PERFORMANCE IMPROVEMENT OF THE FFDIAG ALGORITHM BASED ON PRE-DIAGONALIZATION BY CLOSED-FORM JOINT DIAGONALIZERS

Miho Murota and Akira Tanaka
Division of Computer Science
Hokkaido University
N14W9, Kita-ku, Sapporo, 060-0814 Japan
email: {m-muro,takira}@main.ist.hokudai.ac.jp

ABSTRACT
In this paper, we try to improve the performance of the FFDiag algorithm, which is one of the state-of-the-art iteration-based approximate joint diagonalizers of a given set of real-valued symmetric matrices. The key idea of the improvement is pre-diagonalization by a closed-form joint diagonalizer whose computational cost is smaller than that of the FFDiag algorithm. Numerical experiments for approximate joint diagonalization of a set of real-valued symmetric matrices, that are randomly generated, are conducted to verify the efficacy of the proposed scheme in terms of computational costs and joint diagonalization performance.

KEY WORDS
Joint diagonalization, FFDiag, pre-processing, closed-form, computational cost

1 Introduction
Joint diagonalization of a given set of real-valued symmetric matrices is one of important problems in the field of statistical signal processing such as blind source separation based on the second order statistics of observations. In practical problems, a given set of real-valued symmetric matrices is not always jointly diagonalizable strictly since they are generated from real-world data that is perturbed from an assumed ideal model. Therefore, we need an algorithm to obtain an approximate joint diagonalizer that makes given matrices as diagonal as possible. So far, many iterative algorithms for the approximate joint diagonalization problem have been proposed [1, 2, 3, 4]. Among them, the FFDiag algorithm [4] is recognized as one of the state-of-the-art approximate joint diagonalizers of real-valued symmetric matrices in terms of computational cost and diagonalization performance. The FFDiag algorithm rapidly converges when an initial point is in the neighborhood of the optimal solution. However, when the initial point is far from the optimal solution, its convergence performance degrades since this algorithm is constructed based on the assumption that an intermediate solution in each iteration step is in the neighborhood of the optimal solution.

In our previous works [5, 6], we obtained closed-form joint diagonalizers on the basis of the theorems given in [7]. Although the diagonalization performance of these closed-form solutions is lower than that of the iterative methods such as the FFDiag algorithm for approximate joint diagonalization problems, their computational costs are smaller than that of the iteration-based methods.

In this paper, we construct a novel joint diagonalization scheme that adopts the closed-form joint diagonalizers as pre-processor for the FFDiag algorithm, aiming to obtain a better initial point for the FFDiag algorithm. We also verify the efficacy of the proposed scheme by numerical experiments for approximate joint diagonalization of a set of real-valued symmetric matrices, that are randomly generated, in terms of computational costs and diagonalization performance.

2 Overview of the FFDiag Algorithm
In this section, we formulate the joint diagonalization problem and briefly review the FFDiag algorithm introduced in [4].

Let $C_k \in \mathbb{R}^{N \times N}$, $k \in \{1, \ldots, K\}$ be a series of real-valued symmetric matrices modeled as

$$C_k = A \Lambda_k A\',$$

where $A \in \mathbb{R}^{N \times n}$ and $\Lambda_k \in \mathbb{R}^{n \times n}$ denote a certain non-singular matrix and a certain diagonal matrix; and the superscript $'$ denotes the transposition operator, respectively. The aim of the joint diagonalization is to obtain a non-singular matrix $V \in \mathbb{R}^{N \times N}$ that makes $VC_kV'$ diagonal for all $k \in \{1, \ldots, K\}$.

In the FFDiag algorithm, the cost function for joint diagonalization is defined as

$$J(V) = \sum_{k=1}^{K} \|VC_kV' - \text{diag}(VC_kV')\|_F^2,$$

where $\| \cdot \|_F$ denotes the Frobenius norm of a matrix and $\text{diag}(\cdot)$ denotes the operator that replaces non-diagonal components of a given matrix to zeros. The minimization of $J(V)$ is conducted by the following multiplicative iteration formula:

$$V^{(n+1)} = (I_N + W^{(n)} \cdot V^{(n)}),$$

where $W^{(n)}$...
where \( I_N \) and \( n \) denote the identity matrix of degree \( N \) and the iteration number; and \( W^{(n)} \in \mathbb{R}^{N \times N} \) denotes a certain matrix whose diagonal elements are zeros. The matrix \( W^{(n)} \) has to be constructed to minimize \( J(V) \). Its details are reviewed later. In order to avoid the trivial solution \( V = O_N \), where \( O_N \) denotes the zero matrix in \( \mathbb{R}^{N \times N} \), we have to choose a non-singular matrix as the initial point \( V^{(0)} \) and have to keep the invertibility of \( I_N + W^{(n)} \). The invertibility of \( I_N + W^{(n)} \) is realized by the normalization

\[
W^{(n)} \leftarrow \frac{\theta}{\|W^{(n)}\|_F} W^{(n)}
\tag{4}
\]

with a certain fixed \( \theta \in (0, 1) \) when \( \|W^{(n)}\|_F > \theta \). Note that the validity of Eq.(4) for the invertibility is guaranteed by the Levi-Desplanques theorem (see [4] for more details). The update formula Eq.(3) is repeated until Eq.(2) converges.

Next, we review the construction of \( W^{(n)} \) in the FFDiag algorithm. Let us consider the result of diagonalization for \( C_k \) in the \((n+1)\)th iteration step, written as

\[
C_k^{(n+1)} = (I_n + W^{(n)})C_k^{(n)}(I_n + W^{(n)})',
\tag{5}
\]

where \( C_k^{(n)} = V^{(n)}C_k^{(n)}V^{(n)'} \). Let \( D_k^{(n)} = \text{diag}(C_k^{(n)}) \) and let \( E_k^{(n)} = C_k^{(n)} - D_k^{(n)} \). Then, we have

\[
C_k^{(n+1)} = (I + W^{(n)})(D_k^{(n)} + E_k^{(n)})(I + W^{(n)})' \\
\approx D_k^{(n)} + W^{(n)}D_k^{(n)} + D_k^{(n)}W^{(n)} + E_k^{(n)} \\
= D_k^{(n)} + R_k^{(n)}
\]

by assuming that \( \|W^{(n)}\|_F \) and \( \|E_k^{(n)}\|_F \) are comparatively small and ignoring the quadratic terms consisting of \( W^{(n)} \) and \( E_k^{(n)} \). Since \( D_k^{(n)} \) is diagonal, the cost function Eq.(2) for \((n+1)\)th iteration is reduced to

\[
J(W^{(n)}) = \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} ((R_k^{(n)})_{ij})^2.
\tag{6}
\]

Since

\[
(R_k^{(n)})_{ij} = (W^{(n)})_{ij}(D_k^{(n)})_{jj} + (E_k^{(n)})_{ij}
\tag{7}
\]

holds, \( (W^{(n)})_{ij} \) and \( (W^{(n)})_{ji} \) only depend on \( (D_k^{(n)})_{ii} \), \( (D_k^{(n)})_{jj} \), \( (E_k^{(n)})_{ij} \) and \( (E_k^{(n)})_{ji} \). Therefore, the minimization of \( J(W^{(n)}) \) can be decomposed into the separated minimization problems for each pair of \( (W^{(n)})_{ij} \) and \( (W^{(n)})_{ji} \). Let

\[
D_k^{(n,ij)} = \begin{bmatrix} (D_k^{(n)})_{jj} & (D_k^{(n)})_{ij} \\ (D_k^{(n)})_{ji} & (D_k^{(n)})_{ii} \end{bmatrix},
\tag{8}
\]

\[
w_k^{(n,ij)} = \begin{bmatrix} (W^{(n)})_{ij} \\ (W^{(n)})_{ji} \end{bmatrix},
\tag{9}
\]

\[
e_k^{(n,ij)} = \begin{bmatrix} (E_k^{(n)})_{ij} \\ (E_k^{(n)})_{ji} \end{bmatrix},
\tag{10}
\]

then the cost function for \( w^{(n,ij)} \) is reduced to

\[
J(w^{(n,ij)}) = \sum_{k=1}^{K} \|D_k^{(n,ij)}w^{(n,ij)} + e_k^{(n,ij)}\|^2,
\tag{11}
\]

whose minimization is easily solved. Note that the computational order for obtaining \( W^{(n)} \) is \( O(KN^2) \) and overall computational order for each iteration is \( O(KN^3) \) which is dominated by matrix multiplication Eq.(5) for \( K \) matrices. Also note that the minimizer of Eq.(11) can be explicitly written as shown in [4], which may reduce the actual computational time.

As mentioned in the previous section, the FFDiag algorithm rapidly converges if the initial point \( V^{(0)} \) is in the neighborhood of the optimal solution. However, when the initial point \( V^{(0)} \) is far from the optimal solution, its convergence performance degrades since \( \|W^{(n)}\|_F \) and \( \|E_k^{(n)}\|_F \) are assumed to be comparatively small, which may be broken when the iteration number \( n \) is small in general. Therefore, we have to obtain a better initial point to overcome this problem.

## 3 The Proposed Method

In this section, we introduce a method for improving the convergence performance of the FFDiag algorithm. The key idea of the proposed method is adopting closed-form joint diagonalizers as pre-processor for the FFDiag algorithm.

The procedure of the proposed method is quite simple. Let \( V_c \) be a certain closed-form joint diagonalizer of \( C_k \), \( (k \in \{1, \ldots, K\}) \); and we calculate

\[
\tilde{C}_k = V_c C_k V_c' \tag{12}
\]

for all \( k \in \{1, \ldots, K\} \). Then, the FFDiag algorithm is applied to the set of symmetric matrices \( \tilde{C}_k \), \( (k \in \{1, \ldots, K\}) \). Let \( V_f \) be the joint diagonalizer of \( \tilde{C}_k \) obtained through the FFDiag algorithm, then the overall joint diagonalizer by the proposed method is given as

\[
V = V_f V_c \tag{13}
\]

Since \( \tilde{C}_k \) is expected to be near diagonal than the original \( C_k \), the convergence performance of the FFDiag algorithm is expected to be improved.

In the following subsections, we introduce closed-form joint diagonalizers of a given set of symmetric matrices introduced in our previous works [5, 6].

### 3.1 Closed-Form Joint Diagonalizer for Non-Negative Definite Symmetric Matrices

In many applications of the joint diagonalization, target matrices are non-negative definite (n.n.d.) such as covariance matrices of observations.

Here, we introduce a closed-form joint diagonalizer of a given set of n.n.d. symmetric matrices along with the
descriptions in [5]. The following theorem plays a crucial role for construction of a closed-form joint diagonalizer.

**Theorem 1** [7] Let \( C_k \in \mathbb{C}^{N \times N} \ (k \in \{1, \ldots, K\}) \) be n.n.d. Hermitian matrices and let \( B = \sum_{k=1}^{K} C_k \). There exists a non-singular matrix \( V \) that makes \( VC_kV^* \) diagonal for all \( k \in \{1, \ldots, K\} \), where the superscript * denotes the adjoint (transposition and conjugate) operator, if and only if

\[
C_i B^{-1} C_j = C_j B^{-1} C_i
\]

holds for any \( i, j \in \{1, \ldots, K\} \), where \( B^{-1} \) denotes an arbitrary generalized inverse matrix of \( B[7] \)

Since \( C_k \) is modeled by Eq.(1), \( C_k \) can be jointly diagonalized by \( V = A^{-1} \). Therefore, Eq.(14) holds for any \( i, j \in \{1, \ldots, K\} \). Let \( B^+ = LL^* \) be a full-rank decomposition of \( B^+ \), then we have

\[
C_i L L^* C_j = C_j L L^* C_i,
\]

which implies that

\[
(L^* C_i L)(L^* C_j L) = (L^* C_j L)(L^* C_i L)
\]

holds. Thus, it is concluded that \( L^* C_i L \) and \( L^* C_j L \) are commutative symmetric matrices for any \( i, j \in \{1, \ldots, K\} \); and they can be jointly diagonalized by the same unitary matrix \( U \) consisting of eigenvectors of \( L^* C_i L \) for any \( i \in \{1, \ldots, K\} \). Let

\[
L^* C_k L = U D_k U^*
\]

be the eigenvalue decomposition of \( L^* C_k L \) with an arbitrary \( k \in \{1, \ldots, K\} \), then,

\[
V = U^* L^*
\]

gives a closed-form joint diagonalizer of \( C_k \), \( k \in \{1, \ldots, K\} \) \footnote{Note that we can use the Moore-Penrose generalized inverse matrix \( B^+ \) [7] for \( B^{-1} \) as shown in [5].}. Note that we can replace the symbol * to ' since the considered matrices are real. Also note that the computational order of this closed-form joint diagonalizer is \( O(N^3) \) which is dominated by the full-rank decomposition of \( B^+ \) and the eigenvalue decomposition of \( L^* C_k L \) with a certain \( k \in \{1, \ldots, K\} \).

We call this closed-form joint diagonalizer 'Simple Closed-Form Joint Diagonalizer for N.N.D. matrices' abbreviated by 'SI-CJF-NND', hereafter; and the proposed method by this closed-form is denoted by 'SI-CJF-NND + FFDiag'.

If given n.n.d. symmetric matrices are strictly and jointly diagonalizable, \( U \) in Eq.(17) is essentially invariant for any \( k \in \{1, \ldots, K\} \). However, \( U \) depends on \( k \) in case that given n.n.d. symmetric matrices are not jointly diagonalizable strictly. In such case, Eq.(17) is reduced to

\[
L^* C_k L = U_k D_k U_k^* \approx (U + \delta U_k) D_k (U + \delta U_k)^*,
\]

Therefore, one reasonable resolution to obtain a better \( U_k \) is to adopt \( U_k \) corresponding to the smallest \( \|\delta U_k\|_F \). In [9], we introduced an approximated estimate for \( \|\delta U_k\|_F \) written as

\[
H(k) = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \frac{d_i^{(k)} d_j^{(k)}}{(d_i^{(k)} - d_j^{(k)})^2},
\]

where \( d_i^{(k)} \) is the ith diagonal element of \( D_k \) in Eq.(17). Thus, it is expected that \( U_{k_{opt}} \) with

\[
k_{opt} = \arg \min_k H(k)
\]

gives a better \( U \) for Eq.(18). Please refer to [9] for more details including the assumed perturbation model. Note that the computational order of this scheme is \( O(KN^3) \) which is dominated by the eigenvalue decomposition of \( L^* C_k L \) for all \( k \in \{1, \ldots, K\} \) which is needed to calculate \( H(k) \).

We call the closed-form joint diagonalizer Eq.(18) with the selection scheme shown above 'SElected Closed-Form Joint Diagonalizer for N.N.D. matrices' abbreviated by 'SE-CJF-NND', hereafter; and the proposed method by this closed-form is denoted by 'SE-CJF-NND + FFDiag'.

### 3.2 Closed-Form Joint Diagonalizer for General Cases

In the FFDiag algorithm, input matrices are not limited to n.n.d. Thus, in order to apply the proposed scheme to symmetric matrices which may have negative eigenvalues, we have to provide a closed-form joint diagonalizer for such matrices. The following theorems play crucial roles for this problem.

**Theorem 2** [7] Let \( C_k \in \mathbb{C}^{N \times N} \ (k \in \{1, \ldots, K\}) \) be semisimple matrices. \( C_k C_i = C_i C_k \) holds for any \( i, j \in \{1, \ldots, K\} \) if and only if \( C_k \in C_{2f} \), which is needed to calculate \( H(k) \).

We call this closed-form joint diagonalizer 'Simple Closed-Form Joint Diagonalizer for N.N.D. matrices' abbreviated by 'SI-CJF-NND', hereafter; and the proposed method by this closed-form is denoted by 'SI-CJF-NND + FFDiag'.

If given n.n.d. symmetric matrices are strictly and jointly diagonalizable, \( U \) in Eq.(17) is essentially invariant for any \( k \in \{1, \ldots, K\} \). However, \( U \) depends on \( k \) in case that given n.n.d. symmetric matrices are not jointly diagonalizable strictly. In such case, Eq.(17) is reduced to

\[
L^* C_k L = U_k D_k U_k^* \approx (U + \delta U_k) D_k (U + \delta U_k)^*,
\]

for a certain \( C_1 \)

\footnote{Note that we can use the Moore-Penrose generalized inverse matrix \( B^+ \) [7] for \( B^{-1} \) as shown in [5].}

\footnote{Note that we can use \( C_1^+ \) for \( C_1 \) as shown in [6].}
As the same with the \textit{n.n.d.} case discussed in the previous subsection, \( C_k \) can be jointly diagonalized by \( V = A^{-1} \) since \( C_k \) is modeled by Eq.\((1)\). Therefore,

\[
C_iC_j^+C_j = C_jC_i^+C_i
\]  
(22)

holds for any \( i, j \in \{1, \ldots, K\} \) from Theorem 3, which implies that

\[
(C_iC_j^+)(C_jC_i^+) = (C_jC_i^+)(C_iC_j^+)
\]  
(23)

holds for any \( i, j \in \{1, \ldots, K\} \). Moreover, it is guaranteed that \( C_iC_j^+ \) is semisimple and its eigenvalues are real from Theorem 3. From Eq.\((23)\) and Theorem 2, it is concluded that \( C_iC_j^+ \) can be represented by a polynomial of a certain semisimple matrix \( C \) for any \( i \in \{1, \ldots, K\} \). Accordingly, eigenvectors of \( C \) are also eigenvectors of \( C_iC_j^+ \) for any \( i \in \{1, \ldots, K\} \) and vice versa.

In practical problems, given matrices \( C_k \), \( k \in \{1, \ldots, K\} \) are non-singular and simple. Therefore, we assume that \( \mathcal{R}(C_k) \subset \mathcal{R}(C_i) \) holds. Let

\[
C_kC_k^{-1} = UD_kU^{-1}
\]  
(24)

be the eigenvalue decomposition of \( C_kC_k^{-1} \) with an arbitrarily fixed \( k \), then

\[
C_iC_i^{-1} = UD_iU^{-1}
\]  
(25)

holds for any \( i \in \{1, \ldots, K\} \). Since \( C_k \) is modeled by Eq.\((1)\), we have

\[
C_i^{-1} = (A')^{-1}A_1^{-1}A^{-1}
\]  
(26)

and

\[
C_kC_k^{-1} = AA_kA_1^{-1}A^{-1}
\]  
(27)

Therefore, \( U \) and \( A \) are essentially identical except the permutation and scaling ambiguities, which implies \( V = U^{-1} \) gives a closed-from joint diagonalizer of \( C_k \), \( k \in \{1, \ldots, K\} \). Note that the computational order of the above closed-form joint diagonalizer is \( O(N^3) \) which is dominated by the eigenvalue decomposition of \( C_kC_k^{-1} \). Also note that more general cases including singular cases are discussed in [6]. Please refer to [6] for more details.

We call this closed-form joint diagonalizer ‘Simple Closed-form Joint Diagonalizer for GENerall matrices’ abbreviated by ‘SI-CJD-GEN’ hereafter; and the proposed method by this closed-form is denoted by ‘SI-CJD-GEN + FFDiag’. When given symmetric matrices are not jointly diagonalizable strictly, eigenvalues of \( C_kC_k^{-1} \) may have complex eigenvalues. In such cases, we can not use this closed-form joint diagonalizer as the pre-processor for the FFDiag algorithm and we have to use the FFDiag algorithm directly.

4 Numerical Experiments

In this section, we investigate the performance of the proposed scheme by numerical experiments for approximate joint diagonalization of randomly generated matrices in terms of computational time and the performance of the joint diagonalization.

As the criterion to evaluate the performance of the joint diagonalization, we adopt

\[
Z(G) = \frac{1}{2} \left[ \sum_{i=1}^{N} \left( \sum_{j=1}^{N} \frac{|G_{ij}|^2}{\max{\{G_{ik}\}^2}} - 1 \right) \right] + \frac{1}{2} \sum_{j=1}^{N} \left( \sum_{i=1}^{N} \frac{|G_{ij}|^2}{\max{\{G_{jk}\}^2}} - 1 \right)
\]  
(28)

introduced in [4], where \( G = VA \). Note that \( Z(G) \) is positive and \( Z(G) = 0 \) is achieved only when \( G \) is a permuted version of a strict diagonal matrix, which implies that a smaller \( Z(G) \) implies a better performance. All methods were implemented by the GNU Octave 3.0.3; and we adopt \( \theta = 0.9 \) in the FFDiag algorithm.

4.1 Non-Negative Definite Case

We investigate the performance of the proposed scheme for a set of \textit{n.n.d.} symmetric matrices. We adopt \( N = 5, 10 \) and \( K = 15, 30 \). The elements of the matrix \( A \) are drawn from a standard normal distribution; and diagonal elements of \( A_k \) are drawn from a uniform distribution in the range \((0, 1]\). Target matrices are generated by

\[
C_k = AA_kA' + \sigma^2 R_k R_k',
\]  
(29)

where elements of \( R_k \) are also drawn from a standard normal distribution and \( \sigma (= 0.001, 0.01, 0.1) \) is a parameter that controls the strength of the non-diagonalizable component \( R_k R_k' \).

We compare computational time and the performance of the joint diagonalization evaluated by \( \log_{10} Z(G) \) for ‘SI-CJD-NND’, ‘SE-CJD-NND’, ‘SI-CJD-NND + FFDiag’, ‘SE-CJD-NND + FFDiag’, and ‘FFDiag’. In ‘SI-CJD-NND’ and ‘SI-CJD-NND + FFDiag’, we adopt \( L'C_kL \) to obtain the orthogonal matrix \( U \). The averaged results over 100 trials are shown in Table 1.

According to these results, it is confirmed that the proposed scheme outperforms the FFDiag algorithm in terms of computational time in all cases. However, its advantage is small with a larger \( \sigma \), which seems to be natural since the joint diagonalization performance of the closed form solutions degrades with a larger \( \sigma \). It is also confirmed that the joint diagonalization performance of the proposed scheme outperforms that of the FFDiag algorithm even though the proposed scheme does not include explicit idea to improve the joint diagonalization performance. Its theoretical background will be investigated in our future works.

4.2 General Case

We investigate the performance of the proposed scheme for a set of general symmetric matrices that may have negative
Table 1. Averaged computational time and averaged joint diagonalization performance for n.n.d. case (upper : $\sigma = 0.001$ / middle : $\sigma = 0.01$ / lower : $\sigma = 0.1$).

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>$K = 15$</th>
<th>$K = 30$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(ms)</td>
<td>$\log_{10} Z(G)$</td>
</tr>
<tr>
<td>$\sigma = 0.001$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N = 5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_NND</td>
<td>14.1</td>
<td>-6.42</td>
</tr>
<tr>
<td>SE_CJD_NND</td>
<td>23.9</td>
<td>-6.98</td>
</tr>
<tr>
<td>SL_CJD_NND + FFDiag</td>
<td>130.5</td>
<td>-8.11</td>
</tr>
<tr>
<td>SE_CJD_NND + FFDiag</td>
<td>137.4</td>
<td>-8.11</td>
</tr>
<tr>
<td>FFDiag</td>
<td>280.5</td>
<td>-7.52</td>
</tr>
<tr>
<td>$N = 10$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_NND</td>
<td>14.8</td>
<td>-6.60</td>
</tr>
<tr>
<td>SE_CJD_NND</td>
<td>52.3</td>
<td>-5.04</td>
</tr>
<tr>
<td>SL_CJD_NND + FFDiag</td>
<td>528.9</td>
<td>-6.67</td>
</tr>
<tr>
<td>SE_CJD_NND + FFDiag</td>
<td>370.2</td>
<td>-6.67</td>
</tr>
<tr>
<td>FFDiag</td>
<td>1498.0</td>
<td>-6.00</td>
</tr>
<tr>
<td>$\sigma = 0.01$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N = 5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_NND</td>
<td>14.2</td>
<td>-2.44</td>
</tr>
<tr>
<td>SE_CJD_NND</td>
<td>23.9</td>
<td>-2.87</td>
</tr>
<tr>
<td>SL_CJD_NND + FFDiag</td>
<td>177.2</td>
<td>-4.18</td>
</tr>
<tr>
<td>SE_CJD_NND + FFDiag</td>
<td>204.9</td>
<td>-4.18</td>
</tr>
<tr>
<td>FFDiag</td>
<td>294.8</td>
<td>-3.71</td>
</tr>
<tr>
<td>$N = 10$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_NND</td>
<td>14.7</td>
<td>-1.03</td>
</tr>
<tr>
<td>SE_CJD_NND</td>
<td>52.3</td>
<td>-1.45</td>
</tr>
<tr>
<td>SL_CJD_NND + FFDiag</td>
<td>1100.8</td>
<td>-3.00</td>
</tr>
<tr>
<td>SE_CJD_NND + FFDiag</td>
<td>1135.4</td>
<td>-3.00</td>
</tr>
<tr>
<td>FFDiag</td>
<td>1804.1</td>
<td>-2.65</td>
</tr>
<tr>
<td>$\sigma = 0.1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N = 5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_NND</td>
<td>14.2</td>
<td>-0.04</td>
</tr>
<tr>
<td>SE_CJD_NND</td>
<td>23.9</td>
<td>-0.28</td>
</tr>
<tr>
<td>SL_CJD_NND + FFDiag</td>
<td>585.8</td>
<td>-0.90</td>
</tr>
<tr>
<td>SE_CJD_NND + FFDiag</td>
<td>594.1</td>
<td>-0.90</td>
</tr>
<tr>
<td>FFDiag</td>
<td>647.5</td>
<td>-0.78</td>
</tr>
<tr>
<td>$N = 10$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_NND</td>
<td>14.8</td>
<td>0.69</td>
</tr>
<tr>
<td>SE_CJD_NND</td>
<td>52.3</td>
<td>0.65</td>
</tr>
<tr>
<td>SL_CJD_NND + FFDiag</td>
<td>3951.6</td>
<td>-0.11</td>
</tr>
<tr>
<td>SE_CJD_NND + FFDiag</td>
<td>4004.0</td>
<td>-0.11</td>
</tr>
<tr>
<td>FFDiag</td>
<td>4376.0</td>
<td>-0.02</td>
</tr>
</tbody>
</table>

The diagonal elements of $A_k$ are drawn from a uniform distribution in the range $[-1, 1]$. Other conditions are the same with the n.n.d. case.

We compare computational time and the performance of the joint diagonalization evaluated by $\log_{10} Z(G)$ for ‘SI-CJD-GEN’, ‘SI-CJD-GEN + FFDiag’, and ‘FFDiag’. In ‘SI-CJD-GEN’, ‘SI-CJD-GEN + FFDiag’, and ‘FFDiag’, we adopt $C_2C_1^{-1}$ to obtain $U$. In ‘SI-CJD-GEN + FFDiag’, if $C_2C_1^{-1}$ has complex eigenvalues, we abandon the closed-form joint diagonalizer, which implies that the performance of ‘SI-CJD-GEN + FFDiag’ is almost the same with that of ‘FFDiag’. The averaged results over 100 trials are shown in Table 2.

According to these results, it is confirmed that the proposed scheme outperforms the FFDiag algorithm in terms of computational time in all cases as the same with n.n.d. case. However, its advantage is quite small with a larger $\sigma$, which is caused by the failure to obtain a closed-form joint diagonalizer in many cases due to complex eigenvalues of $C_2C_1^{-1}$. As the same with the n.n.d. case, it is confirmed that the joint diagonalization performance of the proposed scheme outperforms that of the FFDiag algorithm.

## 5 Conclusion

In this paper, we improved the performance of the FFDiag algorithm, which is one of the state-of-the-art joint diagonalization algorithms, by using closed-form joint diagonalizers as pre-processors for the FFDiag algorithm. We confirmed that the proposed scheme outperformed the original FFDiag algorithm in terms of both computational time and joint diagonalization performance. Theoretical analy-
Table 2. Averaged computational time and averaged joint diagonalization performance for general case (upper : $\sigma = 0.001$ / middle : $\sigma = 0.01$ / lower : $\sigma = 0.1$).

<table>
<thead>
<tr>
<th>$\sigma = 0.001$</th>
<th>$K = 15$</th>
<th>$K = 30$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(ms)</td>
<td>$\log_{10} Z(G)$</td>
</tr>
<tr>
<td>N = 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_GEN</td>
<td>20.7</td>
<td>-6.66</td>
</tr>
<tr>
<td>SL_CJD_GEN + FFDiag</td>
<td>137.7</td>
<td>-8.82</td>
</tr>
<tr>
<td>FFDiag</td>
<td>235.8</td>
<td>-8.53</td>
</tr>
<tr>
<td>N = 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_GEN</td>
<td>20.8</td>
<td>-4.31</td>
</tr>
<tr>
<td>SL_CJD_GEN + FFDiag</td>
<td>571.1</td>
<td>-6.90</td>
</tr>
<tr>
<td>FFDiag</td>
<td>1590.5</td>
<td>-6.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\sigma = 0.01$</th>
<th>$K = 15$</th>
<th>$K = 30$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(ms)</td>
<td>$\log_{10} Z(G)$</td>
</tr>
<tr>
<td>N = 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_GEN</td>
<td>20.5</td>
<td>-2.61</td>
</tr>
<tr>
<td>SL_CJD_GEN + FFDiag</td>
<td>230.4</td>
<td>-4.51</td>
</tr>
<tr>
<td>FFDiag</td>
<td>331.1</td>
<td>-4.24</td>
</tr>
<tr>
<td>N = 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_GEN</td>
<td>20.7</td>
<td>-0.79</td>
</tr>
<tr>
<td>SL_CJD_GEN + FFDiag</td>
<td>1199.8</td>
<td>-3.25</td>
</tr>
<tr>
<td>FFDiag</td>
<td>1767.5</td>
<td>-3.04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\sigma = 0.1$</th>
<th>$K = 15$</th>
<th>$K = 30$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time(ms)</td>
<td>$\log_{10} Z(G)$</td>
</tr>
<tr>
<td>N = 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_GEN</td>
<td>20.5</td>
<td>-0.12</td>
</tr>
<tr>
<td>SL_CJD_GEN + FFDiag</td>
<td>589.3</td>
<td>-0.87</td>
</tr>
<tr>
<td>FFDiag</td>
<td>615.1</td>
<td>-0.87</td>
</tr>
<tr>
<td>N = 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL_CJD_GEN</td>
<td>21.1</td>
<td>1.09</td>
</tr>
<tr>
<td>SL_CJD_GEN + FFDiag</td>
<td>3681.8</td>
<td>-0.14</td>
</tr>
<tr>
<td>FFDiag</td>
<td>3742.6</td>
<td>-0.13</td>
</tr>
</tbody>
</table>

sis for the better joint diagonalization performance of the proposed scheme is one of our future works that should be undertaken.

Acknowledgment

This work was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), 24500001.

References


