Journal of Machine Learning Research 5 (2004) 777–800 Submitted 12/03; Revised 5/04; Published 7/04

A Fast Algorithm for Joint Diagonalization with Non-orthogonal Transformations and its Application to Blind Source Separation

Andreas Ziehe ZIEHE@FIRST.FHG.DE
Pavel Laskov LASKOV@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany

Guido Nolte NOLTEG@NINDS.NIH.GOV
National Institutes of Health
10 Center Drive MSC 1428
Bethesda, MD 20892, USA

Klaus-Robert Müller KLAUS@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany
and
University of Potsdam, Department of Computer Science
August-Bebel-Strasse 89
14482 Potsdam, Germany

Editor: Michael Jordan

Abstract

A new efficient algorithm is presented for joint diagonalization of several matrices. The algorithm is based on the Frobenius-norm formulation of the joint diagonalization problem, and addresses diagonalization with a general, non-orthogonal transformation. The iterative scheme of the algorithm is based on a multiplicative update which ensures the invertibility of the diagonalizer. The algorithm's efficiency stems from the special approximation of the cost function resulting in a sparse, block-diagonal Hessian to be used in the computation of the quasi-Newton update step. Extensive numerical simulations illustrate the performance of the algorithm and provide a comparison to other leading diagonalization methods. The results of such comparison demonstrate that the proposed algorithm is a viable alternative to existing state-of-the-art joint diagonalization algorithms. The practical use of our algorithm is shown for blind source separation problems.

Keywords: joint diagonalization, common principle component analysis, independent component analysis, blind source separation, nonlinear least squares, Newton method, Levenberg-Marquardt algorithm

1. Introduction

Joint diagonalization of square matrices is an important problem of numeric computation. Many applications make use of joint diagonalization techniques as their main algorithmic tool, for example independent component analysis (ICA) and blind source separation (BSS) (Comon, 1994; Molgedey and Schuster, 1994; Belouchrani et al., 1997; Wu and Principe, 1999; Cardoso, 1999; Ziehe and Müller, 1998; Ziehe et al., 2003; Pham and Cardoso, 2000; Ziehe et al., 2000a; Yeredor, 2002; Haykin, 2000; Hyvärinen et al., 2001), common principal component analysis (CPC) (Flury, 1988; Airoldi and Flury, 1988; Fengler et al., 2001), various signal processing applications (van der Veen et al., 1992, 1998) and, more recently, kernel-based nonlinear BSS (Harmeling et al., 2003).

This paper pursues two goals. First, we propose a new efficient algorithm for joint approximate matrix diagonalization. Our algorithm is based on the second-order approximation of a cost function for the simultaneous diagonalization problem. Second, we demonstrate an application of our algorithm to BSS, which allows to perform BSS without pre-whitening the data.

Let us begin by defining the notion of joint diagonalization. It is well known that exact joint diagonalization is in general only possible for two matrices and amounts to the generalized eigenvalue problem. Extensive literature exists on this topic (e.g. Noble and Daniel, 1977; Golub and van Loan, 1989; Bunse-Gerstner et al., 1993; Van der Vorst and Golub, 1997, and references therein). When more than two matrices are to be diagonalized, exact diagonalization may also be possible if the matrices possess a certain common structure. Otherwise one can only speak of approximate joint diagonalization. Our paper focuses on the investigation of algorithms for exact—whenever this is possible—or otherwise approximate diagonalization of more than two matrices. In the remainder of the paper we will refer to such problems as "joint diagonalization" problems.

A number of algorithms for joint diagonalization have been previously proposed in the literature (Flury and Gautschi, 1986; Cardoso and Souloumiac, 1993, 1996; Hori, 1999; Pham, 2001; van der Veen, 2001; Yeredor, 2002; Joho and Rahbar, 2002). To understand the challenges of the joint diagonalization problem, as well as the need for further improvement of currently known algorithms and possible directions of such improvement some insight into the main issues of joint diagonalization is now provided.

Let us consider a set $\{C^1, \ldots, C^K\}$ of real-valued symmetric matrices of size $N \times N$.[1] The goal of a joint diagonalization algorithm is to find a transformation V that in some sense "diagonalizes" all the given matrices. The notion of diagonality and the corresponding formal statement of the joint diagonalization problem can be defined in at least three different ways:

1. *Frobenius norm formulation*. This formulation is used in Cardoso and Souloumiac (1993, 1996); Joho and Rahbar (2002) and, in a generalized form, in Hori (1999). Let

$$F^k = VC^kV^T \tag{1}$$

denote the result of applying transformation V to matrix C^k. Joint diagonalization is defined as the following optimization problem:

$$\min_{V \in \mathbb{R}^{N\times N}} \sum_{k=1}^{K} \mathfrak{M}_D(F^k), \tag{2}$$

where the diagonality measure $\mathfrak{M}_D$ is the Frobenius norm of the off-diagonal elements in F^k:

$$\mathfrak{M}_D(F^k) = \mathrm{off}(F^k) = \sum_{i\neq j} (F^k_{ij})^2. \tag{3}$$

1. The formulations and the proposed algorithm will be presented for symmetric matrices. Extensions to the unsymmetric or complex-valued case can be obtained in a similar manner.

A more careful look at the cost function in Equation (2) reveals a serious problem with the Frobenius norm formulation: this cost function is obviously minimized by the trivial solution $V = 0$. The problem can be avoided by additionally requiring orthogonality of V. In fact, this assumption is very natural if the joint diagonalization problem is seen as an extension of the eigenvalue problem to several matrices. However, restricting V to the group of orthogonal matrices may limit the applicability and unduly degrade the performance of the method.[2]

2. *Positive definite formulation*. Another reasonable assumption on the initial problem is the positive-definiteness of the matrices C^k. This assumption is motivated by the fact that in many applications matrices C^k are covariance matrices of some random variables. In this case, as proposed in Matsuoka et al. (1995); Pham (2001) the criterion[3]

$$\mathfrak{M}_D(F^k) = \log\det(\mathrm{ddiag}(F^k)) - \log\det(F^k) \tag{4}$$

can be used in the cost function (2) instead of the criterion (3). The additional advantage of this criterion is that it allows for super-efficient estimation (Pham and Cardoso, 2001). However in certain applications, such as blind source separation based on time-delayed decorrelation (Belouchrani et al., 1997; Ziehe and Müller, 1998), correlation matrices are no longer guaranteed to be positive-definite, and diagonalization based on this criterion may fail.

3. *Subspace fitting formulation*. The fact that exact joint diagonalization may not be possible can be explicitly accounted for in the problem formulation. This is to say that, instead of applying the transformation directly to the matrices C^k, another set of diagonal matrices Λ^k is sought for, along with the transformation so as to best approximate the target matrices. The optimization problem resulting from this approach

$$\min_{A,\Lambda^1,\ldots,\Lambda^K} \sum_{k=1}^{K} ||C^k - A\Lambda^k A^T||_F^2 \tag{5}$$

constitutes an instance of a subspace fitting problem (van der Veen, 2001; Yeredor, 2002).

Compared to the previous approaches, the algorithms based on subspace fitting have two advantages: they do not require orthogonality, positive-definiteness or any other normalizing assumptions on the matrices A and C^k, and they are able to handle non-square mixture matrices. These advantages, however, come at a high computational cost: the algorithm of van der Veen (2001) has quadratic convergence in the vicinity of the minimum but its running time per iteration is $O(KN^6)$, whereas the AC-DC algorithm of Yeredor (2002) converges linearly with a running time per iteration of order $O(KN^3)$.

As a short resume of the above mentioned approaches we notice the following. The algorithms using the Frobenius norm formulation are efficient but rely on the orthogonality assumption to prevent convergence to the trivial solution. The algorithms using the positive-definiteness assumption are also quite efficient but they may fail if this assumption is not satisfied. Subspace fitting algorithms, which do not require such strong prior assumptions, are computationally much more demanding. A

2. In principle, a pre-sphering step could be applied to alleviate this problem, nevertheless a performance degradation is to be expected in this case, especially in the context of blind source separation (Cardoso, 1994a; Yeredor, 2002).
3. Here the operator $\mathrm{ddiag}(F)$ returns a diagonal matrix containing only the diagonal entries of F.

natural question arises: could a single algorithm combine the positive and avoid the negative features of the previous joint diagonalization algorithms? In this contribution we present an algorithm using the Frobenius norm formulation that strives towards this goal. In particular, the algorithm, to be called FFDIAG (Fast Frobenius Diagonalization), possesses the following features:

- computational efficiency: quadratic convergence (in the neighborhood of the solution) and $O(KN^2)$ running time per iteration,
- guaranteed avoidance of the trivial solution,
- no orthogonality and no positive-definiteness assumptions; nonetheless, orthogonality can be used to constrain the solution, which further reduces the computational complexity by a factor of two.

On top of that, the algorithm is simple and easy to implement.

The remainder of the paper is organized as follows. In Section 2 the main idea of the FFDIAG algorithm is proposed. The computational details regarding the algorithm's update rule are derived in Section 3. Section 4, in a slight digression from the main topic of the article, presents a connection of our algorithm to the classical Levenberg-Marquardt algorithm, and points out the main differences between the two. The application of the FFDIAG algorithm to blind source separation is developed in Section 5. Extensive numerical simulations are presented in Section 6. Finally, Section 7 is devoted to the discussion and conclusions.

2. General Structure of the Algorithm

The FFDIAG algorithm is an iterative scheme to approximate the solution of the following optimization problem:

$$\min_{V \in \mathbb{R}^{N \times N}} \sum_{k=1}^{K} \sum_{i \neq j} ((VC^k V^T)_{ij})^2. \tag{6}$$

The basic idea is to use the invertibility of the matrix V as a constraint preventing convergence of the minimizer of the cost function in Equation (6) to the zero solution. Invertibility is tacitly assumed in many applications of diagonalization algorithms, e.g. in blind source separation, therefore making use of such constraint is very natural and does not limit the generality from the practical point of view.

Invertibility can be enforced by carrying out the update of V in multiplicative form as

$$V_{(n+1)} \leftarrow (I + W_{(n)}) V_{(n)}, \tag{7}$$

where I denotes the identity matrix, the update matrix $W_{(n)}$ is constrained to have zeros on the main diagonal, and n is the iteration number. Such update is rarely used in classical unconstrained optimization algorithms; however, it is common for many successful BSS algorithms, such as relative-gradient (Laheld and Cardoso, 1996; Amari et al., 2000), relative Newton (Akuzawa and Murata, 2001; Zibulevsky, 2003), as well as for joint diagonalization (Pham, 2001). The off-diagonal component $W_{(n)}$ of the update multiplier is to be determined so as to minimize the cost function (6). In order to maintain invertibility of V it clearly suffices to enforce invertibility of $I + W_{(n)}$. The latter can be carried out using the following well-known results of matrix analysis (Horn and Johnson, 1985).

Definition 1 *An $n \times n$ matrix A is said to be strictly diagonally dominant if*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad \text{for all } i = 1, \ldots, n.$$

Theorem 2 (Levi-Desplanques) *If an $n \times n$ matrix A is strictly diagonally-dominant, then it is invertible.*

The Levi-Desplanques theorem can be used to control invertibility of $I + W_{(n)}$ in a straightforward way. Observe that the diagonal entries in $I + W_{(n)}$ are all equal to 1; therefore, it suffices to ensure that

$$\max_i \sum_{j \neq i} |W_{ij}| = ||W_{(n)}||_\infty < 1.$$

This can be done by dividing $W_{(n)}$ by its infinity norm whenever the latter exceeds some fixed $\theta < 1$. An even stricter condition can be imposed by using a Frobenius norm in the same way:

$$W_{(n)} \leftarrow \frac{\theta}{||W_{(n)}||_F} W_{(n)}. \tag{8}$$

To determine the optimal updates $W_{(n)}$ at each iteration, first-order optimality constraints for the objective (6) are used. A special approximation of the objective function will enable us to efficiently compute $W_{(n)}$. For this reason, we consider the expression for updating the matrices to be diagonalized

$$C^k_{(n+1)} \leftarrow F^k = (I + W_{(n)})\, C^k_{(n)}\, (I + W_{(n)})^T. \tag{9}$$

Let $D^k_{(n)}$ and $E^k_{(n)}$ denote the diagonal and off-diagonal parts of $C^k_{(n)}$, respectively. In order to simplify the optimization problem we assume that the norms of $W_{(n)}$ and $E^k_{(n)}$ are small, i.e. quadratic terms in the expression for the new set of matrices can be ignored

$$\begin{aligned} C^k_{(n+1)} &= (I + W_{(n)})(D^k_{(n)} + E^k_{(n)})(I + W_{(n)})^T \\ &\approx D^k_{(n)} + W_{(n)} D^k_{(n)} + D^k_{(n)} W^T_{(n)} + E^k_{(n)}. \end{aligned} \tag{10}$$

With these simplifications, and ignoring already diagonal terms D^k, the diagonality measure (3) can be computed using expressions linear in W.[4]

$$F^k \approx \tilde{F}^k = W D^k + D^k W^T + E^k. \tag{11}$$

The linearity of terms (11) allows us to explicitly compute the optimal update matrix $W_{(n)}$ minimizing the approximated diagonality criterion

$$\min_W \sum_{k=1}^{K} \sum_{i \neq j} ((W D^k + D^k W^T + E^k)_{ij})^2. \tag{12}$$

Details of the efficient solution of problem (12) are presented in Section 3.

The simplifying assumptions used in (10) require some further discussion. The motivation behind them is that in the neighborhood of the optimal solution, the optimal steps W that take us to

4. The iteration indices will be dropped in the following if all quantities refer to the same iteration.

the optimum are small and the matrices C^k are almost diagonal. Hence, in the neighborhood of the optimal solution the algorithm is expected to behave similarly to Newton's algorithm and converge quadratically. The assumption of small E^k is potentially problematic, especially in the case where exact diagonalization is impossible. A similar derivation can be carried out with E^k fully accounted for, which leads to additional WE^k and E^kW^T terms in the expression (12). However, the resulting algorithm, will not give rise to a computationally efficient solution of problem (12). As for the assumption of small W, it is crucial for the convergence of the algorithm and needs to be carefully controlled. The latter is done by the normalization (8).

Pseudo-code describing the FFDIAG method is outlined in Algorithm 1.[5]

Algorithm 1 FFDIAG

INPUT: C^k { Matrices to be diagonalized}
$W_{(1)} \leftarrow 0$, $V_{(1)} \leftarrow I$, $n \leftarrow 1$ { $V_{(1)}$ could also be initialized by a more clever guess.}
$C^k_{(1)} \leftarrow V_{(1)}\, C^k\, V^T_{(1)}$
repeat
  compute $W_{(n)}$ from $C^k_{(n)}$ according to Equation (17) or (18)
  if $||W_{(n)}||_F > \theta$ **then**
    $W_{(n)} \leftarrow \frac{\theta}{||W_{(n)}||_F} W_{(n)}$
  end if
  $V_{(n+1)} \leftarrow (I + W_{(n)})\, V_{(n)}$
  $C^k_{(n+1)} \leftarrow (I + W_{(n)})\, C^k_{(n)}\, (I + W_{(n)})^T$
  $n \leftarrow n + 1$
until convergence
OUTPUT: V, C^k

Some remarks on convergence properties of the proposed algorithmic scheme are due at this point. In general, Newton-like algorithms are known to converge only in the neighborhood of the optimal solution; however, when they converge, the rate of convergence is quadratic (e.g. Kantorovich, 1949). Since the essential components of our algorithm—the second-order approximation of the objective function and the computation of optimal steps by solving the linear system arising from first-order optimality conditions—are inherited from Newton's method, the same convergence behavior can be expected. In practice, however, the known theoretical estimates of convergence regions of Newton's method, such as the ones provided, e.g., in Theorems 1 and 2 in Kantorovich (1949), are of little utility since they provide no guidance how to reach the convergence region from an arbitrary starting point.

3. Computation of the Update Matrix

The key to computational efficiency of the FFDIAG algorithm lies in exploiting the sparseness introduced by the approximation (11). The special structure of the problem can be best seen in the matrix-vector notation presented next.

5. MATLAB code for FFDIAG can be obtained at `http://www.first.fhg.de/~ziehe/research/FFDiag`.

The $N(N-1)$ off-diagonal entries of the update matrix W are arranged as

$$w = (W_{12}, W_{21}, \ldots, W_{ij}, W_{ji}, \ldots)^T. \tag{13}$$

Notice that this is *not* the usual vectorization operation $\operatorname{vec} W$, as the order of elements in w reflects the pairwise relationship of the elements in W. In a similar way the $KN(N-1)$ off-diagonal entries of the matrices E^k are arranged as

$$e = (E^1_{12}, E^1_{21}, \ldots, E^1_{ij}, E^1_{ji}, \ldots, E^k_{ij}, E^k_{ji}, \ldots)^T. \tag{14}$$

Finally, a large but very sparse, $KN(N-1) \times N(N-1)$ matrix J is built, in the form:

$$J = \begin{pmatrix} J_1 \\ \vdots \\ J_K \end{pmatrix} \text{ with } J_k = \begin{pmatrix} \mathcal{D}^k_{12} & & & \\ & \ddots & & \\ & & \mathcal{D}^k_{ij} & \\ & & & \ddots \end{pmatrix},$$

where each J_k is block-diagonal, containing $N(N-1)/2$ matrices of dimension 2×2

$$\mathcal{D}^k_{ij} = \begin{pmatrix} D^k_j & D^k_i \\ D^k_j & D^k_i \end{pmatrix}, \quad i,j = 1,\ldots,N,\ i \neq j,$$

where D^k_i is a short-hand notation for the ii-th entry of a diagonal matrix D^k. Now the approximate cost function can be re-written as the linear least-squares problem

$$\mathcal{L}(w) = \sum_k \sum_{i \neq j} (\tilde{F}^k_{ij})^2 = (Jw + e)^T (Jw + e).$$

The well-known solution of this problem (Press et al., 1992) reads

$$w = -(J^T J)^{-1} J^T e. \tag{16}$$

We can now make use of the sparseness of J and e to enable the direct computation of the elements of w in (16). Writing out the matrix product $J^T J$ yields a block-diagonal matrix

$$J^T J = \begin{pmatrix} \sum_k (\mathcal{D}^k_{12})^T \mathcal{D}^k_{12} & & & \\ & \ddots & & \\ & & \sum_k (\mathcal{D}^k_{ij})^T \mathcal{D}^k_{ij} & \\ & & & \ddots \end{pmatrix}$$

whose blocks are 2×2 matrices. Thus the system (16) actually consists of decoupled equations

$$\begin{pmatrix} W_{ij} \\ W_{ji} \end{pmatrix} = -\begin{pmatrix} z_{jj} & z_{ij} \\ z_{ij} & z_{ii} \end{pmatrix}^{-1} \begin{pmatrix} y_{ij} \\ y_{ji} \end{pmatrix}, \quad i,j = 1,\ldots,N,\ i \neq j,$$

where

$$z_{ij} = \sum_k D^k_i D^k_j$$

$$y_{ij} = \sum_k D^k_j \frac{E^k_{ij} + E^k_{ji}}{2} = \sum_k D^k_j E^k_{ij}.$$

The matrix inverse can be computed in closed form, leading to the following expressions for the update of the entries of W:

$$\begin{aligned} W_{ij} &= \frac{z_{ij}y_{ji} - z_{ii}y_{ij}}{z_{jj}z_{ii} - z_{ij}^2} \\ W_{ji} &= \frac{z_{ij}y_{ij} - z_{jj}y_{ji}}{z_{jj}z_{ii} - z_{ij}^2}. \end{aligned} \tag{17}$$

(Here, only the off-diagonal elements ($i \neq j$) need to be computed and the diagonal terms of W are set to zero.) Thus, instead of performing inversion and multiplication of large matrices, which would have brought us to the same $O(KN^6)$ complexity as in van der Veen (2001), computation of the optimal $W_{(n)}$ leads to a simple formula (17) which has to be evaluated for each of $N(N-1)$ components of $W_{(n)}$. Since the computation of z_{ij} and y_{ij} also involves a loop over K, the overall complexity of the update step is $O(KN^2)$.

An even simpler solution can be obtained if the diagonalization matrix V is assumed to be orthogonal from the very beginning. Orthogonality of V can be preserved to the first order by requiring W to be skew-symmetric, i.e., $W = -W^T$. Hence only one of each pair of its entries needs to be computed. In this case the structure of the problem is already apparent in the scalar notation, and one can easily obtain the partial derivatives of the cost function. Equating the latter to zero yields the following expression for the update of W:

$$W_{ij} = \frac{\sum_k E_{ij}^k (D_i^k - D_j^k)}{\sum_k (D_i^k - D_j^k)^2}, \quad i,j = 1,\ldots,N,\ i \neq j, \tag{18}$$

which agrees with the result of Cardoso (1994b). To ensure orthogonality of V beyond the first order the update (7) should be replaced by the matrix exponential update

$$V_{(n+1)} \leftarrow \exp(W_{(n)})V_{(n)},$$

where $W_{(n)}$ is skew-symmetric (cf. Akuzawa and Murata, 2001).

4. Comparison with the Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (LM) algorithm (Levenberg, 1944; Marquardt, 1963) is one of the powerful and popular algorithms for solving nonlinear least-squares problems. Interestingly, the motivation in the original article of Marquardt (1963) was somewhat similar to ours: he knew that quadratic convergence of Newton's method was attainable only in the neighborhood of the solution, and therefore he looked for an efficient means of steering the algorithm to the area of quadratic convergence. The particular mechanism used in the LM algorithm consists of a controllable trade-off between Newton and gradient steps.

Although the problem of simultaneous diagonalization is essentially a nonlinear (quadratic) least-squares problem, the LM algorithm cannot be directly applied to it. An implicit assumption of the simultaneous diagonalization problem is the invertibility of the diagonalizing matrix, and the classical LM algorithm does not provide for incorporation of additional constraints. In what follows we present a modification which allows one to incorporate the additional structure of our problem into the LM algorithm.

A general problem of the nonlinear regression to be solved by the LM algorithm is usually formulated as

$$\min_{p} \sum_{k} ||y_k - f_p(x_k)||^2.$$

The goal of the optimization is to find the parameters p of the regression function f so as to minimize the squared deviations between $f_p(x_k)$ and y_k for all data points $1,\ldots,k$. The signature of the function f can be arbitrary, with an appropriate norm to be chosen. The cost function (6) can be seen as a nonlinear regression problem over the vector-valued function $f_V(C)$, parameterized by V, of matrix argument C, with zero target values:

$$\min_{V\in\mathbb{R}^{n\times n}} \sum_{k} ||0 - f_V(C^k)||^2.$$

The construction and dimensionality of the function f is explained below, and the zero vector is set to have the appropriate dimension.

To enforce invertibility, the diagonal entries of V are set to 1 and only off-diagonal entries are considered as free parameters. As in Section 3 such a representation of V is constructed by means of symmetric vectorization $\operatorname{vecs} V \stackrel{\text{def}}{=} [V_{21}, V_{12}, V_{31}, V_{13}, \ldots]^T$.[6]

The same vectorization is applied to construct the regression function

$$f_V(C) : \mathbb{R}^{N\times N} \to \mathbb{R}^{N(N-1)\times 1} \stackrel{\text{def}}{=} \operatorname{vecs} VCV^T.$$

As a result of such vectorization, the diagonal entries of VCV^T are discarded, and the Euclidean norm of this vector is equivalent to the “off” function.

The LM algorithm requires computation of the Jacobian matrix of the regression function (w.r.t. parameters $\operatorname{vecs}V$) at all data points:

$$J_{LM} \stackrel{\text{def}}{=} [\mathsf{D}_{\operatorname{vecs}V} f_V(C^1), \ldots, \mathsf{D}_{\operatorname{vecs}V} f_V(C^K)]^T.$$

The Jacobian matrices (of dimension $N(N-1) \times N(N-1)$) at individual data points can be computed as:

$$\mathsf{D}_{\operatorname{vecs}V} f_V(C^k) = S_{NN}(I_{N^2} + K_{NN})(VC^k \otimes I_N)S_{NN}^T,$$

where K_{NN} is the commutation matrix (Lütkepohl, 1996), I is the identity matrix of the appropriate size, and $\otimes$ is the Kronecker matrix product.

Denoting $f = [f^T(C^1), \ldots, f^T(C^K)]^T$, the main step of the LM algorithm consists of solving the following linear system:

$$((J_{LM})^T J_{LM} + \lambda I) \operatorname{vecs} V = -(J_{LM})^T f. \tag{20}$$

The parameter λ controls the trade-off between Newton-like and gradient-based strategies: $\lambda = 0$ results in the pure Newton-direction, whereas with large λ, the steps approach the gradient direction. We use the original heuristic of Marquardt to choose λ: if the value of the cost function provided by the current step $\operatorname{vecs} V$ decreases, λ can be decreased by a factor of 10 while descent is maintained; if the value of the cost function increases, increase λ by a factor of 10 until descent is achieved. This heuristic is very intuitive and easy to implement; however, since it doesn't involve any line search,

6. The symmetric vectorization vecs is related to column vectorization vec by the special permutation matrix S_{NN} such that $\operatorname{vecs} X = S_{NN} \operatorname{vec} X$.

the algorithm may fail to converge. More sophisticated strategies with convergence guarantees of Osborne (1976) and Moré (1978) can also be deployed.

From the theoretical point of view, one can draw the following parallels between the FFDIAG and LM algorithms:

- Both algorithms pursue a Newton direction (cf. equations (16) and (20)) to compute the update matrix V. Whereas the LM algorithms computes the update step directly on V, the update of the FFDIAG is performed in a multiplicative way by computing W to be used in the update rule (7).

- Unlike the LM algorithm using the Hessian of the original cost function, the Newton direction in FFDIAG is computed based on the Hessian of the second-order approximation of the cost function.[7] Taking advantage of the resulting special structure, this computation can be carried out very efficiently in FFDIAG.

- Regularization in the LM algorithm results in a gradual shift from the gradient to the Newton directions (and back when necessary). Regularization in the FFDIAG algorithm is of quite different flavor. Since the computed direction is only approximately Newton, one cannot fully trust it, and therefore the update heuristic (8) limits the impact of inaccurate computation of W. On the other hand, when FFDIAG converges to a close neighborhood of the optimal solution, the heuristic is turned off, and Newton-like convergence is no longer impeded.

It is interesting to compare performance of the LM and FFDIAG algorithms experimentally. We use two criteria: the cost function and the convergence ratio

$$\text{convergence ratio} = \frac{||f_{(n+1)} - f^*||}{||f_{(n)} - f^*||}.$$

The zero value of the convergence ratio indicates super-linear convergence. The evolution of our criteria in two runs of the algorithms are shown in Figure 1. In the cost function plot one can see that convergence of both algorithms is linear for the most part of their operation, with a gradual shift to quadratic convergence as the optimal solution is approached. The same conclusion can be drawn from the convergence ratio plot, in which one can see that this criterion approaches zero in the neighborhood of the optimal solution. Thus one can conclude that, similarly to the LM algorithm, the heuristic (8) steers the algorithm to the area where Newton-like convergence is achieved. Furthermore, we note that due to the use of the special structure, the per-iteration complexity of the FFDIAG algorithm is significantly lower than that of the LM algorithm.

5. Application to Blind Source Separation

First, we recall the definition of the blind source separation (BSS) problem (Jutten and Herault, 1991). We are given the instantaneous linear mixtures $x_i(t)$ of a number of source signals $s_j(t)$, obeying the model

$$x_i(t) = \sum_{j=1}^{m} a_{ij} s_j(t), \ (i = 1, \dots, n, \ j = 1, \dots, m), \tag{21}$$

7. In fact, in both algorithms, the Hessians are approximated by the product of the Jacobian matrices.

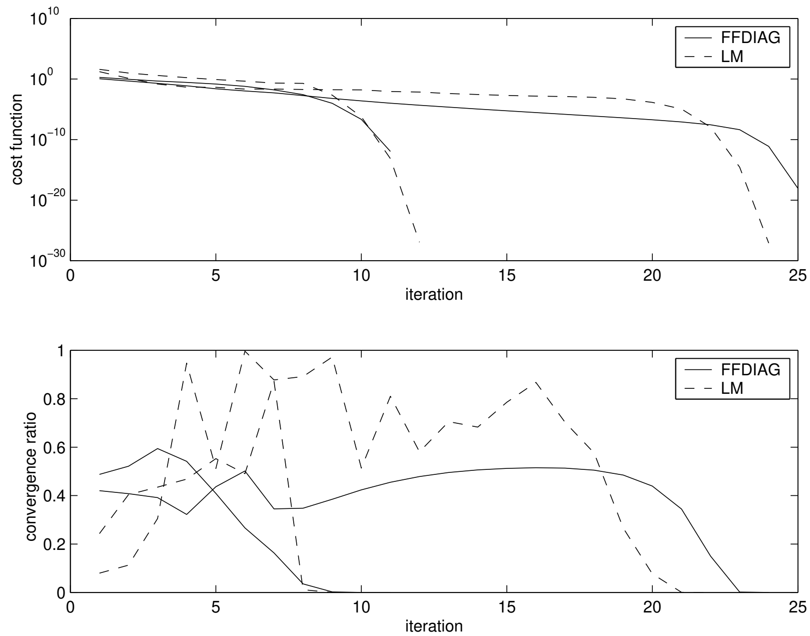

Figure 1: Comparison of the LM and the FFDIAG algorithms. The data matrices are generated as described in Section 6.1 with $K = 10$, $N = 10$. Two illustrative runs are shown.

with A being non-singular and $s_j(t)$ statistically independent. The goal is to estimate both A and $s(t)$ from $x(t)$.

Linear BSS methods have been successfully applied to a variety of problems. An example of such an application is the reduction of artifacts in electroencephalographic (EEG) and magnetoencephalographic (MEG) measurements. Due to the fact that the electromagnetic waves superimpose linearly and virtually instantaneously (because of the relatively small distance from sources to sensors) the model (21) is valid (Makeig et al., 1996; Vigário et al., 1998; Wübbeler et al., 2000; Ziehe et al., 2000a). Note, however, that in other applications, such as the so called cocktail-party problem in auditory perception (von der Malsburg and Schneider, 1986), the model from Equation (21) may be too simplistic, since time-delays in the signal propagation are no longer negligible. Extended models to deal with such convolutive mixtures have been considered (e.g. Parra and Spence, 1998; Lee et al., 1998; Murata et al., 2001). We will in the following only discuss how to solve the linear, instantaneous BSS problem stated in Equation (21). The usual approach is to define an appropriate cost function that can subsequently be optimized. Here our goal is to use the general joint diagonalization cost function (6) and to construct certain matrices in such a way that their approximate joint diagonalizer is an estimate for the demixing matrix V (up to an arbitrary permutation and scaling of its rows).

Let us consider for example the spatial covariance matrix of the mixed signals $x(t)$,

$$C_{(x)} \stackrel{\text{def}}{=} E\{x(t)x(t)^T\} = E\{(As(t))(As(t))^T\} = A\,E\{s(t)s(t)^T\}A^T,$$

where the expectation is taken over t. We see that theoretically $C_{(x)} = AC_{(s)}A^T$ is similar to a diagonal matrix, because the cross-correlation terms that form the off-diagonal part of $C_{(s)}$ are zero for independent signals. There are many more possibilities to define matrices that have the same property as the covariance matrix, namely that they are diagonal for the source signals and ‘similar to diagonal’ for the observed mixtures and, most important, that the inverse $V = A^{-1}$ of the mixing matrix A diagonalizes them all simultaneously. Examples are time-lagged covariances (Molgedey and Schuster, 1994; Belouchrani et al., 1997; Ziehe and Müller, 1998), covariance matrices of different segments of the data (Pham and Cardoso, 2000; Choi et al., 2001), matrices obtained from spatial time-frequency distributions (Pham, 2001), slices of the cumulant tensor (Cardoso, 1999) or Hessians of the characteristic function (Yeredor, 2000). Generally, for stationary and temporally correlated signals we can define a set of matrices C^k with entries

$$(C_{(x)})^k_{ij} = \frac{1}{2}\sum_{t=1}^{T} x_i(t)(\Phi^k \star x_j)(t) + x_j(t)(\Phi^k \star x_i)(t), \tag{23}$$

where $\star$ denotes convolution and $\Phi^k(t)$, $k = 1,\ldots,K$ are linear filters (Ziehe et al., 2000b).

We note that in the popular special case where the Φ^k are simple time-shift operators $\Phi^\tau(t) = \delta_{t\tau}$ (cf. Tong et al., 1991; Molgedey and Schuster, 1994; Belouchrani et al., 1997) the matrices defined by Equation (23) may become indefinite for certain choices of τ. Furthermore, in practice, the above target matrices have always to be estimated from the available data which inevitably gives rise to estimation errors. Hence the best we can do is to find the matrix which diagonalizes the estimated target set “as good as possible”. Since we are able to perform the approximate joint diagonalization with a non-orthogonal transformation, we avoid the problematic pre-whitening step and obtain an estimate of the mixing matrix $A = V^{-1}$ by applying our FFDIAG algorithm directly to the empirical matrices (23). Algorithm 2 summarizes the typical steps in an application to BSS.

Algorithm 2 The FFSEP algorithm

INPUT: $x(t)$, Φ^k
$C^k = \ldots$ {Estimate a number of matrices C^k according to Equation (23)}
$V = \text{FFDIAG}(C^k)$ {Apply joint diagonalization method}
$u(t) = Vx(t)$ {unmix signals}
OUTPUT: u(t), V

6. Numerical Simulations

The experiments in this Section are intended to compare the FFDIAG algorithm with state-of-the-art algorithms for simultaneous diagonalization and to illustrate the performance of our algorithm in BSS applications. As we mentioned in the introduction, there exist at least three alternative formulations of the simultaneous diagonalization problem. The most successful algorithms representing the respective approaches were chosen for comparison.

We present the results of five progressively more complex experiments. First, we perform a “sanity check” experiment on a relatively easy set of perfectly diagonalizable matrices. This experiment is intended to emphasize that for small-size diagonalizable matrices the algorithm’s performance matches the expected quadratic convergence. In the second experiment we compare the FFDIAG algorithm with the extended Jacobi method as used in the JADE algorithm of Cardoso and Souloumiac (1993) (orthogonal Frobenius norm formulation), Pham’s algorithm for positive-definite matrices (Pham, 2001) and Yeredor’s AC-DC algorithm (Yeredor, 2002) (non-orthogonal, subspace fitting formulation). In the third experiment we investigate the scaling behavior of our algorithm as compared to AC-DC. Furthermore, the performance of the FFDIAG algorithm is tested and compared with the AC-DC algorithm on noisy, non-diagonalizable matrices. Finally, the application of our algorithm to BSS is illustrated.

6.1 "Sanity Check" Experiment

The test data in this experiment is generated as follows. We use $K = 15$ diagonal matrices D^k of size 5×5 where the elements on the diagonal are drawn from a uniform distribution in the range $[-1, \ldots, 1]$ (cf. Joho and Rahbar, 2002). These matrices are 'mixed' by an orthogonal matrix A according to AD^kA^T to generate the set of target matrices $\{C^k\}$ to be diagonalized.[8] The FFDIAG algorithm is initialized with the identity matrix $V_{(0)} = I$, and the skew-symmetric update rule (18) is used.

The convergence behavior of the algorithm in 10 runs is shown in Figure 2. The diagonalization error is measured by the off$(\cdot)$ function. One can see that the algorithm has converged to the correct solution after less than 10 iterations in all trials. The quadratic convergence is observed from early iterations.

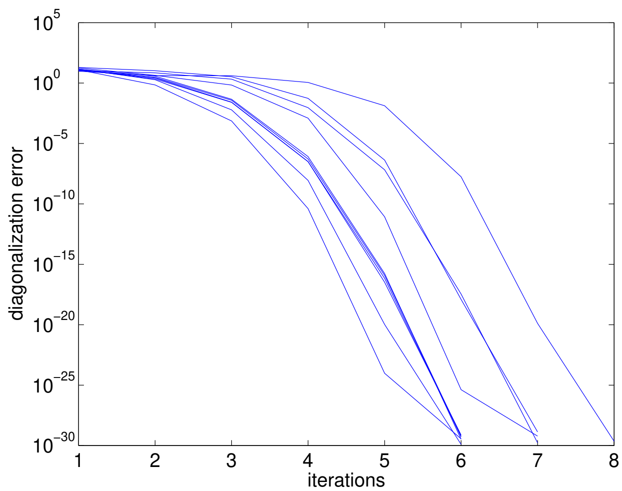

Figure 2: Diagonalization errors of the FFDIAG algorithm for a diagonalizable problem.

6.2 Comparison with the State-of-the-Art Algorithms

Two scenarios are considered for a comparison of the four selected algorithms: FFDIAG, the extended Jacobi method, Pham's algorithm and AC-DC. First, we test these algorithms on diagonalizable matrices under the conditions satisfying the assumptions of all of them. Such conditions are: positive-definiteness of the target matrices C^k and orthogonality of the true transformation A used to generate those matrices. These conditions are met by generating the target matrices $C^k = AD^kA^T$ where D^k are diagonal matrices with positive entries on the main diagonal. The data set consists of 100 random matrices of size 10×10 satisfying the conditions above.

8. The orthogonal matrix was obtained from a singular value decomposition of a random 5×5 matrix, where the entries are drawn from a standard normal distribution.

A comparison of the four algorithms on orthogonal positive-definite matrices is shown in Figure 3. Two runs of the algorithms are presented, for the AC-DC algorithm 5 AC steps were interlaced with 1 DC step at each iteration. Although the algorithms optimize different objective functions, the $\text{off}(\cdot)$ function is still an adequate evaluation criterion provided that the arbitrary scale is properly normalized.

To achieve this, we evaluate $\sum_k \text{off}(\hat{A}^{-1}C^k\hat{A}^{-T})$ where $\hat{A}$ is the normalized estimated mixing matrix. At the true solution the criterion must attain zero. One can see that the convergence of Pham's algorithm, the extended Jacobi method and FFDIAG is quadratic, whereas the AC-DC algorithm converges linearly. The average iteration complexity of the four algorithms is shown in Table 1. It

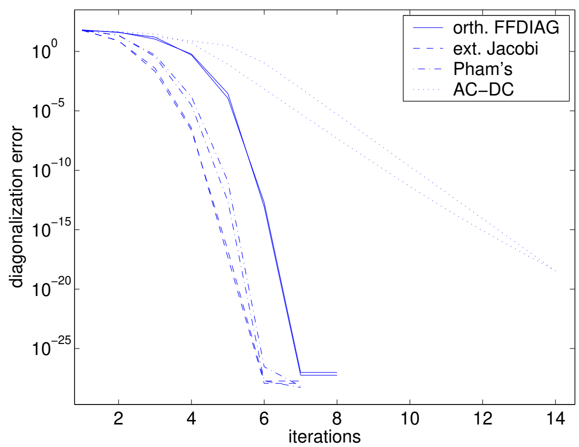

Figure 3: Comparison of the FFDIAG, the extended Jacobi method, Pham's algorithm and AC-DC in the orthogonal, positive-definite case: diagonalization error per iteration measured by the $\text{off}(\cdot)$ criterion.

follows from this table that the FFDIAG algorithm indeed lives up to its name: its running time per iteration is superior to both Pham's algorithm and AC-DC, and is comparable to the extended Jacobi method algorithm.[9]

FFDIAG	ext. Jacobi	Pham's	AC-DC
0.025	0.030	0.168	2.430

Table 1: Comparison of the FFDIAG, ext. Jacobi, Pham's and AC-DC algorithms in the orthogonal, positive-definite case: average running time per iteration in seconds.

In the second scenario, the comparison of the FFDIAG and the AC-DC algorithms is repeated for non-positive-definite matrices obtained from a non-orthogonal mixing matrix. This case cannot be handled by the other two algorithms, therefore they are omitted from the comparison. The convergence plots are shown in Figure 4; average running time per iteration is reported in Table 2.

9. In all experiments, MATLAB implementations of the algorithms were run on a standard PC with a 750MHz clock.

Convergence behavior of the two algorithms is the same as in the orthogonal, positive-definite case; the running time per iteration of FFDIAG increases due to the use of non-skew-symmetric updates.

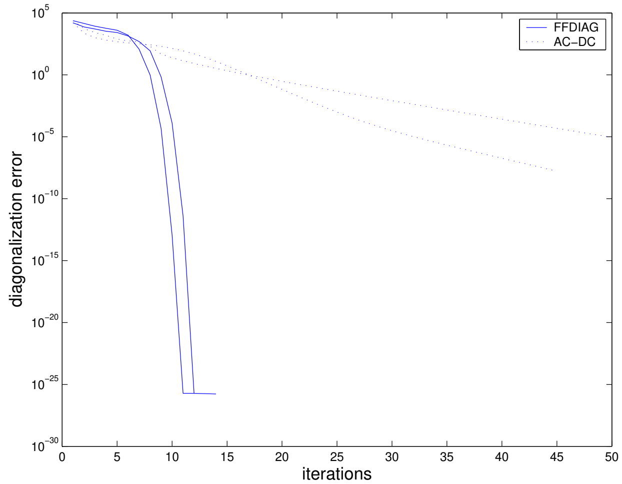

Figure 4: Comparison of the FFDIAG and AC-DC algorithms in the non-orthogonal, non-positive-definite case: diagonalization error per iteration measured by the off($\cdot$) criterion.

FFDIAG	AC-DC
0.034	2.64

Table 2: Comparison of the FFDIAG and AC-DC algorithms in the non-orthogonal, non-positive-definite case: average running time per iteration in seconds.

6.3 Scaling Behavior of FFDIAG

Scalability is essential for application of an algorithm to real-life problems. The most important parameter of the simultaneous diagonalization problem affecting the scalability of an algorithm is the size of the matrices. Figure 5 shows the running time per iteration of the FFDIAG and the AC-DC algorithms for problems with increasing matrix sizes, plotted at logarithmic scale. One can see that both algorithm exhibit running times of $O(N^2)$; however, in absolute terms the FFDIAG algorithm is almost two orders of magnitude faster.[10]

6.4 Non-diagonalizable Matrices

We now investigate the impact of non-diagonalizability of the set of matrices on the performance of the FFDIAG algorithm. Again, two scenarios are considered: the one of the "sanity check" ex-

10. This seemingly controversial result—theoretically expected scaling factor of AC-DC is $O(N^3)$—is due to high constants hidden in the setup phase of AC-DC. The setup phase has $O(N^2)$ complexity, but because of the constants it outweighs the main part of the algorithm in our experiment.

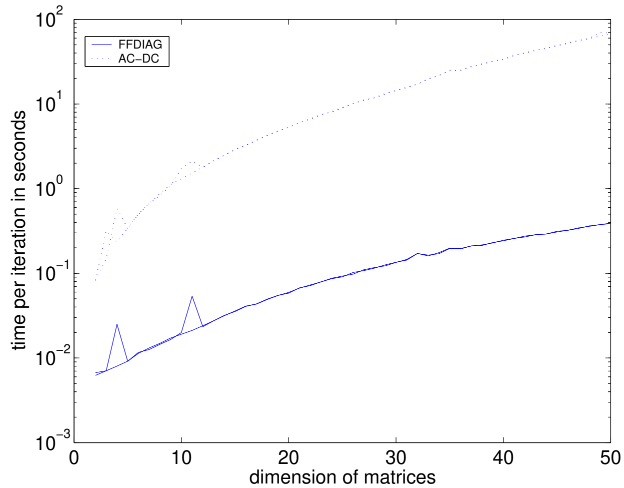

Figure 5: Scaling of the FFDIAG and AC-DC algorithms with respect to the matrix size. Two repetitions of the experiment have been performed.

periment and the comparative analysis against the established algorithms. Non-diagonalizability is modeled by adding a random non-diagonal symmetric "noise" matrix to each of the input matrices:

$$C^k = A\,D^k A^T + \sigma^2 (R^k)(R^k)^T,$$

where the elements of R^k are drawn from a standard normal distribution. The parameter σ allows one to control the impact of the non-diagonalizable component. Another example, with a more realistic noise model, will be presented in subsection 6.5.

Figure 6 shows the convergence plots of FFDIAG for various values of σ. The experimental setup is the same as in Section 6.1, apart from the additive noise. The impact of the latter can be quantified by computing the off$(\cdot)$ function on the noise terms only (averaged over all runs), which is shown by the dotted line in Figure 6. One can see that the algorithm converges quadratically to the level determined by the noise factor.

Similar to the second scenario in Section 6.2, the previously mentioned algorithms are tested on the problem of approximate joint diagonalization with non-orthogonal transforms. (Only the extended Jacobi algorithm had to be excluded from the comparison since it is not designed to work with non-orthogonal diagonalizers.) However, in contrast to Section 6.2, positive-definite target matrices were generated in order to enable a comparison with Pham's method.

Furthermore, we introduce another measure to assess the algorithms' performance in the non-orthogonal, non-diagonalizable case. In synthetic experiments with artifical data the distance from the true solution is a good evaluation criterion. To be meaningful, this distance has to be invariant w.r.t. the irrelevant scaling and permutation ambiguities. For this reason, we choose a performance index that is commonly used in the context of ICA/BSS where the same invariances exist (see e.g. in Amari and Cichocki, 1998; Cardoso, 1999). Following the formulation of Moreau (2001) a suitable performance index is defined on the normalized "global" matrix $G \stackrel{\text{def}}{=} VA$ according to

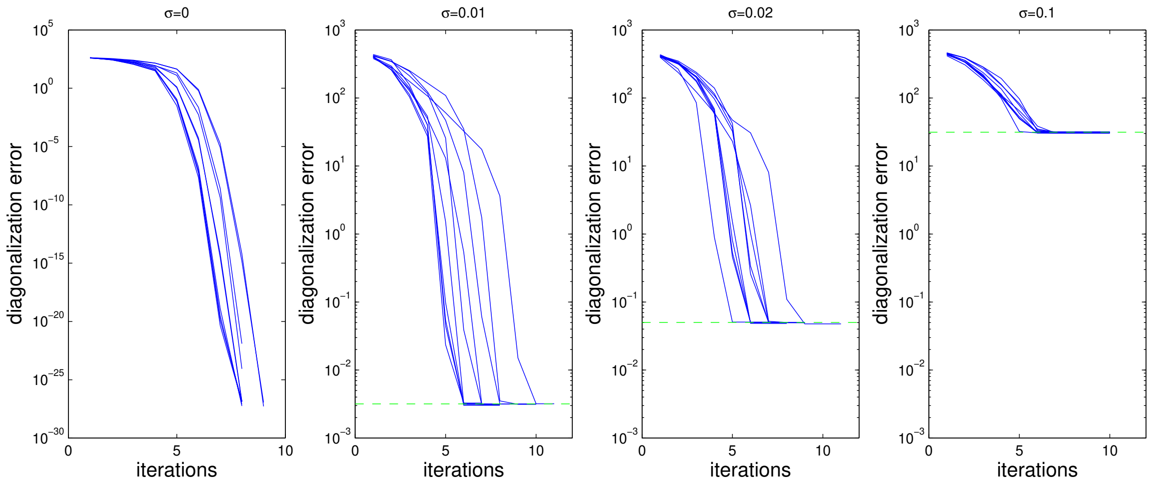

Figure 6: Diagonalization errors of the FFDIAG algorithm on non-diagonalizable matrices.

$$\mathrm{score}(G) = \frac{1}{2}\left[\sum_i\left(\sum_j \frac{|G_{ij}|^2}{\max_l |G_{il}|^2} - 1\right) + \sum_j\left(\sum_i \frac{|G_{ij}|^2}{\max_l |G_{lj}|^2} - 1\right)\right]. \qquad (24)$$

Clearly, this non-negative index attains zero iff G is a product of an invertible diagonal matrix D and of a permutation matrix P, i.e., $G = DP$.

The results of the comparison of the FFDIAG, Pham's and AC-DC algorithms on a non-orthogonal positive-definite problem (5 matrices of dimension 5×5) at various noise levels are shown in Figure 7 for three typical runs. The graphs illustrate some interesting ascpects of the convergence behavior of the algorithms. Both the FFDIAG and Pham's algorithm converge within a small number of iterations to approximately the same error level. The AC-DC algorithm converges linearly, and occasionally convergence can be very slow, as can be seen in each of the plots in Figure 7. However, when AC-DC converges, it exhibits better performance as measured by the score function; the higher the noise level, the stronger the difference.

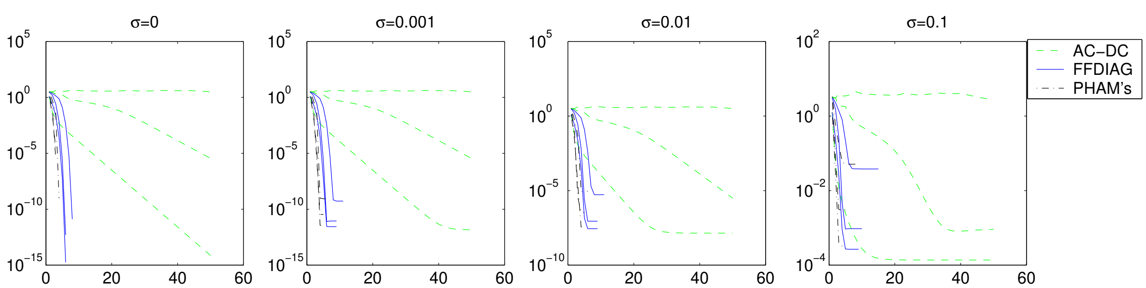

Figure 7: Comparison of the FFDIAG, Pham's and AC-DC algorithms in the non-diagonalizable, non-orthogonal, positive-definite case at various noise levels: performance index as measured by the score function (24).

6.5 Blind Source Separation

Finally, we apply our method to a blind source separation task. The signal matrix S contains seven audio signals containing 10000 points recorded at 8kHz and one Gaussian noise source of the same length. These signals are mixed by a 8×8 Hadamard matrix,

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

This scaled orthogonal matrix produces a complete mixture, in the sense that each observation contains a maximal contribution from each source. We compute 50 symmetrized, time-lagged correlation matrices according to Equation (23) with $\Phi^\tau(t) = \delta_{t\tau}$ and apply the FFDIAG algorithm with $V_{(0)} = I$. Figure 8 shows the evolution of performance measure defined in (24) and of the entries of the (normalized) global system $V_{(n)}A$. One can see that the difference from the true solution, in terms of the score function, approaches zero and that $V_{(n)}A$ converges to a permutation matrix (as shown in the middle and the right panels).

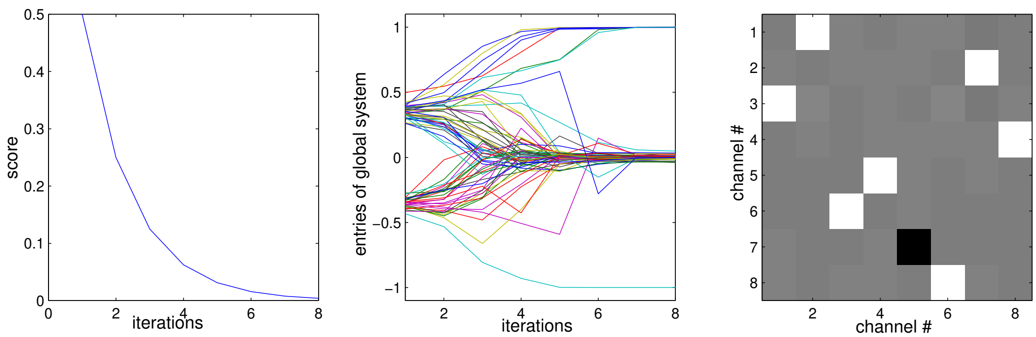

Figure 8: Convergence progress of the FFSEP algorithm on the BSS task. The middle panel shows the evolution of the entries of the normalized global matrix G. The right panel shows those entries for the final (8th) iteration in matrix form and indicates that the cross-talk is minimized since the matrix G resembles a scaled and permutated identity matrix. Here, black, white and gray squares correspond to values -1, 1 and 0, respectively.

In order to study the behavior of the FFDIAG algorithm in a more realistic noisy scenario the following experiment is conducted. The data is generated by mixing three stationary, time-correlated sources with the fixed matrix $A = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$. The sources are generated by feeding an i.i.d. random noise signal into a randomly chosen, auto-regressive (AR) model of order 5 whose coefficients are drawn from a standard normal distribution and are sorted in decreasing order (to ensure stability). The generated signals have a total length of 50000 samples. To separate the sources we estimate 10 symmetrized, time-lagged correlation matrices of the mixed signals according to Equation (23) with $\Phi^\tau(t) = \delta_{t\tau}$ and perform simultaneous diagonalization of these matrices.

Clearly, the quality of the estimation depends on the number T of samples used to estimate these correlation matrices. By varying T we can simulate different noise levels corresponding to the variance of the estimates, which is more realistic than corrupting the target matrices with small i.i.d. additive noise.

The results of the experiment are shown in Figure 9. Performance of the FFDIAG and the AC-DC algorithms, as measured by the score (24), is displayed for four different sample sizes, the smaller samples corresponding to the higher noise level. 100 repetitions are performed for each sample size, and the 25%, 50% and 75% quantiles of the log-score are shown in the plots. Two observations can be made from Figure 9: FFDIAG converges much faster than AC-DC, and when converged, FFDIAG yields a better score (on average), with the difference more pronounced for samples sizes 10000 and 30000 in our experiment.

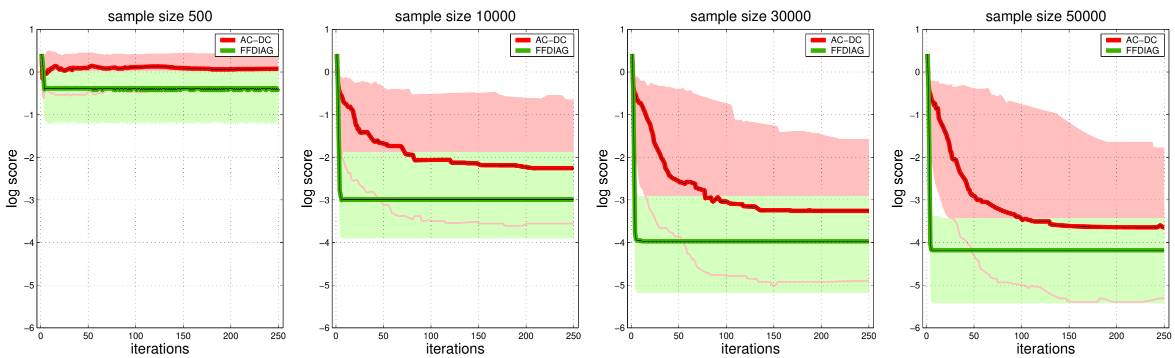

Figure 9: Performance of FFDIAG and AC-DC measured by the log of the score (24) for different sample sizes and 100 trials each. 25% (lower edge of the shaded region), 50% (thick line in the middle) and 75% quantiles (upper edge of the shaded region) are shown.

7. Discussion and Conclusions

We have presented a new algorithm FFDIAG for simultaneous diagonalization of a set of matrices. The algorithm is based on the Frobenius norm formulation of the simultaneous diagonalization problem and provides an efficient means of diagonalization in the absence of additional constraints, such as orthogonality or positive-definiteness. The important feature of our algorithm is the direct enforcement of invertibility of the diagonalizer; in previous work this was usually achieved by an orthogonality constraint which reduces the space of solutions.

The efficiency of the FFDIAG algorithm lies in the special second-order approximation of the cost function, which yields a block-diagonal Hessian and thus allows for highly efficient computation of the Newton update step. Although, theoretically, such approximation can be seen as a weakness of the approach—and raise the question of whether the point of the algorithm's convergence is indeed an optimizer of the full cost function—we have empirically observed that the solution found by the algorithm is of good quality for practical applications.

A series of comparisons of the FFDIAG algorithm with state-of-the-art diagonalization algorithms is presented under a number of conditions that can or cannot be handled by other algorithms. The main conclusions of this comparative evaluation is that our algorithm is competitive with the best algorithms (i.e. Jacobi-based and Pham's algorithm) that impose additional constraints either on the class of solutions or the type of input data. FFDIAG is significantly more efficient—as far as both the scaling factors and the absolute constants are concerned—than the AC-DC algorithm, the only general algorithm applicable under the same conditions as ours. The FFDIAG algorithm can be applied to matrices of dimensions in the hundreds of rows/columns, under no additional assumptions. It also performs reliably on non-diagonalizable data, for which only an approximate solution is possible.

Several interesting research topics can be anticipated. From a theoretical point of view, convergence analysis could yield further insights into the numerical behavior of FFDIAG as well as a better understanding of the general techniques for optimization over nonholonomic manifolds that the algorithm belongs to. Further investigation of the robustness of joint diagonalization algorithms in the presence of various forms of noise is a very interesting practical issue. Numerous applications of the algorithm to real-life problems can be clearly foreseen.

Acknowledgments

The authors thank Benjamin Blankertz, Steven Lemm, Christin Schäfer, Sebastian Mika, Stefan Harmeling, Frank Meinecke, Guido Dornhege, Motoaki Kawanabe, David Tax, Julian Laub, Matthias Krauledat, Marcel Joho, Michael Zibulevsky and Arie Yeredor for sharing their insights and expertise in many fruitful discussions. Furthermore, the in-depth comments and valuable suggestions of the three anonymous reviewers are highly appreciated. This helped us to improve the initial version of the manuscript.

A.Z., P.L. and K.-R.M. acknowledge partial funding in the EU project (IST-1999-14190 – BLISS), by BMBF under contract FKZ 01IBB02A, the SFB 618 and the PASCAL Network of Excellence (EU #506778). G.N. has been supported by a grant from the National Foundation for Functional Brain Imaging.

References

J. P. Airoldi and B. Flury. An application of common principal component analysis to cranial morphometry of microtus californicus and m. ochrogaster (mammalia, rodentia). *Journal of Zoology*, 216:21–36, 1988.

T. Akuzawa and N. Murata. Multiplicative nonholonomic newton-like algorithm. *Chaos, Solitons & Fractals*, 12(4):785–793, 2001.

S.-I. Amari, T.-P. Chen, and A. Cichocki. Nonholonomic orthogonal learning algorithms for blind source separation. *Neural Computation*, 12:1463–1484, 2000.

S.-I. Amari and A. Cichocki. Adaptive blind signal processing – neural network approaches. *Proceedings of the IEEE*, 9:2026–2048, 1998.

A. Belouchrani, K. Abed Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique based on second order statistics. *IEEE Trans. on Signal Processing*, 45(2):434–444, 1997.

A. Bunse-Gerstner, R. Byers, and V. Mehrmann. Numerical methods for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 14(4):927–949, 1993.

J.-F. Cardoso. On the performance of orthogonal source separation algorithms. In *Proc. EUSIPCO*, pages 776–779, 1994a.

J.-F. Cardoso. Perturbation of joint diagonalizers. ref# 94d027. Technical report, Télécom Paris, 1994b.

J.-F. Cardoso. High-order contrasts for independent component analysis. *Neural Computation*, 11 (1):157–192, January 1999.

J.-F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings F*, 140(6):362–370, 1993.

J.-F. Cardoso and A. Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM J. Mat. Anal. Appl.*, 17(1):161–164, January 1996.

S. Choi, A. Cichocki, and A. Belouchrani. Blind separation of second-order nonstationary and temporally colored sources. In *Proc. IEEE Workshop on Statistical Signal Processing (IEEE SSP 2001)*, pages 444–447, Singapore, 2001.

P. Comon. Independent component analysis, a new concept? *Signal Processing, Elsevier*, 36(3): 287–314, 1994.

M. R. Fengler, W. Härdle, and C. Villa. The dynamics of implied volatilities: A common principal components approach. Technical Report Discussion paper 2003-38, SFB 373, Humboldt-Universität zu Berlin, 2001.

B. Flury. *Common Principal Components and Related Multivariate Models*. Wiley, New York, 1988.

B. Flury and W. Gautschi. An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form. *SIAM Journal on Scientific and Statistical Computing*, 7(1):169–184, January 1986.

G. H. Golub and C. F. van Loan. *Matrix Computation*. The Johns Hopkins University Press, London, 1989.

S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15:1089–1124, 2003.

S. Haykin, editor. *Unsupervised adaptive filtering, Volume 1, Blind Source Separation*. John Wiley & Sons, New York, 2000.

G. Hori. Joint diagonalization and matrix differential equations. In *Proc. of NOLTA'99*, pages 675–678. IEICE, 1999.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press., Cambridge, 1985.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, New York, 2001.

M. Joho and K. Rahbar. Joint diagonalization of correlation matrices by using Newton methods with application to blind signal separation. In *Proc. of IEEE Sensor Array and Multichannel Signal Processing Workshop SAM*, pages 403–407, 2002.

C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

L. V. Kantorovich. On Newton's method. In *Trudy Mat. Inst. Steklov*, volume 28, pages 104–144. Akad. Nauk SSSR, 1949. Translation: Selected Articles in Numerical Analysis by C. D. Benster.

B. Laheld and J.-F. Cardoso. Equivariant adaptive source separation. *IEEE Trans. on Signal Processing*, 44(12):3017–3030, 1996.

T.-W. Lee, A. Ziehe, R. Orglmeister, and T. J. Sejnowski. Combining time-delayed decorrelation and ICA: Towards solving the cocktail party problem. In *Proc. ICASSP98*, volume 2, pages 1249–1252, Seattle, 1998.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, pages 164–168, 1944.

H. Lütkepohl. *Handbook of matrices*. John Wiley & Sons, 1996.

S. Makeig, A. J. Bell, T.-P. Jung, and T. J. Sejnowski. Independent component analysis of electroencephalographic data. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems (NIPS'95)*, volume 8, pages 145–151. The MIT Press, 1996.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of SIAM*, 11(2):431–441, jun 1963.

K. Matsuoka, M. Ohya, and M. Kawamoto. A neural net for blind separation of nonstationary signals. *Neural Networks*, 8:411–419, 1995.

L. Molgedey and H. G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Physical Review Letters*, 72(23):3634–3637, 1994.

J. J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. In G. Watson, editor, *Lecture Notes in Mathematics*, volume 630, pages 105–116. Springer-Verlag, 1978.

E. Moreau. A generalization of joint-diagonalization criteria for source separation. *IEEE Trans. on Signal Processing*, 49(3):530–541, March 2001.

N. Murata, S. Ikeda, and A. Ziehe. An approach to blind source separation based on temporal structure of speech signals. *Neurocomputing*, 41(1-4):1–24, August 2001.

B. Noble and W. Daniel. *Applied matrix algebra*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1977.

M. R. Osborne. Nonlinear least squares - the Levenberg algorithm revisited. *Journal of Australian Mathematical Society, Series B*, 19:343–357, 1976.

L. Parra and C. Spence. Convolutive blind source separation based on multiple decorrelation. In *Proc. IEEE Workshop on Neural Networks for Signal Processing (NNSP'97)*, Cambridge, UK, 1998.

D.-T. Pham. Joint approximate diagonalization of positive definite matrices. *SIAM J. on Matrix Anal. and Appl.*, 22(4):1136–1152, 2001.

D.-T. Pham and J.-F. Cardoso. Blind separation of instantaneous mixtures of non-stationary sources. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 187–193, Helsinki, Finland, 2000.

D.-T. Pham and J.-F. Cardoso. Blind separation of instantaneous mixtures of non-stationary sources. *IEEE Trans. Sig. Proc.*, 49(9):1837–1848, 2001.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in C*. Cambridge University Press., Cambridge, 1992.

L. Tong, V. C. Soon, and Y. Huang. Indeterminacy and identifiability of identification. *IEEE Trans. on Circuits and Systems*, 38(5):499–509, 1991.

A.-J. van der Veen. Joint diagonalization via subspace fitting techniques. In *Proc. ICASSP*, volume 5, 2001.

A. J. van der Veen, P. B. Ober, and E. F. Deprettere. Azimuth and elevation computation in high resolution DOA estimation. *IEEE Trans. Signal Processing*, 40(7):1828–1832, 1992.

A. J. van der Veen, M. C. Vanderveen, and A. Paulraj. Joint angle and delay estimation using shift-invariance techniques. *IEEE Trans. Signal Processing*, 46(2):405–418, 1998.

H. A. Van der Vorst and G. H. Golub. 150 years old and still alive: Eigenproblems. In *The State of the Art in Numerical Analysis*, volume 63, pages 93–120. Oxford University Press, 1997. URL `citeseer.nj.nec.com/vandervorst97years.html`.

R. Vigário, V. Jousmäki, M. Hämäläinen, R. Hari, and E. Oja. Independent component analysis for identification of artifacts in magnetoencephalographic recordings. In Jordan, Kearns, and Solla, editors, *Proc. NIPS 10*. The MIT Press, 1998.

C. von der Malsburg and W. Schneider. A neural cocktail-party processor. *Biological Cybernetics*, 54:29–40, 1986.

H.-C. Wu and J. C. Principe. Simultaneous diagonalization in the frequency domain (SDIF) for source separation. In *Proc. Int. Workshop on Independent Component Analysis and Blind Source Separation (ICA'99)*, pages 245–250, Aussois, France, January 11–15, 1999.

G. Wübbeler, A. Ziehe, B.-M. Mackert, K.-R. Müller, L. Trahms, and G. Curio. Independent component analysis of non-invasively recorded cortical magnetic DC-fields in humans. *IEEE Transactions on Biomedical Engineering*, 47(5):594–599, 2000.

A. Yeredor. Blind source separation via the second characteristic function. *Signal Processing*, 80 (5):897–902, 2000.

A. Yeredor. Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. *IEEE Trans. on Sig. Proc.*, 50(7):1545–1553, July 2002.

M. Zibulevsky. Relative Newton method for quasi-ML blind source separation. In *Proc. 4th Intern. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 897–902, Nara, Japan, 2003.

A. Ziehe, P. Laskov, K.-R. Müller, and G. Nolte. A linear least-squares algorithm for joint diagonalization. In *Proc. 4th Intern. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 469–474, Nara, Japan, 2003.

A. Ziehe and K.-R. Müller. TDSEP–an efficient algorithm for blind separation using time structure. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN'98)*, pages 675–680, Skövde, Sweden, 1998.

A. Ziehe, K.-R. Müller, G. Nolte, B.-M. Mackert, and G. Curio. Artifact reduction in magnetoneurography based on time-delayed second-order correlations. *IEEE Trans Biomed Eng.*, 47(1): 75–87, January 2000a.

A. Ziehe, G. Nolte, G. Curio, and K.-R. Müller. OFI: Optimal filtering algorithms for source separation. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 127–132, Helsinki, Finland, 2000b.