Received: 14 June 2022 | Revised: 21 October 2022 | Accepted: 29 December 2022

DOI: 10.1002/nla.2490

RESEARCH ARTICLE

WILEY

The Lawson-Hanson algorithm with deviation maximization: Finite convergence and sparse recovery

Monica Dessole[1,2] | **Marco Dell'Orto**[1] | **Fabio Marcuzzi**[1]

[1]Department of Mathematics "Tullio Levi Civita", University of Padua, Padua, Italy

[2]Leonardo Labs, Leonardo S.p.A., Genoa, Italy

Correspondence
Fabio Marcuzzi, Department of Mathematics "Tullio Levi Civita", University of Padua, Via Trieste 63, 35121 Padua, Italy.
Email: marcuzzi@math.unipd.it

Funding information
University of Padova, Grant/Award Number: BIRD192932

Summary

The Lawson-Hanson with Deviation Maximization (LHDM) method is a block algorithm for the solution of NonNegative Least Squares (NNLS) problems. In this work we devise an improved version of LHDM and we show that it terminates in a finite number of steps, unlike the previous version, originally developed for a special class of matrices. Moreover, we are concerned with finding sparse solutions of underdetermined linear systems by means of NNLS. An extensive campaign of experiments is performed in order to evaluate the performance gain with respect to the standard Lawson-Hanson algorithm. We also show the ability of LHDM to retrieve sparse solutions, comparing it against several ℓ_1-minimization solvers in terms of solution quality and time-to-solution on a large set of dense instances.

KEYWORDS

block pivoting, nonnegative least squares, sparse recovery

1 | INTRODUCTION

Nonnegative least squares (NNLS) problems arise in many applications where data points can be represented as nonnegative linear combinations of some meaningful components. Such problems often arise in signal and image processing and they are core problems in more complex computations, such as nonnegative matrix and tensor decompositions.[1,2] In this paper, we are concerned with finding sparse solutions of underdetermined linear systems by means of NNLS. Indeed, when dealing with underdetermined systems of equations, the nonnegativity constraint is known to naturally enhance sparsity of the solution, that is the solution attained has few nonzeros.[3-6] An important outcome of this body of work is that nonnegativity alone may attain a satisfactory sparse recovery. Moreover, NNLS solvers can be adopted to solve unconstrained least squares problems with minor adjustments. Over the last two decades, sparsity has become one of the most relevant topics in signal processing, as it describes the phenomenon where high dimensional data can be expressed with few measurements and it results in finding a sparse solution to underdetermined systems of equations. This problem is highly nonconvex and it is NP-hard in general, although it is well known that ℓ_1-minimization leads to the sparsest solution under suitable assumptions. This fact is known as $\ell_0 - \ell_1$ equivalence and it has been found empirically[7] and theoretically.[8,9] The $\ell_0 - \ell_1$ equivalence holds for some restricted classes of matrices, while for general problems, one could solve a regularized ℓ_1 least squares problem by for example, the well-known LASSO model.[10] Algorithms that cope with such problems are usually tested on instances whose matrices are very well-conditioned and whose solution vectors are very sparse, that is, can be expressed as linear combinations of few columns of the matrices. In this paper we introduce an improved NNLS solver for sparse recovery, which is best suited to instances with larger condition number and where the optimal solution has a higher degree of sparsity, as shown in the numerical section.

The standard NNLS problem can be formulated as follows: given $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$ find $\mathbf{x} \in \mathbb{R}^n$ that solves

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2, \qquad \text{s.t. } \mathbf{x} \geq 0. \tag{1}$$

The first algorithm devised for NNLS has been the Lawson-Hanson algorithm.[11] It is an active set method, that is, an optimal solution is built by identifying at each iteration a new active constraint in such a way that the objective value strictly decreases while the current solution is kept within the feasible region. The algorithm selects one column at a time and then computes the current iterate by solving a strictly convex unconstrained least square problem via QR updates/downdates, therefore it mainly relies on BLAS–2 operations, resulting in poor performance. In general, column pivoting makes it more difficult to achieve high performances in QR computation.[12-16]

Since the seminal work of Lawson and Hanson, many modifications have been proposed in order to improve it. The algorithm "fast NNLS" (FNNLS)[17] is a variation specifically designed for use in nonnegative tensor decompositions and it performs well with multiple right-hand-sides problems, which is not the case here discussed, thus we omit a comparison. The algorithm "fast combinatorial NNLS" (FCNNLS)[18] is also designed for the specific case of a large number of right-hand sides. Principal block pivoting (PBP)[19] deals with linear complementary problems and it manipulates multiple columns at a time. Unlike the Lawson Hanson method, the sequence of iterates produced does not necessarily fall into the feasible region. This algorithm only deals with matrices with full column rank, therefore this algorithm may fail in sparse recovery problems in which matrices are underdetermined. The Lawson-Hanson algorithm does not suffer from this drawback as it solves a sequence of strictly convex subproblems even in the underdetermined setting.

In this work, we address the NNLS problem in which the objective function matrix is dense and we present a modified Lawson-Hanson algorithm in which multiple column selection is performed in order to exploit BLAS–3 operations for efficiency. The column selection is based on the "deviation maximization" pivoting,[20] which identifies a block of sufficiently linearly independent columns for QR computations. Here, the column selection takes into account first order information to solve the nonnegative least squares problem. The proposed algorithm is named Lawson-Hanson with Deviation Maximization (LHDM). An early version of this algorithm[21] exhibits a significant performance gain with respect to the standard Lawson-Hanson algorithm when dealing with underdetermined systems, without compromising the sparsity of the retrieved solution. However, the algorithm there presents a main drawback. The computation is not prevented to fall in an infinite loop without converging to a solution. In the present paper, we introduce a modification which ensures the termination in a finite number of steps with a similar performance gain. Moreover, we assess its sparsity recovery ability against several ℓ_1-minimization solvers, extending a comparison that has been done in the recent literature.[22] Numerical tests are performed on a large set of instances with respect to the degree of sparsity of the solution vectors and the condition numbers of the matrices.

The rest of the paper is organized as follows. In Section 2, we formally introduce the NNLS problem and the Lawson-Hanson algorithm. We present the deviation maximization algorithm for column selection and the modified Lawson-Hanson with Deviation Maximization algorithm. We provide a theoretical result about finite convergence of LHDM by generalizing the analogous result for the standard Lawson-Hanson. In Section 3, we introduce the ℓ_0-minimization problem as sparse recovery and the convex ℓ_1-minimization problem. We recall results that ensure the so-called $\ell_0 - \ell_1$ equivalence and how arbitrary signed sparse recovery can be attained by NNLS solvers. In Section 5, we present a comparison of LHDM against several ℓ_1-minimization solvers in terms of performance and solution found. The results are carried out with an extensive campaign of experiments. Finally, Section 6 concludes the paper.

2 | ACTIVE SET ALGORITHMS FOR NONNEGATIVE LEAST SQUARES

Let $A = (\mathbf{a}_1 \ \dots \ \mathbf{a}_n)$ be a matrix of size $m \times n$, $\mathbf{b}$ a vector of length m. When not otherwise stated, $\|\cdot\|$ denotes the 2-norm. We make use of the so called "colon notation", that is for any matrix A of size $m \times n$ we denote by $A(k : l, p : q)$ the submatrix of A obtained considering the entries with row indices $k \leq i \leq l$ and column indices $p \leq j \leq q$. When using colon notation, we write $A(:, p : q)$ ($A(k : l, :)$) as a shorthand for $A(1 : m, p : q)$ ($A(k : l, 1 : n)$). For any index set $S \subseteq \{1, \dots, n\}$, A_S denotes the submatrix of A obtained by considering columns indexed in S. In this work, we address the case in which A is a dense matrix, that is, it is stored as a full array with mn entries. The case in which A is sparse would require the design of a specific implementation in order to exploit the presence of many null entries. The gradient of the

objective function (1) evaluated at a point $\mathbf{x}$, that is, the opposite of the steepest descent direction at $\mathbf{x}$, is directed as the following vector

$$A^TA\mathbf{x} - A^T\mathbf{b} = -A^T(\mathbf{b} - A\mathbf{x}) = -A^T\mathbf{r}(\mathbf{x}),$$

where $\mathbf{r}$ is the residual function $\mathbf{r}(\mathbf{x}) = \mathbf{b} - A\mathbf{x}$. The $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq 0\}$ is called feasible region, and a point $\mathbf{x} \in \Omega$ is said to be feasible. A vector $\mathbf{s} \in \mathbb{R}^n$ is a *feasible direction* at $\mathbf{x}$ if there exists $\varepsilon > 0$ such that $\mathbf{x} + \varepsilon\mathbf{s} \geq 0$. Notice that if $\mathbf{s}$ is a feasible direction at $\mathbf{x}$ and $s_i < 0$, then we must have $x_i > 0$. Karush-Kuhn-Tucker conditions[11] for the convex problem (1) can be stated as the following linear complementarity problem

$$\mathbf{w} = A^T(\mathbf{b} - A\mathbf{x}),\ \mathbf{w} \leq 0,\ x_i > 0 \ \text{ for } \ i \in P,\ x_i = 0 \ \text{ for } \ i \in Z,\ \mathbf{w}^T\mathbf{x} = 0, \tag{2}$$

where $P \cup Z$ is a partition of the variable index set $\{1, \dots, n\}$ and $\mathbf{x}, \mathbf{w}$ are referred as primal and dual variables, respectively. We are interested in investigating the sparse recovery setting, that is, when $n \gg m$ and the right-hand side $\mathbf{b}$ belongs to the subspace spanned by the columns of A. In this case, the matrix A does not have full rank by columns and $\mathbf{b}$ may not have a unique expression as a linear combination of columns of A. In other words, problem (1) may not have a unique solution $\mathbf{x}$. Sparse recovery aims at finding the representation of $\mathbf{b}$ that involves the minimum number of columns of A. At an optimal solution of (1), we have $\mathbf{w} = 0$ and problem (2) lacks of strict complementarity, that is, it does not hold that $w_i < 0$ for $i \in Z$. The problems (1) and (2) are said to be degenerate and it is known that algorithms may exhibit problems like zigzagging and may not settle down at the optimal value.[23] The PBP algorithm[19] only deals with the full column rank case and it does not take into account degeneracy, therefore it is not suited for sparse recovery.

2.1 | The Lawson-Hanson algorithm

The first algorithm to solve (1) is due to Lawson and Hanson[11] and it is presented in Algorithm 1. We present the pseudocode for an actual implementation of the algorithm, based on Householder and Givens orthogonal transformations, as we detail in Section 4. It is a particular case of an active set algorithm[24] for the least squares problem with linear inequality constraints.

Let us now describe the main features of the Lawson-Hanson algorithm. Let P_s denote the passive set and Z_s denote the active set at s-th iteration, with cardinality respectively n_s and $n - n_s$. Let us define the following submatrices

$$A_P^{(s)} = \left(\mathbf{a}_{i_1} \ \dots \ \mathbf{a}_{i_{n_s}}\right) \in \mathbb{R}^{m\times n_s},\ \left\{i_1, \dots\ i_{n_s}\right\} = P_s, \qquad A_Z^{(s)} = \left(\mathbf{a}_{j_1} \ \dots \ \mathbf{a}_{j_{n-n_s}}\right) \in \mathbb{R}^{m\times(n-n_s)},\ \left\{j_1, \dots\ j_{n-n_s}\right\} = Z_s. \tag{3}$$

We define the s-th iterate as $\mathbf{x}^{(s)} = \left(x_1^{(s)}, \dots, x_n^{(s)}\right)^T \in \mathbb{R}^n$. With an analogous notation to the one introduced above for matrices, we have

$$\mathbf{x}_P^{(s)} = \left(x_{i_1}^{(s)}, \dots, x_{i_{n_s}}^{(s)}\right)^T \in \mathbb{R}^{n_s},\ \left\{i_1, \dots\ i_{n_s}\right\} = P_s, \qquad \mathbf{x}_Z^{(s)} = \left(x_{j_1}^{(s)}, \dots, x_{j_{n-n_s}}^{(s)}\right)^T \in \mathbb{R}^{n-n_s},\ \left\{j_1, \dots\ j_{n-n_s}\right\} = Z_s. \tag{4}$$

Since the iterates produced by Lawson-Hanson algorithm do not leave the feasible region, we always have $\mathbf{x}_Z^{(s)} = 0$. In Algorithm 1, the vector $\mathbf{y}_P^{(s)}$ is the least squares solution of the unconstrained subproblem

$$\min_{\mathbf{y}} \left\| A_P^{(s)}\mathbf{y} - \mathbf{b} \right\|_2^2, \tag{5}$$

and $\mathbf{y}_Z^{(s)} = 0$. Notice that these vectors change size at each iteration.

With the notation introduced, we have

$$\mathbf{r}^{(s)} = \mathbf{b} - A\mathbf{x}^{(s)} = \mathbf{b} - A_P^{(s)}\mathbf{x}_P^{(s)}, \qquad \mathbf{w}^{(s)} = A^T\left(\mathbf{b} - A\mathbf{x}^{(s)}\right) = A^T\left(\mathbf{b} - A_P^{(s)}\mathbf{x}_P^{(s)}\right) = A^T\mathbf{r}^{(s)}, \tag{6}$$

Algorithm 1. LH $(A, \mathbf{b})$

$P_0 = \emptyset, Z_0 = \{1, \dots, M\}, \mathbf{x}^{(0)} = 0, \mathbf{w}^{(0)} = A^T\mathbf{b}, s = 0$
while $Z_s \neq \emptyset$ and $\max(\mathbf{w}_Z^{(s)}) > 0$ **do** ▷ outer loop
  $j_s \in \operatorname{argmax}_i w_i^{(s)}$
  set $P_{s+1} = P_s \cup \{j_s\}, Z_{s+1} = Z_s \setminus \{j_s\}$
  if $s = 0$ **then**
    set $R^{(1)}$ as the Householder reflector related to $\mathbf{a}_{j_s}$ and compute $\mathbf{b}^{(1)}$
  else
    use rank-1 modification to update $R^{(s+1)}$ and update $\mathbf{b}^{(s+1)}$
  end if
  compute $\mathbf{y}_P^{(s+1)}$ as the solution of $R^{(s+1)}\mathbf{y} = \mathbf{b}^{(s+1)}(1 : n_{s+1})$, set $\mathbf{y}_Z^{(s+1)} = 0$
  while $\min\left(\mathbf{y}_P^{(s+1)}\right) \leq 0$ **do** ▷ inner loop
    $s = s + 1$
    $\alpha = \min\left\{ x_i^{(s-1)} \left(x_i^{(s-1)} - y_i^{(s)}\right)^{-1} : i \in P_s,\ y_i^{(s)} \leq 0 \right\}$
    $\mathbf{x}^{(s)} = \mathbf{x}^{(s-1)} + \alpha\left(\mathbf{y}^{(s)} - \mathbf{x}^{(s-1)}\right)$
    $Q = \left\{ i : x_i^{(s)} \leq 0 \right\}$
    $P_{s+1} = P_s \setminus Q$
    $Z_{s+1} = Z_s \cup Q$
    $R^{(s+1)} = R^{(s)}$
    for $j \in Q$ **do**
      delete column corresponding to index j from $R^{(s+1)}$
      reduce $R^{(s+1)}$ to triangular form by Givens transformation and update $\mathbf{b}^{(s+1)}$
    end for
    compute $\mathbf{y}_P^{(s+1)}$ as the solution of $R^{(s+1)}\mathbf{y} = \mathbf{b}^{(s+1)}(1 : n_{s+1})$, set $\mathbf{y}_Z^{(s+1)} = 0$
  end while
  $\mathbf{x}^{(s+1)} = \mathbf{y}^{(s+1)}$
  $\mathbf{w}_Z^{(s+1)} = \left(A_Z^{(s+1)}\right)^T \mathbf{b}, \mathbf{w}_P^{(s+1)} = 0$
  $s = s + 1$
end while
return $P^\star = P_s, Z^\star = Z_s, \mathbf{x}^\star = \mathbf{x}^{(s)}$

and, in particular

$$\mathbf{w}_Z^{(s)} = \left(A_Z^{(s)}\right)^T \left(\mathbf{b} - A_P^{(s)}\mathbf{x}_P^{(s)}\right) = \left(A_Z^{(s)}\right)^T \mathbf{r}^{(s)}, \qquad \mathbf{w}_P^{(s)} = \left(A_P^{(s)}\right)^T \left(\mathbf{b} - A_P^{(s)}\mathbf{x}_P^{(s)}\right) = 0. \tag{7}$$

where the last identity is a consequence of normal equations for (5), since at the end of the outer loop we have $\mathbf{x}_P^{(s)} = \mathbf{y}_P^{(s)}$. Clearly we have $\mathbf{w}^{(s)} \cdot \mathbf{x}^{(s)} = 0$ for every s. Algorithm 1 terminates in a finite number of steps[11] and $\mathbf{x}^{(s)}, \mathbf{w}^{(s)}, P_s, Z_s$ satisfy KKT conditions (2) on termination.

Since the columns of $A_P^{(s)}$ are linearly independent, the vector $\mathbf{y}_P^{(s)}$ can be obtained by standard QR decomposition. If we consider the whole matrix A, we can write the following partial QR decomposition with column pivoting

$$A\ \Pi^{(s)} = \left(A_P^{(s)} A_Z^{(s)}\right) = Q^{(s)} \begin{pmatrix} R^{(s)} & B^{(s)} \\ 0 & C^{(s)} \end{pmatrix}, \tag{8}$$

where $\Pi^{(s)}$ is the permutation matrix that permuted the elements in P_s in the leftmost positions. In an actual implementation,[11] the columns are not physically permuted, rather a permutation vector is used to keep track of the column ordering. Here, $B^{(s)}$ has size $n_s \times (n - n_s)$ and $C^{(s)}$ has size $(m - n_s) \times (n - n_s)$. Similarly, the updated right-hand side is given by $\mathbf{b}^{(s)} = \left(Q^{(s)}\right)^T \mathbf{b}$, and then $\mathbf{y}_P^{(s)}$ is the solution of the triangular system $R^{(s)}\mathbf{y}_P^{(s)} = \mathbf{b}^{(s)}(1 : n_s)$.

2.2 | The Lawson-Hanson algorithm with deviation maximization

In this section we propose a new active set method in which multiple columns are manipulated at a time and for which finite termination can be established in exact arithmetic. It has been already pointed out that in the sparse recovery setting, the right-hand side $\mathbf{b}$ does not have a unique representation as a linear combination of columns of A. We are interested in the positive representation that involves the minimum number of columns, or equivalently the nonnegative solution with the sparsest support $S^\star$, as it will be detailed in Section 3. Let $s^\star$ be the cardinality of the sparsest support $S^\star$ and recall that P_s is the passive set at iteration s. Since the LH method presented in Algorithm 1 moves at most one column at a time, we have $|P_s| < s^\star$ if the iteration index s is less than $s^\star$. Therefore, the current solution can reach the optimum after at least $s^\star$ iterations. The deviation maximization aims at moving a block of columns at each iteration in order to easily reach a cardinality of the passive set which is at least $s^\star$ while taking into account only linearly independent columns of A for handling degeneracy. Before coming to the description of the main algorithm, let us discuss the strategy for columns selection. Recall that for an $m \times k$ matrix $C = (\mathbf{c}_1 \ \dots \ \mathbf{c}_k)$ whose columns $\mathbf{c}_j$ are non-null, the *correlation matrix* Θ has entries

$$\theta_{ij} = \frac{\mathbf{c}_i^T \mathbf{c}_j}{\|\mathbf{c}_i\|_2 \|\mathbf{c}_j\|_2}, \quad 1 \le i,j \le k. \tag{9}$$

In particular, we have $\Theta = \left(CD^{-1}\right)^T CD^{-1} = D^{-1}C^T CD^{-1}$, where D is the diagonal matrix with entries $d_i = \|\mathbf{c}_i\|$, $1 \le i \le k$. It is immediate to see that Θ is symmetric positive semidefinite, it has only ones on the diagonal, and its entries range from -1 to 1. Notice that θ_{ij} is the cosine of $\alpha_{ij} = \alpha(\mathbf{c}_i, \mathbf{c}_j) \in [0, \pi)$, the angle (modulo π) between $\mathbf{c}_i$ and $\mathbf{c}_j$. In order to emphasize this geometric interpretation, from now on we refer to Θ as the *cosine matrix*.

Algorithm 2 presents the deviation maximization algorithm for column selection within a matrix C. The algorithm selects at most $k_{\max}$ columns with large pairwise deviations, that is, columns with large pairwise angles (modulo π), indexed within a set of candidate I; in formulae, we choose columns with indices $i, j \in I$ such that $|\theta_{ij}| < \delta$ for $i \neq j$. The set of candidate I is made of those column indices such that the corresponding entries of $\mathbf{u}_1, \mathbf{u}_2$ are "large enough" with respect to thresholds τ_1, τ_2, respectively. If $\mathbf{u}_2$ is the vector of column norms of C, then the deviation maximization can select numerically linearly independent columns.[20] Here we introduce $\mathbf{u}_1$ as the steepest descend direction at the current solution point in order to take into account first order information to fit the Lawson-Hanson algorithm.

Let us now relate the columns selected by the deviation maximization with the columns already processed.

Theorem 1. *Let $A_P^{(s)}$ be a full column rank matrix, and let J_s be the set of indices chosen by the deviation maximization applied to the matrix $C^{(s)}$ obtained by the partial QR decomposition (8). Finally, let $R^{(s+1)}$ be the R factor of the QR decomposition of $A_P^{(s+1)} = \left(A_P^{(s)}\ A_J^{(s)}\right)$, where $A_J^{(s)}$ is the submatrix of A that is obtained by taking columns with indices in the set J_s.*

Algorithm 2. Deviation maximization: $[J] = \mathrm{DM}\,(C, \mathbf{u}_1, \mathbf{u}_2, \tau_1, \tau_2, \delta, k_{\max})$

$J = \left\{ j : j \in \operatorname{argmax} \mathbf{u}_1 \right\}$
$I = \{ i : \mathbf{u}_k(i) \ge \tau_k \max \mathbf{u}_k,\ i \neq j, k = 1, 2 \}$
if $|I| > k_{\max}$ **then**
    $I = \{ i_k \in I : \ k = 1, \dots, k_{\max} \}$
end if
compute the cosine matrix Θ associated to $[C]_{:,I}$
for $i \in I$ **do**
    if $|\theta_{ij}| < \delta, \forall j \in J$ **then**
        $J = J \cup \{i\}$
    end if
end for
return J

If $\tau_1, \tau_2 \in (0, 1]$, $k_{\max} > 1$, *and*

$$\delta \leq \frac{\tau_2}{k_{\max} - 1}, \tag{10}$$

then

$$\sigma_{\min}\left(R^{(s+1)}\right) \geq \sigma_{n_{s+1}}(A)\frac{\sigma_{\min}\left(R^{(s)}\right)}{\sigma_1(A)}\frac{1}{\sqrt{2(n-n_{s+1})n_{s+1}}}\frac{\tau_2\sqrt{1-\tau_2}}{k_s^2 n_s} > 0, \tag{11}$$

where $k_s = |J_s|$.

Proof. Apply Theorem 2[20] to conclude. ∎

The result above gives a lower bound on the smallest singular value of $R^{(s+1)}$, and hence on $A_P^{(s+1)}$. However, it is known[20] that even if the matrix $A_P^{(s)}$ is well conditioned, there could be a potentially dramatic increase in the condition number of $A_P^{(s+1)}$ when using the deviation maximization strategy, as well as for the standard Lawson-Hanson algorithm. A famous example is the Kahan matrix, however it is a very unlikely occurrence in practice. Equation (10) implies in particular $\delta \leq \tau_2$. However, the values of δ and τ_2 can be set independently with minor modifications[20] without compromising the nonsingularity of $R^{(s+1)}$. Therefore, from now on we will not require (10) to hold.

Algorithm 3 presents the new version of the Lawson-Hanson with Deviation Maximization (LHDM).

In this work, we add the loop in steps 14–20, and it plays a central role in proving the following result about finite termination.

Theorem 2. *Consider the NNLS problem*

$$\min \|A\mathbf{x} - \mathbf{b}\|_2^2, \qquad \text{s.t. } \mathbf{x} \geq 0.$$

If $k_{\max} \geq 1$, *Algorithm 3 terminates in a finite number of steps. On termination,* $\mathbf{x}$ *will be a solution vector and* $\mathbf{w}$ *will be a dual solution vector*.

Proof. In order to ensure finite convergence of Algorithm 3, we have to prove the following:

1. the termination of the outer loop, by showing that $\mathbf{s}^{(s+1)} = \mathbf{y}^{(s+1)} - \mathbf{x}^{(s)}$ is a feasible descent direction;
2. the termination of the inner loop with a nonempty passive set.

Let us address the first task. The discussion following Theorem 1 ensures that $A_P^{(s+1)} = \left(A_P^{(s)}\ A_J^{(s)}\right)$ has numerically linearly independent columns. Let us claim the following statement: if $\mathbf{y}_J^{(s+1)} > 0$, then $\mathbf{s}^{(s+1)} = \mathbf{y}^{(s+1)} - \mathbf{x}^{(s)}$ is a feasible descent direction at $\mathbf{x}^{(s)}$. Then, if $\mathbf{y}_j^{(s+1)} \leq 0$ for some $j \in J_s$, the last column of $A_J^{(s)}$ is dropped and the corresponding index j_{k_s} is moved back to the active set Z_{s+1}. We then look for a new value of $\mathbf{y}^{(s+1)}$ such that solves

$$R^{(s+1)}(1 : n_{s+1} - 1, 1 : n_{s+1} - 1)\mathbf{y}_P^{(s+1)} = \mathbf{b}_1^{(s+1)}(1 : n_{s+1} - 1),$$

where the right-hand side $\mathbf{b}_1^{(s+1)}$ has been properly updated, for example, by applying the Givens rotation that zeros out the last diagonal element of $R^{(s+1)}$, corresponding to the deleted column. We continue dropping a column index at a time from J_s until $\mathbf{y}_j^{(s+1)} > 0$ for all indices j left in J_s. It should be noticed that this procedure terminates at most when only the first column $\mathbf{a}_{j_1}$ of $A_J^{(s)}$ is left, corresponding to the column index j_1 that would be selected by the standard Lawson-Hanson algorithm, which ensures the positivity of the corresponding solution entry.

For what concerns the second task, let us show that the inner loop terminates in at most $n_{s+1} - 1$ steps. Set $s = s + 1$ and notice that the first time the inner loop is entered, we have $\mathbf{y}_J^{(s)} > 0$, implying $\mathbf{x}_J^{(s)} = \alpha\mathbf{y}_J^{(s)} > 0$ and thus J_s and Q do not intersect. Therefore, indices in J_s are not removed before computing $\mathbf{y}^{(s+1)}$. Moreover, at least the index j_1 of J_s is left,

Algorithm 3. LHDM $(A, \mathbf{b}, \tau_1, \tau_2, \tau_c, k_{\max})$

1: $P_0 = \emptyset, Z_0 = \{1, \dots, M\}, \mathbf{x}^{(0)} = 0, \mathbf{w}^{(0)} = A^T\mathbf{b}, \mathbf{y}^{(0)} = 0, s = 0$
2: $R^{(0)} = 0, B^{(0)} = 0, C^{(0)} = A$
3: initialize $\mathbf{u}^{(0)}$ as the vector of column norms of A
4: **while** $Z_s \neq \emptyset$ and $\max\left(\mathbf{w}^{(s)}\right) > 0$ **do** ▷ outer loop
5: $[J_s] = \mathrm{DM}(C^{(s)}, \mathbf{w}^{(s)}, \mathbf{u}^{(s)}, \tau_1, \tau_2, \delta, k_{\max})$
6: $P_{s+1} = P_s \cup J_s$
7: $Z_{s+1} = Z_s \setminus J_s$
8: **if** $s = 0$ **then**
9: set $R^{(1)}$ as the R factor of the QR decomposition of $A_J^{(1)}$ and compute $\mathbf{b}^{(1)}$, $B^{(1)}$ and $C^{(1)}$
10: **else**
11: use rank-k update to get $R^{(s+1)}$, $B^{(s+1)}$, $C^{(s+1)}$, and update $\mathbf{b}^{(s+1)}$
12: **end if**
13: compute $\mathbf{y}_P^{(s+1)}$ as the solution of $R^{(s+1)}\mathbf{y} = \mathbf{b}^{(s+1)}(1 : n_{s+1})$, set $\mathbf{y}_Z^{(s+1)} = 0$
14: **while** $\min\left(\mathbf{y}_J^{(s+1)}\right) \leq 0$ **do** ▷ ensure a feasible descent direction
15: delete the last element from J_s
16: $P_{s+1} = P_s \cup J_s$
17: $Z_{s+1} = Z_s \setminus J_s$
18: delete the last column of $R^{(s+1)}$ and update $\mathbf{b}^{(s+1)}$, $B^{(s+1)}$, $C^{(s+1)}$,
19: compute $\mathbf{y}_P^{(s+1)}$ as the solution of $R^{(s+1)}\mathbf{y} = \mathbf{b}^{(s+1)}(1 : n_{s+1})$, set $\mathbf{y}_Z^{(s+1)} = 0$
20: **end while**
21: **while** $\min\left(\mathbf{y}_P^{(s+1)}\right) \leq 0$ **do** ▷ inner loop
22: $s = s + 1$
23: $\alpha = \min\left\{ x_i^{(s-1)} \left(x_i^{(s-1)} - y_i^{(s)}\right)^{-1} \ : \ i \in P_s,\ y_i^{(s)} \leq 0 \right\}$
24: $\mathbf{x}^{(s)} = \mathbf{x}^{(s-1)} + \alpha\left(\mathbf{y}^{(s)} - \mathbf{x}^{(s-1)}\right)$
25: $Q = \left\{ i : x_i^{(s)} \leq 0 \right\}$
26: $P_{s+1} = P_s \setminus Q$
27: $Z_{s+1} = Z_s \cup Q$
28: $R^{(s+1)} = R^{(s)}$
29: **for** $j \in Q$ **do**
30: delete column corresponding to index j from $R^{(s+1)}$
31: reduce $R^{(s+1)}$ to triangular form by Givens transformation and update $\mathbf{b}^{(s+1)}$
32: **end for**
33: compute $\mathbf{y}_P^{(s+1)}$ as the solution of $R^{(s+1)}\mathbf{y} = \mathbf{b}^{(s+1)}(1 : n_{s+1})$, set $\mathbf{y}_Z^{(s+1)} = 0$
34: **end while**
35: $\mathbf{x}^{(s+1)} = \mathbf{y}^{(s+1)}$
36: $\mathbf{w}^{(s+1)} = A^T\left(\mathbf{b} - A\mathbf{x}^{(s+1)}\right)$
37: update $\mathbf{u}^{(s)}$ as the vector of column norms of $C^{(s+1)}$
38: $s = s + 1$
39: **end while**
40: **return** $P^\star = P_s, Z^\star = Z_s, \mathbf{x}^\star = \mathbf{x}^{(s)}$

because of Lemma 23.17.[11] Thus, the inner loop must terminate at most in $n_{s+1} - 1$ step with a nonempty passive set $P_{s+1} \supseteq \{j_1\}$.

Let us now prove the claim, that is if $\mathbf{y}_J^{(s+1)} > 0$ then $\mathbf{s}^{(s+1)} = \mathbf{y}^{(s+1)} - \mathbf{x}^{(s)}$ is a feasible descent direction at $\mathbf{x}^{(s)}$. Consider the least squares solution $\mathbf{y}^{(s+1)}$ computed at step 13 of Algorithm 3, that is

$$\mathbf{y}_P = \underset{\mathbf{y}}{\operatorname{argmin}} \left\| A_P^{(s+1)}\mathbf{y} - \mathbf{b} \right\|_2^2 \quad \mathbf{y}_Z = 0. \tag{12}$$

where $Z = Z_{s+1}$ and $P = P_{s+1}$. The additional inner loop at steps 14–20 ensures that $y_j^{(s+1)} > 0$ for $j \in J_s$ on termination. This, together with $\mathbf{x}^{(s)} \geq 0$, implies that $\mathbf{s}^{(s+1)} := \mathbf{y}^{(s+1)} - \mathbf{x}^{(s)}$ is a feasible direction. Let us now prove that $\mathbf{s}^{(s+1)}$ is also a descent direction. Said $\mathbf{r}^{(s)} = \mathbf{b} - A\mathbf{x}^{(s)}$, let us define $\mathbf{z}$ as follows

$$\mathbf{z} = \underset{\mathbf{z}_Z=0}{\operatorname{argmin}} \|A\mathbf{z} - \mathbf{r}^{(s)}\|_2^2 = \underset{\mathbf{z}_Z=0}{\operatorname{argmin}} \|A(\mathbf{x}^{(s)} + \mathbf{z}) - \mathbf{b}\|_2^2, \tag{13}$$

where $Z = Z_{s+1}$ and $P = P_{s+1}$. Notice that $\mathbf{z}$ can be obtained as

$$\mathbf{z}_P = \underset{\mathbf{z}}{\operatorname{argmin}} \|A_P^{(s+1)}\mathbf{z} - \mathbf{r}^{(s)}\|_2^2, \quad \mathbf{z}_Z = 0. \tag{14}$$

Then we have

$$\|A\mathbf{z} - \mathbf{r}^{(s)}\|_2 = \|A\mathbf{z} - (\mathbf{b} - A\mathbf{x}^{(s)})\|_2 = \|A(\mathbf{z} + \mathbf{x}^{(s)}) - \mathbf{b}\|_2 \geq \|A\mathbf{y}^{(s+1)} - \mathbf{b}\|_2, \tag{15}$$

since $\mathbf{z} + \mathbf{x}^{(s)}$ vanishes on Z_{s+1} and the minimum is reached at $\mathbf{y}^{(s+1)}$. On the other hand

$$\|A\mathbf{y}^{(s+1)} - \mathbf{b}\|_2 = \|A\mathbf{y}^{(s+1)} - A\mathbf{x}^{(s)} - \mathbf{b} + A\mathbf{x}^{(s)}\|_2 = \|A(\mathbf{y}^{(s+1)} - \mathbf{x}^{(s)}) - \mathbf{r}^{(s)}\|_2 \geq \|A\mathbf{z} - \mathbf{r}^{(s)}\|_2, \tag{16}$$

since $\mathbf{y}^{(s+1)} - \mathbf{x}^{(s)}$ vanishes on Z_{s+1} and the minimum is reached at $\mathbf{z}$. Therefore

$$\|A\mathbf{z} - \mathbf{r}^{(s)}\|_2 = \|A\mathbf{y}^{(s+1)} - \mathbf{b}\|_2,$$

and the inequality chain (16) collapses in an equality chain. As a consequence of the uniqueness of minimizers of (12) and (13), we have

$$\mathbf{z} = \mathbf{s}^{(s+1)} = \mathbf{y}^{(s+1)} - \mathbf{x}^{(s)}. \tag{17}$$

Consider the function

$$\varphi(\varepsilon) = \frac{1}{2}\|A(\mathbf{x}^{(s)} + \varepsilon\mathbf{s}^{(s+1)}) - \mathbf{b}\|_2^2.$$

We have

$$\varphi'(0) = \left(\mathbf{s}^{(s+1)}\right)^T A^T(A\mathbf{x}^{(s)} - \mathbf{b}) = -(\mathbf{s}^{(s+1)})^T\mathbf{w}^{(s)} = -(\mathbf{y}^{(s+1)})^T\mathbf{w}^{(s)} = \sum_{i\in J} y_i^{(s+1)}(-w_i^{(s)}) < 0, \tag{18}$$

since $w_i^{(s)} > 0$ and $y_i^{(s+1)} > 0$ for all $i \in J_s$. Therefore there exists $\varepsilon > 0$ such that

$$\|A(\mathbf{x}^{(s)} + \varepsilon\mathbf{s}^{(s+1)}) - \mathbf{b}\|_2^2 < \|A\mathbf{x}^{(s)} - \mathbf{b}\|_2^2.$$

By (13) and (17), we have

$$\|A(\mathbf{x}^{(s)} + \mathbf{s}^{(s+1)}) - \mathbf{b}\|_2^2 \leq \|A(\mathbf{x}^{(s)} + \varepsilon\mathbf{s}^{(s+1)}) - \mathbf{b}\|_2^2 < \|A\mathbf{x}^{(s)} - \mathbf{b}\|_2^2. \tag{19}$$

Thus, if $\mathbf{y}^{(s+1)}$ is feasible we take $\varepsilon = 1$, that is, $\mathbf{x}^{(s+1)} = \mathbf{y}^{(s+1)}$, otherwise the algorithm enters the inner loop and we choose a smaller step length α, more precisely the largest possible to keep the new iterate within the feasible region. It's easy to check that

$$\|A(\mathbf{x}^{(s)} + \alpha\mathbf{s}^{(s+1)}) - \mathbf{b}\|_2^2 = \min_{\varepsilon} \left\{ \|A(\mathbf{x}^{(s)} + \varepsilon\mathbf{s}^{(s+1)}) - \mathbf{b}\|_2^2 : \ \mathbf{x}^{(s)} + \varepsilon\mathbf{s}^{(s+1)} \geq 0 \right\} \leq \|A\mathbf{x}^{(s)} - \mathbf{b}\|_2^2. \tag{20}$$

We deduce that the objective function is always non increasing. Equation (19) also implies that when step 35 is reached, we have a strictly smaller value of the objective function. Hence at every iteration of the outer loop, the current objective value is strictly smaller than the previous one. Since the solution solves a least squares subproblem like those of steps 13

and 33, the corresponding passive set must be different from the preceding one. As the number of possible combinations of active and passive sets is finite, namely it is equal to 2^n, the algorithm must terminate in at most 2^n outer loop repetitions. ■

In Section 5, we give experimental evidence of the significant performance gain that this strategy can lead.

3 | SPARSE RECOVERY

In this section we present the problem of sparse recovery and its connections with nonnegative least squares problem. There is a wide literature[25-27] about sparse recovery in the field of signal processing, in which a matrix is usually referred as a dictionary and its columns are called atoms. Here, we abandon this terminology in favour of the classical linear algebra one.

Given a vector $\mathbf{x} \in \mathbb{R}^n$, we define its support $S = S(\mathbf{x}) = \{i : x_i \neq 0\}$. We aim at identifying the solution $\mathbf{x}^\star$ of the linear system $A\mathbf{x} = \mathbf{b}$ with the sparsest support $S^\star$. Formally, we consider the optimization problem

$$\min \|\mathbf{x}\|_0, \qquad \text{s.t. } A\mathbf{x} = \mathbf{b}. \tag{21}$$

Recall the "ℓ_0-norm" of a vector $\mathbf{x}$ is defined as $\|\mathbf{x}\|_0 = |S(\mathbf{x})|$ and it is not an actual norm because it does not hold that for any scalar s we have $\|s\mathbf{x}\|_0 = |s|\ \|\mathbf{x}\|_0$. Actually, problem (21) of combinatorial search is NP-hard, and in general we seek an approximate solution. There are two well known approaches: "greedy algorithms", that focus on the determination of the support and the least squares solution supported at it; "relaxation methods", that focus on smoothing the ℓ_0 norm and then adopt smooth optimization. A famous greedy algorithm is the Orthogonal-Matching-Pursuit (OMP), while a popular convex relaxation technique is the so-called *Basis Pursuit* (BP)[28] problem

$$\min \|\mathbf{x}\|_1, \qquad \text{s.t. } A\mathbf{x} = \mathbf{b}, \tag{22}$$

which is a convex optimization problem, since $\|\mathbf{x}\|_1$ is a convex function of $\mathbf{x}$ and the solutions of $A\mathbf{x} = \mathbf{b}$ form a convex subspace. We will concentrate on this second approach. Indeed, the two problems (21) and (22) yield the same solution provided it is "sparse enough", where this bound depends on the properties of the matrix A. Results of this type fall under the so-called $\ell_0 - \ell_1$ equivalence.[7-9] For example, it can be shown that $\ell_0 - \ell_1$ equivalence holds provided that the support of the solution satisfies $\|\mathbf{x}\|_0 < (1 + M^{-1})/2$, where $M = \sup_{i \neq j} |\theta_{ij}|$ and $\Theta = (\theta_{ij})$ is the cosine matrix of A.[8]

3.1 | Exact recovery

There are different conditions under which we have $\ell_0 - \ell_1$ equivalence, that is, an exact recovery. In the literature, these conditions are usually found by showing the uniqueness of solution to problems (21) and (22) and their coincidence. As a consequence of these results, efficient algorithms to solve (22) can be used to find the sparsest solution to an underdetermined system of equations.

The unique solution of problem (22) can be proved under the so-called Restricted Isometry Property (RIP),[29] and the same solution also solves problem (21) under suitable conditions. However, the problem of establishing whether a given matrix A fulfills the RIP is NP-hard in general.[30] An alternative proof can be obtained under the so-called *Exact Recovery Condition* (ERC).[31]

Theorem 3 (ERC). *Consider a linear system $A\mathbf{x} = \mathbf{b}$. If there exists a solution $\mathbf{x}^\star$ with support $S^\star = S(\mathbf{x})^\star$ such that*

$$\max_{i \notin S^\star} \left\| A^\dagger_{S^\star} \mathbf{a}_i \right\|_1 < 1, \tag{23}$$

then $\mathbf{x}^\star$ is the unique solution to the minimum ℓ_0 problem (21), which can be recovered by solving the minimum ℓ_1 problem (22).

This provides an easy sufficient optimality check for a given support S, but finding a support S satisfying (23) is again a combinatorial problem.

3.2 | Sparse recovery by NNLS

The nonnegativity constraint is known to naturally produce sparse solutions.[3-6] An important outcome of this body of work is that nonnegativity alone may attain a satisfactory sparse recovery. Algorithms to tackle NNLS problem (1) exhibit important similarities with methods for solving ℓ_1 minimization problem (22). For example,[4] shows that the before mentioned Orthogonal-Matching-Pursuit (OMP) resembles the Lawson-Hanson algorithm except for the internal loop, which has indeed an evident benefit on nonnegative recovery.

Notice also that arbitrary signed sparse recovery is easily achievable. Given $\overline{\mathbf{x}} \in \mathbb{R}^n$, decompose it as $\overline{\mathbf{x}} = \overline{\mathbf{x}}^+ - \overline{\mathbf{x}}^-$, where $\overline{\mathbf{x}}^+ \geq 0$ and $\overline{\mathbf{x}}^- \geq 0$. Then the solutions of the linear system $A\overline{\mathbf{x}} = \mathbf{b}$ can be attained as the solutions of the nonnegative least squares problem

$$\min_{\mathbf{x}} \|(A \ -A)\mathbf{x} - \mathbf{b}\|_2^2, \quad \mathbf{x} \geq 0, \tag{24}$$

where $\mathbf{x} = \begin{pmatrix} \overline{\mathbf{x}}^+ \\ \overline{\mathbf{x}}^- \end{pmatrix} \in \mathbb{R}^{2n}$. This technique has been presented for example, in Reference 4 and it is sometimes referred as "positivity trick".

4 | UPGRADING THE QR DECOMPOSITION

In an actual implementation of Algorithms 1 and 3, the triangular matrix $R^{(s)}$ is never computed from scratch, instead it is updated or downdated by means of Householder reflections or Givens rotations, respectively. Let us now detail the addition and the removal of k columns first in Section 4.1 for general NNLS problems and then in Section 4.2 the case in which an arbitrary signed solution is sought by means of the positivity trick (24).

4.1 | The general case

Basic algorithms for QR updating and downdating are described in Reference 32. Let us first consider the problem of updating the QR decomposition in the outer loop. Let A be an $m \times n$ matrix and B an $m \times k$ matrix, both full rank by columns, such that $k + n \leq m$, and suppose moreover that the columns in B are linearly independent from those in A. We want to exploit the knowledge of the decomposition $A = QR$ in order to compute $\tilde{Q}, \tilde{R}$ such that

$$\tilde{A} := (A \ B) = \tilde{Q}\tilde{R}.$$

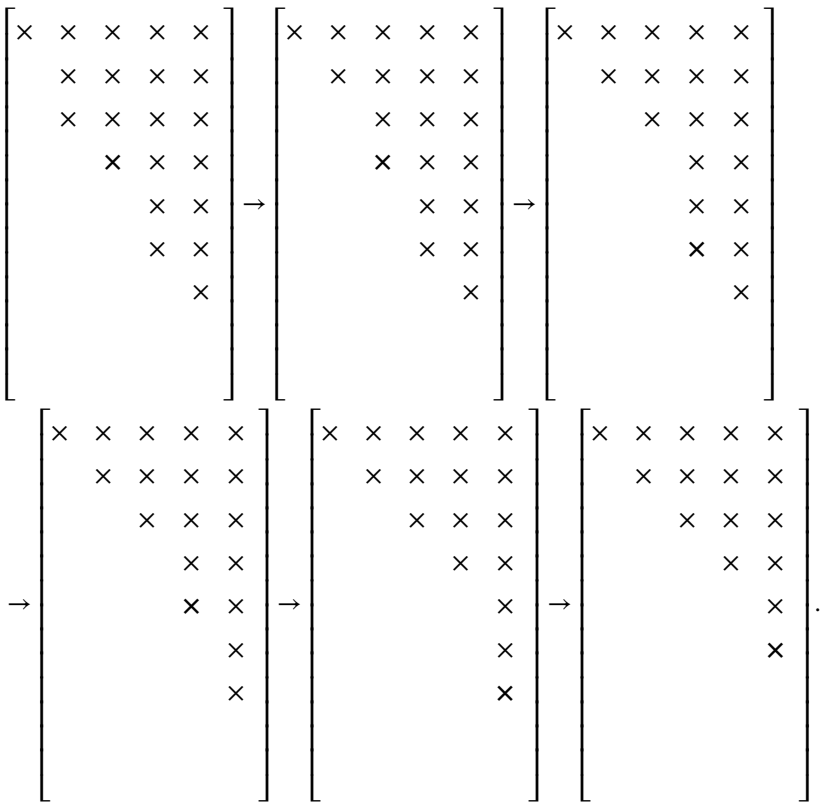
This can be done by applying the standard Householder QR algorithm to the matrix $Q^T B(n+1 : m, :)$, which involves about $2k^2(m - (n+1) - k/3)$ flops. The cost for computing $Q^T B$ is about $kn(2m - n)$ flops. The whole procedure involves about $2k^2(m - (n+1) - k/3) + kn(2m - n) = \mathcal{O}(k^2 m + knm)$ flops.

Let us now consider the downdating problem. Consider a set of k column indices $\{j_1, \ldots, j_k\}$, with $1 \leq j_1 < \cdots < j_k < n$, and let $\tilde{A}$ be the $m \times (n-k)$ matrix whose columns are the columns of A with index not in $\{j_1, \ldots, j_k\}$. Again, we want to exploit the knowledge of the decomposition $A = QR$ in order to compute $\tilde{Q}, \tilde{R}$ such that

$$\tilde{A} = \tilde{Q}\tilde{R}.$$

The matrix $\tilde{A}$ has a structure close to the Hessenberg form. Let us describe the elimination process in the particular case with $m = 8$, $n = 7$, $k = 2$, $j_1 = 2$ and $j_2 = 5$. The following scheme illustrates how Givens rotations can be used to

annihilate the highlighted entries one at a time, in a column-major fashion:



Thus, columns with indices $j_l < j < j_{l+1}$ have exactly l subdiagonal entries to annihilate, for $l = 1, \dots, k-1$. Note that one could also zero out one subdiagonal at a time, with exactly the same number of floating point operations. The following algorithm describes the resulting procedure (in column-major fashion; Algorithm 4).

The explicit computation of $\tilde{Q}$ is not necessary if we express it as a product of Givens rotation matrices. The operation count can be computed by observing that this procedure boils down to the annihilation of k subdiagonals of different length. After removing column j_l, we have to zero out the last element of each column of the remaining $m \times (n - j_l - (k - l))$ matrix: in fact the columns from j_l-th (not included) until the last one are $n - j_l$, but there are also $k - l$ columns we don't have to downdate, those with indices $j_{l+1}, \dots, j_k$, that have already been removed. Define $n_l = (n - j_l - (k - l))$. Then, this operation takes $\sum_{i=0}^{n_l} 6(n_l - i) = \mathcal{O}(3n_l^2) = \mathcal{O}(3(n - j_l - (k - l))^2)$. We compute the worst case of the total operation count,

Algorithm 4. Downdating

$A = QR, j_1 < \dots < j_k$
$\tilde{Q} = Q, \tilde{R} = R, j_{k+1} = n + 1$
for $l = 1, \dots, k$ **do**
  for $j = j_l + 1, \dots, j_{l+1} - 1$ **do**
    for $p = 1, \dots, l$ **do**
      $i = j - p + 1$
      $\tilde{G}$ =givens($\tilde{A}(i-1 : i, j)$) ▷ compute Givens rotation
      $\tilde{R}(i-1 : i, j+1 : m) = \tilde{G}\tilde{R}(i-1 : i, j+1 : m)$
      $\tilde{Q}(:, i-1 : i) = \tilde{Q}(:, i-1 : i)\tilde{G}^T$
    end for
  end for
end for
$\tilde{R}(:, j_1, \dots, j_k) = [\,]$ ▷ delete columns $j_1, \dots, j_k$

in which the first k columns are removed: we have $j_l = l$ with $1 \le l \le k$, thus we have to zero out the last k subdiagonals of the remaining $m \times (n-k)$ matrix. This involves $\mathcal{O}(\sum_{l=1}^{k} 3(n-k)^2) = \mathcal{O}(3n^2k + 3k^3 - 6nk^2)$ flops.

The updating and downdating procedures allow to modify the current QR decomposition with a remarkable computational saving. In a practical implementation, one can apply each time the orthogonal transformations to the whole matrix, or only to the columns whose indices are in the current passive set. In the first case, it is not necessary to keep trace of the orthogonal transformations themselves. In the second case, we need to keep in memory the sequence of orthogonal transformations computed so far; moreover, an additional vector is needed to keep trace of the number of orthogonal transformations that have been applied to each column. In the context of sparse recovery, the second strategy proves better than the first one.

4.2 | The positivity trick case

Let us now consider the particular case in which an arbitrary signed solution is sought by solving the NNLS problem with positivity trick (24). In such a case, the objective function's matrix is $(A \quad -A) \in \mathbb{R}^{m \times 2n}$. The updating procedure by Householder triangularization is unchanged, while the downdating procedure by Givens rotations in the inner loop can be avoided with a simple "sign flip", as we show here.

Suppose that $\mathbf{y}^{(s)}$ computed after the loop at steps 14–20 of Algorithm 3 is not a feasible point, thus $y_k^{(s)} < 0$, for some $k \in P_{s-1}$. Since $\mathbf{s}^{(s)} = \mathbf{y}^{(s)} - \mathbf{x}^{(s-1)}$ is a feasible descent direction at $\mathbf{x}^{(s-1)}$, we have $k \notin J_{s-1}$. If $k \le n$, the index k corresponds to a column of A, otherwise if $n < k \le 2n$, it corresponds to a column of $-A$.

With a little abuse of notation, we denote by k both the column index and its position within the passive set P_s, so that $P_s = \{j_1, \ldots, j_{k-1}, k, j_{k+1}, \ldots, j_{n_s}\}$. Let us denote by $\overline{k}$ the twin index of k, that is, $\overline{k} = k + n$ if $k \le n$, and $\overline{k} = k - n$ if $n < k \le 2n$. Let us show that if we let $P^{(s+1)} = \left\{j_1, \ldots, j_{k-1}, \overline{k}, j_{k+1}, \ldots, j_{n_s}\right\}$, then $\mathbf{y}^{(s+1)}$ is a feasible point and $\mathbf{s}^{(s+1)} = \mathbf{y}^{(s+1)} - \mathbf{x}^{(s-1)}$ is a feasible descent direction at $\mathbf{x}^{(s-1)}$.

We first show that $\mathbf{y}^{(s+1)}$ is a feasible point. If we write by columns $A_P^{(s)} = (\mathbf{a}_{j_1} \ \ldots \ \mathbf{a}_{j_{k-1}} \ \mathbf{a}_k \mathbf{a}_{j_{k+1}} \ \ldots \ \mathbf{a}_{j_{n_s}})$, then we have $A_P^{(s+1)} = (\mathbf{a}_{j_1} \ \ldots \ \mathbf{a}_{j_{k-1}} - \mathbf{a}_k \ \mathbf{a}_{j_{k+1}} \ \ldots \ \mathbf{a}_{j_{n_s}})$. Consider the QR decomposition

$$A_P^{(s)} = Q^{(s)} \, R^{(s)} = Q^{(s)} \ \left(\mathbf{r}_1 \ \ldots \ \mathbf{r}_{k-1} \ \mathbf{r}_k \ \mathbf{r}_{k+1} \ \ldots \ \mathbf{r}_{n_s}\right),$$

then trivially $Q^{(s)}$ also reduces $A_P^{(s+1)}$ to triangular form, and we have

$$A_P^{(s+1)} = Q^{(s)} \, R^{(s+1)} = Q^{(s)} \ \left(\mathbf{r}_1 \ \ldots \ \mathbf{r}_{k-1} \ -\mathbf{r}_k \ \mathbf{r}_{k+1} \ \ldots \ \mathbf{r}_{n_s}\right).$$

Therefore, we set $Q^{(s+1)} = Q^{(s)}$. Now, $\mathbf{y}^{(s)}$ is the solution of the least squares problem

$$\mathbf{y}_P^{(s)} = \underset{\mathbf{y}}{\operatorname{argmin}} \left\| A_P^{(s)} \mathbf{y} - \mathbf{b} \right\|_2^2, \quad \mathbf{y}_Z^{(s)} = 0, \tag{25}$$

which can be obtained as the solution of

$$R^{(s)} \mathbf{y}_P^{(s)} = \mathbf{b}_P^{(s)} = \left(Q^{(s)}\right)^T \mathbf{b} \ (1 : n_s), \tag{26}$$

Moreover, we have $\mathbf{b}_P^{(s+1)} = \left(Q^{(s+1)}\right)^T \mathbf{b} \ (1 : n_{s+1}) = \left(Q^{(s)}\right)^T \mathbf{b} \ (1 : n_s) = \mathbf{b}_P^{(s)}$. Let us split $\mathbf{y}_P^{(s)} = (\mathbf{y}_1^{(s)}, y_k^{(s)}, \mathbf{y}_2^{(s)})^T$ and $\mathbf{y}_P^{(s+1)} = (\mathbf{y}_1^{(s+1)}, y_{\overline{k}}^{(s+1)}, \mathbf{y}_2^{(s+1)})^T$, where $\mathbf{y}_1^{(s)}, \mathbf{y}_1^{(s+1)} \in \mathbb{R}^{k-1}$, $\mathbf{y}_2^{(s)}, \mathbf{y}_2^{(s+1)} \in \mathbb{R}^{n_s - k}$, and $y_k^{(s)} < 0$. Similarly, we write $\mathbf{b}_P^{(s)} = (\mathbf{b}_1^{(s)}, b_k^{(s)}, \mathbf{b}_2^{(s)})^T$, where $\mathbf{b}_1^{(s)} \in \mathbb{R}^{k-1}$, $\mathbf{b}_2^{(s)} \in \mathbb{R}^{m-k}$. The triangular matrices can be split as

$$R^{(s)} = \begin{pmatrix} R_{11} & \mathbf{r}_k(1 : k-1) & R_{12}(1 : k-1, :) \\ 0^T & r_{kk} & R_{12}(k, :) \\ 0 & 0 & R_{22} \end{pmatrix}, \qquad R^{(s+1)} = \begin{pmatrix} R_{11} & -\mathbf{r}_k(1 : k-1) & R_{12}(1 : k-1, :) \\ 0^T & -r_{kk} & R_{12}(k, :) \\ 0 & 0 & R_{22} \end{pmatrix}, \tag{27}$$

where R_{11} is upper triangular of order $k-1$, R_{12} has size $k \times (n_s - k)$, and R_{22} is upper triangular of size $(m-k) \times (n_s - k)$. Since $\mathbf{y}_2^{(s)}$ minimizes $\|R_{22}\mathbf{y} - \mathbf{b}_2^{(s)}\| = \|R_{22}\mathbf{y} - \mathbf{b}_2^{(s+1)}\|$, we have $\mathbf{y}_2^{(s)} = \mathbf{y}_2^{(s+1)}$. By backsubstitution, we obtain

$$y_k^{(s)} = \frac{1}{r_{kk}} \left(b_k^{(s)} - R_{12}(k, 1 : n_s - k)^T \mathbf{y}_2^{(s)} \right) < 0, \tag{28}$$

and similarly

$$y_{\overline{k}}^{(s+1)} = -\frac{1}{r_{kk}} \left(b_k^{(s)} - R_{12}(k, 1 : n_s - k)^T \mathbf{y}_2^{(s+1)} \right) = -y_k^{(s)} > 0, \tag{29}$$

where we used the fact that $\mathbf{y}_2^{(s+1)} = \mathbf{y}_2^{(s)}$. Last, we have

$$\mathbf{y}_1^{(s)} = R_{11}^{-1} \left(\mathbf{b}_1^{(s)} - y_k^{(s)} \mathbf{r}_k(1 : k-1) - R_{12}(1 : k-1, :) \mathbf{y}_2^{(s)} \right), \tag{30}$$

and

$$\begin{aligned} \mathbf{y}_1^{(s+1)} &= R_{11}^{-1} \left(\mathbf{b}_1^{(s)} + y_{\overline{k}}^{(s+1)} \mathbf{r}_k(1 : k-1) - R_{12}(1 : k-1, :) \mathbf{y}_2^{(s+1)} \right) \\ &= R_{11}^{-1} \left(\mathbf{b}_1^{(s)} - y_k^{(s)} \mathbf{r}_k(1 : k-1) - R_{12}(1 : k-1, :) \mathbf{y}_2^{(s)} \right) = \mathbf{y}_1^{(s)}. \end{aligned} \tag{31}$$

Therefore, $\mathbf{y}_P^{(s+1)} = (\mathbf{y}_1^{(s)}, -y_k^{(s)}, \mathbf{y}_2^{(s)})^T > 0$, meaning that $\mathbf{y}^{(s+1)}$ is a feasible point and $\mathbf{s}^{(s+1)} = (\mathbf{y}^{(s+1)} - \mathbf{x}^{(s-1)})$ is a feasible direction.

Let us show that $\mathbf{s}^{(s+1)}$ is also a descent direction. Suppose now without loss of generality that $k \leq n$, that is, k corresponds to a column of A and $\overline{k} > n$. Recall that, at the beginning of the inner loop, the vector $\mathbf{s}^{(s)} = (\mathbf{y}^{(s)} - \mathbf{x}^{(s-1)})$ is a descent direction, that is,

$$\|\mathbf{r}^{(s)}\|_2 = \|(A \ -A)\mathbf{y}^{(s)} - \mathbf{b}\|_2 < \|(A \ -A)\mathbf{x}^{(s-1)} - \mathbf{b}\|_2.$$

We have

$$\mathbf{r}^{(s+1)} = \mathbf{b} - (A \ -A)\mathbf{y}^{(s+1)} = \mathbf{b} - \sum_{\substack{i \in P_{s+1} \\ i \leq n}} y_i^{(s+1)} \mathbf{a}_i + \sum_{\substack{i \in P_{s+1} \\ i > n, \ i \neq \overline{k}}} y_i^{(s+1)} \mathbf{a}_i + y_{\overline{k}}^{(s+1)} \mathbf{a}_{\overline{k}} = \mathbf{b} - \sum_{\substack{i \in P_s \\ i \leq n, \ i \neq k}} y_i^{(s)} \mathbf{a}_i + \sum_{\substack{i \in P_s \\ i > n}} y_i^{(s)} \mathbf{a}_i + y_k^{(s)} \mathbf{a}_k = \mathbf{r}^{(s)}, \tag{32}$$

therefore $\mathbf{s}^{(s+1)}$ is a feasible descent direction with $\mathbf{y}^{(s+1)}$ feasible. We can set $\mathbf{x}^{(s+1)} = \mathbf{y}^{(s+1)}$. Notice that once computed $\mathbf{y}^{(s)}$, no further computation is needed to obtain $\mathbf{y}^{(s+1)}$, except for one sign flip.

5 | NUMERICAL EXPERIMENTS

In this section, we present the test results on a quite large set of instances. The algorithms LHDM and LH, both written in Matlab and available at https://github.com/NLALDlab/LHDM, are compared with other convex relaxation techniques:

- l1_homotopy: the homotopy method, https://intra.ece.ucr.edu/~sasif/homotopy;
- SolveBP: primal-dual log-barrier method to an equivalent linear program of BP (22), https://sparselab.stanford.edu;
- spgl1: a specialization of the spectral projected gradient method, https://www.cs.ubc.ca/labs/scl/spgl1;
- isal1: infeasible-point subgradient algorithm (ISA) for basis pursuit (22), https://www.opt.mathematik.tu-darmstadt.de/spear
- solveOMP: the orthogonal matching pursuit algorithm, https://sparselab.stanford.edu.

Note that we do not compare LHDM with other block NNLS solvers since we are not aware of such a method that can handle degeneracy. We tested e.g. the well-known PBP algorithm[19] on the same dataset and we observed it fails in

retrieving a solution for all instances. Moreover, we carried out experiments with randomly generated instances with an increasing number of columns, from square to underdetermined matrices, and they revealed PBP is initially inaccurate and then it fails. This is because this method may select linearly dependent columns during iterations.

Moreover, we do not compare LHDM with l1_magic, a primal-dual interior-point method for a LP reformulation of basis pursuit (22), from https://www.l1-magic.org, since in all our experiments it is dominated by l1_homotopy.

As previously done in the literature,[22] we use the default settings for all optional parameters of the solvers listed above. Similarly, in the deviation maximization algorithm we use the parameters' default values,[20] that is $\tau_1 = 0.6$, $\delta = 0.9$, $\tau_2 = 0.15$ and $k_{\max} = 32$, hence giving evidence of the wide applicability of the method proposed without the need of choosing problem dependent parameters. However, a fine tuning of the parameters can give slight improvements in specific problem families. This fine tuning is out of the scope of this paper since here we want to exploit the general applicability of the method.

The algorithms are compared over a dataset consisting of the 548 instances widely discussed in the literature.[22] We refer to this dataset as (D1). In our tests, we have used 444 instances over 548, since we restrict our investigation to dense matrices from this dataset. In fact, dealing with sparse matrices would require a dedicated implementation of LHDM which can be addressed in the future. In order to simplify the comparison of the results here presented with the ones available in the literature,[22] we keep the same indexing from 1 to 548 to show the results. Results concerning sparse instances are left blank and corresponding speedups are set as one.

The comparisons are made in terms of execution times and solution quality expressed as distance to optimum $\|\mathbf{x} - \mathbf{x}_{calc}\|_2$, that is, the ℓ_2 norm of the error, where the vector $\mathbf{x}_{calc}$ is the computed solution and $\mathbf{x}$ is the sparsest solution, unique when ERC condition (23) holds. Note that the distance to optimum is meaningful for the instances where the uniqueness of the sparsest solution is guaranteed, for example, under the ERC property (23). Our purpose is not to provide an exhaustive experimental comparison between these methods but, instead, to assess the substantial performance gain obtained by adding the deviation maximization to the Lawson-Hanson algorithm, that is, by the LHDM algorithm. Also, we aim to assess its competitiveness with the other publicly available methods for sparse recovery of dense underdetermined linear systems, on a wide class of instances, for example: with solutions arbitrarily signed, in particular when the matrices are not well conditioned, and solutions constrained to be nonnegative. In order to do this, we generate modified datasets (D2), (D3) and (D4). In the datasets (D1) and (D4), many instances satisfy the ERC (23), thus representing a favorable situation for compressed sensing in terms of sparse solutions retrieval via ℓ_1 minimization. Instances not satisfying the ERC are also included. Instances satisfying the ERC have indices ranging in $\{1, \dots, 200\}$ and $\{275, \dots, 474\}$. This helps the understanding of the Figures. It must be considered that, we have enforced the ERC condition on all instances of the modified datasets (D2) and (D3). We used the positivity trick (24) in order to achieve an arbitrary signed solution by means of LHDM, which succeeds in exact recovery for all tests verifying the ERC, meaning that the sparsest solution is found. The results are grouped into subsections, each devoted to a single experimental aspect.

5.1 | Efficiency of the "sign flip"

We have seen in Section 4.2 that the inner loop of LH can be substituted by the much simpler "sign flip" when the positivity trick is used (24). In Figure 1 we see the difference between running the inner loop or the sign-flip in the LH algorithm: there is an interesting speedup, around 1.5×, for the most expensive tests. Thefore, we use by default the sign flip in LH and in LHDM, for all the tests that require the positivity trick.

5.2 | Speedup of LHDM versus LH

Figure 2 shows the speedup of LHDM with respect to LH, for solution vectors at different sparsity degrees. It shows the two extreme situations: the dataset (D1), where solution vectors have a very small support, and a modified dataset (D2), where solution vectors have a support whose cardinality is equal to the number of rows of the matrices. For each instance $A\mathbf{x} = \mathbf{b}$ in (D1), we obtain the corresponding instance of (D2) as follows:

1. generate a random support S with $\text{card}(S) = m$, where m is the number of rows, and a random solution vector $\mathbf{x}$ supported at S;

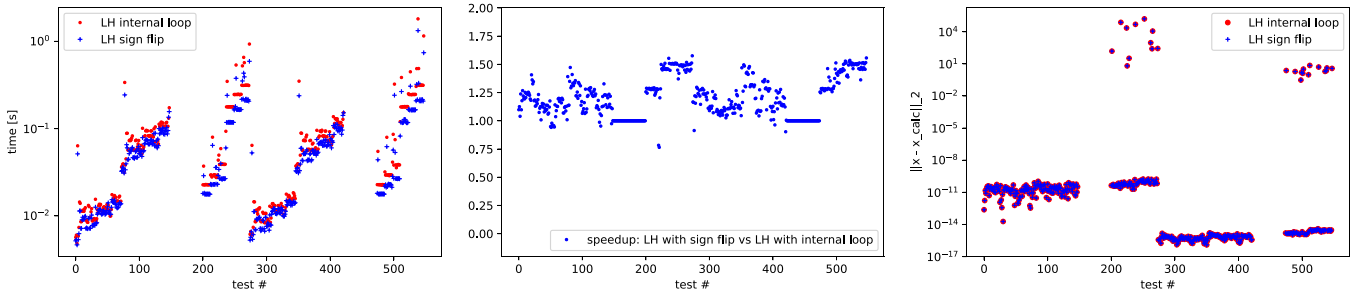


FIGURE 1 Execution times (left), speedup (center) and distance to optimum $\|\mathbf{x} - \mathbf{x}_{calc}\|_2$ (right) of the LH algorithm with sign flip versus LH with inner loop for the dataset (D1)

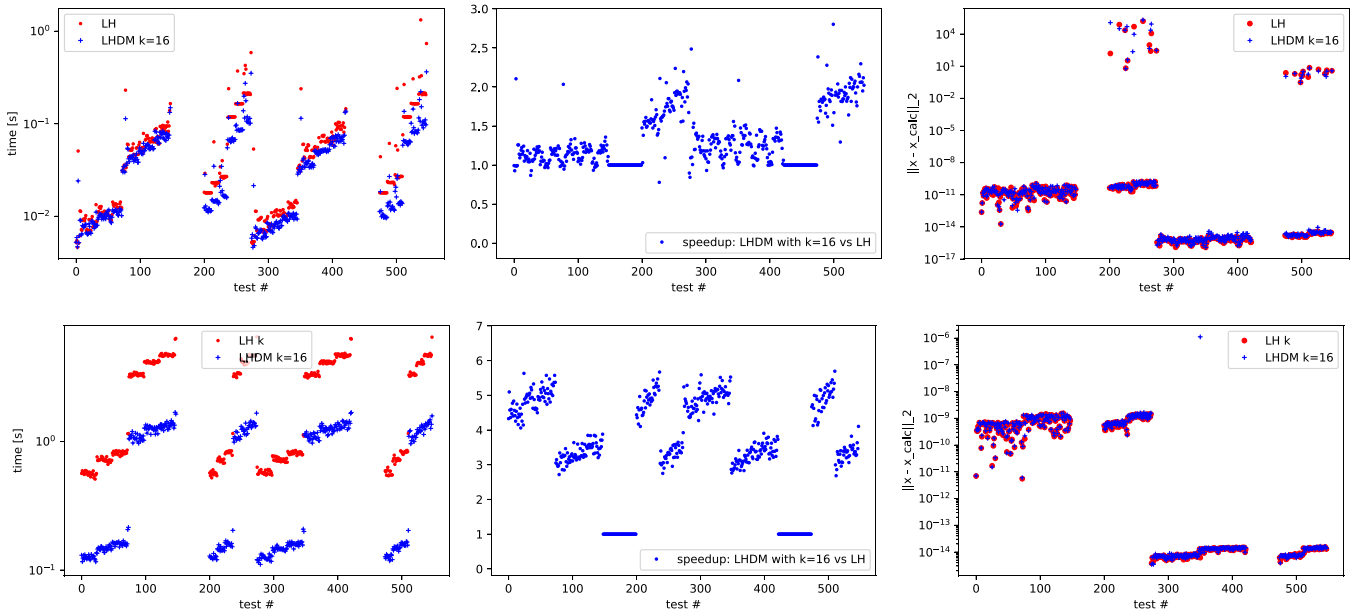


FIGURE 2 Execution times (left), speedup (center) and distance to optimum $\|\mathbf{x} - \mathbf{x}_{calc}\|_2$ (right) of LHDM versus LH. Figures on the top show results on the the dataset (D1); figures on the bottom show results for the dataset (D2)

2. modify the matrix until it satisfies the ERC condition (23), by adding an increasing constant value to the main diagonal of the submatrix A_S whose columns correspond to the support S of the solution $\mathbf{x}$;
3. generate the corresponding right-hand side $\mathbf{b} = A\mathbf{x}$.

5.3 | Comparison with the other convex relaxation solvers

In Figure 3, we compare LHDM with the other convex relaxation solvers listed at the beginning of this section. Tests are performed on instances of the dataset (D1). The algorithm LHDM succeeds in recovery the sparsest solution for instances satisfying the ERC (figures on the right) and it recovers a solution of comparable cardinality for non-ERC instances. In terms of execution times, the algorithm LHDM outperforms all ℓ_1 solvers except SolveOMP (an implementation of the Orthogonal Matching Pursuit), that surprisingly has not been considered in previous comparison studies[22] for theoretical reasons, but actually outperforms all the other methods on instances of the dataset (D1). If we carry out tests on instances of the dataset (D2) introduced in Section 5.2, some relaxation methods (l1_magic, SolveBP, isal1) regain efficiency and compete with LHDM and OMP (figures are omitted, but the software is available for testing). Actually, this is a limit case for sparse recovery, and in the overall sparsity range they perform worse. Therefore, in the next subsection we detail the comparison between LHDM and SolveOMP, which is an algorithm much more closely related to LH and thus deserves a better comprehension for our aim.

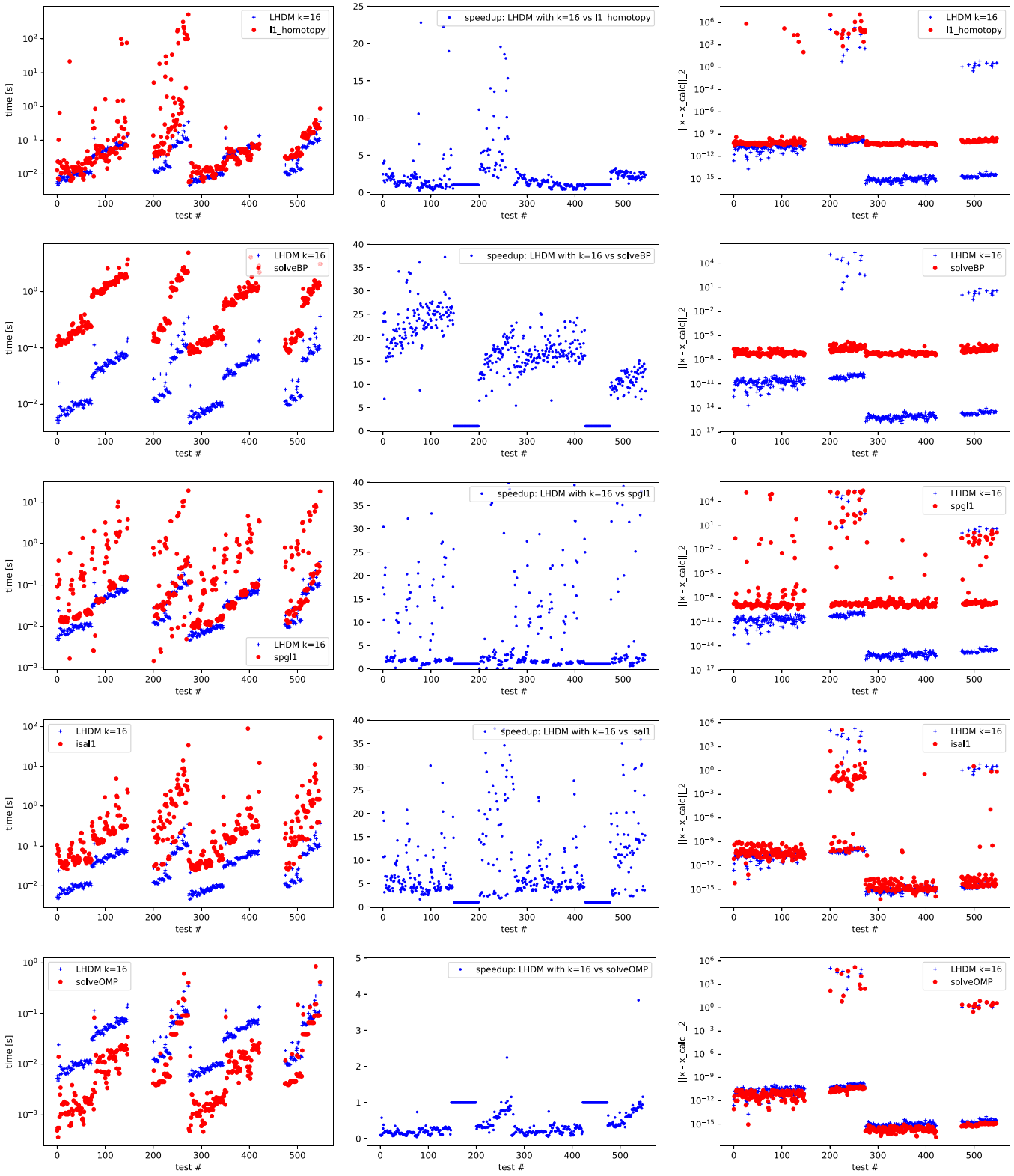


FIGURE 3 Execution times (left), speedup (center) and distance to optimum $\|\mathbf{x} - \mathbf{x}_{calc}\|_2$ (right) of LHDM versus ℓ_1 solvers for the dataset (D1). Each line corresponds to a different solver, in this order starting at the top: l1_homotopy, solveBP, spgl1, isal1 and solveOMP

5.4 | LHDM versus SolveOMP and the DM role

We show results using three different datasets in order to investigate the performance of LHDM ans SolveOMP with respect to the degree of sparsity, condition number of the system matrix, and nonnegativity of the solution.

Let us start with the dataset (D2) introduced in in Section 5.2, where solution vectors have a number of nonzeros which is equal to the number of rows. As Figure 4 shows, in this case LHDM outperforms SolveOMP, with a substantial speedup (compare with Figure 3, where the original database with very sparse solution vectors is used). If we consider also the speedup gain of LHDM versus LH on the same modified dataset, shown in Figure 2, we can conclude that the performance gain of LHDM versus SolveOMP is due to the Deviation Maximization (DM) and not to LH alone.

The matrices of the dataset (D1)[22] are very well conditioned (condition number $< 10^2$). In order to investigate the methods' performance with respect to the condition number, we generate a dataset (D3) whose matrices have moderate values of condition numbers, that is, between 10^5 and 10^6. For each instance of the dataset (D1), we obtain the corresponding instance of (D3) as follows. We generate a random matrix A of the same size as in (D1) with a given condition number and then modify it, by adding an increasing constant value to the main diagonal of the submatrix A_S whose columns correspond to the original support S of the solution $\mathbf{x}$, until the ERC condition (23) is satisfied. The right-hand side is generated accordingly to the original solution vector, that is, $\mathbf{b} = A\mathbf{x}$. SolveOMP fails $\approx 70\%$ of instances, while LHDM solves all of them. Moreover, as we can notice in Figure 5, LHDM keeps its performance gain versus LH, independently from the conditioning of the problem (compare with Figure 2). Moreover, in this case also the convex relaxation solvers here tested behave badly: they fail in retrieving the sparsest solution and they exhibit longer execution times (figures are omitted).

Last, we generate the dataset (D4) whose instances have nonnegative solution vectors. For each instance of (D1), we obtain the corresponding instance of (D4) by taking the component-wise absolute value of the solution vector and then generating the corresponding right-hand side. Note that this does not compromise the ERC property, which only depends on the support of the solution. In this case, SolveOMP does not retrieve the sparsest solution, while LHDM does. The convex relaxation solvers here tested do not retrieve the solution either. Again, as Figure 6 shows, LHDM keeps its performance gain versus LH.

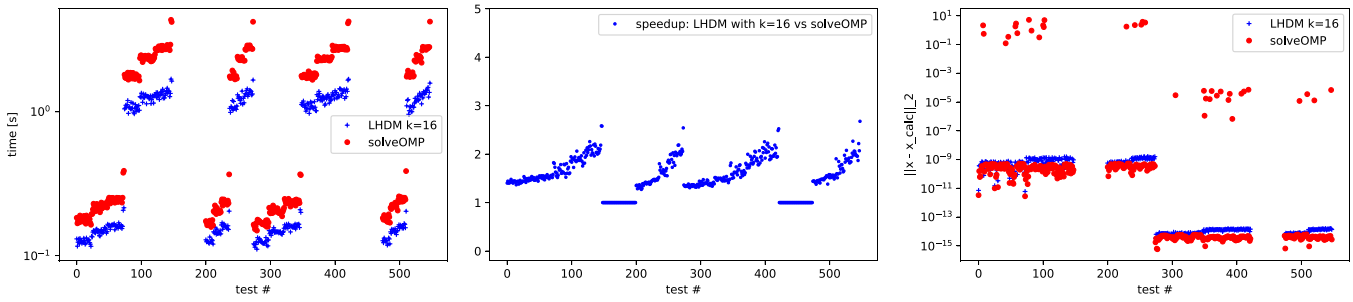


FIGURE 4 Execution times (left), speedup (center) and distance to optimum $\|\mathbf{x} - \mathbf{x}_{calc}\|_2$ (right) of LHDM versus SolveOMP for the dataset (D2)

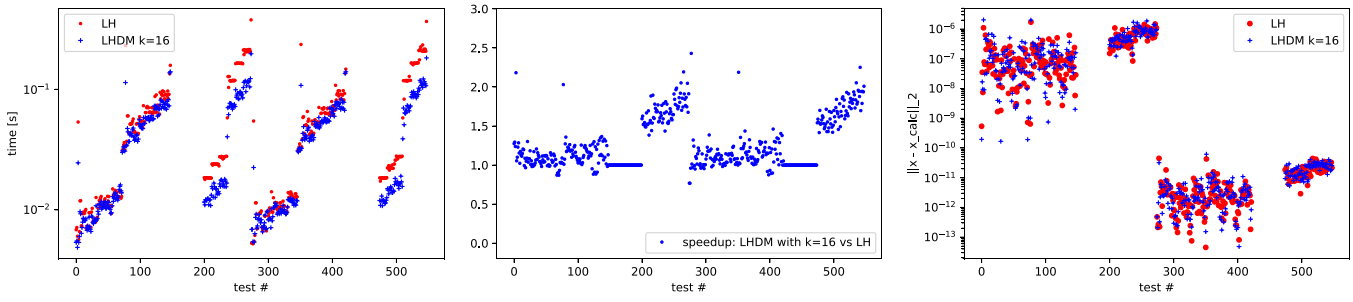


FIGURE 5 Execution times (left), speedup (center) and distance to optimum $\|\mathbf{x} - \mathbf{x}_{calc}\|_2$ (right) of LHDM versus LH for the dataset (D3)

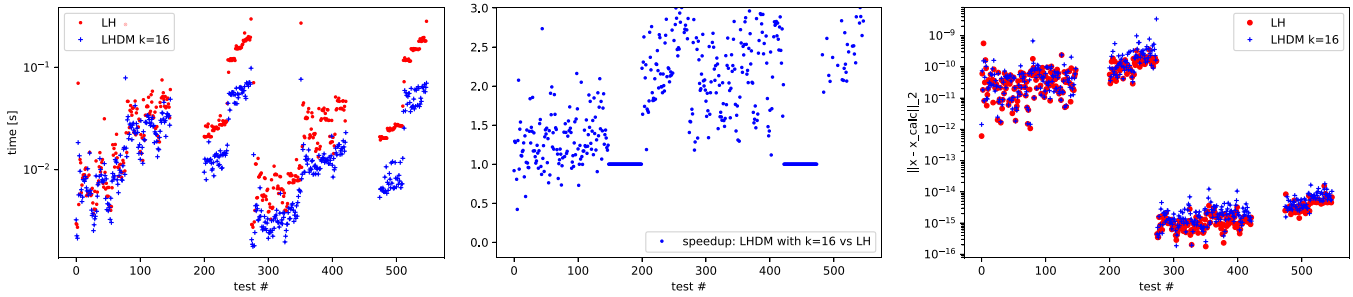


FIGURE 6 Execution times (left), speedup of (center) and distance to optimum $\|\mathbf{x} - \mathbf{x}_{calc}\|_2$ (right) LHDM versus LH for the dataset (D4)

6 | CONCLUSIONS

In this work we have presented an improved version of the Lawson-Hanson with Deviation Maximization algorithm (LHDM), for which finite termination is theoretically proved. We have detailed its efficient implementation, through Householder and Givens orthogonal transformations. In particular, the algorithmic clue given for the positivity-trick case in Section 4.2 is by itself a novel contribution, to the best of our knowledge. Extensive numerical experiments have been carried out over a wide set of instances, confirming that LHDM yields a significant performance gain over LH with an average speedup of 3× with peaks up to 6×. Numerical testing has confirmed that LHDM is competitive also with existing ℓ_1 solvers for sparse recovery in terms of solution quality and execution times on a wide class of instances, in particular for moderately ill-conditioned matrices or for nonnegative solutions, where ℓ_1 solvers fail in retrieving the solution. The proposed LHDM algorithm exploits BLAS–3 operations for efficiency. A thorough experimental assessment is not possible in this Matlab implementation and it would require an implementation in a compiled programming language, for example, the C language. This is an ongoing project, for an open-source delivery. We did not cope with sparse instances, since it would require a dedicated implementation based on sparse QR with column pivoting and it would be interesting to compare with randomization techniques.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the doctoral grant funded by BeanTech s.r.l. "GPU computing for modeling, nonlinear optimization and machine learning". This work was partially supported by the Project BIRD192932 of the University of Padova.

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are described in the article Lorenz DA, Pfetsch ME, and Tillmann AM. Solving basis pursuit: heuristic optimality check and solver comparison. ACM Trans Math Softw. 2015 Feb;41(2), https://doi.org/10.1145/2689662 and modifications described in this article.

REFERENCES

1. Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature. 1999;401(6755):788–91. Available from. https://doi.org/10.1038/44565
2. Gillis N. Nonnegative matrix factorization. Philadelphia: SIAM; 2020.
3. Bruckstein AM, Elad M, Zibulevsky M. On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. IEEE Trans Inf Theory. 2008;54(11):4813–20.
4. Foucart S, Koslicki D. Sparse recovery by means of nonnegative least squares. Signal Process Lett IEEE. 2014;4(21):498–502.
5. Wang M, Tang A. Conditions for a unique non-negative solution to an underdetermined system. Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing. Allerton'09, IEEE Press; 2009:301–307.
6. Wang M, Xu W, Tang A. A unique "nonnegative" solution to an underdetermined system: from vectors to matrices. IEEE Trans Signal Process. 2011;59(3):1007–16.
7. Chen SS, Donoho DL, Saunders MA. Atomic decomposition by basis pursuit. SIAM Rev. 2001;43(1):129–59. Available from. https://doi.org/10.1137/S003614450037906X

8. Donoho DL, Huo X. Uncertainty principles and ideal atomic decomposition. IEEE Trans Inf Theory. 2001;47(7):2845–62.
9. Elad M, Bruckstein AM. A generalized uncertainty principle and sparse representation in pairs of bases. IEEE Trans Inf Theory. 2002;48:2558–67.
10. Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc Ser B (Methodol). 1996;58(1):267–88. Available from. http://www.jstor.org/stable/2346178
11. Lawson CL, Hanson RJ. Solving least squares problems. Vol 15. Philadelphia: SIAM; 1995.
12. Bischof C, Hansen P. A block algorithm for computing rank-revealing QR factorizations. Numer Algorithms. 1992;10(2):371–91.
13. Bischof C, Quintana-Ortí G. Computing rank-revealing QR factorizations of dense matrices. ACM Trans Math Softw. 1998;6(24):226–53.
14. Bischof C, Quintana-Ortí G. Algorithm 782: codes for rank-revealing QR factorizations of dense matrices. ACM Trans Math Softw. 1998;7(24):254–7.
15. Quintana-Ortí G, Sun X, Bischof CH. A BLAS-3 version of the QR factorization with column pivoting. SIAM J Sci Comput. 1998;19(5):1486–94.
16. Bischof JR. A block QR factorization algorithm using restricted pivoting. Supercomputing '89: Proceedings of the 1989 ACM/IEEE Conference on Supercomputing; 1989:248–256.
17. Bro R, Jong S. A fast non-negativity-constrained least squares algorithm. J Chemom. 1997;9(11):393–401.
18. Van Benthem MH, Keenan MR. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. J Chemom. 2004;18(10):441–50. Available from. https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.10%02/cem.889
19. Portugal LF, Júdice JJ, Vicente LN. A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables. Math Comput. 1994;63(208):625–43.
20. Dessole M, Marcuzzi F. Deviation maximization for rank-revealing QR factorizations. Numer Algorithms. 2022;91:1047–1079. Available from. https://doi.org/10.1007/s11075-022-01291-1
21. Dessole M, Marcuzzi F, Vianello M. Accelerating the Lawson-Hanson NNLS solver for large-scale Tchakaloff regression designs. Dolomites Res Notes Approx. 2020;13:20–9.
22. Lorenz DA, Pfetsch ME, Tillmann AM. Solving basis pursuit: heuristic optimality check and solver comparison. ACM Trans Math Softw. 2015;41(2):1–29. Available from. https://doi.org/10.1145/2689662
23. Nocedal J, Wright SJ. Numerical optimization. 2nd ed. New York, NY, USA: Springer; 2006.
24. Stoer J. On the numerical solution of constrained least-squares problems. SIAM J Numer Anal. 1971;8(2):382–411.
25. Donoho DL, Elad M. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. Proc Natl Acad Sci. 2003;100(5):2197–202.
26. Donoho D. Compressed sensing. IEEE Trans Inf Theory. 2006;5(52):1289–306.
27. Elad M. Sparse and redundant representations: from theory to applications in signal and image processing. 1st ed. New York, NY: Springer Publishing Company, Incorporated; 2010.
28. Chen S, Donoho D. Basis pursuit. Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers. vol. 1; 1994:41–44.
29. Candès EJ. The restricted isometry property and its implications for compressed sensing. C R Math. 2008;346(9):589–92. Available from. https://www.sciencedirect.com/science/article/pii/S1631073X08000964
30. Tillmann AM, Pfetsch ME. The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing. IEEE Trans Inf Theory. 2014;60(2):1248–59.
31. Tropp JA. Greed is good: algorithmic results for sparse approximation. IEEE Trans Inf Theory. 2004;50(10):2231–42.
32. Björck A. Numerical methods for least squares problems. Philadelphia: Society for Industrial and Applied Mathematics; 1996. Available from. https://epubs.siam.org/doi/abs/10.1137/1.9781611971484

How to cite this article: Dessole M, Dell'Orto M, Marcuzzi F. The Lawson-Hanson algorithm with deviation maximization: Finite convergence and sparse recovery. Numer Linear Algebra Appl. 2023;e2490. https://doi.org/10.1002/nla.2490