This paper was presented in the Sixth International Conference on Independent Component Analysis and Blind Source Separation held in Charleston, SC in March 2006. It is also published in the proceedings of the Conference: J. Rosca et al. (Eds.). ICA 2006, LNCS 3889, pp. 17, 2006.c Springer-Verlag Berlin Heidelberg 2006

Simple LU and QR based Non-Orthogonal Matrix Joint Diagonalization

Bijan Afsari

Institute for Systems Research and Department of Applied Mathematics
University of Maryland, College Park
20742 MD,USA
{bijan@glue.umd.edu}

Abstract. A class of simple Jacobi-type algorithms for non-orthogonal matrix joint diagonalization based on the LU or QR factorization is introduced. By appropriate parametrization of the underlying manifolds, i.e. using triangular and orthogonal Jacobi matrices we replace a high dimensional minimization problem by a sequence of simple one dimensional minimization problems. In addition, a new scale-invariant cost function for non-orthogonal joint diagonalization is employed. These algorithms are step-size free. Numerical simulations demonstrate the efficiency of the methods.

1 Introduction

The problem of matrix (approximate) Joint Diagonalization (JD) has found applications in many blind signal processing algorithms, see for example [4,6]. In one formulation it can be presented as: given a set of $n \times n$ symmetric matrices $\{C_i\}_{i=1}^N$ find a non-singular B such that the matrices $\{BC_iB^T\}_{i=1}^N$ are "as diagonal as possible". We call such a B a joint diagonalizer. In general diagonalization can happen only approximately. If B is restricted to the set of orthogonal $n \times n$ matrices $\mathrm{O}(n)$, the problem is referred to as orthogonal JD. Here, we are interested in non-orthogonal JD or NOJD, i.e. where B is in the set of non-singular $n \times n$ matrices $\mathrm{GL}(n)$. The reader is referred to [2,8] for further references on this subject. We remind that in [7] the NOJD problem is formulated differently.

A natural and effective cost function for orthogonal JD is [4]:

$$J_1(\Theta) = \sum_{i=1}^{n} \left\| \Theta C_i \Theta^T - \mathrm{diag}(\Theta C_i \Theta^T) \right\|_F^2 \tag{1}$$

where $\mathrm{diag}(X)$ is the diagonal part of matrix X, $\|.\|_F$ is the Frobenius norm and $\Theta \in \mathrm{O}(n)$. The algorithm introduced in [4], which is a part of the JADE algorithm, to minimize $J_1(\Theta)$ is an iterative minimization method using orthogonal

Jacobi matrices. This algorithm breaks the $\frac{n(n-1)}{2}$ dimensional minimization problem to a sequence of one dimensional minimization problems and also uses the group structure of $\mathrm{O}(n)$ by using multiplicative updates. Here, we extend this idea to the NOJD problem.

In many cases, such as noisy ICA, the joint diagonalizer sought can not assumed to be orthogonal. The NOJD problem is more challenging than orthogonal JD. It is natural to consider the NOJD as a minimization problem. Motivating physical problems such as ICA and BSS suggest that a good cost function J for NOJD should be invariant under permutation $\varPi$ and under scaling by a non-singular diagonal matrix $\varLambda$, i.e. $J(\varLambda\varPi B) = J(B)$[1]. If we extend J_1 to $\mathrm{GL}(n)$ then clearly $J_1(\varLambda B) \neq J_1(B)$. In fact by reducing the norm of B we can reduce $J_1(B)$ arbitrarily. In order to still use J_1 we can extend J_1 to a smaller subgroup of $\mathrm{GL}(n)$ such as $\mathrm{SL}(n)$ [3] or as in [3,8] we can restrict the "reduction" of J_1 only to the directions that do not correspond to multiplication by diagonal matrices. The latter results in updates of the form:

$$B_{k+1} = (I + \varDelta_k)B_k \tag{2}$$

where I is the $n \times n$ identity matrix, $\mathrm{diag}(\varDelta_k) = 0$ and $\varDelta_k$ is found such that $J_1(B_{k+1})$ is reduced at each step. This can be done, for example, from a gradient descent step as in [3]. One consequence of the update in (2) is that if the norm of $\varDelta_k$ is small enough [8] we can guarantee invertibility of B_{k+1}. Also if we choose $\varDelta_k$ to be a triangular matrix with $\mathrm{diag}(\varDelta_k) = 0$ and if $B_0 = I$ then $\det B_{k+1} = 1$ for all k and hence $\|B_{k+1}\|_2 \geq 1$. The significance of the latter is that it ensures that the cost J_1 is not reduced merely due to reducing the norm of B_k. In this article we consider triangular $\varDelta_k$ with only one non-zero element and we refer to $I + \varDelta_k$ as a Jacobi triangular matrix. In Section 2, we describe a class of NOJD methods using orthogonal and triangular Jacobi matrices which are based on the LU or QR factorization of the sought diagonalizer.

Another idea in devising NOJD algorithms is to use cost functions other than J_1. In [8,2] some different cost functions are mentioned. In [1] a scale-invariant cost function is used for NOJD which has the form:

$$J_2(B) = \sum_{i=1}^{N} \left\| C_i - B^{-1}\mathrm{diag}(BC_iB^T)B^{-T} \right\|_F^2 \tag{3}$$

Note that $J_2(\varLambda B) = J_2(B)$ for diagonal $\varLambda$ and that $J_2(\varTheta) = J_1(\varTheta)$ for $\varTheta \in \mathrm{O}(n)$. J_2 is the normalized version of J_1 in the sense that:

$$\frac{J_1(B)}{n\|B\|_F^4} \leq J_2(B) \leq n\|B^{-1}\|_F^4 J_1(B) \tag{4}$$

A drawback of J_2 is that in its calculation we need to compute the inverse of B. In Section 3 we propose a simple algorithm for minimization of J_2, too. In Section 4 we test the developed methods numerically and provide a comparison with one existing efficient NOJD method.

[1] Intuitively, we do not expect that $\varLambda\varPi C_i \varPi^T \varLambda$ can become more diagonal than C_i

2 Use of LU and QR factorizations in minimization of J_1

Any non-singular matrix B admits the LU factorization:

$$B = \Pi \Lambda L U \tag{5}$$

where Π is a permutation matrix, Λ is a non-singular diagonal matrix and L and U are $n \times n$ unit lower and upper triangular matrices, respectively. By a unit triangular matrix we mean a triangular matrix with diagonal elements of one [5]. The factorization in (5) exactly matches the invariances in NOJD. On the other hand the SVD factorization, for example, can not match this. Unit lower and upper triangular matrices of dimension n, form Lie groups denoted by $\mathcal{L}(n)$ and $\mathcal{U}(n)$, respectively. This fact simplifies the minimization process a lot. B also admits the QR factorization:

$$B = \Lambda L \Theta \tag{6}$$

where $\Theta \in \mathrm{O}(n)$ and $L \in \mathcal{L}(n)$. The idea is to find L and U separately in the LU form or L and Θ in the QR form such that J_1 is reduced at each step and repeat this till convergence. If the initial condition is the identity matrix, by construction, the solution's determinant will remain unity. Furthermore, we replace each of these $\frac{n(n-1)}{2}$ dimensional minimization problems by a sequence of simple one-dimensional problems by using triangular and orthogonal Jacobi matrices.

2.1 Triangular and Orthogonal Jacobi Matrices

A lower triangular Jacobi matrix with parameter a corresponding to the position $(i, j), i > j$ is denoted by $L_{ij}(a)$. $L_{ij}(a)$ is an element of $\mathcal{L}(n)$ whose $(i, j)^{th}$ entry is a and the rest of its off-diagonal entries are zero. In a similar fashion we define the upper triangular Jacobi matrix with parameter a corresponding to the position $(i, j), i < j$ and denote it by $U_{ij}(a)$. Any element of $\mathcal{L}(n)$ ($\mathcal{U}(n)$) can be represented as a product of lower (upper) triangular Jacobi matrices. We replace the problem of minimization of $J_1(L)$ with $L \in \mathcal{L}(n)$ which is a high dimensional problem with a sequence of simple one-dimensional quadratic problems of finding the parameter of triangular Jacobi matrices for minimizing J_1. The following simple proposition solves the one-dimensional problem. For brevity the proof is omitted.

Notation: (MATLAB's indexing) For matrix A, $A(k, index)$ where $index$ is a row vector denotes a row-vector whose elements are from the k^{th} row of A indexed by $index$. $A(index, l)$ is defined similarly. Specificality we are interested in vectors like $A(l, [1 : i-1, i+1 : n])$.

Proposition 1. *If $\hat{a}$ is such that:*

$$\hat{a} = -\frac{\sum_{i=1}^{N} C_i(k, [1 : l-1, l+1 : n]) C_i(l, [1 : l-1, l+1 : n])^T}{\sum_{i=1}^{N} \|C_i(k, [1 : l-1, l+1, : n])\|_F^2} \tag{7}$$

then: with $k < l$, $\hat{a}$ minimizes $J_1(L_{lk}(a))$ and with $k > l$, $\hat{a}$ minimizes $J_1(U_{lk}(a))$. If $\sum_{i=1}^{N} \|C_i(k, [1 : l-1, l+1, : n])\|_F^2 = 0$ set $\hat{a} = 0$, i.e. J_1 can not be reduced by that particular L_{lk} or U_{lk}.

Similarly, if $\Theta_{kl}(\theta)$ is the Jacobi rotation matrix corresponding to the position (k, l) and a counter-clockwise rotation by θ, then we have that[4]:

Proposition 2. *If θ_{kl} is such that $\boldsymbol{v} = [\cos 2\theta_{kl} \quad \sin 2\theta_{kl}]^T$ is a unit-norm eigen vector corresponding to the larger eigen value of the matrix $G_{kl}^T G_{kl}$ where G_{kl} is an $N \times 2$ matrix defined as $G_{kl}(i, 1) = C_i(k, k) - C_i(l, l)$ and $G_{kl}(i, 2) = 2C_i(k, l)$ for $1 \leq i \leq N$, then θ_{kl} minimizes $J_1(\Theta_{kl}(\theta))$.*

Based on these two propositions we can have two algorithms LUJ1D and QRJ1D. We juxtapose the two algorithms together:

Algorithm LUJ1D (QRJ1D):

1. `set` $B = I$. `set` ϵ.
2. U-phase (Q-Phase): `set` $U = I (\Theta = I)$. `for` $1 \leq l < k \leq n$:
   - Find $a_{lk} = \arg\min_a J_1(U_{lk}(a))$ ($\theta_{lk} = \arg\min_\theta J_1(\Theta_{lk}(\theta))$) from Proposition 1 (Proposition 2)
   - $C_i \leftarrow U_{lk}(a_{lk}) C_i U_{lk}(a_{lk})^T$ ($C_i \leftarrow \Theta_{lk}(\theta_{lk}) C_i \Theta_{lk}(\theta_{lk})^T$) `and` $U \leftarrow U_{lk}(a_{lk}) U$ ($\Theta \leftarrow \Theta_{lk}(\theta_{lk}) \Theta$)
3. L-phase (R-Phase): `set` $L = I$. `for` $1 \leq l < k \leq n$:
   - Find $a_{kl} = \arg\min_a J_1(L_{kl}(a))$ from Proposition 1
   - Update $C_i \leftarrow L_{kl}(a_{kl}) C_i L_{kl}(a_{kl})^T$ `and` $L \leftarrow L_{kl}(a_{kl}) L$
4. `if` $\|LU - I\|_F > \epsilon$ ($\|L\Theta - I\|_F > \epsilon$), `then` $B \leftarrow LUB$ ($B \leftarrow L\Theta B$) `and goto` 2, `else end`

We could use other stoping criteria such as keeping track of J_1 or J_2. The LUJ1D (as well as QRJ1D) algorithm is iterative in the sense that we find the L and U matrices repetitively, and it is sequential in the sense that the problem of finding a triangular matrix minimizing J_1 has been replaced (or approximated) by a finite sequence of one dimensional problems. Note that for updating C_i, the matrix multiplications can be realized by few vector scalings and vector additions. We also mention that, as other Jacobi methods, these methods are suitable for parallel implementation. For parallel implementation we may combine (multiply) all the lower triangular matrices corresponding to the same column and find the minimizing parameters of this new matrix at one shot [2].

2.2 Row Balancing

In practice, if the rows of the large matrix $C = [C_1, ...C_N]$ are not balanced in their norms, especially when n and N are large, the value found for a can be inaccurate (see for example (7)). To alleviate this, after every few iterations, we use updates $C_i \leftarrow DC_iD$ and $B \leftarrow DB$ where D is a diagonal matrix that

[2] This unit triangular matrix is also known as the Gauss transformation [5]

approximately balances the rows of C. We choose $D(k,k) = \frac{1}{\sqrt{\|C(k,:)\|_F}}$ where $C(k,:)$ is the k^{th} row of C. With this modification, the algorithms perform desirably. As mentioned we could keep track of the values of a cost function (either J_1 or J_2) as a stopping criterion. Since J_1 is not scale invariant and it can change dramatically as a result of row balancing, J_2 is more preferable in this case.

3 Minimization of J_2 for Joint Diagonalization

Now, we introduce LU and QR based algorithms using Jacobi matrices for minimization of J_2. The inverse of a Jacobi matrix is a linear function of its elements. For example, the inverse of $L_{ij}(a) \in \mathcal{L}$ is $L_{ij}(-a)$. This fact can mitigate the effect of the presence of B^{-1} in J_2. We, again, replace the high dimensional minimization problem with a sequence of one dimensional problems involving parameters of Jacobi matrices in LU or QR factorizations. The difference is that here $J_2(L_{ij}(a))$ is a quadric function of a and in order to minimize it we need to employ either an iterative scheme or the known formulae to find the roots of the cubic polynomial $\frac{\partial J_2(L_{ij}(a))}{\partial a}$. Proposition 3 gives $J_2(L_{lk}(a))$ and $J_2(U_{lk}(a))$ in terms of the elements of C_i's:

Proposition 3. *If $k < l$ then: $J_2(L_{lk}(a)) = a_4a^4 + a_3a^3 + a_2a^2 + a_1a + a_0$ and if $k > l$ then $J_2(U_{lk}(a)) = a_4a^4 + a_3a^3 + a_2a^2 + a_1a + a_0$, where:*

$$a_4 = 4\sum_{i=1}^{N} C_i(k,k)^2, \quad a_3 = 8\sum_{i=1}^{N} C_i(k,k)C_i(k,l), \quad a_2 = 2\sum_{i=1}^{N} C_i(k,k)^2 + 2C_i(k,l)^2$$

and:

$$a_1 = 4\sum_{i=1}^{N} C_i(k,l)C_i(k,k), \quad a_0 = 2\sum_{i=1}^{N} C_i(k,l)^2 \tag{8}$$

As mentioned the corresponding minimization is a straight forward task. Similar to QRJ1D and LUJ1D we can have QRJ2D and LUJ2D algorithms by replacing steps referring to Proposition 1 with steps referring to Proposition 3. As it can be seen from the above formula the value of a in minimization of $J_2(L_{lk}(a))$ depends only on the elements of the matrices $\{C_i\}_{i=1}^N$ at positions (k,k) and (k,l). Note that a for minimization of $J_1(L_{lk}(a))$ depends on the elements of $\{C_i\}$ at other positions too. As a result, assuming the computation cost in minimization of $J_2(L_{lk}(a))$ is mainly due to calculating the coefficients, we can see that the complexity of calculating a is of the order $\mathcal{O}(N)$, whereas for $J_1(L_{lk}(a))$ it is of the order $\mathcal{O}(Nn)$. However, the complexity of one iteration (including the costly update of the C_i's) for all the methods is of the order $\mathcal{O}(Nn^3)$. We mention that here also row balancing proves to be useful.

4 Numerical Experiments

We examine the performance of the developed methods by joint diagonalization of a set of matrices that are generated as:

$$C_i = A\Lambda_i A^T + tN_i, \quad \Lambda_i = \text{diag}(\text{randperm}(n))$$

where $\text{diag}(x)$ for a vector x denotes a diagonal matrix whose diagonal is x, $\text{randperm}(n)$ denotes a random permutation of the set $\{1, 2, ..., n\}$, N_i is the symmetric part of a matrix whose elements are i.i.d standard normal random variables and t measures the noise contribution. We try $n = 10$, $N = 100$ with values for $t = 0$ and $t = 0.1$. A is randomly generated. We apply QRJ1D, LUJ1D, QRJ2D and LUJ2D methods [3] with row balancing to find B. The row balancing is performed once per each three iterations. The index:

$$\text{Index}(P) = \sum_{i=1}^{n}(\sum_{j=1}^{n}\frac{|p_{ij}|}{\max_k |p_{ik}|} - 1) + \sum_{j=1}^{n}(\sum_{i=1}^{n}\frac{|p_{ij}|}{\max_k |p_{kj}|} - 1) \qquad (9)$$

which measures how far $P = BA$ is from being permuted diagonal is used to measure the performance. Plots (1.a) and (1.b) show the result. Note that for

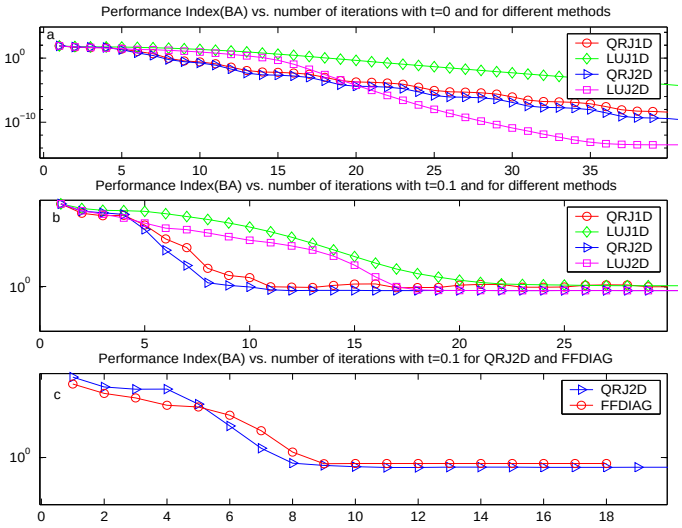

Fig. 1. (a), (b) The performance index Index(BA) for different methods with two noise levels $t = 0$ and $t = 0.1$, respectively. (c) Performance index vs. number of iterations for QRJ2D and FFDIAG with noise level $t = 0.1$

$t = 0$ the index values are very small. Of course, $t = 0.1$ is a more realistic case

[3] Matlab code is available at http://www.isr.umd.edu/Labs/ISL/ICA2006/

for which the convergence is faster. For both $t = 0$ and $t = 0.1$ the QRJ2D and LUJ2D outperform the J_1 based methods. Yet, since in simulations this has not been consistently observed we refrain from any comparison of the methods. In another experiment we compare the QRJ2D method and the FFDIAG [8] algorithm for which the available MATLAB code has been used. With $t = 0.1$ we repeat the previous example and apply both the algorithms. Plot (1.c) shows the index for the two methods. QRJ2D outperform FFDIAG little bit, both in terms of speed and performance. Again, this situation may vary in different experiments. However, we can confirm comparable performance for FFDIAG and the developed methods.

5 Conclusion

We presented simple NOJD algorithms based on the QR and LU factorizations. Using Jacobi matrices we replaced high dimensional minimization problems with a sequence of simple one-dimensional problems. Also a new scale invariant cost function has been introduced and used for developing NOJD algorithms. A comparison with one efficient existing method shows the competence of the developed methods. The idea of resorting to a matrix factorization and solving a sequence of minimization sub-problems over one-parameter subgroups can be useful in other minimization problems over matrix groups.

6 Acknowledgments

This research was supported in part by Army Research Office under ODDR&E MURI01 Program Grant No. DAAD19-01-1-0465 to the Center for Communicating Networked Control Systems (through Boston University). The author is grateful to Dr. P.S. Krishnaprasad for his support as well as his comments on this paper. The author would also like to thank Dr. U. Helmke for his helpful hints and guidance during his September 2004 visit in College Park. The author is also greatly indebted to the anonymous reviewers for their useful comments on this work.

References

1. B. Afsari, P.S .Krishnaprasad: A Novel Non-orthogonal Joint Diagonalization Cost Function for ICA, ISR technical report, 2005(Available at:`http://techreports.isr.umd.edu/reports/2005/TR_2005-106.pdf`)
2. B. Afsari: Gradient Flow Based Matrix Joint Diagonalization for Independent Componenet Analysis, MS Thesis, ECE Department, University of Maryland, College Park, May 2004.(Available at: `http://techreports.isr.umd.edu/reports/2004/MS_2004-4.pdf`)
3. B. Afsari, P.S. Krishnaprasad: Some Gradient Based Joint Diagonalization Methods for ICA, in C. G.Puntonet and A. Prieto(ed's), Proceedings of ICA 2004, Springer LNCS series, Vol. 3195, pp 437-444, 2004

4. J.F. Cardoso and A. Soulumiac:Blind Beamforming For Non-Gauusian Signals, IEE-Proceedings, Vol.140, No 6, Dec 1993
5. G. H. Golub, C. F. Van Loan: Matrix Computations, third eddition, Johns Hopkins University Press, 1996
6. D.T. Pham and J.F. Cardoso: Blind separation of instantaneous mixtures of non stationary sources, IEEE Trans. Signal Processing, pp 1837-1848, vol 49, no 9, 2001
7. A.Yeredor: Non-Orthogonal Joint Diagonalization in the Least-Squares Sense With Application in Blind Source Separation, IEEE Transactions on Signal Processing, Vol 50, No.7.July 2002.
8. A. Ziehe, M. Kawanabe, S. Harmeling, and K.-R. Mller: A Fast Algorithm for Joint Diagonalization with Non-orthogonal Transformations and its Application to Blind Source Separation. Journal of Machine Learning Research; 5(Jul):801–818, 2004