EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Orthogonal approach to independent component analysis using quaternionic factorization

Adam Borowicz

Correspondence:
a.borowicz@pb.edu.pl
Faculty of Computer Science, Bialystok University of Technology, Wiejska 45A, 15-351 Bialystok, Poland

Abstract

Independent component analysis (ICA) is a popular technique for demixing multichannel data. The performance of a typical ICA algorithm strongly depends on the presence of additive noise, the actual distribution of source signals, and the estimated number of non-Gaussian components. Often, a linear mixing model is assumed and source signals are extracted by data whitening followed by a sequence of plane (Jacobi) rotations. In this article, we develop a novel algorithm, based on the quaternionic factorization of rotation matrices and the Newton-Raphson iterative scheme. Unlike conventional rotational techniques such as the JADE algorithm, our method exploits 4×4 rotation matrices and uses approximate negentropy as a contrast function. Consequently, the proposed method can be adjusted to a given data distribution (e.g., super-Gaussians) by selecting a suitable non-linear function that approximates the negentropy. Compared to the widely used, the symmetric FastICA algorithm, the proposed method does not require an orthogonalization step and is more accurate in the presence of multiple Gaussian sources.

Keywords: BSS, ICA, Jacobi rotations, Negentropy, Quaternions

1 Introduction

Blind source separation (BSS) problems occur in various engineering applications, including the multichannel speech enhancement, image restoration, analysis of electroencephalographic (EEG) signals, and telecommunications [1]. The BBS consists of recovering of unobservable source signals from their observed mixtures. Most often, the following linear model [2] of the observation vector is assumed:

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

where $\mathbf{A} \in \mathbb{R}^{n \times d}$ is an unknown mixing matrix and $\mathbf{s}$ is a random vector of d unknown source signals. Please note that there exist other models that can deal, for example, with spherical noise [3], under-determination, and mixture synchronization issues. In the under-determined case, the number of source signals is greater than the number of mixtures. Obviously, it makes the BSS problem ill-posed. In such cases, a concept of sparsity

is often employed for a better representation of source signals [4, 5]. To tackle the synchronization issues, a compressively sensed BSS (CS-BSS) model [6] was proposed which combines a compressive sensing theory [7] with a linear mixing model.

In this work, we consider the conventional model (1) due to its simplicity, and unless otherwise stated, assume that $n = d$. Thus, from a mathematical point of view, we are looking for some matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{y} = \mathbf{W}\mathbf{x} = \mathbf{\Lambda}\mathbf{P}\mathbf{s}, \tag{2}$$

where $\mathbf{\Lambda}$ and $\mathbf{P}$ are scaling and permutation matrices, respectively. This means that the BSS problem can be solved only up to sign, scale, and permutation of the output signals [8].

Independent component analysis (ICA) is the most natural approach to solving the BBS problem. Namely, the ICA aims at transforming multivariate data into components that are as statistically independent from each other as possible. The ICA can be also viewed as an optimization method that explicitly maximizes some measure of statistical independence between the sources. Most frequently, it is based on the Central Limit Theorem and attempts to find directions in which some measure of non-Gaussianity is maximized [9]. Theoretically, it is possible to extract all components as long as no more than one source is Gaussian and all of the non-Gaussian sources are mutually independent. In other words, only non-Gaussian components can be separated.

In fact, many ICA algorithms are known. For general overviews, see for example [1, 10–12]. The classical ICA methods include the InfoMax [13, 14], FastICA [9], and JADE [15]. In this study, we are interested in the so-called orthogonal approaches. Since independence implies uncorrelatedness, these methods directly constrain the unmixed components to be uncorrelated, i.e., for zero-mean and unit-variance signals, it is assumed that $E\{\mathbf{y}\mathbf{y}^T\} = \mathbf{I}$. This constraint is usually satisfied by whitening data before rotating them [9, 16, 17]:

$$\mathbf{y} = \mathbf{B}\mathbf{C}_{\mathbf{xx}}^{-1/2}\mathbf{x}, \tag{3}$$

where the matrix $\mathbf{B}$ is constrained to be orthogonal. The whitening transformation can be computed as follows:

$$\mathbf{C}_{\mathbf{xx}}^{-1/2} = \mathbf{U}\mathbf{\Sigma}^{-1/2}\mathbf{U}^T, \tag{4}$$

where $\mathbf{U}$ is the matrix of eigenvectors of $\mathbf{C}_{\mathbf{xx}} = E\{\mathbf{x}\mathbf{x}^T\}$, and $\mathbf{\Sigma}$ is the diagonal matrix of the corresponding eigenvalues.

Among the orthogonal approaches, the FastICA method [9] and their variants [18] are probably the most popular techniques. They are based on maximizing some non-linear contrast functions that approximate negentropy. It was shown in [8] that finding a demixing matrix $\mathbf{W}$ that minimizes the mutual information [13] is roughly equivalent to finding directions in which the negentropy is maximized. Therefore, such contrast functions can also be viewed as the measures of non-Gaussianity. There are two basic variants of the FastICA method: a deflation-based and symmetric algorithm. In the case of the deflation approach, the independent components are estimated successively one by one and after every iteration step, the Gram-Schmidt orthogonalization is performed. The major drawback of this approach is that estimation errors of the first vectors are cumulated in the subsequent ones by the orthogonalization [19]. The symmetric variant of the FastICA

algorithm [20] estimates the components in parallel. This consists in computing the one-unit updates for each component, followed by subsequent symmetric orthogonalization of the estimated demixing matrix after each iteration step.

Symmetric orthogonalization in its classical form involves calculating the square root of the inverse matrix in (4), which could be computationally expensive, for large n. Alternatively, simpler iterative methods can be used [9]. Nevertheless, the orthogonalization presents some difficulties, especially when the FastICA is implemented in hardware architectures [21]. Moreover, as we will show, the algorithms with symmetric orthogonalization could be more vulnerable to performance degradation, when the assumption regarding the maximum number of Gaussian sources is not met.

The ICA literature also contains examples of methods that do not require orthogonalization, for instance, Jacobi algorithms [2, 16, 22]. In these approaches, the matrix $\mathbf{B}$ is considered to be a product of orthogonal matrices, specifically the Jacobi rotations. The ICA is carried out by optimizing a contrast function for each pair of components that corresponds to a given rotation plane. Thus, by sweeping the local optimization over all possible pairs, the global independent sources can be extracted. The most important advantage of the Jacobi algorithms is that the local optimization can be solved analytically if the contrast function is given as a 4th-order polynomial (e.g., kurtosis). An example is the JADE technique [15] which consists in jointly approximately diagonalizing a set of eigenmatrices. The eigenmatrices are estimated by 4th-order moments [23]. In fact, this is a variant of the Jacobi algorithm, because the diagonalization is performed using the Jacobi rotations. Unfortunately, the use of the 4th-order moments as a contrast function may be discouraged due to poor asymptotic efficiency for super-Gaussians and lack of robustness to outliers [24]. Also, the computational complexity of the Jacobi-like algorithms can be relatively high, since the number of required data rotations increases quadratically with n. Theoretically, the number can be reduced by replacing the conventional Jacobi rotations with higher-dimensional rotations. For instance, 4×4 orthogonal matrices that represent rotations in R^4 can be used. It has been shown in [25] that such a matrix can be uniquely described by only two unit quaternions.

In our preliminary work [26], we proposed a novel four-channel ICA algorithm that uses quaternion-based factorizations of 4×4 orthogonal matrices and exploits a contrast function based on negentropy approximation. A solution based on the gradient method with the L2-norm constraint was developed. In this article, we propose to use the Newton-Raphson iterative scheme for solving the optimization task that results in a faster convergence rate as compared to the previous solution. Moreover, we extended our algorithm for the case of $n \geq 4$ channels, where n is even, by breaking n-dimensional problem up into a set of four-dimensional subproblems. For this purpose, a new sweep pattern for data-driven Jacobi-like algorithms is developed. Also, a more detailed comparative evaluation is performed that involves not only symmetric but also the deflation-based FastICA method and the JADE algorithm.

This paper is organized as follows. Section 2 describes quaternionic factorizations of 4×4 orthogonal matrices. In Section 3, a novel ICA algorithm is presented, and the solution based on the Newton-Raphson iterative scheme is derived. Also, a new sweep pattern for data-driven algorithms is developed. Section 4 investigates the performance of the proposed method via numerical simulations. Finally, the conclusions are given in Section 5.

2 Quaternions and rotations in $\mathbb{R}^4$

Most frequently a quaternion $Q \in \mathbb{H}$ is represented in the rectangular form:

$$Q = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3, \quad q_0, q_1, q_2, q_3 \in \mathbb{R}, \tag{5}$$

where $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ denote imaginary units. The real part of Q is q_0 and the pure quaternion part is $\boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3$. The multiplication of quaternions is determined by the following rules:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1. \tag{6}$$

It is associative and distributes over vector addition, but it is not commutative. Similarly as for ordinary complex numbers, the conjugate of Q is given by $\bar{Q} = q_0 - \boldsymbol{i}q_1 - \boldsymbol{j}q_2 - \boldsymbol{k}q_3$, and the norm (modulus), is defined as

$$|Q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \tag{7}$$

Since any quaternion Q can also be represented by the vector $\mathbf{q} = [\, q_0\, q_1\, q_2\, q_3]^T$, a set $\mathbb{H}$ can be identified with the $\mathbb{R}^4$ vector space.

Let $\{P, Q\}$ be a pair of unit quaternions, i.e., $|P| = |Q| = 1$, corresponding to the vectors $\mathbf{p}, \mathbf{q} \in \mathbb{R}^4$, and W be a quaternion corresponding to an arbitrary vector $\mathbf{w} \in \mathbb{R}^4$. Consider a real linear transformation from $\mathbb{H}$ to $\mathbb{H}$ that maps W to $PW\bar{Q}$. It can be shown that this transformation can be uniquely represented by a product of the 4×4 orthogonal matrices [25]:

$$\mathbf{M}^+(\mathbf{p}) = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix}, \tag{8}$$

$$\mathbf{M}^-(\mathbf{q}) = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix}. \tag{9}$$

In particular

$$PW\bar{Q} \quad \Leftrightarrow \quad \mathbf{M}^+(\mathbf{p})\mathbf{M}^-(\mathbf{q})\mathbf{w}. \tag{10}$$

It means that for an arbitrary 4×4 orthogonal matrix $\mathbf{R}$, there exist a unique pair of unit quaternions P and Q such that $\mathbf{R} = \mathbf{M}^+(\mathbf{p})\mathbf{M}^-(\mathbf{q})$. The proof of this little known theorem can be found in [27]. It should be noted that this product is commutative, and it is satisfied by the negative quaternions $-P, -Q$ as well. In fact, the pairs $\{P, Q\}$, and $\{-P, -Q\}$ map to same rotation of $\mathbb{R}^4$. The possibility of using two parameters to describe a rotation is a special feature of the group of rotations in four dimensions.

3 Methods

In this section, we introduce a recently proposed four-channel ICA method [26] and develop a new solution to an optimization problem based on the Newton-Raphson iterative scheme. We also propose a new ICA algorithm for $n \geq 4$ channels and give some illustrative examples of possible applications of this approach.

3.1 Jacobi-based ICA estimation framework

Our method strictly follows the Jacobi-based estimation framework. Namely, the independent components are estimated by recursively applying the so-called sweep operations:

$$\mathbf{y}_k = \mathbf{R}_k \mathbf{y}_{k-1}, \quad k \geq 1, \tag{11}$$

$$\mathbf{y}_0 = \mathbf{C}_{\mathbf{xx}}^{-1/2} \mathbf{x}, \tag{12}$$

where $\mathbf{R}_k$ is an orthogonal matrix and k is a sweep number. In the conventional Jacobi-based approaches [2, 16, 22], the matrix $\mathbf{R}_k$ is the product of the rotation matrices:

$$\mathbf{R}_k = \prod_{i=1}^{n(n-1)/2} \mathbf{G}(p_i, q_i, \theta_{k,i}), \tag{13}$$

where

$$\mathbf{G}(p, q, \theta) = \begin{bmatrix} \mathbf{I}_{p-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \cos\theta & \mathbf{0} & \sin\theta & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{q-p-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\sin\theta & \mathbf{0} & \cos\theta & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{n-q-1} \end{bmatrix}, \tag{14}$$

represents Jacobi rotation by the θ angle in the plane determined by the p and q axes. Please note that the matrix (14) is also known as the Givens matrix and the corresponding Givens rotations are no different from Jacobi rotations [28]. By using the factorization (13), an n-dimensional ICA problem can be decomposed into sequence of $n(n-1)/2$ one-dimensional subproblems. Each subproblem consists in searching for the angle that maximizes some contrast function [16] in the corresponding rotation plane. The data are rotated and the process is repeated cyclically, until all rotation angles converge to zeros.

3.2 Four-channel algorithm using quaternionic factorization

The key idea of the proposed method is to replace the Jacobi rotations with their four-dimensional quaternion-based counterparts. For simplicity, suppose that $n = 4$. Then, six ordinary Jacobi rotations can be replaced with only two quaternion-based rotations as follows:

$$\mathbf{R}_k = \mathbf{M}^{+}(\mathbf{p}_k)\mathbf{M}^{-}(\mathbf{q}_k). \tag{15}$$

Thus, in each sweep, two vectors $\mathbf{p}_k$, $\mathbf{q}_k \in \mathbb{R}^4$ have to be estimated that, for example, maximize some measure of non-Gaussianity of the rotated data. We propose to optimize these parameters in two consecutive steps in a similar way as the optimal rotation angles are determined for the Jacobi rotations. Firstly, we look for the vector $\mathbf{q}_k$ and rotate the data using the matrix $\mathbf{M}^{-}(\mathbf{q}_k)$. Then, we look for $\mathbf{p}_k$ and rotate the result of the first step by using $\mathbf{M}^{+}(\mathbf{p}_k)$. These steps have to be repeated until both rotation matrices are approximately equal to the identity matrices.

It was shown in [29] that measures of non-Gaussianity based on some non-linear functions are often considerably more robust than cumulant-based approximations. Moreover, by properly choosing the non-linear function, we can adjust the ICA algorithm to work for practically any distribution of source signals. One-unit contrast function

for measuring non-Gaussianity of any random variable y in the negentropy-based ICA framework is given by [9]:

$$J_g(y) = [E\{g(y)\} - E\{g(\nu)\}]^2, \tag{16}$$

where ν is a standardized Gaussian variable, and g is any non-linear function. Examples of such functions can be found in [30]. The random variable y is assumed to be zero-mean and unit-variance. Please note that the measure (16) is always non-negative, and it equals to zero if y is Gaussian. As suggested in [9], a contrast function for several channels (units) can be obtained by simply maximizing the sum of the one-unit functions. Thus, by taking into account the unit-norm constraints (that enforces orthogonality), we defined the following optimization problems:

$$\mathbf{q}_k = \arg\max_{\mathbf{u}} \sum_{i=1}^{4} E\{\bar{g}([\mathbf{M}^-(\mathbf{u})]_{i*}\,\mathbf{y}_k^-)\}^2 \quad \text{s.t.} \quad ||\mathbf{u}|| = 1 \tag{17}$$

$$\mathbf{p}_k = \arg\max_{\mathbf{u}} \sum_{i=1}^{4} E\{\bar{g}([\mathbf{M}^+(\mathbf{u})]_{i*}\,\mathbf{y}_k^+)\}^2 \quad \text{s.t.} \quad ||\mathbf{u}|| = 1 \tag{18}$$

where the operator $[\mathbf{A}]_{i*}$ denotes ith row of an arbitrary matrix $\mathbf{A}$, $\bar{g}(y) = g(y) - E\{g(\nu)\}$ is a centered version of the non-linear function $g(y)$, and

$$\mathbf{y}_k^- = \mathbf{y}_{k-1} = \mathbf{R}_{k-1}\mathbf{R}_{k-2}\cdots\mathbf{R}_1\mathbf{C}_{\mathbf{xx}}^{-1/2}\mathbf{x}, \tag{19}$$

$$\mathbf{y}_k^+ = \mathbf{M}^-(\mathbf{q}_k)\mathbf{y}_{k-1}, \tag{20}$$

$$\mathbf{y}_k = \mathbf{M}^+(\mathbf{p}_k)\mathbf{M}^-(\mathbf{q}_k)\mathbf{y}_{k-1}, \tag{21}$$

are the rotated random vectors for $k \geq 1$. We assume that the expectations in (17), (18) exist and are finite. Therefore, they can be approximated by sums, and the vector $\mathbf{x}$ in (19) can be replaced with an observation matrix $\mathbf{X}$ of size $4 \times m$ consisting of m mixture vectors stacked in column-wise order. The vectors (19), (20) can be replaced with transformed data matrices $\mathbf{Y}_k^-, \mathbf{Y}_k^+ \in R^{4\times m}$ as well. Thus, by using the matrix-vector notation, we can redefine the contrast functions in (17) and (18) as follows:

$$J(\mathbf{u}) = \mathbf{1}_m^T \bar{g}[\mathbf{M}(\mathbf{u})\mathbf{Y}_k]^T\, \bar{g}[\mathbf{M}(\mathbf{u})\mathbf{Y}_k]\, \mathbf{1}_m, \tag{22}$$

where the terms $\{\mathbf{M}, \mathbf{Y}_k\}$ stand either for $\{\mathbf{M}^+, \mathbf{Y}_k^+\}$ or $\{\mathbf{M}^-, \mathbf{Y}_k^-\}$. Please note that the problems (17) and (18) are similar, and the solution depends only on the current sweep data. Therefore, in the rest of the article, the subscripts $+$, $-$, and the swept number k are dropped to simplify the notation.

3.3 Solution based on the Newton-Raphson method

In accordance with the Kuhn-Tucker conditions, any stationary point of (22) must satisfy the following equation:

$$\nabla_{\mathbf{u}} J(\mathbf{u}) + \lambda\mathbf{u} = \mathbf{0}, \tag{23}$$

where λ is the Lagrange multiplier, and

$$\nabla_{\mathbf{u}} J(\mathbf{u}) = \left[\frac{\partial J(\mathbf{u})}{\partial u_0} \cdots \frac{\partial J(\mathbf{u})}{\partial u_3} \right]^T, \tag{24}$$

denotes the gradient of $J(\mathbf{u})$. The partial derivatives can be computed as follows:

$$\frac{\partial J(\mathbf{u})}{\partial u_i} = 2\mathbf{a}(\mathbf{u})^T \mathbf{b}_i(\mathbf{u}), \tag{25}$$

with

$$\mathbf{a}(\mathbf{u}) = \bar{g}(\mathbf{M}(\mathbf{u})\mathbf{Y})\mathbf{1}_m, \tag{26}$$

$$\mathbf{b}_i(\mathbf{u}) = \left[\bar{g}'(\mathbf{M}(\mathbf{u})\mathbf{Y}) \circ \frac{\partial \mathbf{M}(\mathbf{u})}{\partial u_i} \mathbf{Y} \right] \mathbf{1}_m, \quad i = 0, \ldots, 3, \tag{27}$$

where $\circ$ denotes the Hadamard product, $\bar{g}'$ is the first-order derivative of the non-linear function $\bar{g}$, and $\frac{\partial \mathbf{M}(\mathbf{u})}{\partial u_i}$ is ith partial derivative of a given quaternionic matrix. These matrices take the form of simple permutation matrices containing only zeros and ones (see the Appendix for details), so that, in practice, the multiplication by $\mathbf{Y}$ can be avoided.

One can try to solve Eq. 23 numerically by using, for example, the quasi-Newton method in a similar way as in the FastICA approach [9]. However, in this case, it could be a more difficult task as it involves a derivation of the second-order derivative (Hessian matrix) of the quadratic form (25) or finding its approximation. Even if an explicit expression for this matrix will be obtained, its computation could be very expensive.

In our previous work [26], we proposed a much simpler solution based on a gradient method with L_2-norm constraint [31]. A geometric interpretation of this method is given in Fig. 1. In a such approach, a pre-scaled gradient of $J(\mathbf{u})$, say

$$\mathbf{h}(\mathbf{u}) = \mu \nabla_{\mathbf{u}} J(\mathbf{u}), \tag{28}$$

with $\mu > 0$ being an empirically chosen scaling factor, is projected onto hyperplane orthogonal to the unit vector $\mathbf{u}$:

$$\boldsymbol{\delta} = \mathbf{h}(\mathbf{u}) - \mathbf{u}\mathbf{u}^T \mathbf{h}(\mathbf{u}). \tag{29}$$

The set of all such vectors at a given point $\mathbf{u}$ is known as the tangent space of the constraint surface $\|\mathbf{u}\| = 1$. In our case, the constraint surface is a unit four-dimensional hypersphere (also known as 3-sphere). Then, the solution vector is updated by using the following rule:

$$\mathbf{u}^* \leftarrow \mathbf{u} \cos\theta + \frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|} \sin\theta, \tag{30}$$

which is equivalent to the rotating $\mathbf{u}$, the current solution vector, in the direction of $\boldsymbol{\delta}$ by some angle θ along the geodesic, i.e., curve on the constraint surface that connects two arbitrary points by an arc of shortest length.

It is a well-known fact that the convergence rate of the gradient methods depends on the value of the scaling factor μ. In particular, these methods do not necessarily converge for large values of μ. On the other hand, for small values of μ, the algorithm may converge very slowly. In the work [26], we assumed that $\theta = \|\boldsymbol{\delta}\|$, and the scaling factor μ was empirically set to some positive value that depends on a given non-linear function $\bar{g}(y)$. Here, we propose a more sophisticated approach that follows the Newton-Raphson iterative scheme and does not require the empirically determined parameters. Please note

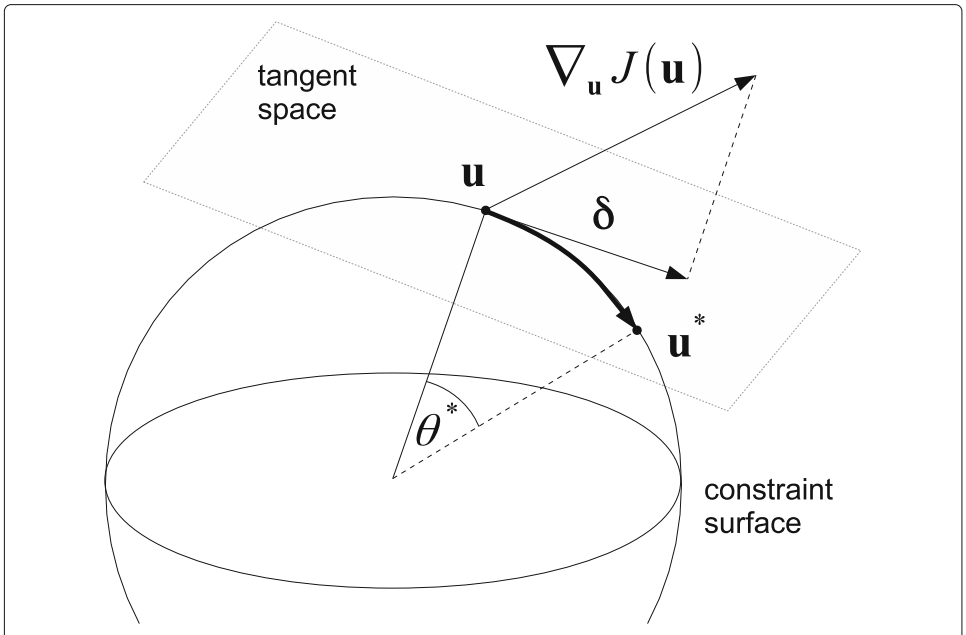

Fig. 1 Optimization using gradient method with L_2-norm constraint.

that instead of searching for optimal μ, we can equivalently set $\mu = 1$ and maximize the following cost function with respect to rotation angle θ:

$$\hat{J}(\theta) = \mathbf{1}_m^T \bar{g}[\,\hat{\mathbf{M}}(\theta)\mathbf{Y}]^T\, \bar{g}[\,\hat{\mathbf{M}}(\theta)\mathbf{Y}]\,\mathbf{1}_m, \tag{31}$$

where

$$\hat{\mathbf{M}}(\theta) = \mathbf{M}\left(\mathbf{u}\cos\theta + \frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|}\sin\theta\right), \tag{32}$$

and $\mathbf{u}$ is any vector that satisfies $\|\mathbf{u}\| = 1$. Thus, each constrained optimization task (17), (18) can be simplified to one-dimensional angle search on the unit circle. Now, the Newton-Raphson iterative procedure can be used for finding the optimal rotation angle:

$$\theta^* \leftarrow \theta + \frac{\partial\hat{J}(\theta)}{\partial\theta} \Bigg/ \left|\frac{\partial^2\hat{J}(\theta)}{\partial\theta}\right|, \tag{33}$$

where the first and second derivatives of (31) are given by

$$\frac{\partial\hat{J}(\theta)}{\partial\theta} = 2\hat{\mathbf{a}}(\theta)^T\hat{\mathbf{b}}(\theta), \tag{34}$$

$$\frac{\partial^2\hat{J}(\theta)}{\partial\theta} = 2\left[\hat{\mathbf{a}}(\theta)^T\mathbf{C}(\theta)\mathbf{1}_m, +\|\hat{\mathbf{b}}(\theta)\|^2\right] \tag{35}$$

respectively, and

$$\hat{\mathbf{a}}(\theta) = \bar{g}(\hat{\mathbf{M}}(\theta)\mathbf{Y})\mathbf{1}_m, \tag{36}$$

$$\hat{\mathbf{b}}(\theta) = \left[\bar{g}'(\hat{\mathbf{M}}(\theta)\mathbf{Y}) \circ \frac{\partial\hat{\mathbf{M}}(\theta)}{\partial\theta}\mathbf{Y}\right]\mathbf{1}_m, \tag{37}$$

$$\mathbf{C}(\theta) = \bar{g}'(\hat{\mathbf{M}}(\theta)\mathbf{Y}) \circ \left[\frac{\partial\hat{\mathbf{M}}(\theta)}{\partial\theta}\mathbf{Y}\right]^{\circ 2} - \bar{g}''(\hat{\mathbf{M}}(\theta)\mathbf{Y}) \circ \hat{\mathbf{M}}(\theta)\mathbf{Y}. \tag{38}$$

In the above expression, the operator $\circ 2$ denotes element-wise (Hadamard) power, and $\bar{g}''$ is the second-order derivative of the non-linear function $\bar{g}$. Please note that we slightly modified the Newton-Raphson iteration (33) by using plus sign and taking the absolute value of the denominator. As suggested in [32], in this way, we can enforce the algorithm to find local maxima only.

In the case of the Jacobi-based algorithms, the rotation hyperplanes could overlap each other. As a result, local optimizations may not preserve (and often destruct) the global optimality. Therefore, during the early sweeps, it may not be necessary to compute the rotation angles with the highest accuracy [25]. Moreover, each rotation is performed in a local coordinate system. For these reasons, in each sweep, we perform only one iteration of the Newton-Raphson method per each optimization task (17), (18), assuming that

$$\mathbf{u} \leftarrow \mathbf{u}_0 = [1\,0\,0\,0]^T, \tag{39}$$

$$\theta \leftarrow \theta_0 = 0. \tag{40}$$

Please note that a similar simplification can also be found in [33]. It can be easily verified that

$$\frac{\partial \hat{\mathbf{M}}(\theta)}{\partial \theta} = \mathbf{M}\left(\frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|}\cos\theta - \mathbf{u}\sin\theta\right). \tag{41}$$

Since $\hat{\mathbf{M}}(0) = \mathbf{I}$, and $\frac{\partial \hat{\mathbf{M}}(\theta=0)}{\partial \theta} = \mathbf{M}\left(\frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|}\right)$ the expressions (36) - (38) can be simplified to:

$$\hat{\mathbf{a}}(0) = \bar{g}(\mathbf{Y})\mathbf{1}_m, \tag{42}$$

$$\hat{\mathbf{b}}(0) = \left[\bar{g}'(\mathbf{Y}) \circ \mathbf{M}\left(\frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|}\right)\mathbf{Y}\right]\mathbf{1}_m, \tag{43}$$

$$\mathbf{C}(0) = \bar{g}'(\mathbf{Y})\left[\mathbf{M}\left(\frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|}\right)\mathbf{Y}\right]^{\circ 2} - \bar{g}''(\mathbf{Y}) \circ \mathbf{Y}, \tag{44}$$

and the optimal rotation angle can be estimated as follows:

$$\theta^* \leftarrow \frac{\hat{\mathbf{a}}(0)^T\hat{\mathbf{b}}(0)}{|\hat{\mathbf{a}}(0)^T\mathbf{C}(0)\mathbf{1}_m + \|\hat{\mathbf{b}}(0)\|^2| + \epsilon}. \tag{45}$$

where ϵ is some small positive constant introduced to prevent division by zero. Once the rotation angle is computed, the current data matrix is transformed as $\mathbf{Y} \rightarrow \hat{\mathbf{M}}(\theta^*)\mathbf{Y}$. Recalling the factorization (15), we see that in each sweep, up to two such transformations are needed. This procedure is repeated cyclically until the convergence is achieved, i.e., for both transformations, we have $|\theta^*| < \zeta$, where ζ is sufficiently small positive constant.

3.4 Quaternionic ICA for $n \geq 4$ channels

An implementation of the four-channel algorithm is rather straightforward as it comprises only two four-dimensional optimization subproblems per sweep. As shown in the previous section, these subproblems can be solved numerically using a quasi-Newton method. An algorithm for $n \geq 4$ channels can be developed once we determine how to design a sweep pattern consisting of a complete set of four-dimensional subproblems. Intuitively, every component (row of the data matrix $\mathbf{Y}$) should be part of any four-dimensional subproblem at least once during the sweep, so no component of the whole data set is exempt from direct participation in the local optimization preserving a

drive towards the global optimum. This idea follows the conventional Jacobi-based ICA estimation framework implemented as a sequence of plane rotations.

It turns out that the sweep patterns comprising four-dimensional subproblems are not difficult to construct and have already been studied in the context of structured eigenproblems [34, 35]. In these approaches, the location of the four-dimensional subproblem was controlled by the position of the element a_{ij} of arbitrary symmetric or skew-symmetric matrix $\mathbf{A} \in \mathbb{R}^{2r\times 2r}$, chosen from the $r \times r$ upper diagonal block, where $r \geq 2$. This element uniquely determines 4×4 principal submatrix located in the rows and columns $i, j, r+i$ and $r+j$ as shown below:

$$\left[\begin{array}{cc|cc} a_{i,i} & a_{i,j} & a_{i,r+i} & a_{i,r+j} \\ a_{j,i} & a_{j,j} & a_{j,r+i} & a_{j,r+j} \\ \hline a_{r+i,i} & a_{r+i,j} & a_{r+i,r+i} & a_{r+i,r+j} \\ a_{r+j,i} & a_{r+j,j} & a_{r+j,r+i} & a_{r+j,r+j} \end{array}\right] \tag{46}$$

where $1 \leq i < r$ and $i < j \leq r$. Please note that such submatrices preserve the structure of the parent matrix $\mathbf{A}$, i.e., symmetry or skew-symmetry. Furthermore, a sweep comprised entirely of four-dimensional subproblems can be generated by using any cyclic or quasi-cyclic sweep of the upper-left block of the matrix $\mathbf{A}$. As reported in [35], such a sweep pattern is inherently parallelizable, is numerically stable, and shows asymptotic quadratic convergence.

In this work, we propose to extend this idea to a data-driven ICA algorithm for even $n = 2r$. Instead, to diagonalize the set of eigenmatrices as in the JADE algorithm, we operate directly on the data by following the aforementioned sweep pattern. Namely, our four-dimensional subproblems are constructed by cyclically selecting the components of indexes $i, j, r+i$, and $r+j$. All steps of the proposed method are summarized as the pseudocode in Algorithm 1. In order to generate the pairs $\{i, j\}$, a typical two-dimensional row-cycling ordering was used, but similarly to conventional Jacobi-based methods, other arrangements are also possible [36].

A rigorous theoretical proof of convergence of the algorithm seems to be a challenging task and is out of the scope of this article. However, we observed empirically that the proposed method is asymptotically quadratically convergent. Please note that our algorithm requires only $n(n-2)/4$ quaternion-based rotations per sweep in place of $n(n-1)/2$ Jacobi rotations as in the JADE method. Obviously, it suggests a faster convergence of the proposed method, especially for large n. However, the rotations performed in the JADE method have the lower computational complexity as they are used to transform cumulant matrices only and optimal rotation angles are computed analytically. These issues will be examined in more detail in the experimental section.

3.5 Illustrative examples

Figure 2 demonstrates an example of the source recovery using the proposed method. As input data, a linear mixture of the eight basic signals has been used. The original and randomly mixed waveforms are depicted in Fig. 2a and b, respectively. Figure 2c presents the source signals recovered by using the proposed approach. In order to illustrate the convergence characteristics, in Fig. 2d, we also show the negentropy approximation computed after each sweep for all channels as well as for local four-dimensional subproblems.

Algorithm 1: Pseudocode of the proposed method

Data: $\mathbf{X}$ - data matrix of size $n \times m$ containing n observed signals (stacked in rows) each of size m samples, k_{max} - maximum number of sweeps, ζ - minimum rotation angle

Result: $\hat{\mathbf{Y}}$ - matrix of size $n \times m$ containing estimated sources (stacked in rows)

$\mathbf{C_{xx}} \leftarrow \frac{1}{m}\mathbf{X}\mathbf{X}^T$

$\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, ..., \hat{\mathbf{y}}_n]^T \leftarrow \mathbf{C_{xx}}^{-1/2}\mathbf{X}$

$\mathbf{u}_0 \leftarrow [1, 0, 0, 0]^T$

for $k \leftarrow 1, 2, ..., k_{max}$ **do**

  isRotation $\leftarrow$ *false*

  for $i \leftarrow 1, 2, ..., n/2 - 1$ **do**

    for $j \leftarrow i + 1, i + 2, ..., n/2$ **do**

      $\mathbf{Y} \leftarrow [\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j, \hat{\mathbf{y}}_{n/2+i}, \hat{\mathbf{y}}_{n/2+j}]^T$

      compute $\boldsymbol{\delta}$ using (29) with $\mathbf{M}(\mathbf{u}) \stackrel{\text{def}}{=} \mathbf{M}^-(\mathbf{u}_0), \mu = 1$

      compute θ angle using (45)

      if $|\theta| > \zeta$ **then**

        $\mathbf{Y} \leftarrow \hat{\mathbf{M}}(\theta)\mathbf{Y}$

        isRotation $\leftarrow$ *true*

      end

      compute $\boldsymbol{\delta}$ using (29) with $\mathbf{M}(\mathbf{u}) \stackrel{\text{def}}{=} \mathbf{M}^+(\mathbf{u}_0), \mu = 1$

      compute θ angle using (45)

      if $|\theta| > \zeta$ **then**

        $\mathbf{Y} \leftarrow \hat{\mathbf{M}}(\theta)\mathbf{Y}$

        isRotation $\leftarrow$ *true*

      end

      $[\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j, \hat{\mathbf{y}}_{n/2+i}, \hat{\mathbf{y}}_{n/2+j}]^T \leftarrow \mathbf{Y}$

    end

  end

  if *isRotation* == *false* **then**

    break

  end

end

It can be seen that the method converges in less than six sweeps, and the source signals have been successfully recovered (up to permutation and sign).

We also considered a linear mixture of five images of natural scenes and Gaussian noise. The test images were selected from the BSDS300 database [37]. They were of size 481×321 pixels with an 8-bit grayscale. The original images and mixtures are presented in Fig. 3a and b. Figure 3c and d present recovered images after $k = 2$ and $k = 10$ sweeps, respectively. These experiments show that the proposed algorithm may have good convergence properties in general settings, i.e., for $n > 4$.

In the next example (see Fig. 4), we show that the proposed method can also be useful if the number of sources is smaller than four. Suppose that we have only three independent sources: two speech signals and some noise. The sources were mixed using a random

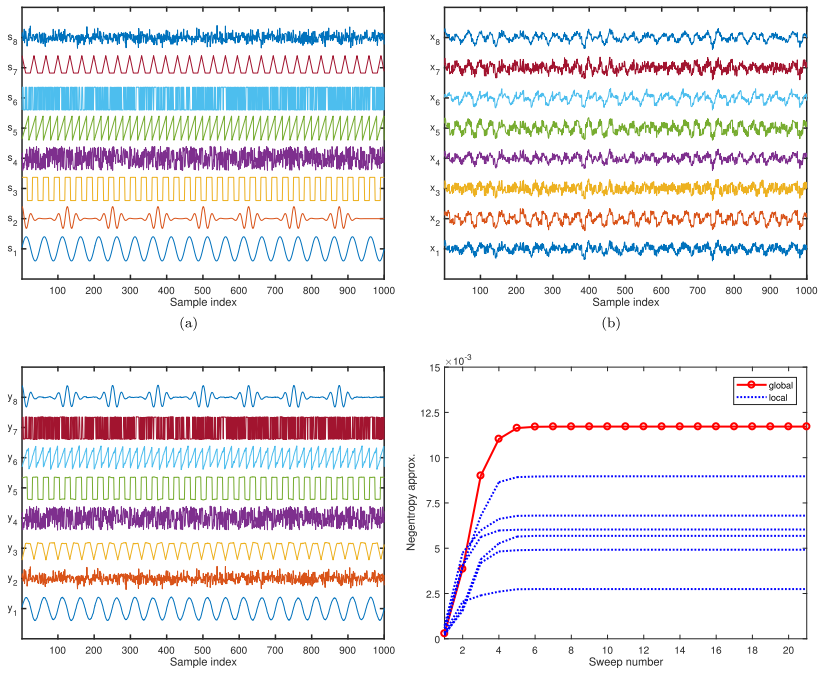

Fig. 2 Demonstration of source extraction using the proposed method. **a** Source signals. **b** Observed mixtures of the source signals. **c** Estimates of the source signals. **d** Contrast function values versus the sweep number

matrix of size 4×3 so that there is a linear dependence between the rows of the matrix. In the case of conventional ICA approaches, the input signals are usually whitened using principal component analysis (PCA), and the dimensionality of the data is decreased at the same time. Only components that correspond to sufficiently large eigenvalues are taken into account. However, in the case of our method, we cannot reduce the dimension-

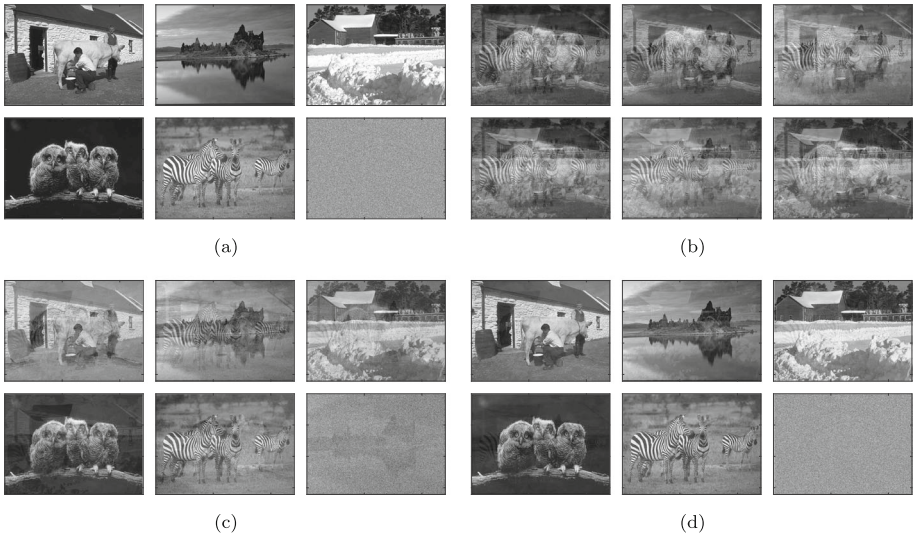

Fig. 3 Separation of images using the proposed method. **a** Original images of natural scenes and Gaussian noise. **b** Linearly mixed images. **c** Separated images after 2 sweeps. **d** Separated images after 10 sweeps

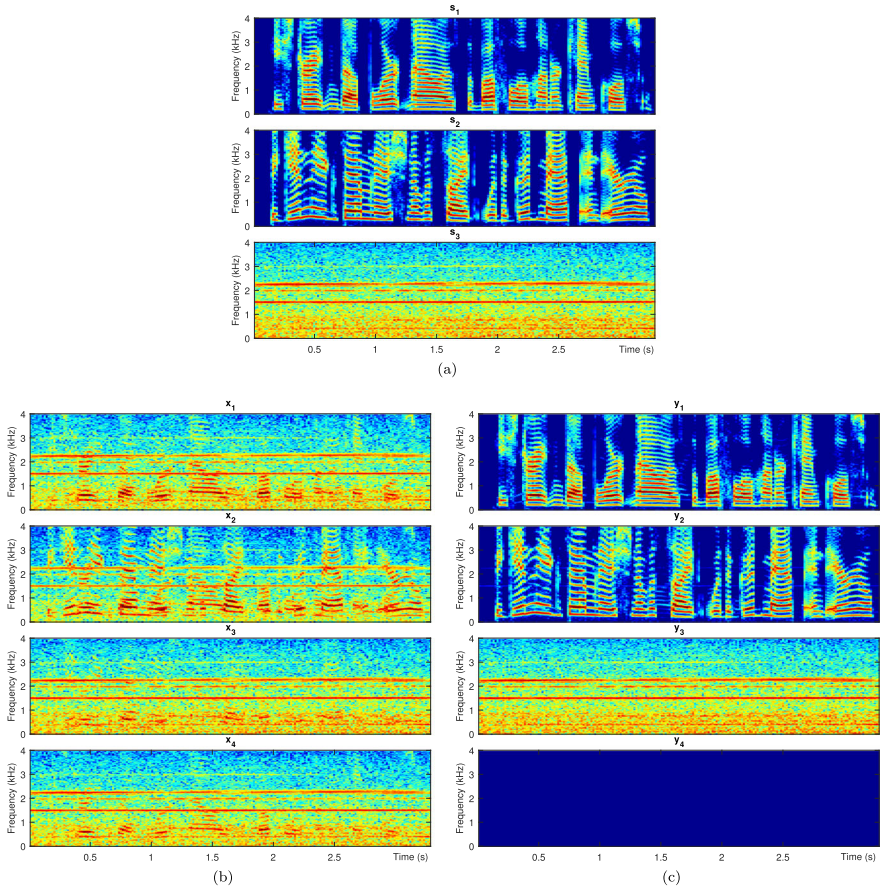

Fig. 4 Extraction of speech signals. **a** Original speech signals. **b** Mixtures of two speech signals and one noise source projected onto four-dimensional space. **c** Source signals estimated using the proposed method

ality of data as easily as it is closely related to the size of the rotation matrices. Therefore, for $n < 4$, the whitening matrix (4) must be properly regularized:

$$\mathbf{C}_{\mathbf{xx}}^{-1/2} \approx \mathbf{U} \begin{bmatrix} \boldsymbol{\Sigma}_s^{-1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{U}^T = \mathbf{U}_s \boldsymbol{\Sigma}_s{}^{-1/2} \mathbf{U}_s^T, \tag{47}$$

where $\boldsymbol{\Sigma}_s$ is a diagonal matrix of strictly positive eigenvalues of $\mathbf{C}_{\mathbf{xx}}$ and $\mathbf{U}_s$ is a (possibly rectangular) matrix of corresponding eigenvectors. In other words, the whitening stage is performed in the signal-subspace only, while the minor-components are replaced by zero vectors. Although such solution is suboptimal from the computational point of view, it makes the implementation much easier. As can be seen in Fig. 4c, the original sources have been properly extracted, and the 4th component has been estimated as the zero vector. These results suggest that for $n < 4$ the minor-components that correspond to nullspace can be treated as non-Gaussian components. Indeed, the cost function (16) takes positive values for y being a zero signal. Therefore, by maximizing this function, we can recover the zero vectors as well as the original sources.

4 Results and discussion

4.1 Experimental setup

The proposed algorithm has been implemented in the MATLAB environment. For the purpose of empirical evaluation, we denote our method as the QICA (Quaternionic ICA). It has been compared to the symmetric FastICA, deflation-based FastICA [9], and JADE algorithm [15]. For FastICA implementations, we used freely available MATLAB packages from [38]. The implementation of the JADE algorithm is a part of the EEGLAB open-source package [39]. The recent implementation of the JADE algorithm in R language as well as some joint diagonalization functions is also described in [40].

In the case of the QICA and FastICA methods, the experiments were conducted for the three non-linear functions that are frequently considered in the publications [9, 30]:

$$\begin{aligned} \text{POW3:} &\quad \bar{g}(y) = y^4 - 3, \\ \text{TANH:} &\quad \bar{g}(y) = \log(\cosh(y)) - 0.3746, \\ \text{GAUS:} &\quad \bar{g}(y) = -\exp(-y^2/2) + 0.7071. \end{aligned}$$

We use these names in order to match the literature, even though one can consider them to be misleading. Namely, these names are related to the first-order derivatives, not to the functions themselves.

Theoretical stability analysis of the proposed method is out of the scope of this article. Therefore, the effectiveness of the QICA method was evaluated numerically. In particular, their convergence characteristics, source separation quality, and computation time were measured for several conditions involving synthetic and real data. Please note that it makes little sense to measure convergence speed in terms of iteration number or average computation time taken by an algorithm to converge as the reference methods (except the JADE) use different stop criteria. Therefore, in order to make the comparison meaningful, we decided to disable stop conditions in all methods and perform simulation for a fixed number of iterations measuring the separation quality in each iteration step. In this way, we were also able to measure the convergence characteristics of the evaluated algorithms.

Synthetic sources were generated for the following distributions: uniform, BPSK, Laplacian, and generalized Gaussians (GG(α) with parameter $\alpha = 3$, and 0.5) [30]. The length of each source signal was $m = 1000$ samples. In our experiments, we also considered "speech" distribution that was represented by data frames randomly selected from real speech recordings. These recordings were sourced from the publicly available database that was used to develop ITU-T Recommendation [41]. The database consists of short few-seconds sentences spoken by female and male speakers in various languages. Depending on the scenario, we have selected from 4 up to 12 sentences from the American English set. Original samples were available in the WAV 32-bit floating-point format with sampling frequency 16 kHz, but for our purposes, they were downsampled to 8 kHz. Both synthetic and real sources were normalized to have zero-mean and unit variance.

In a basic scenario, we linearly mixed $n = 4$ sources of the same non-Gaussian distribution. In order to evaluate the robustness of the compared algorithms to the Gaussian sources, we also considered the scenarios in which one or two sources were Gaussian and the others were non-Gaussian but of the same distribution. Additionally, the experiments with $n = 8$ and $n = 12$ non-Gaussian sources have been conducted to examine a global convergence of the QICA algorithm.

In all scenarios, the coefficients of the mixing matrix were generated from the uniform distribution. Ill-conditioned matrices (with a condition number greater than 1000) were excluded from evaluation. The performance indexes were averaged over 1000 random realizations of the sources and the mixing matrices.

4.2 Separation quality

The separation quality was estimated using signal to interference ratios (SIRs) [30, 42] and averaged over all non-Gaussian components. In particular, for the kth source signal, the SIR index is given by:

$$\mathrm{SIR}_k = \frac{[\mathbf{G}]_{kk}^2}{\sum_{l=1}^{n}[\mathbf{G}]_{kl}^2}, \tag{48}$$

where $\mathbf{G} = \hat{\mathbf{W}}\mathbf{A}\mathbf{D}^{1/2}$ is the gain matrix with $\hat{\mathbf{W}}$ being the estimated demixing matrix of the corresponding algorithm, and $\mathbf{D} = \mathrm{diag}\{\sigma_1^2, \sigma_2^2, ..., \sigma_n^2\}$ being the diagonal matrix containing the variances of the original signals. Since the BSS problem can be solved only up to permutation, the rows of the matrix $\hat{\mathbf{W}}$ must be appropriately reordered which involves a pairing process [43]. For this purpose, we followed [44] and used a greedy algorithm. In practice, for some distributions, the off-diagonal elements of the gain matrix can be close to zeros. Therefore, it is often necessary to add some small positive constant to the denominator of (48) in order to prevent dividing by zero. In our experiments, the constant of value 10^{-9} was used.

The results presented in Tables 1, 2, and 3 refer to the scenarios with $n = 4$ components containing from 0 up to 2 Gaussian sources. The maximum number of iterations was set to $k = 40$ for all methods. In the case of the deflation-based FastICA, the components were estimated, one by one, enforcing 40 iterations per component. The best results obtained for each distribution were denoted using bold fonts. It can be observed that the performance strongly depends on the distribution of source signals and the number of Gaussian sources.

In the first scenario (Table 1—no Gaussians), we see that the QICA and symmetric FastICA with the POW3 function gives similar results as the JADE algorithm. It is rather not surprising since the cost function of the JADE algorithm is based on 4th-order cumulants and the POW3 function is closely related to the kurtosis.

In opposition to the JADE algorithm, the performance of the QICA and FastICA methods can be improved for some distributions by selecting appropriate non-linear functions. It can be observed that if the TANH or GAUSS function is selected, we get higher SIR indexes for mixtures with the Laplacian and GG (0.5) distributions.

Table 1 Average SIR (dB) for 4 components with the same distribution, after $k = 40$ iterations

	QICA			Symm. FastICA			Deflat. FastICA			JADE
PDF	POW3	TANH	GAUSS	POW3	TANH	GAUSS	POW3	TANH	GAUSS	
Uniform	31.49	30.44	30.36	31.50	30.45	30.37	29.36	27.90	27.80	**31.51**
BPSK	34.24	34.24	34.24	34.25	34.23	34.24	76.80	**80.88**	80.33	34.24
GG(3.0)	21.48	**21.72**	21.62	21.31	21.59	21.46	18.28	18.29	18.18	21.37
Laplacian	23.10	27.32	**27.57**	23.48	27.40	**27.57**	20.70	24.17	24.44	23.09
GG(0.5)	27.33	31.91	31.45	27.64	31.86	**32.37**	26.18	30.03	30.61	27.46
Speech	20.64	24.79	**25.58**	19.65	22.82	22.98	19.13	22.06	22.67	21.68

Numbers in boldface show the best results

Table 2 Average SIR (dB) for 3 components with the same distribution excluding 1 Gaussian source, after $k = 40$ iterations

	QICA			Symm. FastICA			Deflat. FastICA			JADE
PDF	POW3	TANH	GAUSS	POW3	TANH	GAUSS	POW3	TANH	GAUSS	
Uniform	**29.46**	28.65	28.58	26.99	27.22	27.35	28.02	26.90	26.81	29.38
BPSK	31.82	31.86	31.86	30.25	31.54	31.60	**34.18**	34.12	34.12	31.83
GG(3.0)	19.73	**20.01**	19.89	17.80	18.64	18.79	17.59	17.47	17.46	19.52
Laplacian	21.90	25.63	**25.98**	21.92	25.05	25.30	20.26	23.62	23.98	21.89
GG(0.5)	25.21	29.61	**30.00**	25.46	29.30	29.69	24.67	28.24	28.84	25.34
Speech	15.96	21.87	**22.85**	15.31	19.71	20.41	16.10	19.98	20.76	18.24

Numbers in boldface show the best results

By analyzing Tables 1, 2, and 3, it is also clear that the performance of all methods degrades with increasing the number of the Gaussian sources. However, the QICA method appeared to be more robust in the presence of Gaussian sources. In the scenario where one Gaussian source was used (Table 2), our algorithm provides substantially better separation quality in almost all cases. This observation is even more evident in the last scenario (Table 3), where two Gaussians were used. A similar examination can be made for the JADE method, but the proposed method outperforms the JADE algorithm for the Laplacian and GG(0.5) distributions.

Please note, that if there are no Gaussian sources, then each local minimum or maximum of the one-unit contrast function corresponds to one independent component. Since the contrast function (16) is blind to Gaussian sources, in the presence of one or more Gaussians, the number of local optima is usually lower than n, so that symmetric orthogonalization, as in FastICA approach, may pose some stability issues. Most frequently, this problem is mitigated by reducing the dimensionality of the observed data, which in fact, involves the identification of non-Gaussian subspace. The simplest methods for dimensionality reduction assume that information contained in the high-dimensional data is essentially non-Gaussian, and it is located on a lower-dimensional manifold. Furthermore, the Gaussian components are assumed to be one order of magnitude smaller than the non-Gaussian components. In such a case, the dimensionality of the observed data can be reduced by using, for example, the PCA technique. However, this is often a too strict assumption on the data, and in general, the identification of the non-Gaussian subspace is not a trivial task [45, 46].

Unlike the FastICA method, the QICA as well as the JADE approaches are inherently orthogonal and thus more robust to signal model mismatch, i.e., situation where there is

Table 3 Average SIR (dB) for 2 components with the same distribution excluding 2 Gaussian sources, after $k = 40$ iterations

	QICA			Symm. FastICA			Deflat. FastICA			JADE
PDF	POW3	TANH	GAUSS	POW3	TANH	GAUSS	POW3	TANH	GAUSS	
Uniform	26.89	26.34	26.23	21.74	23.84	23.93	26.50	25.58	25.44	**26.97**
BPSK	29.47	29.53	**29.54**	26.10	28.53	28.62	29.53	29.49	29.47	29.34
GG(3.0)	17.54	**18.23**	18.13	14.80	16.06	16.07	16.39	16.62	16.32	17.40
Laplacian	20.43	23.93	**24.16**	20.03	22.91	22.75	19.40	22.62	22.85	20.41
GG(0.5)	23.55	27.62	**28.03**	23.72	27.06	27.21	23.28	26.87	27.31	23.69
Speech	8.85	15.75	**16.82**	7.93	13.28	13.67	8.92	14.22	15.60	9.97

Numbers in boldface show the best results

more than one Gaussian source. It can be verified empirically that the data rotations are performed in the non-Gaussian subspace only. Thus, these methods are able to successfully recover non-Gaussian components even if the observation mixtures contain multiple Gaussians.

4.3 Convergence rate

Figure 5 presents average SIR indexes versus iteration number obtained for various data distributions and the number of components ranging from 4 to 12. In this experiment, no Gaussian sources were used. In the case of the QICA and symmetric FastICA algorithm, the GAUSS non-linear function was selected. The deflation-based FastICA is not considered here, as it works differently by recovering the components one by one. We observed empirically that, for the first component, the accuracy of this method is similar to that of the other approaches. However, for the remaining components, we noticed a substantial performance drop due to deflationary orthogonalization. Thus, on average, this method performs rather poorly as compared to other algorithms (see Table 1).

In the case of $n = 4$ channels, we do not observe any significant difference in the convergence rate of the QICA and symmetric FastICA algorithms for most distributions. However, the proposed method evidently wins with the FastICA, for $n > 4$. The

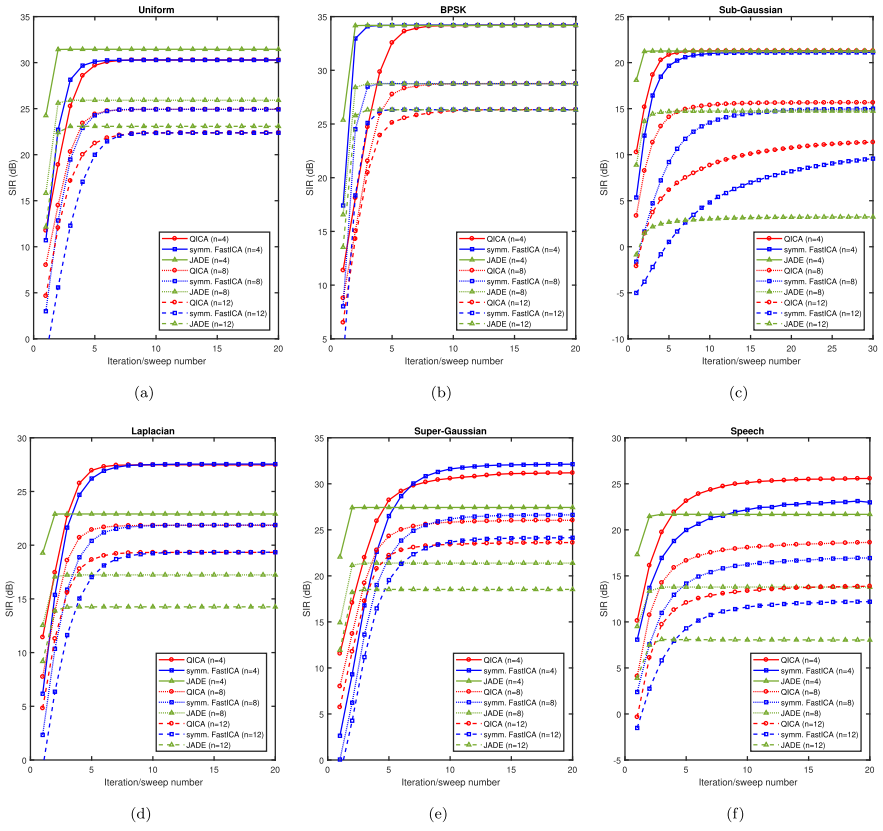

Fig. 5 Average SIR index as a function of sweep/iteration number, obtained using QICA (circles), symmetric FastICA (squares), and JADE algorithm (triangles) for various data distributions. **a** Uniform distribution. **b** BPSK signals. **c** Sub-Gaussian distribution. **d** Laplacian distribution. **e** Super-Gaussian distribution. **f** Speech distribution represented by randomly selected frames of the real speech recordings

only exception is the BPSK distribution, where the QICA algorithm converges slowly as compared to the reference methods. Although the SIR decreases with the increase in the number of components, this degradation is similar for all approaches and can be compensated by increasing the number of samples [30].

It can be observed that the JADE algorithm needs the smallest number of sweeps to converge, but its accuracy is considerably lower than that of the QICA or symmetric FastICA for most distributions. This is because the JADE algorithm is based on 4th-order cumulants which are known to be less sensitive for some mixtures, e.g., Laplacians and super-Gaussians.

Since there is a difference in the number of rotations per sweep between the QICA and JADE algorithms, and both methods can be configured to use the same stop criteria, it may be interesting to measure the total number of rotations that are needed to achieve convergence. Hence, in the next experiment, the maximum number of sweeps was set to $k_{\max} = 100$ and the minimum rotation angle was $\zeta = 0.01$ for both methods. Figure 6 presents the total number of rotations and corresponding SIR index measured for Laplacian distribution versus the number of components. As indicated in Section 3.4, the QICA method requires approximately 2 times fewer rotations per sweep than the JADE algorithm, but it also needs more sweeps to converge. Therefore, the total number of rotations is similar for both methods.

4.4 Computational complexity

Convergence rate measured by the number of iterations or data rotations is not the only factor that determines the speed of the algorithms. In fact, the computational cost of the single iteration, as well as the total execution time, is different for each method. Figure 7 presents the average execution times measured for various data sizes and algorithms. Due to significant differences in the implementation of the compared algorithms, we analyze both the execution time of an initialization stage and the time of a single iteration.

It can be seen in Fig. 7a that the initialization stage of the QICA algorithm is identical to that of the FastICA approach. Since it only consists in whitening the input data, it is much simpler and less memory demanding than the initialization stage of the JADE method. In

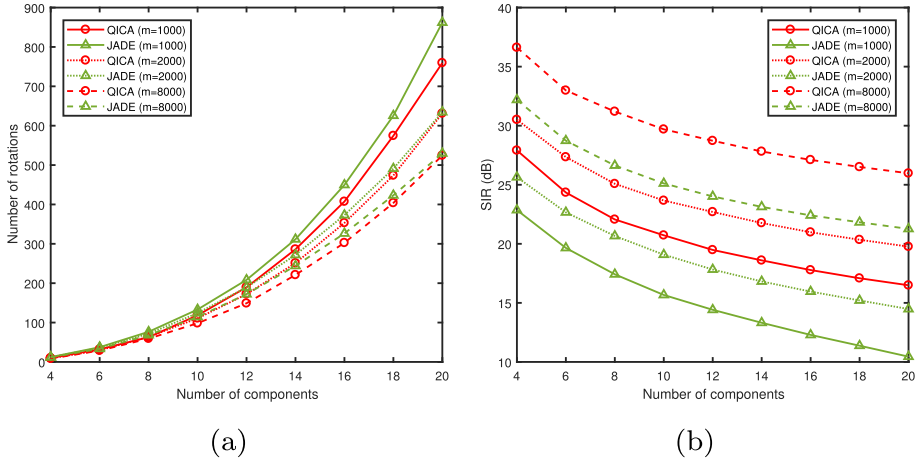

Fig. 6 Comparison of the QICA and JADE algorithms for Laplacian distribution and various numbers of components. **a** The number of rotations needed to achieve convergence. **b** SIR index measured after reaching the convergence

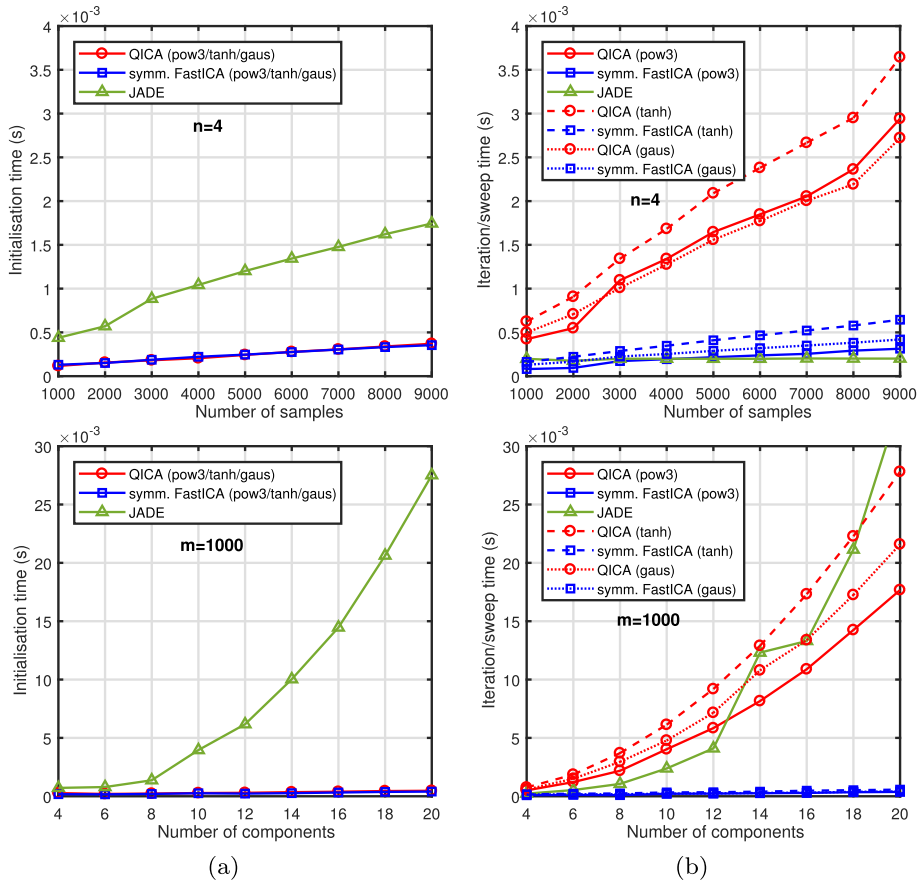

Fig. 7 Execution times obtained for QICA, symmetric FastICA, and JADE algorithms for various numbers of samples with fixed $n = 4$ (top) and various numbers of components with fixed $m = 1000$ (bottom). **a** Initialization time. **b** Average time of single iteration/sweep

the case of the JADE method, apart from whitening the input data, it is also necessary to estimate $n(n+1)/2$ cumulant matrices each of size $n \times n$. Obviously, these operations involve the processing of the whole data set of size $n \times m$ samples, but they are performed only once. Thus, the memory requirement for the JADE method is of order $O(nm + n^4)$ as opposed to $O(nm)$ for the QICA method.

It can be verified in Fig. 7b that the execution time of a single sweep of the QICA method increases quadratically with the number of components and linearly with the number of samples. So our method is characterized by the computational complexity of order $O(n^2 m)$. The computation time of a single iteration of the JADE algorithm also increases quadratically with the number of components, but it is constant with respect to the number of samples. This is because it consists in updating the cumulant matrices only, which does not involve any explicit transformation of the input signals. Thus, its complexity is of order $O(n^2)$.

Although the QICA method requires fewer rotations per single sweep than the JADE algorithm (or the same number in total), these rotations are more complex, and similarly to the FastICA method, in every sweep all data samples must be transformed. As we see in Fig. 7b, for a relatively small number of samples and a large number of components, the average execution time of the single iteration of the proposed method is similar or even

smaller than that of the JADE method. On the other hand, we also see that even for such a small number components as $n = 4$ the single iteration time of the QICA method is about six times longer than that of the symmetric FastICA approach.

Although the contrasts for the QICA and FastICA are slightly different, both methods maximize the negentropy measure which involves the evaluation of the same non-linear functions. Regardless of the implementation, both methods must, at some point, compute these functions and/or their derivatives. In the case of the QICA approach, for each rotation, three functions: $\bar{g}$, $\bar{g}'$, and $\bar{g}''$ have to be evaluated twice per every four-dimensional subproblem. Thus, at each sweep, we have $3nm(n-2)$ function evaluations in total. Please note that these operations can be redundant when the rotation angles are approximately close to zero. The FastICA approach requires evaluating the functions g' and g'' only once per component [9], so each iteration is related to $2nm$ evaluations in total.

The computational complexity of the proposed method can be reduced by taking into account the specific structure of the matrices $\mathbf{M}^{\pm}(\mathbf{u})$. In this case, the matrix-vector multiplication can be implemented using only 8 real multiplications [47]. On the other hand, multiplication of the 4×4 matrices can be implemented more efficiently on specific hardware architectures. Our method consists in decomposing high-dimensional BSS problem into four-dimensional subproblems. So it well suits modern GPUs and CPUs which support processing four-element vectors. In particular, typical GPUs are able to read 128 bits in one cycle and one instruction. Hence, accessing four-element vector-data can be faster than accessing four scalar elements (i.e., 32-bit single-precision floating-point numbers). Moreover, for some architectures (AMD GPUs), it is essential to organize data into four-element vectors to maximize efficiency [48]. Please note that in the case of the CPU-based implementation it is also possible to use 128-bit SSE/AVX instructions. Other ICA algorithms can be expressed in terms of operations on four-element vectors, but this is unnatural, is difficult, and results in suboptimal implementations compared to our approach.

5 Conclusion

We have shown that the quaternionic factorizations of 4×4 orthogonal matrices can be successfully used in the Jacobi ICA estimation framework for even $n \geq 4$. Namely, a novel ICA method has been developed, which, on the one hand, follows the Jacobi-based framework and, on the other hand, utilizes the contrast function based on negentropy approximation. We have shown that the optimization task can be reduced to the problem of a one-dimensional search on a unit circle and solved using the Newton-Raphson method.

The proposed algorithm outperforms the FastICA approach in terms of separation quality if the observation mixtures contain one or more Gaussian sources. It also shows a faster convergence rate than FastICA approach for most PDFs. Unlike the JADE algorithm, the proposed method can be adjusted to match a given distribution by selecting an appropriate non-linear function that approximates the negentropy. The developed method tends to be more computationally demanding than the state of art approaches, i.e., symmetric FastICA and JADE algorithm. However, we believe that the method can be optimized by taking into account specific structure of the quaternion-based rotation matrices.

Future works include rigorous stability analysis of the designed algorithm, further optimizations, and developing a GPU-optimized implementation, where the utilization of the four-dimensional rotations seems to be especially promising.

Appendix

The derivatives of the quaternionic rotation matrices with respect to the components of the quaternion are given by:

$$\frac{\partial \mathbf{M}^{-}(\mathbf{u})}{\partial u_0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \frac{\partial \mathbf{M}^{-}(\mathbf{u})}{\partial u_1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\frac{\partial \mathbf{M}^{-}(\mathbf{u})}{\partial u_2} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, \frac{\partial \mathbf{M}^{-}(\mathbf{u})}{\partial u_3} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix},$$

$$\frac{\partial \mathbf{M}^{+}(\mathbf{u})}{\partial u_0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \frac{\partial \mathbf{M}^{+}(\mathbf{u})}{\partial u_1} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\frac{\partial \mathbf{M}^{+}(\mathbf{u})}{\partial u_2} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, \frac{\partial \mathbf{M}^{+}(\mathbf{u})}{\partial u_3} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Abbreviations

ICA: Independent component analysis; JADE: Joint approximate diagonalization of eigenmatrices; BSS: Blind signal separation; EEG: Electroencephalography; PCA: Principal component analysis; QICA: Quaternionic independent component analysis; GG: Generalized Gaussian; SIR: Signal to interference ratio; GPU: Graphics processing unit; CPU: Central processing unit; SSE: Streaming SIMD extensions; AVX: Advanced vector extensions; PDF: Probability density function

Acknowledgements

The author would like to thank the anonymous reviewers for their insightful comments that helped to improve this article.

Authors' contributions

AB is the sole author of this work and approved the final manuscript.

Funding

This work was supported by the Bialystok University of Technology under the Grant WZ/WI-IIT/4/2020.

Availability of data and materials

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Consent for publication

Not applicable.

Competing interests

The author declares that there are no competing interests.

Received: 6 February 2020 Accepted: 18 August 2020
Published online: 04 September 2020

References

1. G. Naik, W. Wang, *Blind Source Separation: Advances in Theory, Algorithms and Applications*. (Springer, Berlin, Heidelberg, 2014)
2. P. Comon, Independent component analysis, a new concept?. Signal Process. **36**(3), 287–314 (1994)
3. J. Virta, K. Nordhausen, Estimating the number of signals using principal component analysis. Stat. **8**(1), 231 (2019)
4. M. Zibulevsky, B. A. Pearlmutter, Blind source separation by sparse decomposition in a signal dictionary. Neural Comput. **13**(4), 863–882 (2001)
5. Y. Li, A. Cichocki, S. Amari, Analysis of sparse representation and blind source separation. Neural Comput. **16**(6), 1193–1234 (2004)
6. M. Kleinsteuber, H. Shen, Blind source separation with compressively sensed linear mixtures. IEEE Signal Process. Lett. **19**(2), 107–110 (2012)
7. M. Rani, S. B. Dhok, R. B. Deshmukh, A systematic review of compressive sensing: concepts, implementations and applications. IEEE Access. **6**, 4875–4894 (2018)
8. A. Hyvärinen, E. Oja, Independent component analysis: algorithms and applications. Neural Netw. **13**(4), 411–430 (2000)
9. A. Hyvärinen, Fast and robust fixed-point algorithms for independent component analysis. IEEE Trans. Neural Netw. **10**(3), 626–634 (1999)
10. P. Comon, C. Jutten (eds.), *Handbook of Blind Source Separation. Independent Component Analysis and Applications* (Academic Press, Oxford, 2010)
11. X. Yu, D. Hu, J. Xu, *Blind Source Separation - Theory and Applications*. (Wiley, Singapore, 2014)
12. G. Chabriel, M. Kleinsteuber, E. Moreau, H. Shen, P. Tichavsky, A. Yeredor, Joint matrices decompositions and blind source separation: a survey of methods, identification, and applications. IEEE Signal Process. Mag. **31**(3), 34–43 (2014)
13. A. J. Bell, T. J. Sejnowski, An information-maximization approach to blind separation and blind deconvolution. Neural Comput. **7**(6), 1129–1159 (1995)
14. T.-W. Lee, M. Girolami, T. J. Sejnowski, Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. Neural Comput. **11**(2), 417–441 (1999)
15. J. F. Cardoso, A. Souloumiac, Blind beamforming for non-Gaussian signals. IEE Proc. F Radar Signal Process. **140**(6), 362–370 (1993)
16. J. F. Cardoso, High-order contrasts for independent component analysis. Neural Comput. **11**(1), 157–192 (1999)
17. P. Ablin, J. Cardoso, A. Gramfort, in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Faster ICA under orthogonal constraint (IEEE, Calgary, 2018), pp. 4464–4468
18. J. Miettinen, K. Nordhausen, S. Taskinen, fICA: FastICA algorithms and their improved variants. R J. **10**, 148–158 (2019)
19. K. Nordhausen, P. Ilmonen, A. Mandal, H. Oja, E. Ollila, in *Proc. 19th European Signal Processing Conference (EUSIPCO)*, Deflation-based fastICA reloaded (IEEE, Barcelona, 2011), pp. 1854–1858
20. A. Hyvärinen, The fixed-point algorithm and maximum likelihood estimation for independent component analysis. Neural Process. Lett. **10**(1), 1–5 (1999)
21. M. Plauth, F. Feinbube, P. Tröger, A. Polze, in *Proc. 15th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Fast ICA on modern GPU architectures (IEEE, Hong Kong, 2014), pp. 69–75
22. E. G. Learned-Miller, J. W. Fisher, ICA using spacings estimates of entropy. J. Mach. Learn. Res. **4**, 1271–1295 (2003)
23. J. Miettinen, S. Taskinen, K. Nordhausen, H. Oja, Fourth moments and independent component analysis. Stat. Sci. **30**, 372–390 (2015)
24. A. Hyvärinen, in *Proc. IEEE Signal Processing Society Workshop. Neural Networks for Signal Processing VII*, One-unit contrast functions for independent component analysis: a statistical analysis (IEEE, Amelia Island, 1997), pp. 388–397
25. N. Mackey, Hamilton and Jacobi meet again: quaternions and the eigenvalue problem. SIAM J. Matrix Anal. Appl. **16**(2), 421–435 (1995)
26. A. Borowicz, in *Proc. Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, On using quaternionic rotations for indpendent component analysis (IEEE, Poznań, Poland, 2018), pp. 114–119
27. H. G. Baker, Quaternions and orthogonal 4x4 real matrices (1996). http://archive.gamedev.net/archive/reference/articles/article428.html. Accessed 09 Jan 2020
28. G. H. Golub, C. F. Van Loan, *Matrix Computations*. (Johns Hopkins University Press, USA, 2013)
29. A. Hyvärinen, in *Proc. Conference on Advances in Neural Information Processing Systems 10*, New approximations of differential entropy for independent component analysis and projection pursuit (MIT Press, Denver, 1997), pp. 273–279
30. P. Tichavsky, Z. Koldovsky, E. Oja, Performance analysis of the FastICA algorithm and Cramér-Rao bounds for linear independent component analysis. IEEE Trans. Signal Process. **54**(4), 1189–1203 (2006)
31. S. C. Douglas, S. Amari, S. Y. Kung, On gradient adaptation with unit-norm constraints. IEEE Trans. Signal Process. **48**(6), 1843–1847 (2000)
32. W. Murray, *Newton-Type Methods, Wiley Encyclopedia of Operations Research and Management Science*. (Wiley, Hoboken, 2011)
33. W. Ouedraogo, A. Souloumiac, C. Jutten, in *Proc. Latent Variable Analysis and Signal Separation (LVA/ICA)*, Non-negative independent component analysis algorithm based on 2D Givens rotations and a Newton optimization (Springer, Berlin, Heidelberg, 2010), pp. 522–529
34. H. Faßbender, D. S. Mackey, N. Mackey, Hamilton and Jacobi come full circle: Jacobi algorithms for structured Hamiltonian eigenproblems. Linear Algebra Appl. **332-334**, 37–80 (2001)
35. D. S. Mackey, N. Mackey, D. M. Dunlavy, Structure preserving algorithms for perplectic eigenproblems. ELA. Electron. J. Linear Algebra. **13**, 10–39 (2005)
36. M. Parfieniuk, in *Proc. International Conference on Parallel Processing and Applied Mathematics (PPAM)*, A parallel factorization for generating orthogonal matrices (Springer, Bialystok, Poland, 2019), pp. 567–578
37. D. Martin, C. Fowlkes, D. Tal, J. Malik, in *Proc. 8th Int'l Conf. Computer Vision, vol. 2*, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics (IEEE, Vancouver, 2001), pp. 416–423

38. H. Gävert, J. Hurri, J. Särelä, A. Hyvärinen, Matlab FastICA v 2.5 (2005). http://research.ics.aalto.fi/ica/fastica/code/dlcode.shtml. Accessed 09 Jan 2020
39. A. Delorme, S. Makeig, EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. J. Neurosci. Methods. **134**, 9–21 (2004)
40. J. Miettinen, K. Nordhausen, S. Taskinen, Blind source separation based on joint diagonalization in R: the packages JADE and BSSasymp. J. Stat. Softw. **76**(2), 1–31 (2017)
41. (International Telecommunication Union - Telecommunication Standardization Sector, Geneva, 1998). http://handle.itu.int/11.1002/1000/4412. Accessed 09 Jan 2020
42. Z. Koldovský, P. Tichavsky, E. Oja, Efficient variant of algorithm FastICA for independent component analysis attaining the Cramér-Rao lower bound. IEEE Trans. Neural Netw. **17**, 1265–77 (2006)
43. P. Tichavsky, Z. Koldovský, Optimal pairing of signal components separated by blind techniques. IEEE Signal Process. Lett. **11**, 119–122 (2004)
44. V. Zarzoso, P. Comon, Robust independent component analysis by iterative maximization of the kurtosis contrast with algebraic optimal step size. IEEE Trans. Neural Netw. **21**(2), 248–261 (2010)
45. G. Blanchard, M. Kawanabe, M. Sugiyama, V. Spokoiny, K. Müller, In search of non-Gaussian components of a high-dimensional distribution. J. Mach. Learn. Res. **7**, 247–282 (2006)
46. H. Sasaki, G. Niu, M. Sugiyama, in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, vol 51*, Non-Gaussian component analysis with log-density gradient estimation (Proceedings of Machine Learning Research - PMLR, Cadiz, 2016), pp. 1177–1185. http://proceedings.mlr.press/v51/sasaki16.html
47. T. D. Howel, J.-C. Lafon, The complexity of quaternion product. Technical Report TR 75-245, Cornell University, Department of Computer Science (1975)
48. L. Buatois, G. Caumon, B. Lévy, in *High Performance Computing and Communications, Lecture Notes in Computer Science*, Concurrent number cruncher: an efficient sparse linear solver on the GPU, vol. 4782 (Springer, Berlin, 2007), pp. 358–371

Publisher's Note