in \bar{S}} \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S > 0$ **do**

  Select an index $\ell \in \bar{S}$ by a selection rule $\mathcal{S}(\boldsymbol{y}, H, S)$;

  Call $[C, \boldsymbol{x}] = \texttt{AS\_NNLS}(\boldsymbol{y}, H, S \cup \{\ell\}, S, \boldsymbol{x})$;

  $S \leftarrow C$;

  $\boldsymbol{r}_S = \boldsymbol{y} - H\boldsymbol{x}$;

end

C. Step-by-step illustration of NNOG

Fig. 1 displays a schematic step-by-step illustration of NNOG. NNOG iterates S_k are represented with bullets. NNOG starts with the empty support. In this example, it turns out that during the first three iterations, NNOG yields a positive support $S_{k-1} \cup \{\ell\}$ ($\hat{\boldsymbol{x}}_{S_{k-1}\cup\{\ell\}} \geq \mathbf{0}$), hence $S_k \leftarrow S_{k-1} \cup \{\ell\}$. Therefore, $S_1 \subset S_2 \subset S_3$ are of cardinalities 1, 2 and 3, respectively. At iteration 4, $S_3 \cup \{\ell\}$ is not positive so AS-NNLS performs two support changes, namely the selection of ℓ and a deselection. The next NNOG iterate reads $S_4 \leftarrow S_3 \cup \{\ell\}\backslash\{\ell_1\}$. Iteration 5 is more tricky (and unlikely). Here again, $S_4 \cup \{\ell\}$ is not a positive support. The first deselection does not yield a positive support either, so another deselection is carried out, yielding $V \leftarrow S_4 \cup \{\ell\}\backslash\{\ell_1, \ell_2\}$, V being a positive support. However, the stopping condition of AS-NNLS (Line 2 of Algorithm 2, with $T \leftarrow S_4 \cup \{\ell\}$) is not met since $\boldsymbol{h}_{\ell_1}$ is a descending atom. Therefore, ℓ_1 is reselected: $V \leftarrow V \cup \{\ell_1\}$. Since V is now a positive support and there are no descending atoms anymore in $T\backslash V$, the convergence of AS-NNLS is reached. In the last two NNOG iterations, $S_{k-1} \cup \{\ell\}$ are positive supports, so single selection moves are done within AS-NNLS.

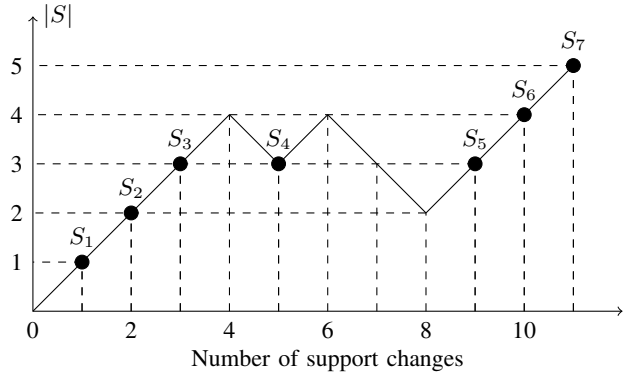


Fig. 1. Step-by-step illustration of NNOG after each change of support. Bullets represent supports corresponding to the first seven NNOG iterates whereas other intermediate supports found during the calls to AS-NNLS are represented without bullets.

D. Connection between AS-NNLS and NNOMP

In Algorithm 3, the structure of the NNOG main iteration consists of one atom selection followed by a variable number of updates (selections or deselections) of the support S when the ULS solution $\hat{\boldsymbol{x}}_S$ has some negative entries. The first of these updates is necessarily an atom deselection. Interestingly, [27] and [38] pointed out that the AS-NNLS algorithm initialized with the zero vector, has a structure similar to NNOMP: each main NNLS iteration consists of an atom selection followed by a series of atom deselections (respectively, Line 3 and Lines 6-12 of Algorithm 2). This connection led Peharz and

Pernkopf to propose AS-NNLS with an early stopping rule $|S| = K$ as a stand-alone NNOG algorithm (called Sparse NNLS in [27]). Given the strong similarity of Sparse NNLS and NNOMP, an interesting question is to determine whether their iterates always coincide. It turns out that this is not always true. However, as long as Sparse NNLS performs only simple support changes, both algorithms yield the same iterates, according to the following proposition.

Proposition 3 *Let us consider Sparse NNLS, i.e., Algorithm 2 initialized with the empty support together with the early stopping rule $|S| = K$. In any case where no iteration produces two or more successive removals in Lines 6–12, the output of Sparse NNLS identifies with that of NNOMP (i.e., Algorithm 1 with rule $\mathcal{S}_1$ in (12)).*

Proof: See Appendix A-B. ■

IV. Acceleration of NNOG algorithms

The NNOMP, SNNOLS and NNOLS selection rules (12)-(14) all read as the optimization of a criterion with respect to the candidate index $i \notin S$. Since all three ensure the selection of an atom in $\mathcal{D}_S$, an obvious acceleration of SNNOLS and NNOLS consists of pre-selecting the descending atoms according to $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S > 0$ (see Proposition 1) to carry out the optimization tasks (13) and (14) over $i \in \mathcal{D}_S$ only. This operation is referred to as *type-I pruning* of the dictionary[2]. Testing the sign of $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S$ requires $\mathcal{O}(m)$ operations. This is much less than the (recursive) computation of the criteria $\tilde{\boldsymbol{g}}_i^{\mathrm{t}}\boldsymbol{r}_S$ and $\|\boldsymbol{r}^+_{S\cup\{i\}}\|^2$, which costs at least $\mathcal{O}(|S|^2 + km)$ operations.

A. Atom selection

The atom selection step of both NNOMP and SNNOLS can be efficiently implemented using vectorized computations, which allow one to benefit from the inherent parallelism of SIMD processors. The NNOMP case is the simplest, since $H_{\bar{S}}^{\mathrm{t}}\boldsymbol{r}_S$ directly yields the expected set of inner products $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S$. Indeed, the selection steps of OMP and NNOMP are both based on the minimization of (16), so they share the same possibilities of parallel computations. Likewise, OLS and SNNOLS being both based on the minimization of (17), they share the same possibilities of parallel implementation. In coherence with our choice of recursive implementation of ULS solutions (see Appendix B), we have adopted a MIL based solution to solve (17) in a vectorized way (see Matlab code in supplementary material).

[2]Contrary to screening techniques, see *e.g.,* [42], note that the atoms indexed by $i \notin \mathcal{D}_S$ are pruned from the dictionary for the current NNOG iteration only, but they are considered again in further iterations.

In contrast, the NNOLS selection rule (14) does not lend itself to fully vectorized computations, since we have as many NNLS subproblems to solve as candidate atoms, with a variable number of subiterations of AS-NNLS for each of them. Fortunately, the structure of the NNOLS rule can be made closer to that of SNNOLS, with the benefit of vectorized computations for the largest part. The key point is that for each candidate atom, the initial step of AS-NNLS corresponding to Lines 1-5 of Algorithm 2 yields the same unconstrained minimizer $\hat{\boldsymbol{x}}_{S\cup\{i\}}$ as the one involved in the SNNOLS rule (17). Hence, these vectors can be obtained using vectorized computations that exactly identify to the main step of SNNOLS atom selection. The extra computations induced by NNOLS reside in the additional AS-NNLS iterations required for each non-positive support $S \cup \{i\}$. According to our empirical tests, only a small minority of atoms needs more than one iteration. Moreover, a lot of them can be pruned without actually performing any additional AS-NNLS iterations. Let us denote by $e_{\rm opt}$ the smallest residual error produced by atoms i for which $S \cup \{i\}$ is a positive support. Since $\|\boldsymbol{r}^+_{S\cup\{i\}}\| \geq \|\boldsymbol{r}_{S\cup\{i\}}\|$ for all i, one can immediately ignore the atoms i for which $\|\boldsymbol{r}_{S\cup\{i\}}\| \geq e_{\rm opt}$. Moreover, since we sequentially visit the remaining ones, the threshold $e_{\rm opt}$ can be gradually lowered whenever a new atom is found to improve the smallest current residual error. This operation called *type-II pruning* is specific to NNOLS implementation.

B. Coefficient update

Once an atom i is selected, NNOG algorithms need to update the coefficients by solving an NNLS subproblem (Algorithm 1, Line 5). However, in the case of NNOLS, the update is already being performed in the selection step. In the case of NNOMP, one needs to call the AS-NNLS algorithm (Algorithm 2) from the initial set S to the target set $S \cup \{\ell\}$. For SNNOLS, a call to Algorithm 2 is needed only when $S \cup \{\ell\}$ is not positive.

C. Software

A Matlab implementation of the acceleration strategies detailed above is provided as supplementary material. This software contains a fully recursive, vectorized version of NNOMP, NNOLS and SNNOLS together with test programs.

V. Simulated sparse deconvolution

A. Data generation

In order to assess the behavior of NNOG algorithms, we consider a convolution problem with a Gaussian kernel $\boldsymbol{h}$ of standard deviation σ. $\boldsymbol{h}$ is approximated by a finite impulse response of length 6σ by thresholding the smallest values. The corresponding normalized dictionary H is a Toeplitz matrix.

Simulated data are generated according to $\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{n}$ where $\boldsymbol{x}^*$ and $\boldsymbol{n}$ stand for the ground truth and white Gaussian noise, respectively. The support S^* of $\boldsymbol{x}^*$, of cardinality K, is randomly generated with a uniform distribution whereas the non-zero coefficients of $\boldsymbol{x}^*$ are either set to a positive constant or randomly generated from an i.i.d. folded normal distribution. The signal-to-noise ratio is defined by $\text{SNR} = 10\log_{10}\left(P_{H\boldsymbol{x}^*}/P_{\boldsymbol{n}}\right)$ where $P_{H\boldsymbol{x}^*} = \left\|H\boldsymbol{x}^*\right\|^2/m$ is the average power of the noise-free data and $P_{\boldsymbol{n}}$ is the noise variance. The results presented below are achieved on a macOS X system with 16 GB RAM and Intel Core i7 processor at 2.7 GHz.

B. Validation of NNOG accelerations

We generate a convolution dictionary of size 1200×1140 with $\sigma = 10$. For each value of $K \in \{20, 40, 60, 80\}$, 200 trials are carried out in which the support S^* is generated. The non-zero coefficients of $\boldsymbol{x}^*$ are set to 1 and the SNR is set to 30 dB. OMP, OLS, NNOMP, SNNOLS and NNOLS are run until the support cardinality equals K. Note that NNOG algorithms may need more than K iterations because of support compression. The following quantities are computed and averaged out over the number of trials:

- *Acceleration gain*: CPU time ratio between the canonical and accelerated implementations of NNOG algorithms.
- *Non-negativity loss*: CPU time ratio between accelerated implementation of NNOG algorithms and the corresponding unconstrained versions.
- *Iterations:* average number of iterations of NNOG algorithms needed to yield a support of cardinality K. This number is larger than K when support compression occurs.

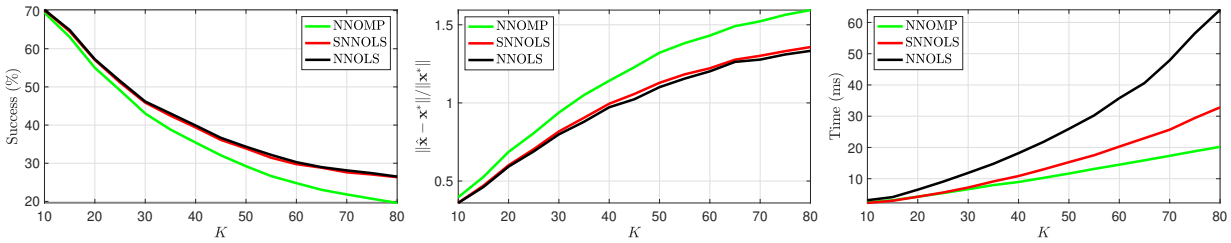
Before going further, let us clarify that the so-called "canonical implementations" refer to the following settings. The AS-NNLS algorithm is called from scratch from the initial zero solution. Moreover, obvious accelerations are taken into account such as type-I pruning, computation of the ULS solutions for candidate supports $S \cup \{i\}$ so as to avoid calling AS-NNLS when the augmented support is positive, and recursive computation of the AS-NNLS iterates. On the other hand, advanced accelerations such as type-II pruning and warm start initialization are not included. Support compression is included in such a way that both canonical and accelerated versions of a given NNOG algorithm yield the same iterates.

The scores can be found in Table I. Accelerated implementations yield a gain in time that increases with K. Since NNOLS needs to solve many NNLS subproblems per iteration, the gain is much larger. The time gain is intermediate for NNOMP. We further notice that using accelerated implementations, the cost of NNOMP and SNNOLS becomes comparable with that of OMP and OLS, respectively, the non-negativity loss remaining below 1.3. Regarding NNOLS *vs* OLS, the non-negativity loss remains

TABLE I

ACCELERATION GAIN OF NNOG ALGORITHMS FOR A SPARSE DECONVOLUTION PROBLEM WITH GAUSSIAN KERNEL ($\sigma = 10$). MEAN OVER 200 TRIALS. THE DICTIONARY SIZE IS 1200×1140.

K	Acceleration gain			Non-negativity loss			Iterations		
	NNOMP	SNNOLS	NNOLS	NNOMP	SNNOLS	NNOLS	NNOMP	SNNOLS	NNOLS
20	2.7	1.2	9	1.2	1.0	1.5	20	22	21
40	4.5	2.1	59	1.2	1.2	1.9	41	51	43
60	6.4	3.4	120	1.1	1.2	2.1	65	97	73
80	8.5	4.0	128	1.1	1.3	4.1	95	152	121



(a) Support recovery (b) Coefficient accuracy (c) CPU time

Fig. 2. Average accuracy and CPU time of NNOMP, SNNOLS, NNOLS over 200 trials in a simulated sparse deconvolution problem with Gaussian kernel ($\sigma = 5$). The SNR is set to 18 dB. The dictionary size is 1200×1170.

lower than 5 in these simulations. At last, the fact that the number of iterations is often larger than K reveals that support compression is happening quite often.

C. Comparison of NNOG algorithms

Algorithms NNOMP, SNNOLS and NNOLS are further compared in terms of average CPU time and recovery accuracy. Recovery accuracy is quantified by two factors:

- *Support recovery:* ratio of true positives to K;
- *Coefficient accuracy:* relative error for the recovered coefficients ($\|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\| \,/\, \|\boldsymbol{x}^*\|$).

The curves shown in Fig. 2 are obtained for a convolution dictionary of size 1200×1170 ($\sigma = 5$) and various settings of $K \in [10, 80]$. For each K, 200 trials are carried out in which the support S^* is drawn according to the uniform distribution and the non-zero coefficients of $\boldsymbol{x}^*$ are drawn from a folded normal distribution. The SNR is set to 18 dB. NNOMP turns out to be the fastest but the least accurate. NNOLS slightly outperforms SNNOLS at the price of a doubled computing time. In many other sparse deconvolution tests we have done, SNNOLS and NNOLS yield similar accuracies.

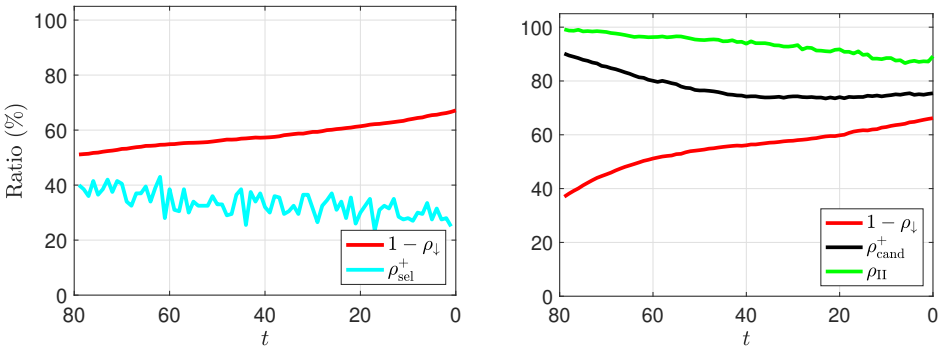
D. Computation burden of SNNOLS and NNOLS

Hereafter, the computation burden of SNNOLS and NNOLS is assessed more thoroughly so as to evaluate the accelerations proposed in Section IV. Four indicators are computed at each iteration:

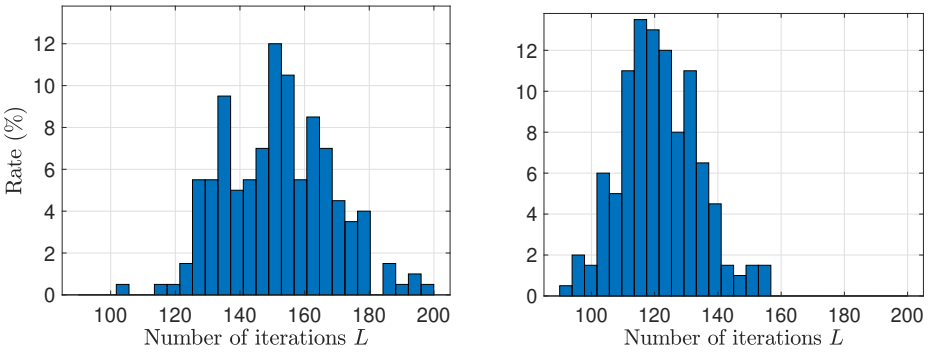
1) $\rho_{\downarrow} = |\mathcal{D}_S|/|\bar{S}|$: rate of descending atoms. The rate of discarded atoms after type-I pruning reads $1 - \rho_{\downarrow}$.
2) ρ^{+}_{cand}: rate of descending candidate atoms for which $S \cup \{i\}$ is a positive support.
3) ρ_{II}: rate of candidate atoms discarded by type-II pruning among all atoms i for which $S \cup \{i\}$ is not a positive support.
4) ρ^{+}_{sel}: rate of selected atoms yielding a positive support $S \cup \{\ell\}$.

The computational cost of an NNOG iteration is closely related to the values of these ratios. Indeed, large scores indicate that the cost of testing candidate atoms is dramatically reduced. Specifically, $\rho_{\downarrow}$ is the rate of candidates that are truly considered in the selection rule (15) of NNOG algorithms. Both ratios ρ^{+}_{cand} and ρ_{II} quantify the computational burden of the NNOLS selection step: large values of ρ^{+}_{cand} and ρ_{II} indicate that AS-NNLS has to be run for a few candidate atoms only, since other atoms are either pruned or yield a non-negative ULS solution. ρ^{+}_{sel} is defined similar to ρ^{+}_{cand}. However, ρ^{+}_{sel} applies to the selected atoms, which brings information on the computational burden of the coefficient update stage of SNNOLS.

Using the dictionary of size 1200×1140 and the settings $K = 80$ and SNR = 30 dB of Subsection V-B, the computational burden of SNNOLS and NNOLS is assessed in Fig. 3. It is noticeable that the number of iterations L required to reach a support of cardinality K is larger than K because of support compression. Specifically, the histograms of Fig. 3(b) show that on average, L is larger than K for NNOLS and even larger for SNNOLS (the average values are given in the last columns of Tab. I). This is consistent with the fact that the NNOLS selection rule is more involved but more reliable. Moreover, the standard deviation of L corresponding to the histograms of Fig. 3(b) is 17 and 12 for SNNOLS and NNOLS, respectively, which indicates that the size of the support found after k iterations may significantly vary between trials. In order to get meaningful evaluations, we choose to compute the average values of each indicator over the last t iterations, with $t \in \{0, \ldots, K-1\}$. When $t = 0$, only the last iteration is taken into account, so the current support is of size K. For larger values of t, the supports found during the last t iterations have varying sizes but the averaging operation remains meaningful, especially for the last iterations, which are the most costly. The curves displaying the average of each indicator over 200 trials and over the last t iterations are shown in Fig. 3(a). One can observe from the curve $(1-\rho_{\downarrow})$ that the rate of non-descending atoms gradually increases for both SNNOLS and NNOLS. Therefore, type-I pruning is more effective at



(a) Complexity indicators of SNNOLS (left) and NNOLS (right)



(b) Histograms of SNNOLS (left) and NNOLS (right)

Fig. 3. Complexity analysis of SNNOLS and NNOLS for a sparse deconvolution problem with Gaussian kernel ($\sigma = 10$) and a SNR of 30 dB. The dictionary size is 1200×1140, $K = 80$ and 200 trials are performed. (a) Evolution of complexity factors during the last t iterations (from $L - t$ to L). $(1 - \rho_{\downarrow})$: atoms discarded by type-I pruning; ρ^{+}_{sel}: positive supports found by SNNOLS; ρ^{+}_{cand}: candidate atoms yielding a positive support in NNOLS; ρ_{II}: atoms discarded by type-II pruning in NNOLS. (b) Histogram of the average number of iterations L required to reach a support of cardinality $K = 80$.

late iterations, where half of the atoms are discarded.

The efficiency of NNOLS implementation is measured by ρ^{+}_{cand} and ρ_{II} in Fig. 3(a). Large values of ρ^{+}_{cand} and ρ_{II} are obtained, which indicates that the computing cost of NNOLS is dramatically reduced. The score of ρ^{+}_{cand} always remains above 70%. This means that the computation of the NNLS solutions reduces to a single ULS update for more than 70% of candidate atoms. Moreover, many of the remaining atoms can be discarded due to type-II pruning. For instance, in the last iterations, around 25% of descending atoms do not yield a positive support and type-II pruning eliminates 90% of them. As a result, only 3% of descending atoms require to proceed with a complete NNLS solving. Let us stress that the extra cost of NNOLS as compared with SNNOLS essentially comes from the number of atoms related to a complete NNLS solving, so it directly depends on the efficiency of type-II pruning.

Regarding SNNOLS, the ratio ρ^{+}_{sel} remains above 20%, which indicates that the computation of the

SNNOLS iterate reduces to a single ULS update for at least 20% of the trials. Generally speaking, the slight decrease of ρ^{+}_{sel}, ρ^{+}_{cand} and ρ_{II} shows that both SNNOLS and NNOLS call AS-NNLS more often at the late iterations. A possible reason for this behavior might be that the average correlation between selected atoms increases with the cardinality of the support.

VI. Sparse decomposition of NIR spectra

A. Real data and generated dictionary

More than 300 wood samples with different compositions (raw wood, plywood, particle boards, MDF-HDF) and finitions (raw, painted, varnished) were collected on a wood waste park and scanned using a Nicolet 8700 FTIR spectrometer. The resulting reflectance spectra are composed of 1647 wavenumbers covering the NIR range 3600–10000 cm^{-1} (which corresponds to wavelengths in the interval $[1, 2.8]$ μm). The aim is to identify a subset of wavelengths that are informative for detecting the so-called non recyclable samples (*i.e.*, MDF-HDF) [43], [44]. As the final objective is to design an industrial (fast) sorting system, the number of selected wavelengths has to be as small as possible (typically between 16 and 32).

Hereafter, we consider the decomposition of 50 NIR spectra, seen as data vectors $\boldsymbol{y}$ of length 1647. Data pre-processing includes baseline removal, offset correction ensuring zero lower bound, and unit energy normalization. To decompose the spectra, we build a dictionary H with Gaussian-shaped columns obtained by discretizing the parameters of a Gaussian function (centers and widths). It is formed by appending the columns of the convolution dictionaries (corresponding to a fixed width σ) used in Section V for 60 equally spaced values of $\sigma \in [10, 600]$ cm^{-1}. The generated dictionary is composed of 2998 atoms. Note that the centers of Gaussian atoms of same width σ are sampled with a step equal to σ, whereas the sampling step of the input signals $\boldsymbol{y}$ equals 4 cm^{-1}.

B. Decomposition results

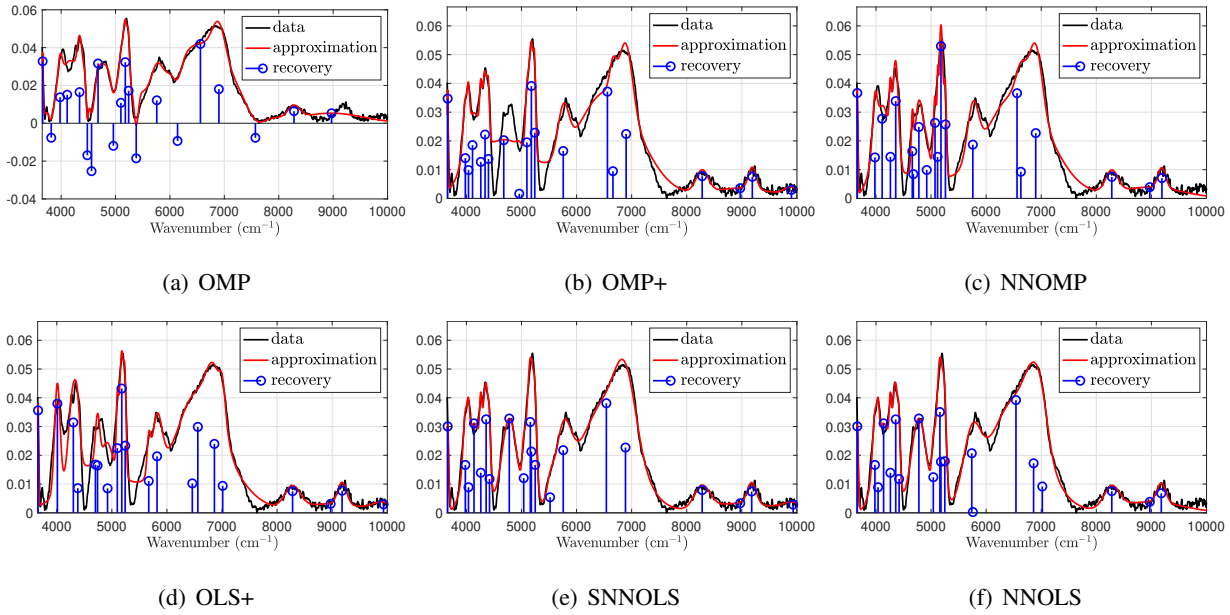
For each spectrum, NNOMP, SNNOLS, NNOLS, Sparse NNLS, OMP, and OLS are run until a support of cardinality $K = 20$ is reached. In order to obtain non-negative coefficients from OMP (OLS), a possibility is to apply a post-processing step (with NNLS) using the solution support yielded by OMP (OLS). The resulting schemes are denoted by OMP+ (OLS+). For each competing algorithm, the CPU time and the normalized approximation error $\|\boldsymbol{y} - H\boldsymbol{x}\|^2/\|\boldsymbol{y}\|^2$, averaged over 50 spectra, are displayed in Table II.

NNOMP and Sparse NNLS have roughly the same cost and performance. Furthermore, SNNOLS and NNOLS yield lower approximation errors as compared to NNOMP and Sparse NNLS. The computing

TABLE II

CPU TIME AND NORMALIZED APPROXIMATION ERROR OF GREEDY ALGORITHMS FOR SPARSE DECOMPOSITION OF NIR SPECTRA. AVERAGE OVER 50 SPECTRA. SYMBOL ** INDICATES THAT THE CONSIDERED ALGORITHM DOES NOT ENFORCE THE NON-NEGATIVE CONSTRAINT.

Algorithm	**OMP	OMP+	NNOMP	Sparse NNLS	**OLS	OLS+	SNNOLS	NNOLS
Time (ms)	23	24	28	25	25	26	30	40
Error $\times 10^{-3}$	4.3	26.9	9.3	9.3	3.2	27.4	4.1	4.1



(a) OMP (b) OMP+ (c) NNOMP

(d) OLS+ (e) SNNOLS (f) NNOLS

Fig. 4. Data approximation and sparse recovery of an NIR spectrum using various algorithms. The Gaussian dictionary contains 2998 atoms and the sparsity level is set to $K = 20$.

time of NNOLS is significantly larger, while SNNOLS has the same performance as NNOLS and a computing time closer to that of NNOMP. On the other hand, the additional cost of OMP+ and OLS+ with respect to OMP and OLS is small, because it amounts to solving a small size NNLS problem (of size $K = 20$). However, OMP+ and OLS+ perform poorly, which indicates the weak capacity of OMP and OLS to reconstruct correct supports from non-negative sparse representations. Finally, the ratios of CPU times of NNOG algorithms *vs* their related unconstrained versions are consistent with the non-negativity losses gathered in Tab. I.

These results are further illustrated for a specific spectrum in Fig. 4, where approximations and sparse recoveries are displayed for competing algorithms. Note that the Sparse NNLS result (not shown) identifies with that of NNOMP. One can see that OMP yields seven negative peaks. Besides, the OMP+ and OLS+

approximations around 4500–7000 cm^{-1} and 5000 cm^{-1}, respectively, are rather poor. SNNOLS and NNOLS outputs almost coincide. They outperform that of NNOMP, in particular, around 4200 and 7000 cm^{-1}.

VII. Conclusion

Until now, greedy algorithms dedicated to non-negative sparse signal reconstruction have been considered as slow schemes, requiring the repeated resolution of constrained least square problems. In order to accelerate the computation, approximate schemes have been proposed [29], [30] at the price of some loss of control on the algorithmic behavior, and possibly degraded performance. Another commonly found option has been to replace the ℓ_0 "norm" by the ℓ_1-norm and to solve the resulting convex programming problem, with a possible loss in terms of performance (see [27] for an interesting case of sparse NMF).

The first contribution of this paper is to provide a unified framework to define non-negative orthogonal greedy algorithms in a formal way, ensuring well-defined iterations. The second and probably most important one in terms of practical impact, is to show that the additional cost of non-negative greedy algorithms to handle the sign constraint can be strongly reduced using three ingredients. The main one is that non-negative greedy algorithms can be made fully recursive. Moreover, several pruning strategies can be combined to reduce the number of tests at the atom selection stage. Finally, the latter step can benefit from vectorized computations. According to our practical tests, we can conclude that the computing time of NNOMP then becomes comparable to that of OMP, so that the additional cost should not prevent users from skipping from OMP to NNOMP in any nonnegative sparse problem. On the other hand, we have obtained a dramatic acceleration of NNOLS compared to the canonical version proposed in [30]. The computing cost of NNOLS remains larger than that of OLS. However, whenever OLS can be used, our exact implementation of NNOLS is a realistic option to handle nonnegativity constraints, given that the potential of performance gain between NNOMP and NNOLS is comparable to the one between OMP and OLS in the unconstrained case. We have also studied SNNOLS, which is a suboptimal version of NNOLS originally introduced in [30]. Our conclusion is that SNNOLS represents a good trade-off between NNOMP and NNOLS: it is structurally simpler than NNOLS and hence significantly faster, with very similar performance in terms of estimation error. Likewise, we have compared NNOMP with Sparse NNLS and concluded that Sparse NNLS is structurally simpler than NNOMP with almost the same performance as NNOMP, and a slightly reduced computing cost.

Our contributions can be extended in several directions. A straightforward generalization can be made to deal with nonnegativity-constrained simultaneous sparse decomposition, which is useful in several applications such as hyperspectral imaging [36], dynamic PET [45], and diffusion MRI [35]. On the other

hand, other greedy algorithms such as CoSaMP [46], BOMP [39] and SBR [40] could also be extended to the non-negative setting using similar principles and using a recursive implementation. Finally, let us mention that [47] establishes the first K-step recovery analysis of NNOMP, NNOLS and SNNOLS under the Mutual Incoherence Property condition.

APPENDIX A
PROOF OF TECHNICAL RESULTS

A. Proof of Proposition 1

Let us first prove that $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+ > 0$ implies $i \in \mathcal{D}_S$. Let $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+ > 0$. According to (1) for $\boldsymbol{r} = \boldsymbol{r}_S^+$, we deduce that $i \in \mathcal{D}_S$ since

$$\|\boldsymbol{r}_{S\cup\{i\}}^+\|^2 \leq \min_{v\geq 0} \left\|\boldsymbol{r}_S^+ - v\boldsymbol{h}_i\right\|^2 < \|\boldsymbol{r}_S^+\|^2.$$

Conversely, let $i \in \mathcal{D}_S$. Define $f(\boldsymbol{z}) = \left\|\boldsymbol{y} - H_{S\cup\{i\}}\boldsymbol{z}\right\|^2$ where $\boldsymbol{z} \in \mathbb{R}^{|S|+1}$. Let us also define the subvectors $\boldsymbol{z}_S := \hat{\boldsymbol{x}}_S^+(S\cup\{i\})$ and $\boldsymbol{z}_{S\cup\{i\}} := \hat{\boldsymbol{x}}_{S\cup\{i\}}^+(S\cup\{i\})$, $\hat{\boldsymbol{x}}_S^+$ and $\hat{\boldsymbol{x}}_{S\cup\{i\}}^+$ being two NNLS solutions related to S and $S\cup\{i\}$. Condition $i \in \mathcal{D}_S$ reads

$$f(\boldsymbol{z}_{S\cup\{i\}}) < f(\boldsymbol{z}_S).$$

Since f is convex, one has

$$(\boldsymbol{z}_{S\cup\{i\}} - \boldsymbol{z}_S)^{\mathrm{t}}\nabla f(\boldsymbol{z}_S) \;\leq\; f(\boldsymbol{z}_{S\cup\{i\}}) - f(\boldsymbol{z}_S) \;<\; 0 \tag{21}$$

where the gradient of f is defined by

$$\nabla f(\boldsymbol{z}_S) = 2H_{S\cup\{i\}}^{\mathrm{t}}(H_{S\cup\{i\}}\boldsymbol{z}_S - \boldsymbol{y}) = -2H_{S\cup\{i\}}^{\mathrm{t}}\boldsymbol{r}_S^+.$$

Denoting by $C := \mathrm{supp}(\hat{\boldsymbol{x}}_S^+)$ the compressed support, we have from Lemma 1 that $H_{S\backslash C}^{\mathrm{t}}\boldsymbol{r}_S^+ \leq \boldsymbol{0}$ and $H_C^{\mathrm{t}}\boldsymbol{r}_S^+ = \boldsymbol{0}$. Since $\hat{\boldsymbol{x}}_S^+$ is supported by C, the latter equality implies that $\boldsymbol{z}_S^{\mathrm{t}}\nabla f(\boldsymbol{z}_S) = 0$. (21) yields $(\boldsymbol{z}_{S\cup\{i\}})^{\mathrm{t}}H_{S\cup\{i\}}^{\mathrm{t}}\boldsymbol{r}_S^+ > 0$, *i.e.*,

$$(\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(S\cup\{i\}))^{\mathrm{t}}H_{S\cup\{i\}}^{\mathrm{t}}\boldsymbol{r}_S^+ \;>\; 0. \tag{22}$$

Since $H_C^{\mathrm{t}}\boldsymbol{r}_S^+ = \boldsymbol{0}$, (22) rereads:

$$(\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(S\backslash C))^{\mathrm{t}}H_{S\backslash C}^{\mathrm{t}}\boldsymbol{r}_S^+ + (\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+)\,\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(i) \;>\; 0 \tag{23}$$

and since $H_{S\backslash C}^{\mathrm{t}}\boldsymbol{r}_S^+ \leq \boldsymbol{0}$ and $\hat{\boldsymbol{x}}_{S\cup\{i\}}^+ \geq \boldsymbol{0}$, (23) implies that

$$(\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+)\,\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(i) \;>\; 0 \tag{24}$$

and thus $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+ > 0$.

Let us now assume that S is a positive support. According to Lemma 2, we have $\hat{\boldsymbol{x}}_S^+ = \hat{\boldsymbol{x}}_S$ and $\boldsymbol{r}_S^+ = \boldsymbol{r}_S$, so (7) and (8) are identical, as well as (9) and (10). To show that (7)-(8) are equivalent to (9)-(10), we first notice that $\boldsymbol{r}_S \in (\mathrm{span}(H_S))^\perp$, thus $\tilde{\boldsymbol{h}}_i^\mathrm{t}\boldsymbol{r}_S = \boldsymbol{h}_i^\mathrm{t}\boldsymbol{r}_S$ since $\tilde{\boldsymbol{h}}_i - \boldsymbol{h}_i \in \mathrm{span}(H_S)$. Therefore, (8) rereads $0 < \tilde{\boldsymbol{h}}_i^\mathrm{t}\boldsymbol{r}_S$, which implies that $\tilde{\boldsymbol{h}}_i \neq \boldsymbol{0}$ and $\tilde{\boldsymbol{g}}_i = \tilde{\boldsymbol{h}}_i / \|\tilde{\boldsymbol{h}}_i\| \neq \boldsymbol{0}$. Hence, (8) rereads $0 < \tilde{\boldsymbol{g}}_i^\mathrm{t}\boldsymbol{r}_S$, which identifies with (10).

Finally, let us show that $i \in \mathcal{D}_S$ is equivalent to condition (11). Consider the function $f(\boldsymbol{z}) = \left\|\boldsymbol{y} - H_{S\cup\{i\}}\boldsymbol{z}\right\|^2$ and the notations $\boldsymbol{z}_S$ and $\boldsymbol{z}_{S\cup\{i\}}$ defined above. Assuming that $H_{S\cup\{i\}}$ is full column rank, we have that f is strictly convex, so f admits a unique minimizer $\boldsymbol{z}_{S\cup\{i\}}$. If $\hat{\boldsymbol{x}}^+_{S\cup\{i\}}(i) > 0$, then $\boldsymbol{z}_{S\cup\{i\}} \neq \boldsymbol{z}_S$ and

$$\|\boldsymbol{r}^+_{S\cup\{i\}}\|^2 = f(\boldsymbol{z}_{S\cup\{i\}}) < \|\boldsymbol{r}_S^+\|^2 = f(\boldsymbol{z}_S),$$

that is, $i \in \mathcal{D}_S$. Conversely, $\hat{\boldsymbol{x}}^+_{S\cup\{i\}}(i) = 0$ implies $\hat{\boldsymbol{x}}^+_{S\cup\{i\}} = \hat{\boldsymbol{x}}_S^+$, hence $\boldsymbol{r}^+_{S\cup\{i\}} = \boldsymbol{r}_S^+$ and $i \notin \mathcal{D}_S$.

B. Proof of Proposition 3

Any iteration of AS-NNLS starts with the addition of a new atom to the current support (Line 4 of Algorithm 2). Then, a variable number of atoms are removed from it one after the other (Lines 6 to 12). Let r denote the number of removals at the current AS-NNLS iteration. Let also $V \subset \{1, \ldots, n\}$ and $\hat{\boldsymbol{x}}_V$ respectively stand for the current support and solution obtained at Lines 4 and 5, and let $V' \subset V$ and $\hat{\boldsymbol{x}}_{V'}$ denote the corresponding quantities after r removals. Let us first show that any AS-NNLS iteration for which $r = 0$ or $r = 1$ yields a solution of the NNLS problem restricted to the support V.

If $\hat{\boldsymbol{x}}_V \geq \boldsymbol{0}$, then $r = 0$, so $V' = V$ and $\hat{\boldsymbol{x}}_{V'} = \hat{\boldsymbol{x}}_V^+ \geq \boldsymbol{0}$. Otherwise, we have $\min(\hat{\boldsymbol{x}}_V) < 0$ and $r > 0$. If $r = 1$, a single index ℓ^- is removed at Lines 6-12, so that $V' = V \setminus \{\ell^-\}$, and $\hat{\boldsymbol{x}}_{V'} \geq \boldsymbol{0}$. Let us remark that we have $\hat{\boldsymbol{x}}_V(\ell^-) < 0$ according to Line 7. Let us then prove that $\hat{\boldsymbol{x}}_{V'} = \hat{\boldsymbol{x}}_V^+$ by showing that KKT conditions are satisfied at $\hat{\boldsymbol{x}}_{V'}$ for the NNLS problem related to support V. Note that the NNLS solution is unique since the supports generated by AS-NNLS are such that H_V is full column rank, as pointed out in subsection III-B. According to Lemma 1, the KKT conditions read:

$$H_{V'}^\mathrm{t}(\boldsymbol{y} - H\hat{\boldsymbol{x}}_{V'}) = \boldsymbol{0}, \tag{25a}$$

$$\boldsymbol{h}_{\ell^-}^\mathrm{t}(\boldsymbol{y} - H\hat{\boldsymbol{x}}_{V'}) \leq 0. \tag{25b}$$

(25a) is obviously satisfied. On the other hand, remark that $\hat{\boldsymbol{x}}_{V'} = \hat{\boldsymbol{x}}^+_{V'}$ since $\hat{\boldsymbol{x}}_{V'} \geq \boldsymbol{0}$ and that according to Proposition 1, $\hat{\boldsymbol{x}}_V(\ell^-) < 0$ implies that $\boldsymbol{h}_{\ell^-}^\mathrm{t}\boldsymbol{r}_{V'} \leq \boldsymbol{0}$, which identifies with (25b). This concludes the proof.

APPENDIX B
RECURSIVE IMPLEMENTATION OF ULS

Algorithm 4 recalls the recursive ULS computation using MIL [21] for selection (forward move $V \leftarrow V \cup \{\ell\}$) and deselection (backward move $V \leftarrow V \backslash \{\ell\}$) operations. The ULS solution $\hat{\boldsymbol{x}} := \hat{\boldsymbol{x}}_V$ is updated. Moreover, $\Theta := (H_V^{\mathrm{t}} H_V)^{-1}$ refers to the inverse of the Gram matrix related to subset V. The Boolean entry `fw` is set to `true` and `false` for selection and deselection updates, respectively. Finally, e^2 stands for the squared residual error $\|\boldsymbol{r}_V\|^2$. All these factors are updated in Algorithm 4. Notation $-j$ refers to all indices except j, and $\boldsymbol{\theta}_j$ stands for the j-th column of Θ.

Algorithm 4: Recursive ULS [21].

Format: ULS$(\boldsymbol{y}, H, V, \mathtt{fw}, \ell, \hat{\boldsymbol{x}}, \Theta, e^2)$

if `fw` **then**

$\quad \boldsymbol{\phi} \leftarrow H_V^{\mathrm{t}} \boldsymbol{h}_\ell$;

$\quad \delta \leftarrow (1 - \boldsymbol{\phi}^{\mathrm{t}} \Theta \boldsymbol{\phi})^{-1}$;

$\quad \beta \leftarrow \boldsymbol{\phi}^{\mathrm{t}} \hat{\boldsymbol{x}}(V) - \boldsymbol{h}_\ell^{\mathrm{t}} \boldsymbol{y}$;

$\quad e^2 \leftarrow e^2 - \delta \beta^2$;

$\quad \hat{\boldsymbol{x}}(V \cup \{\ell\}) \leftarrow \hat{\boldsymbol{x}}(V \cup \{\ell\}) + \delta \beta \begin{bmatrix} \Theta \boldsymbol{\phi} \\ -1 \end{bmatrix}$;

$\quad \Theta \leftarrow \begin{bmatrix} \Theta & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{bmatrix} + \delta \begin{bmatrix} \Theta \boldsymbol{\phi} \\ -1 \end{bmatrix} \begin{bmatrix} \Theta \boldsymbol{\phi} \\ -1 \end{bmatrix}^{\mathrm{t}}$;

$\quad V \leftarrow V \cup \{\ell\}$;

else

$\quad j \leftarrow$ index of ℓ in V ;

$\quad e^2 \leftarrow e^2 + (\hat{\boldsymbol{x}}(\ell))^2 / \boldsymbol{\theta}_j(j)$;

$\quad \hat{\boldsymbol{x}}(V) \leftarrow \hat{\boldsymbol{x}}(V) - \hat{\boldsymbol{x}}(\ell)\, \boldsymbol{\theta}_j / \boldsymbol{\theta}_j(j)$;

$\quad \Theta \leftarrow \Theta(-j, -j) - \boldsymbol{\theta}_j(-j) \boldsymbol{\theta}_j(-j)^{\mathrm{t}} / \boldsymbol{\theta}_j(j)$;

$\quad V \leftarrow V \setminus \{\ell\}$;

end

The calls to Algorithm 4 for updating ULS solutions at Lines 4-5 and 10-11 of Algorithm 2 take the respective forms:

$$\texttt{ULS}(\boldsymbol{y}, H, V, 1, \ell^+, \hat{\boldsymbol{x}}_V, \Theta, \|\boldsymbol{r}_V\|^2),$$

$$\texttt{ULS}(\boldsymbol{y}, H, V, 0, \ell^-, \hat{\boldsymbol{x}}_V, \Theta, \|\boldsymbol{r}_V\|^2).$$

REFERENCES

[1] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition", in *Proc. 27th Asilomar Conf. on Signals, Sys. and Comp.*, Nov. 1993, vol. 1, pp. 40–44.

[2] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification", *Int. J. Control*, vol. 50, no. 5, pp. 1873–1896, Nov. 1989.

[3] A. J. Miller, *Subset selection in regression*, Chapman and Hall, London, UK, 2nd edition, Apr. 2002.

[4] S. F. Cotter, J. Adler, B. D. Rao, and K. Kreutz-Delgado, "Forward sequential algorithms for best basis selection", *IEE Proc. Vision, Image and Signal Processing*, vol. 146, no. 5, pp. 235–244, Oct. 1999.

[5] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach", *IEEE Signal Process. Lett.*, vol. 9, no. 4, pp. 137–140, Apr. 2002.

[6] S. Foucart, "Stability and robustness of weak orthogonal matching pursuits", in *Recent advances in harmonic analysis and applications*, D. Bilyk, L. De Carli, A. Petukhov, A. M. Stokolos, and B. D. Wick, Eds. 2013, vol. 25, pp. 395–405, Springer Proc. in Mathematics & Statistics.

[7] B. L. Sturm and M. G. Christensen, "Comparison of orthogonal matching pursuit implementations", in *Proc. Eur. Sig. Proc. Conf.*, Bucharest, Romania, Aug. 2012, pp. 220–224.

[8] M.-D. Iordache, J. M. Bioucas-Dias, and A. Plaza, "Sparse unmixing of hyperspectral data", *IEEE Trans. Geosci. Remote Sensing*, vol. 49, no. 6, pp. 2014–2039, June 2011.

[9] E. Esser, Y. Lou, and J. Xin, "A method for finding structured sparse solutions to nonnegative least squares problems with applications", *SIAM J. Imaging Sci.*, vol. 6, no. 4, pp. 2010–2046, Oct. 2013.

[10] T. Virtanen, J. F. Gemmeke, and B. Raj, "Active-set Newton algorithm for overcomplete non-negative representations of audio", *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 11, pp. 2277–2289, Nov. 2013.

[11] R. Bro and S. De Jond, "A fast non-negativity-constrained least squares algorithm", *Journal of Chemometrics*, vol. 11, no. 5, pp. 393–401, 1997.

[12] M. Slawski, R. Hussong, A. Tholey, T. Jakoby, B. Gregorius, A. Hildebrandt, and M. Hein, "Isotope pattern deconvolution for peptide mass spectrometry by non-negative least squares/least absolute deviation template matching", *BMC Bioinformatics*, vol. 13, no. 291, pp. 1–18, Nov. 2012.

[13] J. A. Högbom, "Aperture synthesis with a non-regular distribution of interferometer baselines", *Astron. Astrophys. Suppl.*, vol. 15, pp. 417–426, 1974.

[14] S. Petra and C. Schnörr, "Average case recovery analysis of tomographic compressive sensing", *Linear Alg. Appl.*, vol. 441, pp. 168–198, 2014.

[15] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints", *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, 2004.

[16] J. Rapin, J. Bobin, A. Larue, and J.-L. Starck, "Sparse and non-negative BSS for noisy data", *IEEE Trans. Signal Process.*, vol. 61, no. 22, pp. 5620–5632, Nov. 2013.

[17] P. L. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing*, chapter 10, pp. 185–212, Springer-Verlag, New York, 2011.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning *via* the alternating direction method of multipliers", *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, July 2011.

[19] G. Gasso, A. Rakotomamonjy, and S. Canu, "Recovering sparse signals with a certain family of nonconvex penalties and DC programming", *IEEE Trans. Signal Process.*, vol. 57, no. 12, pp. 4686–4698, Dec. 2009.

[20] H. A. Le Thi, B. T. Nguyen Thi, and H. M. Le, "Sparse signal recovery by difference of convex functions algorithms", in *Intelligent Information and Database Systems*, A. Selamat, N. T. Nguyen, and H. Haron, Eds., Berlin, 2013, vol. 7803 of *Lect. Notes Comput. Sci.*, pp. 387–397, Springer Verlag.

[21] A. Björck, *Numerical Methods for Least Squares Problems*, Society for Industrial and Applied Mathematics, Philadephia, PA, Apr. 1996.

[22] C. L. Lawson and R. J. Hanson, *Solving least squares problems*, pp. 149–199, Society for Industrial and Applied Mathematics, 1974.

[23] M. H. Wright, "Interior methods for constrained optimization", *Acta Numerica*, vol. 1, pp. 341–407, Jan. 1992.

[24] F. Benvenuto, R. Zanella, L. Zanni, and M. Bertero, "Nonnegative least-squares image deblurring: Improved gradient projection approaches", *Inverse Probl.*, vol. 26, no. 2, pp. 1–18, Feb. 2010.

[25] M. Slawski and M. Hein, "Sparse recovery by thresholded non-negative least squares", *Adv. Neural Inf. Process. Syst.*, vol. 24, pp. 1926–1934, 2011.

[26] M. Slawski and M. Hein, "Non-negative least squares for high-dimensional linear models: Consistency and sparse recovery without regularization", *Electron. J. Stat.*, vol. 7, pp. 3004–3056, 2013.

[27] R. Peharz and F. Pernkopf, "Sparse nonnegative matrix factorization with ℓ^0 constraints", *Neurocomputing*, vol. 80, pp. 38–46, Mar. 2012.

[28] A. M. Bruckstein, M. Elad, and M. Zibulevsky, "On the uniqueness of nonnegative sparse solutions to underdetermined systems of equation", *IEEE Trans. Inf. Theory*, vol. 54, no. 11, pp. 4813–4820, Nov. 2008.

[29] M. Yaghoobi, D. Wu, and M. E. Davies, "Fast non-negative orthogonal matching pursuit", *IEEE Signal Process. Lett.*, vol. 22, no. 9, pp. 1229–1233, Sept. 2015.

[30] M. Yaghoobi and M. E. Davies, "Fast non-negative orthogonal least squares", in *Proc. Eur. Sig. Proc. Conf.*, Nice, France, Aug. 2015, pp. 479–483.

[31] T. Blumensath and M. E. Davies, "On the difference between Orthogonal Matching Pursuit and Orthogonal Least Squares", Tech. Rep., Univ. Edinburgh, Mar. 2007.

[32] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer texts in Operations Research and Financial Engineering. Springer Verlag, New York, 2nd edition, July 2006.

[33] S. Leichner, G. Dantzig, and J. Davis, "A strictly improving linear programming Phase I algorithm", *Ann. Oper. Res.*, vol. 47, pp. 409–430, 1993.

[34] K. N. Ramamurthy, J. J. Thiagarajan, and A. Spanias, "Recovering non-negative and combined sparse representations", *Digital Signal Process.*, vol. 26, no. 1, pp. 21–35, Mar. 2014.

[35] D. Kim and J. P. Haldar, "Greedy algorithms for nonnegativity-constrained simultaneous sparse recovery", *Signal Process.*, vol. 125, pp. 274–289, 2016.

[36] Z. Wang, R. Zhu, K. Fukui, and J. Xue, "Cone-based joint sparse modelling for hyperspectral image classification", *Signal Process.*, vol. 144, pp. 417–429, 2018.

[37] X. Chen, F. Xu, and Y. Ye, "Lower bound theory of nonzero entries in solutions of ℓ_2-ℓ_p minimization", *SIAM J. Sci. Comput.*, vol. 32, no. 5, pp. 2832–2852, 2010.

[38] S. Foucart and D. Koslicki, "Sparse recovery by means of nonnegative least squares", *IEEE Signal Process. Lett.*, vol. 21, no. 4, pp. 498–502, Apr. 2014.

[39] C. Herzet and A. Drémeau, "Bayesian pursuit algorithms", in *Proc. Eur. Sig. Proc. Conf.*, Aalborg, Denmark, Aug. 2010, pp. 1474–1478.

[40] C. Soussen, J. Idier, D. Brie, and J. Duan, "From Bernoulli-Gaussian deconvolution to sparse signal restoration", *IEEE Trans. Signal Process.*, vol. 59, no. 10, pp. 4572–4584, Oct. 2011.

[41] T. Zhang, "Sparse recovery with orthogonal matching pursuit under RIP", *IEEE Trans. Inf. Theory*, vol. 57, no. 9, pp. 6215–6221, Sept. 2011.

[42] A. Bonnefoy, V. Emiya, L. Ralaivola, and R. Gribonval, "Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso", *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5121–5132, Oct. 2015.

[43] L. Belmerhnia, E.-H. Djermoune, C. Carteret, and D. Brie, "Simultaneous regularized sparse approximation for wood wastes NIR spectra features selection", in *Proc. CAMSAP*, Cancun, Mexico, Dec. 2015.

[44] K. Wagner, T. Schnabel, M.-C. Barbu, and A. Petutschnigg, "Analysis of selected properties of fibreboard panels manufactured from wood and leather using the near infrared spectroscopy", *Int J. Spectrosc.*, vol. 2015, pp. 691796, 2015.

[45] Y. Lin, J. P. Haldar, Q. Li, P. Conti, and R. M. Leahy, "Sparsity constrained mixture modeling for the estimation of kinetic parameters in dynamic PET", *IEEE Trans. Med. Imag.*, vol. 33, no. 1, pp. 173–185, Jan. 2014.

[46] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples", *Appl. Comp. Harmonic Anal.*, vol. 26, no. 3, pp. 301–321, May 2009.

[47] T. T. Nguyen, C. Soussen, J. Idier, and E.-H. Djermoune, "Sign preservation analysis of orthogonal greedy algorithms", Tech. rep., Univ. Lorraine, CentraleSupélec, LS2N, https://hal.archives-ouvertes.fr/hal-01971697, Nancy, France, Jan. 2019.