DOI: 10.15439/2023F3368
ISSN 2300-5963 ACSIS, Vol. 35

L1-Norm Principal Component Analysis Using Quaternion Rotations

Adam Borowicz
0000-0003-0320-5530
Faculty of Computer Science, Bialystok University of Technology
Wiejska str. 45A, 15-351 Bialystok, Poland
Email: a.browicz@pb.edu.pl

***Abstract*—Principal component analysis (PCA) based on L1-norm has drawn growing interest in recent years. It is especially popular in the machine learning and pattern recognition communities for its robustness to outliers. Although optimal algorithms for L1-norm maximization exist, they have very high computational complexity and can be used for evaluation purposes only. In practice, only approximate techniques have been considered so far. Currently, the most popular method is the bit-flipping technique, where the L1-norm maximization is viewed as a combinatorial problem over the binary field. Recently, we proposed exhaustive, but faster algorithm [1] based on two-dimensional Jacobi rotations that also offer high accuracy. In this paper, we develop a novel variant of this method that uses three-dimensional rotations and quaternion algebra. Our experiments show that the proposed approach offers higher accuracy than other approximate algorithms, but at the expense of the additional computational cost. However, for large datasets, the cost is still lower than that of the bit-flipping technique.**

I. Introduction

PRINCIPAL component analysis (PCA) is a method for multivariate data analysis with various uses, including dimensionality reduction, feature extraction and noise reduction [2]. The PCA tries to identify orthogonal directions, along which the data exhibit the greatest variability. The projections of the data on these directions are viewed as principal components. This technique is also referred as L2-PCA, because the data variability is measured using Frobenius norm (L2-norm on matrices). It can be easily implemented using, for example, singular value decomposition (SVD) of the observation data matrix [3]. However, it is also sensitive to the presence of outliers, i.e., data points that differ significantly from the other observations. In order to mitigate this drawback, several PCA techniques have been proposed that are based on L1-norm [4], [5], [6], [7]. Interestingly, the L1-norm criterion can also be used to perform independent component analysis (ICA) after data whitening [8], [9]. The L1-norm optimization problem can be formulated in several ways [5], but unlike in the case of the L2-PCA, these formulations are not equivalent. In this paper, we consider the following maximization:

$$\mathbf{Q}_{\text{L1}} = \underset{\substack{\mathbf{Q}=[\mathbf{q}_1,\ldots,\mathbf{q}_k]\in\mathbb{R}^{d\times k} \\ \mathbf{Q}^T\mathbf{Q}=\mathbf{I}_k}}{\operatorname{argmax}} \sum_{i=1}^{k} \|\mathbf{X}^T\mathbf{q}_i\|_1, \tag{1}$$

This work was supported by Bialystok University of Technology under the grant WZ/WI-IIT/5/2023

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ is a data matrix of rank $r_x \leq \min\{d, n\}$, consisting a sequence of observation vectors $(\mathbf{x}_i)_{i=1}^n$, and $\|.\|_1$ denotes L1-norm that return the sum of the absolute values of the individual entries. The parameter k denotes the number of the L1 principal components. Please note that the problem (1) is not scalable, i.e. it can not be translated into a sequence of the one-unit problems simply by projecting the data-matrix onto the null-space of the previous solution as in the L2-PCA algorithms. Furthermore, absolute value function is non-differentiable. For these reasons, obtaining the exact solution is a rather challenging task. In [5] it was shown that, if $\mathbf{XB}_{\text{opt}} \overset{\text{SVD}}{=} \mathbf{U}\Sigma\mathbf{V}^T$, and

$$\mathbf{B}_{\text{opt}} = \underset{\mathbf{B}\in\{\pm1\}^{n\times k}}{\operatorname{argmax}} \|\mathbf{XB}\|_*, \tag{2}$$

where $\|.\|_*$ denotes nuclear norm, then $\mathbf{Q}_{\text{L1}} = \mathbf{UV}^T$ is the optimal solution to (1). Therefore, the L1-norm maximization can be viewed as a combinatorial problem over the binary field. Unfortunately, the exhaustive search algorithm [5] has complexity $\mathcal{O}(n^{dk-k+1})$ and is difficult to use in practice. A faster, yet suboptimal, version of this approach is based on consecutive bit-flipping operations [6]. Its time complexity is of order $\mathcal{O}(nd\min\{n,d\} + n^2(k^4 + dk^2) + ndk^3)$, which can still be prohibitive for large data sizes. The most computationally efficient algorithm based on the fixed-point (FP) iterations was developed earlier in [4]. Unfortunately, it is rather inaccurate.

Recently, we proposed two L1-PCA algorithms [1] based on the Jacobi estimation framework. This framework is commonly used for diagonalizing symmetric matrices [3], [10] through the two-dimensional (plane) rotations. It also found applications in data-driven algorithms [11], [12] for iterative transformations of multi-dimensional data. It was shown in [1] that the Jacobi-based L1-PCA approaches provide high accuracy as compared to the existing suboptimal algorithms. They are also considerably faster than currently the most accurate method based on bit-flipping. In this paper, we propose to replace the conventional Jacobi rotations with higher-dimensional quaternion-based rotations. It is expected that, in this way, the convergence properties of an algorithm can be improved. A similar approach has been proposed in work [13] where we used quaternionic factorization of the 4×4 orthogonal matrices and Newton-Raphson iterative scheme to solve the ICA problem. Here, we present a simpler approach based on three-dimensional rotations to solve the L1-norm maximization problem. When compared to our previous method [1], a novel algorithm offers a higher probability of finding a solution that is closer to the optimal one at the expense of the additional computational cost. However, our experiments show that for large datasets, this cost is still lower than that of the bit-flipping method.

II. Preliminaries on quaternions

A quaternion $Q \in \mathbb{H}$ can be represented using the rectangular form as follows [14]:

$$Q = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3, \quad q_0, q_1, q_2, q_3 \in \mathbb{R}, \tag{3}$$

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:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1. \tag{4}$$

It is associative and distributes over vector addition, but it is not commutative.

When describing properties of the quaternions it is convenient to express them as the combination of a scalar part, $q_0 \in \mathbb{R}$, and a vector part, $\mathbf{q} = [q_1, q_2, q_3]^T \in \mathbb{R}^3$: $Q = [\![q_0, \mathbf{q}]\!]$. For example, the conjugate of Q can be written as $\bar{Q} = [\![q_0, -\mathbf{q}]\!]$, and the norm (modulus), is given by:

$$|Q| = \sqrt{q_0^2 + ||\mathbf{q}||^2}. \tag{5}$$

For non-null quaternion, the inverse is defined as follows:

$$Q^{-1} = \bar{Q}|Q|^{-2}, \quad QQ^{-1} = Q^{-1}Q = 1. \tag{6}$$

Since

$$e^Q = \sum_{k=0}^{\infty} \frac{Q^k}{k!} = e^{q_0} \left[\!\!\left[\cos ||\mathbf{q}||, \frac{\mathbf{q}}{||\mathbf{q}||} \sin ||\mathbf{q}||\right]\!\!\right] \tag{7}$$

every quaternion Q can also be expressed in an exponential (polar) form:

$$Q = |Q| e^{\theta \mathbf{q}/||\mathbf{q}||} = |Q| \left[\!\!\left[\cos\theta, \frac{\mathbf{q}}{||\mathbf{q}||}\sin\theta\right]\!\!\right], \tag{8}$$

where $0 \leq \theta < 2\pi$ is an angle such that:

$$\cos\theta = \frac{q_0}{|Q|}, \quad \sin\theta = \frac{||\mathbf{q}||}{|Q|}. \tag{9}$$

This form is especially useful, because it allows us to express rotation in SO(3), The notation SO(d) denotes special orthogonal group in a d-dimensional Euclidean space, consisting all orthogonal matrices of determinant 1. Let $P = [\![0, \mathbf{p}]\!]$ be a pure quaternion that corresponds to a vector $\mathbf{p} \in \mathbb{R}^3$ and

$$U = u_0 + \boldsymbol{i}u_1 + \boldsymbol{j}u_2 + \boldsymbol{k}u_3 = \left[\!\!\left[\cos\frac{\theta}{2}, \mathbf{u}\sin\frac{\theta}{2}\right]\!\!\right], \tag{10}$$

be a unit-norm quaternion, where $\mathbf{u} = [u_1, u_2, u_3]^T \in \mathbb{R}^3$ is a unit vector indicating the direction of an axis of rotation, and an angle $0 \leq \theta < 2\pi$ is the magnitude of the rotation about the axis. Then the rotation of the vector $\mathbf{p}$ with an angle θ around a vector $\mathbf{u}$ can be expressed as follows:

$$P' = [\![0, \mathbf{p}']\!] = UPU^{-1} = UP\bar{U}. \tag{11}$$

Above operation can be expressed equivalently using matrix/vector multiplication by $\mathbf{p}' = \mathbf{R}(U)\mathbf{p}$, where:

$$\mathbf{R}(U) = \tag{12}$$
$$\begin{bmatrix} 1 - 2u_2^2 - 2u_3^2 & 2u_1u_2 + 2u_0u_3 & 2u_1u_3 - 2u_0u_2 \\ 2u_1u_2 - 2u_0u_3 & 1 - 2u_1^2 - 2u_3^2 & 2u_2u_3 + 2u_0u_1 \\ 2u_1u_3 + 2u_0u_2 & 2u_2u_3 - 2u_0u_1 & 1 - 2u_1^2 - 2u_2^2 \end{bmatrix},$$

is a rotation matrix. Theoretically, any rotation matrix can also be constructed using Euler angles, as a product of the three rotation matrices about the axes of the fixed coordinate system. Unfortunately, the Euler angles differing in many ways can give the same rotation matrix. In our case, this leads to multiple cost function calculations for the same point. Since the representation (10) corresponds almost uniquely to a given rotation matrix, such ambiguities can easily be avoided when working with quaternions.

III. Methods

A. Jacobi-based estimation framework

In the conventional Jacobi estimation framework [11], [1] the solution matrix is considered to be a product of the rotations in SO(2). These rotations are applied successively to the data matrix so that some objective function is optimized. For instance, in our previous work [1], the L1-norm metric is maximized as follows:

$$\mathbf{X}^{(t)} = \mathbf{G}(p_t, q_t, \theta_t)\mathbf{X}^{(t-1)}, \quad t = 1, 2, ..., \tag{13}$$

with $\mathbf{X}^{(0)} = \mathbf{WX}$, where $\mathbf{W} \in \mathbb{R}^{d \times d}$ is an arbitrary orthonormal matrix defining initialization point. The matrix $\mathbf{G}(p, q, \theta)$ represents Jacobi/Givens rotation [3] by the θ angle in the (p, q) plane, i.e.:

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

where p, q are two integers such that such $1 \leq p < q \leq d$. Thus, the solution matrix $\hat{\mathbf{Q}}_{\text{L1}} \in \mathbb{R}^{d \times k}$ is given by:

$$\hat{\mathbf{Q}}_{\text{L1}} = \mathbf{W}^T \left[\overset{\curvearrowright}{\prod_t} \mathbf{G}(p_t, q_t, \theta_t)^T\right]_{*1:k}, \tag{15}$$

where $[\,.\,]_{*1:k}$ denotes the first k columns of an argument matrix. All possible rotations represented by pairs (p_t, q_t) are arranged in so-called sweeps. These sweeps are repeated cyclically until the maximum number of iterations is reached or when, for all rotations in the current sweep, we have $|\theta_t| \approx 0$. In fact, any rotation order is allowed [12], [15], but most frequently a row-cycling ordering is used as presented in Tab. I. For a fixed arrangement of the plane rotations, each

TABLE I: Row-cycling ordering for $d = 3$.

sweep no.	1			2			...
t	1	2	3	4	5	6	...
(p_t, q_t)	(1,2)	(1,3)	(2,3)	(1,2)	(1,3)	(2,3)	...

transformation in (13) depends on a single parameter θ_t, and hence d-dimensional optimization problem can be reduced to the sequence of $d(d-1)/2$ simpler one-dimensional subproblems per sweep. Let us denote by $\hat{x}_{ij}^{(t)}(\theta)$ the (i,j)-th entry of the data matrix (13) evaluated for the angle $\theta_t = \theta$. Then, the 'local' L1-norm maximization problem at tth rotation can be defined as follows:

$$\theta_t = \underset{-\pi/2 \leq \theta < \pi/2}{\operatorname{argmax}} \sum_{\substack{i \in \{p_t, q_t\} \\ i \leq k}} \sum_{j=1}^{n} \left| \hat{x}_{ij}^{(t)}(\theta) \right|. \tag{16}$$

Since we are interested in finding only the first k principal components, the outer summation range in (16) covers only indices less than or equal to k. In this way, the rotations that would have to be performed entirely in the null-space can simply be omitted. Also note that, the matrix (14) modifies only the rows p_t, q_t of the data matrix $\mathbf{X}^{(t-1)}$, so that the summation coefficients can be computed directly as

$$\hat{x}_{p_t j}^{(t)}(\theta) = x_{p_t j}^{(t-1)} \cos\theta + x_{q_t j}^{(t-1)} \sin\theta, \tag{17}$$
$$\hat{x}_{q_t j}^{(t)}(\theta) = x_{q_t j}^{(t-1)} \cos\theta - x_{p_t j}^{(t-1)} \sin\theta. \tag{18}$$

In work [1], we proposed two methods for solving (16). The first one performs exhaustive angle search, and the second one uses a differentiable approximation for absolute value function and calculates the rotation angles using the simplified Newton method. In this paper, we consider only the exhaustive algorithm due to its simplicity and high accuracy. Namely, the objective function in (16) is evaluated at the set of equidistant points, i.e.: $\{-\pi/2 + i\pi/m : i = 0, 1, ..., m-1\}$. We call this set the dictionary. The parameter m is an integer value controlling an angular resolution, i.e., the smallest non-zero angle that is used to represent rotation. Theoretically, greater the value of m, the higher angular resolution and better accuracy of the optimization. However, by increasing this value, we do not prevent the method from falling into local optima.

B. Proposed method

Key idea of the proposed method is to modify the Jacobi estimation framework by replacing rotations in SO(2) with rotations in SO(3). Please note that the conventional approach can guarantee a global convergence only for $d = 2$. For higher-dimensional problems, the Jacobi rotations are performed sequentially. Therefore, we may easily get trapped in local maximum due to non-convexity of the cost function. Similarly, rotations in SO(3) do not guarantee finding a global optimum for $d > 3$, however, such replacement can increase frequency with which the method finds an optimal solution. Furthermore, with the higher-dimensional rotations, more data samples are used when computing the local objective functions, and thus these functions should be smoother, which may result in a faster convergence. Namely, we propose to replace (14) with a quaternion based matrix:

$$\mathbf{R}(p, q, r, U) = \tag{19}$$
$$\begin{bmatrix} \mathbf{I}_{p-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & u_{11} & \mathbf{0} & u_{12} & \mathbf{0} & u_{13} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{q-p-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & u_{21} & \mathbf{0} & u_{22} & \mathbf{0} & u_{23} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{r-q-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & u_{31} & \mathbf{0} & u_{32} & \mathbf{0} & u_{33} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{d-r-1} \end{bmatrix},$$

where $1 \leq p < q < r \leq d$ and the coefficient u_{ij} is (i,j)-th entry of the matrix (12) computed for the quaternion:

$$U(\lambda, \phi, \theta) = \left[\!\left[\cos\frac{\theta}{2}, \sin\frac{\theta}{2} \mathbf{u}(\lambda, \phi) \right]\!\right]. \tag{20}$$

Since the matrix (12) is orthogonal, the matrix (19) is also orthogonal. For convenience, the rotation axis is factorized using spherical coordinates, i.e.:

$$\mathbf{u}(\lambda, \phi) = [\cos\lambda \sin\phi,\ \sin\lambda \sin\phi,\ \cos\phi]^T, \tag{21}$$

where $0 \leq \lambda < 2\pi$ and $0 \leq \phi \leq \pi$ denotes azimuthal and polar angle, respectively. More formally, our optimization problem can be stated as follows:

$$(\lambda_t, \phi_t, \theta_t) = \underset{\substack{0 \leq \lambda < 2\pi \\ 0 \leq \phi \leq \pi \\ 0 \leq \theta < 2\pi}}{\operatorname{argmax}} \sum_{\substack{i \in \{p_t, q_t, r_t\} \\ i \leq k}} \sum_{j=1}^{n} \left| \hat{x}_{ij}^{(t)}(\lambda, \phi, \theta) \right|, \tag{22}$$

where $\hat{x}_{ij}^{(t)}(\lambda, \phi, \theta)$ denotes the (i,j)-th entry of transformed data matrix $\mathbf{R}(p_t, q_t, r_t, U)\mathbf{X}^{(t-1)}$. The coefficients $\hat{x}_{ij}^{(t)}(\lambda, \phi, \theta)$ for $i \in \{p_t, q_t, r_t\}$ can be stacked in the vector representing imaginary part of the following quaternion:

$$[\![0, \mathbf{x}_j^{(t)}(\lambda, \phi, \theta)]\!] = U(\lambda, \phi, \theta) X_j^{(t-1)} U(\lambda, \phi, \theta)^{-1}, \tag{23}$$
$$X_j^{(t-1)} = \left[\!\left[0, [x_{p_t j}^{(t-1)}, x_{q_t j}^{(t-1)}, x_{r_t j}^{(t-1)}]^T \right]\!\right]. \tag{24}$$

As before, the simplest solution to (22) is to use an exhaustive search method. Please note that there is not necessary to discretize the entire sphere because for a given vector (21) and an angle θ there is an opposite vector $-\mathbf{u}(\lambda, \phi)$ that generates the same rotation matrix for the angle $2\pi - \theta$. Let us denote by $\mathbb{A} = \{i\pi/m : i = 0, 1, ..., m-1\}$ and $\mathbb{B} = \{j\pi/m : j = 0, 1, ..., 2m-1\}$ the sets of equidistant points on interval $[0; \pi)$ and $[0; 2\pi)$, respectively. Then our spherical coordinate search dictionary can be defined as the following set:

$$\begin{aligned} \mathbb{D} = \{(\lambda, \phi, \theta) :\ & (\lambda, \phi, \theta) \in \mathbb{A} \times \mathbb{A} \times \mathbb{B} \wedge \\ & (\phi > 0 \vee \lambda = 0) \wedge (\phi = 0 \vee \theta > 0)\}, \end{aligned} \tag{25}$$

where $\times$ denotes Cartesian product. The condition in the second line removes from the set redundant coordinates and those for which the rotation matrix is equal to the identity matrix (for $\theta = 0$), except the point at the north pole.

TABLE II: Arrangements of rotation subspaces for various signal dimensionalities.

$d = 4$

t	(p_t, q_t)	(p_t, q_t, r_t)
1	(1, 2)	(1, 2, 3)
2	(1, 3)	(1, 2, 4)
3	(1, 4)	(2, 3, 4)
4	(2, 3)	...
5	(2, 4)	...
6	(3, 4)	...

$d = 5$

t	(p_t, q_t)	(p_t, q_t, r_t)
1	(1, 2)	(1, 2, 3)
2	(1, 3)	(1, 4, 5)
3	(1, 4)	(2, 4, 5)
4	(1, 5)	(3, 4, 5)
5	(2, 3)	...
6	(2, 4)	...
7	(2, 5)	...
8	(3, 4)	...
9	(3, 5)	...
10	(4, 5)	...

It can be verified that the cardinality of the set (25) is $l = 2m^3 - 3m^2 + 3m$. Obviously, for large m searching for the solution exhaustively may be unpractical, as the sequence length l grows rapidly with m. However, as we will show in the experimental section, the rotations in SO(3) can be represented with a much lower resolution than rotations in SO(2). In other words, the parameter m can be much smaller than that of the conventional Jacobi-based framework. Therefore, we can still use the exhaustive method at reasonable runtime.

Similarly to the conventional method, the consecutive rotations are organized in sweeps and repeated cyclically until convergence. However, since we deal with rotations in SO(3), a three-dimensional subspace must be defined for each rotation. Theoretically, for $d > 3$, the rotations can be performed in all possible subspaces defined as the 3-combinations of the row indices. However, we observed that to have convergence, not all combinations are needed. Namely, any combination (x, y, z) can be removed if all three pairs (x, y), (x, z) and (y, z) can also be found in other combinations. As presented in Tab. II, for $d = 4$ the combination $(1, 3, 4)$ was removed because there are the combinations $(1, 2, 3)$, $(1, 2, 4)$ and $(2, 3, 4)$ containing the pairs $(1, 3)$, $(1, 4)$ and $(3, 4)$, respectively. Such subsets of the combinations can also be obtained by joining plane rotations sharing the same dimensions. For example, two pairs (1,2) and (1,3) can be joined in the triple $(1, 2, 3)$, and the pair $(2, 3)$ can be removed as it is already present in the triple. In our simulations, we use this algorithm to generate arrangements presented in Tab. II for $d = 4, 5$. It can be verified that for $d \geq 3$ we have $n_r = \lceil d(d-2)/4 - \mathrm{mod}(d, 2) + 1 \rceil$ 3D rotations per sweep, where $\lceil . \rceil$ denotes ceiling operation. Similarly to Jacobi-based framework, there may exist other arrangements of the rotation subspaces, and some of them may be better than others. However, this issue is out of scope of this paper, and will be studied in a future work.

The pseudo-code of the proposed method is presented in Alg. 1. The sequence of triples defined in line 5 is defined according to the Tab. II. Please note that, at tth rotation, the matrix (19) modifies only the rows p, q, r. Therefore it is not necessary to compute it explicitly. In fact, only the matrices (12) are needed. Furthermore, they can be pre-computed for a given dictionary $\mathbb{D}$ once and used in subsequent iterations.

Algorithm 1 Pseudo-code of the proposed algorithm

```
Require:
    $\mathbf{X} \in \mathbb{R}^{d \times n}, \mathbf{W} \in \mathbb{R}^{d \times d}, \mathbf{W}\mathbf{W}^T = \mathbf{I}_d, k \leq \mathrm{rank}(\mathbf{X}), \mathbb{D}$
Ensure: $\mathbf{Q} \in \mathbb{R}^{d \times k}, \mathbf{Q}^T\mathbf{Q} = \mathbf{I}_k$

 1: $\mathbf{X}^{(0)} \leftarrow \mathbf{W}\mathbf{X}$
 2: $t \leftarrow 1$
 3: for $sweepNum = 1 : maxSweepNum$ do
 4:     $encore \leftarrow 0$
 5:     for $(p, q, r) = \{(1, 2, 3), ...\}$ do
 6:         $(\lambda_t, \phi_t, \theta_t) = \underset{(\lambda,\phi,\theta) \in \mathbb{D}}{\mathrm{argmax}} \sum_{\substack{i \in \{p,q,r\} \\ i \leq k}} \sum_{j=1}^{n} \left| \hat{x}_{ij}^{(t)}(\lambda, \phi, \theta) \right|$
 7:         $\mathbf{R}_{\text{opt}} \leftarrow \mathbf{R}(p, q, r, U(\lambda_t, \phi_t, \theta_t))$
 8:         $\mathbf{W} \leftarrow \mathbf{R}_{\text{opt}}\mathbf{W}$
 9:         $\mathbf{X}^{(t)} \leftarrow \mathbf{R}_{\text{opt}}\mathbf{X}^{(t-1)}$
10:         $t \leftarrow t + 1$
11:         if $\mathbf{R}_{\text{opt}} \neq \mathbf{I}$ then $encore \leftarrow 1$
12:     if $encore = 0$ then break
13: $\mathbf{Q} \leftarrow [\mathbf{W}^T]_{*1:k}$
```

IV. Experiments

The proposed method has been implemented and evaluated in the Matlab environment. For convenience, it was denoted as L1-JQ, which stands for Jacobi method with quaternion rotations. For comparative purposes we also evaluated three other approximate methods for maximization of the L1-norm: bit-flipping algorithm [6] (L1-BF), fixed-point iterations [4] (L1-FP), and Jacobi exhaustive method with SO(2) rotations [1] (L1-JEX). In the case of the L1-JEX method, the parameter m was set to 512. We verified empirically that, for this method, rotations at a smaller angle than $\pi/512$ are not statistically significant. In order to explore how the estimation error is affected by the angular resolution, the algorithm L1-JQ was evaluated for two different values of the parameter $m \in \{10, 20\}$. For all approaches, the identity matrix was used as the initialisation point.

A. Accuracy

The performance degradation ratio attained by the algorithms was measured using a similar procedure to that in [6]. Namely, the following metric was considered:

$$\Delta(\mathbf{Q}, \mathbf{X}) = \frac{\|\mathbf{X}^T\mathbf{Q}_{\text{L1}}\|_1 - \|\mathbf{X}^T\mathbf{Q}\|_1}{\|\mathbf{X}^T\mathbf{Q}_{\text{L1}}\|_1}, \tag{26}$$

where $\mathbf{Q}$ is the orthonormal matrix estimated using an evaluated method. Ideally, the matrix $\mathbf{Q}_{\text{L1}}$ should be the matrix obtained by an optimal L1-PCA algorithm [5], for the same data matrix. Unfortunately, the computational complexity of the optimal method is extremely high. Thus, such an approach is possible only for very small data sizes ($n \ll 100$). The presented method is intended for larger data sets. For these reasons, in this experiment, we replaced the matrix $\mathbf{Q}_{\text{L1}}$ with the matrix representing the best solution among all methods. In order to measure which method gives the best result most

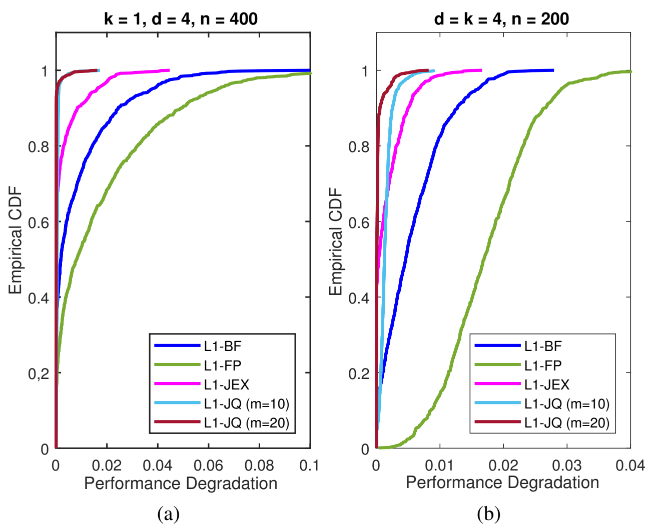

Fig. 1: Empirical CDF of performance degradation ratio estimated for various L1-PCA algorithms

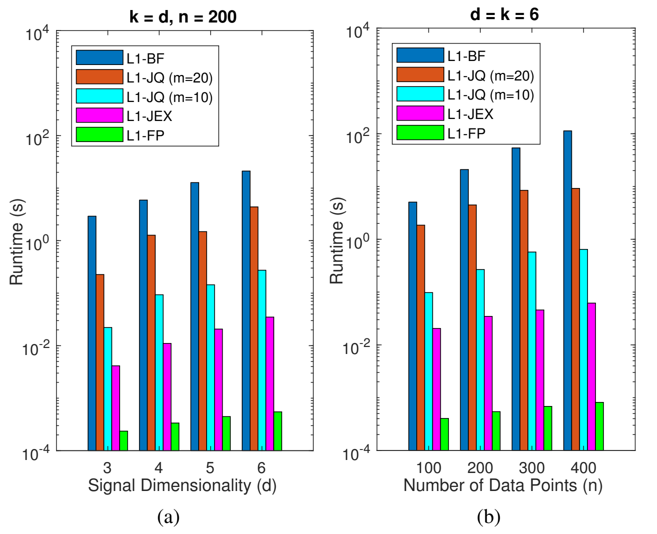

Fig. 2: Comparison of average runtimes (in seconds) measured for all methods and various data sizes across 100 Monte Carlo runs. (a) Runtime vs signal dimensionality. (b) Runtime vs number of observation samples.

frequently, the empirical cumulative distribution functions (ECDFs) were computed. The ECDFs are the fractions of the measurements (26) that are less than or equal to the specified values. Thus, the higher the value of the ECDF, the better accuracy. We considered two scenarios: the first one with $d = 4$, $k = 1$, $n = 400$, and the second one with $d = k = 4$, $n = 200$. In both scenarios, 1000 random data matrices were randomly generated with entries drawn independently from a Gaussian distribution $\mathcal{N}(0, 1)$ as in [6], [1]. The results are presented in Fig. 1. In the first scenario (on the left), the proposed approach with $m = 20$ gives a zero or close to zero value of the degradation ratio in about 95 percent of runs. This is the best score among all methods. In the case of the L1-JEX, L1-BF and L1-FP methods, these fractions are 65, 30, and 15 percent, respectively. The performance loss due to smaller dictionary size is rather not noticeable in this scenario. Both versions of the proposed method perform equally well regardless of the value of the parameter m. In the second scenario (Fig. 1b), we see that each method attains the best result less frequently. However, once again, the L1-JQ algorithm for $m = 20$ achieves lower values of the metric (26) more frequently than any other method. It gives the best or close to best result in about 75 percent of runs, while for the L1-JEX and L1-BF methods, these frequencies are 40 and 10 percent, respectively. The most significant performance loss can be seen for the L1-FP method. It comes from the fact that the L1-FP method for $k > 1$ is based on successive null-space projections that violate the non-scalability principle of the L1-PCA. We also see that the reduction in accuracy of the L1-JQ method due to the decrease in a value of the parameter m is more prominent. This reduction is especially noticeable in a frequency with which the method obtains the best solution. Please note that even if the proposed approach does not give the best solution most frequently, the metric (26) usually takes relatively small values. Namely, the degradation ratio attained by L1-JQ method with $m = 10$, computed with respect to the best solution, is with empirical probability 1 less than 0.008. Other methods attain significantly greater values of the degradation ratio. For example, the largest values of the metric (26) returned by the L1-JEX and L1-FB methods were 0.016 and 0.027 respectively.

B. *Execution time*

In order to compare the computational performance of the proposed algorithm with other methods, we measured their average execution times for various data sizes. The experiments were carried out on the system with AMD Ryzen 5 3550H processor. Once again, two scenarios have been considered. In the first one (Fig. 2a), we examined how the dimensionality of the signal affects the computation time. Here, we assumed that the number of data samples $n = 200$ and $k = d$. It can be seen that even for the higher angular resolution ($m = 20$), the proposed method is a faster than L1-BF algorithm. On the other hand, it is slower than the method based on SO(2) rotations, even when the angular resolution is low ($m = 10$). None of the methods can compete with the L1-FP method, which turns out to be the fastest approach. Also note that the execution time of the all rotational methods increases quite fast with the dimension number. It is not surprising, as the number of rotations per sweep increases quadratically with d.

In the second scenario (see Fig. 2b), we assumed that $k = d = 6$ and checked how the computation time is affected by the number of samples n. The results are similar to that of the previous case. It can be seen that all rotational methods are generally faster than the L1-BF algorithm. In addition their execution times increase linearly with the number of samples. This is serious improvement compared to the L1-BF algorithm, where the execution time increases quadratically with n. Our experiments clearly show that the precision of the

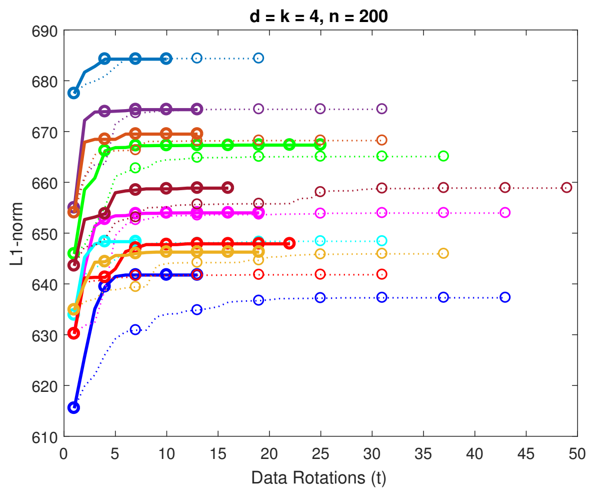

Fig. 3: Convergence curves obtained during 10 Monte Carlo runs for the L1-JQ method with $m = 20$ (solid lines) and the L1-JEX algorithm (dotted lines). Each Monte Carlo run is presented using different colour. The first data rotation in each sweep is depicted by circle.

L1-JQ method is paid for by increased computational burden. Nonetheless, its computational complexity can be still smaller than or at least comparable to that of the L1-BF algorithm. Also note that the rotational algorithms, including the proposed one, may not be the best choice for large d. For instance, when $d > n$, the L1-BF algorithm may offer better performance. However, a such scenario is rarely encountered in practice and thus less interesting.

In Fig. 3, we also show the convergence curves obtained for L1-JQ ($m = 20$) and L1-JEX methods. The L1-norm was measured after each data rotation for 10 independent Monte Carlo runs. The maximum number of sweeps was limited to 100, but none of the methods reached this limit. As we see, both methods converge in a relatively small number of sweeps (from 2 to 8), but the proposed method offers higher convergence rates. It also achieves higher values of the L1-norm more frequently, which is consistent with our previous findings. On the other hand, the computational cost of the single rotation of the proposed method is higher than that of the L1-JEX method. Thus, in overall, the L1-JEX method remains computationally more efficient.

V. Conclusion

In this paper, we proposed a novel version of the exhaustive Jacobi-based algorithm for maximization of the L1-norm. It was shown that the Jacobi rotations can be replaced by the quaternion-based rotations in SO(3). In this way, it is possible to increase the accuracy of the estimation at the expense of additional computational cost. Indeed, the simulation results show that the proposed method gives the best solution more frequently than other approximate methods. Although the algorithm was implemented using exhaustive search, the improvement in the accuracy was obtained for relatively small dictionary size. The results suggest that precision of angular representation of the rotations in higher-dimensions can be substantially lower than that of the two-dimensional rotations. Thus, a solution can be found exhaustively at a reasonable computational cost. Furthermore, for large datasets, the execution time of the proposed method is still smaller than that of the bit-flipping technique.

Future works include implementation optimizations, practical applications and more rigorous estimation error analysis. It could be especially interesting to establish the theoretical bounds of estimation error with respect to the precision of angular representation of the rotations.

References

[1] A. Borowicz, "Maximization of L1-norm using Jacobi rotations," in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022. doi: 10.23919/EUSIPCO55093.2022.9909924 pp. 1951–1955.

[2] I. T. Jolliffe, *Principal Component Analysis*. New York: Springer Verlag, 2002.

[3] G. Golub and C. Van Loan, *Matrix Computations*. USA: Johns Hopkins University Press, 2013.

[4] N. Kwak, "Principal component analysis based on L1-norm maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1672–1680, 2008. doi: 10.1109/TPAMI.2008.114

[5] P. P. Markopoulos, G. N. Karystinos, and D. A. Pados, "Optimal algorithms for L_1-subspace signal processing," *IEEE Transactions on Signal Processing*, vol. 62, no. 19, pp. 5046–5058, 2014. doi: 10.1109/TSP.2014.2338077

[6] P. P. Markopoulos, S. Kundu, S. Chamadia, and D. A. Pados, "Efficient L1-norm principal-component analysis via bit flipping," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4252–4264, 2017. doi: 10.1109/TSP.2017.2708023

[7] M. Dhanaraj and P. P. Markopoulos, "Novel algorithm for incremental L1-norm principal-component analysis," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018. doi: 10.23919/EUSIPCO.2018.8553239 pp. 2020–2024.

[8] R. Martın-Clemente and V. Zarzoso, "On the link between L1-PCA and ICA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 3, pp. 515–528, 2017. doi: 10.1109/TPAMI.2016.2557797

[9] A. Borowicz, "Independent component analysis based on Jacobi iterative framework and L1-norm criterion," in *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, 2022. doi: 10.15439/2022F157 pp. 305–312.

[10] J. Cardoso and A. Souloumiac, "Blind beamforming for non-Gaussian signals," *IEE Proceedings F - Radar and Signal Processing*, vol. 140, no. 6, pp. 362–370, 1993. doi: 10.1049/ip-f-2.1993.0054

[11] J. Cardoso, "High-order contrasts for independent component analysis," *Neural Computation*, vol. 11, no. 1, pp. 157–192, 1999. doi: 10.1162/089976699300016863

[12] W. Ouedraogo, A. Souloumiac, and C. Jutten, "Non-negative independent component analysis algorithm based on 2D Givens rotations and a Newton optimization," in *Latent Variable Analysis and Signal Separation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-15995-4 pp. 522–529.

[13] A. Borowicz, "Orthogonal approach to independent component analysis using quaternionic factorization," *EURASIP Journal on Advances in Signal Processing*, vol. 2020, no. 39, p. 23, September 2020. doi: 10.1186/s13634-020-00697-0

[14] J. B. Kuipers, *Quaternions and Rotation Sequences: a Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton, N.J.: Princeton University Press, 2002.

[15] M. Parfieniuk, "A parallel factorization for generating orthogonal matrices," in *International Conference on Parallel Processing and Applied Mathematics (PPAM) 2019*. Bialystok, Poland: Springer, 2019. doi: 10.1007/978-3-030-43229 pp. 567–578.