Invariancy of Sparse Recovery Algorithms

Milad Kharratzadeh, Arsalan Sharifnassab, and Massoud Babaie-Zadeh

***Abstract*—In this paper, a property for sparse recovery algorithms, called '*invariancy*', is introduced. The significance of invariancy is that the performance of the algorithms with this property is less affected when the sensing (i.e., the dictionary) is ill-conditioned. This is because for this kind of algorithms, there exists *implicitly* an equivalent well-conditioned problem which is being solved. Some examples of sparse recovery algorithms will also be considered and it will be shown that some of them, such as SL0, Basis Pursuit (using interior point LP solver), FOCUSS, and hard thresholding algorithms are invariant and some others, like Matching Pursuit and SPGL1, are not. Then, as an application example of invariancy property, a sparse-decomposition-based method for Direction of Arrival (DOA) estimation is reviewed, and it is shown that if an invariant algorithm is utilized for solving the corresponding sparse recovery problem, the spatial characteristics of the sensors will have essentially no effect on the final estimation, provided that the number of sensors is large enough.**

***Index Terms*—Invariancy, sparse decomposition, compressed sensing, Invariance of optimization algorithms, Direction of Arrival (DOA) estimation.**

I. Introduction

A. Underdetermined system of linear equations

Let $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_M]$ be an $N \times M$ matrix with $M > N$, where $\mathbf{a}_i$'s, $i = 1, \dots, M$ denote its columns, and consider the Underdetermined System of Linear Equations (USLE)

$$\mathbf{A}\mathbf{s} = \mathbf{x}. \tag{1}$$

Being underdetermined, this system has generally infinitely many solutions. By the sparsest solution of the above system one means a solution, $\mathbf{s}$, which has as small as possible number of non-zero elements. In other words, it is the solution of

$$P_0: \quad \underset{\mathbf{s}}{\text{Minimize}} \ \|\mathbf{s}\|_0 \text{ subject to } \mathbf{x} = \mathbf{A}\mathbf{s}, \tag{2}$$

where $\|\cdot\|_0$ denotes the so called ℓ^0 norm of a vector, that is, the number of its non-zero elements. In signal (or atomic) decomposition viewpoint, $\mathbf{x}$ is a signal which is to be decomposed as a linear combination of $\mathbf{a}_i$'s (columns of $\mathbf{A}$), $i = 1, \dots, M$, hence, $\mathbf{a}_i$'s are usually called 'atoms' [1], and $\mathbf{A}$ is called the 'dictionary' over which the signal is to be decomposed.

This sparse solution of (1), which is unique under certain conditions [2], [3], [4], [5], has attracted the attention of many researchers from different viewpoints during the last decade, because of its numerous applications. It is used, for example, in Compressed Sensing (CS) [6], [7], [8], [9], underdetermined Sparse Component Analysis (SCA) and source separation [10], [11], [12], [13], [14], atomic decomposition on overcomplete dictionaries [15], [16], [17], Blind Source Separation (BSS) [18] of sparse sources, decoding real field codes [19], [20], image deconvolution [21], [22], [23], image denoising [24], [25], electromagnetic imaging, Direction of Arrival (DOA) estimation [26], etc.

Finding the sparsest solution of a USLE by directly solving (2) requires a combinatorial search and is generally NP-hard. Thus, many different algorithms, called sparse recovery algorithms, have been proposed for estimating the sparse solution. Examples include Basis Pursuit (BP) [15], Matching Pursuit (MP) [1], Smoothed L0 (SL0) [27], [28], [29], SPGL1 [30], [31], and FOCUSS [26]. Basis Pursuit, as one of the most successful ideas, minimizes ℓ^1 norm instead of ℓ^0 norm; that is, it solves the problem

$$P_1: \quad \underset{s}{\text{Minimize}} \ \sum_{i=1}^{M} |s_i| \quad \text{subject to } \mathbf{x} = \mathbf{A}\mathbf{s},$$

where s_i's are elements of vector $\mathbf{s}$.

B. Conditioning of a dictionary

In this paper, we use the term 'conditioning' to refer to how 'different' the atoms of the dictionary are. There are many different ways to quantitatively measure conditioning. One of the most prevalent measures is the condition number of a matrix [32], which arises in studying the sensitivity of the solution of a system of linear equations to noise. *Mutual coherence* [2], [3] defined as the maximum correlation coefficient between the columns of the dictionary and Restricted Isometry Property (RIP) constant [33] can also be seen as measures of conditioning of the dictionary (arisen in the context of finding the sparse solution of a USLE). The s-th order *restricted isometry constant* δ_s of matrix $\mathbf{A}$, also called RIP constant or Restricted Isometry Constant (RIC) of $\mathbf{A}$, is defined [33] as the smallest number satisfying:

$$1 - \delta_s \le \frac{\|\mathbf{A}\mathbf{s}\|_2^2}{\|\mathbf{s}\|_2^2} \le 1 + \delta_s, \quad \forall \mathbf{s} \in \mathbb{R}^m: \quad \|\mathbf{s}\|_0 = s.$$

where $\|.\|_2$ represents Euclidean norm.

The performance of a sparse recovery algorithm is generally affected by 'conditioning' of the dictionary. For example, a large mutual coherence means having two very similar atoms (i.e., a bad conditioning in the dictionary) which can cause difficulties for sparse recovery algorithms to find the sparse solution. In general, conditioning of the dictionary can affect sparse recovery algorithms in different ways, such as convergence rate, accuracy of the final estimation, theoretical

Authors are with the Electrical Engineering Department, Sharif University of Technology, Tehran, Iran, P. O. Box 11155-8639 (e-mail: milad.kharratzadeh@gmail.com, sharifnassab@ee.sharif.edu, and mbzadeh@yahoo.com).

performance guarantees (e.g., Theorems 1 and 2 below), and solution sensitivity.

As an example, for Basis Pursuit algorithm, there are different conditions under which the equivalence of P_1 and P_0 problems is guaranteed. Let $M(\mathbf{A})$ denote the mutual coherence of $\mathbf{A}$. It is shown in [2] and [3] that:

Theorem 1. *If (1) has a solution $\mathbf{s}_0$, for which $\|\mathbf{s}_0\|_0 < \frac{1}{2}(1+M(\mathbf{A})^{-1})$, then, it is the unique solution of both ℓ^1 and ℓ^0 minimization problems.*

Another sufficient condition for the equivalence of P_0 and P_1 is given in [33] based on RIC:

Theorem 2. *If $\delta_{2s} < \sqrt{2}-1$, then, P_0 and P_1 problems are equivalent.*

There has been several papers providing larger upper bounds for δ_{2s} than the one in Theorem 2. For instance, recently, the condition has been improved to $\delta_{2s} < 0.493$ [34].

C. Our contributions

In this paper, we introduce a property for sparse recovery algorithms, which we call *invariancy*. We see that a sparse recovery algorithm with such a property is more robust against the conditioning of the dictionary. For example, as a result of this property, the theoretical bounds for performance of some invariant algorithms (e.g., the bounds for BP mentioned in Theorem 1 and 2) can be improved.

After introducing the invariancy property and discussing its significance, the invariancy of some well-known sparse recovery algorithms will be studied. In particular, we show that BP using interior-point Linear Programming (LP) solver, SL0, and FOCUSS are invariant, while MP and SPGL1 are not. As an example application, we consider the problem of Direction of Arrival (DOA) estimation using sparse representation [26]. We show that if an invariant sparse recovery algorithm is used for solving this problem, and if the number of sensors is sufficiently large, then the estimated DOA is almost independent of the positions of the sensors.

Moreover, we show that some non-invariant algorithms can be modified to become invariant. As an example, with a simple modification, we convert a class of greedy algorithms (which are normally non-invariant) to invariant ones. The improvement in performance of the modified algorithms will then be numerically studied.

The paper is organized as follows. In Section II, the invariancy property of sparse recovery algorithms is defined and its significance is discussed. In Section III, we study the invariancy of some of the existing sparse recovery algorithms. Section IV studies the application of invariancy in DOA estimation. Finally, Section V provides some numerical results.

II. Definition of Invariancy and its Significance

A. Definition

Algorithms for finding the sparse solution of (1) take $\mathbf{A}$ and $\mathbf{x}$ as inputs and give $\hat{\mathbf{s}}$ in their output as an estimation of the sparsest solution. Mathematically, such an algorithm and its estimate can be denoted as $\hat{\mathbf{s}} = \mathcal{F}(\mathbf{A}, \mathbf{x})$, in which $\mathcal{F}$ represents the algorithm. Usually, sparse recovery algorithms are iterative, and they can be seen as creating a converging sequence of improving approximate solutions. For an iterative algorithm $\mathcal{F}$, denote the sequence of these approximate solutions by $\{\mathbf{s}_i\}_{i=1}^{K}$, where $\mathbf{s}_K = \hat{\mathbf{s}}$ is the final solution given by the algorithm. We call this sequence of estimations, a *solution path*[1].

Now, consider a second problem,

$$(\mathbf{BA})\mathbf{s} = (\mathbf{Bx}). \tag{3}$$

where $\mathbf{B}$ is an $N \times N$ invertible matrix. This linear transformation of the dictionary may arise for example as a result of sensors displacement (see section IV for an application to DOA estimation). An invariant algorithm is expected to show identical performance—iteration by iteration—when solving problems (1) and (3). More specifically, it is expected that the solution paths that the algorithm gives for these two problems are identical. Formally:

Definition 1 (Invariancy)**.** Consider problems (1) and (3) and assume that an iterative algorithm, $\mathcal{F}$, respectively gives solution paths $\{\mathbf{s}_i\}_{i=1}^{K_1}$ and $\{\tilde{\mathbf{s}}_i\}_{i=1}^{K_2}$ for these problems. Then, the sparse recovery algorithm $\mathcal{F}$ is called *invariant* if for any given $\mathbf{A}$ and $\mathbf{x}$ and any invertible $N \times N$ matrix $\mathbf{B}$,

$$\begin{aligned} K_1 &= K_2, \\ \mathbf{s}_i &= \tilde{\mathbf{s}}_i, \qquad \forall i \in \{1, 2, \cdots, K_1\}. \end{aligned}$$

B. Significance of invariancy

Invariant algorithms have the interesting property of being *more robust against the conditioning of the dictionary*. The reason is the existence of an equivalent, *implicit* problem, with a well-conditioned dictionary. For instance, assume that $\mathbf{A}$ in problem (1) is ill-conditioned (e.g., it has a large RIP). If there exists a matrix $\mathbf{B}$ such that $\mathbf{BA}$ is better conditioned, then problem (3) will have a dictionary which is better conditioned than $\mathbf{A}$. So, an invariant algorithm solves the ill-conditoned problem (1) as easy as the better-conditioned problem (3). We discuss the existence of such $\mathbf{B}$ in appendix A. Note, however, that there is no need to explicitly find $\mathbf{B}$, and its existence is sufficient, as we still solve problem (1) to find the solution.

In other words, the estimation quality of invariant algorithms is less affected by the sensing conditions, due to the existence of an equivalent problem with more proper sensing conditions. Since invariant algorithms find the same solutions for both problems, we can say that they are more robust to ill-conditioned sensing.

For example, in atomic decomposition, if there are two highly correlated atoms, the dictionary will have relatively high mutual coherence and thus will be ill-conditioned, in mutual coherence sense. In this case, the correlated atoms may be mistaken by the algorithm, affecting its estimation. However, there usually exists a matrix $\mathbf{B}$ that reduces the

[1]The term 'solution path' has been used in many settings to refer to sequences of solutions. For instance, in the sparse regression literature, the 'solution path' of LASSO has been extensively studied (e.g., [35]).

mutual coherence[2], and hence an invariant algorithm will be less affected by the ill-conditioned dictionary, because *implicitly* it is working with a low coherence dictionary.

As another example, in Blind Source Separation (BSS) [18], where the number of sensors is smaller than or equal to the number of sources, one may encounter highly correlated rows in the mixing matrix $\mathbf{A}$, where some sensors are close to each other. Then, $\mathbf{A}$ is ill-conditioned (in condition number sense), and a sparse recovery algorithm may have difficulties in finding the sparse solution of (1), since signals picked up by the close sensors are very similar. However, if the algorithm is invariant, it would be insensitive to this high correlation between the rows. This is because there exists a matrix $\mathbf{B}$, for which $\mathbf{BA}$ has orthogonal rows (this $\mathbf{B}$ can be obtained for example by Gram-Schmidt orthogonalization; see also Appendix A), and the invariant algorithm is *implicitly* working with this new mixing matrix with orthogonal rows. Another practical example will be studied in detail in Section IV.

C. Improved performance bounds

In addition to the robustness of algorithms, the analytical guarantee for performance of some invariant algorithms can be improved, since we can use the performance bounds of well-conditioned problems for ill-conditioned ones. For example, the guarantee bound of BP given in Theorem 1 can be enhanced as follows.

Proposition 1. Let:

$$\tilde{M}(\mathbf{A}) \triangleq \inf_{\mathbf{B}} \left(M(\mathbf{BA}) \right),$$

where $M(\cdot)$ is the mutual coherence and the infimum is taken over invertible square matrices $\mathbf{B}$. Then, $\mathbf{s}_0$ is the unique solution of P_1 and P_0 problems if

$$\|\mathbf{s}_0\|_0 < \frac{1}{2}\left(1 + \tilde{M}(\mathbf{A})^{-1}\right). \tag{4}$$

Proof: First, note that $M(\mathbf{BA})$ is always greater than or equal to zero, and its infimum exists. Thus, there is a sequence of invertible matrices $\mathbf{B}_i$ for which $M(\mathbf{B}_i\mathbf{A})$ converges to $\inf M(\mathbf{BA})$ over invertible matrices $\mathbf{B}$. Therefore, it follows from (4) that there exists a $\mathbf{B}$ such that $\frac{1}{2}\left(1 + M(\mathbf{BA})^{-1}\right) - \|\mathbf{s}_0\|_0 > 0$. So, on the grounds of Theorem 1, $\mathbf{s}_0$ is the unique solution of P_1 and P_0 problems for (3). However, every solution of (1) is a solution of (3) and vice versa. Hence, P_1 and P_0 are equivalent also for (1). ∎

The upper bound given in Proposition 1 is larger than or equal to the bound in Theorem 1, because by definition $\tilde{M}(\mathbf{A}) \leq M(\mathbf{A})$. There are some numerical results in the Appendix A, presenting upper bounds for the average value of $\tilde{M}(\mathbf{A})/M(\mathbf{A})$ and the average value of the ratio between the bounds given by Theorem 1 and Proposition 1, in the case of randomly generated matrices.

Similarly, the following proposition improves the bound given in Theorem 2 for equivalence of P_0 and P_1.

[2]For example, if $\mathbf{A}$ is an invertible square matrix, by multiplying it by its inverse, $\mathbf{A}^{-1}$, the mutual coherence of the resulted dictionary (identity dictionary) will be zero. Refer to appendix A for further discussions and algorithmic approaches to find such a $\mathbf{B}$.

Proposition 2. Define the s-th order enhanced RIC of matrix $\mathbf{A}$ as:

$$\tilde{\delta}_s(\mathbf{A}) \triangleq \inf_{\mathbf{B}} \delta_s(\mathbf{BA}), \tag{5}$$

where $\delta_s(\mathbf{A})$ is the RIC of $\mathbf{A}$. Then, if $\tilde{\delta}_{2s} < \sqrt{2}-1$ (or any of the improved upper bounds), the solutions of P_0 and P_1 found by BP are equal (i.e., P_0 and P_1 problems are equivalent).

Since $\tilde{\delta}_{2s} \leq \delta_{2s}$, Proposition 2 presents a stronger statement than Theorem 2. Note, however, that since the computation of the RIC is an NP hard problem, so is the computation of $\tilde{\delta}_s$ in (5). We will present some numerical results in Appendix A in which we obtain lower bounds for how much (5) enhances the RIC for random matrices of small sizes, for which the RICs can be practically computed.

As an example, it will be seen in Section III that BP with interior-point LP solver is invariant, and hence its successful recovery is guaranteed under conditions (4) and (5).

D. Performance enhancement of non-invariant algorithms

Having the invariancy insight in mind, it may be possible to increase the robustness of some non-invariant algorithms against ill-conditioning by modifying them to become invariant. For example, Matching Pursuit (MP) family of algorithms, namely original MP, Orthogonal MP (OMP) [36], Regularized OMP (ROMP) [37], and Compressive Sampling MP (CoSaMP) [38] as well as Iterative hard thresholding (IHT) [39] and hard thresholding pursuit (HTP) [40] algorithms are non-invariant (as will be studied in Section III). However, we can modify these algorithms to become invariant. This modification is done by adding an initial step of orthogonalizing the rows of the dictionary before utilizing the algorithm to solve the P_0 problem. In other words, one should first find a matrix $\mathbf{C}$, for which the rows of $\mathbf{CA}$ are orthonormal (such a $\mathbf{C}$ can be found for example from Gram-Schmidt Orthogonalization; note that $\mathbf{C}$ is different for different dictionaries and is not unique), and then utilize the greedy algorithm to find the sparsest solution of $\mathbf{CAs} = \mathbf{Cx}$. We call the resulting algorithms (with the orthogonalization step) IMP, IOMP, IROMP, ICoSaMP, IIHT, and IHTP respectively, whose invariancy will be verified in Section III.

For instance, it is guaranteed that for the s-sparse solution vector $\mathbf{s}$, if $\delta_{4s} \leq 0.1$, then the CoSaMP algorithm finds the sparsest solution $\mathbf{s}$ of the P_0 problem correctly [38]. Considering ICoSaMP being invariant, we can state a similar performance guarantee condition for ICoSaMP as in the following proposition.

Proposition 3. For a given dictionary $\mathbf{A}$, let $\mathbf{C}$ denote a matrix for which the rows of $\mathbf{CA}$ are orthonormal. Also define $\bar{\delta}_s(\mathbf{A}) \triangleq \delta_s(\mathbf{CA})$. Then, if P_0 has an s-sparse solution and $\bar{\delta}_{4s}(\mathbf{A}) \leq 0.1$, then ICoSaMP finds this solution correctly.

Note that row orthogonalization cannot convert every non-invariant algorithm to an invariant one. In fact, it only works for algorithms having the property $\mathcal{F}(\mathbf{UA}, \mathbf{Ux}) = \mathcal{F}(\mathbf{A}, \mathbf{x})$, for all unitary matrices $\mathbf{U}$. Row orthogonalization is just an example we used here to achieve invariancy for MP family. However, it is neither the only, nor necessarily the optimal

approach. For example, we could achieve invariancy by choosing a $\mathbf{C}$ that minimizes the RIC of $\mathbf{CA}$ (which is of course impractical, since finding such $\mathbf{C}$ is NP–hard). Let us call this algorithm that preconditions $\mathbf{A}$ using this specific $\mathbf{C}$ before applying CoSaMP, ICoSaMP2 (to diffrentiate it with ICoSaMP that applies an orthogonalizing $\mathbf{C}$). Thus, the performance bound of CoSaMP would be further improved by ICoSaMP2 than by ICoSaMP.

A similar concept in the literature is the idea of pre-multiplying the measurement matrix and observation vector by an invertible matrix which is known to be useful in the optimization literature (where it is known as "preconditioning") and has also been used in the context of sparse signal recovery [41], [42], [43], [44]. In our proposed framework, the preconditioning techniques can be seen as ways to improve the performance of non–invariant algorithms. Note that the definition of invariancy implies that invariant algorithms do *not* need preconditioning as they are handling it inherently (i.e., without actually computing and pre-multiplying any matrices).

In Section V, we show through some numerical experiments how the proposed modifications (e.g., orthogonalization) improves the performance of these algorithms.

E. Presence of noise

In practice, there is always a noise level present in SCA problems, and one needs to find the sparse solution of $\mathbf{x} = \mathbf{As} + \mathbf{n}$, where, $\mathbf{n}$ is the $n \times 1$ vector of sensing noise (which is usually assumed to be iid and Gaussian). We are especially interested in the question: how this noise affects the performance of invariant algorithms. Actually, as shown in [16], both P_0 and P_1 problems are inherently unstable against noise, and instead of them, their noise-aware variants should be considered. The key-point in stabilizing these problems is replacing the exact sparse decomposition ($\mathbf{x} = \mathbf{As}$) with an approximate sparse decomposition ($\|\mathbf{x} - \mathbf{As}\|_2 \leq \epsilon$) [45].

Performance analysis of invariant algorithms in the noisy case is probably very tricky, and is not studied in this paper. To see the reason, consider an invariant algorithm $\mathcal{F}$ computing an estimation of $\mathbf{s}$ based on inputs $\mathbf{A}$ and $\mathbf{x}$ for the problem

$$\mathbf{x} = \mathbf{As} + \mathbf{n}_1, \tag{6}$$

Now, if we take an invertible $N \times N$ matrix $\mathbf{B}$ and use $\mathbf{BA}$ as the sensing dictionary, the problem converts to:

$$\mathbf{y} = (\mathbf{BA})\mathbf{s} + \mathbf{n}_2, \tag{7}$$

where $\mathbf{n}_2$ is the noise vector with the same distribution as $\mathbf{n}_1$. By definition, we have $\mathcal{F}(\mathbf{BA}, \mathbf{y}) = \mathcal{F}(\mathbf{A}, \mathbf{x}')$, where $\mathbf{x}' = \mathbf{B}^{-1}\mathbf{y} = \mathbf{As} + \mathbf{B}^{-1}\mathbf{n}_2$. Compared to $\mathbf{x} = \mathbf{As} + \mathbf{n}_1$, the vector $\mathbf{x}'$ may have smaller or larger noise level depending on the singular values of $\mathbf{B}$. Moreover, the entries of the noise term in $\mathbf{x}'$, i.e., $\mathbf{B}^{-1}\mathbf{n}_2$, would not be iid if $\mathbf{B}$ is not unitary.

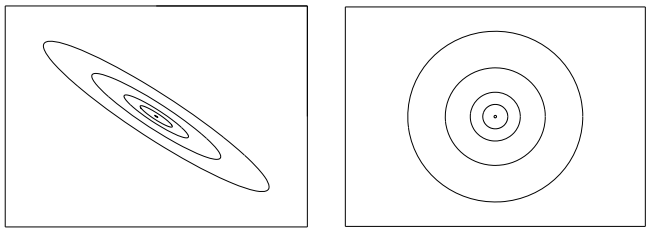
However, the following example shows that to have a formal analysis, we need to make further assumptions. Here, we focus our attention on a special case where $\mathbf{B}$ is unitary. In this case, the distribution of $\mathbf{B}^{-1}\mathbf{n}_2$ would be the same as $\mathbf{n}_1$. Thus, the two problems $\mathbf{As} = \mathbf{x}'$ and $\mathbf{As} = \mathbf{x}$ are equivalent (note that we are dealing with problems, not problem instances). Hence, for an invariant algorithm $\mathcal{F}$, we have $\mathcal{F}(\mathbf{BA}, \mathbf{y}) = \mathcal{F}(\mathbf{A}, \mathbf{x}') = \mathcal{F}(\mathbf{A}, \mathbf{x})$. In other words, performance of invariant algorithms in presence of noise is not affected if the dictionary undergoes unitary linear transformations.

Although formal analysis of invariant algorithms for the noisy case is tricky, intuition proposes that if the amount of noise is "small enough", then the performance of invariant algorithms is not highly affected by the conditioning of the dictionary, where "small enough" depends actually on the conditioning of the dictionary. A similar intuitive statement has been stated in [46, Section IV-D] in discussion of the affect of noise on the performance of '*equivariant*' BSS algorithms (see the next subsection for a brief review on equivariancy in BSS community).

F. Related concepts in other fields

A similar concept, also called *invariancy* (or *affine invariancy*), exists in optimization [47, Section 3.3] [48]. To review the definition of invariancy in that context, consider the problem of optimizing a cost function $f(\mathbf{x})$ and an iterative optimization algorithm which produces a sequence of points, $\{\mathbf{x}_0, \mathbf{x}_1, \dots\}$, converging to the estimated optimizer of this function. Consider now the affine transformation $\mathbf{y} = \mathbf{Bx} + \mathbf{b}$, where $\mathbf{B}$ is an invertible square matrix. Then, a function $f(\mathbf{x})$ can be regarded as being computed either from $\mathbf{x}$ (say $f_x(\mathbf{x})$) or from $\mathbf{y}$ (say $f_y(\mathbf{y}) = f_x(\mathbf{B}^{-1}(\mathbf{y} - \mathbf{b}))$). Let now $\{\mathbf{x}_0, \mathbf{x}_1, \dots\}$ be the sequence produced by applying an optimization algorithm on $f_x(\mathbf{x})$, and $\{\mathbf{y}_0, \mathbf{y}_1, \dots\}$ be the sequence produced by applying it on $f_y(\mathbf{y})$. If for all k, $\mathbf{y}_k = \mathbf{Bx}_k + \mathbf{b}$, then this optimization algorithm is said to be *invariant*. It is important to note on the significance of this property: an invariant optimization algorithm is not affected by the conditioning of the Hessian of the cost function; this is because it is '*implicitly*' working in another domain with a well-conditioned Hessian matrix. In fact, the problem can be transformed to another domain in which the Hessian is the identity matrix [47] (see Fig.1). For example, steepest descent algorithm is not invariant, while Newton algorithm is [47]. Hence, large condition number of the Hessian of the cost function is not important for the Newton algorithm, while it has a significant impact on the convergence of the steepest descent algorithm. Another well-known example is the conjugate gradient algorithm which is widely used in many applications because of its invariancy [47]. See [48] for more details.

A relatively similar concept exists in BSS which is called '*equivariancy*' [46]. To express the BSS problem, suppose that M source signals are recorded by N sensors, each of which records a combination of all sources. In linear instantaneous (noiseless) model, it is assumed that $\mathbf{x}(t) = \mathbf{As}(t)$ in which $\mathbf{x}(t) = [x_1(t), \dots, x_N(t)]^T$ and $\mathbf{s}(t) = [s_1(t), \dots, s_M(t)]^T$ are the $N \times 1$ and $M \times 1$ vectors of source and recorded signals, respectively, and $\mathbf{A}$ is the $N \times M$ (unknown) mixing matrix. The goal of BSS is then to find $\mathbf{s}(t)$ only by observing $\mathbf{x}(t)$ (hence the term '*Blind*'). If the number of sensors is equal to the number of sources ($M = N$), then a



(a) (b)

Fig. 1. Contours of cost functions of optimization problems with Hessian matrices (a) $\mathbf{H}_1 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$ and (b) $\mathbf{H}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Although non-invariant algorithms may have difficulty working with ill-conditioned Hessian of the first problem (e.g., in terms of speed of convergence) it is not different for an invariant optimization algorithm to solve either problem.

BSS algorithm takes the observed vectors $\mathbf{x}(t)$ and estimates the source vectors $\hat{\mathbf{s}}(t)$, usually by estimating the mixing matrix. Mathematically, such an algorithm and its estimate can be denoted as $\hat{\mathbf{A}}(t) = \mathcal{G}(\mathbf{x}(t))$, in which $\mathcal{G}$ represents the algorithm. A BSS algorithm is said to be '*equivariant*' if $\mathcal{G}(\mathbf{B}\mathbf{x}(t)) = \mathbf{B}\mathcal{G}(\mathbf{x}(t))$. The significance of equivariant algorithms is that they can separate the sources without sensitivity to the '*hardness*' of the mixing system, *i.e.*, the conditioning of the mixing matrix (mathematically, $\hat{\mathbf{s}}(t) = \mathcal{G}(\mathbf{x}(t))^{-1}\mathbf{x}(t) = [\mathbf{A}\mathcal{G}(\mathbf{s}(t))]^{-1}\mathbf{A}\mathbf{s}(t) = \mathcal{G}(\mathbf{s}(t))^{-1}\mathbf{s}(t)$). For example, an equivariant BSS algorithm retrieves the sources that are mixed by $\mathbf{A} = [1, 0.99;\ 0.99, 1]$ with the same quality as they were mixed by $\mathbf{A} = [1, 0.1;\ 0.1, 1]$. This is because an equivariant algorithm *implicitly* is working on a problem in which sources are not mixed at all. This is derived by setting $\mathbf{B} = \mathbf{A}^{-1}$.

As mentioned earlier, pre-multiplying the measurement matrix and observation vector by an invertible matrix, known as preconditioning, is useful in the optimization and sparse signal recovery literature [41], [42], [43], [44]. We differ from the preconditioning literature in an important way (in fact, the two approaches can be seen as complementary). To see this, note that by introducing invariancy, we show that invariant algorithms do not need preconditioning as they are handling it inherently. Moreover, we show how some performance bounds can be improved by the invariancy concept. Therefore, using preconditioning is only meaningful in the case of non-invariant algorithms. In appendix A, we discuss and introduce some novel preconditioning techniques based on the concept of invariancy.

III. Invariancy of Some Sparse Recovery Algorithms

In this section, we study the invariancy of some existing sparse recovery algorithms such as BP [49], SL0 [28], MP [1], IMP (see Section II-D), and FOCUSS [26]. The results are stated in the form of a set of lemmas with proofs in the Appendix B.

A. Basis Pursuit (BP)

BP, that is solving P_1 instead of P_0, is not by itself an algorithm; it is just an approach for estimating the solution of P_0. So, depending on the algorithm used for solving P_1, it can be invariant or non-invariant. Usually, BP is solved by converting the problem to linear programming (LP), and so, its invariancy depends on the LP solver that is used.

Lemma 1. BP based on primal-dual interior-point algorithm [50] (as its LP solver) is invariant.

Lemma 2. Basis Pursuit with SPGL1 [30], [31] implementation is non-invariant.

Note that P_1 is a convex problem and both BP, SPGL1 and many other algorithms are guaranteed to find its exact solution. However, non-invariant algorithms like SPGL1 require substantially more effort to converge when the dictionary tends to be ill-conditioned. As a result, in the case the maximum number of iterations is limited or some tolerance parameters are not chosen to be sufficiently small, the non-invariant algorithm fails to converge to the correct solution for ill-conditioned dictionaries. Analytically, by being more robust against the sensing conditions, the invariant algorithms converge in the ill-conditioned problems, as quickly as in the case of well-conditioned problems.

B. Smoothed ℓ^0 (SL0)

SL0 is another successful algorithm for sparse recovery, in which the l^0 norm is approximated by a series of smooth functions whose minima can be obtained by gradient projection [28]. More precisely, let:

$$F_\sigma(\mathbf{s}) \triangleq \sum_{i=1}^{M} \exp(-\frac{{s_i}^2}{2\sigma^2}),$$

where $\mathbf{s}$ is a vector of length M with elements s_i. When $\sigma \to 0$, F_σ gives an approximation to $M - \|\mathbf{s}\|_0$. Hence, SL0 presents the following idea:

1) Take the starting point as $\mathbf{s}_0 = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{x}$, which is the maximum of F_∞ [29]. Choose a suitable decreasing sequence for σ, i.e. $[\sigma_1, \cdots, \sigma_k]$, where k is the number of iterations.
2) For each i, maximize (approximately) the function F_{σ_i} by applying a fixed number of Gradient-Projection iterations, that is, $\mathbf{s} \leftarrow \mathbf{s} + (\mu\sigma^2)\nabla F_\sigma$ followed by $\mathbf{s} \leftarrow \mathbf{s} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{s} - \mathbf{x})$.

Lemma 3. SL0 is invariant.

Remark 1. The complex-valued version of SL0 [28] is obtained just by replacing transpositions with Hermitians in (28). All the arguments made for proving the invariancy of the real-valued SL0 (in the appendix) still hold after replacing transpositions with Hermitians and hence, complex-valued SL0 is also invariant. The same argument can be made for complex-valued BP introduced in [51].

C. Matching Pursuit (MP) family

MP is a greedy algorithm that expands the support set of $\mathbf{s}$ in each iteration such that the correlation of the new atom with the residual of the previous iteration is maximized. The first step is to find the column of $\mathbf{A}$ (atom) that has the largest

correlation with $\mathbf{x}$. Then this atom is added to the set of selected variables and an approximation of $\mathbf{x}$ based on this selected variable is calculated. In the next iteration, the atom that has the largest correlation with the residual (difference between actual and approximated $\mathbf{x}$) is added to the set of selected atoms, a new approximation and residual based on the new set of selected atoms is calculated, and the whole process continues until a convergence criterion is met.

Lemma 4. MP, OMP, ROMP, and CoSaMP are non-invariant.

Lemma 5. The modified greedy algorithms IMP, IOMP, IROMP, and ICoSaMP, which are obtained form MP, OMP, ROMP, and CoSaMP by row orthogonalization (discussed in Section II-D) are invariant.

D. FOCUSS

In FOCUSS [26], the starting point of the algorithm is supposed to be chosen based on domain-specific knowledge for the application. In the case that there is no suitable solution available, it is shown through simulation in [52] that the minimum ℓ^2 norm solution is a good choice. In the following lemma, a starting point is invariant if it is identical for problems $\mathbf{As} = \mathbf{x}$ and $\mathbf{BAs} = \mathbf{Bx}$ for any given $\mathbf{A}$, $\mathbf{x}$, and square invertible $\mathbf{B}$. An example of an invariant starting point is the minimum l^2 norm solution.

Lemma 6. FOCUSS algorithm is invariant if it starts from an invariant starting point.

E. Exhaustive search

Exhaustive search solves a system of linear equations for every support of $\mathbf{s}$ of size rank$(\mathbf{A})$ and chooses the solution with least l^0. Although exhaustive search is guaranteed to find the exact solution, it needs combinatorially many computations which is impractical. The next lemma states that this algorithm is invariant.

Lemma 7. Exhaustive search algorithm is invariant.

This is an example which shows invariancy property does not reflect efficiency of algorithms, but only robustness against sensing conditions, in the sense of uniform performance.

F. Hard thresholding algorithms

Iterative hard thresholding (IHT) [39] and hard thresholding pursuit (HTP) [40] are two other iterative algorithms for sparse recovery. To find the s-sparse solution, IHT typically starts with $\mathbf{s}_0 = \mathbf{0}$ and updates the solution at the i'th iteration as follows:

$$\text{IHT:} \qquad \mathbf{s}_{i+1} = H_s(\mathbf{s}_i + \mathbf{A}^T(\mathbf{x} - \mathbf{A}\mathbf{s}_i)), \tag{8}$$

where H_s is the hard thresholding operator that keeps the s largest components of a vector and sets the rest to zero. HTP extends this simple algorithm and first, finds a good candidate for the support and then, finds the vector with this support that best fits the measurements:

$$\text{HTP1: } T_{i+1} = \{\text{indices of the } s \text{ largest entries of } \mathbf{s}_i + \mathbf{A}^T(\mathbf{x} - \mathbf{A}\mathbf{s}_i)\} \tag{9}$$

$$\text{HTP2: } \mathbf{s}_{i+1} = \text{argmin}\{\|\mathbf{As} - \mathbf{x}\|_2,\ \text{supp}(\mathbf{s}) \in T_{i+1}\} \tag{10}$$

Lemma 8. IHT and HTP are non-invariant.

Lemma 9. The modified greedy algorithms IIHT and IHTP, which are obtained form IHT and HTP by row orthogonalization (discussed in Section II-D) are invariant.

IV. An Application Example of Invariancy: Invariant DOA Estimation Methods

In this section, we consider an application of sparse decomposition: Direction of Arrival (DOA) estimation [26]. We show that utilizing invariant algorithms for solving the resulted sparse decomposition problem leads to DOA estimators which are almost invariant to the spatial locations of the sensors, provided that the number of sensors is sufficiently large.

A. DOA estimation via sparse decomposition

Consider the following equation for a plane wave [53]:

$$\mathbf{E}(r, \varphi) = e^{jkr\cos(\varphi - \theta)}, \tag{11}$$

where $\mathbf{E}$ is the electric field, k is the wave number defined as $\frac{2\pi}{\lambda}$ (in which λ is the wavelength), (r, φ) are the polar coordinates, and θ is the direction of arrival of the wave. Here, only a narrow band signal model is considered. In this paper, the data vector received at the array of sensors due to a plane wave in the direction θ is briefly called *data vector at direction* θ, and according to (11) is equal to:

$$\mathbf{a}(\theta) = [e^{jkr_1\cos(\varphi_1 - \theta)}, \cdots, e^{jkr_N\cos(\varphi_N - \theta)}]^T, \tag{12}$$

where $\{(r_i, \varphi_i) :\ i = 1, \cdots, N\}$ are the polar coordinates of the sensors and N is the number of sensors.

The objective in DOA estimation is to find the angles of arrivals of some impinging waves on a set of sensors, based on the observed data vector at these sensors. Here, only two-dimensional DOA estimation is studied and thus only one angle (the so-called DOA) should be estimated for each wave. There are many classic methods for estimating DOA, including MUSIC [54], ESPRIT [55], and Matrix Pencil [56].

Gorodnitsky and Rao [26] converted the DOA estimation problem into a sparse decomposition problem by introducing a dictionary whose columns are data vectors of a set of angles, $\Theta = \{\theta_i | i = 1, \cdots, M\}$, where $-\pi/2 \le \theta_i < \pi/2$. Note that the choice of θ_i's (their total number, M, and whether they are equidistant or not) is arbitrary and depends on the application. They showed that if $\mathbf{x}$ denotes the data vector (outputs of the sensors) and $\mathbf{A}$ denotes this dictionary, one should first find the sparse solution of

$$\mathbf{x} = \underbrace{\left[\mathbf{a}(\theta_1)\middle|\mathbf{a}(\theta_2)\middle|\cdots\middle|\mathbf{a}(\theta_M)\right]}_{\mathbf{A}} \cdot \mathbf{s}, \tag{13}$$

where $\mathbf{a}(\theta_i)$ is the data vector at direction θ_i. Then, the nonzero entries of $\mathbf{s}$ determine the angular directions of the sources that compose the received signal. For example, if the i-th entry of $\mathbf{s}$ is nonzero, it is understood that there is an incoming wave at the angular direction of θ_i, where $\mathbf{a}(\theta_i)$ is the i-th column of the dictionary.

B. Invariant DOA estimation using invariant sparse recovery algorithms

In this section, we show that utilizing invariant algorithms for solving (13) leads to DOA estimation methods which are almost invariant to the locations of sensors, provided that the number of sensors is large enough. To show this, we have to prove that if the number of sensors is sufficiently large, then any two sensor locations leading to $\mathbf{A}_1\mathbf{s} = \mathbf{x}_1$ and $\mathbf{A}_2\mathbf{s} = \mathbf{x}_2$ can be converted to each other through a multiplicative square matrix. In other words, there exists a matrix $\mathbf{B}$ such that $\mathbf{A}_2 \approx \mathbf{B}\mathbf{A}_1$ and $\mathbf{x}_2 \approx \mathbf{B}\mathbf{x}_1$.

We first note that from (12) and (13) the data vectors and the dictionary depend on both the locations of sensors, (r_n, φ_n), and the angular direction of the wave, θ. We use now a decomposition that separates the dependence of data vectors on positions of sensors and propagation direction.

From the series expansion [57, Chapter 9]:

$$e^{jx\cos\alpha} = \sum_{l=-\infty}^{+\infty} (j)^l J_l(x) e^{-jl\alpha},$$

where $J_l(\cdot)$ is the l-th order Bessel function of the first kind, we have:

$$a_n(\theta) = e^{jkr_n\cos(\varphi_n-\theta)} = \sum_{l=-\infty}^{+\infty} (j)^l J_l(kr_n) e^{-jl\varphi_n} e^{jl\theta}, \tag{14}$$

where a_n is the n-th element of the data vector, $\mathbf{a}$. The above equation can be rewritten in the vector form as:

$$\mathbf{a}(\theta) = \mathbf{P}(\mathbf{r}, \boldsymbol{\varphi}) \cdot \mathbf{d}(\theta), \tag{15}$$

where $\mathbf{r} = (r_1, r_2, \cdots, r_N)^T$ and $\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \cdots, \varphi_N)^T$ are the vectors of polar coordinates of the sensors. Elements of $N \times \infty$ matrix $\mathbf{P}(\mathbf{r}, \boldsymbol{\varphi})$ and the $\infty \times 1$ vector $\mathbf{d}(\theta)$ are given by:

$$P_{nl}(\mathbf{r}, \boldsymbol{\varphi}) = (j)^l J_l(kr_n) e^{-jl\varphi_n}, \tag{16}$$

$$d_l(\theta) = e^{jl\theta}, \tag{17}$$

for $n = 1, \cdots, N$ and $l = \cdots, -2, -1, 0, 1, 2, \cdots$.

In this decomposition, $\mathbf{P}(\mathbf{r}, \boldsymbol{\varphi})$ is only a function of the positions of sensors and $\mathbf{d}(\theta)$ is only a function of the propagation direction. The main problem is that $\mathbf{P}$ and $\mathbf{d}$ are of infinite dimensions. However, according to (14), $a_n(\theta) = \lim_{L\to\infty} \sum_{l=-L}^{L} (j)^l J_l(kr_n) e^{-jl\varphi_n} e^{jl\theta}$, hence:

$$\forall \epsilon > 0,\ \exists n_\epsilon \in \mathbb{N}\ :\quad \forall 1 \le n \le N,\ 1 \le m \le M,$$
$$|a_n(\theta_m) - \sum_{l=-n_\epsilon}^{n_\epsilon} (j)^l J_l(kr_n) e^{-jl\varphi_n} e^{jl\theta_m}| < \epsilon.$$

This means that we can truncate $\mathbf{P}(\mathbf{r}, \boldsymbol{\varphi})$ and $\mathbf{d}(\theta)$ at $|l| = n_\epsilon$ to achieve finite dimension $\hat{\mathbf{P}}(\mathbf{r}, \boldsymbol{\varphi})$ and $\hat{\mathbf{d}}(\theta)$, such that $\mathbf{a}(\theta)$ in (15) can be approximated by $\hat{\mathbf{P}}(\mathbf{r}, \boldsymbol{\varphi})\hat{\mathbf{d}}(\theta)$ with elemental error less than ϵ. Therefore, we have the following approximation for the dictionary:

$$\mathbf{A} \simeq \hat{\mathbf{P}}(\mathbf{r}, \boldsymbol{\varphi}) \cdot \hat{\mathbf{D}}(\theta), \tag{18}$$

where:

$$\hat{\mathbf{D}}(\theta) = \left[\hat{\mathbf{d}}(\theta_1) \middle| \hat{\mathbf{d}}(\theta_2) \middle| \cdots \middle| \hat{\mathbf{d}}(\theta_M)\right].$$

We call $\hat{\mathbf{P}}$ the *approximated location matrix*.

Suppose now that we have two distinct sets of sensors with polar coordinate sets $(\mathbf{r}_1, \boldsymbol{\varphi}_1)$ and $(\mathbf{r}_2, \boldsymbol{\varphi}_2)$. The number of sensors in these two sets are assumed to be equal and large enough to ensure the validity of the approximation in (18). Now, by setting:

$$\mathbf{B} \triangleq \hat{\mathbf{P}}_2(\mathbf{r}_2, \boldsymbol{\varphi}_2) \cdot \underbrace{(\hat{\mathbf{P}}_1^H(\mathbf{r}_1, \boldsymbol{\varphi}_1)\hat{\mathbf{P}}_1(\mathbf{r}_1, \boldsymbol{\varphi}_1))^{-1}\hat{\mathbf{P}}_1^H(\mathbf{r}_1, \boldsymbol{\varphi}_1)}_{\text{pseudo-inverse of } \hat{\mathbf{P}}_1} \tag{19}$$

where $\hat{\mathbf{P}}_1$ and $\hat{\mathbf{P}}_2$ are the approximated location matrices of the two distinct sets of sensors, we will have:

$$\hat{\mathbf{P}}_2(\mathbf{r}_2, \boldsymbol{\varphi}_2) = \mathbf{B} \cdot \hat{\mathbf{P}}_1(\mathbf{r}_1, \boldsymbol{\varphi}_1).$$

Hence, if $\mathbf{A}_1$ and $\mathbf{A}_2$ are the corresponding dictionaries of these two sets of sensors, i.e. $\mathbf{A}_1 = \hat{\mathbf{P}}_1(\mathbf{r}_1, \boldsymbol{\varphi}_1) \cdot \hat{\mathbf{D}}(\theta)$ and $\mathbf{A}_2 = \hat{\mathbf{P}}_2(\mathbf{r}_2, \boldsymbol{\varphi}_2) \cdot \hat{\mathbf{D}}(\theta)$, then:

$$\mathbf{A}_2 = \mathbf{B} \cdot \mathbf{A}_1. \tag{20}$$

Remark 1. $\hat{\mathbf{P}}$ is an $N \times (2n_\epsilon + 1)$ matrix. In (19), we need $\hat{\mathbf{P}}$ to have a wide pseudo-inverse (i.e., with no less rows than columns) which requires $\hat{\mathbf{P}}$ to be tall, i.e. $N > 2n_\epsilon$. The smallest n_ϵ itself can as well be upper bounded where the sensors are enclosed in a disk of radius R:

$$n_\epsilon = \max\left(\left\lceil \frac{e^2}{2} kR \right\rceil,\ \left\lceil kR + \ln\frac{1}{\epsilon} \right\rceil\right), \tag{21}$$

which is independent of N and M. Hence, the number of sensors must be greater than $2n_\epsilon$, which only depends on the required precision. We provide a proof for (21) in Appendix B.

Since $\mathbf{s}$ only depends on the impinging signal (and not on the locations of sensors), it does not change with sensors transformations. Hence, from (20) we have:

$$\mathbf{x}_2 = \mathbf{B} \cdot \mathbf{x}_1$$

In summary, we showed that for any two distinct sets of sensors, there exists a matrix that converts their dictionaries and data vectors to each other, provided that the number of sensors is large enough (to ensure the validity of the approximation (18)). Therefore, if an invariant algorithm is utilized for solving (13), the final solution will be approximately independent of the positions of the sensors. Note that the existence of the matrices $\mathbf{P}$, $\mathbf{D}$, and $\mathbf{B}$ is only implicit, and they are not explicitly used.

V. Experimental Results

In this section, we perform four experiments. The first two experiments are to compare the performance of specific invariant and non-invariant algorithms when the dictionary tends to be ill-conditioned. The third experiment is on DOA estimation and compares the robustness of the estimations against the locations of sensors where specific invariant and non-invariant algorithms are used. Finally, another experiment supports the non-invariancy of OMP and SPGL1 algorithms.

Experiment 1. *Robustness of sparse recovery algorithms against the correlation of the columns of the dictionary*

In this experiment, we compare the robustness of five sparse recovery algorithms against the correlation of the columns of the dictionary (in this case, a large mutual coherence signifies ill-conditioning). These algorithms are SL0 (invariant), interior-point BP (invariant), IOMP (invariant), SPGL1 (non-invariant) and OMP (non-invariant)[3]. To construct a 60×100 dictionary with a controlled correlation between its columns, we first randomly created the elements of its odd columns (columns 1, 3, ..., 59) by a uniform distribution over $(-1, 1)$. Then, we created each even column by rotating the previous odd column by an angle θ. Consequently, in the constructed dictionary, the correlation between the $(2k-1)$'th and $(2k)$'th columns, for all $k = 1, 2, \ldots, 50$, is equal to $\cos\theta$.

Next, we created a sparse 100×1 vector $\mathbf{s}$ with 15 non-zero entries, whose positions and values are selected randomly. Then, $\mathbf{x} = \mathbf{A}\mathbf{s}$ was calculated and $(\mathbf{A}, \mathbf{x})$ was given to the above mentioned sparse recovery algorithms to give an estimation of $\mathbf{s}$ denoted by $\hat{\mathbf{s}}$. To measure the accuracy of the estimation, we used Normalized Euclidean Distance (NED) between $\mathbf{s}$ and $\hat{\mathbf{s}}$, which is defined as

$$\text{NED} = \frac{\|\hat{\mathbf{s}} - \mathbf{s}\|_2}{(\|\hat{\mathbf{s}}\|_2 \cdot \|\mathbf{s}\|_2)^{\frac{1}{2}}}.$$

The parameters of the algorithms used for the simulation are $\sigma_{\min} = 0.001$ for SL0, `MaxTol` $= 0.01$ for BP (in the options of MATLAB's '`linprog`' function), $\sigma = 0$ for SPGL1, and MaxIteration$= \infty$ for OMP and IOMP; all other parameters of all algorithms are their default values. The values of θ were changed from 5 to 90 degrees, and for each θ, the simulation was run 200 times with different randomly generated dictionaries ($\mathbf{A}$) and sparse vectors ($\mathbf{s}$).

Figure 2(a) shows the averaged NEDs over these 200 runs versus θ. Moreover, in Fig. 2(b) we have depicted the averaged number of iterations over these 200 runs after normalizing it with respect to the minimum number of iterations. In other words, Fig. 2(b) shows how the relative cost of calculations varies where the problem tends to be ill-conditioned, *i.e.* as θ becomes smaller. As it is seen in the figure, invariant algorithms (SL0, BP, and IOMP) are more robust against the correlation of the columns of the dictionary. Moreover, it can be seen that IOMP performs essentially more accurately and is more robust than its non-invariant counterpart, OMP. Note that in the number of iterations of BP we also consider the number of iterations of its primal-dual interior-point LP solver.

[3]The implementations used for these algorithms are as follows: 1) The MATLAB code of SL0 is taken from 'http://ee.sharif.edu/~SLzero/', 2) Interior-point BP has been implemented using '`linprog`' function in MATLAB's optimization toolbox by setting off its '`simplex`' option and setting on its '`largescale`' option, 3) For SPGL1, we have taken its code from 'http://www.cs.ubc.ca/labs/scl/spgl1/', and 4) For OMP and IOMP, we have used the OMP implementation in 'SparseLab' toolbox available at 'http://sparselab.stanford.edu/'.

Experiment 2. *Robustness of sparse recovery algorithms against the correlation of the rows of the dictionary*

We repeated the previous experiment for the case where the rows of the dictionary tend to be highly correlated (in this case, ill-conditioning is signified by condition number). Here, a 60×100 dictionary $\mathbf{A}$ is created similar to the previous experiment, where the correlation between its $(2k-1)$'st and $(2k)$'th rows, as opposed to columns in the previous experiment, is equal to $\cos\theta$ for all $k = 1, 2, \ldots, 30$. The rest of the experiment is the same as the previous one.

Figure 3 shows the averaged NEDs and the normalized average number of iterations versus θ through 200 runs of the simulation. As it can be seen, the invariant algorithms are more robust against dictionary row correlations as well.

It is also seen in Figs. 2 and 3 that invariant algorithms show more robustness against correlation of rows than correlation of columns. A careful reader would probably expect such an observation from the discussions of Section II and Appendix A: In fact, as stated there, where the rows are correlated, there exists always a matrix $\mathbf{B}$ that converts the dictionary to a new dictionary with completely uncorrelated rows. However, where the columns are correlated, by multiplying a matrix $\mathbf{B}$ we can only reduce column correlation (i.e., mutual coherence) to some extent (not eliminating it as in the case of row correlation). However, such a $\mathbf{B}$ is not explicitly used anywhere, and the above experiment verifies that in the case of high column correlation case, too, invariant algorithms are more robust than non-invariant ones.

Experiment 3. *Robustness of DOA estimators against the location of sensors*

In this experiment, we study the effect of planar distribution of sensors on the quality of DOA estimation. The sensors are posed on two crossing straight lines (like a $\times$) with angle θ, where the distance between two adjacent sensors in a line is 0.4 meters. We assumed that the wave number is $k = 1$, and that the total number of sensors posed on two crossing lines is $N = 139$. With this choice of parameters, if n_ϵ is set to 69, maximum absolute value between elements of the matrices in the two sides of (18) would be approximately 10^{-4} yielding validity of approximation in (18). We refer to θ as a measure of the conditioning of the constellation: sensing is well-conditioned for θ's near 90° and tends to be ill-conditioned where θ becomes very small. We would like to estimate the directions of 4 impinging signals with different angles from $M = 180$ possible propagation directions (*i.e.* a 1 degree resolution; see (13)). Since we are dealing with complex numbers, two complex-valued sparse decomposition algorithms are used to estimate DOA's: SL0 (invariant), and SPGL1 (non-invariant). For the parameters of the algorithms, we used $\sigma_{\min} = 0.01$ and σ-decreasing factor $= 0.8$ for SL0, and $\sigma = 0$ for SPGL1. The value of θ was changed from 5 to

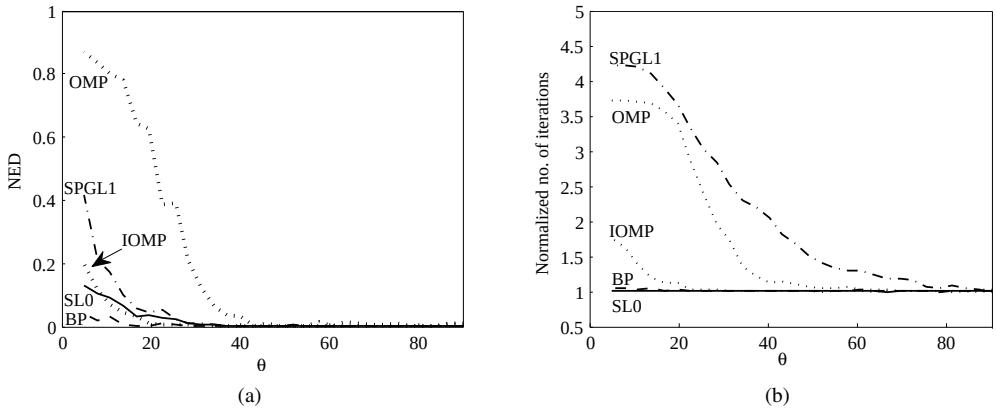


Fig. 2. Effect of correlation of columns on the quality of estimation in invariant (SL0, BP, and IOMP) and non-invariant (OMP and SPGL1) sparse recovery algorithms for the case of $M = 100$ and $N = 60$. The correlation of the $(2k-1)$'st and $(2k)$'th column is equal to $\cos\theta$ for all $k = 1, 2, \ldots, 50$. (a) The averaged NEDs of the five algorithms over 200 runs versus θ in degrees. (b) The average number of iterations over 200 runs versus θ, normalized with respect to the number of iterations done in $\theta = 90^\circ$.

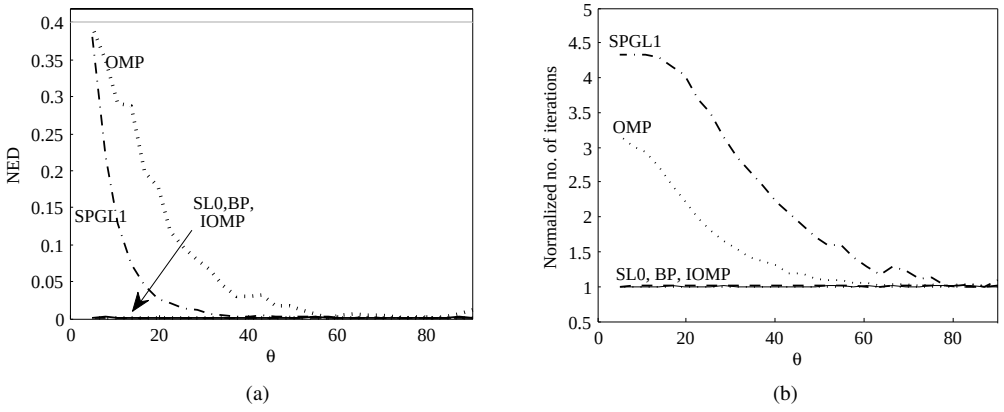


Fig. 3. Effect of correlation of rows on the quality of estimation in invariant (SL0, BP, and IOMP) and non-invariant (OMP and SPGL1) sparse recovery algorithms for the case of $M = 100$ and $N = 60$. The correlation of the $(2k-1)$'st and $(2k)$'th row is equal to $\cos\theta$ for all $k = 1, 2, \ldots, 30$. (a) The averaged NEDs of the five algorithms over 200 runs versus θ in degrees. (b) The average number of iterations over 200 runs versus θ, normalized with respect to the number of iterations done in $\theta = 90^\circ$.

80 degrees and the experiment was repeated 100 times for each θ. Figure 4 shows the Percentage Of Success (POS) versus θ, where by a success we mean that maximum elemental difference between original sparse vector and the estimated vector is less than 0.1. Note that the performance of the non-invariant algorithm improves as the correlation decreases (i.e., larger θs). However, the invariant algorithm (SL0) has a consistent performance, independent of the correlation (θ), showing more robustness against the location of the sensors, as expected.

Experiment 4. *Non-invariancy of OMP and SPGL1*

We provide self-contained proofs of non-invariancy for some algorithms in Appendix B. In this experiment, we provide only empirical support for the non-invariancy of OMP, SPGL1, IHT and HTP algorithms. We generate a 60×100 random matrix $\mathbf{A}$ with elements uniformly and independently distributed in $(-1, 1)$, and use the orthogonalization method mentioned in Appendix A to obtain orthonormal rows. Then, for a given $0 < \sigma < 1$, we create a matrix $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$, where $\mathbf{U}$ and $\mathbf{V}$ are 60×60 random unitary matrices and $\mathbf{\Sigma}$ is the diagonal matrix with half of diagonal elements being 1 and the other half equal to σ. Hence, the condition number of $\mathbf{B}$ would be $\frac{1}{\sigma}$, and $\mathbf{BA}$ tends to be ill-conditioned when σ decreases. For a random 15-sparse vector $\mathbf{s}$ (similar to Experiment 1), we compute $\mathbf{x} = \mathbf{As}$ and calculate $\mathcal{F}(\mathbf{BA}, \mathbf{Bx})$ using the five algorithms used in Experiment 1. The experiment is repeated 200 times for each σ and the resulted average NEDs are plotted in Fig. 5. It can be seen that linear transformation have essentially no effect on performance of invariant algorithms (reflecting the robustness of these algorithms against high condition number), whereas the error of the estimation of OMP, SPGL1, IHT and HTP increases with condition number of $\mathbf{B}$, showing their non-invariancy.

VI. Conclusions

In this article, we introduced the concept of invariancy of sparse recovery algorithms. The significance of invariant algorithms is that they are more robust where the sensing (the dictionary) is ill-conditioned, because '*implicitly*' there exists an equivalent well-conditioned problem, which is being solved. In other words, being more robust against the sensing

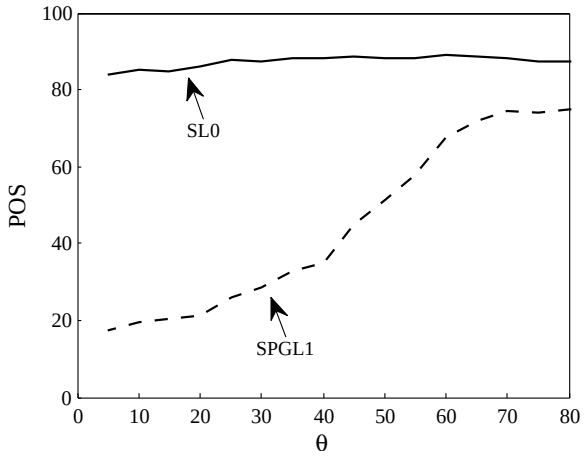


Fig. 4. Performance of invariant and non-invariant algorithms in the cases of well-conditioned and ill-conditioned constellation of sensors in DOA estimation problem in the case of M=180, $N = 139$, $k = 1$, and 4 signals in different DOAs composing the impinging signal. Here, the percentage Of Success (i.e., percent of times that the estimation of algorithm has an elemental error less than an 0.1) of two algorithms are plotted versus the angle between crossing lines (in degrees, *i.e.* $5^\circ \leq \theta \leq 80^\circ$) for the sensor constellation of two crossing lines.

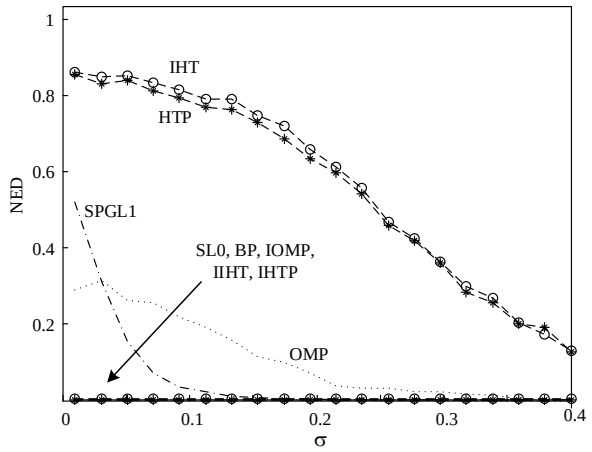


Fig. 5. Performance of invariant and non-invariant sparse recovery algorithms when a 60×100 dictionary, $\mathbf{A}$, undergoes a linear transformation (i.e., $\mathbf{A} \leftarrow \mathbf{BA}$). Here, $\mathbf{B}$ is square matrix half of whose singular values are 1 and the other half equal to σ (hence the condition number of $\mathbf{B}$ is $\frac{1}{\sigma}$). The figure shows the resulted NEDs versus σ.

conditions, the invariant algorithms will converge in the ill-conditioned problems, as quickly as in the case of well-conditioned problems. Besides these advantages, we showed that it is possible to improve existing theoretical performance bounds for some invariant algorithms, because we can use the bounds of well-conditioned problems for ill-conditioned ones. As some examples of invariant algorithms, we proved the invariancy of BP using interior-point LP solver, SL0, FOCUSS, and the non-invariancy of MP, OMP, ROMP, ICoSaMP, IHT, and HTP. We showed that these non-ivariant algorithms can be modified, by adding an initial step of row orthogonalization, to become invariant, and hence enjoy all the advantages of invariant algorithms. As an example application, we studied a group of DOA estimators that are almost invariant to the spatial distribution of sensors on a plane. The main direction for future research will be the analysis of the noisy case. Although the main emphasis of this paper is on the noiseless case, as we briefly discussed in Section II-E, invariant algorithms can be advantageous in noisy settings too. Future research will mainly focus on providing better bounds for stability and recovery conditions as well as improving the signal to noise ratio for better recovery.

APPENDIX A
EXISTENCE OF *Conditioning Enhancing* LINEAR TRANSFORMATIONS

We mentioned three measures for conditioning of a dictionary in Section II, namely condition number, mutual coherence, and RIC. In this appendix, we study improving these measures by multiplying the dictionary by a matrix from the left. First, we show that the condition number of the dictionary can be simply set to unity (which is the least possible value). Then, we propose a method to find an approximation of the optimum matrix $\mathbf{B}$ that minimizes the mutual coherence of $\mathbf{BA}$ (i.e., $M(\mathbf{BA})$). Next, we evaluate the extent to which these methods can improve conditioning of the randomly generated matrices through some numerical simulations. For the case of RIP, since computation of the RIC is itself an NP-hard problem, finding a $\mathbf{B}$ that minimizes RIC of $\mathbf{BA}$ can be very difficult. Here, we use the same matrix, $\mathbf{B}$, as the one enhancing the condition number, and derive a lower bound for how much (5) improves the RIC of random matrices, through a simulation.

A. Condition Number

Consider a wide dictionary, $\mathbf{A}$, and its singular value decomposition (SVD) [58],

$$\mathbf{A} = \mathbf{U\Sigma V},$$

where $\mathbf{U}$ ($N \times N$) and $\mathbf{V}$ ($M \times M$) are unitary matrices and

$$\mathbf{\Sigma} = \left[\begin{array}{ccc|ccc} \sigma_1 & & 0 & 0 & \cdots & 0 \\ & \ddots & & \vdots & \ddots & \vdots \\ 0 & & \sigma_n & 0 & \cdots & 0 \end{array}\right],$$

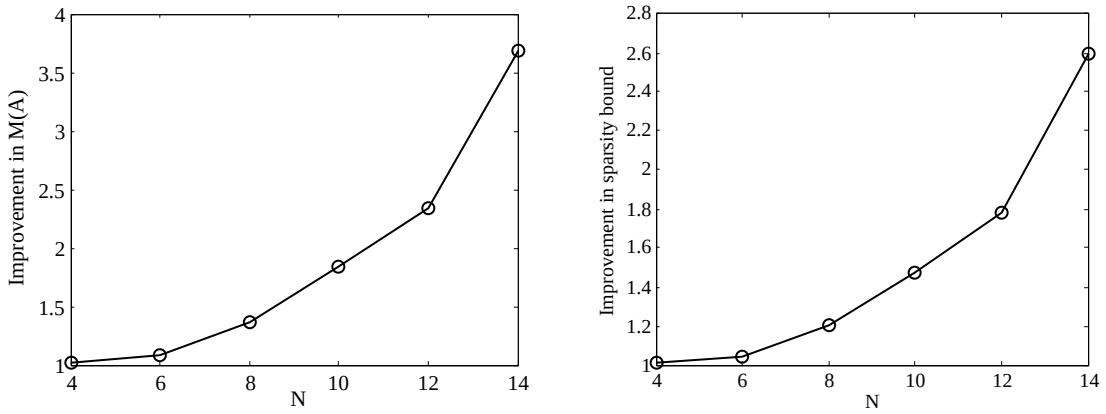
in which $\sigma_1, \cdots, \sigma_n$ are called singular values of $\mathbf{A}$. The condition number of $\mathbf{A}$ is then defined as $\max_{i,j} \frac{\sigma_i}{\sigma_j}$. Matrix $\mathbf{B}$ that minimizes the condition number of $\mathbf{BA}$ can be written as

$$\mathbf{B} = \left[\begin{array}{ccc} {\sigma_1}^{-1} & & 0 \\ & \ddots & \\ 0 & & {\sigma_n}^{-1} \end{array}\right] \mathbf{U^{-1}}.$$

Using such a preconditioner, the condition number of $\mathbf{BA}$ would be 1, which is the least possible value.

B. Mutual Coherence

Here, we present an algorithmic approach to find an approximation of the matrix $\mathbf{B}$ that minimizes the mutual coherence of $\mathbf{BA}$. After the columns of $\mathbf{A}$ are normalized, the mutual



(a) (b)

Fig. 6. Improvement in mutual coherence and the bound guaranteeing equivalence of P_1 and P_0 problems when a suitable linear transformation is applied on dictionaries with 15 columns (i.e., $M = 15$). (a) Improvement in mutual coherence in terms of $\frac{M(\mathbf{A})}{M(\mathbf{BA})}$ versus N (i.e., the number of rows of the dictionary). (b) Improvement in sparsity bound in terms of $\frac{(1+(M(\mathbf{BA})^{-1}))}{(1+(M(\mathbf{A})^{-1}))}$ versus N.

coherence of $\mathbf{A}$ is equal to the maximum absolute value of non-diagonal elements of $\mathbf{A}^T\mathbf{A}$. Moreover, the diagonal elements of $\mathbf{A}^T\mathbf{A}$ are equal to 1. Thus, the problem of finding a nearly optimal matrix $\mathbf{B}$ can be formulated as an optimization problem,

$$\underset{\mathbf{B}}{\operatorname{argmin}} \left(\max_{i \neq j} |\left((\mathbf{BA})^T\mathbf{BA}\right)_{ij}| \right) \quad \text{s.t.} \quad \left((\mathbf{BA})^T\mathbf{BA}\right)_{ii} = 1, \ 1 \leq i \leq m, \tag{22}$$

where $\left((\mathbf{BA})^T\mathbf{BA}\right)_{ij}$ is the element in the i-th row and the j-th column of $(\mathbf{BA})^T\mathbf{BA}$. However, this gives just an approximation, because for the optimum matrix $\mathbf{B}$, it is not necessary to have $\left((\mathbf{BA})^T\mathbf{BA}\right)_{ii} = 1$, for $1 \leq i \leq m$ (i.e., the constraint in (22) may possibly not hold for the optimal matrix $\mathbf{B}$). Let $\mathbf{D} = \mathbf{B}^T\mathbf{B}$, and hence $(\mathbf{BA})^T\mathbf{BA} = \mathbf{A}^T\mathbf{DA}$. In this case, $\mathbf{D}$ is positive semidefinite. On the other hand, if we have such a $\mathbf{D}$ (i.e., positive semidefinite), Cholesky decomposition [59] gives a $\mathbf{B}$ such that $\mathbf{B}^T\mathbf{B} = \mathbf{D}$. This $\mathbf{D}$ can be found using the following convex optimization problem:

$$\underset{\mathbf{D}}{\operatorname{argmin}} \left(\max_{i \neq j} |\left(\mathbf{A}^T\mathbf{DA}\right)_{ij}| \right) \quad \text{s.t.} \quad \left(\mathbf{A}^T\mathbf{DA}\right)_{ii} = 1, \quad 1 \leq i \leq m, \quad \text{and } \mathbf{D} \text{ is positive semidefinite.} \tag{23}$$

The problem in (23) is a convex problem and can be solved using semidefinite programming (SDP) [59], which is handled for example by CVX toolbox for Matlab [60].

C. Numerical Results

In this subsection, we conduct two experiments in order to see how much linear transformations can enhance mutual coherence and RIP of random dictionaries.

Experiment 5. Mutual Coherence

In this experiment, we utilize the above mentioned method to find a lower bound for the extent to which the mutual coherence of a random dictionary can be enhanced, and consequently how much Proposition 1 improves the bound of Theorem 1 for equivalence of P_1 and P_0 problems. We fix the number of columns of the dictionary to 15 (i.e., $M = 15$), vary N from 4 to 14 and construct random matrices of size $N \times M$ whose elements are independently drawn from uniform distribution on $[-1, 1]$. Then, we apply the approach presented in the previous subsection to find a matrix $\mathbf{B}$ that minimizes the mutual coherence of $\mathbf{BA}$ and compute $\frac{M(\mathbf{A})}{M(\mathbf{BA})}$ and $\frac{(1+(M(\mathbf{BA})^{-1}))}{(1+(M(\mathbf{A})^{-1}))}$, which represent the improvement in mutual coherence and the bound guaranteeing the equivalence of P_1 and P_0 problems, respectively. The results are plotted versus N in Fig. 6, where each point is an average over 100 experiments. It can be seen that as the number of rows increases, the improvement enhances.

Experiment 6. RIP

Due to the difficulty of finding a $\mathbf{B}$ that minimizes the RIC of $\mathbf{B}$, here we use the same $\mathbf{B}$ that minimizes condition number. Simulations show that this choice of $\mathbf{B}$ also reduces the RIC to some extent. The experiment is performed on randomly generated dictionaries with elements independently drawn from a uniform distribution on $[-1, 1]$. We consider matrices of size 12×15 and computed the ratio of $\delta_k/\tilde{\delta}_k$ for different k's, where δ_k and $\tilde{\delta}_k$ are the RICs of $\mathbf{A}$ and $\mathbf{BA}$, respectively. The average value of this ratio is plotted versus k in Fig. 7, in which each point is an average over 100 experiments.

APPENDIX B
PROOFS.

In the following, for every variable in $\mathcal{F}(\mathbf{A}, \mathbf{x})$, we denote the corresponding variable in $\mathcal{F}(\mathbf{BA}, \mathbf{Bx})$ by adding a '$\sim$' over it.

Proof of Lemma 1

P_1 can be expressed in standard LP format as

$$\underset{\bar{\mathbf{s}}}{\text{Minimize}} \ \mathbf{c}^T\bar{\mathbf{s}} \qquad \text{s.t.} \qquad \bar{\mathbf{A}}\bar{\mathbf{s}} = \mathbf{x}, \ \bar{\mathbf{s}} \geq 0,$$

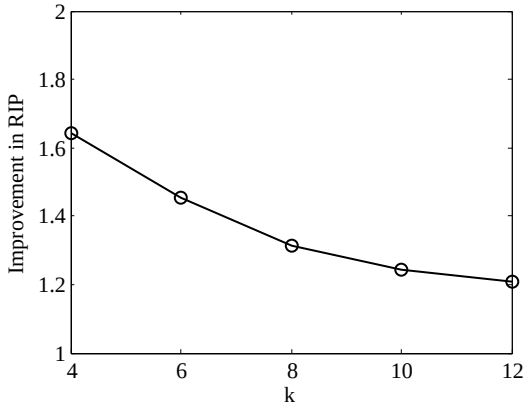


Fig. 7. A lower bound for the improvement of RIC when a linear transformation is applied on random dictionaries of size 12×15. In this figure $\delta_k/\tilde{\delta}_k$ is plotted versus k, where δ_k and $\tilde{\delta}_k$ are the RICs of $\mathbf{A}$ and $\mathbf{BA}$, respectively. Here, $\mathbf{B}$ is chosen to be the matrix that minimizes the condition number of $\mathbf{BA}$.

where $\bar{\mathbf{A}} \triangleq [\mathbf{A} \quad -\mathbf{A}]$ and $\mathbf{c} \triangleq [1, \cdots, 1]^T$. Having solved this LP problem, the solution of P_1, $\mathbf{s}$, is obtained from $s_i = \bar{s}_i - \bar{s}_{i+M}$, where s_i and $\bar{s}_i$ are the i-th elements of $\mathbf{s}$ and $\bar{\mathbf{s}}$, respectively. Now, if the LP solver gives the same solutions for the input pairs $(\bar{\mathbf{A}}, \mathbf{x})$ and $(\mathbf{B}\bar{\mathbf{A}}, \mathbf{Bx})$ the algorithm would be invariant.

Due to its convexity, there are several algorithms for finding the solution of LP. One such algorithm is the primal-dual interior-point method. In the following, we show that this algorithm is itself affine invariant. As such, not only the output of the LP subroutine would be the same for (1) and (3), but also the internal solution paths in the LP subroutine would be identical for the two problems. Therefore, the computational complexity of LP (and hence BP) is not affected by linear transformations of the input. This does not hold, for example, when the simplex algorithm is used for solving LP, in which case the time complexity of the transformed problem may grow exponentially compared to the untransformed problem.

Using the notation of [50, Chapters 13 and 14], an interior-point LP solver aims to solve:

$$\underset{\mathbf{x}}{\text{Minimize}} \qquad \mathbf{c^T x} \qquad \text{s.t.} \qquad \mathbf{Ax} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}.$$

The dual problem can be written as:

$$\underset{\mathbf{x}}{\text{Maximize}} \qquad \mathbf{b^T \lambda} \qquad \text{s.t.} \qquad \mathbf{A^T \lambda} + \mathbf{s} = \mathbf{c},\ \mathbf{s} \geq \mathbf{0}.$$

Thus, $\mathbf{s}$ and $\boldsymbol{\lambda}$ are the variables of the dual problem. In the following, we will show that for the starting point and for all iterations, the equality

$$(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{s}}) = (\mathbf{x}, \mathbf{B}^{-T}\boldsymbol{\lambda}, \mathbf{s}) \tag{24}$$

holds.

a) Checking the starting point: Denote by $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \mathbf{s}_0)$ the starting point of the algorithm. Let:

$$\mathbf{x}' \triangleq \mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{b}, \qquad \boldsymbol{\lambda}' \triangleq (\mathbf{AA}^T)^{-1}\mathbf{Ac},$$
$$\mathbf{s}' \triangleq \mathbf{c} - \mathbf{A}(\mathbf{AA}^T)^{-1}\mathbf{Ac}.$$

It is easy to verify that $\tilde{\mathbf{x}}' = \mathbf{x}'$, $\tilde{\boldsymbol{\lambda}}' = \mathbf{B}^{-T}\boldsymbol{\lambda}'$, and $\tilde{\mathbf{s}}' = \mathbf{s}'$. As an example, we verify the first equality:

$$\begin{aligned}\tilde{\mathbf{x}}' &= (\mathbf{BA})^T(\mathbf{BAA}^T\mathbf{B}^T)^{-1}\mathbf{Bb} \\ &= \mathbf{A}^T\mathbf{B}^T\mathbf{B}^{-T}(\mathbf{AA}^T)^{-1}\mathbf{B}^{-1}\mathbf{Bb} \\ &= \mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{b} = \mathbf{x}'.\end{aligned}$$

The starting point is then computed from these variables:

$$\mathbf{x}_0 = \mathbf{x}' + \hat{\delta}_x\mathbf{e}, \qquad \boldsymbol{\lambda}_0 = \boldsymbol{\lambda}', \qquad \mathbf{s}_0 = \mathbf{s}' + \hat{\delta}_s\mathbf{e},$$

where $\mathbf{e} = [1, \cdots, 1]^T$ and $\hat{\delta}_x$ and $\hat{\delta}_s$ are scalars given by

$$\hat{\delta}_x = \frac{\mathbf{x}''^T\mathbf{s}''}{2\mathbf{e}^T\mathbf{s}''}, \qquad \hat{\delta}_s = \frac{\mathbf{x}''^T\mathbf{s}''}{2\mathbf{e}^T\mathbf{x}''},$$

in which $\mathbf{x}''$ and $\mathbf{s}''$ are

$$\mathbf{x}'' = \mathbf{x}' + (\max(0, -\frac{3}{2}\min_i \mathbf{x}'_i))\mathbf{e},$$
$$\mathbf{s}'' = \mathbf{s}' + (\max(0, -\frac{3}{2}\min_i \mathbf{s}'_i))\mathbf{e}.$$

Since $\tilde{\mathbf{x}}' = \mathbf{x}'$ and $\tilde{\mathbf{s}}' = \mathbf{s}'$, it directly leads to $\tilde{\mathbf{x}}'' = \mathbf{x}''$ and $\tilde{\mathbf{s}}'' = \mathbf{s}''$, and hence $\tilde{\mathbf{x}}_0 = \mathbf{x}_0$ and $\tilde{\mathbf{s}}_0 = \mathbf{s}_0$.

b) Checking (24) in the iterations: In each iteration of the interior-point LP algorithm, the variables of primial and dual problems are updated according to [50]

$$(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) \leftarrow (\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) + \alpha(\Delta\mathbf{x}, \Delta\boldsymbol{\lambda}, \Delta\mathbf{s}),$$

where α is a fixed coefficient and $(\Delta\mathbf{x}, \Delta\boldsymbol{\lambda}, \Delta\mathbf{s})$ is the solution of the following linear system:

$$\begin{bmatrix} 0 & \mathbf{A}^T & I \\ \mathbf{A} & 0 & 0 \\ \mathbf{S} & 0 & \mathbf{X} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \\ \Delta\mathbf{s} \end{bmatrix} = \begin{bmatrix} -\mathbf{A}^T\boldsymbol{\lambda} - \mathbf{s} + \mathbf{c} \\ -\mathbf{Ax} + \mathbf{b} \\ -\mathbf{XSe} \end{bmatrix}, \tag{25}$$

in which $\mathbf{S} \triangleq \text{diag}(\mathbf{s})$ and $\mathbf{X} \triangleq \text{diag}(\mathbf{x})$. By induction, suppose that (24) holds at the k-th iteration. In order to show that (24) still holds at the $(k+1)$-th iteration, it is enough to prove $(\Delta\tilde{\mathbf{x}}, \Delta\tilde{\boldsymbol{\lambda}}, \Delta\tilde{\mathbf{s}}) = (\Delta\mathbf{x}, \mathbf{B}^{-T}\Delta\boldsymbol{\lambda}, \Delta\mathbf{s})$. Suppose $(\Delta\mathbf{x}, \Delta\boldsymbol{\lambda}, \Delta\mathbf{s})$ is the solution of (25), then:

$$\begin{bmatrix} 0 & \mathbf{A}^T\mathbf{B}^T & I \\ \mathbf{BA} & 0 & 0 \\ \mathbf{S} & 0 & \mathbf{X} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \mathbf{B}^{-T}\Delta\boldsymbol{\lambda} \\ \Delta\mathbf{s} \end{bmatrix} = \begin{bmatrix} -\mathbf{A}^T\boldsymbol{\lambda} - \mathbf{s} + \mathbf{c} \\ \mathbf{B}(-\mathbf{Ax} + \mathbf{b}) \\ -\mathbf{XSe} \end{bmatrix}.$$

Thus $(\Delta\tilde{\mathbf{x}}, \Delta\tilde{\boldsymbol{\lambda}}, \Delta\tilde{\mathbf{s}}) = (\Delta\mathbf{x}, \mathbf{B}^{-T}\Delta\boldsymbol{\lambda}, \Delta\mathbf{s})$ and the invariancy of primal-dual interior-point algorithm (and BP) results. ■

Proof of Lemma 2

SPGL1 is not invariant because it needs to solve a LASSO problem [61] at each iteration, that is:

$$\underset{\mathbf{s}}{\text{Minimize}}\ \|\mathbf{As} - \mathbf{x}\|_2 \text{ subject to } \|\mathbf{s}\|_1 \leq \tau,$$

where τ is a constant. The objective function of the LASSO problem differs when $\mathbf{A}$ and $\mathbf{x}$ are substituted by $\mathbf{BA}$ and $\mathbf{Bx}$, respectively, and thus, the solutions differ in general. This means that SPGL1 is non-invariant. ■

We have provided simple computer tests in Section V which support the non-invariancy of SPGL1.

Proof of Lemma 3

a) Checking the starting point:

$$\begin{aligned}\mathbf{s}_0 &= \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{x}\\ \tilde{\mathbf{s}}_0 &= (\mathbf{B}\mathbf{A})^T(\mathbf{B}\mathbf{A}\mathbf{A}^T\mathbf{B}^T)^{-1}\mathbf{B}\mathbf{x}\\ &= \mathbf{A}^T\mathbf{B}^T\mathbf{B}^{-T}(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{B}^{-1}\mathbf{B}\mathbf{x}\\ &= \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{x} = \mathbf{s}_0. \end{aligned} \tag{26}$$

b) Checking the iterations: each iteration consists of two steps:

i) Gradient ascent step: in this step $\mathbf{s}$ is updated according to

$$\mathbf{s} \leftarrow \mathbf{s} + (\mu\sigma^2)\nabla F_\sigma(\mathbf{s}), \tag{27}$$

Since neither $\mathbf{A}$ nor $\mathbf{x}$ affects (27), $\tilde{\mathbf{s}} = \mathbf{s}$ holds after this step.

ii) Projection step: Assume that $\tilde{\mathbf{s}}_{k-1} = \mathbf{s}_{k-1}$ (the inductive hypothesis). Then we have:

$$\begin{aligned}\mathbf{s}_k &= \mathbf{s}_{k-1} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{s}_{k-1} - \mathbf{x})\\ \tilde{\mathbf{s}}_k &= \tilde{\mathbf{s}}_{k-1} - (\mathbf{B}\mathbf{A})^T(\mathbf{B}\mathbf{A}\mathbf{A}^T\mathbf{B}^T)^{-1}(\mathbf{B}\mathbf{A}\tilde{\mathbf{s}}_{k-1} - \mathbf{B}\mathbf{x})\\ &= \mathbf{s}_{k-1} - \mathbf{A}^T\mathbf{B}^T\mathbf{B}^{-T}(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{B}^{-1}\mathbf{B}(\mathbf{A}\mathbf{s}_{k-1} - \mathbf{x})\\ &= \mathbf{s}_{k-1} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{s}_{k-1} - \mathbf{x}) = \mathbf{s}_k. \end{aligned} \tag{28}$$

■

Proof of Lemma 4

We start by MP and then discuss the others. MP is a greedy algorithm that expands the support set of $\mathbf{s}$ in each iteration such that the correlation of the new atom with the residual of the previous iteration is maximized. Mathematically, the first step is to find the column of $\mathbf{A}$ (atom) that has the highest correlation with $\mathbf{x}$. Then this atom is added to the set of selected variables and an approximation of $\mathbf{x}$ based on this selected variable is calculated. In the next iteration, the atom that has the highest correlation with the residual (difference between actual and approximated $\mathbf{x}$) is added to the set of selected atoms, a new approximation and residual based on the new set of selected atoms is calculated, and the whole process is continued until a convergence criterion is met.

MP is non-invariant since its solutions can differ from the very first iteration for linearly transformed problems. When both the dictionary and observed vector undergo a linear transformation, the correlations between the atoms and the transformed observation vector can be quite different. Denoting the i'th column of $\mathbf{A}$ by $\mathbf{a}_i$ and inner products by $\langle\cdot,\cdot\rangle$, in general, we have $\langle\mathbf{a}_i, \mathbf{x}\rangle \neq \langle\mathbf{B}\mathbf{a}_i, \mathbf{B}\mathbf{x}\rangle$ for an invertible matrix $\mathbf{B}$.

As a small toy example to show this, consider $\mathbf{x} = [1,2,3]^T$, and:

$$\mathbf{A} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}; \mathbf{B} = \begin{bmatrix}1/\sqrt{3} & 0 & 0\\ 1/\sqrt{3} & 1 & 0\\ 1/\sqrt{3} & 0 & 1\end{bmatrix},$$

where $\mathbf{B}$ is chosen such that the columns of $\mathbf{B}\mathbf{A}$ are also normalized. Then the inner products of columns of $\mathbf{A}$ and $\mathbf{x}$ are $[1,2,3]$, and the inner products of columns of $\mathbf{B}\mathbf{A}$ and $\mathbf{B}\mathbf{x}$ (i.e., transformed problem) are $[3.46, 2, 3]$. Therefore, in the first iteration of MP, for the first problem, the third atom is chosen and for the second problem, the first atom is chosen. The same phenomenon in general matrices, causes different selections in the early stages which makes MP non-invariant.

The other three algorithms have similar steps. Despite minor differences, they all have an initial correlation step followed by updating the set of selected atoms and calculating the new approximations and residuals. The correlation step that makes MP non-invariant makes the others non-invariant in exact same way as discussed above. Therefore, OMP, ROMP, and CoSaMP are not invariant.

An empirical support for the non-invariancy of OMP was provided by a simulated counter-example in Section V. ■

Proof of Lemma 5

We start by proving the invariancy of ICoSaMP and then discuss the other algorithms. We need to show that for any matrices $\mathbf{B}_1$ and $\mathbf{B}_2$, ICoSaMP finds the same sparsest solution for the two following problems:

$$\mathbf{B}_1\mathbf{A}\mathbf{s}_1 = \mathbf{B}_1\mathbf{x},$$

$$\mathbf{B}_2\mathbf{A}\mathbf{s}_2 = \mathbf{B}_2\mathbf{x};$$

i.e., $\mathbf{s}_1 = \mathbf{s}_2$.

For the two problems, ICoSaMP first finds matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ such that the rows of $\mathbf{A}_1 \triangleq \mathbf{C}_1\mathbf{B}_1\mathbf{A}$ and $\mathbf{A}_2 \triangleq \mathbf{C}_2\mathbf{B}_2\mathbf{A}$ are orthonormal, and then utilizes CoSaMP [38] to find the sparsest solutions $\mathbf{s}_1$ and $\mathbf{s}_2$ of

$$\mathbf{A}_1\mathbf{s}_1 = \mathbf{x}_1,$$

$$\mathbf{U}\mathbf{A}_1\mathbf{s}_2 = \mathbf{U}\mathbf{x}_1;$$

where $\mathbf{x}_1 \triangleq \mathbf{C}_1\mathbf{B}_1\mathbf{x}$, $\mathbf{x}_2 \triangleq \mathbf{C}_2\mathbf{B}_2\mathbf{x}$, and $\mathbf{U} \triangleq \mathbf{C}_2\mathbf{B}_2(\mathbf{C}_1\mathbf{B}_1)^{-1}$ is a unitary (i.e., orthonormal and square) matrix, because according to the choice of $\mathbf{C}_1$ and $\mathbf{C}_2$, both $\mathbf{A}_1$ and $\mathbf{U}\mathbf{A}_1$ have orthonormal rows. Denote the variables of the two problems with indices 1 and 2; with T_1 and T_2 being the support sets of $\mathbf{s}_1$ and $\mathbf{s}_2$ and $\mathbf{v}_1$ and $\mathbf{v}_2$ being the residuals in each iteration.

a) Starting Point: The starting point is $\mathbf{s}_1 = \mathbf{s}_2 = 0$, $\mathbf{v}_1 = \mathbf{x}_1$, and $\mathbf{v}_2 = \mathbf{x}_2 = \mathbf{U}\mathbf{x}_1$.

b) Iterations: By induction, assume that $\mathbf{s}_1 = \mathbf{s}_2$, $T_1 = T_2$, and $\mathbf{v}_2 = \mathbf{U}\mathbf{v}_1$ at the k-th iteration. The new elements of the support set T at the $(k+1)$-th iteration are chosen as the largest elements of $\mathbf{y}_1 = {\mathbf{A}_1}^H\mathbf{v}_1$ and $\mathbf{y}_2 = {\mathbf{A}_2}^H\mathbf{v}_2 = (\mathbf{U}\mathbf{A}_1)^H\mathbf{U}\mathbf{v}_1 = {\mathbf{A}_1}^H\mathbf{v}_1 = \mathbf{y}_1$, where the third equality is due to orthonormality of $\mathbf{U}$. Hence, $T_1 = T_2$ at the $(k+1)$-th iteration. Denote by $\mathbf{s}|_T$ and $\mathbf{A}|_T$ the matrices obtained by omitting the rows of $\mathbf{s}$ and $\mathbf{A}$ whose indices are not in the set T. Then, the sparse vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ are updated as $\mathbf{s}_1|_{T_1} = (\mathbf{A}_1|_T)^\dagger\mathbf{x}_1 = (\mathbf{A}_2|_T)^\dagger\mathbf{x}_2 = \mathbf{s}_2|_{T_2}$ and $\mathbf{s}_1|_{T_1^c} = \mathbf{s}_2|_{T_2^c} = 0$, where $(\cdot)^\dagger$ denotes pseudo-inverse operation. In the same way, the new residual vectors are $\mathbf{v}_2 = \mathbf{x}_2 - \mathbf{A}_2\mathbf{s}_2 = \mathbf{U}(\mathbf{x}_1 - \mathbf{A}_1\mathbf{s}_1) = \mathbf{U}\mathbf{v}_1$. Thus, $\mathbf{s}_1 = \mathbf{s}_2$ and ICoSaMP is invariant.

IMP, IOMP, and IROMP follow similar steps: in each iteration, adding one or more atoms that have the highest correlation with residual and then, updating the approximation and the residual. In the same way described above, we can show that the orthogonalization causes the original and linearly transformed problems to have the same set of correlations and

consequently, the same set of chosen atoms and approximations. Thus, IMP, IOMP, and IROMP are also invariant. ■

Proof of Lemma 6

a) The starting point: The invariancy of this initial point (which is also the starting point in SL0) was verified in (26).

b) Checking the iterations: The following transformation operates on $\mathbf{s}_k$ in each iteration:

$$\mathbf{s}_k = \mathbf{W}_k(\mathbf{A}\mathbf{W}_k)^\dagger \mathbf{x}$$

where $\mathbf{W}_k \triangleq \text{diag}(\mathbf{s}_{k-1})$, and $(\cdot)^\dagger$ denotes pseudo-inverse operation. Assume that $\tilde{\mathbf{W}}_k = \mathbf{W}_k$ (the inductive hypothesis). Then, we have:

$$\begin{aligned}
\tilde{\mathbf{s}}_k &= \tilde{\mathbf{W}}_k(\mathbf{B}\mathbf{A}\tilde{\mathbf{W}}_k)^\dagger \mathbf{B}\mathbf{x} \\
&= \mathbf{W}_k(\mathbf{B}\mathbf{A}\mathbf{W}_k)^\dagger \mathbf{B}\mathbf{x} \\
&= \mathbf{W}_k(\mathbf{A}\mathbf{W}_k)^T \mathbf{B}^T \mathbf{B}^{-T}(\mathbf{A}\mathbf{W}_k {\mathbf{W}_k}^T \mathbf{A}^T)\mathbf{B}^{-1}\mathbf{B}\mathbf{x} \\
&= \mathbf{W}_k(\mathbf{A}\mathbf{W}_k)^T(\mathbf{A}\mathbf{W}_k {\mathbf{W}_k}^T \mathbf{A}^T)\mathbf{x} \\
&= \mathbf{W}_k(\mathbf{A}\mathbf{W}_k)^\dagger \mathbf{x} = \mathbf{s}_k,
\end{aligned}$$

yielding the invariancy of FOCUSS. ■

Proof of Lemma 7

The exhaustive search algorithm needs to solve exactly $\binom{m}{\text{rank}(\mathbf{A})}$ systems of linear equations and finds the exact solution. Hence, its performance is not affected by preconditioning. ■

Proof of Lemma 8

In IHT with starting point $\mathbf{s}^0 = \tilde{\mathbf{s}}^0 = \mathbf{0}$, we have $\mathbf{s}^1 = H_s(\mathbf{A}^T\mathbf{x})$ and $\tilde{\mathbf{s}}^1 = H_s(\mathbf{A}^T\mathbf{B}^T\mathbf{B}\mathbf{x})$. Similar to the proof of Lemma 4, these two solutions are not equal and thus, IHT is not invariant. This non-invariance is independent of the starting point, because for any starting points $\mathbf{s}^0$, we have $H_s(\mathbf{s}^0 + \mathbf{A}^T(\mathbf{x} - \mathbf{A}\mathbf{s}^0)) \neq H_s(\mathbf{s}^0 + \mathbf{A}^T\mathbf{B}^T(\mathbf{B}\mathbf{x} - \mathbf{B}\mathbf{A}\mathbf{s}^0))$ in general. The same argument can be made for the first step of HTP and conclude that the selected supports will be different for the original and transformed problems. Therefore, HTP is non-invariant too. ■

Proof of Lemma 9

The proof is somewhat similar to the proof of Lemma 5. Define $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{U}$ exactly as defined in the proof of Lemma 5. To prove the inavriancy of IIHT, it suffices to show $\mathbf{A}_1^T(\mathbf{x}_1 - \mathbf{A}_1\mathbf{s}) = \mathbf{A}_1^T\mathbf{U}^T(\mathbf{U}\mathbf{x}_1 - \mathbf{U}\mathbf{A}_1\mathbf{s})$ for any $\mathbf{s}$, $\mathbf{x}$ and unitary $\mathbf{U}$. The equality is obvious because we have $\mathbf{U}^T\mathbf{U} = \mathbf{I}$. Thus, with equal starting points, the updates are exactly the same in each iteration of IIHT, and thus, IIHT is invariant.

For IHTP, the invariancy of the first step (support estimation) follows from the argument in the previous paragraph. It remains to show that the second step is also invariant. For a unitary matrix $\mathbf{U}$, the minimizers of $\|\mathbf{A}_1\mathbf{s} - \mathbf{x}_1\|_2$ and $\|\mathbf{U}\mathbf{A}_1\mathbf{s} - \mathbf{U}\mathbf{x}_1\|_2$ over the same support are the same. Thus, the second step of IHTP is also invariant. This completes the proof. ■

Proof of Bound in Equation (21)

To prove (21), we need to bound the approximation error:

$$\begin{aligned}
\forall r \leq R, \quad & |a_n(\theta_m) - \sum_{l=-n_\epsilon}^{n_\epsilon} (j)^l J_l(kr) e^{-jl\varphi_n} e^{jl\theta_m}| \\
&= \left| \sum_{|l|>n_\epsilon} J_l(kr) e^{jl(\theta_m - \varphi_n + \frac{\pi}{2})} \right| \\
&\leq \sum_{|l|>n_\epsilon} |J_l(kr)| \\
&= \sum_{|l|>n_\epsilon} \left| \sum_{m=0}^{\infty} \frac{(-1)^m}{m!(m+l)!} \left(\frac{kr}{2}\right)^{2m+l} \right| \\
&\leq \sum_{m=0}^{\infty} \frac{1}{m!} \left(\frac{kr}{2}\right)^{2m} \left[\sum_{|l|>n_\epsilon} \frac{1}{(m+l)!} \left(\frac{kr}{2}\right)^{l} \right] \\
&\leq 2 \left(\sum_{m=0}^{\infty} \frac{1}{(m!)^2} \left(\frac{kR}{2}\right)^{2m} \right) \left(\sum_{l>n_\epsilon} \frac{1}{l!} \left(\frac{kR}{2}\right)^{l} \right) \\
&= 2I_0(kR) \sum_{l>n_\epsilon} \frac{1}{l!} \left(\frac{kR}{2}\right)^{l} \\
&\leq 2e^{kR} \sum_{l>n_\epsilon} \frac{1}{l!} \left(\frac{kR}{2}\right)^{l},
\end{aligned} \tag{29}$$

where $I_0(x)$ is the zero order modified bessel function of the first kind and the inequality $I_0(x) \leq e^x$ is proved in [62]. The implication in the third line is due to:

$$J_l(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!(m+l)!} \left(\frac{x}{2}\right)^{2m+l}.$$

It follows from the assumption $n_\epsilon \geq kR$, that:

$$\begin{aligned}
\sum_{l>n_\epsilon} \frac{1}{l!} \left(\frac{kR}{2}\right)^{l} &= \sum_{l\geq 1} \frac{1}{(n_\epsilon + l)!} \left(\frac{kR}{2}\right)^{n_\epsilon + l} \\
&\leq \frac{1}{n_\epsilon!} \left(\frac{kR}{2}\right)^{n_\epsilon} \sum_{l\geq 1} \left(\frac{kR}{2n_\epsilon}\right)^{l} \\
&\leq \frac{1}{n_\epsilon!} \left(\frac{kR}{2}\right)^{n_\epsilon}.
\end{aligned} \tag{30}$$

Finally, using the Stirling approximation, $n! \geq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, we have

$$\begin{aligned}
\forall r \leq R, \quad & |a_n(\theta_m) - \sum_{l=-n_\epsilon}^{n_\epsilon} (j)^l J_l(kr) e^{-jl\varphi_n} e^{jl\theta_m}| \\
&\leq 2e^{kR} \frac{1}{n_\epsilon!} \left(\frac{kR}{2}\right)^{n_\epsilon} \\
&\leq 2e^{kR} \frac{1}{\sqrt{2\pi n_\epsilon}\left(\frac{n_\epsilon}{e}\right)^{n_\epsilon}} \left(\frac{kR}{2}\right)^{n_\epsilon} \\
&\leq 2e^{(n_\epsilon - \ln\frac{1}{\epsilon})} \frac{1}{\sqrt{2\pi n_\epsilon}\left(\frac{n_\epsilon}{e}\right)^{n_\epsilon}} \left(\frac{n_\epsilon}{e^2}\right)^{n_\epsilon} \\
&= \frac{2\epsilon}{\sqrt{2\pi n_\epsilon}} < \epsilon
\end{aligned} \tag{31}$$

■

REFERENCES

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

[2] R. Gribonval and M. Nielsen, "Sparse decompositions in unions of bases," *IEEE Trans. Information Theory*, vol. 49, no. 12, pp. 3320–3325, Dec. 2003.

[3] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization," *in Proceedings of the National Academy of Sciences*, vol. 100, no. 5, pp. 2197–2202, March 2003.

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

[5] S. Foucart and H. Rauhut, "Sparse solutions of underdetermined systems," in *A Mathematical Introduction to Compressive Sensing*, pp. 41–59. Springer, 2013.

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

[7] D. L. Donoho, "Compressed sensing," *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[8] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Processing Mag.*, vol. 24, no. 4, pp. 118–124, July 2007.

[9] R.G. Baraniuk, V. Cevher, M.F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. Information Theory*, vol. 56, no. 4, pp. 1982–2001, April 2010.

[10] R. Gribonval and S. Lesage, "A survey of sparse component analysis for blind source separation: principles, perspectives, and new challenges," in *Proceedings of ESANN'06*, April 2006, pp. 323–330.

[11] P. Bofill and M. Zibulevsky, "Underdetermined blind source separation using sparse representations," *Signal Processing*, vol. 81, pp. 2353–2362, 2001.

[12] P. G. Georgiev, F. J. Theis, and A. Cichocki, "Blind source separation and sparse component analysis for over-complete mixtures," in *Proceedinds of ICASSP'04*, Montreal (Canada), May 2004, pp. 493–496.

[13] Y. Li, A. Cichocki, and S. Amari, "Sparse component analysis for blind source separation with less sensors than sources," in *in Proceedings ICA2003*, 2003, pp. 89–94.

[14] Z. Ma et al., "Sparse principal component analysis and iterative thresholding," *The Annals of Statistics*, vol. 41, no. 2, pp. 772–801, 2013.

[15] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1999.

[16] D. L. Donoho, M. Elad, and V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. Information Theory*, vol. 52, no. 1, pp. 6–18, Jan 2006.

[17] S. Nam, M. E. Davies, M. Elad, and R. Gribonval, "The cosparse analysis model and algorithms," *Applied and Computational Harmonic Analysis*, vol. 34, no. 1, pp. 30–56, 2013.

[18] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*, Academic press, 2010.

[19] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[20] H. Zayyani, M. Babaie-Zadeh, and C. Jutten, "Decoding real-field codes by an iterative expectation-maximization (em) algorithm," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 3169–3172.

[21] M. A. T. Figueiredo and R. D. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Trans. Image Processing*, vol. 12, no. 8, pp. 906–916, 2003.

[22] M. A. T. Figueiredo and R. D. Nowak, "A bound optimization approach to wavelet-based image deconvolution," in *IEEE Internation Conference on Image Processing (ICIP)*, August 2005, pp. II–782–5.

[23] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM review*, vol. 51, no. 1, pp. 34–81, 2009.

[24] M. Elad, "Why simple shrinkage is still relevant for redundant representations?," *IEEE Trans. Image Processing*, vol. 52, no. 12, pp. 5559–5569, 2006.

[25] W. Dong, X. Li, D. Zhang, and G. Shi, "Sparsity-based image denoising via dictionary learning and structural clustering," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 457–464.

[26] 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, March 1997.

[27] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "Fast sparse representation based on smoothed l0 norm," in *Proceedings of 7th International Conference on Independent Component Analysis and Signal Separation (ICA2007), Springer LNCS 4666*, London, UK, September 2007, pp. 389–396.

[28] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "Complex-valued sparse representation based on smoothed ℓ^0 norm," in *Proceedings of* ICASSP2008, Las Vegas, April 2008, pp. 3881–3884.

[29] 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, January 2009.

[30] E. van den Berg and M. P. Friedlander, "Probing the pareto frontier for basis pursuit solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[31] E. van den Berg and M. P. Friedlander, "SPGL1: A solver for large-scale sparse reconstruction," June 2007, http://www.cs.ubc.ca/labs/scl/spgl1.

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

[33] E. Candes, "The restricted isometry property and its implications for compressed sensing," *Elsevier*, vol. 346, pp. 589–592, 2008.

[34] Qun Mo and Song Li, "New bounds on the restricted isometry constant δ2k," *Applied and Computational Harmonic Analysis*, vol. 31, no. 3, pp. 460–468, 2011.

[35] Ryan Joseph Tibshirani, *The solution path of the generalized lasso*, Stanford University, 2011.

[36] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*, Nov. 1993.

[37] D. Needell and R. Vershynin, "Uniform Uncertainty Principle and Signal Recovery via Regularized Orthogonal Matching Pursuit," *Foundations of Computational Mathematics*, vol. 9, no. 3, pp. 317–334, 2009.

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

[39] Thomas Blumensath and Mike E Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 629–654, 2008.

[40] Simon Foucart, "Hard thresholding pursuit: an algorithm for compressive sensing," *SIAM Journal on Numerical Analysis*, vol. 49, no. 6, pp. 2543–2563, 2011.

[41] K. Schnass and P. Vandergheynst, "Dictionary preconditioning for greedy algorithms," *IEEE Trans. Signal Processing*, vol. 56, no. 5, pp. 1994–2002, 2008.

[42] D. Paul, E. Bair, T. Hastie, and R. Tibshirani, "Preconditioning for feature selection and regression in high-dimensional problems," *The Annals of Statistics*, pp. 1595–1618, 2008.

[43] J. C. Huang and N. Jojic, "Variable selection through correlation sifting," in *Research in Computational Molecular Biology*. Springer, 2011, pp. 106–123.

[44] J. Jia and K. Rohe, "Preconditioning to comply with the irrepresentable condition," *arXiv preprint arXiv:1208.5584*, 2012.

[45] M. Babaie-Zadeh and C. Jutten, "On the stable recovery of the sparsest overcomplete representations in presence of noise," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5396–5400, 2010.

[46] J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation," *IEEE Trans. Signal Processing*, vol. 44, no. 12, pp. 3017–3030, Dec. 1996.

[47] R. Fletcher, *Practical methods of optimization*, New York: Wiley, 1981.

[48] Peter Deuflhard, *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, vol. 35, Springer Science & Business Media, 2011.

[49] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 43, no. 1, pp. 129–50, 2001.

[50] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 2006.

[51] Arsalan Sharif-Nassab, Milad Kharratzadeh, Massoud Babaie-Zadeh, and Christian Jutten, "How to use real-valued sparse recovery algorithms for complex-valued sparse recovery?," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE, 2012, pp. 849–853.

[52] S. F. Cotter, *Subset selection algorithms with applications*, Ph.D. dissertation, UC San Diego, La Jolla, CA,, 2001.
[53] H. Krim and M. Viberg, "Two decades of array signal processing research: The parametric approach," *IEEE Signal Processing Mag.*, vol. 13, no. 4, pp. 67–94, July 1996.
[54] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas and Propagation*, vol. 34, no. 3, pp. 276 – 280, March 1986.
[55] R. Roy, A. Paulraj, and T. Kailath, "Estimation of signal parameters via rotational invariance techniques - esprit," in *IEEE Military Communications Conference - Communications-Computers: Teamed for the 90's, MILCOM86,*, oct. 1986, vol. 3, pp. 41.6.1 –41.6.5.
[56] Y. Hua and T.K. Sarkar, "Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 38, no. 5, pp. 814 –824, may 1990.
[57] Milton Abramowitz, Irene A Stegun, et al., "Handbook of mathematical functions," *Applied mathematics series*, vol. 55, no. 62, pp. 39, 1966.
[58] R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge University Press, Cambridge, 1990.
[59] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
[60] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, V. Blondel, S. Boyd, and H. Kimura, Eds., Lecture Notes in Control and Information Sciences, pp. 95–110. Springer-Verlag Limited, 2008, http://stanford.edu/~boyd/graph_dcp.html.
[61] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of Royal Statistical Society Series B*, vol. 58, no. 1, pp. 267–288, 1996.
[62] Andrea Laforgia and Pierpaolo Natalini, "Some inequalities for modified bessel functions," *Journal of Inequalities and Applications*, vol. 2010, no. 1, pp. 253035, 2010.

Milad Kharratzadeh received his B.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran in 2010; and M.Eng. and Ph.D. degrees in electrical and computer engineering from McGill University, Montreal, Canada in 2012 and 2016, respectively. He is currently a postdoctoral research scientist in statistics at Columbia University. His research interests include statistics, machine learning, data science, and computational cognitive science.

Arsalan Sharifnassab received B.Sc. degrees in Electrical Engineering and Mathematics in 2011, and M.Sc. degree in Electrical Engineering in 2013, all from Sharif University of Technology (SUT), Tehran, Iran. Since 2013, he has been a Ph.D. student in the Department of Electrical Engineering at SUT, and he is currently a visiting student in the Laboratory of Information and Decision Systems (LIDS) at MIT. His research interests include network scheduling, distributed computations, and computational complexity.

Massoud Babaie-Zadeh received the B.Sc. degree in electrical engineering from Isfahan University of Technology, Isfahan, Iran in 1994, and the M.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1996, and the Ph.D. degree in Signal Processing from Institute National Polytechnique of Grenoble (INPG), Grenoble, France, in 2002. He received the best Ph.D. thesis award of INPG for his Ph.D. dissertation. Since 2003, he has been a faculty member of the Electrical Engineering Department of Sharif University of Technology, Tehran, IRAN, in which, he is currently a full professor. His main research areas are Blind Source Separation (BSS) and Sparsity-aware Signal Processing.