Gradient-based joint block diagonalization algorithms:
Application to blind separation of FIR convolutive mixtures

Hicham Ghennioui\(^{a,b,1}\), Nadège Thirion-Moreau\(^a,\ast\), Eric Moreau\(^a\), Driss Aboutajdine\(^{b,1}\)

\(^a\) LSEE, UMR CNRS 6017, Université du Sud Toulon Var, F-83957 La Garde Cédex, France
\(^b\) GSCM-LRIT, Faculté des Sciences, av. Ibn Battouta, BP 1014, Rabat, Morocco

ABSTRACT

This article addresses the problem of the non-unitary joint block diagonalization of a given set of complex matrices. Two new algorithms are provided: the first is based on a classical gradient approach and the second is based on a relative gradient approach. For each algorithm, two versions are provided: the fixed stepsize and the optimal stepsize version. Computer simulations are provided to illustrate the behavior of both algorithms in different contexts. Finally, it is shown that these algorithms enable solving the problem of the blind separation of finite impulse response (FIR) convolutive mixtures of (non-stationary correlated) sources. We focus on methods based on the use of spatial quadratic time–frequency spectra or distributions. The suggested approach main advantage is to enable the elimination of the spatial whitening of the observations which has been proven to establish a bound with regard to the best reachable performances in the blind sources separation context.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

In the recent years, the problem of the joint decomposition of matrix (or tensors) sets has often arisen in the signal processing field, especially in blind source separation (BSS) and array processing applications. Most works have geared toward the three ensuing decompositions: joint diagonalization (JD) [7,10,13,23,24,27,33–35], joint zero-diagonalization (JZD) [2,9,13] and joint block-diagonalization (JBD) of matrix sets. For the recall, the first considered problem was the JD of matrices under the unitary constraint and it has led to the nowadays well-known JADE (joint approximate diagonalization of eigen-matrices) [7] and SOBI (second order blind identification) algorithms.

In this article, we focus on the JBD problem which is a rather general problem since it encompasses the JD problem. In such a decomposition, the matrices under consideration assume a specific algebraic structure, being block diagonal matrices i.e. they are block matrices whose diagonal blocks are square matrices of any (possibly even) size while their off-diagonal blocks are all null. This problem was first considered for positive definite and hermitian block-diagonal matrices and for unitary joint-block diagonalizers in [4,12] using Jacobi like algorithms. Then, in [16,17], we have suggested two alternative methods achieving the same task but for non-unitary joint-block diagonalizers. Finally, it has to be noticed that the algorithm suggested in [28], despite its name, considers a less general problem than the aforementioned references since the joint block-diagonlizer assumes a very particular algebraic structure. It is the reason why this algorithm will not be considered in this work.
Here, we still tackle the problem of the approximate JBD of a given matrix set by discarding the unitary constraint on the joint block-diagonizer but our goal is to impose as few assumptions as possible on the matrix set under consideration. Subsequently, we suggest two new algorithms based on gradient-descent approaches. Both fixed and optimal stepsize versions of the algorithms are provided and studied. The exact calculation of the complex gradient matrix of the used cost function is now performed, whereas an approximation in the calculation was made in [16]. The first algorithm relies upon a classical gradient optimization scheme while the second one relies on a relative gradient optimization scheme. The main advantages of these two algorithms are that they are more general since the real, positive definite or hermitian assumptions regarding the matrices of the considered set are no more necessary. Moreover, the possible calculation at each iteration of the optimal stepsize (as the rooting of a 3rd-degree polynomial) enables a decreasing of the number of iterations required to reach convergence while removing the stepsize choice problem. The algorithmic complexity of the proposed algorithms is established. Computer simulations are provided to illustrate the behavior of the different algorithms in various contexts. Notice that the problem of the identifiability of the joint block diagonalizer is not addressed since we only focus on algorithm aspects. The JBD problem encompassing the JD one, a new non-unitary JD algorithm, based on a relative gradient approach is also derived. Finally, we show how these algorithms find applications in blind separation of FIR convolutive mixtures of non-stationary sources. We focus on methods that are based on spatial quadratic time frequency spectra (SQTFS) or spatial quadratic time frequency distributions (SQTFD) [3] because we want to separate mixtures of deterministic signals as well as mixtures of (correlated) non-stationary stochastic processes. The suggested approaches main advantage is to enable the elimination of the spatial whitening of the observations which has been proven to establish a bound with regard to the best reachable performances in the blind sources separation context [6,11]. Computer simulations are provided to emphasize both the effectiveness and the good performances of the proposed approaches. They are also compared with two other methods: one based on a non-unitary joint block diagonalization algorithm [17] and one based on a classical unitary joint block diagonalization algorithm [14].

The article is organized as follows: the non-unitary joint-block diagonalization problem is stated in Section 2. The two proposed (gradient-based) solutions are derived in Section 3. Both fixed and optimal stepsize versions of the suggested algorithms are delineated. Computer simulations are provided to illustrate the effectiveness and the good performance of the resulting JBD (and JD) algorithms. The purpose of the Section 4 is to enhance the usefulness of these algorithms through one of their possible applications, namely the blind separation of FIR convolutive mixtures of non-stationary sources. It is recalled that one possible manner to tackle such a problem consists of opting for a (non-unitary) joint-block diagonalization algorithm, but first particular matrices have to be properly selected. To that aim, a new automatic t–f points selection procedure dedicated to the non-whitening context is introduced. Computer simulations are performed to illustrate the good performance of the suggested methods and to compare them with "state-of-the-art approaches". Finally, in Section 5, a conclusion is drawn.

2. The non-unitary joint block-diagonalization problem

2.1. Problem statement

The problem of the non-unitary joint block-diagonalization is stated in the following way. We consider a set \( \mathcal{M} \) of \( N_m \) square matrices \( \mathbf{M}_i \in \mathbb{C}^{M \times M} \), for all \( i \in \{1, \ldots, N_m\} \) that all admit the following decomposition:

\[
\mathbf{M}_i = \mathbf{A}_i \mathbf{D}_i \mathbf{A}_i^H,
\]

where \((\cdot)^H\) stands for the transpose conjugate operator. The matrices

\[
\mathbf{D}_i = \begin{pmatrix}
\mathbf{D}_{i,11} & \mathbf{0}_{12} & \cdots & \mathbf{0}_{1r} \\
\mathbf{0}_{21} & \mathbf{D}_{i,22} & \cdots & \vdots \\
\vdots & \vdots & \ddots & \mathbf{0}_{r-1,r} \\
\mathbf{0}_{11} & \cdots & \mathbf{0}_{r-1,r} & \mathbf{D}_{i,rr}
\end{pmatrix},
\]

for all \( i \in \{1, \ldots, N_m\} \) are \((N \times N)\) block diagonal matrices with \( r \) the number of considered blocks \( (r \in \mathbb{N}^+) \). \( \mathbf{D}_{ij}, i, j \in \{1, \ldots, N_m\}, j \in \{1, \ldots, r\} \) are \((n_j \times n_j)\) square matrices so that \( n_1 + \ldots + n_r = N \) and with \( \mathbf{0}_{ij} \) denotes the \((n_j \times n_j)\) null matrix. We further assume that \( \mathbf{A} \) is full column rank and belongs to \( \mathbb{C}^{N \times N} \) with \( M \geq N \) (Assumption A0). The set of the \( N_m \) square matrices \( \mathbf{D}_i \in \mathbb{C}^{N \times N} \) is denoted by \( \mathcal{D} \). The block sizes \( n_j \) for all \( j = 1, \ldots, r \) are assumed known.

The general non-unitary JBD problem consists of estimating the matrix \( \mathbf{A} \) and the block-diagonal matrices set \( \mathcal{D} \) from only the matrix set \( \mathcal{M} \). Due to the matrices factorization, a rather classical way to solve the above JBD problem consists of minimizing the following quadratic cost function:

\[
\mathcal{F}_{\text{JBD}}(\mathbf{A}, \{\mathbf{D}_i\}) = \sum_{i=1}^{N_m} \| \mathbf{M}_i - \mathbf{A}_i \mathbf{D}_i \mathbf{A}_i^H \|_F^2,
\]

where \( \| \cdot \|_F \) stands for the Frobenius norm.

However, one can proceed rather differently. By premultiplying \( \mathbf{M}_i \) in Eq. (1) by the pseudo-inverse (Moore–Penrose generalized matrix inverse) \( \mathbf{A}^+ \) of \( \mathbf{A} \) and by post-multiplying by \( (\mathbf{A}^+)^H \), one has

\[
\mathbf{A}^+ \mathbf{M}_i (\mathbf{A}^+)^H = \mathbf{D}_i, \quad \forall i \in \{1, \ldots, N_m\}.
\]

To estimate the pseudo-inverse of \( \mathbf{A} \) (denoted by \( \mathbf{B} \)) instead of \( \mathbf{A} \) itself, it becomes rather advantageous to consider the following quadratic cost function:

\[
\mathcal{C}_{\text{JBD}}(\mathbf{B}, \{\mathbf{D}_i\}) = \sum_{i=1}^{N_m} \| \mathbf{B} \mathbf{M}_i \mathbf{B}^H - \mathbf{D}_i \|_F^2,
\]

since its minimization with respect to \( \mathbf{D}_i \), for all \( i \in \{1, \ldots, N_m\} \) is straightforward if \( \mathbf{B} \) is fixed. But first, we have to introduce some useful notations. If we consider a
square \((N \times N)\) matrix \(\mathbf{M} = (M_{ij})\) such that

\[
\mathbf{M} = \begin{pmatrix}
M_{11} & M_{12} & \cdots & M_{1r} \\
M_{21} & M_{22} & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
M_{r1} & M_{r2} & \cdots & M_{rr}
\end{pmatrix},
\]

(5)

where \(M_{ij}\) for all \(i,j=1, \ldots, r\) are \((n_i \times n_j)\) matrices (and \(n_1 + \cdots + n_r = N\)) and the following vector \(\mathbf{n} = (n_1, n_2, \ldots, n_r)\), two matrix operators \(\text{Bdiag}_{\mathbf{n}}(\cdot)\) and \(\text{OffBdiag}_{\mathbf{n}}(\cdot)\) can be respectively defined as

\[
\text{Bdiag}_{\mathbf{n}}(\mathbf{M}) = \begin{pmatrix}
M_{11} & 0_{12} & \cdots & 0_{1r} \\
0_{21} & M_{22} & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
0_{r1} & 0_{r2} & \cdots & M_{rr}
\end{pmatrix}, \quad \text{OffBdiag}_{\mathbf{n}}(\mathbf{M}) = \mathbf{M} - \text{Bdiag}_{\mathbf{n}}(\mathbf{M}),
\]

(6)

Denoting by \(\hat{\mathbf{D}}_1 = \arg \min_{\mathbf{D}_1} C_{BD}(\mathbf{B}, \{\mathbf{D}_1\})\), one easily finds

\[
\hat{\mathbf{D}}_1 = \text{Bdiag}_n(\mathbf{BB}^H), \quad \forall i = 1, \ldots, N_m.
\]

(7)

Using Eq. (6), it finally leads to

\[
C_{BD}(\mathbf{B}, \{\hat{\mathbf{D}}_1\}) = \sum_{i=1}^{N_m} \|\text{OffBdiag}_{\mathbf{n}}(\mathbf{BB}^H)\|_F^2 = C_{BD}(\mathbf{B}).
\]

(8)

In the literature, one can find several solutions to the JBD problem relying on the optimization of this cost function \((8)\) among which \([4, 12, 16, 17]\). Some of them \([4, 12]\) are dedicated to the case of a unitary matrix \(\mathbf{B}\) and have led to Jacobi-like algorithms. When these algorithms are applied in the blind sources separation context, the unitary constraint can be fulfilled by a classical whitening of the observations, but this preliminary stage has been proven to establish a bound with regard to the best reachable performances. That is why recent efforts have been put into the development of solutions like in \([16, 17]\) which no more rely upon this unitary constraint.

In \([17]\), positive definite and hermitian block diagonal matrices are considered. The criterion \((8)\) is then rewritten to perform a line by line iterative algebraic optimization involving an eigenvalues decomposition. In \([16]\), the considered matrices only have to be hermitian ones and the proposed solution is based on a gradient approach, yet, an approximation in the calculation is made to derive the expression of the complex gradient matrix of the cost function defined in Eq. \((8)\).

Notice that the non-unitary joint block diagonalization problem encompasses the non-unitary joint diagonalization one. In this latter case, \(\mathbf{D}_{ij}, \; i \in \{1, \ldots, N_m\}, \; j \in \{1, \ldots, r\}\) are no more matrices but scalars \((n_{ij} = 1\) for all \(j \in \{1, \ldots, r\}\) and consequently \(r = N\)). The zero block-diagonality operator is simplified: \(\text{OffBdiag}_{\mathbf{j}}(\mathbf{M}) = ((1 - \delta_{ij})M_{ij})1_N = \text{OffDiag}(\mathbf{M})\), with \(\mathbf{M} = (M_{ij})\), \(1_N\) is an \((N \times N)\) matrix whose elements are all equal to 1 and \(\delta_{ij} = 1\) if \(i = j\) and 0 otherwise. The matrix operator \(\text{Bdiag}_{\mathbf{j}}(\cdot)\) is then simply denoted by \(\text{Diag}(:,:,1)\):

\[
\text{Bdiag}_{\mathbf{j}}(\mathbf{M}) = ((\delta_{ij})M_{ij})1_N = \text{Diag}(\mathbf{M}) = \mathbf{M} - \text{OffDiag}(\mathbf{M}).
\]

(9)

It leads to the minimization of the cost function:

\[
C_{BD}(\mathbf{B}) = \sum_{k=1}^{N_m} \|\text{OffDiag}(\mathbf{BB}^H)\|_F^2,
\]

(9)

which has been used in \([7]\) under the unitary constraint (i.e. \(M = N\) and \(\mathbf{BB}^H = \mathbf{B}^H = 1_N\), with \(1_N\) the \((N \times N)\) identity matrix). Its minimization can then be achieved through a generalization of the Jacobi method \([19]\), \(C_{BD}(\mathbf{B})\) has also been considered in \([10, 13, 23, 27, 35, 37]\) for not necessarily unitary matrix \(\mathbf{B}\).

3. Two new joint block-diagonalization algorithms based on gradient approaches

To estimate the matrix \(\mathbf{B} \in \mathbb{C}^{N \times M}\), the cost function \(C_{BD}\) given in Eq. \((8)\) has to be minimized. To that aim, we propose, here, to use gradient-based algorithms (classical gradient and relative gradient). For each algorithm, we also provide an alternative algorithm in which the stepsize is no more fixed but computed algebraically at each iteration.

3.1. Principle of the fixed stepsize gradient-based methods

First, we consider a classical gradient descent algorithm. It is written as \(\Delta \mathbf{B} = -\mu_k \nabla a_{C_{BD}}(\mathbf{B})\) since \(C_{BD}(\mathbf{B})\) has to be minimized versus \(\mathbf{B}\). \(\mu_k\) is a positive small enough number called the stepsize or adaptation coefficient. The complex gradient matrix \(\nabla a_{C_{BD}}(\mathbf{B})\) of the real-valued scalar cost function given Eq. \((8)\) is defined, as (see \([22]\)):

\[
\nabla a_{C_{BD}}(\mathbf{B}) = 2 \frac{\partial C_{BD}(\mathbf{B})}{\partial \mathbf{B}^H},
\]

(10)

where \(\mathbf{B}^H\) stands for the complex conjugate of the complex matrix \(\mathbf{B}\) and \(\partial\) is the partial derivative operator. We also consider a relative gradient approach like in \([8]\) written as \(\Delta \mathbf{B} = -\mu_k \nabla a_{C_{BD}}(\mathbf{B})\mathbf{B}\) with \(\mu_k\) a positive small enough number and \(\nabla a_{C_{BD}}(\cdot)\) defined as

\[
\nabla a_{C_{BD}}(\mathbf{B}) = 2 \frac{\partial C_{BD}(\mathbf{B})}{\partial \mathbf{B}^H} \mathbf{B}^H = \nabla a_{C_{BD}}(\mathbf{B})\mathbf{B}^H.
\]

(11)

The main interest of the relative gradient approach is that for small enough stepizes the invertibility of the matrix \(\mathbf{B}\) can be guaranteed which is not the case with the standard gradient algorithm, see e.g. \([35]\). In any case, the complex gradient matrix \(\nabla a_{C_{BD}}(\mathbf{B})\) has to be evaluated. Its calculation is performed in Appendix A where it is demonstrated that \(\nabla a_{C_{BD}}(\mathbf{B})\) equals:

\[
2 \sum_{i=1}^{N_m} \|\text{OffDiag}_{\mathbf{n}}(\mathbf{BM}_i\mathbf{B}^H)\mathbf{BM}_i^H + (\text{OffDiag}_{\mathbf{n}}(\mathbf{BM}_i\mathbf{B}^H))^H\mathbf{BM}_i\|_F^2.
\]

(12)

Subsequently, the two suggested gradient-based algorithms can be derived. In the classical gradient approach, \(\mathbf{B}\) is updated at each iteration \(k\) according to the following adaptation rule:

\[
\mathbf{B}^{(k)} = \mathbf{B}^{(k-1)} - \mu_k \nabla a_{C_{BD}}(\mathbf{B}^{(k-1)}), \quad \forall k = 1, 2, \ldots.
\]

(13)

In the following, the resulting algorithm will be denoted by JBDG.
In a relative gradient approach, \( B \) is updated at each iteration \( k \) (for all \( k = 1, 2, \ldots \)) according to the following scheme:

\[
B^{(k)} = B^{(k-1)} - \mu_k \nabla_v C_{BD}(B^{(k-1)}) (I_N - \mu_k \nabla_v C_{BD}(B^{(k-1)})) B^{(k-1)}
\]

(14)

In the following, the resulting algorithm will be denoted by JBD_{org}. It can be noticed that whereas these adaptation rules are not equivalent in the general case, they become identical in the particular case of a unitary matrix \( B \).

If gradient descent-algorithms remain one of the simplest approach to solve optimization problems, it is well-known that they reach a local minimum and when the cost function is not convex, the question of initial conditions is very important. A good initial guess is thus important in initializing the algorithm. With regard to the step size, its choice has to be performed carefully too. In fact a small step size will lead to a slow convergence, whereas a too large step size will prevent the algorithm from converging. A simple way to avoid such situations consists of calculating an optimal step size computed in an automatic way at each iteration [36] (it will be explained how to proceed in the next section).

Finally, regarding to the algorithmic complexity, the computational cost per iteration \( k \) approximatively amounts to \( 4N_m NM(M+N) + 2N^2 N_m \) operations\(^2\) (i.e. the cost \( \simeq o(N_m NM^2) \) if \( M \gg N \) or \( \simeq o(N_m N^3) \) in the square case \( M = N \)) for the classic gradient algorithm while it approximatively amounts to \( 4N_m NM(M+N) + 2N^2 N_m \) operations for the relative gradient algorithm (i.e. we find again that the cost \( \simeq o(N_m NM^2) \) if \( M \gg N \) or \( \simeq o(N_m N^3) \) in the square case \( M = N \)). To calculate the global complexity of the algorithm, this computational cost per iteration has to be multiplied by the total number of iterations \( N_i \) needed to reach convergence. In the case of a practical application, the computational time necessary to construct the set of \( N_m \) matrices should be counted too.

3.2. Alternative algorithms: seek of the optimal stepsize

The purpose of this section is to provide two alternative algorithms: to eliminate the difficult problem of the choice of the stepsize, while decreasing the total number of iterations \( N_i \) needed by the previous algorithms to reach convergence, it is possible to compute their optimal stepsizes \( \mu_{opt} \) at each iteration \( k \). When the gradient algorithm is considered, it implies the algebraical calculation of the following quantity \( C_{BD}(B^{(k)}) = C_{BD}(B^{(k-1)}) - \mu \nabla_v C_{BD}(B^{(k-1)}) \) and its minimization with respect to \( \mu \). For the relative gradient algorithm, \( C_{BD}(B^{(k)}) = C_{BD}(B^{(k-1)}) - \mu \nabla_v C_{BD}(B^{(k-1)}) (I_N - \mu \nabla_v C_{BD}(B^{(k-1)})) B^{(k-1)} \) has to be minimized with respect to \( \mu \).

As shown in Appendix B, the first quantity is a 4th-degree polynomial whose expression is given by (we opt to omitting the dependency upon the iteration \( k \) to simplify the different expressions)

\[
C_{BD}(B - \mu \nabla_v C_{BD}(B)) = a_0 + a_1 \mu + a_2 \mu^2 + a_3 \mu^3 + a_4 \mu^4
\]

(15)

where the five coefficients \( a_0, a_1, a_2, a_3 \) and \( a_4 \) are found equal to (see Appendix B)

\[
a_0 = \sum_{i=1}^{N_m} (\text{vec}(M_i))^H P_{BD}^i \text{vec}(M_i),
\]

(16)

\[
a_1 = -\sum_{i=1}^{N_m} (\text{vec}(M_i))^H (P_{BD}^i Q^H + (P_{BD}^i Q^H)^H) \text{vec}(M_i),
\]

(17)

\[
a_2 = \sum_{i=1}^{N_m} (\text{vec}(M_i))^H (P_{BD}^i R^H + (P_{BD}^i R^H)^H + Q^H T_{BD}^i) \text{vec}(M_i),
\]

(18)

\[
a_3 = -\sum_{i=1}^{N_m} (\text{vec}(M_i))^H (Q^H T_{BD}^i R^H + (Q^H T_{BD}^i R^H)^H) \text{vec}(M_i),
\]

(19)

\[
a_4 = \sum_{i=1}^{N_m} (\text{vec}(M_i))^H R_{BD}^i T_{BD}^i \text{vec}(M_i),
\]

(20)

with \( P = B^H \otimes B^H, Q = B^H \otimes (\nabla_v C_{BD}(B))^H + (\nabla_v C_{BD}(B))^H \otimes B^H \) and \( R = (\nabla_v C_{BD}(B))^H \otimes (\nabla_v C_{BD}(B))^H \otimes \) the Kronecker product [5] and (\( \otimes \)) the transpose operator. The vec-operator \( \text{vec}() \) applied on a matrix \( M \) stacks its columns into a column vector. The \( (N^2 \times N^2) \) transformation matrices \( T_{\text{Diag}} \) and \( T_{\text{Boff}} \) are defined as

\[
T_{\text{Diag}} = \text{diag}(\text{vec}(B_{\text{diag}}^0(1_N))),
\]

(21)

\[
T_{\text{Boff}} = I_{N^2} - T_{\text{Diag}} = T_{\text{Boff}}^T,
\]

(22)

where \( I_{N^2} \) is the \( (N^2 \times N^2) \) identity matrix and \( \text{diag}(a) \) is a square diagonal matrix whose diagonal elements are the elements of the vector \( a \).

The derivative with respect to \( \mu \) of the 4th-degree polynomial in Eq. (15) is given by

\[
\frac{dC_{BD}(B - \mu \nabla_v C_{BD}(B))}{d\mu} = 4a_4 \mu^3 + 3a_3 \mu^2 + 2a_2 \mu + a_1.
\]

(23)

The optimal stepsize \( \mu_{opt} \) corresponds, then, to the root of the third order polynomial defined in Eq. (23) leading to the minimum of the criterion given in Eq. (15).

Considering the relative gradient algorithm, we obtain

\[
C_{BD}(B - \mu \nabla_v C_{BD}(B)) = a_0 + a_1 \mu + a_2 \mu^2 + a_3 \mu^3 + a_4 \mu^4,
\]

(24)

\[
\frac{dC_{BD}(B - \mu \nabla_v C_{BD}(B))}{d\mu} = 4a_4 \mu^3 + 3a_3 \mu^2 + 2a_2 \mu + a_1.
\]

(25)

where the coefficients \( a_0, a_1, a_2, a_3, a_4 \) have the same expressions as those given Eq. (16)–(20) but replacing the matrix \( Q \) (Eq. (37)) by the matrix \( Q = B^H \otimes (\nabla_v C_{BD}(B))^H + (\nabla_v C_{BD}(B))^H \otimes B^H \) and the matrix \( R \) (Eq. (38)) by the matrix \( R = (\nabla_v C_{BD}(B))^H \otimes (\nabla_v C_{BD}(B))^H \).

In the following, the non-unitary joint block-diagonalization algorithm based on an optimal stepsize relative gradient approach is denoted by JBD_{org} and the one based on an optimal stepsize gradient approach by JBD_{org}.

As it will be observed in the simulation Section 3.4, in one hand the optimal stepsize can greatly reduce the
iteration times, however, in the other hand, the computational time for one iteration is obviously increased. 

The complexity to compute the Kronecker product involved in the calculation of the matrix \(P = B^t \otimes B^l\), for example, is assumed \(o(N^3M^2)\) (it is also the case for matrices \(Q\) and \(R\)). As a consequence, the computational cost of both algorithms is no more ruled by the calculation of the complex gradient matrix but by the calculation of the five coefficients \(a_4, a_3, a_2, a_1\) and \(d_0\) of the fourth-order polynomial (the search for the roots is not too computational demanding since a direct analytical solution can be used). As a consequence, the computational cost per iteration \(k\) approximatively amounts to \(3N_m^2(N^2M^2 + N^2)\) operations (i.e. the cost \(\approx o(6N_m^2N^2M^2)\) if \(M \gg N\) or \(\approx o(9N_mN^4)\) in the square case \(M = N\)). Finally, a last remark can be done. The best way to deal with the issue of how to “blindly” choose the stepsize in an optimal way, while keeping a low computational cost is certainly to opt for what could be called “a hybrid approach” consisting of either calculating the optimal step size only for the first (say 10) iterations and then to keep it fixed or estimating this step size every (say) ten iterations and keeping it fixed during the other nine iterations.

3.3. Summary

The principle of the different algorithms is summarized in the following table (the optional steps (calculation of the optimal step size) appear in smaller size):

**Optimal stepsize. JBD algorithms based on (relative) gradient-descent approaches.**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBDG</td>
<td>On a gradient approach</td>
</tr>
<tr>
<td>JBDORG</td>
<td>Based on an optimal stepsize gradient approach</td>
</tr>
</tbody>
</table>
| JBDGORG   | On an optimal stepsize relative gradient approach Denote the \(N_m\) square matrices to be joint block-diagonalized as \(M_1, M_2, \ldots, M_{N_m}\). Given a sufficiently small value of the step size \(\mu\). Given an \((N \times M)\) initial matrix \(B^0\) (for example, in the square case \((N = M)\) one can choose \(B^0 = I_d\)). For \(k = 1, 2, \ldots\):
| A1       | Compute \(v_{B_{ij}}(B^{k-1})\) whose expression is given by Eq. (12).
| A2       | Compute \(v_{C_{ij}}(B^{k-1})\) whose expression is given by Eq. (11).
| A1’      | Compute the coefficients \(a_{4}^{k-1}, \ldots, a_{1}^{k-1}\) thanks to Eqs. (16)–(20).
| A2’      | Compute the coefficients \(a_{4}^{k-1}, \ldots, a_{1}^{k-1}\) thanks to Eqs. (16)–(20) replacing \(Q^{k-1}\) by \(B^{k-1}\) and \(B^{k-1}\) by \(B^{k-1}\).
| A1”      | Set the optimal stepsize \(\mu_{opt}^{k-1}\) by the research of the root of the polynomial given in Eq. (23) attaining the minimum in the polynomial given in Eq. (15).
| A2”      | Set the optimal stepsize \(\mu_{opt}^{k-1}\) by the research of the root of the polynomial given in Eq. (25) attaining the minimum in the polynomial given in Eq. (24).
| A1 and A1’| Set \(B^{k+1} = B^{k-1} - \mu_{opt} v_{C_{ij}}(B^{k-1})\).
| A2 and A2’| Set \(B^{k+1} = B^{k-1} - \mu_{opt} \sqrt{v_{C_{ij}}(B^{k-1})}\).
| A2 and A2’| Eventually normalize \(B^{k+1} = B^{k+1}/||B^{k+1}||_F\).
| A2 and A2’| Stop after a fixed number of iterations or when \(||B^{k+1} - B^{k-1}||_F < \varepsilon\) with \(\varepsilon\) a small positive threshold. |

3.4. Computer simulations: a comparison of the proposed JBD algorithms

Simulations are now provided to illustrate the behavior and the performances of the proposed JBD algorithms (JBDG, JBDORG, JBDG, JBDRG) but also to determine their robustness with respect to a deviation from the “block-diagonality assumption”. To that aim, a set \(\mathcal{D}\) of \(N_m = 100\) (or less) complex block-diagonal matrices \(D_i\) (for all \(i = 1, \ldots, N_m\), with random entries chosen from a Gaussian distribution with zero mean and unit variance is considered. Then, complex random entries chosen from a Gaussian distribution with zero mean and variance \(\sigma_0^2\) will be added on the off-diagonal blocks of the previous matrices \(D_i\), for all \(i = 1, \ldots, N_m\). A signal to noise ratio can be defined as \(\text{SNR} = 10\log(1/\sigma_0^2)\).

To establish a comparison between the different algorithms, we need a performance index. We have chosen to use \(\text{JBDORG}^i\) introduced in [17] and defined as

\[
\frac{1}{r(r-1)} \left[ \sum_{i=1}^{r} \left( \sum_{j=1}^{r} \frac{||G_{ij}||^2}{\max_{l=1}^M ||G_{il}||^2} - 1 \right) \right] + \sum_{j=1}^{r} \left( \sum_{i=1}^{r} \frac{||G_{ij}||^2}{\max_{l=1}^M ||G_{lj}||^2} - 1 \right)\]

where \(G_{ij}\) for all \(i, j \in \{1, \ldots, r\}\) is the \((i, j)\)–th (square) block matrix of \(G = BA\). This index will be used in the BSS section too that is why it has to take into account the inherent indeterminacy of the BSS problem (amplitude, permutation). The best results are obtained when the index performance \(\text{JBDORG}^i\) is found to be close to 0 in linear scale (\(\approx \infty\) in logarithmic scale). Regarding to the charts, \(\text{JBDORG}^i\) is given in dB and is then defined by \(\text{JBDORG}^i = 10\log(\text{JBDORG}^i)\)). In all the simulations, the mixing matrix \(A\) has been randomly chosen.

In Figs. 1–3, we consider the JBD problem in the square case, choosing \(M = N = 9\), \(r = 3\) and \(n_j = 3\) for all \(j = 1, \ldots, 3\). In Fig. 4, we consider the rectangular case since \(M = 12, N = 8, r = 2\) and \(n_j = 4\) for all \(j = 1, \ldots, 2\). And finally, in Figs. 5 and 6, we consider the JD problem with \(M = N = r = 3\) and \(n_j = 1\) for all \(j = 1, \ldots, 3\).

In Figs. 1 and 2, we present a very favorable context i.e. nearly noiseless (\(\text{SNR} = 100\) dB) and with many matrices (\(N_m = 100\)). The evolution of the performance index versus the number of iterations is displayed for fixed size versions of the algorithms (seven different values of the step size \(\mu\) are used) and for their optimal stepsize version. In Fig. 1, the results are obtained with the gradient algorithm (JBDG and JBDORG), whereas in Fig. 2 they are obtained under the same conditions (same set of matrices) but with the relative gradient algorithm (JBDRG and JBDORG). The highest value of the stepsize presented in these charts corresponds to the nearly highest value that does not lead to the divergence of the algorithm. One can observe that in the nearly noiseless case, the two algorithms behave quite similarly since they reach nearly the same performances (\(\approx -142\) dB). We also notice that the convergence is effectively quicker with the optimal stepsize version of the proposed algorithms since \(\approx 140\) iterations (resp. 350) are required to reach
the convergence in these examples against more than 1000 (resp. 2000) with the fixed stepsize versions that are presented here. Finally, at fixed stepsize, the convergence speed increases when the stepsize value increases which is classical too. In these examples, when the convergence is reached, the two versions of an algorithm, i.e. fixed stepsize and optimal stepsize, provide the same solution.

From now on, we will focus on the optimal stepsize versions of the two algorithms (but the same kind of results would be obtained with the fixed stepsize version under the constraint that the parameter $\mu$ is sufficiently small to ensure the convergence of the algorithm and that the number of iterations is sufficient to reach the convergence).

In Fig. 3, we have plotted the performance index versus the size of the matrix set for both algorithms. Different values of the SNR are studied (5, 20, 40 and 100 dB). The results have been averaged over 10 Monte-Carlo trials. The following observations can be made: the more matrices to be joint block-diagonalized, the better the obtained results are, the counterpart being that the computational cost increases too. The performances obviously increase when the SNR is higher. And finally, after convergence, the results obtained with the relative gradient-based algorithm are either equivalent to those obtained with the gradient algorithm (at high SNR) or better (at low SNR).

In Fig. 4, we focus on a difficult context ($\text{SNR} = 20 \text{ dB}$ and $N_m = 20$ matrices). If the JBD$_{DRO}$ algorithm outperforms the standard gradient algorithm JBD$_{DG}$ (has already observed in the previous figure), we notice that in counterpart, its convergence speed is slower. The presented results were obtained in the rectangular case, but the same conclusion would be made in the square case.

Finally, in Figs. 5 and 6, we study the behavior of the suggested JBD algorithms in the simpler context of the joint-diagonalization and compare then with four state-of-the-art JD algorithms: the generalized eigenvalue decomposition of two matrices (here, we have chosen matrices $M_1$ and $\sum_{i=1}^{N} M_i$), the JDA$_{a}$ algorithm suggested in [13] (and based on the algebraic optimization of a least mean square criterion), the DOMUNG (diagonalization of matrices using natural gradient) algorithm suggested in [35] (based on an optimal stepsize approximated natural gradient approach) and finally the LSB algorithm suggested in [10]. All the tested algorithms have been initialized with the same solution, for instance the one provided by the generalized eigenvalue decomposition. In Fig. 5, we give the performance index versus the SNR at a fixed size $N_m = 20$, whereas in Fig. 6, it is given versus the size of the matrix set to be joint-diagonalized at a fixed SNR $= 30 \text{ dB}$. The results have been averaged over 10 (resp. 20) Monte-Carlo trials. With regard to the general behavior of our algorithms: we obtain the same kind of results as the one already reached in the context of the joint block-diagonalization. Moreover, we can observe that our relative gradient-based algorithm is quite competitive since it outperforms most of the other algorithms (especially when $\text{SNR} \geq 30 \text{ dB}$).
Fig. 2. Performance index $I_{\text{conv}}(\cdot)$ versus the number of iterations for $\text{SNR} = 100$ dB and $N_m = 100$ matrices. A comparison of the optimal stepsize version with the fixed stepsize version of the relative gradient algorithm for different values of the stepsize. Dashed line: fixed stepsize, solid line: optimal stepsize.

Fig. 3. Evolution of the performance index $I_{\text{conv}}(\cdot)$ versus the number $N_m$ of used matrices for different values of the SNR (5, 20, 40 and 100 dB). Solid line: $\text{JBD}_\text{ORG}$, dotted line: $\text{JBD}_\text{OG}$. 
Fig. 4. Rectangular case. Comparison between the optimal stepsize version of the gradient and the relative gradient algorithms: performance index $I_{\text{conv}}(\cdot)$ versus the number of iteration considering SNR = 20 dB and $N_m = 20$ matrices. Dashed line: gradient, solid line: relative gradient.

Fig. 5. Joint diagonalization algorithms: performance index $I_{\text{conv}}(\cdot)$ versus the SNR for $N_m = 20$ matrices.
3.5. Discussion

These algorithms have the same advantages/drawbacks as all gradient based algorithms. The continual descent of the cost function over the iterations is not always guaranteed: if it is chosen constant, the stepsize has to be small enough, otherwise it has to be determined to ensure a non-increase of the cost function. For too high stepsizes, the algorithms may diverge. Like all gradient algorithms, these algorithms only converge (if so) to a local solution. That is why, the choice of the initial point remains an important issue. One possible way to better initialize is to consider the solution given by the orthogonal joint block-diagonalization \[4,12\] to start in the neighborhood of the solution. Yet, in the simulations that we have presented, the algorithms were always initialized with the identity matrix (except in Figs. 5,6).

The main advantage of these gradient-based algorithms remains their implementation simplicity. They also enable finding a joint block diagonalizer \(B\in \mathbb{C}^{N\times M}\) which is a general (not necessarily orthogonal) matrix, the matrices in \(M\) being not required to be positive definite like in \[17\].

4. An application example: blind separation of convolutive mixtures of non-stationary sources using SQTFD(or S)

In this section, our aim is to provide an application example of the proposed algorithms showing how they enable tackling the problem of the separation of convolutive FIR mixtures of non-stationary sources in a non-whitened context. We focus on BSS methods that are based on spatial time–frequency distributions applied to deterministic signals as well as non-stationary stochastic processes. These methods were first introduced in \[3\] in an instantaneous and whitened context and Amin and Belouchrani were also the first to bridge time–frequency \[15,21,30\] to array processing and blind source separation. In this kind of BSS methods, successful signal separations generally involve four main steps. First, the convolutive mixture is rewritten as an instantaneous one. Second, regions of signal power concentration and localization should be properly determined and time–frequency points of peaky values must be selected. Third, the set of time–frequency matrices of the observations across the array at these time–frequency points must be constructed. Fourth, the mixing matrix should be properly estimated so as to undo the mixing of signals at the multi-sensor receiver.

The methods that will now be introduced constitute a generalization to the convolutive case of the methods suggested in \[2,3,13,18\] for example (see also \[32\] for a more exhaustive review of all the existing methods) but also a generalization to the non-whitened case of the methods presented in \[4\] or \[14\].

4.1. Step 1: from a convolutive mixture to an instantaneous one

In the BSS problem, the goal is to recover multiple sources mixed through an unknown mixing system from the system outputs only (namely the observations). Here, we assume that we have \(m \in \mathbb{N}^*\) observation signals...
whose blocks $H_j(t)$ are the impulse response function between the $j$-th source and the $i$-th sensor with an overall extent of $L + 1$ taps and $n_i(t)$, for all $i = 1, \ldots, m$ are noises.

As suggested in [20], one possible way to deal with Eq. (26) is to reformulate the convolutive mixing model into an instantaneous one (other approaches operating in the frequency domain have been developed (one may refer the reader to [1,25,26,29,31] for example) but it is outside the scope of this article to deal with them). Denoting by $M = mL$, $N = n(L + L') = nQ$ (with $Q = L + L'$ and $L \in \mathbb{N}^*$), we consider the $(N \times 1)$ vector $S(t) = [s_1(t), s_2(t), \ldots, s_n(t)]^T$ and the $(M \times 1)$ vectors $X(t) = [x_1(t), x_2(t), \ldots, x_m(t)]^T$ and $N(t) = [n_1(t), n_2(t), \ldots, n_m(t)]^T$, where the $(Q \times 1)$ vectors $s_j(t)$ (for all $j = 1, \ldots, n$) read $s_j(t) = [s_j(t), s_j(t-1), \ldots, s_j(t-Q+1)]^T$, the $(L' \times 1)$ vectors $x_j(t)$ (for all $i = 1, \ldots, m$) read $x_j(t) = [x_j(t), x_j(t-1), \ldots, x_j(t-L'+1)]^T$ and the $(L' \times 1)$ vectors $n_j(t)$ (for all $i = 1, \ldots, m$) read $n_j(t) = [n_j(t), n_j(t-1), \ldots, n_j(t-L'+1)]^T$. The model described by Eq. (26) can be rewritten in the ensuing matrix form

$$X(t) = AS(t) + N(t),$$

(27)

where the $(M \times N)$ mixing matrix $A$ is a block-matrix defined as $A = (A_{ij})$ for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$ whose blocks $A_{ij}$ are $(L' \times Q)$ Toeplitz matrices:

$$A_{ij} = \begin{pmatrix} H_{ij}(0) & \cdots & H_{ij}(L) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & H_{ij}(0) & \cdots & H_{ij}(L) \end{pmatrix}.$$

(28)

To still deal with an over-determined model, $L'$ is chosen such that $M \geq N$. It is further assumed that:

**Assumption A.** The noises $n_j(t)$ for all $j = 1, \ldots, m$ are stationary, white, zero-mean random signals, mutually uncorrelated and independent from the sources.

**Assumption B.** For each of the $n$ sources $s_j$, there exist time–frequency points where only this source and its delayed versions are present in the time–frequency plane.

**Assumption C.** The number of sensors $m$ and the number of sources $n$ are both known and $m \geq n$ (i.e. the over-determined case is considered, the under-determined case being outside the scope of this article). The FIR filter order $L$ is known too.

Moreover, the matrix $A$ (involved in Eq. (27)) satisfies the Assumption $A_0$ (Section 3). Finally, it is important to notice that the assumption B is a key assumption since it replaces the classical “random independent sources” assumption and it is this assumption that makes it possible to tackle the non-stationary correlated sources separation problem.

### 4.2. Step 2: selection of “useful” time–frequency points

The observations spatial quadratic time–frequency spectrum (SQTFS) is across the array at a given $t-f$ point is a $(M \times M)$ matrix, admits the following decomposition (since the noises are centered and independent from the source signals (Assumption A):

$$D_X(t, v) = AD_S(t, v)A^H + D_N(t, v) + AD_{SN}(t, v) + D_{NS}(t, v)A^H$$

$$= AD_S(t, v)A^H + D_N(t, v),$$

(32)

where, respectively, $D_S(t, v)$ represents the $(N \times N)$ source SQTFS, $D_N(t, v)$ is the $(M \times M)$ noise SQTFS and finally $D_{SN}(t, v)$ and $D_{NS}(t, v)$ are the spatial bilinear time–frequency spectra (SBTFS) between the sources and the noises.

To remove the noise, the weak energy $t-f$ points are eliminated. Only those such that: $\text{min}(|D_{NS}(t, v)|^2) > \epsilon$, $i, j \in \{1, \ldots, M\}$, are kept, where $\epsilon$ is a (sufficiently) small constant (for example $\epsilon \approx 0.01$), min(-) is the minimum value and $D_{SN}(t, v)$, $D_{NS}(t, v)$ are recovered. It has to be noted too that $D_{SN}(t, v) = (D_X(t, v)\gamma_j)$. Thus, Eq. (32) finally reads

$$D_X(t, v) \approx AD_S(t, v)A^H.$$  

(33)

When $t-f$ points $(t, v)$ satisfying Assumption B are considered, the matrix $D_S(t, v)$ possesses a very specific algebraic structure since it is block-diagonal with one single non-null $(Q \times Q)$ block on the diagonal (the same property holds in the pre-whitened case and has already been used by the authors of [14]). This algebraic property is general since in the case of matrices stemming from spatial time–frequency distributions or spectra (SQTFS/STFT), the block-diagonal matrix with one single non-null block in the diagonal is the only possibility of block-diagonal matrix.

Non-stationary signal separation techniques based on SQTFS(SI) require a preliminary stage of selecting (in an automatic or manual way) particular $t-f$ points: those

---

3 We recall that considering two vectors, namely $X(t)$ and $Y(t)$, the STFTS is defined as

$$D_{XY}(t, v) \equiv \int_{-\infty}^{\infty} X(t, \tau) Y(t, \tau)^* d\tau.$$  

(29)

where the function $K(\theta; t, v)$ is the kernel of the transformation and $R_{XY}(t, \tau) = E[X(t + \tau/2, v)Y^*(t - \tau/2)]$ if $E[\cdot]$ is the mathematical expectation operator. The STFTS is a case of the STFTS with $Y(t) = X(t)$:

$$D_X(t, v) \equiv \int_{-\infty}^{\infty} R_X(t, \tau) d\tau.$$  

(30)

where $R_X(\theta, \tau) = E[X(t, \tau)^* X(t + \tau/2, v)]$ is the correlation matrix of signal $X(t)$. To illustrate, consider two important examples: first the spatial pseudo-Wigner spectrum (SPWS), whose expression is given by

$$D_{SPWS}(t, v) = \int_{-\infty}^{\infty} R_X(t, \tau) h(\gamma \exp(-2\pi i \nu v) d\tau,$$

(31)

where $\gamma$ stands for the smoothing (short-time) window (rectangular, Kaiser–Bessel, Hamming, Bartlett, etc.). When deterministic instead of random signals are considered, one simply has to discard the mathematical expectation in Eq. (31) and the spatial pseudo-Wigner spectrum (SPWS), $D_{SPWS}(t, v)$, is recovered. It has to be noted too that the spatial Wigner spectrum (SW), $D_{SW}(t, v)$, is obtained by considering $h(t) = 1$ for all $t$ in Eq. (31), while the spatial Wigner distribution (SWD), $D_{SWD}(t, v)$, is recovered by discarding the mathematical expectation.
whose existence is ensured from Assumption B. This step is important to construct the set of matrices for joint block-diagonalization (this set will be denoted by $M_{\text{JBD}}$). All the matrices belonging to $M_{\text{JBD}}$ admit the decomposition given by Eq. (33) (or equally Eq. (1)) where the source SQTFD(orS) matrices $D_S(t, v)$ assume a specific algebraic structure, being block diagonal matrices. As a consequence, a natural manner to tackle the separation problem will consist of opting for a JBD algorithm (see the Step 4 detailed Section 4.4).

4.3. Step 3: construction of $M_{\text{JBD}}$ (set of t–f matrices of the observations across the array at the chosen t–f points)

Due to our assumptions, $D_X(t, v)$ and $D_S(t, v)$ have the same rank. The aforementioned comment suggests to select the t–f matrices whose rank is equal to Q. A singular value decomposition can be used (SVD) leading to $D_X(t, v) = U(t, v)\Lambda(t, v)V^H(t, v)$ with $V(t, v)$ and $U(t, v)$ are $(M \times M)$ unitary matrices and $\Lambda(t, v) = \text{diag}(\lambda(t, v))$ is a diagonal matrix whose diagonal elements are positive. Denoting by $\lambda(t, v) = (\lambda_1(t, v), \ldots, \lambda_M(t, v))^T$ and assuming that the singular values are sorted in a decreasing order: $\lambda_1(t, v) \geq \lambda_2(t, v) \geq \ldots \geq \lambda_M(t, v) \geq 0$, then one way to check whether a matrix is rank Q or not is given by:

For $M_{\text{JBD}}$ choose SQTFD(or S) corresponding to t–f points $(t, v)$ such that

$$R(t, v) = \frac{\sum_{i=1}^{Q} \lambda_i^2(t, v)}{\|D_X(t, v)\|^2} \geq 1 - \varepsilon, \quad (34)$$

where $\varepsilon$ is a (sufficiently) small positive constant.

This detection procedure is denoted by $C_{\text{conv}}$ and can be seen either as a generalization of the detector proposed in [18] in the instantaneous context for non-whitened signals (it was based on a rank one property) or as a generalization to non-whitened signals of the detector suggested in [14] for the convolutive case but for whitened observations.

4.4. Step 4: non-whitened based separation algorithm

The matrices belonging to the set $M_{\text{JBD}}$ (whose size is denoted by $N_m$ ($N_m \in N^+$)) all admit a particular structure since they can be decomposed into $A D_S(t, v)A^H$ with $D_S(t, v)$ a block-diagonal matrix with only one non-null ($Q \times Q$) block on its diagonal. One possible way to recover the mixing matrix $A$ (or its pseudo-inverse: the separation matrix $B$) is to directly joint block diagonalize the matrix set $M_{\text{JBD}}$. It has to be noticed that the sources recovered after inversion of the system are obtained up to a permutation and up to a filter which are the classical indeterminations of the BSS in the convolutive case.

Two BSS methods can be derived: the first one is called $\text{JBD}_{\text{ORGT}}$ since it combines the JBD algorithm based on a gradient approach $\text{JBD}_{\text{ORG}}$ together with the automatic time–frequency points detector $C_{\text{conv}}$. The second method called $\text{JBD}_{\text{ORGTF}}$ consists of replacing the gradient-based JBD algorithm by the relative gradient algorithm $\text{JBD}_{\text{OGRG}}$.

4.5. Computer simulations

Computer simulations are performed to illustrate the good performance of the two suggested methods and to compare them with the same kind of existing approaches: the first one is denoted by $\text{UJB}D_{\text{TF}}$ and combines the unitary JBD algorithm proposed in [14] and the t–f point detector $C_{\text{conv}}$. The second one is denoted by $\text{JBD}_{\text{ATF}}$ and combines the non-unitary JBD algorithm proposed in [17] with the t–f point detector $C_{\text{conv}}$.

As a consequence, the $\text{JBD}_{\text{ATF}}, \text{JBD}_{\text{ORG}}$ and $\text{JBD}_{\text{ORGTF}}$ methods operate on the same set of t–f matrices $M_{\text{JBD}}$, whereas the $\text{UJB}D_{\text{TF}}$ method is applied after a pre-whitening stage of the selected t–f matrices. We consider $n = 3$ mixtures of $n = 2$ sources of 128 time samples. The first source (resp. the second source) is a linear frequency modulation (resp. a sinusoidal frequency modulation), $L = 2$ and $L = 4$. These sources are mixed according to a mixture matrix $A(t)$ whose components are randomly generated and whose z-transform $A(z)$ is given by

$$A(z) = \begin{pmatrix} -0.3592 + 0.4104z^{-1} + 0.8382z^{-2} & 0.9365 - 0.3316z^{-1} + 0.1138z^{-2} \\ 0.5585 - 0.1182z^{-1} + 0.8221z^{-2} & 0.4876 - 0.3044z^{-1} + 0.8183z^{-2} \\ 0.4550 - 0.7171z^{-1} + 0.5279z^{-2} & -0.8218 + 0.1528z^{-1} + 0.5489z^{-2} \end{pmatrix}.$$ 

We use the SPWVS with a Hamming smoothing window of size 32 and 64 frequency bins (in this example, 100 realizations of signals are computed. The spatial pseudo-Wigner distribution of each realization is calculated and finally the 100 resulting SPWVD are averaged). We have chosen $\varepsilon = 0.01$ for the detector $C_{\text{conv}}$. In Fig. 7, we have plotted the performance index versus the SNR ($\approx 1600$ time–frequency matrices were selected when $\text{SNR} \geq 20$ dB. For $\text{SNR} \leq 20$ dB, the more the SNR decreases, the more this number decreases too). For the noises, we use random entries chosen from a Gaussian distribution with zero mean and variance $\sigma_n^2$ and the same definition of the SNR as the one previously used in Section 3.

The resulting performance index shows that the $\text{JBD}_{\text{ORGT}}$ and $\text{JBD}_{\text{ORGTF}}$ methods exhibit the best performances, whereas the method based on a unitary JBD algorithm exhibits the worst performances. In a noisy environment, it is the $\text{JBD}_{\text{ORGTF}}$ method that provides the best results.

5. Discussion and conclusion

In this article, we have proposed two new joint block diagonalization algorithms (namely $\text{JBD}_{\text{ORG}}$ and $\text{JBD}_{\text{ORGTF}}$). One is based on a gradient approach, the other relies on a relative gradient approach. They do not rely on restrictive assumptions about the matrix set $M_{\text{JBD}}$ on which they operate and the joint block diagonalizer that is found can be either a unitary or a non-unitary matrix. Since the joint block-diagonalization problem encompasses the joint diagonalization one, we have also suggested a new JD algorithm (based on a(n) (optimal stepsize) relative gradient approach) which exhibits good...
performances. For each algorithm, an alternative solution has been provided in which the stepsize is no more fixed but computed algebraically at each iteration. The best performances after convergence even in a difficult context (noisy case and very few matrices) are reached with the JBD\textsubscript{O,JG} algorithm while the best convergence speed is obtained with the JBD\textsubscript{O,GT} algorithm. The usefulness of the proposed JBD algorithms is emphasized in the BSS context. Combined with a \(t\)-\(f\) points detector operating in a non-whitened context (namely \(C_{\text{Conv}}\)), these algorithms allow to obtain two alternative time–frequency based BSS methods (denoted by JBD\textsubscript{O,GT} and JBD\textsubscript{O,GTJ}) devoted to the separation of deterministic signals or mixtures of (correlated) non-stationary stochastic processes.

Appendix A

Considering three \((M \times M)\) square matrices \(D_1\), \(D_2\), and \(D_3\) and two rectangular matrices \(D_4\) \((M \times N)\) and \(D_5\) \((N \times M)\) and a square \(N \times N\) matrix \(D_6\), let \(\{\cdot\}, \{\cdot\}^t, \text{vec} \{\cdot\}\), \(\text{OffBdiag}_{\{\cdot\}}\) and \(T_{\text{off}}\) respectively denote the trace operator, the differential operator, the vec-operator, the zero-block-diagonal operator defined in Eq. (6) and the \((N^2 \times N^2)\) “transformation” matrix defined in Eq. (22). Our developments are based on the ensuing properties [22,23]:

\[
P_1. \quad \|\text{OffBdiag}_{\{\cdot\}}(D_1)\|^2 = \text{tr}(\text{OffBdiag}_{\{\cdot\}}(D_1)\text{OffBdiag}^\dagger_{\{\cdot\}}(D_1)) = \text{tr}(D_1^\dagger \text{OffBdiag}_{\{\cdot\}}(D_1)).
\]

\[
P_2. \quad \text{tr}(D_1) = \text{tr}(D_1^t).
\]

\[
P_3. \quad \text{tr}(D_1 + D_2) = \text{tr}(D_1) + \text{tr}(D_2).
\]

\[
P_4. \quad \text{tr}(D_1 D_2) = \text{tr}(D_2 D_1) = \text{tr}(D_1 D_2) \Rightarrow \text{tr}(D_1 D_2) = \text{tr}(D_2 D_1) = \text{tr}(D_1 D_2).
\]

\[
P_5. \quad \text{tr}(D_1 D_2) = \text{tr}(D_2 D_1) = \text{tr}(D_2 D_1) = \text{tr}(D_2 D_1) \Rightarrow \text{tr}(D_1 D_2) = \text{tr}(D_2 D_1) = \text{tr}(D_2 D_1) = \text{tr}(D_2 D_1).
\]

\[
P_6. \quad \text{vec}(\text{OffBdiag}_{\{\cdot\}}(D_6)) = T_{\text{off}} \text{vec}(D_6).
\]

\[
P_7. \quad d(D_1^t) = (dD_1)^t.
\]

\[
P_8. \quad d(D_1^t) = (dD_1)^t.
\]

\[
P_9. \quad d(D_1 D_2) = d(D_1) D_2 + D_1 d(D_2).
\]

\[
P_{10}. \quad d(D_1 + D_2) = d(D_1) + d(D_2).
\]

\[
P_{11}. \quad d(\{D_1\}) = d(\{D_1\}).
\]

\[
P_{12}. \quad d(\text{vec}(D_1)) = \text{vec}(d(D_1)).
\]

\[
P_{13}. \quad \frac{df(Z, Z')}{dz} = \text{tr}(D_1^t Z + Z'^t D_2^t) \Rightarrow d(f(Z, Z')) = \text{tr}(D_1^t dZ + D_2^t dZ') \Rightarrow \frac{df(Z, Z')}{dz} = D_2.
\]

\[
P_{14}. \quad \text{vec}(D_1^t D_2 D_1) = D_2^t \otimes D_1 \text{vec}(D_2).
\]

\[
P_{15}. \quad (D_1 \otimes D_2)^t = D_2^t \otimes D_1^t.
\]

A.1. Calculation of the complex gradient matrix of the cost function \(C_{\text{BD}}(B)\)

Using the property \(P_1\), the cost function \(C_{\text{BD}}(B)\) can be expressed as:

\[
C_{\text{BD}}(B) = \sum_{i=1}^{N_B} \text{tr}(\{B M B^H\}^t \text{OffBdiag}_{\{\cdot\}}(B M B^H)).
\]

From \(P_1, P_9, P_{10}\) and \(P_{11}\), the cost function differential is rewritten as:

\[
d(C_{\text{BD}}(B)) = \sum_{i=1}^{N_B} \text{tr}(d(\{B M B^H\}^t \text{OffBdiag}_{\{\cdot\}}(B M B^H))).
\]

\[
= \mathcal{F}(B) + \tilde{g}(B),
\]

where

\[
\mathcal{F}(B) = \sum_{i=1}^{N_B} \text{tr}(\{d(B M B^H)\}^t \text{OffBdiag}_{\{\cdot\}}(B M B^H)),
\]

\[
\tilde{g}(B) = \sum_{i=1}^{N_B} \text{tr}(d(\{B M B^H\}^t) \text{OffBdiag}_{\{\cdot\}}(B M B^H)).
\]
\[ \mathcal{G}(B) = \sum_{i=1}^{N_n} \text{tr}((BM_iB_i^{H})^Hd(\text{OffBdiag}_{in}(BM_iB_i^{H}))). \]

The properties \( P_2, P_3, P_4, P_4, P_7, P_9, P_8 \) and \( P_{12} \) imply that

\[ \mathcal{F}(B) = \sum_{i=1}^{N_n} \text{tr}(BM_i^Hd(B_i^{H})\text{OffBdiag}_{in}(BM_iB_i^{H})) \]
\[ + \sum_{i=1}^{N_n} \text{tr}(d(B_i)BM_i^H\text{OffBdiag}_{in}(BM_iB_i^{H})). \]
\[ = \sum_{i=1}^{N_n} \text{tr}((\text{OffBdiag}_{in}(BM_iB_i^{H}))BM_i^Hd(B_i)) \]
\[ + \sum_{i=1}^{N_n} \text{tr}(((\text{OffBdiag}_{in}(BM_iB_i^{H}))B_i^TM_i^T)Td(B_i)). \]

While using the properties \( P_2, P_3, P_4, P_4, P_5, P_6, P_8 \) and \( P_9 \), we show that

\[ \mathcal{G}(B) = \sum_{i=1}^{N_n} \text{vec}(BM_iB_i^{H})^Hd(\text{OffBdiag}_{in}(BM_iB_i^{H})). \]
\[ = \sum_{i=1}^{N_n} (\text{vec}(\text{OffBdiag}_{in}(BM_iB_i^{H})))^Hd(\text{vec}(BM_iB_i^{H})). \]
\[ = \sum_{i=1}^{N_n} \text{tr}((\text{OffBdiag}_{in}(BM_iB_i^{H}))^Hd(B_i)). \]
\[ = \sum_{i=1}^{N_n} \text{tr}(((\text{OffBdiag}_{in}(BM_iB_i^{H}))B_i^TM_i^T)Td(B_i)). \]
\[ + \text{tr}(((\text{OffBdiag}_{in}(BM_iB_i^{H}))B_i^TM_i^T)Td(B_i)). \]

We replace \( \mathcal{F}(B) \) and \( \mathcal{G}(B) \) in Eq. (35) to finally find that

\[ d(C_{BD}(B)) = \sum_{i=1}^{N_n} \text{tr}(((\text{OffBdiag}_{in}(BM_iB_i^{H}))^HB_i^TM_i^T)T) \]
\[ + ((\text{OffBdiag}_{in}(BM_iB_i^{H}))^HB_i^TM_i^T)Td(B_i)) \]
\[ + \text{tr}(((\text{OffBdiag}_{in}(BM_iB_i^{H}))B_i^TM_i^T)Td(B_i)). \]

Using the properties \( P_{13} \), we obtain the following results

\[ \frac{dC_{BD}(B)}{dB} = \sum_{i=1}^{N_n} (\text{OffBdiag}_{in}(BM_iB_i^{H}))^TB_i^TM_i^T \]
\[ + \sum_{i=1}^{N_n} (\text{OffBdiag}_{in}(BM_iB_i^{H}))^TB_i^TM_i^T. \]
\[ \frac{dC_{BD}(B)}{dB} = \sum_{i=1}^{N_n} (\text{OffBdiag}_{in}(BM_iB_i^{H}))BM_i^H \]
\[ + \sum_{i=1}^{N_n} (\text{OffBdiag}_{in}(BM_iB_i^{H}))BM_i^H. \]

It finally leads to the result stated by Eq. (12).

A.2. Coefficients of the 4th-degree polynomial

To simplify, the dependency over the iteration is omitted. Using the properties \( P_4 \) and \( P_5 \), the cost function \( C_{BD}(B) \) is expressed as

\[ C_{BD}(B) = \sum_{i=1}^{N_n} (\text{vec}(BM_iB_i^{H}))^H\text{vec}(\text{OffBdiag}_{in}(BM_iB_i^{H})). \]

From the properties \( P_{6}, P_{14} \) and \( P_{15} \), we find that

\[ C_{BD}(B) = \sum_{i=1}^{N_n} (\text{vec}(M_i^H))B_i^H \otimes \text{Bdiag}_{in}(B_i^H)\text{Bvec}(M_i). \]

Introducing the three following matrices \( P, Q \) and \( R \):

\[ P = B_i^T \otimes B_i^H, \]
\[ Q = B_i^T \otimes (\nabla_a C_{BD}(B))B_i^H + (\nabla_a C_{BD}(B))B_i^T \otimes B_i^H, \]
\[ R = (\nabla_a C_{BD}(B))^T \otimes (\nabla_a C_{BD}(B))B_i^H, \]

we finally find

\[ C_{BD}(B) = \mu \nabla_a C_{BD}(B) \]
\[ = \sum_{i=1}^{N_n} \text{vec}(M_i^H)(P - \mu Q + \mu^2 R)B_i^H + \mu Q^H \text{vec}(M_i) \]
\[ = \sum_{i=1}^{N_n} \text{vec}(M_i^H)PT_{B_i}P_i \text{vec}(M_i) \]
\[ - \mu^2 \sum_{i=1}^{N_n} \text{vec}(M_i^H)(P_iQ_i^H + QT_{B_i}P_i^H) \text{vec}(M_i) \]
\[ + \mu^2 \sum_{i=1}^{N_n} \text{vec}(M_i^H)(P_iR_i^H + QT_{B_i}R_i^H) \text{vec}(M_i) \]
\[ = \mu^4 \sum_{i=1}^{N_n} \text{vec}(M_i^H)RT_{B_i}R_i^H \text{vec}(M_i) \]
\[ = a_0 + a_1 \mu + a_2 \mu^2 + a_3 \mu^3 + a_4 \mu^4. \]

It finally leads to the results stated by Eqs. (16)–(20).

References