IPSJ Transactions on Mathematical Modeling and Its Applications Vol.14 No.3 68–75 (Aug. 2021)

Regular Paper

On an Implementation of the One-Sided Jacobi Method With High Accuracy

Masami Takata[2,a)] Sho Araki[1,b)] Takahiro Miyamae[3,c)] Kinji Kimura[3,d)]
Yoshimasa Nakamura[1,†1,e)]

Received: October 21, 2020, Revised: February 17, 2021,
Accepted: March 12, 2021

Abstract: The one-sided Jacobi method for performing singular value decomposition can compute all singular values and singular vectors with high accuracy. Additionally, the computation cost is insignificant for comparatively small matrices. However, in the case of the conventional implementation in Linear Algebra PACKage, the subroutine may not be able to compute a singular vector with sufficient orthogonality. To avoid this problem, we propose a novel implementation of the one-sided Jacobi method. In the proposed implementation, a Givens rotation with high accuracy and fused multiply-accumulate are adopted.

Keywords: Singular value decomposition, Jacobi method, false-position method, secant method

1. Introduction

Many mathematical applications require a generalized eigenvalue formula that comprises a symmetric matrix and positive definite symmetric matrix. However, they use only some eigenvalues and the corresponding eigen vectors. The Sakurai–Sugiura method [15], also called truncated eigenvalue decomposition, uses a column space, which is computed by decomposing a rectangular matrix via singular value decomposition. Generally, a given matrix is transformed into a bidiagonal one using a Householder transformation [3] as a preprocessing method. In Ref. [1], a computation method for column space, adopted to a bidiagonal matrix, was proposed. The method combined the differential qd algorithm with shifts [7], [12] and orthogonal qd algorithm with shifts (OQDS) [11]. The Sakurai–Sugiura method requires only a column space, which is based on left singular vectors, in a given upper bidiagonal matrix. Because the row space in a lower bidiagonal matrix is equal to the column space in an upper bidiagonal matrix, the row space can be computed using right singular vectors, achieved by employing the OQDS method, which was proposed in Ref. [1].

In Ref. [1], the test matrices were bidiagonal. However, in the Sakurai–Sugiura method, the singular value decomposition of an upper triangular matrix, whose dimension is small, is required. In the case of bidiagonalization using the Householder transformation, a rounding error may occur computationally. Thus, an algorithm which does not employ the Householder transformation should be used.

The Jacobi method for performing singular value decomposition can compute all singular values and singular vectors. James Demmel and Kresimir Veselic reported that the Jacobi method was more accurate than the QR method [4]; additionally, the computation cost for the former was insignificant for comparatively small matrices. One- and two-sided Jacobi methods have been proposed as the implementations of the Jacobi method for singular value decomposition [2], [5], [6], [8], [9]. They aim for computation accuracy and speed, respectively. In this study, to perform singular value decomposition with high accuracy, we propose a novel implementation of the one-sided Jacobi method, whose conventional implementation has theoretically high accuracy. Practically, there exist cases wherein singular vectors with sufficient orthogonality cannot be computed. To avoid this problem, in the proposed implementation, a Givens rotation with high accuracy and the fused multiply-accumulate are adopted. We confirmed that our implemented one-sided Jacobi method has higher accuracy than that of the one-sided Jacobi method implemented in Linear Algebra PACKage (LAPACK) [10].

In Section 2, the outline of the one-sided Jacobi method is shown. Section 3 introduces a conventional implementation of the one-sided Jacobi method. In Section 4, we prepare the Givens rotation with high accuracy. Section 5, proposes a novel implementation of the one-sided Jacobi method. Finally, in Section 6, we compare the proposed implementation with the subroutine in LAPACK.

2. Outline of the One-sided Jacobi Method

In a singular value decomposition, an $m\times n$ $(m \geq n)$ rectangular matrix $A \in \mathbb{R}^{m\times n}$ is decomposed to

1 Kyoto University, Kyoto 606–8501, Japan
2 Nara Women's University, Nara 630–8506, Japan
3 University of Fukui, Fukui 910–8507, Japan
†1 Presently with Osaka Seikei University
a) takata@ics.nara-wu.ac.jp
b) araki@amp.i.kyoto-u.ac.jp
c) ms200319@u-fukui.ac.jp
d) kkimur@u-fukui.ac.jp
e) ynaka@i.kyoto-u.ac.jp

$$A \approx U\Sigma V^\top. \tag{1}$$

Here, the columns in the orthogonal matrices $U \in \mathbb{R}^{m\times L}$ and $V \in \mathbb{R}^{n\times L}$ consist of left and right singular vectors, respectively, and the diagonal matrix $\Sigma \in \mathbb{R}^{L\times L}$ has singular values as diagonal elements. These matrices satisfy the following conditions:

$$A\boldsymbol{v}_i = \sigma_i\boldsymbol{u}_i, \quad A^\top\boldsymbol{u}_i = \sigma_i\boldsymbol{v}_i \ (i = 1, \ldots, L), \tag{2}$$

$$U := \left[\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_L\right] \in \mathbb{R}^{m\times L}, \tag{3}$$

$$V := \left[\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L\right] \in \mathbb{R}^{n\times L}, \tag{4}$$

$$\Sigma := \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_L) \in \mathbb{R}^{L\times L}.$$
$$(\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_L > 0), \quad L = \mathrm{Rank}(A) \tag{5}$$

Here, the left and right singular vectors corresponding to the ith singular value σ_i are $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$, respectively.

A and J_i denote an $m \times n$ real matrix and the following Jacobi rotation of the $n \times n$ matrix, respectively.

$$J_i = \begin{pmatrix} I & 0 & \cdots & \cdots & 0 \\ 0 & \cos(\theta_i) & \cdots & -\sin(\theta_i) & \vdots \\ \vdots & \vdots & I & \vdots & \vdots \\ \vdots & \sin(\theta_i) & \cdots & \cos(\theta_i) & 0 \\ 0 & \cdots & \cdots & 0 & I \end{pmatrix}$$

From the right side of the matrix A, the rotations of $J_i (i = 1, \cdots)$ is effected so that the converged matrix B satisfies the column orthogonality $B^\top B = D$ (D is the diagonal matrix).

$$AJ_1J_2\cdots \to B$$

Here, if the norm of the column vector $\boldsymbol{b}_j (j = 1, \cdots, n)$ of the jth column of the matrix B is greater than zero, then the norm is the singular value $\sigma_j (j = 1, \cdots, n)$ of the matrix A. If $V = J_1J_2\cdots$, V is a matrix of the right singular vectors. For a column vector $\boldsymbol{b}_j$ with a positive value σ_j for $j = 1, \cdots, n$, if we divide a column vector $\boldsymbol{b}_j$ by the positive value σ_j, then each jth column vector of B is the jth left singular vector of the matrix A.

3. Conventional Implementation for the One-sided Jacobi Method

Algorithm 1 shows a pseudo code of the one-sided Jacobi method [4]. It requires an $m \times n$ ($m \geq n$) real matrix A and a convergence-judgment threshold *tol* and ensures Σ, U, and V. The terms Σ, U, and V denote the matrices whose elements are singular values, left singular vectors, and right singular vectors in A, respectively. Generally, *tol* is set to $tol = \sqrt{m}\varepsilon$, where ε denotes a machine epsilon, which is adopted in `xGESVJ` implemented in LAPACK, considering the error in the inner-product computation.

The algorithm comprises a double loop with the main part. The outer loop repeats until the result of the inner loop converges. The inner loop performs through a subscript *pairs*, which is given using a subroutine `jacobi_pairs`. This generates a pair that contains at least one pair of all the integers from 1 to n. Thus, *pairs* contains at least $n(n-1)/2$ entries. As an example, we introduce the *pairs* obtained in the simplest subroutine `jacobi_pairs`,

Algorithm 1 Pseudo code of the one-sided Jacobi method

```
Require: $\boldsymbol{A} = [\boldsymbol{a}_1 \ \boldsymbol{a}_2 \ \cdots \ \boldsymbol{a}_n], tol$
Ensure: $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$
 1: $\boldsymbol{V} := [\boldsymbol{v}_1 \ \boldsymbol{v}_2 \ \cdots \ \boldsymbol{v}_n] := \boldsymbol{I}_{n,n}$
 2: repeat
 3:   $maxt := 0$
 4:   $pairs :=$ jacobi_pairs()
 5:   for $(j, k)$ in $pairs$ do
 6:     $x := \boldsymbol{a}_j^\top \boldsymbol{a}_j$
 7:     $y := \boldsymbol{a}_k^\top \boldsymbol{a}_k$
 8:     $g := \boldsymbol{a}_j^\top \boldsymbol{a}_k$
 9:     $\tau := |g| / \sqrt{x \times y}$
10:     $maxt := \max(maxt, \tau)$
11:     if $\tau > tol$ then
12:       Computation of Jacobi rotation
13:       $f := (x - y)/2$
14:       $t := g / \left(f + \mathrm{sign}\left(\sqrt{g^2 + f^2}, f\right)\right)$
15:       $r := \sqrt{1 + t^2}$
16:       $\cos(\theta) := 1/r$
17:       $\sin(\theta) := t/r$
18:       Effect of Jacobi rotation
19:       $\boldsymbol{q} := \boldsymbol{a}_j$
20:       $\boldsymbol{a}_j := \cos(\theta)\,\boldsymbol{q} + \sin(\theta)\,\boldsymbol{a}_k$
21:       $\boldsymbol{a}_k := -\sin(\theta)\,\boldsymbol{q} + \cos(\theta)\,\boldsymbol{a}_k$
22:       Effect of Jacobi rotation
23:       $\boldsymbol{q} := \boldsymbol{v}_j$
24:       $\boldsymbol{v}_j := \cos(\theta)\,\boldsymbol{q} + \sin(\theta)\,\boldsymbol{v}_k$
25:       $\boldsymbol{v}_k := -\sin(\theta)\,\boldsymbol{q} + \cos(\theta)\,\boldsymbol{v}_k$
26:     end if
27:   end for
28: until $maxt > tol$
29: for $j = 1$ to $n$ do
30:   $\sigma_j := \|\boldsymbol{a}_j\|_2$
31: end for
32: $\boldsymbol{\Sigma} := \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n)$
33: $\boldsymbol{U} := \boldsymbol{A}\boldsymbol{\Sigma}^{-1}$
```

$$pairs = (1, 2), (1, 3), \cdots, (1, n), (2, 3), (2, 4), \cdots, (2, n), \cdots,$$
$$(n-2, n-1), (n-2, n), (n-1, n). \tag{6}$$

In `xGESVJ` implemented in LAPACK, the De Rijk method [13] is adopted. In the subroutine `jacobi_pairs` of Algorithm 1, the construction of *pair* is arbitrary and can be freely designed. The De Rijk method suggests an appropriate way to construct *pair*. As the subroutine `jacobi_pairs`, the proposed implementation also swaps $\boldsymbol{a}_j$ with the longest one of $\boldsymbol{a}_k (k = i, \cdots, n)$. Thus, one can save computation cost and increase speed.

4. Givens Rotation with High Accuracy

Jacobi rotation in the Jacobi method involves the same computation as in Givens rotation. Givens rotation is performed using vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ as follows:

$$\boldsymbol{x} \leftarrow \cos(\theta)\boldsymbol{x} + \sin(\theta)\boldsymbol{y}, \tag{7}$$

$$\boldsymbol{y} \leftarrow -\sin(\theta)\boldsymbol{x} + \cos(\theta)\boldsymbol{y}. \tag{8}$$

For higher accuracy, Eqs. (7) and (8) are converted to adopt the fused multiply-accumulate. Additionally, $\cos(\theta)$ or $\sin(\theta)$ is corrected.

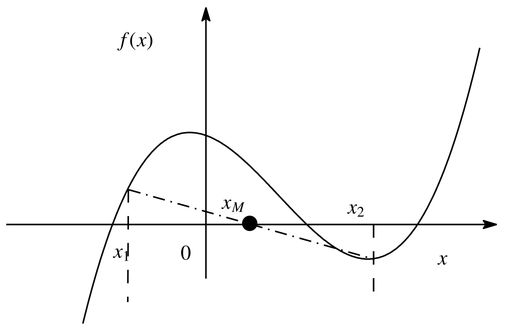


Fig. 1 False-position method.

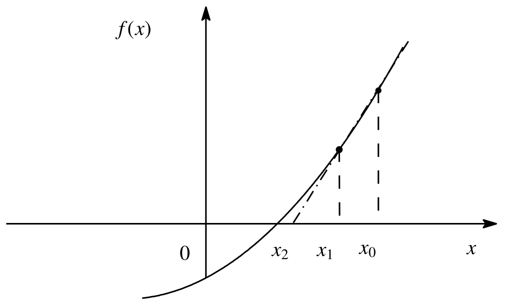


Fig. 2 Secant method.

4.1 False-position Method

We consider a real root in $f(x) = 0$. The false-position method is depicted in **Fig. 1**. In the initial setting, x_1 and x_2 have different values. The sign of $f(x_1)$ is set to be different from that of $f(x_2)$.

In the false-position method, x_M in Eq. (9) is set to a new position to compute the real root x in $f(x) = 0$. Accordingly, x_M is expressed as follows:

$$x_M = \frac{x_1 \times f(x_2) - x_2 \times f(x_1)}{f(x_2) - f(x_1)}. \tag{9}$$

Here, if the sign of $f(x_1)$ is the same as that of $f(x_M)$, then $x_1 \leftarrow x_M$. Alternatively, if the sign of $f(x_2)$ is the same as that of $f(x_M)$, then $x_2 \leftarrow x_M$. As depicted in Fig. 1, x_M is set to a new x_1.

4.2 Secant Method

The secant method is depicted in **Fig. 2**. In this method, the following recurrence relation is adopted to compute the real root x in $f(x) = 0$:

$$\begin{aligned} x_{n+1} &= x_n - f(x_n) \times \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \\ &= \frac{x_{n-1} f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}. \end{aligned} \tag{10}$$

From the initial setting x_0 and x_1, the sequence of $x_2, x_3, \cdots$ converges to the real root x, as the point sequence is computed in order. When $n = 2$ in Eq. (10), we obtain Eq. (9).

4.3 Correction of cos(θ)

Computationally, $\sin(\theta)$ and $\cos(\theta)$ are affected by rounding errors. When θ is close to 0, $\sin(\theta)$ is sufficiently accurate; however, $\cos(\theta)$ often contains computation errors. In these cases, $\cos(\theta)$ is close to 1. To obtain a correct $\cos(\theta)$, we consider the following:

$$f(x) = (x)^2 + (\sin(\theta))^2 - 1 = 0. \tag{11}$$

Because $\cos(\theta)$ is close to 1, x in Eq. (11) is computed using the false-position or scent method via $x_0 = 1, x_1 = \cos(\theta)$ as follows:

$$\begin{aligned} x_2 &= \frac{f(\cos(\theta)) - \cos(\theta) \times f(1)}{f(\cos(\theta)) - f(1)} \\ &= 1 - \sin(\theta) \times \frac{\sin(\theta)}{1 + \cos(\theta)} \end{aligned} \tag{12}$$

4.4 Givens Rotation Using Fused Multiply-accumulate

To adopt the fused multiply-accumulate, z_1 is set to

$$z_1 \leftarrow \frac{\sin(\theta)}{1 + \cos(\theta)}. \tag{13}$$

Accordingly, Eqs. (7) and (8) are transformed using z_1 as follows [14]:

$$\boldsymbol{x} \leftarrow \underline{\underline{\sin(\theta)\left(\underline{\underline{-z_1\boldsymbol{x} + \boldsymbol{y}}}\right) + \boldsymbol{x}}}, \tag{14}$$

$$\boldsymbol{y} \leftarrow \underline{\underline{-\sin(\theta)\left(\underline{\underline{z_1\boldsymbol{y} + \boldsymbol{x}}}\right) + \boldsymbol{y}}}. \tag{15}$$

The fused multiply-accumulate can be adopted in the double underlined part of these equations.

5. Proposed Implementation

5.1 Proposed Implementation

Six improvements have been made in the proposed implementation.

The first improvement is to adopt Givens rotation with high accuracy.

The second improvement is to avoid fatal bugs that could create an infinite loop. In the implementations using the Givens rotation, rounding errors may occur during orthogonalization while computing the *pair* of two vectors $(\boldsymbol{a}_j, \boldsymbol{a}_k)$. Therefore, when there is no effect of improving orthogonality in the *pair* of two vectors $(\boldsymbol{a}_j, \boldsymbol{a}_k)$, even upon performing orthogonalization, an infinite loop occurs. To avoid this infinite loop, the orthogonalization computation should be terminated when the orthogonality in all the *pair* becomes invariant. In other words, after the orthogonalization for all *pairs* of two vectors $(\boldsymbol{a}_j, \boldsymbol{a}_k)$, $\hat{g}$ of all *pairs* of two vectors $(\boldsymbol{a}_j, \boldsymbol{a}_k)$ cannot satisfy $\hat{g} \le tol \cdot \delta_j \cdot s_k$. Here, $\hat{g}$, δ_j, and s_k are described in Section 5.2.

As the third improvement, the orthogonalization computation should be minimized. In Algorithm 1, the same computation is performed for all the vectors including those that possess sufficient orthogonality. In other words, if $\hat{g}$ of the *pair* of two vector $(\boldsymbol{a}_j, \boldsymbol{a}_k)$ satisfies $\hat{g} \le tol \cdot \delta_j \cdot s_k$, then we regard that two vector $(\boldsymbol{a}_j, \boldsymbol{a}_k)$ possess sufficient orthogonality. Here, $\hat{g}$, δ_j, and s_k are described in Section 5.2. In this case, it is redundant to orthogonalize vectors that previously have sufficient orthogonality. Furthermore, the effect of rounding errors increases by repeating unnecessary orthogonalization computation. Therefore, vector $\boldsymbol{a}_j$, which is orthogonal to all the other vectors in the previous iteration, is excluded from the orthogonalization computation. This exclusion can increase accuracy and speed.

The fourth improvement is to avoid overflow and underflow. Computationally, Algorithm 1 includes the possibility of overflow and underflow while computing x and y. In the proposed implementation, lines 6 and 7 in Algorithm 1 are transformed into an equation using the norm of the vector.

The fifth improvement is also to avoid overflow and underflow. Computationally, Algorithm 1 includes the possibility of overflow and underflow while computing g. To avoid this, $\hat{g}$ and $\boldsymbol{w}_j = \boldsymbol{a}_j \times t_j$ is introduced, where t_j is define as follows:

$$\rho_j \equiv \begin{cases} \text{SAFMIN} & \sqrt{\hat{\beta}_{j,j}} < \text{SAFMIN} \\ \sqrt{\hat{\beta}_{j,j}} & \text{otherwise} \end{cases}$$
$$t_j \equiv \frac{1}{\rho_j}$$
$$\boldsymbol{w}_j \equiv \boldsymbol{a}_j \times t_j$$

Here, $\sqrt{\hat{\beta}_{j,j}}$, which is described in Section 5.3.1, is the estimated norm of $\boldsymbol{a}_j$. Thus, the norm $\boldsymbol{w}_j$ is almost equal to 1. Hereafter, SAFMIN means such a safe minimum value that 1/SAFMIN does not overflow. To be more precise, we set SAFMIN as follows,

$$\text{SAFMIN} = 1.17549435 \times 10^{-38} \quad \text{in single precision,}$$
$$\text{SAFMIN} = 2.22507386 \times 10^{-308} \quad \text{in double precision.}$$

Using the upper $\boldsymbol{w}_j$, we can compute $\hat{g} = \left(\boldsymbol{a}_j^\top \boldsymbol{a}_k\right) \times t_j = \boldsymbol{w}_j^\top \boldsymbol{a}_k$, which is adopted in Section 5.2.

The sixth improvement is to compute an accurate norm of a vector under the condition that the approximation of norm is known. That is described in Section 5.3.2.

5.2 Computation with High Accuracy in cos(θ) and sin(θ)

Jacobi rotation matrix makes the off-diagonal components zero as follows:

$$J_1 \begin{pmatrix} \boldsymbol{a}_j^\top \boldsymbol{a}_j & \boldsymbol{a}_j^\top \boldsymbol{a}_k \\ \boldsymbol{a}_j^\top \boldsymbol{a}_k & \boldsymbol{a}_k^\top \boldsymbol{a}_k \end{pmatrix} J_2 = \begin{pmatrix} \hat{\beta}_{j,j} & 0 \\ 0 & \hat{\beta}_{k,k} \end{pmatrix}, \tag{16}$$

where,

$$J_1 = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}, \tag{17}$$
$$J_2 = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}. \tag{18}$$

Generally, cos(θ) and sin(θ) in Eqs. (17) and (18) are computed as follows [*1]:

$$x = \boldsymbol{a}_j^\top \boldsymbol{a}_j, \tag{19}$$
$$y = \boldsymbol{a}_k^\top \boldsymbol{a}_k, \tag{20}$$
$$f = \frac{1}{2}(x - y), \tag{21}$$
$$g = \boldsymbol{a}_j^\top \boldsymbol{a}_k, \tag{22}$$
$$t = \frac{g}{\left(f + \text{sign}\left(\sqrt{g^2 + f^2}, f\right)\right)}, \tag{23}$$
$$r = \sqrt{1 + t^2}, \tag{24}$$
$$\cos(\theta) = \frac{1}{r}, \tag{25}$$
$$\sin(\theta) = \frac{t}{r}, \tag{26}$$
$$\hat{\beta}_{j,j} = \boldsymbol{a}_j^\top \boldsymbol{a}_j + t \times \boldsymbol{a}_j^\top \boldsymbol{a}_k, \tag{27}$$
$$\hat{\beta}_{k,k} = \boldsymbol{a}_k^\top \boldsymbol{a}_k - t \times \boldsymbol{a}_j^\top \boldsymbol{a}_k. \tag{28}$$

However, in the proposed implementation, instead of f and g, we use $\hat{f}$ and $\hat{g}$ to avoid divergence. The following equations are adopted to compute $\hat{f}$ and $\hat{g}$:

$$\delta_j = s_j \times t_j, \tag{29}$$
$$\delta_k = s_k \times t_j, \tag{30}$$
$$\hat{f} = \frac{1}{2}\left(\sqrt{\boldsymbol{a}_j^\top \boldsymbol{a}_j} - \sqrt{\boldsymbol{a}_k^\top \boldsymbol{a}_k}\right) \times \left(\sqrt{\boldsymbol{a}_j^\top \boldsymbol{a}_j} \times t_j + \sqrt{\boldsymbol{a}_k^\top \boldsymbol{a}_k} \times t_j\right), \tag{31}$$
$$= \frac{1}{2}\left(s_j - s_k\right)\left(\delta_j + \delta_k\right), \tag{32}$$
$$\hat{g} = \left(\boldsymbol{a}_j^\top \boldsymbol{a}_k\right) \times t_j = \boldsymbol{w}_j^\top \boldsymbol{a}_k, \tag{33}$$
$$t = \frac{\hat{g}}{\left(\hat{f} + \text{sign}\left(\sqrt{\hat{g}^2 + \hat{f}^2}, \hat{f}\right)\right)}, \tag{34}$$
$$r = \sqrt{\underline{\underline{1 + t^2}}}, \tag{35}$$
$$\cos(\theta) = \frac{1}{r}, \tag{36}$$
$$\sin(\theta) = \frac{t}{r}, \tag{37}$$
$$\hat{\beta}_{j,j} = s_j^2 + t \times \hat{g}/t_j, \tag{38}$$
$$\hat{\beta}_{k,k} = s_k^2 - t \times \hat{g}/t_j, \tag{39}$$

where s_j and s_k denote the norm of $\boldsymbol{a}_j$ and the norm of $\boldsymbol{a}_k$, respectively. As a remark, δ_j is almost 1. δ_k satisfies $\delta_k \leq 1$. Hence $\hat{f} \leq s_j - s_k$, where it is guaranteed by the De Rijk method that $s_j \geq s_k$. Thus, the overflow of $\hat{f}$ can be avoided by using the above computation. The fused multiply-accumulate can be adopted in the double underlined part of these equations. For computing $\sqrt{\hat{g}^2 + \hat{f}^2}$, we employ `xLARTG` implemented in LAPACK.

5.3 Computation of the Norm of Vectors

5.3.1 Approximate Computation of the Norm of Vectors

$\sqrt{\hat{\beta}_{j,j}}$ and $\sqrt{\hat{\beta}_{k,k}}$ represent the approximated norm of $\hat{\boldsymbol{a}}_j$ and the approximated norm of $\hat{\boldsymbol{a}}_k$, respectively.

$$\sqrt{\hat{\beta}_{j,j}} \approx \|\hat{\boldsymbol{a}}_j\|_2, \tag{40}$$
$$\sqrt{\hat{\beta}_{k,k}} \approx \|\hat{\boldsymbol{a}}_k\|_2, \tag{41}$$

where

$$\hat{\boldsymbol{a}}_j = \cos(\theta)\boldsymbol{a}_j + \sin(\theta)\boldsymbol{a}_k, \tag{42}$$
$$\hat{\boldsymbol{a}}_k = -\sin(\theta)\boldsymbol{a}_j + \cos(\theta)\boldsymbol{a}_k. \tag{43}$$

$\sqrt{\hat{\beta}_{j,j}}$ and $\sqrt{\hat{\beta}_{k,k}}$ are computed as follows,

$$\sqrt{\hat{\beta}_{j,j}} = \sqrt{s_j^2 + t \times \left(\hat{g}/t_j\right)}, \tag{44}$$
$$= s_j \times \sqrt{\underline{\underline{1 + t \times \left(\hat{g}/\left(s_j \times \delta_j\right)\right)}}}, \tag{45}$$
$$\sqrt{\hat{\beta}_{k,k}} = \sqrt{s_k^2 - t \times \left(\hat{g}/t_j\right)}, \tag{46}$$

[*1] The proof for the Eqs. (19)–(28) is shown in an appendix.

$$= s_k \times \sqrt{\underline{\underline{1 - t \times (\hat{g}/(s_k \times \delta_k))}}}, \tag{47}$$

where s_j and s_k denote the norm of $\boldsymbol{a}_j$ and the norm of $\boldsymbol{a}_k$, respectively. The fused multiply-accumulate can be adopted in the double underlined part of these equations. If $\underline{\underline{1 - t \times (\hat{g}/(s_k \times \delta_k))}}$ is not positive, we must use xNRM2 implemented in LAPACK for computing the norm of the vector $\boldsymbol{a}_k$.

5.3.2 Accurate Computation of the Norm of Vectors

Let $\boldsymbol{x}$ be the vector whose norm you compute.

$$\boldsymbol{x} = (x_1, x_2, \dots, x_m). \tag{48}$$

Under the condition that the approximation of the norm α is known, the underflow and overflow can be avoided by using the following computation.

$$\|\boldsymbol{x}\|_2 = \alpha \times \sqrt{\left(\frac{x_1}{\alpha}\right)^2 + \left(\frac{x_2}{\alpha}\right)^2 + \cdots + \left(\frac{x_m}{\alpha}\right)^2} \tag{49}$$

Concretely, the implementation is as follows.

$$\beta \equiv \begin{cases} \text{SAFMIN} & \alpha < \text{SAFMIN} \\ \alpha & \text{otherwise} \end{cases}, \tag{50}$$

$$\gamma \equiv \frac{1}{\beta}, \tag{51}$$

$$\ell_0 = 0, \tag{52}$$

$$\ell_1 = \underline{\underline{\ell_0 + (\gamma x_1)^2}}, \tag{53}$$

$$\ell_2 = \underline{\underline{\ell_1 + (\gamma x_2)^2}}, \cdots, \tag{54}$$

$$\ell_m = \underline{\underline{\ell_{m-1} + (\gamma x_m)^2}}, \tag{55}$$

$$\|\boldsymbol{x}\|_2 = \beta \times \sqrt{\ell_m}. \tag{56}$$

The fused multiply-accumulate can be adopted in the double underlined part of these equations.

6. Experiments

We evaluated the proposed implementation to see if it had higher accuracy than those of the QR method, OQDS method in Ref. [1], and xGESVJ implemented in LAPACK-3.9.0, which is a routine for the one-sided Jacobi method. Because the test matrices are upper triangular, the Householder transformation is adopted as a preprocessing method for the QR and OQDS methods.

6.1 Environment

Table 1 summarizes the experimental environment. We use four real random upper triangular matrices, whose dimensions are 500×500, $1{,}000 \times 1{,}000$, $1{,}500 \times 1{,}500$, and $2{,}000 \times 2{,}000$, respectively. The following Frobenius norms are used to evaluate the computation errors:

$$\|\boldsymbol{A} - \boldsymbol{U\Sigma V}^\top\|_F, \tag{57}$$

$$\|\boldsymbol{U}^\top \boldsymbol{U} - \boldsymbol{I}\|_F, \tag{58}$$

$$\|\boldsymbol{V}^\top \boldsymbol{V} - \boldsymbol{I}\|_F. \tag{59}$$

In the experiments, *tol* for the Jacobi method is set to $tol = \sqrt{m}\varepsilon$, where ε denotes a machine epsilon. This setting is adopted in xGESVJ implemented in LAPACK. In our implementation, if $\hat{g} <$ SAFMIN, then $tol = \sqrt{m}\varepsilon$ the otherwise $tol = \varepsilon$. From the line 11 in the Algorithm 1, we obtain

$$\boldsymbol{a}_j^\top \boldsymbol{a}_k > tol \cdot \|\boldsymbol{a}_j\| \cdot \|\boldsymbol{a}_k\|,$$
$$g > tol \cdot s_j \cdot s_k,$$
$$t_j g > tol \cdot t_j \cdot s_j \cdot s_k,$$
$$\hat{g} > tol \cdot \delta_j \cdot s_k.$$

Thus, instead of the line 11 in the Algorithm 1, we adopt $\hat{g} > tol \cdot \delta_j \cdot s_k$.

Table 1 Experimental environment.

CPU	Intel(R) Core(TM) i7-9700 CPU 3.00 GHz
OS	Ubuntu 20.04.1 LTS
RAM	16 GB
Cache	12 MB Intel(R) Smart Cache
Compiler	gfortran 9.3.0
Options	-O3 -mtune=native -march=native
Software	Lapack-3.9.0

Table 2 Number of iterations of xGESVJ and our method in single precision.

dimension size	one-sided Jacobi	Proposal
500	11	13
1,000	13	16
1,500	15	16
2,000	15	19

Table 3 Number of iterations of xGESVJ and our method in double precision.

dimension size	one-sided Jacobi	Proposal
500	14	16
1,000	18	17
1,500	20	21
2,000	22	24

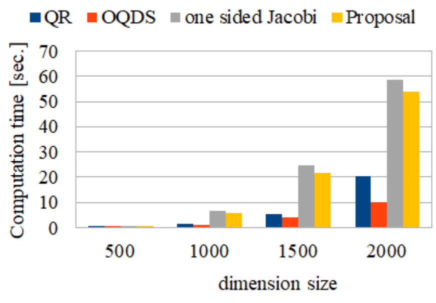
6.2 Result and Consideration

Table 2 and **Table 3** show iteration number of xGESVJ and our method in single precision and double precision, respectively.

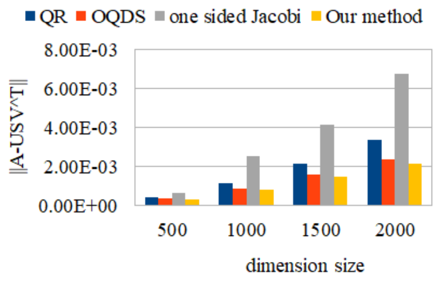
Figure 3 depicts the performance results in single precision. **Figure 4** depicts the performance results in double precision. Figure 4 has the same tendency as Fig. 3.

Table 4 and **Table 5** show computation time in single precision and double precision, respectively. From Fig. 3 (a), Fig. 4 (a), Table 4 and Table 5, it is evident that the proposed implementation is faster than xGESVJ. Computation cost in the Givens rotation with high accuracy is higher than that in the Jacobi rotation in xGESVJ. However, since we reduced redundant computation to speed up the operation, we can archive the speed up of the proposed implementation.

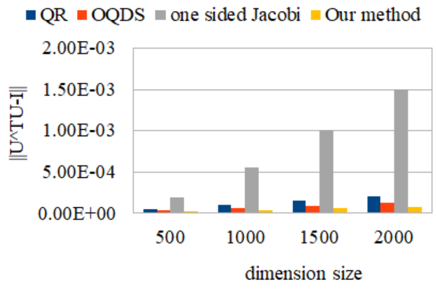
Table 6 and **Table 7** show the comparison of $\|\boldsymbol{A} - \boldsymbol{U\Sigma V}^\top\|_F$ in single precision and double precision, respectively. From Fig. 3 (b) and Table 6, it is evident that $\|\boldsymbol{A} - \boldsymbol{U\Sigma V}^\top\|_F$ in the proposed implementation is the highest among all the implementations. On the other hand, in Fig. 4 (b) and Table 7, the $\|\boldsymbol{A} - \boldsymbol{U\Sigma V}^\top\|_F$ in the proposed implementation is better than that in the QR method and xGESVJ. It is caused that the QR and OQDS methods are affected by rounding errors due to preprocessing via the Householder transformation; the Givens rotation



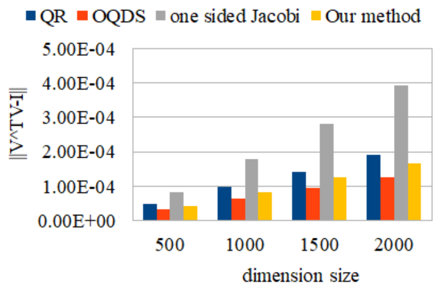
(a) Computation time (s)



(b) $\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}\|_F$

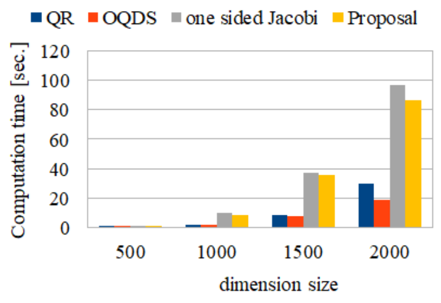


(c) $\|\boldsymbol{U}^{\top}\boldsymbol{U} - \boldsymbol{I}\|_F$

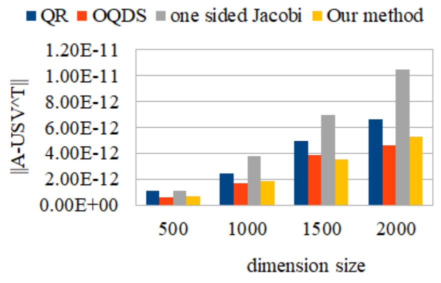


(d) $\|\boldsymbol{V}^{\top}\boldsymbol{V} - \boldsymbol{I}\|_F$

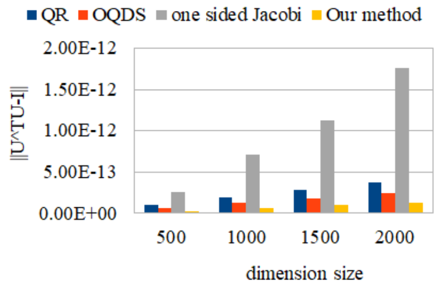
Fig. 3 Comparison in single precision.



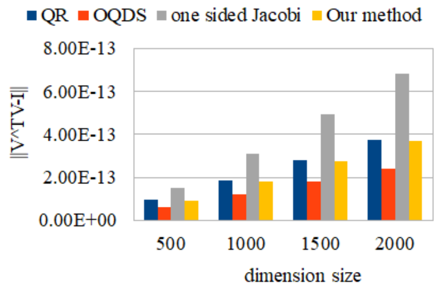
(a) Computation time (s)



(b) $\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}\|_F$



(c) $\|\boldsymbol{U}^{\top}\boldsymbol{U} - \boldsymbol{I}\|_F$



(d) $\|\boldsymbol{V}^{\top}\boldsymbol{V} - \boldsymbol{I}\|_F$

Fig. 4 Comparison in double precision.

of the proposed implementation with high accuracy is better than that of `xGESVJ`.

Table 8 and **Table 9** show the comparison of $\|\boldsymbol{U}^{\top}\boldsymbol{U} - \boldsymbol{I}\|_F$ in single precision and double precision, respectively. From Fig. 3 (c), Fig. 4 (c), Table 8 and Table 9, it is evident that $\|\boldsymbol{U}^{\top}\boldsymbol{U} - \boldsymbol{I}\|_F$ in the proposed implementation is the highest among all the implementations. It is caused for the same reason as Fig. 3 (b) and Table 6. Especially, from Fig. 3 (c), we can see that the orthogonality of left singular vectors in the proposed implementation is approximately 10 times more accurate than that in `xGESVJ`.

Table 10 and **Table 11** show the comparison of $\|\boldsymbol{V}^{\top}\boldsymbol{V} - \boldsymbol{I}\|_F$ in single precision and double precision, respectively. From Fig. 3 (d) and Fig. 4 (d), we say that $\|\boldsymbol{V}^{\top}\boldsymbol{V} - \boldsymbol{I}\|_F$ in the proposed implementation is smaller than that in the QR method and `xGESVJ`. From Table 10 and Table 11, $\|\boldsymbol{V}^{\top}\boldsymbol{V} - \boldsymbol{I}\|_F$ in the proposed implementation is slightly larger than that in the OQDS method but not considerably different.

In this study, we aimed to implement a highly accurate one-

Table 4 Comparison of computation time in single precision (sec.).

dimension size	QR	OQDS	one-sided Jacobi	Proposal
500	0.17	0.15	0.71	0.68
1,000	1.39	1.13	6.47	5.77
1,500	5.32	3.92	24.69	21.88
2,000	20.42	9.99	58.76	53.85

Table 5 Comparison of computation time in double precision (sec.).

dimension size	QR	OQDS	one-sided Jacobi	Proposal
500	0.23	0.24	1.03	0.92
1,000	2.05	2.05	9.98	8.43
1,500	8.42	7.78	37.38	35.42
2,000	30.10	18.73	96.50	86.59

Table 6 Comparison of $\|\boldsymbol{A}-\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\|_F$ in single precision (10^{-5}).

dimension size	QR	OQDS	one-sided Jacobi	Proposal
500	43.30	34.17	63.29	28.13
1,000	113.15	86.04	253.96	77.41
1,500	216.14	155.80	411.77	146.04
2,000	336.20	236.57	677.04	216.10

Table 7 Comparison of $\|\boldsymbol{A}-\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\|_F$ in double precision (10^{-13}).

dimension size	QR	OQDS	one-sided Jacobi	Proposal
500	11.14	6.22	11.16	6.82
1,000	24.85	17.22	38.20	18.69
1,500	49.38	38.72	69.90	35.76
2,000	66.20	46.40	104.53	52.45

Table 8 Comparison of $\|\boldsymbol{U}^\top\boldsymbol{U}-\boldsymbol{I}\|_F$ in single precision (10^{-5}).

dimension size	QR	OQDS	one-sided Jacobi	Proposal
500	5.11	3.17	19.13	1.91
1,000	9.98	6.16	55.07	3.79
1,500	15.21	9.28	100.45	5.77
2,000	20.55	12.29	149.97	7.65

Table 9 Comparison of $\|\boldsymbol{U}^\top\boldsymbol{U}-\boldsymbol{I}\|_F$ in double precision (10^{-13}).

dimension size	QR	OQDS	one-sided Jacobi	Proposal
500	0.99	0.62	2.53	0.30
1,000	1.90	1.22	7.11	0.61
1,500	2.88	1.82	11.23	0.97
2,000	3.78	2.42	17.63	1.29

Table 10 Comparison of $\|\boldsymbol{V}^\top\boldsymbol{V}-\boldsymbol{I}\|_F$ in single precision (10^{-5}).

dimension size	QR	OQDS	one-sided Jacobi	Proposal
500	4.86	3.12	8.20	4.14
1,000	9.73	6.29	17.83	8.34
1,500	14.16	9.48	28.11	12.46
2,000	19.09	12.61	39.14	16.75

Table 11 Comparison of $\|\boldsymbol{V}^\top\boldsymbol{V}-\boldsymbol{I}\|_F$ in double precision (10^{-13}).

dimension size	QR	OQDS	one-sided Jacobi	Proposal
500	0.97	0.61	1.49	0.90
1,000	1.87	1.21	3.13	1.82
1,500	2.83	1.81	4.97	2.78
2,000	3.73	2.43	6.83	3.71

sided Jacobi method. Experimental results confirm that although the computation time of the proposed implementation is smaller than that of the conventional implementation, its accuracy is significantly higher. In the Sakurai–Sugiura method, only left singular vectors $\boldsymbol{U}$ are required. Therefore, the orthogonality of right singular vector $\boldsymbol{V}$ and the accuracy of singular value decomposition $\|\boldsymbol{A}-\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\|_F$ are not relevant. Furthermore, $\|\boldsymbol{V}^\top\boldsymbol{V}-\boldsymbol{I}\|_F$ and $\|\boldsymbol{A}-\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\|_F$ in the proposed implementation are comparable to that in the OQDS method. Hence, the proposed implementation of the one-sided Jacobi method with high accuracy is appropriate for the Sakurai–Sugiura method. In other words, a column space, which is required of the Sakurai–Sugiura method, can be computed higher accuracy with the proposed implementation.

7. Conclusion

We proposed a novel implementation of the one-sided Jacobi method with high accuracy. In the implementation, a Givens rotation with high accuracy and the fused multiply-accumulate were adopted. Notably, overflow and underflow may occur in the conventional method [4]. To avoid this problem, we have carefully improved the one-sided Jacobi method. Because the Givens rotation with high accuracy requires considerable computation time, we reduced redundant computation to speed up the operation. The experimental result confirmed that the proposed implementation is faster than the routine in LAPACK. Furthermore, the orthogonality of the left singular vectors in the proposed implementation was the best among all the implementations. In the Sakurai–Sugiura method, only left singular vectors are required. Consequently, the proposed implementation realized the one-sided Jacobi method with high accuracy. Hence, our proposed method might be applied to the Sakurai–Sugiura method.

As a future work, we wish to further speedup the proposed method.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number JP17H02858 and JP17K00167.

References

[1] Araki, S.,Tanaka, H., Takata, M., Kimura, K. and Nakamura, Y.: Fast Computation Method of Column Space by Using the DQDS Method and the OQDS Method, *Proc. PDPTA 2018*, pp.333–339 (2018).
[2] Brent, R.P., Luk, F.T. and van Loan, C.: Computation of the singular value decomposition using mesh-connected processors, *J. VLSI Comput. Syst.*, Vol.1, pp.242–270 (1985).
[3] Demmel, J.: Applied Numerical Linear Algebra, *SIAM*, Philadelphia (1997).
[4] Demmel, J. and Veselic, K.: Jacobi's method is more accurate than QR, *SIAM J. Matrix Anal. Appl.*, Vol.13, No.4, pp.1204–1245 (1992).
[5] Drmac, J. and Veselic, K.: New fast and accurate Jacobi SVD algorithm: I., *SIAM J. Matrix Anal. Appl.*, Vol.29, pp.1322–1342 (2008).
[6] Drmac, Z. and Veselic, K.: New fast and accurate Jacobi SVD algorithm: II., *SIAM J. Matrix Anal. Appl.*, Vol.29, pp.1343–1362 (2008).
[7] Fernando, K.V. and Parlett, B.N.: Accurate singular values and differential qd algorithms, *Numer. Math.*, Vol.67, pp.191–229 (1994).
[8] Forsythe, G.E. and Henrici, P.: The cyclic Jacobi method for computing the principal values of a complex matrix, *Trans. Amer. Math. Soc.*, Vol.94, pp.1–23 (1960).
[9] Kogbetliantz, E.: Solution of linear equations by diagonalization of coefficients matrix, *Quarterly of Applied Mathematics*, Vol.13, pp.123–132 (1955).
[10] Linear Algebra PACKage, available from ⟨http://www.netlib.org/lapack/⟩ (accessed 2020-07-29).
[11] von Matt, U.: The orthogonal qd-algorithm, *SIAM J. Sci. Comput.*, Vol.18, pp.1163–1186 (1997).
[12] Parlett, B.N. and Marques, O.A.: *An implementation of the dqds algorithm* (*positive case*), *Lin. Alg. Appl*, Vol.309, No.1–3, pp.217–259 (2000).
[13] De Rijk, P.P.M.: A one-sided Jacobi algorithm for computing the singular value decomposition on a vector computer, *SIAM J. Sci. Stat. Comp.*, Vol.10, No.2, pp.359–371 (1998).
[14] Rutishauser, H.: The Jacobi method for real symmetric matrices, *Numerische Mathematik*, Vol.9, No.1, pp.1–10 (1966).
[15] Sakurai, T. and Tadano, H.: CIRR: A Rayleigh-Ritz type method with counter integral for generalized eigenvalue problems, *Hokkaido Math. J.*, Vol.36, pp.745–757 (2007).

Appendix

The proof for the Eqs. (19)–(28) is shown in this appendix. From,

$$J_1\begin{pmatrix} \boldsymbol{a}_j^\top\boldsymbol{a}_j & \boldsymbol{a}_j^\top\boldsymbol{a}_k \\ \boldsymbol{a}_j^\top\boldsymbol{a}_k & \boldsymbol{a}_k^\top\boldsymbol{a}_k \end{pmatrix}J_2 = \begin{pmatrix} \hat{\beta}_{j,j} & 0 \\ 0 & \hat{\beta}_{k,k} \end{pmatrix},$$

where,

$$J_1 = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix},$$

$$J_2 = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix},$$

we obtain,

$$\cos^2(\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_j + \sin(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k + \sin^2(\theta)\boldsymbol{a}_k^\top \boldsymbol{a}_k = \hat{\beta}_{j,j}, \qquad \text{(A.1)}$$

$$\cos(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k - \frac{\sin(2\theta)}{2}\left(\boldsymbol{a}_j^\top \boldsymbol{a}_j - \boldsymbol{a}_k^\top \boldsymbol{a}_k\right) = 0, \qquad \text{(A.2)}$$

$$\sin^2(\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_j - \sin(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k + \cos^2(\theta)\boldsymbol{a}_k^\top \boldsymbol{a}_k = \hat{\beta}_{k,k}. \qquad \text{(A.3)}$$

From Eq. (A.2),

$$\begin{aligned}
\tan(2\theta) &= \frac{2\boldsymbol{a}_j^\top \boldsymbol{a}_k}{\boldsymbol{a}_j^\top \boldsymbol{a}_j - \boldsymbol{a}_k^\top \boldsymbol{a}_k} = \frac{\eta}{\zeta}, \\
2\theta &= \tan^{-1}\left(\frac{\eta}{\zeta}\right), \\
\sin(2\theta) &= \sin\left(\tan^{-1}\left(\frac{\eta}{\zeta}\right)\right) = \frac{\eta}{\zeta\sqrt{1+\left(\frac{\eta}{\zeta}\right)^2}}, \\
\cos(2\theta) &= \cos\left(\tan^{-1}\left(\frac{\eta}{\zeta}\right)\right) = \frac{1}{\sqrt{1+\left(\frac{\eta}{\zeta}\right)^2}}.
\end{aligned}$$

From,

$$\tan(\theta) = \frac{\sin(2\theta)}{1+\cos(2\theta)},$$

the following equations are satisfied,

$$\begin{aligned}
\tan(\theta) &= \frac{\eta}{\zeta + \zeta\sqrt{1+\left(\frac{\eta}{\zeta}\right)^2}}, \\
&= \frac{\eta}{\zeta + \operatorname{sign}\left(\sqrt{\zeta^2+\eta^2}, \zeta\right)}, \\
\cos(\theta) &= \frac{1}{\sqrt{1+\tan^2(\theta)}}, \\
\sin(\theta) &= \frac{\tan(\theta)}{\sqrt{1+\tan^2(\theta)}},
\end{aligned}$$

where $\operatorname{sign}(a,b)$ means $|a| \times \operatorname{sign}(b)$. If $\tan(2\theta) = 0$, two vectors $(\boldsymbol{a}_j, \boldsymbol{a}_k)$ are orthogonal. In addition, θ is restricted by $|\theta| \le \frac{\pi}{4}$. Thus, we can assume $\tan(2\theta) \ne 0$ and $\cos(\theta) \ne 0$. From,

$$\begin{aligned}
\cos(\theta) - \frac{\sin(\theta)}{\tan(2\theta)} &= \cos(\theta) - \sin(\theta)\left(\frac{1-\tan^2(\theta)}{2\tan(\theta)}\right), \\
&= \cos(\theta)\left(\frac{1+\tan^2(\theta)}{2}\right), \\
&= \frac{1}{2\cos(\theta)},
\end{aligned}$$

Equation (A.1), and Eq. (A.3), we obtain,

$$\begin{aligned}
\hat{\beta}_{j,j} &= \cos^2(\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_j + \sin(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k + \sin^2(\theta)\boldsymbol{a}_k^\top \boldsymbol{a}_k, \\
&= \boldsymbol{a}_j^\top \boldsymbol{a}_j + \sin(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k + \sin^2(\theta)\left(\boldsymbol{a}_k^\top \boldsymbol{a}_k - \boldsymbol{a}_j^\top \boldsymbol{a}_j\right), \\
&= \boldsymbol{a}_j^\top \boldsymbol{a}_j + \sin(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k - 2\left(\frac{\sin^2(\theta)}{\tan(2\theta)}\right)\boldsymbol{a}_j^\top \boldsymbol{a}_k, \\
&= \boldsymbol{a}_j^\top \boldsymbol{a}_j + 2\sin(\theta)\left(\cos(\theta) - \frac{\sin(\theta)}{\tan(2\theta)}\right)\boldsymbol{a}_j^\top \boldsymbol{a}_k, \\
&= \boldsymbol{a}_j^\top \boldsymbol{a}_j + \tan(\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k, \\
\hat{\beta}_{k,k} &= \sin^2(\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_j - \sin(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k + \cos^2(\theta)\boldsymbol{a}_k^\top \boldsymbol{a}_k, \\
&= \boldsymbol{a}_k^\top \boldsymbol{a}_k - \sin(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k + \sin^2(\theta)\left(\boldsymbol{a}_j^\top \boldsymbol{a}_j - \boldsymbol{a}_k^\top \boldsymbol{a}_k\right), \\
&= \boldsymbol{a}_k^\top \boldsymbol{a}_k - \sin(2\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k + 2\left(\frac{\sin^2(\theta)}{\tan(2\theta)}\right)\boldsymbol{a}_j^\top \boldsymbol{a}_k, \\
&= \boldsymbol{a}_k^\top \boldsymbol{a}_k - 2\sin(\theta)\left(\cos(\theta) - \frac{\sin(\theta)}{\tan(2\theta)}\right)\boldsymbol{a}_j^\top \boldsymbol{a}_k, \\
&= \boldsymbol{a}_k^\top \boldsymbol{a}_k - \tan(\theta)\boldsymbol{a}_j^\top \boldsymbol{a}_k.
\end{aligned}$$

Masami Takata received her Ph.D. degree from Nara Women's University in 2004. She has been a lecturer of the Research Group of Information and Communication Technology for Life at Nara Women's University. Her research interests include numerical algebra and parallel algorithms.

Sho Araki received his Ph.D. degree from Kyoto University in 2021. His research interests include parallel algorithms for eigenvalue and singular value decomposition with high accuracy.

Takahiro Miyamae recieved his B.E. degree from University of Fukui in 2020. His research interests include parallel algorithms for eigenvalue and singular value decomposition.

Kinji Kimura received his Ph.D. degree from Kobe University in 2004. He became an assistant professor at Kyoto University in 2006, an assistant professor at Niigata University in 2007, a lecturer at Kyoto University in 2008, and associate professor at Fukui University in 2019.

Yoshimasa Nakamura is a vice-president and professor of Osaka Seikei University and is a professor emeritus of Kyoto University from 2021. His subject is to design new numerical algorithms for singular value decomposition by using discrete-time integrable systems. He is a member of JSIAM and MSJ.