From Bernoulli-Gaussian deconvolution to sparse signal restoration

Charles Soussen, Jérôme Idier, David Brie, Junbo Duan

From Bernoulli-Gaussian deconvolution to sparse signal restoration

Charles Soussen*, Jérôme Idier, *Member, IEEE,* David Brie, and Junbo Duan

Abstract

Formulated as a least-square problem under an ℓ_0 constraint, sparse signal restoration is a discrete optimization problem, known to be NP complete. Classical algorithms include, by increasing cost and efficiency, Matching Pursuit (MP), Orthogonal Matching Pursuit (OMP), Orthogonal Least Squares (OLS) and the exhaustive search algorithm. In inverse problems involving highly correlated dictionaries, OMP and OLS are not guaranteed to find the optimal solution. It is of interest to develop slightly slower sub-optimal search algorithms yielding better approximations. We revisit the Single Most Likely Replacement (SMLR) algorithm, developed in the mid-80's for Bernoulli-Gaussian signal restoration. We show that the formulation of sparse signal restoration as a limit case of Bernoulli-Gaussian signal restoration leads to an ℓ_0-penalized least-square minimization problem, to which SMLR can be straightforwardly adapted. The resulting algorithm, called Single Best Replacement (SBR), can be interpreted as a forward-backward extension of OLS. A fast and stable implementation is proposed. The approach is illustrated on a deconvolution problem with a Gaussian impulse response and on the joint detection of discontinuities at different orders in a signal.

Index Terms

Sparse signal estimation; inverse problems; Bernoulli-Gaussian signal restoration; SMLR algorithm; mixed ℓ_2-ℓ_0 criterion minimization; Orthogonal Least Squares; forward-backward greedy algorithms.

C. Soussen, D. Brie, and J. Duan are with the Centre de Recherche en Automatique de Nancy (CRAN, UMR 7039, Nancy-University, CNRS). Boulevard des Aiguillettes, B.P. 70239, F-54506 Vandœuvre-lès-Nancy, France. Tel: (+33)-3 83 68 44 71, Fax: (+33)-3 83 68 44 62. E-mail: {FirstName.SecondName}@cran.uhp-nancy.fr.

J. Idier is with the Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN, UMR CNRS 6597), BP 92101, 1 rue de la Noë, 44321 Nantes Cedex 3, France. Tel: (+33)-2 40 37 69 09, Fax: (+33)-2 40 37 69 30. E-mail: Jerome.Idier@irccyn.ec-nantes.fr.

I. INTRODUCTION

Sparse signal restoration arises in inverse problems such as Fourier synthesis, mono- and multidimensional deconvolution, and statistical regression. It consists in the decomposition of a given signal $\boldsymbol{y}$ as a linear combination of a limited number of elements from a dictionary $\boldsymbol{A}$. While formally very similar, sparse signal restoration has to be distinguished from sparse signal approximation. The main difference is that in sparse signal restoration, the choice of the dictionary is imposed by the inverse problem at hand whereas in sparse signal approximation, the dictionary has to be chosen according to its ability to represent the data with a limited number of coefficients. A more subtle difference is that in sparse signal restoration, the focus is set on the estimation of the weights of the linear combination while in sparse signal approximation, the goal is to reproduce the data $\boldsymbol{y}$ as well as possible at a given level of sparsity.

Sparse signal restoration can be formulated as the minimization of a least-square cost function of the form $\mathcal{E}(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ under the constraint that the ℓ_0 pseudo-norm of $\boldsymbol{x}$, defined as the number of non-zero entries in $\boldsymbol{x}$, is lower than a given number k. This problem is often referred to as *subset selection*, because imposing the sparsity constraint consists in selecting a subset of columns of $\boldsymbol{A}$. This yields a discrete problem (since there are a finite number of possible subsets) which is known to be NP-complete [1]. In this paper, we focus on "difficult" problems in which some of the columns of $\boldsymbol{A}$ are highly correlated, the unknown weight vector $\boldsymbol{x}^\star$ is only approximately sparse, and/or the data are noisy. Hereafter, we distinguish two approaches to address the subset selection problem in a fast and sub-optimal manner and we discuss their relevance for difficult problems.

The first approach, which has been the most popular in the last decade, approximates the subset selection problem by a continuous optimization problem, convex or not, that is easier to solve [2–5]. In particular, the approach that replaces the ℓ_0-norm by the ℓ_1-norm [4, 5] has been increasingly investigated, leading to the LASSO optimization problem. Its popularity relies on efficient algorithms, such as LARS which finds the set of solutions for all degrees of sparsity [6]. Several authors have provided sufficient conditions under which the ℓ_0- and ℓ_1-constrained least-square problems lead to solutions having the same support [5, 7]. These conditions typically state that the unknown weight signal has to be highly sparse, that the correlation between any pair of columns of $\boldsymbol{A}$ must be sufficiently small, and that the noise level must be low. They are often not satisfied when dealing with real data.

The second approach addresses the *exact* subset selection problem using either thresholding algorithms [8, 9] or greedy search algorithms. The latter gradually increase or decrease by one the set of active columns. The simplest greedy algorithms are Matching Pursuit (MP) [10] and the improved version

Orthogonal Matching Pursuit (OMP) [11]. Both are referred to as forward greedy algorithms, since they start from an empty active set and then gradually increase it by one element. In contrast, the backward algorithm of Couvreur and Bresler [12] starts from a complete active set which is gradually decreased by one element. It is, however, only valid if the dictionary is not overcomplete. A few authors have introduced forward-backward algorithms in which insertions and removals of dictionary elements into the active set are both allowed [13, 14]. This strategy yields better recovery performance since an early wrong selection can be counteracted by its further removal from the active set. In contrast, the insertion of a wrong element is irreversible when using forward algorithms.

The choice of the algorithm depends on the amount of time available and on the structure of matrix $\boldsymbol{A}$. In favorable cases, the sub-optimal search algorithms described above (belonging to the first or the second approach) provide solutions having the same support as the exhaustive search solution. For example, if the unknown signal $\boldsymbol{x}^\star$ is highly sparse and if the correlation between any pair of columns of $\boldsymbol{A}$ is low, the ℓ_1-norm approximation provides optimal solutions [5, 7]. In other cases, however, the only guarantee to recover the optimal solution is to use the exhaustive search algorithm. When fast sub-optimal algorithms lead to unsatisfactory results, it is of great interest to develop slower sub-optimal algorithms providing more accurate solutions, but remaining very fast compared to the exhaustive search. The Orthogonal Least Squares algorithm (OLS) [15] which is sometimes confused with OMP [16], falls into this category. Both OLS and OMP share the same structure, the difference being that at each iteration, OLS solves a large number of least-square problems ($n-k$, where k is the cardinal of the current active set) while OMP only computes the $n-k$ inner products between the current residual $\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}$ and the candidate columns $\boldsymbol{a}_i$ and chooses the column yielding the maximal inner product. OMP solves only one least-square problem per iteration, once the column to be inserted is selected (in order to update all the active weights). In the following, we propose a forward-backward extension of OLS allowing an insertion or a removal at each iteration, each iteration requiring to solve n least-square problems. It differs from the FoBa algorithm of Zhang [14] which is an OMP forward-backward extension. It is closer to the bidirectional OLS based algorithm of Haugland [13], the main differences relying on the search and implementation strategies.

The starting point of our forward-backward algorithm is the Single Most Likely Replacement (SMLR) algorithm, which proved to be a very efficient tool for the deconvolution of a sparse signal modeled as a Bernoulli-Gaussian process [17–20]. This approach relies on a Bayesian formulation of a deconvolution problem of the form $\boldsymbol{y}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{n}$ (where $\boldsymbol{A}$ denotes the convolution matrix) and on the maximum *a posteriori* (MAP) estimation of the sparse signal. The Bernoulli-Gaussian model is a probabilistic model for sparse signals, in which (binary) Bernoulli random variables are associated to the position of the

non zero entries in $\boldsymbol{x}$ while the corresponding amplitudes are distributed according to an independent identically distributed (i.i.d.) centered Gaussian distribution of variance σ_x^2. SMLR is a deterministic ascent algorithm which maximizes the posterior likelihood in a sub-optimal manner. It consists in updates (increase or decrease) of the support of $\boldsymbol{x}$ by one element and the subsequent estimation of the non-zero amplitudes. Sparse signal restoration can be seen as a limit case of Bernoulli-Gaussian MAP restoration in which the variance σ_x^2 of the amplitudes is set to infinity. We will deduce an adaptation of SMLR to subset selection relying on a single insertion or a single removal of a column into/from the active set.

The paper is organized as follows. In Section II, we introduce the Bernoulli-Gaussian model and the Bayesian framework from which we formulate the sparse signal restoration problem. In Section III, we adapt the SMLR algorithm resulting in the so-called Single Best Replacement (SBR) algorithm. In Section IV, a fast and stable SBR implementation is proposed, based on the efficient update of the squared error when the active set is modified by one element. Finally, Sections V and VI illustrate the method on the sparse spike deconvolution with a Gaussian impulse response and on the joint detection of discontinuities at different orders in a signal.

II. From Bernoulli-Gaussian signal modeling to sparse signal representation

We consider the restoration of a sparse signal $\boldsymbol{x}$ from a linear observation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$, where $\boldsymbol{n}$ stands for the observation noise. An acknowledged probabilistic model dedicated to sparse signals is the Bernoulli-Gaussian (BG) model [17, 18, 20]. For such model, deterministic optimization algorithms [20] and Markov chain Monte Carlo techniques [21] are used to compute the MAP and the posterior mean, respectively. We first recall the known BG models and the formulation of BG signal restoration in the Bayesian framework. Then, we extend this formulation to a more general representation of sparse signals.

A. Preliminary definitions and working assumptions

Given an observation vector $\boldsymbol{y} \in \mathbb{R}^m$ and a dictionary $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n] \in \mathbb{R}^{m \times n}$, a subset selection algorithm aims at computing a weight vector $\boldsymbol{x} \in \mathbb{R}^n$ yielding an accurate approximation $\boldsymbol{y} \approx \boldsymbol{A}\boldsymbol{x}$ of the observation. The columns $\boldsymbol{a}_i$ of $\boldsymbol{A}$ whose indices correspond to the non-zero components x_i of $\boldsymbol{x}$ are referred to as the active (or selected) columns.

Throughout this paper, no assumption is made on the size of $\boldsymbol{A}$: m can be either smaller or larger than n. Here, $\boldsymbol{A}$ is assumed to satisfy the unique representation property (URP). This assumption is usual in the case $m \leqslant n$ [22]. It is a stronger assumption than the full rank assumption. We now recall this definition and extend it to the case where $m > n$. Notation $\|\cdot\|$ refers to the Euclidean norm.

Definition 1 *When $m \leqslant n$, $\boldsymbol{A}$ satisfies the URP if and only if any selection of m columns of $\boldsymbol{A}$ forms a family of linearly independent vectors. When $m > n$, $\boldsymbol{A}$ satisfies the URP if and only if it is full rank.*

Before going further, let us mention that this assumption can be relaxed providing that the search strategy can guarantee that the selected columns of $\boldsymbol{A}$ result in a full rank matrix (see Section VI-C for details).

Under the URP assumption, when $m \leqslant n$, the system $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ has a number of solutions whose ℓ_0-norm are lower or equal to m since any selection of m columns yields a solution of the system. When $m > n$, there is generally no solution to $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ but the least-square estimator $\boldsymbol{x} = (\boldsymbol{A}^t\boldsymbol{A})^{-1}\boldsymbol{A}^t\boldsymbol{y}$ is unique, although not necessarily sparse.

Definition 2 *The support of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is the set $\mathcal{S}(\boldsymbol{x}) \subseteq \{1, \ldots, n\}$ defined by $i \in \mathcal{S}(\boldsymbol{x})$ if and only if $x_i \neq 0$.*

Definition 3 *We denote by $\mathcal{Q} \subseteq \{1, \ldots, n\}$ the active set and we define the related vector $\boldsymbol{q} \in \{0,1\}^n$ by $q_i = 1$ if and only if $i \in \mathcal{Q}$. Let $\boldsymbol{A}_\mathcal{Q}$ be the submatrix of size $m \times \mathrm{Card}\,[\mathcal{Q}]$ formed of the active columns of $\boldsymbol{A}$ ($\boldsymbol{a}_i$, $i \in \mathcal{Q}$). The observation model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$ also reads $\boldsymbol{y} = \boldsymbol{A}_\mathcal{Q}\boldsymbol{t} + \boldsymbol{n}$, where the reduced vector $\boldsymbol{t}$ of size $\mathrm{Card}\,[\mathcal{Q}]$ gathers the values $\{x_i,\, i \in \mathcal{Q}\}$.*

Definition 4 *For all $\mathcal{Q} \subseteq \{1, \ldots, n\}$ such that $\mathrm{Card}\,[\mathcal{Q}] \leqslant \min(m, n)$, let $\boldsymbol{x}_\mathcal{Q}$ be the least-square solution and let $\mathcal{E}_\mathcal{Q}$ be the associated squared error:*

$$\boldsymbol{x}_\mathcal{Q} \triangleq \underset{\mathcal{S}(\boldsymbol{x}) \subseteq \mathcal{Q}}{\arg\min} \{\mathcal{E}(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2\} \tag{1}$$

$$\mathcal{E}_\mathcal{Q} \triangleq \mathcal{E}(\boldsymbol{x}_\mathcal{Q}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_\mathcal{Q}\|^2. \tag{2}$$

B. Bernoulli-Gaussian models

A BG process[1] $\boldsymbol{x}$ can be defined by means of a Bernoulli random vector $\boldsymbol{q} \in \{0,1\}^n$ corresponding to the active set, and a Gaussian random vector $\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}, \sigma_x^2 \boldsymbol{I}_n)$, where $\boldsymbol{I}_n$ stands for the identity matrix of size $n \times n$. Each sample x_i of $\boldsymbol{x}$ is modeled as $x_i = q_i r_i$ [17, 18]. Thus, r_i code for the amplitudes of the nonzero entries in $\boldsymbol{x}$. In the vector form, $\boldsymbol{x}$ reads $\boldsymbol{x} = \boldsymbol{\Delta}_\boldsymbol{q}\boldsymbol{r}$ where $\boldsymbol{\Delta}_\boldsymbol{q}$ is the diagonal matrix of size $n \times n$ whose diagonal elements are equal to q_i. The Bernoulli random variables q_i code for the presence ($q_i = 1$) or absence ($q_i = 0$) of signal at location i, the Bernoulli parameter $\rho = \Pr(q_i = 1)$ being the probability

[1]For convenience, we use the same notations for random vectors and their realization.

of presence of signal. It is easy to check that the prior likelihood of $\boldsymbol{q}$ reads $l(\boldsymbol{q}) = \rho^{\|\boldsymbol{q}\|_0}(1-\rho)^{n-\|\boldsymbol{q}\|_0}$. Because $\boldsymbol{q}$ and $\boldsymbol{r}$ are independent random vectors, the prior likelihood of $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{r})$ reads:

$$l(\boldsymbol{q}, \boldsymbol{r}) = l(\boldsymbol{r})\, l(\boldsymbol{q}) = g(\boldsymbol{r}; \sigma_x^2 \boldsymbol{I}_n)\, \rho^{\|\boldsymbol{q}\|_0}(1-\rho)^{n-\|\boldsymbol{q}\|_0}, \tag{3}$$

where $g(.\,; \boldsymbol{\Gamma})$ denotes the probability density of the centered Gaussian with covariance matrix $\boldsymbol{\Gamma}$.

C. Bayesian formulation of sparse signal restoration

The Bayesian formulation of the inverse problem $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$ consists in inferring the distribution of $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{r})$ knowing $\boldsymbol{y}$. The MAP estimator of $\boldsymbol{x}$ can be obtained by maximizing the marginal distribution $l(\boldsymbol{q} \,|\, \boldsymbol{y})$ [20] or the joint distribution $l(\boldsymbol{q}, \boldsymbol{r} \,|\, \boldsymbol{y})$ [18, 19]. Following [18], we focus on the joint likelihood $l(\boldsymbol{q}, \boldsymbol{r} \,|\, \boldsymbol{y})$ which leads to a cost function involving the squared error $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ and the ℓ_0-norm of $\boldsymbol{x}$.

Assuming a Gaussian white noise $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \sigma_n^2 \boldsymbol{I}_m)$, independent from $\boldsymbol{x}$, it is easy to obtain

$$\begin{aligned} \mathcal{L}(\boldsymbol{q}, \boldsymbol{r}) &\triangleq -2\sigma_n^2 \log[l(\boldsymbol{q}, \boldsymbol{r} \,|\, \boldsymbol{y})] \\ &= \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\Delta}_{\boldsymbol{q}}\boldsymbol{r}\|^2 + \frac{\sigma_n^2}{\sigma_x^2}\|\boldsymbol{r}\|^2 + \lambda\|\boldsymbol{q}\|_0 + C, \end{aligned} \tag{4}$$

where $\lambda = 2\sigma_n^2 \log(1/\rho - 1)$ and C is a constant. Given $\boldsymbol{q}$, let us split $\boldsymbol{r}$ into two subvectors $\boldsymbol{u}$ and $\boldsymbol{t}$ indexed by the null and non-null entries of $\boldsymbol{q}$, respectively. Since $\boldsymbol{A}\Delta_{\boldsymbol{q}}\boldsymbol{r} = \boldsymbol{A}_{\mathcal{Q}}\boldsymbol{t}$ does not depend on $\boldsymbol{u}$, it is easy to check that $\min_{\boldsymbol{u}} \mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{u}) = \mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$. Finally, the joint MAP estimation problem reduces to the minimization of $\mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$ w.r.t. $(\boldsymbol{q}, \boldsymbol{t})$.

D. Mixed ℓ_2-ℓ_0 minimization as a limit case

A signal $\boldsymbol{x}$ is sparse if some entries x_i are equal to 0. Since this definition does not impose constraints on the range of values of the non zero amplitudes, we choose to describe a sparse signal by a limit Bernoulli-Gaussian model in which the amplitude variance σ_x^2 is set to infinity. The minimization of $\mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$ thus rereads:

$$\min_{\boldsymbol{q}, \boldsymbol{t}} \{\mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0}) = \|\boldsymbol{y} - \boldsymbol{A}_{\mathcal{Q}}\boldsymbol{t}\|^2 + \lambda\|\boldsymbol{q}\|_0\}. \tag{5}$$

Theorem 1 *The above formulation* (5) *is equivalent to:*

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \{\mathcal{J}(\boldsymbol{x}; \lambda) \,=\, \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 + \lambda\|\boldsymbol{x}\|_0\} \tag{6}$$

which is referred to as the ℓ_0-penalized least-square problem. *The term "equivalent" means that given a minimizer* $(\boldsymbol{q}, \boldsymbol{t})$ *of* (5)*, the related vector* $\boldsymbol{x} = \{\boldsymbol{t}, \boldsymbol{0}\}$ *is a minimizer of* (6)*, and conversely, given*

a minimizer $\boldsymbol{x}$ of (6)*, the vectors $\boldsymbol{q}$ and $\boldsymbol{t}$ defined as the support of $\boldsymbol{x}$ and its non-zero amplitudes, respectively, are such that $(\boldsymbol{q}, \boldsymbol{t})$ is a minimizer of* (5)*.*

Proof: To prove the equivalence, we first prove that $\min_{\boldsymbol{x}} \mathcal{J}(\boldsymbol{x}; \lambda) = \min_{\boldsymbol{q},\boldsymbol{t}} \mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$:

- Let $\boldsymbol{x}$ be a minimizer of $\mathcal{J}(.\,; \lambda)$. We set $\boldsymbol{q}$ and $\boldsymbol{t}$ to the support and the non zero amplitudes of $\boldsymbol{x}$, respectively. Obviously, $\mathcal{J}(\boldsymbol{x}; \lambda) = \mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$. It follows that $\min_{\boldsymbol{x}} \mathcal{J}(\boldsymbol{x}; \lambda) \geqslant \min_{\boldsymbol{q},\boldsymbol{t}} \mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$.
- Let $(\boldsymbol{q}, \boldsymbol{t})$ be a minimizer of $\mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$. The vector $\boldsymbol{x} = \{\boldsymbol{t}, \boldsymbol{0}\}$ is such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}_{\mathcal{Q}}\boldsymbol{t}$ and $\|\boldsymbol{x}\|_0 = \|\boldsymbol{t}\|_0 \leqslant \|\boldsymbol{q}\|_0$. Therefore, $\mathcal{J}(\boldsymbol{x}; \lambda) \leqslant \mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$. It follows that $\min_{\boldsymbol{q},\boldsymbol{t}} \mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0}) \geqslant \min_{\boldsymbol{x}} \mathcal{J}(\boldsymbol{x}; \lambda)$.

We have shown that $\min_{\boldsymbol{x}} \mathcal{J}(\boldsymbol{x}; \lambda) = \min_{\boldsymbol{q},\boldsymbol{t}} \mathcal{L}(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0})$, but also that the minimizers of both problems coincide, *i.e.,* are vectors describing identical signals. ■

In the following, we focus on the minimization problem (6) involving the penalization term $\|\boldsymbol{x}\|_0$. The hyperparameter λ is fixed. It controls the level of sparsity of the desired solution. The algorithm that will be developed is based on an efficient search of the support of $\boldsymbol{x}$. In that respect, the ℓ_0-penalized least-square problem does not drastically differ from the ℓ_0-constrained problem $\min \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ subject to $\|\boldsymbol{x}\|_0 \leqslant k$.

III. Adaptation of SMLR to ℓ_0-penalized least-square optimization

We propose to adapt the SMLR algorithm to the minimization of the mixed ℓ_2-ℓ_0 cost function $\mathcal{J}(\boldsymbol{x}; \lambda)$ defined in (6). To clearly distinguish SMLR which specifically aims at minimizing (4), the adapted algorithm will be termed as Single Best Replacement (SBR).

A. Principle of SMLR and main notations

The Single Most Likely Replacement algorithm [17] is a deterministic coordinatewise ascent algorithm to maximize log-likelihood functions of the form $l(\boldsymbol{q} \,|\, \boldsymbol{y})$ (marginal MAP estimation) or $l(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0} \,|\, \boldsymbol{y})$ (joint MAP estimation). In the latter case, it is easy to check from (4) that given $\boldsymbol{q}$, the maximizer of $l(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{0} \,|\, \boldsymbol{y})$ w.r.t. $\boldsymbol{t}$ has a closed form expression $\boldsymbol{t} = \boldsymbol{t}(\boldsymbol{q})$. Consequently, the joint MAP estimation reduces to the maximization of $l(\boldsymbol{q}, \boldsymbol{t}(\boldsymbol{q}), \boldsymbol{0} \,|\, \boldsymbol{y})$ w.r.t. $\boldsymbol{q}$. At each SMLR iteration, all the possible single replacements of the support $\boldsymbol{q}$ (set $q_i = 1 - q_i$ while keeping the other q_j, $j \neq i$ unchanged) are tested, then the replacement yielding the maximal increase of $l(\boldsymbol{q}, \boldsymbol{t}(\boldsymbol{q}), \boldsymbol{0} \,|\, \boldsymbol{y})$ is chosen. This task is repeated until no single replacement can increase $l(\boldsymbol{q}, \boldsymbol{t}(\boldsymbol{q}), \boldsymbol{0} \,|\, \boldsymbol{y})$ anymore. The number of possible supports $\boldsymbol{q}$ being finite (2^n) and SMLR being an ascent algorithm, it terminates after a finite number of iterations.

Before adapting SMLR, let us introduce some useful notations.

- We denote by $\mathcal{Q} \bullet i$ a single replacement, *i.e.*, the insertion or removal of an index i into/from the active set $\mathcal{Q}$:

$$\mathcal{Q} \bullet i \triangleq \begin{cases} \mathcal{Q} \cup \{i\} & \text{if } i \notin \mathcal{Q}, \\ \mathcal{Q} \backslash \{i\} & \text{otherwise.} \end{cases} \tag{7}$$

- If $\mathrm{Card}\,[\mathcal{Q}] \leqslant \min(m, n)$, we define the cost functions:

$$\mathcal{J}_{\mathcal{Q}}(\lambda) \triangleq \mathcal{J}(\boldsymbol{x}_{\mathcal{Q}}; \lambda) = \mathcal{E}_{\mathcal{Q}} + \lambda \|\boldsymbol{x}_{\mathcal{Q}}\|_0 \tag{8}$$

$$\mathcal{K}_{\mathcal{Q}}(\lambda) \triangleq \mathcal{E}_{\mathcal{Q}} + \lambda \mathrm{Card}\,[\mathcal{Q}] \tag{9}$$

where the least-square solution $\boldsymbol{x}_{\mathcal{Q}}$ and the corresponding error $\mathcal{E}_{\mathcal{Q}}$ have been defined in (1) and (2). Obviously, $\mathcal{J}_{\mathcal{Q}}(\lambda) = \mathcal{K}_{\mathcal{Q}}(\lambda)$ *if and only if* $\boldsymbol{x}_{\mathcal{Q}}$ has a support equal to $\mathcal{Q}$. In subsection III-B, we introduce a first version of SBR involving $\mathcal{J}_{\mathcal{Q}}(\lambda)$ only, and in subsection III-C, we present an alternative (simpler) version relying on the computation of $\mathcal{K}_{\mathcal{Q}}(\lambda)$ instead of $\mathcal{J}_{\mathcal{Q}}(\lambda)$ and we discuss in which extent both versions differ. Then, subsection III-D describes the behavior of SBR and states its main properties.

B. The Single Best Replacement algorithm (first version)

SMLR can be seen as an exploration strategy for discrete optimization rather than an algorithm specific to a posterior likelihood function. Here, we use the same strategy to minimize the cost function $\mathcal{J}(\boldsymbol{x}; \lambda)$. We rename the algorithm Single Best Replacement to remove any statistical connotation. The SBR algorithm works as follows. At each iteration, the n possible single replacements $\mathcal{Q} \bullet i\,(i = 1, \ldots, n)$ are tested, then the best is selected, *i.e.*, the replacement yielding the maximal decrease of $\mathcal{J}(\boldsymbol{x}; \lambda)$. This task is repeated until $\mathcal{J}(\boldsymbol{x}; \lambda)$ cannot decrease anymore. Let us detail an SBR iteration.

Consider the current active set $\mathcal{Q}$. For each index i, we compute the minimizer $\boldsymbol{x}_{\mathcal{Q}\bullet i}$ of $\mathcal{E}$ whose support is included in $\mathcal{Q} \bullet i$ and we keep in memory the value of $\mathcal{J}_{\mathcal{Q}\bullet i}(\lambda)$. If the minimum of $\{\mathcal{J}_{\mathcal{Q}\bullet i}(\lambda),\, i = 1, \ldots, n\}$ is lower than $\mathcal{J}_{\mathcal{Q}}(\lambda)$, then we select the index yielding this minimal value:

$$\ell \in \underset{i \in \{1,\ldots,n\}}{\arg\min}\ \mathcal{J}_{\mathcal{Q}\bullet i}(\lambda). \tag{10}$$

The next SBR iterate is thus defined as $\mathcal{Q}' = \mathcal{Q} \bullet \ell$, yielding the vector $\boldsymbol{x}_{\mathcal{Q}'}$.

Except when an initial support estimate (of cardinality lower than $\min(m, n)$) is available, we suggest to use an initial empty active set.

Remark 1 (Relationship between SBR and SMLR) *We introduced SBR as the application of the SMLR search strategy to the ℓ_0-penalized least square cost function* (6) *which is obtained by taking the limit of*

the cost function (4) *when* σ_x *tends towards infinity. Conversely, applying SMLR to* (4) *and then, taking the limit of the SMLR formula when* σ_x *tends to infinity also yields the SBR algorithm.*

Actually, the main difference between SMLR and SBR is that SMLR (which can take several forms depending on the use of the joint distribution $l(\boldsymbol{q}, \boldsymbol{r} \,|\, \boldsymbol{y})$ *or the marginal distribution* $l(\boldsymbol{q} \,|\, \boldsymbol{y})$*) involves the inversion of a matrix of the form* $\boldsymbol{A}_{\mathcal{Q}}^t \boldsymbol{A}_{\mathcal{Q}} + \alpha \boldsymbol{I}_{\mathrm{Card}[\mathcal{Q}]}$ *whereas SBR involves the inverse of the Gram matrix* $\boldsymbol{A}_{\mathcal{Q}}^t \boldsymbol{A}_{\mathcal{Q}}$*. For this reason, instabilities may occur while using SBR when* $\boldsymbol{A}_{\mathcal{Q}}$ *is ill conditioned. The use of the term* $\alpha \boldsymbol{I}_{\mathrm{Card}[\mathcal{Q}]}$*, which acts as a regularization on the amplitude values, avoids such instability while using SMLR at the price of handling the additional hyperparameter* α*.*

C. Modified version of SBR (final version)

We introduce a slight modification of SBR by replacing (10) with:

$$\ell \in \underset{i \in \{1,\ldots,n\}}{\arg\min}\ \mathcal{K}_{\mathcal{Q} \bullet i}(\lambda). \tag{11}$$

We propose this modification because $\mathcal{K}_{\mathcal{Q}}(\lambda) = \mathcal{E}_{\mathcal{Q}} + \lambda \mathrm{Card}\,[\mathcal{Q}]$ can be computed more efficiently than $\mathcal{J}_{\mathcal{Q}}(\lambda)$, the computation of $\boldsymbol{x}_{\mathcal{Q}}$ being no longer necessary. The use of $\mathcal{K}_{\mathcal{Q}}(\lambda)$ makes the penalization term very easy to update when $\mathcal{Q}$ is modified by one element (add or remove λ), and the only necessary update is that of $\mathcal{E}_{\mathcal{Q}}$. We now show that there is almost surely no difference between both versions of SBR provided that the data $\boldsymbol{y}$ are corrupted with "non degenerate" noise.

Theorem 2 *Let* $\boldsymbol{y} = \boldsymbol{y}_0 + \boldsymbol{n}$*, where* $\boldsymbol{y}_0$ *is a given vector of* $\mathbb{R}^m$ *and* $\boldsymbol{n}$ *is a random vector. We assume that* $\boldsymbol{n}$ *is an absolute continuous random vector, i.e., admitting a probability density w.r.t. the Lebesgue measure. Then, when* $\mathrm{Card}\,[\mathcal{Q}] \leqslant \min(m, n)$*, the probability that* $\|\boldsymbol{x}_{\mathcal{Q}}\|_0 < \mathrm{Card}\,[\mathcal{Q}]$ *is equal to 0, i.e.,* $\|\boldsymbol{x}_{\mathcal{Q}}\|_0 = \mathrm{Card}\,[\mathcal{Q}]$ *almost surely.*

Proof: Let $k = \mathrm{Card}\,[\mathcal{Q}]$ and let $\boldsymbol{t}_{\mathcal{Q}}$ be the minimizer of $\|\boldsymbol{y} - \boldsymbol{A}_{\mathcal{Q}}\boldsymbol{t}\|^2$ over $\mathbb{R}^k$. Obviously, $\|\boldsymbol{x}_{\mathcal{Q}}\|_0 = \|\boldsymbol{t}_{\mathcal{Q}}\|_0 \leqslant k$. Let $\boldsymbol{V}_{\mathcal{Q}} = (\boldsymbol{A}_{\mathcal{Q}}^t \boldsymbol{A}_{\mathcal{Q}})^{-1} \boldsymbol{A}_{\mathcal{Q}}^t$ be the matrix of size $k \times m$ such that $\boldsymbol{t}_{\mathcal{Q}} = \boldsymbol{V}_{\mathcal{Q}} \boldsymbol{y}$. Denoting by $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^k \in \mathbb{R}^m$ the row vectors of $\boldsymbol{V}_{\mathcal{Q}}$, $\|\boldsymbol{t}_{\mathcal{Q}}\|_0 < k$ *if and only if* there exists i such that $\langle \boldsymbol{y}, \boldsymbol{v}^i \rangle = 0$ (where $\langle \,.\,, \,.\, \rangle$ denotes the inner product). Because $\boldsymbol{A}_{\mathcal{Q}}$ is full rank, $\boldsymbol{V}_{\mathcal{Q}}$ is full rank and then $\forall i,\ \boldsymbol{v}^i \neq \boldsymbol{0}$. Denoting by $\mathcal{H}^{\perp}(\boldsymbol{v}^i)$ the hyperplane of $\mathbb{R}^m$ which is orthogonal to $\boldsymbol{v}^i$, we have

$$\|\boldsymbol{x}_{\mathcal{Q}}\|_0 < k \iff \boldsymbol{y} \in \bigcup_{i=1}^{k} \mathcal{H}^{\perp}(\boldsymbol{v}^i). \tag{12}$$

Because the set $\bigcup_i \mathcal{H}^{\perp}(\boldsymbol{v}^i)$ has a Lebesgue measure equal to zero and the random vector $\boldsymbol{y}$ admits a probability density, the probability of event (12) is zero, thus $\Pr(\|\boldsymbol{x}_{\mathcal{Q}}\|_0 < k) = 0$. ■

TABLE I

SBR ALGORITHM (FINAL VERSION). BY DEFAULT, THE INITIAL ACTIVE SET IS EMPTY: $\mathcal{Q}_1 = \emptyset$.

Input: $\boldsymbol{A}$, $\boldsymbol{y}$, λ and active set $\mathcal{Q}_1$ ($\mathrm{Card}\,[\mathcal{Q}_1] \leqslant \min(m,n)$)
Step 1: Set $j = 1$.
Step 2: For $i \in \{1,\ldots,n\}$, compute $\mathcal{K}_{\mathcal{Q}_j \bullet i}(\lambda)$.
Compute ℓ using (11).
If $\mathcal{K}_{\mathcal{Q}_j \bullet \ell}(\lambda) < \mathcal{K}_{\mathcal{Q}_j}(\lambda)$,
Set $\mathcal{Q}_{j+1} = \mathcal{Q}_j \bullet \ell$.
else,
Terminate SBR.
End if.
Set $j = j + 1$ and go to Step 2.
Output: active set $\mathcal{Q}_j = \mathrm{SBR}(\mathcal{Q}_1; \lambda)$

Theorem 2 implies that when dealing with real noisy data, it is almost sure that no active component x_i is exactly equal to 0. Thus, the original and modified versions of SBR almost surely lead to exactly the same iterates. Even in the noiseless case, an active component is rarely numerically evaluated to 0 due to the round-off errors. In all cases, the modified version of SBR can be applied without restriction and the properties stated below (*e.g.*, termination after a finite number of iterations) remain valid even if an SBR iterate satisfies $\|\boldsymbol{x}_{\mathcal{Q}}\|_0 < \mathrm{Card}\,[\mathcal{Q}]$.

We adopt the modified version of SBR in the rest of the paper. It is summarized in Table I.

D. Behavior and adaptations of SBR

Termination: SBR is a descent algorithm because the value of $\mathcal{K}_{\mathcal{Q}}(\lambda)$ is always decreasing. Consequently, a set $\mathcal{Q}$ cannot be explored twice and similarly to SMLR, SBR terminates after a finite number of iterations. Notice that the size of $\mathcal{Q}$ remains lower or equal to $\min(m,n)$. Indeed, if a set $\mathcal{Q}$ of cardinality $\min(m,n)$ is reached, then $\mathcal{E}_{\mathcal{Q}}$ is equal to 0 due to the URP assumption. Hence, any replacement of the form $\mathcal{Q}' = \mathcal{Q} \cup \{i\}$ yields an increase of the cost function ($\mathcal{K}_{\mathcal{Q}'}(\lambda) = \mathcal{K}_{\mathcal{Q}}(\lambda) + \lambda$). We emphasize that no stopping condition is needed unlike many algorithms which require to set a maximum number of iterations (MP and variations, OLS) and/or a threshold on the squared error variation (CoSaMP, Iterative Hard Thresholding).

Proposition 1 *Under the assumptions of Theorem 2, each SBR iterate $\boldsymbol{x}_{\mathcal{Q}}$ is almost surely a local minimizer of $\mathcal{J}(\boldsymbol{x};\lambda)$ and of the ℓ_0-constrained problem $\min \mathcal{E}(\boldsymbol{x})$ subject to $\|\boldsymbol{x}\|_0 \leqslant k$, with $k = \mathrm{Card}\,[\mathcal{Q}]$. This property holds in particular for the SBR output.*

Proof: Let $\boldsymbol{x} = \boldsymbol{x}_{\mathcal{Q}}$ be an SBR iterate. According to Theorem 2, the support $\mathcal{S}(\boldsymbol{x}) = \mathcal{Q}$ almost surely. Setting $\varepsilon = \min_{i\in\mathcal{Q}} |x_i| > 0$, it is easy to check that if $\boldsymbol{x}' \in \mathbb{R}^n$ satisfies $\|\boldsymbol{x}' - \boldsymbol{x}\| < \varepsilon$, then $\forall i \in \mathcal{Q}$, $x'_i \neq 0$. Thus, $\|\boldsymbol{x}' - \boldsymbol{x}\| < \varepsilon$ implies that $\mathcal{S}(\boldsymbol{x}') \supseteq \mathcal{S}(\boldsymbol{x})$ and then $\|\boldsymbol{x}'\|_0 \geqslant k$.

Now, let $\boldsymbol{x}'$ satisfy $\|\boldsymbol{x}' - \boldsymbol{x}\| < \varepsilon$.

- If $\|\boldsymbol{x}'\|_0 = k$, then necessarily, $\mathcal{S}(\boldsymbol{x}') = \mathcal{S}(\boldsymbol{x}) = \mathcal{Q}$. By definition of $\boldsymbol{x} = \boldsymbol{x}_{\mathcal{Q}}$, $\mathcal{E}(\boldsymbol{x}') \geqslant \mathcal{E}(\boldsymbol{x})$. It follows that $\boldsymbol{x}$ is a local minimizer of the ℓ_0-constrained problem. Also, $\mathcal{J}(\boldsymbol{x}';\lambda) \geqslant \mathcal{J}(\boldsymbol{x};\lambda)$ holds.
- If $\|\boldsymbol{x}'\|_0 > k$, then $\mathcal{J}(\boldsymbol{x}';\lambda) = \mathcal{E}(\boldsymbol{x}') + \lambda\|\boldsymbol{x}'\|_0 \geqslant \mathcal{E}(\boldsymbol{x}') + \lambda(k+1)$. By continuity of $\mathcal{E}$, there exists a neighborhood $\mathcal{V}(\boldsymbol{x})$ of $\boldsymbol{x}$ such that if $\boldsymbol{x}' \in \mathcal{V}(\boldsymbol{x})$, $|\mathcal{E}(\boldsymbol{x}') - \mathcal{E}(\boldsymbol{x})| < \lambda$. Thus, if $\boldsymbol{x}' \in \mathcal{V}(\boldsymbol{x})$ and $\|\boldsymbol{x}' - \boldsymbol{x}\| < \varepsilon$, $\mathcal{J}(\boldsymbol{x}';\lambda) > \mathcal{E}(\boldsymbol{x}) + \lambda k = \mathcal{J}(\boldsymbol{x};\lambda)$.

Finally, if $\boldsymbol{x}' \in \mathcal{V}(\boldsymbol{x})$ and $\|\boldsymbol{x}' - \boldsymbol{x}\| < \varepsilon$, then $\mathcal{J}(\boldsymbol{x}';\lambda) > \mathcal{J}(\boldsymbol{x};\lambda)$. ■

OLS as a special case: When $\lambda = 0$, SBR coincides with the well known Orthogonal Least Squares (OLS) algorithm [15, 23]. The removal operation never occurs, because it automatically leads to an increase of the squared error $\mathcal{K}_{\mathcal{Q}}(0) = \mathcal{E}_{\mathcal{Q}}$. Consequently, only insertions are worth being tested.

Empty solutions: We now characterize the λ-values for which SBR yields an empty solution.

Proposition 2 *The output of SBR($\emptyset;\lambda$) is the empty set if and only if $\lambda \geqslant \lambda_{\max}$, with $\lambda_{\max} \triangleq \max_i(\langle \boldsymbol{a}_i, \boldsymbol{y}\rangle^2/\|\boldsymbol{a}_i\|^2)$.*

Proof: SBR stops during its first iteration if all the insertion trials fail: $\forall i$, $\mathcal{E}_{\{i\}} + \lambda \geqslant \mathcal{E}_\emptyset = \|\boldsymbol{y}\|^2$. Given i, $\|\boldsymbol{y} - x_i\boldsymbol{a}_i\|^2$ is minimal when $x_i = \langle \boldsymbol{a}_i, \boldsymbol{y}\rangle/\|\boldsymbol{a}_i\|^2$, leading to $\mathcal{E}_{\{i\}} = \|\boldsymbol{y}\|^2 - \langle \boldsymbol{a}_i, \boldsymbol{y}\rangle^2/\|\boldsymbol{a}_i\|^2$. Thus, SBR stops during its first iteration *if and only if* $\forall i$, $\lambda \geqslant \langle \boldsymbol{a}_i, \boldsymbol{y}\rangle^2/\|\boldsymbol{a}_i\|^2$, *i.e.*, $\lambda \geqslant \lambda_{\max}$. ■

This result allows us to design an automatic procedure which sets a number of λ-values adaptively to the data in order to compute SBR solutions at different sparsity levels (see Section VI-D).

Reduced search: Instead of testing all the replacements $\mathcal{Q}' = \mathcal{Q} \bullet i$ at each SBR iteration, it is advantageous, if possible, to explore only a subset of these n replacements. We give two ideas to reduce the number of tests: the first is an acceleration of SBR, yielding the same iterates with a reduced search. The second idea is a modification of SBR.

Given an active set $\mathcal{Q}$, a removal $\mathcal{Q}' = \mathcal{Q}\backslash\{i\}$ yields an increase of the squared error and a decrease of the penalty equal to λ. Hence, the maximum decrease of the ℓ_0-penalized cost function which can be

expected with a removal is λ: $\mathcal{K}_{\mathcal{Q}}(\lambda) - \mathcal{K}_{\mathcal{Q}'}(\lambda) \leqslant \lambda$. Consequently, if a given insertion $\mathcal{Q}' = \mathcal{Q} \cup \{i\}$ is such as $\mathcal{K}_{\mathcal{Q}}(\lambda) - \mathcal{K}_{\mathcal{Q}'}(\lambda) > \lambda$, then no removal is worth being tested. The acceleration of SBR thus consists in testing all the insertions first, and if the best insertion yields a decrease larger than λ, selecting the best insertion. Otherwise, all the removals have to be tested as stated in Table I. This acceleration does not modify the SBR iterates. However, the gain is limited when the level of sparsity is high, *i.e.*, when the number of removals to be tested is reduced.

Haugland and Zhang pointed out that in a forward-backward strategy, it can be helpful to favor removals [13, 14]. Adapted to SBR, this idea leads to a modified algorithm in which removals are tested first, and the insertions are tested only if no removal yields a decrease of $\mathcal{K}_{\mathcal{Q}}(\lambda)$. If some removals decrease $\mathcal{K}_{\mathcal{Q}}(\lambda)$, then the removal yielding the maximal decrease is selected. In our experiments, the average qualitative performance of SBR and this modified version are quite comparable (there is no obvious gain or loss of quality nor a significant saving in computation time). Thus, in the following, we keep the version of SBR presented in Table I.

IV. Implementation issues

Given the current active set $\mathcal{Q}$, an SBR iteration consists in computing the squared error $\mathcal{E}_{\mathcal{Q}'}$ for any replacement $\mathcal{Q}' = \mathcal{Q} \bullet i$, leading to the computation of $\mathcal{K}_{\mathcal{Q}'}(\lambda) = \mathcal{E}_{\mathcal{Q}'} + \lambda \mathrm{Card}\,[\mathcal{Q}']$. We first describe a basic implementation in which $\mathcal{E}_{\mathcal{Q}'}$ is computed independently of the knowledge of $\mathcal{E}_{\mathcal{Q}}$. Then, we present an efficient implementation allowing a fast update when $\mathcal{Q}$ is modified. We will denote by $k \triangleq \mathrm{Card}\,[\mathcal{Q}]$ the cardinality of the active set ($k \leqslant \min(m, n)$).

A. Basic implementation

The minimization problem (1) reduces to the unconstrained minimization of $\|\boldsymbol{y} - \boldsymbol{A}_{\mathcal{Q}}\boldsymbol{t}\|^2$ w.r.t. $\boldsymbol{t} \in \mathbb{R}^k$. Because $\boldsymbol{A}_{\mathcal{Q}}$ is full rank, this problem has a unique minimizer that reads:

$$\boldsymbol{t}_{\mathcal{Q}} \triangleq \arg\min_{\boldsymbol{t}} \|\boldsymbol{y} - \boldsymbol{A}_{\mathcal{Q}}\boldsymbol{t}\|^2 = (\boldsymbol{A}_{\mathcal{Q}}^t \boldsymbol{A}_{\mathcal{Q}})^{-1} \boldsymbol{A}_{\mathcal{Q}}^t \boldsymbol{y} \tag{13}$$

and the minimal squared error reads:

$$\mathcal{E}_{\mathcal{Q}} = \|\boldsymbol{y} - \boldsymbol{A}_{\mathcal{Q}}\boldsymbol{t}_{\mathcal{Q}}\|^2 = \|\boldsymbol{y}\|^2 - \boldsymbol{y}^t \boldsymbol{A}_{\mathcal{Q}} \boldsymbol{t}_{\mathcal{Q}}. \tag{14}$$

Finally, given the active set $\mathcal{Q}$, an SBR iteration involves the computation of $\boldsymbol{t}_{\mathcal{Q}'}$ and $\mathcal{E}_{\mathcal{Q}'}$ for all possible replacements $\mathcal{Q}' = \mathcal{Q} \bullet i$, using (13) and (14).

B. Recursive implementation

At each SBR iteration, n least-square problems of the form (13) must be solved, each requiring the inversion of the Gram matrix $\boldsymbol{G}_{\mathcal{Q}} \triangleq \boldsymbol{A}_{\mathcal{Q}}^t \boldsymbol{A}_{\mathcal{Q}}$ of size $k \times k$. The computation cost can be high since in the general case, a matrix inversion costs $\mathcal{O}(k^3)$ scalar operations. Following an idea widely spread in the subset selection literature, we propose to solve (13) in a recursive manner.

A first possibility is to use the Gram-Schmidt procedure [15, 23] which yields an orthogonal decomposition of $\boldsymbol{A}_{\mathcal{Q}} = \boldsymbol{W}\boldsymbol{U}$, where $\boldsymbol{W}$ is an $m \times k$ matrix with orthogonal columns and $\boldsymbol{U}$ is a $k \times k$ upper triangular matrix. Although it yields an efficient updating strategy when including an index into the active set (leading to the update of $\boldsymbol{A}_{\mathcal{Q}'} = [\boldsymbol{A}_{\mathcal{Q}}, \boldsymbol{a}_i]$), the Gram-Schmidt procedure does not extend with the same level of efficiency when an index removal is considered [12].

An alternative is to use the block matrix inversion lemma [24] allowing an efficient update of $\boldsymbol{G}_{\mathcal{Q}}^{-1}$ for both index insertion and removal. An efficient SMLR implementation is proposed in [20], based on the recursive update of matrices of the form $(\boldsymbol{G}_{\mathcal{Q}} + \alpha \boldsymbol{I}_k)^{-1}$. This approach can easily be adapted to SBR where the matrix to update is $\boldsymbol{G}_{\mathcal{Q}}^{-1}$ (see also [25] for the downdate step). However, we have observed numerical instabilities when the selected columns of $\boldsymbol{A}$ are highly correlated.

A more stable solution is based on the Cholesky factorization $\boldsymbol{G}_{\mathcal{Q}} = \boldsymbol{L}_{\mathcal{Q}}\boldsymbol{L}_{\mathcal{Q}}^t$, where $\boldsymbol{L}_{\mathcal{Q}}$ is a lower triangular matrix. Updating $\boldsymbol{L}_{\mathcal{Q}}$ is advantageous since it is better conditioned than $\boldsymbol{G}_{\mathcal{Q}}^{-1}$. This update can be easily done in the insertion case [26]. It is less easy for removals, since they break the triangular structure of $\boldsymbol{L}_{\mathcal{Q}}$. A recursive update of the Cholesky factor of $\boldsymbol{G}_{\mathcal{Q}}^{-1}$ was recently proposed [27]. Here, we introduce a simpler strategy relying on the factorization of $\boldsymbol{G}_{\mathcal{Q}}$.

C. Efficient strategy based on the Cholesky factorization

First, we notice that a new column $\boldsymbol{a}_i$ can be inserted at the last position in $\boldsymbol{A}_{\mathcal{Q}\cup\{i\}}$ to compute the value of $\mathcal{E}_{\mathcal{Q}\cup\{i\}}$. On the contrary, when removing a column, we do not know *a priori* the position of the column to be removed, thus it cannot be assumed to be the last column of $\boldsymbol{A}_{\mathcal{Q}}$. We will hence describe the cases where:

- a non active element $i \notin \mathcal{Q}$ is included after the other columns: $\boldsymbol{A}_{\mathcal{Q}'} = [\boldsymbol{A}_{\mathcal{Q}}, \boldsymbol{a}_i]$;
- an active element $i \in \mathcal{Q}$ is to be removed, the column $\boldsymbol{a}_i$ being in an arbitrary position.

$\boldsymbol{G}_{\mathcal{Q}}$ being a symmetric positive-definite matrix, it reads $\boldsymbol{G}_{\mathcal{Q}} = \boldsymbol{L}_{\mathcal{Q}}\boldsymbol{L}_{\mathcal{Q}}^t$ where the Cholesky factor $\boldsymbol{L}_{\mathcal{Q}}$ is a lower triangular matrix of size $k \times k$. Applying (13), the least-square minimizer rereads $\boldsymbol{t}_{\mathcal{Q}} = \boldsymbol{L}_{\mathcal{Q}}^{-t}\boldsymbol{L}_{\mathcal{Q}}^{-1}\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{y}$ where the superscript $-t$ refers to the inverse transposition operator, and (14) yields:

$$\mathcal{K}_{\mathcal{Q}}(\lambda) \;=\; \mathcal{E}_{\mathcal{Q}}(\lambda) + \lambda k \;=\; \|\boldsymbol{y}\|^2 - \|\boldsymbol{L}_{\mathcal{Q}}^{-1}\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{y}\|^2 + \lambda k. \tag{15}$$

Given $\boldsymbol{L}_{\mathcal{Q}}$, $\mathcal{O}(k^2)$ scalar operations are required to solve the triangular system $\boldsymbol{L}_{\mathcal{Q}}^{-1}(\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{y})$.

Insertion of a new column after the existing columns: Including a new column leads to $\boldsymbol{A}_{\mathcal{Q}'} = [\boldsymbol{A}_{\mathcal{Q}}, \boldsymbol{a}_i]$. Thus, the new Gram matrix can be expressed as a 2×2 block matrix:

$$\boldsymbol{G}_{\mathcal{Q}'} = \begin{bmatrix} \boldsymbol{G}_{\mathcal{Q}} & \boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{a}_i \\ (\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{a}_i)^t & \|\boldsymbol{a}_i\|^2 \end{bmatrix} \tag{16}$$

and the Cholesky factor of $\boldsymbol{G}_{\mathcal{Q}'}$ can be straightforwardly updated:

$$\boldsymbol{L}_{\mathcal{Q}'} = \begin{bmatrix} \boldsymbol{L}_{\mathcal{Q}} & \boldsymbol{0} \\ \boldsymbol{l}_{\mathcal{Q},i}^t & \alpha_{\mathcal{Q},i} \end{bmatrix}, \tag{17}$$

with $\boldsymbol{l}_{\mathcal{Q},i} = \boldsymbol{L}_{\mathcal{Q}}^{-1}\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{a}_i$ and $\alpha_{\mathcal{Q},i} = (\|\boldsymbol{a}_i\|^2 - \|\boldsymbol{l}_{\mathcal{Q},i}\|^2)^{1/2}$.

The computation of $\mathcal{K}_{\mathcal{Q}'}(\lambda)$ using (15) requires two triangular system inversions (computation of $\boldsymbol{l}_{\mathcal{Q},i}$ and computation of $\mathcal{K}_{\mathcal{Q}'}(\lambda)$). However, by computing

$$\mathcal{K}_{\mathcal{Q}'}(\lambda) - \mathcal{K}_{\mathcal{Q}}(\lambda) \;=\; \lambda - \left(\boldsymbol{l}_{\mathcal{Q},i}^t\boldsymbol{L}_{\mathcal{Q}}^{-1}\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{y}\right)^2/\alpha_{\mathcal{Q},i}^2, \tag{18}$$

the cost can be reduced up to the pre-computation of $\boldsymbol{L}_{\mathcal{Q}}^{-1}(\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{y})$ at the beginning of the SBR iteration. The computation of $\mathcal{K}_{\mathcal{Q}'}(\lambda)$ only requires one triangular system inversion (computation of $\boldsymbol{l}_{\mathcal{Q},i}$).

Removal of an arbitrary column: When removing a column $\boldsymbol{a}_i$, updating $\boldsymbol{L}_{\mathcal{Q}}$ remains possible although slightly more expensive. This idea was first developed by Ge *et al.* [27] who update the Cholesky factorization of matrix $\boldsymbol{G}_{\mathcal{Q}}^{-1}$. We adapt it to the direct (simpler) factorization of $\boldsymbol{G}_{\mathcal{Q}}$. Let I be the index such that $\boldsymbol{a}_i$ is the I-th column of $\boldsymbol{A}_{\mathcal{Q}}$ (with $1 \leqslant I \leqslant k$). $\boldsymbol{L}_{\mathcal{Q}}$ can be written in a block matrix form:

$$\boldsymbol{L}_{\mathcal{Q}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{b}^t & d & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{e} & \boldsymbol{F} \end{bmatrix}, \tag{19}$$

where the lowercase characters refer to the scalar (d) and vector quantities ($\boldsymbol{b}$, $\boldsymbol{e}$) which appear in the I-th row and in the I-th column. The computation of $\boldsymbol{G}_{\mathcal{Q}} = \boldsymbol{L}_{\mathcal{Q}}\boldsymbol{L}_{\mathcal{Q}}^t$ and the removal of the I-th row and the I-th column in $\boldsymbol{G}_{\mathcal{Q}}$ leads to

$$\boldsymbol{G}_{\mathcal{Q}'} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{F} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}^t & \boldsymbol{C}^t \\ \boldsymbol{0} & \boldsymbol{F}^t \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{e} \end{bmatrix} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{e}^t \end{bmatrix}. \tag{20}$$

By identification of this expression with the Cholesky factorization $\boldsymbol{G}_{\mathcal{Q}'} = \boldsymbol{L}_{\mathcal{Q}'}\boldsymbol{L}_{\mathcal{Q}'}^t$ and because the Cholesky factorization is unique, $\boldsymbol{L}_{\mathcal{Q}'}$ necessarily reads:

$$\boldsymbol{L}_{\mathcal{Q}'} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{X} \end{bmatrix}, \tag{21}$$

TABLE II

EFFICIENT IMPLEMENTATION OF AN SBR ITERATION.

Input: $\mathcal{Q}$, λ
Pre-computed quantities: $\boldsymbol{A}^t\boldsymbol{y}$ and $\|\boldsymbol{a}_i\|^2$ for all i
Stored quantities: $\mathcal{K}_{\mathcal{Q}}(\lambda)$, $\boldsymbol{L}_{\mathcal{Q}}$, and $\boldsymbol{L}_{\mathcal{Q}}^{-1}(\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{y})$
Set $\ell = 0$, `least_cost` $= \mathcal{K}_{\mathcal{Q}}(\lambda)$.
For $i = 1$ to n,
If $i \notin \mathcal{Q}$, `/* Insertion test */`
Compute $\boldsymbol{l}_{\mathcal{Q},i} = \boldsymbol{L}_{\mathcal{Q}}^{-1}\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{a}_i$ and $\mathcal{K}_{\mathcal{Q}'}(\lambda)$ using (18).
else, `/* Removal test */`
Update the Cholesky decomposition: $\boldsymbol{X} =$ `cholupdate`$(\boldsymbol{F}, \boldsymbol{e}, +)$.
Compute $\boldsymbol{L}_{\mathcal{Q}'}$ and $\mathcal{K}_{\mathcal{Q}'}(\lambda)$ using (21) and (15).
End if.
If $\mathcal{K}_{\mathcal{Q}'}(\lambda) <$ `least_cost`,
Set $\ell = i$, `least_cost` $= \mathcal{K}_{\mathcal{Q}'}(\lambda)$.
End if.
End for.
If $\ell \neq 0$, `/* Perform the single replacement */`
Set $\mathcal{Q}' = \mathcal{Q} \bullet \ell$, $\mathcal{K}_{\mathcal{Q}'}(\lambda) =$ `least_cost`.
Compute $\boldsymbol{L}_{\mathcal{Q}'}$ using (17) or (21), and then $\boldsymbol{L}_{\mathcal{Q}'}^{-1}(\boldsymbol{A}_{\mathcal{Q}'}^t\boldsymbol{y})$.
else,
Terminate SBR.
End if.
Output: next iterate $\mathcal{Q}' = \mathcal{Q} \bullet \ell$, $\mathcal{K}_{\mathcal{Q}'}(\lambda)$, $\boldsymbol{L}_{\mathcal{Q}'}$, and $\boldsymbol{L}_{\mathcal{Q}'}^{-1}(\boldsymbol{A}_{\mathcal{Q}'}^t\boldsymbol{y})$

where $\boldsymbol{X}$ is a lower triangular matrix satisfying $\boldsymbol{X}\boldsymbol{X}^t = \boldsymbol{F}\boldsymbol{F}^t + \boldsymbol{e}\boldsymbol{e}^t$. The problem of computing $\boldsymbol{X}$ from $\boldsymbol{F}$ and $\boldsymbol{e}$ is classical; it is known as a positive rank 1 Cholesky update and there exists a stable algorithm in $\mathcal{O}(f^2)$ operations, where $f = k - I$ is the size of $\boldsymbol{F}$ [28].

Finally, the computation of $\mathcal{K}_{\mathcal{Q}'}(\lambda)$ involves a positive Cholesky update and a triangular system inversion in (15). Thus, its overall cost is in $\mathcal{O}(k^2)$. Notice that matrix $\boldsymbol{F}$ is of size $k - I$. Therefore, the cost of a Cholesky update completely depends on the position I of the column $\boldsymbol{a}_i$ to be removed. The larger I, the more expensive is the Cholesky update.

D. Memory requirements and computation burden

The efficient (fast and stable) procedure is summarized in Table II. Given the current active set $\mathcal{Q}$, the index ℓ defining the next SBR iterate $\mathcal{Q} \bullet \ell$ is chosen according to (11) and $\boldsymbol{L}_{\mathcal{Q}\bullet\ell}$ is finally updated. No

update of the amplitudes is necessary. The actual implementation may vary depending on the size and the structure of matrix $\boldsymbol{A}$. We briefly detail the main possible implementations and their requirements in terms of storage and computation. Regarding the computation burden, we count the number of elementary operations expressed in terms of scalar multiplications since the cost of a scalar addition is negligible with respect to that of a multiplication.

When $\boldsymbol{A}$ is relatively small, the computation and storage of the full Gram matrix $\boldsymbol{A}^t\boldsymbol{A}$ prior to any SBR iteration (storage of n^2 scalar elements) avoids to recompute the vectors $\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{a}_i$ which are needed when the insertion of $\boldsymbol{a}_i$ into the active set is tested. Similarly, we store $\boldsymbol{A}^t\boldsymbol{y}$ and the values $\|\boldsymbol{a}_i\|^2$ in two 1D arrays of size n, prior to any SBR loop. The storage of the other quantities (mainly $\boldsymbol{L}_{\mathcal{Q}}$) that are being updated amounts to $\mathcal{O}(k^2)$ scalar elements and each test costs $\mathcal{O}(k^2)$ elementary operations, as it involves the inversion of a triangular system of size $k \times k$, plus a positive rank 1 Cholesky update in the removal case. This cost has to be compared with the $\mathcal{O}(k^3)$ scalar operations which are necessary when inverting the Gram matrix in the basic implementation of SBR.

When $\boldsymbol{A}$ is larger, the storage of $\boldsymbol{A}^t\boldsymbol{A}$ is no longer possible and vectors $\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{a}_i$ must be recomputed at any SBR iteration, for each insertion test $\mathcal{Q}' = \mathcal{Q} \cup \{i\}$. The computation of $\boldsymbol{A}_{\mathcal{Q}}^t\boldsymbol{a}_i$ costs km elementary operations and represents the most important cost of an insertion test. Indeed, the remaining part is in $\mathcal{O}(k^2)$ and for sparse representations, k is expected to be much lower than m. The cost of a single replacement finally amounts to $\mathcal{O}(k^2) + \mathcal{O}(km)$ elementary operations.

When the dictionary has some specific structure, the above storage limitation can be alleviated, enabling a fast implementation even when n is large. For instance, if a large number of pairs of columns of $\boldsymbol{A}$ are orthogonal to each other, $\boldsymbol{A}^t\boldsymbol{A}$ can be stored as a sparse array. Also, finite impulse response deconvolution problems enable a fast implementation since $\boldsymbol{A}^t\boldsymbol{A}$ is then a Toeplitz matrix (save north-west and/or south-east submatrices, depending on the boundary conditions). The knowledge of the auto-correlation of the impulse response is sufficient to describe most of the Gram matrix.

All these variants have been implemented (Matlab codes are available to academic users from the authors upon request). In the following, we analyze the behavior of SBR on two difficult problems, in which the dictionaries are highly correlated: the deconvolution of a sparse signal with a Gaussian impulse response (Section V) and the joint detection of discontinuities at different orders in a signal (Section VI).

V. Deconvolution of a sparse signal with a Gaussian impulse response

This is a typical problem for which SMLR was introduced [20]. It affords us to study the ability of SBR to perform an exact recovery in a simple noise-free case (separation of two Gaussian features from noise-free data) and to test the behavior of SBR in a noisy case (estimation of a larger number of Gaussian features). For simulated problems, we denote by $\boldsymbol{x}^\star$ the exact sparse signal and we generate noisy data according to $\boldsymbol{y} = \boldsymbol{y}^\star + \boldsymbol{n} = \boldsymbol{A}\boldsymbol{x}^\star + \boldsymbol{n}$, where $\boldsymbol{y}^\star = \boldsymbol{A}\boldsymbol{x}^\star$ denotes the noise-free data and $\boldsymbol{n}$ stands for the observation noise. The dictionary columns $\boldsymbol{a}_i$ are always normalized: $\|\boldsymbol{a}_i\|^2 = 1$. The signal to noise ratio (SNR) is defined by $\text{SNR} = 10\,\log(P_Y/P_N)$, where $P_Y = \|\boldsymbol{y}^\star\|^2/m$ is the average power of the noise-free data and P_N is the variance of the noise process $\boldsymbol{n}$.

A. Dictionary and simulated data

The impulse response $\boldsymbol{h}$ is a Gaussian signal of standard deviation σ, sampled on a regular grid at integer locations. It is approximated by a finite impulse response of length 6σ by thresholding the smallest values, allowing a fast implementation even for large size problems (see subsection IV-D). The deconvolution problem leads to a Toeplitz matrix $\boldsymbol{A}$ whose columns $\boldsymbol{a}_i$ are obtained by shifting the signal $\boldsymbol{h}$. The dimension of $\boldsymbol{A}$ is chosen to have any Gaussian feature resulting from the convolution $\boldsymbol{h} * \boldsymbol{x}^\star$ belonging to the observation window $\{1, \ldots, m\}$. This implies that $\boldsymbol{A}$ is slightly undercomplete ($m > n$). Denoting by $n_h = 1 + 2\text{round}(3\sigma)$ the size of the support of $\boldsymbol{h}$, the data size reads $m = n + n_h - 1$.

B. Separation of two close Gaussian features

We first analyze the ability of SBR to separate two Gaussian features ($\|\boldsymbol{x}^\star\|_0 = 2$) from noise-free data. The centers of both Gaussian features lay at a relative distance d (expressed as a number of samples) and their weights $x_i^\star$ are set to 1. We generate the corresponding noise-free data $\boldsymbol{y}^\star$ and we run SBR$(\emptyset; \lambda)$ with a number of predefined λ-values. We analyze the SBR outputs $\mathcal{Q}(\lambda; d)$ by computing their size $\text{Card}\,[\mathcal{Q}(\lambda; d)]$ and by testing if $\mathcal{Q}(\lambda; d)$ is equal to the true support $\mathcal{S}(\boldsymbol{x}^\star)$. Table III shows the results obtained for a problem of size 300×270 ($m = 300, \sigma = 5$, and $n_h = 31$) with distances equal to $d = 20$, 13, and 6 samples. The grid of λ-values for which SBR is run is common to the three tests. The maximal value λ_0 is chosen in such a way that the output $\mathcal{Q}(\lambda_0; d)$ is empty (see Proposition 2) and the other values are set according to $\lambda_j = \lambda_0/10^j$. It is noticeable that the exact recovery is always reached provided that λ is sufficiently small. This result remains true even for smaller distances (from $d = 2$). When the Gaussian features strongly overlap, *i.e.,* for $d \leqslant 13$, the size of the support obtained as output first increases while λ decreases, and then for lower λ-values, removals start to occur, enabling

TABLE III

SEPARATION OF TWO GAUSSIAN FEATURES FROM NOISE-FREE DATA WITH SBR. d STANDS FOR THE DISTANCE BETWEEN THE GAUSSIAN FEATURES. WE DISPLAY THE SIZE OF THE SUPPORT $\mathcal{Q}(\lambda; d)$ OBTAINED WITH A SEQUENCE OF DECREASING λ-VALUES $\lambda_0 > \lambda_1 > \ldots > \lambda_7$. THE LABEL * INDICATES AN EXACT RECOVERY FOR A SUPPORT OF CARDINALITY 2.

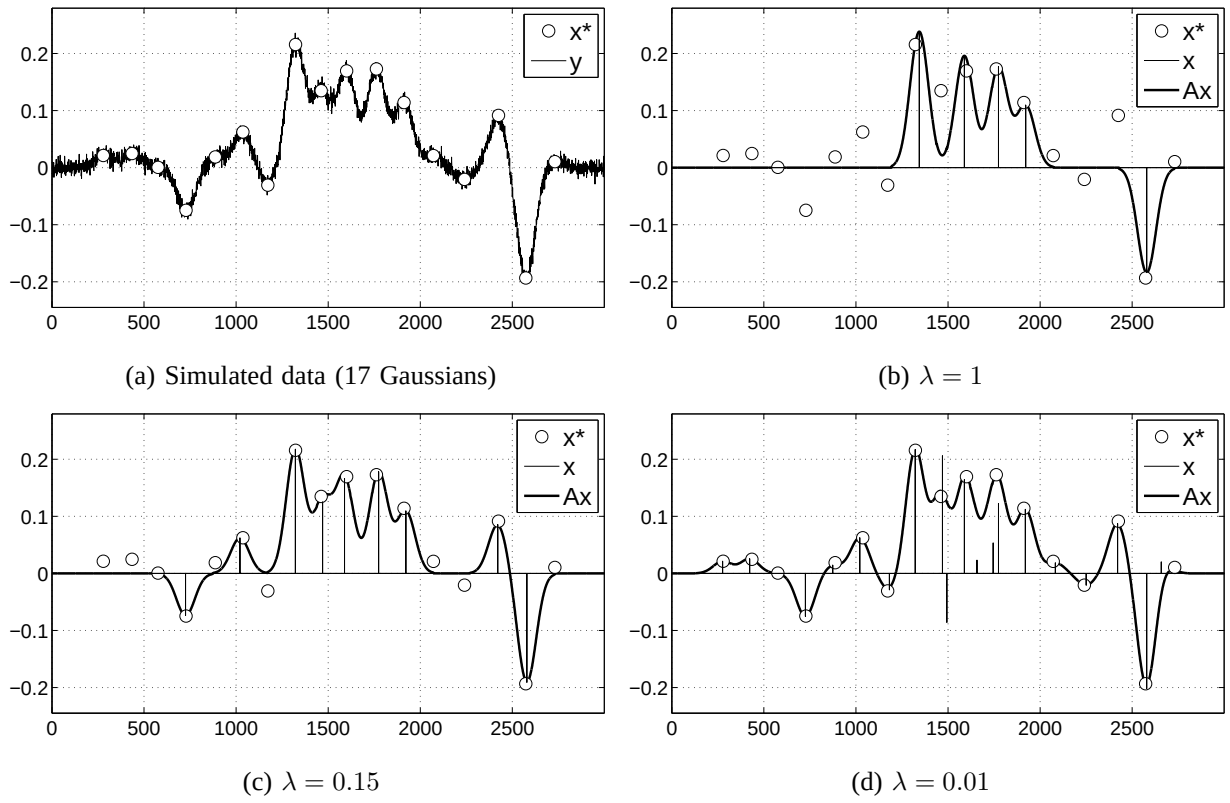
λ	λ_0	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	$\lambda \leqslant \lambda_7$
$d = 20$	0	0	$2^\star$	$2^\star$	$2^\star$	$2^\star$	$2^\star$	$2^\star$
$d = 13$	0	1	3	4	5	$2^\star$	$2^\star$	$2^\star$
$d = 6$	0	1	1	3	5	6	8	$2^\star$

the exact recovery. Similarly to SBR, forward algorithms such as OMP and OLS start by positioning a (wrong) Gaussian feature in between the two Gaussians in their first iteration but in the latter case, the early wrong detection disables an exact recovery.

C. Behavior of SBR for noisy data

We run SBR on more realistic noisy data and on a larger problem ($m = 3000$ samples). The unknown sparse signal $\boldsymbol{x}^\star$ is composed of 17 Gaussian features. The impulse response $\boldsymbol{h}$ is of size $n_h = 301$ ($\sigma = 50$) yielding an observation matrix $\boldsymbol{A}$ of size 3000×2700, and the SNR is set to 20 dB.

Fig. 1 displays the simulated data and the SBR results obtained with a few λ-values. When λ decreases, the SBR approximations are of better quality but less sparse. For large λ-values, only the main Gaussian features are found, and then, when λ decreases, the smaller features are being recovered together with unwanted features. Removals rarely occur for coarse approximations. They occur more frequently when two estimated features are overlapping and for low λ-values. On the simulation of Fig. 1, removals occur for $\lambda \leqslant 0.15$, yielding approximations that are more accurate than those obtained with OLS and for the same cardinality (the residual $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ is lower), while when $\lambda > 0.15$, the SBR output coincides with the OLS iterate of same cardinality. Although the performance of SBR is at least equal to that of OLS, the exact support of $\boldsymbol{x}^\star$ is never found. However, it must be stressed that the problem is very difficult because the data are noisy and the neighboring columns of $\boldsymbol{A}$ are highly correlated. In such difficult case, one needs to perform a wider exploration of the discrete set $\{0, 1\}^n$ by introducing moves that are more complex than single replacements. Such extensions were already proposed in the case of SMLR. One can for instance shift a detected spike x_i forwards of backwards [29] or update a block of neighboring components jointly (*e.g.*, x_i and x_{i+1}) [30].



(a) Simulated data (17 Gaussians) (b) $\lambda = 1$

(c) $\lambda = 0.15$ (d) $\lambda = 0.01$

Fig. 1. Gaussian deconvolution results. Problem of size 3000×2700 ($\sigma = 50$). (a) Generated data, with 17 Gaussian features and with SNR = 20 dB. The exact locations $\boldsymbol{x}^\star$ are labeled o. (b,c,d) SBR outputs and data approximations with empirical settings of λ. The estimated amplitudes $\boldsymbol{x}$ are shown with vertical spikes. The SBR outputs (supports) are of size 5, 9, and 19, respectively. The computation time always remains below 8 seconds (Matlab implementation).

VI. Joint detection of discontinuities at different orders in a signal

We now consider another challenging problem, the joint detection of discontinuities at different orders in a signal. We process both simulated and real data and compare the performance of SBR with respect to other sparse approximation algorithms (OMP and OLS) in terms of discontinuity estimation, approximation accuracy, and computation time. Firstly, we formulate the detection of discontinuities at a single order p as a spline approximation problem. Then, we take advantage of this formulation to introduce the joint detection problem more easily.

A. Approximation of a spline of degree p

In the continuous case, a signal is a spline of degree p with k knots *if and only if* its $(p+1)$-th derivative is a stream of k weighted Diracs [31]. In the discrete case, we introduce the dictionary $\boldsymbol{A}^p$

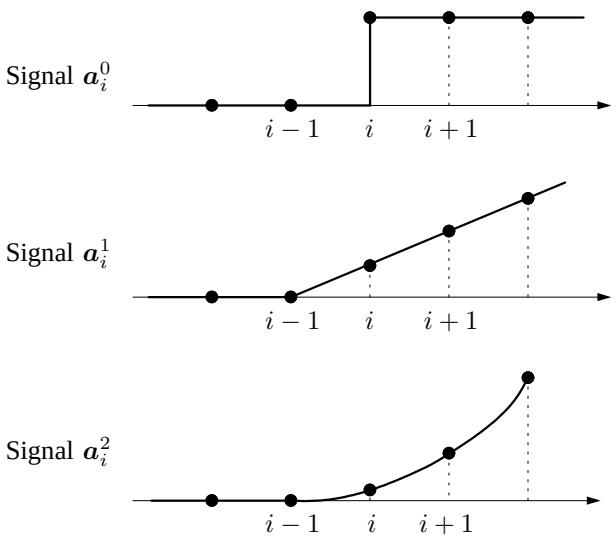


Fig. 2. Signals $\boldsymbol{a}_i^p$ related to the p-th order discontinuities at location i. $\boldsymbol{a}_i^0$ is the Heaviside step function, $\boldsymbol{a}_i^1$ is the ramp function, and $\boldsymbol{a}_i^2$ is the one-sided quadratic function. Each signal is equal to 1 at location i and its support is equal to $\{i, \ldots, m\}$.

formed of shifted versions of the one-sided power function $k \mapsto k_+^p \triangleq [\max(k, 0)]^p$ for all possible shifts (see Fig. 2). $\boldsymbol{A}^p$ represents the integration operator of degree $p+1$. Denoting by $\{1, \ldots, m\}$ the support of the data signal $\boldsymbol{y}$, the shifted signals $\boldsymbol{a}_i^p$ (for $i \in \{1, \ldots, m\}$) read

$$\forall k \in \{1, \ldots, m\},\ \boldsymbol{a}_i^p(k) = (k - i + 1)_+^p \tag{22}$$

and their support is equal to $\{i, \ldots, m\}$. Finally, we form the dictionary $\boldsymbol{A}^p = [\boldsymbol{a}_1^p, \ldots, \boldsymbol{a}_{m-p}^p]$ of size $m \times (m - p)$. It does not make sense to allow the occurrence of a p-th order discontinuity for the last samples (*i.e.,* to include $\boldsymbol{a}_i^p$ for $i > m - p$) since the spline approximation would require to reconstruct a polynomial of degree p in the range $\{i, \ldots, m\}$ from less than $p + 1$ data samples.

We address the spline approximation problem as the sparse approximation of $\boldsymbol{y}$ by the piecewise polynomial $\boldsymbol{g}^p = \boldsymbol{A}^p \boldsymbol{x}^p$ (actually, we impose as initial condition that the spline function is equal to 0 for $k \leqslant 0$). The sparse approximation consists in the detection of the discontinuity locations (also referred to as knots in the spline approximation literature) and the estimation of their amplitudes: x_i^p codes for the amplitude of a jump at location i ($p = 0$), the change of slope at location i ($p = 1$), *etc.* Here, the notion of sparsity is related to the number of discontinuity locations.

B. Approximation of a piecewise polynomial of maximum degree P

Following [31], we formulate this problem as the joint detection of discontinuities at orders $p = 0, \ldots, P$. Let us append the elementary dictionaries $\boldsymbol{A}^p$ in a global dictionary $\boldsymbol{A} = [\boldsymbol{A}^0, \ldots, \boldsymbol{A}^P]$. The

approximation $\boldsymbol{g} = \boldsymbol{A}\boldsymbol{x}$ of a given signal rereads $\boldsymbol{g} = \sum_p \boldsymbol{A}^p\boldsymbol{x}^p$ where vector $\boldsymbol{x} = \{\boldsymbol{x}^0, \ldots, \boldsymbol{x}^P\}$ gathers the p-th order amplitudes $\boldsymbol{x}^p$ for all p. When $\boldsymbol{x}$ is sparse, all vectors $\boldsymbol{x}^p$ are sparse and the approximation signal $\boldsymbol{g}$ is the sum of piecewise polynomials of degree lower than P with a limited number of pieces.

The dictionary $\boldsymbol{A}$ is overcomplete since it is of size $m \times s$, with $s = (P+1)(m - P/2) > m$ for all $P \geqslant 1$. Moreover, it is highly correlated: any column $\boldsymbol{a}_i^p$ is strongly correlated with *all* other columns $\boldsymbol{a}_j^q$ because their respective supports are the intervals $\{i, \ldots, m\}$ and $\{j, \ldots, m\}$, and hence overlap. The discontinuity detection problem is difficult, as most algorithms are very likely to position wrong discontinuities in their first iterations. For example, when approximating a signal with two discontinuities at distinct locations i and j, they start to position a first (wrong) discontinuity in between i and j, and forward algorithms cannot remove it (see Section VI-E and Fig. 5 for details).

C. Adaptation of SBR

It is important to notice that the dictionary defined above does not satisfy the unique representation property. For instance, the difference between two discrete ramps at locations i and $i+1$ yields the discrete Heaviside function at location i: $\boldsymbol{a}_i^1 - \boldsymbol{a}_{i+1}^1 = \boldsymbol{a}_i^0$. More generally, for $p > 1$, $\boldsymbol{a}_i^p - \boldsymbol{a}_{i+1}^p$ reads as a linear combination of $\boldsymbol{a}_i^0$ and $\boldsymbol{a}_{i+1}^q$ ($q = 1, \ldots, p-1$).

As mentioned in Section II, the SBR algorithm basically requires that the dictionary satisfies the URP to ensure that the Gram matrix $\boldsymbol{G}_\mathcal{Q} = \boldsymbol{A}_\mathcal{Q}^t\boldsymbol{A}_\mathcal{Q}$ is invertible, but this assumption can be relaxed provided that only full rank matrices $\boldsymbol{A}_\mathcal{Q}$ are explored. Here, SBR is slightly modified, based on the following proposition which gives a sufficient condition of invertibility of $\boldsymbol{G}_\mathcal{Q}$.

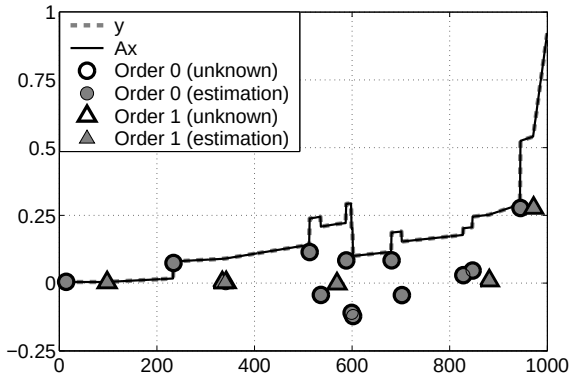
Proposition 3 *Let n_i denote the number of discontinuities $\boldsymbol{a}_i^p$, $p = 0, \ldots, P$ which are being activated at sample i, i.e., for which $x_i^p \neq 0$. Let us define the binary condition $\mathcal{C}(i)$:*

- *if $n_i = 0$, $\mathcal{C}(i) \triangleq 1$;*
- *if $n_i \geqslant 1$, $\mathcal{C}(i) \triangleq \big(\forall j \in \{1, \ldots, n_i - 1\},\, n_{i+j} = 0\big)$.*

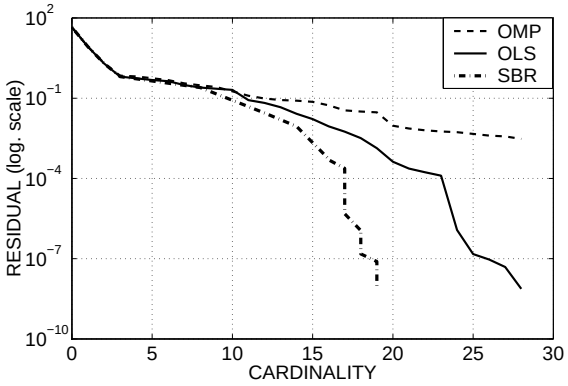
If $\mathcal{Q}$ is such that for all i, $\mathcal{C}(i) = 1$, then $\boldsymbol{G}_\mathcal{Q}$ is invertible.

Proposition 3 is proved in Appendix A.

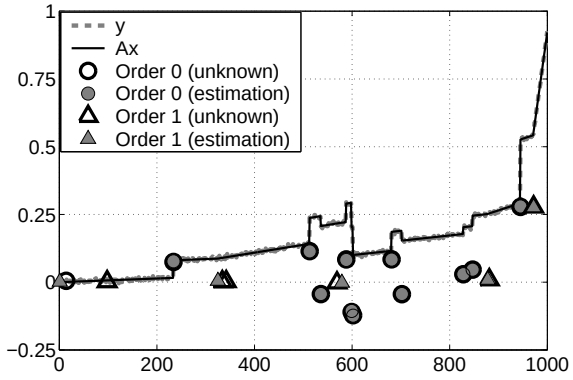
Basically, it states that we can allow several discontinuities to be active at the same location i, but then, the next samples $i+1, \ldots, i+n_i-1$ must not host any discontinuity. This condition ensures that there are at most n_i discontinuities in the interval $\{i, \ldots, i+n_i-1\}$ of length n_i. The adaptation of SBR consists in testing insertions into the current active set only if the above condition remains true.



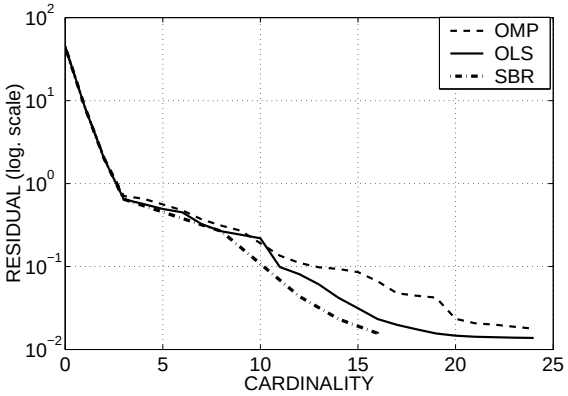
(a) Noise-free data and SBR approximation



(b) "ℓ_2-ℓ_0" curves (noise-free data)



(c) Noisy data (SNR = 35 dB) and SBR approximation



(d) "ℓ_2-ℓ_0" curves (noisy data)

Fig. 3. Joint detection of discontinuities at orders 0 and 1. The dictionary is of size 1000×1999 and the data signal $\boldsymbol{y}$ includes 18 discontinuities. The true and estimated discontinuity locations are represented with unfilled black and filled gray labels. The shape of the labels (circular or triangular) indicates the discontinuity order. The dashed gray and solid black curves represent the data signal $\boldsymbol{y}$ and its approximation $\boldsymbol{A}\boldsymbol{x}$ for the least λ-value. (a) Signal approximation from noise-free data. The recovery is exact and both curves are superimposed. (b) "ℓ_2-ℓ_0" curves showing the squared residual versus the cardinality for SBR, OLS, and OMP. The SBR performance is expressed only for the λ-values that are larger than λ_{20}, because below this value, the recovery is exact and the log-residual is equal to $-\infty$. (c,d) Similar results for noisy data (SNR = 35 dB).

D. Numerical simulations

Let us first consider the case $P = 1$, leading to the joint detection of discontinuities of order zero and one, *i.e.*, the piecewise affine approximation problem. We simulate noise-free data $\boldsymbol{y}^\star = \boldsymbol{A}\boldsymbol{x}^\star$ of size $m = 1000$ and with $\|\boldsymbol{x}^\star\|_0 = 18$ discontinuities (see Fig. 3(a)). We use the result of Proposition 2 to compute the value $\lambda_{\max}$ below which the SBR output is not the empty set, and we run SBR with $\lambda_j = \lambda_{\max}\, 10^{-j/2}$ for $j = 0, \ldots, J_{\max}$, with $J_{\max} = 20$. These λ-values provide a sequence of solutions at different sparsity levels. For comparison purpose, we also run 27 iterations of OMP and OLS.

The SBR approximation shown in Fig. 3(a) corresponds to the least λ-value $\lambda_{J_{\max}}$. The recovery is exact. The "ℓ_2-ℓ_0" curves represented on Fig. 3(b) express the squared residual $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ versus the cardinality $\|\boldsymbol{x}\|_0$ for the output of each algorithm. Whatever the level of sparsity, SBR yields the least residual. We did the same experiment with noisy data $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^\star + \boldsymbol{n}$, setting the SNR to 35 dB (see Figs. 3(c,d)). Again, the "ℓ_2-ℓ_0" curve corresponding to SBR lays below the OMP and OLS curves. For most sparsity levels, SBR outperforms the other algorithms. Note that for more noisy data (*e.g.*, SNR = 15 dB), the SBR and OLS curves coincide and still lay below the OMP curve.

E. AFM data processing

In Atomic Force Microscopy (AFM), a force curve measures the interatomic forces exerting between a probe associated to a cantilever and a nano-object. More precisely, the recorded signal $z \mapsto y(z)$ shows the force evolution versus the probe-sample distance z, expressed in nanometers. Researching discontinuities (location, order, and amplitude) in a force curve is a challenging task because they are used to provide a precise characterization of the physico-chemical properties of the nano-object (topography, energy of adhesion, *etc.*) [32].

The data displayed on Fig. 4(a) are related to a bacterial cell *Shewanella putrefaciens* laying in aqueous solution, interacting with the tip of the AFM probe [33]. A force curve is recorded in two steps. Firstly, the tip is positioned far away from the sample. It is moved towards the sample until the contact is reached and the surface of the bacterial cell is deformed (approach curve). Secondly, the tip is retracted from the sample until it loses contact. The experimental curve shown on Fig. 4(a) is a retraction curve composed of $m = 2167$ force measurements. From right to left, three regions of interest can be distinguished. The linear region on the right characterizes the rigid contact between the probe and the sample. It describes the mechanical interactions of the cantilever and the sample. The rigid contact is maintained until $z \approx -2840$ nm. The interactions occurring in the interval $z \in [-3050, -2840]$ nm are adhesion forces during the retraction of the tip. In the flat part on the left, no interaction occurs as the cantilever has lost contact with the sample.

We search for the discontinuities of orders 0, 1, and 2. Similarly to the processing of simulated data, we run SBR with $J_{\max} = 14$ λ-values and we run OLS and OMP until iteration 41. For each algorithm, we plot the "ℓ_2-ℓ_0" curve representing the squared residual $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ versus the cardinality $\|\boldsymbol{x}\|_0$, and a curve displaying the time of reconstruction versus the cardinality (see Figs. 4(b,c)). These figures show that the performance of SBR is at least equal and sometimes better than that of OLS. Both algorithms yield results that are far more accurate than OMP at the price of a larger computation time. However, notice

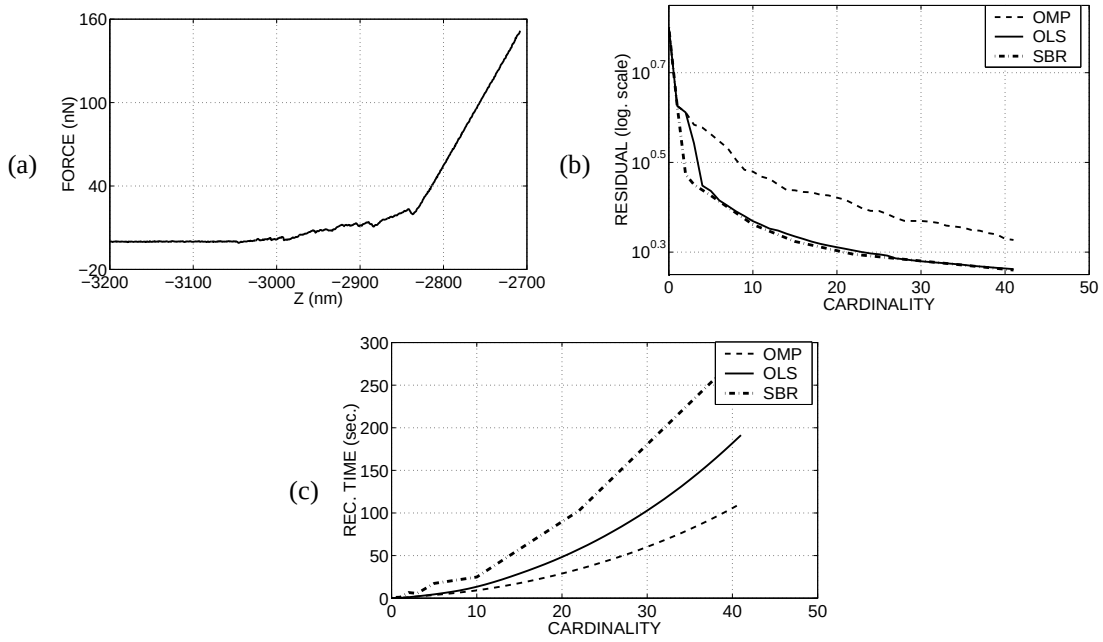
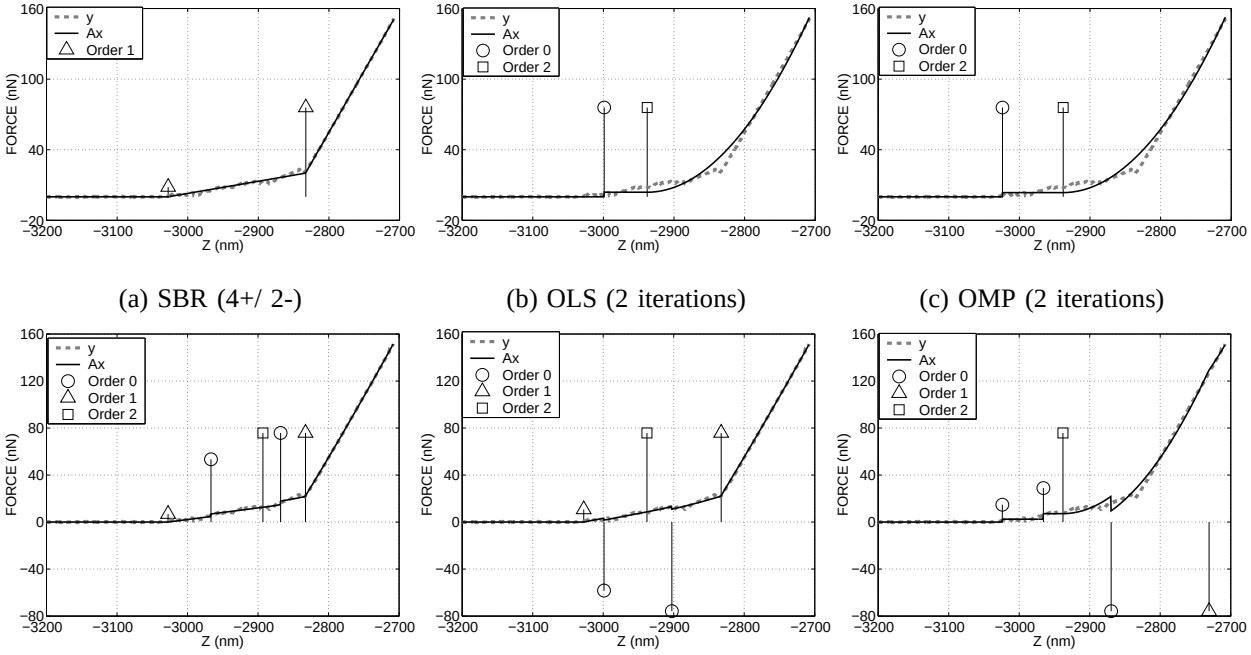


Fig. 4. AFM data processing: joint detection of discontinuities at orders 0, 1, and 2 (problem of size 2167×6498). (a) Experimental data showing the force evolution versus the probe-sample distance z. (b) Squared residual versus cardinality for the SBR, OLS, and OMP outputs. (c) Time of reconstruction versus cardinality for the three algorithms.

that the recorded computation time always remains below 350 seconds in the case of SBR (in a Matlab implementation taking advantage of the block Toeplitz structure of the dictionary: see Section IV-D).

Fig. 5 displays the approximations yielded by the three algorithms for supports of cardinality 2 and 5. SBR actually runs during 6 iterations (4 insertions and 2 removals are performed) to reach a support of cardinality 2. This approximation is very accurate compared to the OMP and OLS results obtained after 2 iterations (Figs. 5(a,b,c)). SBR provides a very precise localization of both first order discontinuities, which are crucial information for the physical interpretation of the data. On the contrary, OLS does not succeed after two iterations; it is able to locate accurately both discontinuities once 5 iterations have been performed (the desired discontinuities are the first and the last ones among the 5) while OMP fails even after 5 iterations (Figs. 5(d,e,f)). The residual yielded by the SBR approximation of cardinality 5 remains lower than the corresponding OLS and OMP residuals.

In order to better understand the forward and backward moves (respectively, insertions and removals) occurring during the SBR iterations, we display in Table IV the residual $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ and the cardinality of each iterate for both SBR executions. Because SBR is a descent algorithm, the penalized cost $\mathcal{J}(\boldsymbol{x}; \lambda)$ keeps decreasing but when a removal occurs, $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ increases. For the coarse approximation of



(a) SBR (4+/ 2-) (b) OLS (2 iterations) (c) OMP (2 iterations)

(d) SBR (7+/ 2-) (e) OLS (5 iterations) (f) OMP (5 iterations)

Fig. 5. AFM data processing: joint detection of discontinuities at orders 0, 1, and 2. The estimated discontinuities $\boldsymbol{x}$ are represented with vertical spikes and with a label indicating the discontinuity order. The dashed gray and solid black curves represent the data signal $\boldsymbol{y}$ and its approximation $\boldsymbol{Ax}$, respectively. (a) SBR output of cardinality 2: 4 insertions and 2 removals have been done ($\lambda = 2085$). (b,c) OLS and OMP outputs after 2 iterations. (d,e,f) Same simulation with a lower λ-value ($\lambda = 66$). The SBR output is of cardinality 5 (7 insertions and 2 removals) and we stop OLS and OMP after 5 iterations.

TABLE IV

BEHAVIOR OF THE SBR ITERATES FOR BOTH APPROXIMATIONS OF FIG. 5(A,D). THE TABLES DISPLAY THE SQUARED ERROR $\|\boldsymbol{y} - \boldsymbol{Ax}\|^2$ VERSUS THE CARDINALITY $\|\boldsymbol{x}\|_0$ FOR EACH ITERATE. i STANDS FOR THE ITERATION INDEX.

SBR ($\lambda = 2085$, 4+/ 2-)

i	$\|\boldsymbol{x}\|_0$	Error
0	0	2101.408
1	1	16.870
2	**2**	**12.266**
3	3	3.074
4	4	642
5	3	663
6	**2**	**938**

SBR ($\lambda = 66$, 7+/2-)

i	$\|\boldsymbol{x}\|_0$	Error
0	0	2101.408
1	1	16.870
2	2	12.266
3	3	3.074
4	4	642
5	3	663
6	4	555
7	**5**	**480**
8	4	532
9	**5**	**464**

Fig. 5(a), the residual is much smaller than the residual yielded by OLS since two configurations of cardinality 2 have been explored (see the left table and the rows in bold). We observed the same behavior for the finer approximation of cardinality 5 (right table).

F. Discussion

In the two previous subsections, we chose to compare SBR with OMP and OLS. We did not consider simpler algorithms like MP which are well suited to rapidly solve easier problems in which the dictionary columns are almost orthogonal. Because SBR involves more complex operations (matrix inversions), we chose to compare it with OMP and OLS which also require to solve at least one least-square problem per iteration. Their target is to provide results which are more accurate than the MP approximations in the case of difficult problems.

Up to our knowledge, the only minimization algorithm dedicated to the ℓ_0-penalized cost function $\mathcal{J}(\boldsymbol{x};\lambda) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 + \lambda\|\boldsymbol{x}\|_0$ is Blumensath and Davies' Iterative Hard Thresholding (IHT) [9]. It relies on gradient based iterations of the form $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{A}^t(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x})$, followed by the thresholding to 0 of all the non-zero components x_i such that $|x_i| \leqslant \lambda^{0.5}$. We tested this version of IHT on both deconvolution and discontinuity detection problems and we observed that it is less efficient than the standard version of IHT, dedicated to the ℓ_0-constrained problem. In the constrained version, the k components $|x_i|$ having the largest amplitudes are kept and the others are being thresholded. Generally speaking, we observed that IHT is competitive when the correlation between any pair of dictionary columns is limited, but for highly correlated dictionaries, IHT needs a very large number of iterations ($\mathcal{O}(m^2)$) to reach convergence. SBR seems to be better suited to such difficult problems. It is less sensitive to the initial solution and "skips" some local minimizers having a large cost $\mathcal{J}(\boldsymbol{x};\lambda)$. We here recall that according to Proposition 1, each SBR iterate is almost surely a local minimizer of $\mathcal{J}(\boldsymbol{x};\lambda)$.

In order to link up our approach to the forward-backward algorithm of [14], we also tested an OMP-like adaptation of SBR in which only one least-square problem is solved per iteration, instead of n. This adaptation consists in replacing the selection rule (11) in the following way. When an insertion $\mathcal{Q} \cup \{i\}$ is tested, all the active components x_j are kept constant and x_i is set to the minimizer of $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{\mathcal{Q}} - x_i\boldsymbol{a}_i\|^2$. This leads to an approximation of $\mathcal{K}_{\mathcal{Q}\bullet i}(\lambda)$ without solving any least-square problem. Similarly, the removal test consists in setting x_i to 0 and leaving the other components x_j unchanged. In brief, this adapted version is an algorithm aimed at the minimization of $\mathcal{J}(\boldsymbol{x};\lambda)$ at a cost which is comparable to that of OMP. In all our trials, SBR performs better than the OMP-like version except in very simple cases (limited correlation between the columns $\boldsymbol{a}_i$) where both versions yield the same

result. The performance of the OMP-like version fluctuates below or above that of OMP but is almost always far less accurate than the OLS and SBR approximations.

VII. Conclusion

We have evaluated the SBR algorithm on two problems in which the dictionary columns are highly correlated. SBR provides solutions which are at least as accurate as the OLS solutions and sometimes more accurate, with a cost of the same order of magnitude. For these difficult problems, MP and OMP provide poor approximations within a lower computation time. Compared to OLS, we believe that performing removals is the price to pay if one expects an enhanced quality approximation. Although removals rarely occur in comparison with the insertions, they play an important role in the enhancement of the approximation.

In the proposed approach, the main difficulty relies in the choice of the λ-value. If a specific sparsity level or approximation residual is desired, one can resort to a trial and error procedure in which a number of λ-values are tried until the desired approximation level is found. In [34], we sketched a continuation version in which a series of SBR solutions are successively estimated with a decreasing level of sparsity λ, and the λ-values are recursively computed. The first λ-value is set to $\lambda_0 = +\infty$, and at a given value λ_i, the initial solution (input of SBR) is set to the SBR output at $\lambda = \lambda_{i-1}$. This continuation version provides promising results and will be the subject of a future extended contribution. A similar perspective is actually proposed by Zhang to generalize his FoBa algorithm in a path-following algorithm (see the discussion section in [14]).

Another important perspective is to investigate whether SBR can guarantee exact recovery in the noise-free case under some conditions on matrix $\boldsymbol{A}$ and on the unknown sparse signal $\boldsymbol{x}^\star$. In the simulations done in Section V, we observed that SBR is able to exactly recover two close Gaussian features whatever their distance, provided that the hyperparameter λ is sufficiently small. This promising result is a first step towards a more general theoretical study. The FoBa algorithm [14] yields exact recovery results for problems satisfying the Restricted Isometry Property (RIP). Since the structure of SBR is somewhat close to that of FoBa, we expect that SBR shares similar theoretical properties. We will investigate whether the proofs provided in [14] are extendable to SBR.

Appendix A
Proof of Proposition 3

The following lemma is a key element to prove Proposition 3.

Lemma 1 *Consider an active set $\mathcal{Q}$ satisfying the condition of Proposition 3, and let $i^- = \min\{i \,|\, n_i > 0\}$ denote the lowest location of an active entry. Up to a reordering of the columns, $\boldsymbol{A}_\mathcal{Q}$ rereads $\boldsymbol{A}_\mathcal{Q} = [\boldsymbol{A}_{i^-},\, \boldsymbol{A}_{\mathcal{Q}\setminus\{i^-\}}]$ with obvious notations. If $\boldsymbol{A}_{\mathcal{Q}\setminus\{i^-\}}$ is full rank, then $\boldsymbol{A}_\mathcal{Q}$ is also full rank.*

Proof: Let $I = n_{i^-}$ denote the number of discontinuities at location i^- and let $0 \leqslant p_1 < p_2 < \ldots < p_I$ denote their order, sorted in the ascending order. Suppose that there exist two families of scalars $\{\mu_{i^-}^{p_1}, \ldots, \mu_{i^-}^{p_I}\}$ and $\{\mu_i^p \,|\, i \neq i^- \text{ and } i \text{ is active at order } p\}$ such that

$$\sum_{j=1}^{I} \mu_{i^-}^{p_j}\, \boldsymbol{a}_{i^-}^{p_j} + \sum_{i\neq i^-} \sum_{p} \mu_i^p\, \boldsymbol{a}_i^p = \boldsymbol{0}. \tag{23}$$

Let us show that all μ-values are then equal to 0.

Rewriting the first I nonzero equations in this system and because $\mathcal{Q}$ satisfies the condition of Proposition 3, we have, for all $k \in \{i^-, \ldots, i^- + I - 1\}$, $\sum_{j=1}^{I} \mu_{i^-}^{p_j} (k + i^- - 1)^{p_j} = 0$. In other words, the polynomial $F(X) = \sum_{j=1}^{I} \mu_{i^-}^{p_j} X^{p_j}$ has I positive roots. It is shown in [35, p. 76] that a non-zero polynomial formed of I monomials of different degree has at most $I-1$ positive roots. Therefore, F is the zero polynomial and all scalars $\mu_{i^-}^{p_j}$ are 0. We deduce from (23) and from the full rankness of $\boldsymbol{A}_{\mathcal{Q}\setminus\{i^-\}}$ that $\mu_i^p = 0$ for all (i,p).

We have shown that the column vectors of $\boldsymbol{A}_\mathcal{Q}$ are linearly independent, *i.e.,* that $\boldsymbol{A}_\mathcal{Q}$ is full rank. ■

The proof of Proposition 3 directly results from a recursive application of Lemma 1. Starting from the empty set, all the indices, sorted by decreasing order, are included successively.

REFERENCES

[1] B. K. Natarajan, "Sparse approximate solutions to linear systems", *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, Apr. 1995.

[2] B. D. Rao, K. Engan, S. F. Cotter, J. Palmer, and K. Kreutz-Delgado, "Subset selection in noise based on diversity measure minimization", *IEEE Trans. Signal Processing*, vol. 51, no. 3, pp. 760–770, Mar. 2003.

[3] G. H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm", *IEEE Trans. Signal Processing*, vol. 57, no. 1, pp. 289–301, Jan. 2009.

[4] 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, 1998.

[5] D. L. Donoho and Y. Tsaig, "Fast solution of l_1-norm minimization problems when the solution may be sparse", *IEEE Trans. Inf. Theory*, vol. 54, no. 11, pp. 4789–4812, Nov. 2008.

[6] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression", *Annals Statist.*, vol. 32, no. 2, pp. 407–451, 2004.

[7] J.-J. Fuchs, "Recovery of exact sparse representations in the presence of bounded noise", *IEEE Trans. Inf. Theory*, vol. 51, no. 10, pp. 3601–3608, Oct. 2005.

[8] 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.

[9] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations", *The Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, Dec. 2008.

[10] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries", *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

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

[12] C. Couvreur and Y. Bresler, "On the optimallity of the backward greedy algorithm for the subset selection problem", *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 3, pp. 797–808, Feb. 2000.

[13] D. Haugland, *A Bidirectional Greedy Heuristic for the Subspace Selection Problem*, vol. 4638 of *Lecture Notes in Computer Science*, pp. 162–176, Springer Verlag, Berlin, Germany, Engineering stochastic local search algorithms. Designing, implementing and analyzing effective heuristics edition, 2007.

[14] T. Zhang, "Adaptive forward-backward greedy algorithm for learning sparse representations", Tech. Rep., Rutgers Statistics Department, Apr. 2008.

[15] 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.

[16] T. Blumensath and M. E. Davies, "On the difference between orthogonal matching pursuit and orthogonal least squares", Tech. Rep., University of Edinburgh, Mar. 2007.

[17] J. J. Kormylo and J. M. Mendel, "Maximum-likelihood detection and estimation of Bernoulli-Gaussian processes", *IEEE Trans. Inf. Theory*, vol. 28, pp. 482–488, 1982.

[18] J. M. Mendel, *Optimal Seismic Deconvolution*, Academic Press, New York, NY, 1983.

[19] Y. Goussard, G. Demoment, and J. Idier, "A new algorithm for iterative deconvolution of sparse spike trains", in *Proc. IEEE ICASSP*, Albuquerque, NM, Apr. 1990, pp. 1547–1550.

[20] F. Champagnat, Y. Goussard, and J. Idier, "Unsupervised deconvolution of sparse spike trains using stochastic approximation", *IEEE Trans. Signal Processing*, vol. 44, no. 12, pp. 2988–2998, Dec. 1996.

[21] Q. Cheng, R. Chen, and T.-H. Li, "Simultaneous wavelet estimation and deconvolution of reflection seismic signals", *IEEE Trans. Geosci. Remote Sensing*, vol. 34, pp. 377–384, Mar. 1996.

[22] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm", *IEEE Trans. Signal Processing*, vol. 45, no. 3, pp. 600–616, Mar. 1997.

[23] S. Chen and J. Wigger, "Fast orthogonal least squares algorithm for efficient subset model selection", *IEEE Trans. Signal Processing*, vol. 43, no. 7, pp. 1713–1715, July 1995.

[24] R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge University Press, 1985.

[25] S. J. Reeves, "An efficient implementation of the backward greedy algorithm for sparse signal reconstruction", *IEEE Signal Processing Letters*, vol. 6, no. 10, pp. 266–268, Oct. 1999.

[26] 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.

[27] D. Ge, J. Idier, and E. Le Carpentier, "Enhanced sampling schemes for MCMC based blind Bernoulli-Gaussian deconvolution", Tech. Rep., Institut de Recherche en Communication et Cybernétique de Nantes, Sept. 2009.

[28] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, “Methods for modifying matrix factorizations”, *Mathematics of Computation*, vol. 28, no. 126, pp. 505–535, Apr. 1974.

[29] C. Y. Chi and J. M. Mendel, “Improved maximum-likelihood detection and estimation of Bernoulli-Gaussian processes”, *IEEE Trans. Inf. Theory*, vol. 30, pp. 429–435, Mar. 1984.

[30] M. Allain and J. Idier, “Efficient binary reconstruction for non-destructive evaluation using gammagraphy”, *Inverse Problems*, vol. 23, no. 4, pp. 1371–1393, Aug. 2007.

[31] M. Vetterli, P. Marziliano, and T. Blu, “Sampling signals with finite rate of innovation”, *IEEE Trans. Signal Processing*, vol. 50, no. 6, pp. 1417–1428, June 2002.

[32] H.-J. Butt, B. Cappella, and M. Kappl, “Force measurements with the atomic force microscope: Technique, interpretation and applications”, *Surface Science Reports*, vol. 59, no. 1–6, pp. 1–152, Oct. 2005.

[33] F. Gaboriaud, B. S. Parcha, M. L. Gee, J. A. Holden, and R. A. Strugnell, “Spatially resolved force spectroscopy of bacterial surfaces using force-volume imaging”, *Colloids and Surfaces B: Biointerfaces*, vol. 62, no. 2, pp. 206–213, Apr. 2008.

[34] J. Duan, C. Soussen, D. Brie, and J. Idier, “A continuation approach to estimate a solution path of mixed L2-L0 minimization problems”, in *Signal Processing with Adaptive Sparse Structured Representations (SPARS workshop)*, Saint-Malo, France, Apr. 2009, pp. 1–6.

[35] F. R. Gantmacher and M. G. Krein, *Oscillation matrices and kernels and small vibrations of mechanical systems*, AMS Chelsea Publishing, Providence, RI, revised edition, 2002.