Incoherent frames design and dictionary learning using a distance barrier[☆]

Denis C. Ilie-Ablachim, Bogdan Dumitrescu

Department of Automatic Control and Computers
University Politehnica of Bucharest, Romania
Emails: denis.ilie_ablachim@upb.ro, bogdan.dumitrescu@upb.ro.

Abstract

We present a unitary approach to the design of incoherent frames and to dictionary learning, by using a single function that promotes incoherence for both problems. This function has a context-dependent quadratic term and a distance barrier term that was never used in this context. We provide simple and efficient algorithms for both problems. Numerical results show that we can obtain large frames whose incoherence is better than of those designed by other methods. Also, in dictionary learning, we can improve both the representation error and the incoherence of the dictionary, compared with the standard approach.

Keywords: mutual coherence, frames, dictionary learning, sparse representations, optimization

1. Introduction

Incoherence is the property of overcomplete bases to have vectors with well separated directions. Frames [1] are matrices $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ that satisfy the relation $A\|\boldsymbol{y}\|^2 \leq \|\boldsymbol{D}^T\boldsymbol{y}\|^2 \leq B\|\boldsymbol{y}\|^2$, $\forall \boldsymbol{y} \in \mathbb{R}^m$, for some positive constants A, B . Incoherent frames have applications in coding and communications, see [2, 3, 4, 5] among many others. In sparse representations [6] and dictionary learning (DL) [7], incoherence allows provable properties of achievable representations and of important algorithms. So, there is continuous inter-

[☆]This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS - UEFISCDI, project number PN-III-P4-PCE-2021-0154, within PNCDI III.

est in designing incoherent frames and dictionaries and, despite very good results in the last decade, progress is still possible.

Problems. We give algorithms for two incoherence problems, both involving a matrix $\boldsymbol{D} \in \mathbb{R}^{m\times n}$, with $m > n$, whose columns are denoted $\boldsymbol{d}_j$, $j = 1 : n$, have unit norm, and are named atoms. The first is to find Grassmannian frames, which are the solution of the problem

$$\begin{array}{ll} \min\limits_{\boldsymbol{D}} & \max\limits_{1\le i<j\le n} |\boldsymbol{d}_i^T \boldsymbol{d}_j| \\ \text{s.t.} & \|\boldsymbol{d}_j\| = 1,\ j = 1 : n \end{array} \tag{1}$$

The objective function is the mutual coherence of the frame. The atoms are as far apart as possible.

The second is that of incoherent DL, formulated as

$$\begin{array}{ll} \min\limits_{\boldsymbol{D},\boldsymbol{X}} & \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 + \gamma f(\boldsymbol{D}) \\ \text{s.t.} & \|\boldsymbol{d}_j\| = 1,\ j = 1 : n \\ & \|\boldsymbol{x}_i\|_0 \le s,\ i = 1 : N \end{array} \tag{2}$$

where $\boldsymbol{Y} \in \mathbb{R}^{m\times N}$ is the data matrix, $\|\boldsymbol{x}_i\|_0$ is the number of nonzeros in the i-th column of $\boldsymbol{x}_i$, s is the sparsity level, and $\gamma > 0$ is a trade-off factor. The first term of the objective of (2) is the overall sparse representation error, and $f(\cdot)$ is a function promoting incoherence, to be given later.

The same function is used to approximate (1) with

$$\begin{array}{ll} \min\limits_{\boldsymbol{D}} & f(\boldsymbol{D}) \\ \text{s.t.} & \|\boldsymbol{d}_j\| = 1,\ j = 1 : n \end{array} \tag{3}$$

Previous work. Grassmannian frames can be designed using analytical approaches [8, 9], which are possible only for a small number of (m, n) pairs. Some of the early algorithms [10, 11, 12] used the Gram matrix $\boldsymbol{D}^T\boldsymbol{D}$ and its spectral properties; they give good results for small dimensions, but are slow and do not scale well; a notable exception is [13], which applies a proximal mapping method to a modified objective that contains both the frame and the Gram matrix. Algorithms based on repelling forces, like those between electrical charges [14, 15], or collision models [16] behave very well for small and sometimes medium values of m and n, but their convergence tends to become very slow for large frames. Approximation with convex problems [17], extended to constrained incoherence problems [18], is also a successful approach. Other methods in the same vein are [19, 20, 21], showing good complexity for large dimensions. Most of the

methods rely more or less on heuristics, either in formulating the problem or in the optimization algorithm; there are few algorithms with guaranteed convergence (to a local minimum, since the problem is not convex) regardless of parameter values, notably [10, 20]; although this is certainly a desirable quality, other algorithms were also successful in practice, especially for large frames where complexity is paramount. In [22, 23], incoherent frames are designed in the context of compressed sensing, with the purpose of improving recovery properties of measurement matrices.

Some works for incoherent DL use (2), with $f(\boldsymbol{D}) = \|\boldsymbol{D}^T\boldsymbol{D} - \boldsymbol{I}\|_F^2$ and gradient [24, 25] or proximal [26] methods. A different approach uses $f(\boldsymbol{D}) = \|\boldsymbol{D}^T\boldsymbol{D} - \boldsymbol{I}\|_\infty$ and proximal algorithms [27] (an alternative algorithm imposes a hard constraint on the mutual coherence). Other papers [28, 29] introduce special decorrelating operations in a standard DL algorithm, with the purpose of keeping coherence under a given value. A more detailed presentation of these issues can be found in [7].

Contribution and contents. In Section 2, we propose a new function for designing Grassmannian frames via (3). This function contains a weighted quadratic term that replaces the infinity norm associated with the max from (1) and a distance barrier; the first term was optimized in [19], but the second was never used in this context. We solve the problem with a gradient algorithm. A key parameter is the target coherence, which can be selected using a bisection approach, for which we give all details. Numerical results presented in Section 3 show that the algorithm can robustly obtain the best known mutual coherences for large frames of various dimensions. We then use the same function, in Section 4, for the DL problem (2), which is solved with an algorithm in AK-SVD [30] style. The experimental results in Section 5 show that we can obtain dictionaries that are both better fitted to the data and more incoherent than those obtained by AK-SVD.

2. Incoherent frames

2.1. Objective function

Let μ be the (desired) mutual coherence. The constraint

$$|\boldsymbol{d}_i^T\boldsymbol{d}_j| \leq \mu, \ \forall i \neq j \tag{4}$$

is equivalent to

$$\left\{ \begin{array}{l} \|\boldsymbol{d}_i - \boldsymbol{d}_j\|^2 \geq M \\ \|\boldsymbol{d}_i + \boldsymbol{d}_j\|^2 \geq M \end{array} \right. , \ \forall i \neq j \tag{5}$$

with $M = 2(1-\mu)$ (and so $0 < M < 2$). Note that one of the above two conditions is automatically satisfied, the first when $\boldsymbol{d}_i^T \boldsymbol{d}_j < 0$, the second when $\boldsymbol{d}_i^T \boldsymbol{d}_j > 0$.

We can thus define a soft distance barrier function

$$b(\boldsymbol{d}_j) = \sum_{i \neq j} [\max(0, M - \|\boldsymbol{d}_i - \boldsymbol{d}_j\|^2) + \max(0, M - \|\boldsymbol{d}_i + \boldsymbol{d}_j\|^2)] \quad (6)$$

that will serve to loosely bound the coherence. Such terms were introduced in a contrastive loss function in [31].

To find Grassmannian frames, we use in (3) an objective function f that has two terms: one is quadratic and the other is the barrier. The function is

$$f(\boldsymbol{D}) = \frac{1}{2} \sum_{j=1}^{n} f_j(\boldsymbol{d}_j), \quad (7)$$

where

$$f_j(\boldsymbol{d}_j) = \|\boldsymbol{W}_j \bar{\boldsymbol{D}}_j^T \boldsymbol{d}_j\|^2 + \lambda b(\boldsymbol{d}_j). \quad (8)$$

Here, $\bar{\boldsymbol{D}}_j$ is the matrix $\boldsymbol{D}$ without its j-th column, $\lambda > 0$ is a trade-off factor and $\boldsymbol{W}_j = \mathrm{diag}_i(w_{ij})$ is a diagonal weighting matrix defined by

$$w_{ij}^2 = \max(|\boldsymbol{d}_i^T \boldsymbol{d}_j|/\mu, 1). \quad (9)$$

We note that the first term from (8) was used in [19]; the addition of the second term provides more flexibility. Both terms encourage the current atom to get farther away more from nearby atoms than from more distanced atoms; the first term imposes conditions for all atoms, as the weights w_{ij} are always at least equal to 1; in the second term, only the nearby atoms matter: if atom $\boldsymbol{d}_i$ is such that $|\boldsymbol{d}_i^T \boldsymbol{d}_j| \leq \mu$, then its contribution to the barrier (6) is zero.

2.2. Optimization algorithm

To minimize (7), we adopt the typical block coordinates descent procedure that optimizes one atom while all the others are fixed. Due to the quadratic form of (7), optimizing atom $\boldsymbol{d}_j$ is equivalent to minimizing (8).

Assuming also that the weights (9) are fixed for the current $\boldsymbol{d}_j$, we use the gradient of (8) with respect to $\boldsymbol{d}_j$ to define

$$g_j(\boldsymbol{d}_j) = \frac{1}{2} \frac{\partial f_j}{\partial \boldsymbol{d}_j} = \bar{\boldsymbol{D}}_j \boldsymbol{W}_j^2 \bar{\boldsymbol{D}}_j^T \boldsymbol{d}_j + \lambda \left[\sum_{\|\boldsymbol{d}_i - \boldsymbol{d}_j\| \leq M} (\boldsymbol{d}_i - \boldsymbol{d}_j) + \sum_{\|\boldsymbol{d}_i + \boldsymbol{d}_j\| \leq M} (-\boldsymbol{d}_i - \boldsymbol{d}_j) \right] \quad (10)$$

We attempt the minimization of (8) using a gradient method. The basic step is

$$\boldsymbol{d}_j \leftarrow \boldsymbol{d}_j - \gamma_k g_j(\boldsymbol{d}_j), \tag{11}$$

where the step size γ_k at the k-th frame update iteration is

$$\gamma_k = \gamma_0 \frac{\rho^k}{2^{\nu_k}}, \tag{12}$$

where $\rho < 1$ is a step size decrease factor and $\nu_k \in 0 : \nu_{\max}$ is an integer, chosen as explained below. We tacitly assume that the atom given by (11) is normalized after the gradient step. In a frame update iteration, all atoms are updated via (12) in a random order different at each iteration.

Algorithm 1 presents the operations of our method, named IDB (Incoherent frames via Distance Barrier). As typical in gradient algorithms, it is useful to decrease the step size as we approach the solution (in our case, a local minimum, since the overall problem is not convex). A decrease factor ρ is usual for this purpose. However, while large steps are useful especially in the first iterations, they can also produce undesired effects. We introduce a simple adaptive measure to cope with this problem. Let $c_{\max} = \|\bar{\boldsymbol{D}}_j^T \boldsymbol{d}_j\|_\infty$ be the maximum correlation between the current atom and another atom. We first use (11) with step size given by (12) with $\nu_k = 0$. If the maximum correlation of the new atom with the other atoms is less than $c_{\max}$, meaning that coherence has been reduced, then we keep the atom; if not, we halve the step size, i.e., we increase ν_k and proceed similarly until either the maximum correlation decreases or $\nu_k = \nu_{\max}$; in the latter case, the maximum correlation may not decrease, but the step size is small and instability is unlikely to occur.

For stopping the algorithm, we provide a tolerance ε. If the current frame reaches the desired coherence within this tolerance, the algorithm is considered successful and the iterations are stopped. Otherwise, the algorithm is run for a maximum number of iterations, K.

Remark 1. *(Complexity)* Due to the stopping criterion, a successful frame design with IDB may take a small number of iterations. However, if the desired coherence is not reached, the full number of iterations is run.

The complexity of an iteration is low. The computation of the gradient needs two matrix-vector multiplications. Similarly, computing the weights in steps 5-6 of Algorithm 1 and coherences in steps 9 and 12 require matrix-vector multiplications. All other operations are vectorial. Since an iteration consists of a sweep of all atoms, its complexity is $O(mn^2)$. ■

Algorithm 1: IDB (Incoherent frames via Distance Barrier)

Data: frame size $m \times n$
maximum number of iterations K
target mutual coherence μ
trade-off factor λ from (8)
initial step size γ_0
step size decrease factor ρ
maximum halving steps number $\nu_{\max}$
stopping tolerance ε

Result: incoherent frame $\boldsymbol{D}_{\text{best}}$

Initialization: $\boldsymbol{D} \leftarrow$ random matrix with $\|\boldsymbol{d}_j\| = 1,\ \forall j$
Set $\boldsymbol{D}_{\text{best}} = \boldsymbol{D}$ and μ_{best} equal to the coherence of $\boldsymbol{D}$
for $k = 1$ **to** K **do**
    for $j =$ random permutation of $2 : n$ **do**
        Set $\boldsymbol{v} = \boldsymbol{D}^T \boldsymbol{d}_j$, $\boldsymbol{v}_j \leftarrow 0$
        Compute weights: $\boldsymbol{w}^2 = \max(|\boldsymbol{v}|/\mu, 1)$
        Compute gradient $g_j(\boldsymbol{d}_j)$ with (10)
        Initialize step size: $\gamma \leftarrow \gamma_0$
        Set maximum correlation: $c_{\max} \leftarrow \max_i |v_i|$
        for $\nu_k = 0 : \nu_{\max}$ **do**
            Compute new atom with (11): $\boldsymbol{d} \leftarrow \boldsymbol{d}_j - \gamma g_j(\boldsymbol{d}_j)$ and normalize
            if $\max_{i \neq j} |\boldsymbol{d}_i^T \boldsymbol{d}| < c_{\max}$ **then**
                break
            Halve step size: $\gamma \leftarrow \gamma/2$
        Save atom: $\boldsymbol{d}_j \leftarrow \boldsymbol{d}$
    Decrease step size: $\gamma_0 \leftarrow \rho\gamma_0$
    Compute μ_k, the coherence of $\boldsymbol{D}$
    if $\mu_k < \mu_{\text{best}}$ **then**
        $\mu_{\text{best}} \leftarrow \mu_k$, $\boldsymbol{D}_{\text{best}} \leftarrow \boldsymbol{D}$
    if $\mu_k - \mu < \varepsilon$ **then**
        break

Remark 2. *(Convergence)* As the objective function (7) is not convex, the gradient method we propose can only reach a local minimum. This would be guaranteed only if the gradient step size is small enough, which we only enforce heuristically. However, as it will be seen, once IDB reaches a frame having the coherence equal to the target μ, it tends to stay there. Here is an argument.

When the mutual coherence attains the target μ in Algorithm 1, the soft barrier (6) is equal to zero and $\boldsymbol{W}_j = \boldsymbol{I}$ for all j. So, the gradient 10 becomes

$$g_j(\boldsymbol{d}_j) = \bar{\boldsymbol{D}}_j \bar{\boldsymbol{D}}_j^T \boldsymbol{d}_j. \tag{13}$$

Consider the ideal case of an equiangular frame $\boldsymbol{D}$ and assume that the target coherence μ is equal to the coherence of $\boldsymbol{D}$ and so $\boldsymbol{d}_i^T \boldsymbol{d}_j = \pm\mu$ for all $i \neq j$. Since, for any vector $\boldsymbol{x} \in \mathbb{R}^m$, it holds that [2]

$$\boldsymbol{x} = \frac{1}{a} \sum_{i=1}^{n} (\boldsymbol{d}_i^T \boldsymbol{x}) \boldsymbol{d}_i,$$

where a is a constant, by taking $\boldsymbol{x} = \boldsymbol{d}_j$ and the right combination of signs, it results that

$$\boldsymbol{d}_j = \frac{1}{a} \boldsymbol{d}_j + \mu \sum_{i \neq j} \boldsymbol{d}_i, \tag{14}$$

which means that $\boldsymbol{d}_j$ has the direction of the sum of all other atoms. Since (13) becomes

$$g_j(\boldsymbol{d}_j) = \mu \sum_{i \neq j} \boldsymbol{d}_i,$$

we conclude that the gradient $g_j(\boldsymbol{d}_j)$ has the same direction as $\boldsymbol{d}_j$. So, even though the gradient is not zero, the atom remains unchanged after normalization. This shows that, at least for equiangular frames, our objective 7 has the same global minimum as (1) and this minimum is also a stationary point for our method.

Of course, not all frames are equiangular and the above conditions are not met in practice, but they suggest that a small step size can assure near-stationarity. So, this is an explanation for the nice behaviour of IDB when mutual coherence reaches values near μ. ■

2.3. Bisection search

The most important parameter of IDB is the target coherence μ. Similarly to algorithm ISPM [19], IDB has the property of either converging to μ independently of the random initialization or hovering rather far from it;

Fig. 1, to be explained later, shows the typical behavior in case of success. Although empirical choice of μ can give quick results for an experienced user, a bisection procedure can be used for systematic design. Since bisection was only mentioned and ISPM was used under its possible performance in comparisons reported in some recent works, we detail here the procedure in Algorithm 2.

The initial search interval $[\mu_{\min}, \mu_{\max}]$ can be chosen easily as follows. The minimum value of the target coherence, $\mu_{\min}$, is taken as the lower bound of the coherence, namely the largest between the Welch [32] and Levenstein [14] bounds (the latter valid for large n/m):

$$\max\left(\sqrt{\frac{n-m}{m(n-1)}}, \sqrt{\frac{3n-m^2-2m}{(m+2)(n-m)}}\right). \tag{15}$$

For the upper end of the interval, we can take $\mu_{\max}$ as the coherence of a random frame.

Bisection runs as usual. We run IDB with μ at the middle of the current search interval. The design is considered a success if the target coherence is reached within 10ε; the upper end of the interval is lowered to the obtained coherence value. If the design is a failure, we raise the lower end of the interval to μ; however, we check if the obtained coherence is lower than the upper end; if so, we move it accordingly; this may happen in the first few bisections, when the interval is still large. We run the bisection for a maximum number of iterations, K_{bis}, or until the search interval has become smaller than the tolerance ε used in IDB; a special exit condition is when the obtained coherence is outside the search interval, which means that the decrease potential of the method has been exhausted.

3. Results—incoherent frames

In the IDB Algorithm 1, with very few exceptions that will be explicitly mentioned, we have used the following values of the parameters: $\lambda = 0.2$, $\gamma_0 = 0.1$, $\rho = 0.999$, $\nu_{\max} = 4$. They were found using an empirical grid search over a few dimensions m and n. Although these values may not be necessarily the best for each m and n, they are remarkably robust over a broad range of dimensions. The number of iterations and the stopping tolerance are $K = 2000$, $\varepsilon = 10^{-4}$. In the bisection Algorithm 2, we have used a maximum number of iterations $K_{\text{bis}} = 15$.

For comparison we have used the best algorithms for frames of medium and large size, as indicated by the results in recent literature. The experi-

Algorithm 2: Best coherence using bisection.

Data: frame size $m \times n$
initial coherence bounds $\mu_{\min}$, $\mu_{\max}$
stopping tolerance ε
maximum number of bisections K_{bis}
Result: incoherent frame $\boldsymbol{D}$

Set $\mu_{\text{best}} = 1$
for $i = 1 : K_{\text{bis}}$ **do**
  Compute $\mu = (\mu_{\min} + \mu_{\max})/2$
  Run IDB$(m, n, \ldots, \varepsilon)$ and obtain frame $\boldsymbol{D}_i$ with coherence μ_i
  if $\mu_i < \mu_{\text{best}}$ **then**
    $\mu_{\text{best}} \leftarrow \mu_i$, $\boldsymbol{D} \leftarrow \boldsymbol{D}_i$
  if $\mu_i < \mu + 10\varepsilon$ **then**
    if $\mu_i \geq \mu_{\max}$ **then**
      break
    $\mu_{\max} \leftarrow \mu_{\text{best}}$
  else
    $\mu_{\min} \leftarrow \mu$
    $\mu_{\max} \leftarrow \min(\mu_i, \mu_{\max})$
  if $\mu_{\max} - \mu_{\min} \leq \varepsilon$ **then**
    break

ments have been performed on a laptop with an i7 CPU at 2.6GHz (6 cores) and 16GB RAM.

ISPM [19] is in direct relation with IDB: its objective is the first term of (7); however, the optimization algorithm is inspired from the power method, hence quite different from IDB; like in IDB, we have added a stopping criterion to the original algorithm ISPM and used the bisection procedure from Algorithm 2. Unlike most other methods, ISPM designs unit norm tight frames (UNTF), hence it has an extra constraint. The code for IDB and ISPM can be found at `https://asydil.upb.ro/software/`.

Iterative Collision-Based Packing (ICBP) [16] uses an heuristic algorithm that attempts to pack spheres with maximal radius, centered on the unit sphere. The radius is adjusted iteratively. The algorithm may give different results for different initializations. Like for IDB and ISPM, the coherence is not necessarily decreasing in each iteration. We have used authors' im-

Table 1: Coherence results for dimensions from [20].

m	n	bound	ISPM	FLIP	ISPM +FLIP	ICBP	IDB
50	60	0.0582	0.0762	0.0666	0.0634	0.0644	**0.0626**
90	100	0.0335	0.0532	0.0384	0.0384	0.0383	**0.0374**
100	110	0.0303	0.0494	0.0349	0.0350	0.0345	**0.0340**
200	210	0.0155	0.0282	0.0181	0.0181	0.0185	**0.0176**
300	310	0.0104	0.0203	0.0122	0.0123	0.0124	**0.0120**
500	510	0.0063	0.0134	0.0074	0.0075	0.0077	**0.00732**
500	550	0.0135	0.0201	0.0156	**0.0151**	0.0155	**0.01511**
700	710	0.0045	0.0101	**0.0053**	0.0054	0.0056	**0.00526**
900	1100	0.0142	0.0185	0.0163	0.0156	0.0160	**0.0155**
2000	2500	0.0100	0.0130	0.0126	0.0121	-	**0.01094**

plementation[1], running the algorithm for 2000 iterations if not otherwise mentioned.

FLIP [20] attempts to directly optimize (1), with a procedure based on linear programming (LP), with guaranteed convergence (to a local minimum). Authors' implementation[2] recommends using the 'active-set' option of the MATLAB function `linprog`; since this option is obsolete and with the 'dual-simplex' option the LP algorithm often fails to converge for the FLIP problem, we have used the 'interior-point' option. We have used a stopping tolerance of 10^{-3}, like in the original code; smaller values give a slight improvement of the result, with a significant increase of the execution time. The results and execution times are in line with those from [20], with small variations.

For other methods, like TELET [21], where we did not find a public version of the software, we take the coherence results directly from the respective papers, but we are unable to provide execution times.

Table 1 shows results for frame sizes used in [20], with low values of the overcompleteness factor n/m. The lower bound from the third column is given by (15). The results for FLIP are the best between those from [20] and those obtained by us (in most cases, they were nearly equal). ISPM+FLIP denotes the algorithm in which the frame produced by ISPM is used as

[1] `https://codeocean.com/capsule/7026868/tree/v1`

[2] `https://se.mathworks.com/matlabcentral/fileexchange/98164-design-of-grassmannian-frames`

Table 2: Execution times (in seconds) for designing the frames from Table 1.

m	n	ISPM	FLIP	ICBP	IDB
50	60	2	3	6	14
90	100	6	8	24	34
100	110	5	13	29	62
200	210	31	120	206	275
300	310	81	530	533	768
500	510	211	3490	5880	2220
500	550	174	4230	8040	3770
700	710	589	12400	16500	5300
900	1100	805	45800	120000	31200

initialization for FLIP; its results are taken from [20]. It is visible that the best results are given by IDB, although FLIP and ICBP are not far behind. We note that ICBP converges for most of the dimensions in the given number of iterations. The performance of ISPM is not that good for near-square frames. We have also tested IFD-AMPP [13], which gives results similar to those of FLIP and ICBP in the first half of the table; for the larger frames, the results are worse, even behind ISPM; for large n/m IFD-AMPP appears to diverge, so we did not use it in further tests.

Table 2 gives the execution times of the algorithms. ISPM is the fastest, ICBP the slowest, while IDB and FLIP have comparable execution times, with an advantage for FLIP at low dimensions and for IDB at high dimensions, which suggests that IDB scales better. The low execution time of ISPM explains why it was used as initialization for FLIP.

Since IDB and ISPM are run through the bisection algorithm, the total number of iterations for a design can vary a lot, as the successful designs take sometimes much less than 2000 iterations; also, the maximum number of 15 bisection steps is not necessarily attained. For IDB, the total number of iterations vary between 12950 and 22654 (clearly less than the maximum of 30000 iterations).

The case 2000×2500, not shown in Table 2, deserves some explanations. In this case, we have run IDB on a different computer, about 20-30% slower than our main one. We have used an empirical search, since 2000 iterations took about two days on that computer; we needed 8 runs to reach the coherence shown in the table. On the same computer, ICBP needed more than 3 hours for 10 iterations, so we abandoned the run. The coherence results for the other methods are taken from [20], where the execution times

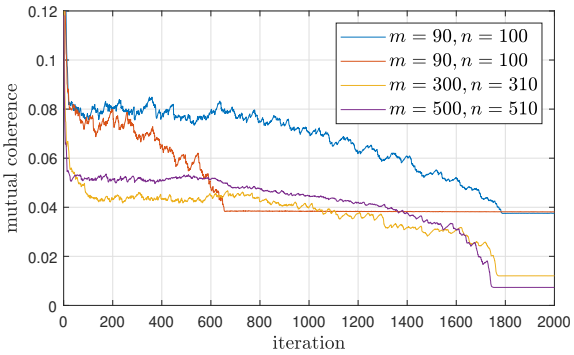

Figure 1: Mutual coherence evolution for algorithm IDB.

are not reported.

Figure 1 shows the mutual coherence evolution in several runs of our IDB algorithm. Three of them give the best coherence for the respective dimensions (see again Table 1). The second run for $m = 90$, $n = 100$ is for $\mu = 0.0380$, only slightly higher than the best result of 0.0374; less than 700 iterations are necessary. If we set $\mu = 0.04$, less than 100 iterations are required for a satisfactory result, and so on as μ grows. So, upper bounds of the attainable coherence can be quickly obtained, which helps reducing the total number of iterations in the bisection algorithm. Note that here the stopping criterion was not imposed and the IDB algorithm has run for 2000 iterations; the figure also illustrates the stable behavior when the target coherence has been attained, as discussed in Remark 2.

Table 3 shows results for dimensions used in [21], with high overcompletenes factor n/m; the corresponding execution times are given in Table 4. In most cases, the winner is clearly ISPM (inserted in the current bisection algorithm), with IDB a close second, with occasionally the best performance. ISPM is about three times faster than IDB, a fact which makes it more attractive. We note that we changed the initial gradient step size of IDB to $\gamma_0 = 0.01$ in the 20×5000 example; in the same example, ICBP was stopped after 30 hours, in which only 500 iterations were made.

As we were preparing the current manuscript, we learned about a new paper [33], transforming the minimization of the coherence under a tightness constraint into a unconstrained problem, based on smoothing techniques. So, the produced frame is near to an UNTF. The resulting algorithm is called CPM. Some results for frames with overcompleteness factor going from 2 to 64 are given in Table 5. The results for CPM and ICBP are taken from [33]; both algorithms are run there for 5000 iterations. We see that

Table 3: Coherence results for dimensions from [21].

m	n	bound	TELET	ISPM	FLIP	ICBP	IDB
23	500	0.2785	0.3703	**0.3517**	0.3841	0.5386	0.3521
25	600	0.2692	0.3641	**0.3434**	0.3746	0.5255	0.3440
25	800	0.2871	0.3900	**0.3680**	0.3966	0.5949	0.3687
25	1000	0.2972	0.4340	**0.3850**	0.4108	0.6383	0.3863
25	1200	0.3036	0.4486	**0.3979**	0.4251	0.6636	0.4003
30	800	0.2417	0.3458	**0.3171**	0.3482	0.4888	0.3174
30	1000	0.2564	0.3655	**0.3346**	0.3471	0.5717	0.3352
30	1200	0.2655	0.3851	**0.3481**	0.3753	0.5839	0.3489
40	800	0.1542	0.2909	**0.2514**	0.2821	0.3213	0.2516
40	1000	0.1809	0.3119	0.2654	0.2949	0.3825	**0.2650**
40	1200	0.1985	0.3275	0.2776	0.3054	0.3998	**0.2770**
50	1000	0.1379	0.2788	**0.2240**	0.2497	0.2902	**0.2240**
20	5000	0.3645	0.7422	**0.5535**	0.7448	0.9336	0.5625

IDB is the best in 8 of the 11 examples (tied with ISPM in one example), while ISPM is best in 3 examples and ICBP only in one.

The overall conclusion is that our IDB method appears to be the best for frames with small overcompleteness factor n/m. When this factor grows, our previous ISPM method, used in the proposed bisection procedure, becomes very competitive, having the best trade-off between coherence results and execution times. However, IDB gives the best results for high n/m in almost half of the designs we have tried.

4. Incoherent dictionary learning

To design an algorithm for solving the incoherent DL problem (2), we adopt the general structure of AK-SVD [30]. In each iteration, since the function f in (2) does not depend on $\boldsymbol{X}$, the sparse representations $\boldsymbol{X}$ are found as usual with Orthogonal Matching Pursuit (OMP) [34]. Then, the atoms of the dictionary are updated one by one, while keeping the other atoms fixed; we give below the details of this operation.

Denoting $\boldsymbol{E} = \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}$ the error matrix, the error without contribution of atom $\boldsymbol{d}_j$ is

$$\boldsymbol{F} = [\boldsymbol{E}]_{\mathcal{I}_j} + \boldsymbol{d}_j \boldsymbol{x}_j^T, \tag{16}$$

where the row vector $\boldsymbol{x}_j^T$ contains the nonzero elements of the j-th row of $\boldsymbol{X}$, whose (column) indices are $\mathcal{I}_j$. By $[\boldsymbol{E}]_{\mathcal{I}_j}$ we denote the restriction of

Table 4: Execution times (in seconds) for designing the frames from Table 3.

m	n	ISPM	FLIP	ICBP	IDB
23	500	190	157	790	551
25	600	197	270	1760	573
25	800	377	514	3630	1180
25	1000	611	1150	6100	1380
25	1200	716	1300	9470	2090
30	800	356	567	3700	890
30	1000	413	704	6170	1200
30	1200	762	1770	8860	2030
40	800	368	855	3980	952
40	1000	408	1560	7180	1360
40	1200	703	2740	11300	2000
50	1000	508	2250	6680	1220
20	5000	8130	18800	112000	25800

$\boldsymbol{E}$ to the columns with indices in $\mathcal{I}_j$. The objective for the optimization of atom $\boldsymbol{d}_j$, when everything else is fixed in (2), is

$$\|\boldsymbol{F} - \boldsymbol{d}_j \boldsymbol{x}_j^T\|_F^2 + \gamma f_j(\boldsymbol{d}_j), \tag{17}$$

where the function f_j is given in (8) and is the same used for designing incoherent frames. We note that the first term of (8) is similar to that in [24], where however there was no weight.

Setting the gradient to zero gives the optimal atom

$$\boldsymbol{d}_j \leftarrow \boldsymbol{F}\boldsymbol{x}_j - \gamma g_j(\boldsymbol{d}_j), \tag{18}$$

where $g_j(\cdot)$ is given by (10). The vector produced by (18) must be normalized.

Then, the representations are updated like in AK-SVD, since the incoherence inducing function (8) does not depend on $\boldsymbol{x}_j^T$. The relation is

$$\boldsymbol{x}_j \leftarrow \boldsymbol{F}^T \boldsymbol{d}_j, \tag{19}$$

where we use the updated atom given by (18). Once the atom and its representation are updated, we recompute the error

$$[\boldsymbol{E}]_{\mathcal{I}_j} \leftarrow [\boldsymbol{E}]_{\mathcal{I}_j} - \boldsymbol{d}_j \boldsymbol{x}_j^T \tag{20}$$

and we go to the next atom.

Table 5: Coherence results for dimensions from [33].

m	n	bound	CPM	ISPM	ICBP	IDB
64	128	0.0887	0.0982	0.1025	0.0965	**0.0964**
64	256	0.1085	0.1316	0.1311	**0.1282**	0.1301
64	384	0.1143	0.1484	**0.1473**	0.1478	**0.1473**
64	640	0.1187	0.1701	**0.1679**	0.1725	0.1684
64	960	0.1208	0.1877	**0.1850**	0.1965	0.1851
64	1280	0.1219	0.2065	0.1975	0.2861	**0.1972**
64	1600	0.1225	0.2178	0.2077	0.3131	**0.2071**
64	2240	0.1232	0.2353	0.2235	0.3715	**0.2225**
64	2880	0.1236	0.2484	0.2356	0.4247	**0.2341**
64	3520	0.1239	0.2590	0.2455	0.4630	**0.2442**
64	4096	0.1240	0.2668	0.2527	0.4981	**0.2525**

The above operations are summarized in Algorithm 3, named IDB-DL. Only step 5, gradient computation, adds to the complexity of the algorithm, compared to AK-SVD.

5. Results—dictionary learning

The DL experiments were performed in Matlab (R2022b), on a desktop with an i7 processor with 16 cores, a base frequency of 2.90 GHz, and 80 GB RAM. The source code for our algorithm is available at `https://asydil.upb.ro/software/`.

The data are generated artificially, using random dictionaries, with an SNR of about 30 dB. The generated samples are matrices with $N = 1000$ columns (signals) and $m = 64$ rows (features). The dictionaries have $n = 100$ or $n = 160$ atoms, with a sparsity constraint of $s = 5$, $s = 8$, or $s = 10$. AK-SVD and our IDB-DL algorithm have been run for 100 iterations. The execution time of IDB-DL is less than 10% larger than that of AK-SVD, which is one of the fastest DL algorithms.

Our algorithm has three parameters: M (or $\mu = 1 - M/2$), γ and λ. We have run simulations on a grid covering a relatively large space for the three parameters. For γ we took values from 10^{-2} to 10^{1} and for λ from 10^{-2} to 10^{3}, both logarithmically spaced. For the barrier margin M we used values in $\{1, 1.2, 1.4, 1.6, 1.8\}$, corresponding to $\mu \in \{0.5, 0.4, 0.3, 0.2, 0.1\}$.

In Figs. 2 and 3 we give representative results obtained for $n = 100$ and $s = 5$, averaged over 100 runs, each using the same random initialization for IDB-DL and AK-SVD; the results are not much different in other

Algorithm 3: An iteration of IDB-DL.

Data: frame size $m \times n$
data matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times N}$
initial dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times n}$
sparsity level s
target mutual coherence μ
trade-off factor λ from (8)
trade-off factor γ from (17)
Result: optimized dictionary $\boldsymbol{D}$

Compute sparse representations: $\boldsymbol{X} = \mathrm{OMP}(\boldsymbol{Y}, \boldsymbol{D}, s)$
Compute error matrix $\boldsymbol{E} = \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}$
for $j = 1 : n$ **do**
    Compute error $\boldsymbol{F}$ with (16)
    Compute gradient $g_j(\boldsymbol{d}_j)$ with (10)
    Update atom $\boldsymbol{d}_j$ with (18) and normalize
    Update representations with (19)
    Update error $\boldsymbol{E}$ with (20)

cases. We see that there are many good trade-offs between the two performance indices, representation error (per element) $\|\boldsymbol{E}\|_F/\sqrt{mN}$ and mutual coherence, including the case where *both are better* than in AK-SVD.

In Figure 2, $\gamma = 0.5$ is fixed. For small values of λ, which basically cancel the barrier term in (8), there is a small improvement in both error and coherence with respect to AK-SVD. When λ grows, there is an interval where performance improves in both indices. When λ is large, the error increases while coherence does not change much or even gets worse; as the gradient depends only on local information, too aggressive atom updates (18) take the current atom farther from nearby atoms but may move it too close to other atoms. So, for many λ values, the barrier term has a clear contribution to the overall improvement.

In Figure 3, $\lambda = 25$ is fixed. Small values of the incoherence factor γ in (17) expectedly make little difference, but usually to the better, with respect to AK-SVD. As γ grows, positive effects are visible in both error and coherence. However, error is improved on a smaller interval than the coherence; a too large value of γ worsens the error and, after a while, also the coherence. Here the effects of a too aggressive approach, manifested in a large γ, are obvious, as they lead to a clear deterioration of the coherence.

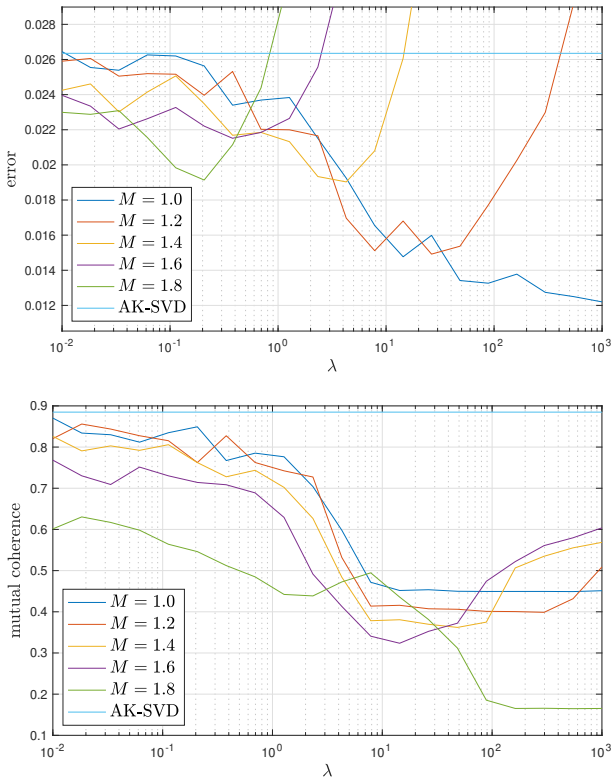

Figure 2: Error (up) and mutual coherence (down) results for algorithm IDB-DL, with $\gamma = 0.5$.

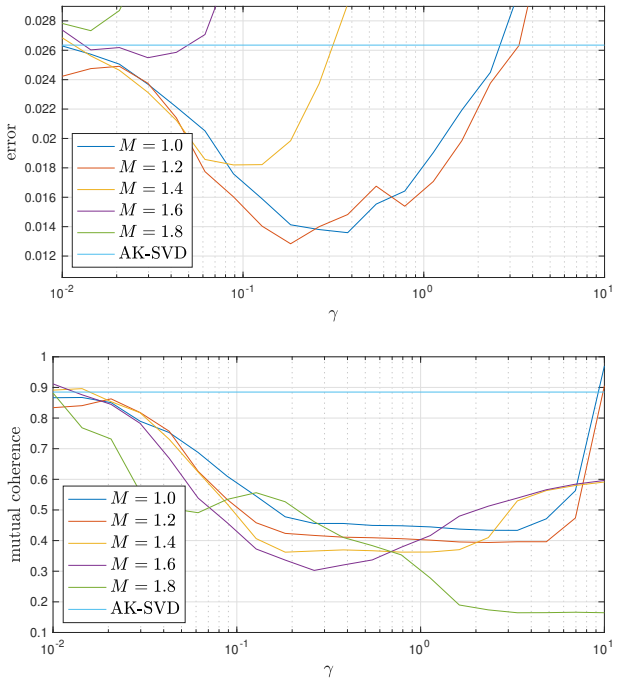

Figure 3: Error (up) and mutual coherence (down) results for IDB-DL, with $\lambda = 25$.

The ideal role of the barrier parameter M is to define a hypersphere of radius M around an atom that does not contain other atoms. Since the barrier is soft, this ideal condition is not met, but still the atoms are kept well apart. A small M value imposes mild constraints on the coherence, thus allowing also error improvement. We see from Figs. 2 and 3 that, for $M = 1$ and $M = 1.2$, simultaneous improvement in both coherence and error is possible on a larger range of the parameters γ and λ than for higher values of M. Expectedly, the best coherence values are obtained for large M, but only at the expense of a large error.

For a single choice of parameters, the evolution of the error and the final distribution of the scalar products between atoms are shown in Figure 4, averaged over 100 runs. The error is shown with a mean and standard deviation plot and the improvement given by our algorithm is obvious. The change in the scalar products between atoms is visible almost only at the large values, i.e., where it matters more.

Finally, to illustrate the robustness of the parameters γ and λ, we compare IDB-DL and AK-SVD on a grid with $\gamma \in \{0.2, 0.5, 1.1.5, 2\}$ and $\lambda \in \{5, 10, 20, 30, 40, 50\}$. We generated sets of $N = 1000$ signals having $m \in \{64, 128\}$ features. The dictionary size is $n \in \{100, 160\}$ atoms, with sparsity constraints $s \in \{5, 8, 10\}$. So, for each γ, λ and M, there are 12 different experiments, repeated 10 times with different initializations of the dictionaries. Figs. 5 and 6 show the number of times in which IDB-DL gave a better result than AK-SVD, in terms of error and mutual coherence, for $M = 1$ and $M = 1.2$, respectively. We see that the mutual coherence is almost always better for IDB-DL, which is not surprising, but also that error is improved most of the times for many values of the parameters.

6. Conclusion

We have given efficient and fast algorithms for designing incoherent frames and for solving the incoherent dictionary learning problem. The designed frames have better mutual coherence than that produced by the best existing algorithms. The proposed IDB-DL algorithm has the potential to replace the algorithms solving the standard DL problem, like AK-SVD, as it produces dictionaries that are not only more incoherent, but also provide better representations.

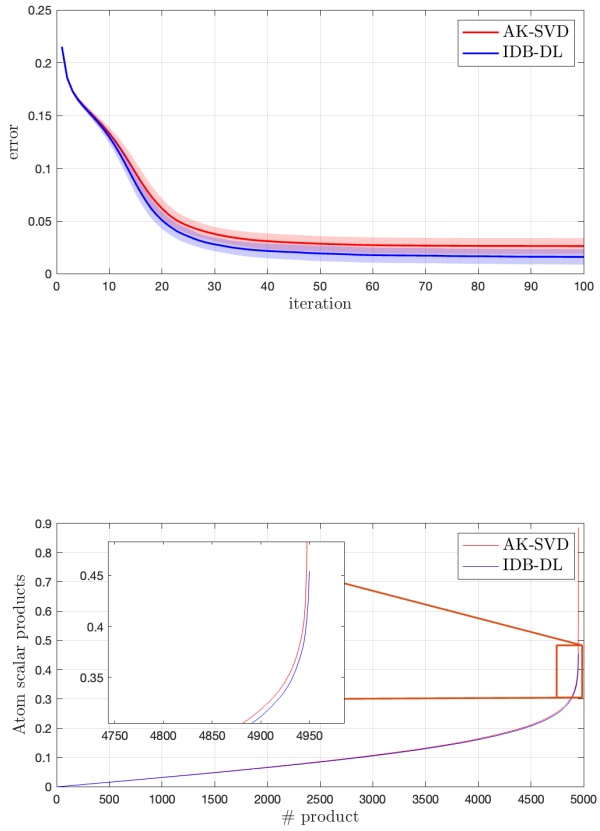

Figure 4: Error evolution (up) and sorted final products $|\boldsymbol{d}_i^T \boldsymbol{d}_j|$ (down) for IDB-DL, with $M = 1$, $\gamma = 0.5$ and $\lambda = 25$.

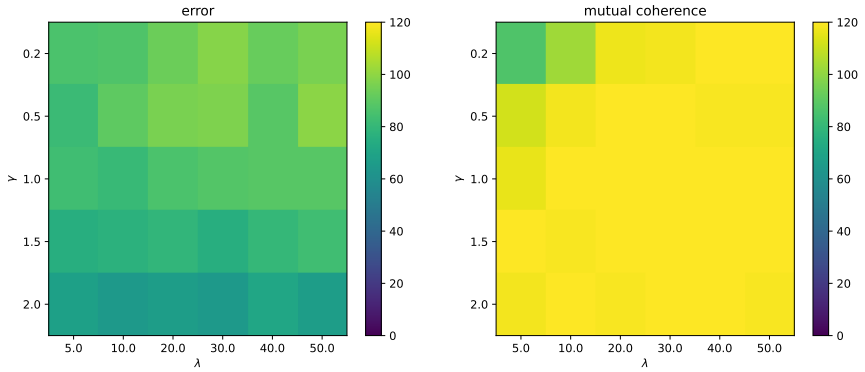

Figure 5: Number of cases where IDB-DL is better than AK-SVD, for $M = 1$. Left: error. Right: mutual coherence.

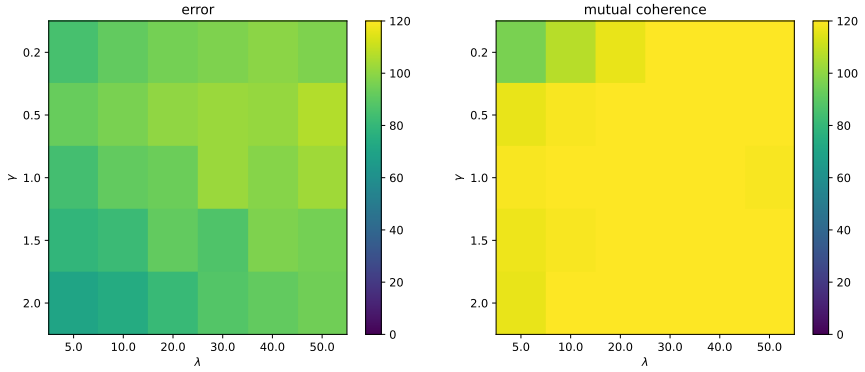

Figure 6: Number of cases where IDB-DL is better than AK-SVD, for $M = 1.2$. Left: error. Right: mutual coherence.

References

[1] O. Christensen, An Introduction to Frames and Riesz Bases, Birkhauser, 2016.

[2] T. Strohmer, R. Heath, Grassmannian frames with applications to coding and communication, Appl. Comput. Harmon. Anal. 14 (2003) 257–275.

[3] R. Gohary, T. Davidson, Noncoherent MIMO communication: Grassmannian constellations and efficient detection, IEEE Transactions on Information Theory 55 (2009) 1176–1205.

[4] J. Kim, K. Cheun, S. Choi, Unitary space-time constellations based on quasi-orthogonal sequences, IEEE Transactions on Communications 58 (2010) 35–39.

[5] A. Medra, T. Davidson, Flexible codebook design for limited feedback systems via sequential smooth optimization on the Grassmannian manifold, IEEE Transactions on Signal Processing 62 (2014) 1305–1318.

[6] M. Elad, Sparse and redundant representations: from theory to applications in signal and image processing, Springer, 2010.

[7] B. Dumitrescu, P. Irofti, Dictionary Learning Algorithms and Applications, Springer, 2018.

[8] W. Bajwa, R. Calderbank, D. Mixon, Two are better than one: Fundamental parameters of frame coherence, Appl. Comput. Harmon. Anal. 33 (2012) 58–78.

[9] M. Fickus, D. Mixon, J. Tremain, Steiner equiangular tight frames, Linear algebra and its applications 436 (2012) 1014–1027.

[10] J. Tropp, I. Dhillon, R. Heath Jr., T. Strohmer, Designing structured tight frames via an alternating projection method, IEEE Trans. Info. Th. 51 (2005) 188–209.

[11] G. Li, Z. Zhu, D. Yang, L. Chang, H. Bai, On Projection Matrix Optimization for Compressive Sensing Systems, IEEE Trans. Signal Proc. 61 (2013) 2887–2898.

[12] E. Tsiligianni, L. Kondi, A. Katsaggelos, Construction of Incoherent Unit Norm Tight Frames With Application to Compressed Sensing, IEEE Trans. Info. Theory 60 (2014) 2319–2330.

[13] M. Sadeghi, M. Babaie-Zadeh, Incoherent Unit-Norm Frame Design via an Alternating Minimization Penalty Method, IEEE Signal Proc. Letters 24 (2017) 32–36.

[14] H. Zörlein, M. Bossert, Coherence Optimization and Best Complex Antipodal Spherical Codes, IEEE Trans. Signal Proc. 63 (2015) 6606–6615.

[15] H. Laue, W. Du Plessis, A coherence-based algorithm for optimizing rank-1 Grassmannian codebooks, IEEE Signal Processing Letters 24 (2017) 823–827.

[16] B. Tahir, S. Schwarz, M. Rupp, Constructing Grassmannian frames by an iterative collision-based packing, IEEE Signal Processing Letters 26 (2019) 1056–1060.

[17] C. Rusu, N. González-Prelcic, Designing Incoherent Frames Through Convex Techniques for Optimized Compressed Sensing, IEEE Trans. Signal Proc. 64 (2016) 2334–2344.

[18] C. Rusu, N. González-Prelcic, R. Heath Jr, Algorithms for the construction of incoherent frames under various design constraints, Signal Processing 152 (2018) 363–372.

[19] B. Dumitrescu, Designing Incoherent Frames With Only Matrix-Vector Multiplications, IEEE Signal Proc. Letters 24 (2017) 1265–1269.

[20] R. Jyothi, P. Babu, P. Stoica, Design of high-dimensional grassmannian frames via block minorization maximization, IEEE Communications Letters 25 (2021) 3624–3628.

[21] R. Jyothi, P. Babu, TELET: A monotonic algorithm to design large dimensional equiangular tight frames for applications in compressed sensing, Signal Processing 195 (2022) 108503.

[22] H. Bai, S. Li, X. He, Sensing matrix optimization based on equiangular tight frames with consideration of sparse representation error, IEEE Transactions on Multimedia 18 (2016) 2040–2053.

[23] F. Tong, L. Li, H. Peng, D. Zhao, Progressive coherence and spectral norm minimization scheme for measurement matrices in compressed sensing, Signal Processing 194 (2022) 108435.

[24] C. Sigg, T. Dikk, J. Buhmann, Learning Dictionaries With Bounded Self-Coherence, IEEE Signal Proc. Letters 19 (2012) 861–865.

[25] V. Abolghasemi, S. Ferdowsi, S. Sanei, Fast and incoherent dictionary learning algorithms with application to fMRI, Signal, Image and Video Processing 9 (2015) 147–158.

[26] C. Bao, Y. Quan, H. Ji, A convergent incoherent dictionary learning algorithm for sparse coding, in: European Conference on Computer Vision, Springer, 2014, pp. 302–316.

[27] M. Sadeghi, M. Babaie-Zadeh, Dictionary learning with low mutual coherence constraint, Neurocomputing 407 (2020) 163–174.

[28] B. Mailhé, D. Barchiesi, M. Plumbley, INK-SVD: Learning incoherent dictionaries for sparse representations, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Kyoto, Japan, 2012, pp. 3573–3576.

[29] D. Barchiesi, M. Plumbley, Learning Incoherent Dictionaries for Sparse Approximation Using Iterative Projections and Rotations, IEEE Trans. Signal Proc. 61 (2013) 2055–2065.

[30] R. Rubinstein, M. Zibulevsky, M. Elad, Efficient Implementation of the K-SVD Algorithm Using Batch Orthogonal Matching Pursuit, Technical Report CS-2008-08, Technion Univ., Haifa, Israel, 2008.

[31] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, IEEE, 2006, pp. 1735–1742.

[32] L. Welch, Lower bounds on the maximum cross correlation of signals, IEEE Trans. Info. Th. 20 (1974) 397–399.

[33] F. Tong, D. Zhao, C. Chen, L. Li, Coherence-penalty minimization method for incoherent unit-norm tight frame design, Signal Processing (2022) 108864.

[34] Y. Pati, R. Rezaiifar, P. Krishnaprasad, Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition, in: 27th Asilomar Conf. Signals Systems Computers, volume 1, 1993, pp. 40–44.