IPSJ Transactions on Mathematical Modeling and Its Applications Vol.14 No.1 12–20 (Jan. 2021)

Regular Paper

On an Implementation of the Two-Sided Jacobi Method

Sho Araki[1,a) Masana Aoki[1] Masami Takata[2,b)] Kinji Kimura[3,c)] Yoshimasa Nakamura[1,d)]

Received: March 23, 2020, Revised: April 10, 2020,
Accepted: September 23, 2020

Abstract: The Jacobi method for performing singular value decomposition can compute all singular values and singular vectors with high accuracy. In addition, previously published studies have reported that it is more accurate than the QR algorithm. However, the computation cost of the Jacobi method is higher than that of the computation method that combines the QR method and bidiagonalization using the Householder transformation. Notably, the computation cost is insignificant for considerably small matrices. Based on the Jacobi method, one- and two-sided Jacobi methods are proposed. The one-sided method was implemented in LAPACK. However, many facets can still be improved in the implementation of the two-sided Jacobi method. Therefore, in this paper, we improve the two-sided Jacobi method. We experimentally confirmed that for small matrices, the two-sided Jacobi method had a shorter computation time and a higher accuracy than those of the one-sided Jacobi method.

Keywords: singular value decomposition, one-sided Jacobi method, two-sided Jacobi method, False position method, secant method, fused multiply-accumulate

1. Introduction

In production sites, we must detect the anomalies in factory equipment. Notably, anomaly detection does not mean to detect the anomaly occurred, but to capture the signs just before the anomaly occurs. Therefore, the singular spectrum transformation (SST) method [5], [7] that detects the changing points in the time-series data of the target system should be adopted. This method calculates the singular value decomposition of the data along the time axis and determines the number of dimensions that are sufficient for approximating the original time-series data. However, it is difficult to apply the SST method in real time using limited hardware such as an embedded computer because of the computational complexity involved. Therefore, in the current method, it is predicted that an abnormality will occur after an abnormality occurs. To perform computations in real time, singular value decompositions, which can be computed with high speed and high accuracy, must be developed. In the SST method, the time-series data can be treated by replacing it with a small matrix. Consequently, during implementation, the developed singular value decomposition should compute small matrices with high speed and high accuracy.

James Demmel and Kresimir Veselic reported that the Jacobi method was more accurate than QR [2]. Because the Jacobi method is for singular value decomposition, one- and two-sided Jacobi methods were proposed [1], [3], [4], [6], [9]. The one-sided Jacobi method was implemented in LAPACK [10]. However, no implementation of two-sided Jacobi method is provided in LAPACK. Moreover, if we do not introduce our implementation technique described in Sections 4.7, 4.8, and 5, the performance of the two-sided Jacobi method [8] is not better than that of the one-sided Jacobi method implemented in LAPACK in terms of computation time and accuracy. In addition, because the two-sided Jacobi method actively converges the off-diagonal elements to 0, the possibility of non-convergence on the computer need not be considered. Therefore, in this paper, we improve the two-sided Jacobi method. To compute the two-sided Jacobi method, we propose three implementation methods that use the arctangent function, implementation procedure by Rutishauser [11], and the Givens rotation, respectively. The experimental results confirmed that for small matrices, the two-sided Jacobi method had shorter computation time and higher accuracy than those of the one-sided Jacobi method.

In Section 2, we introduce the SST method. In Section 3, we show target matrices in the two-sided Jacobi method. In Section 4, we propose 3 implementation types for the two-sided Jacobi method. In Section 5, we explain a correction method of variables in the two-sided Jacobi method, which uses the implementation procedure by Rutishauser [11]. In Section 6, we perform experiments using 8 matrices. In order to improve the accuracy of the computation, our implementation technique in Section 4.7 and Section 5 are introduced. Our implementation technique in Section 4.8 is used to improve the computational speed.

2. Singular Spectrum Transformation

In the SST method, the singular value decomposition is performed on time-series data $(\xi_1, \xi_2, \xi_3, \cdots)$ that are extracted as follows:

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

$$\boldsymbol{x}_1 := (\xi_1, \xi_2, \cdots, \xi_M)^\top, \boldsymbol{x}_2 := (\xi_2, \xi_3, \cdots, \xi_{M+1})^\top, \cdots, \quad (1)$$

$$X_t := \left[\begin{array}{ccccc} \boldsymbol{x}_{t-n-M+1} & \boldsymbol{x}_{t-n-M+2} & \cdots & \boldsymbol{x}_{t-M-1} & \boldsymbol{x}_{t-M} \end{array}\right] = U_t^{(X)} \Sigma_t^{(X)} \left(V_t^{(X)}\right)^\top, \quad (2)$$

$$Z_t := \left[\begin{array}{ccccc} \boldsymbol{x}_{t-k-M+L+1} & \boldsymbol{x}_{t-k-M+L+2} & \cdots & \boldsymbol{x}_{t-M+L-1} & \boldsymbol{x}_{t-M+L} \end{array}\right] = U_t^{(Z)} \Sigma_t^{(Z)} \left(V_t^{(Z)}\right)^\top. \quad (3)$$

where $\Sigma_t^{(X)}$, $U_t^{(X)}$, and $V_t^{(X)}$ denote the matrices whose elements are singular values, left singular vectors, and right singular vectors, respectively, in X_t. Furthermore, $\Sigma_t^{(Z)}$, $U_t^{(Z)}$, and $V_t^{(Z)}$ denote the matrices whose elements are singular values, left singular vectors, and right singular vectors, respectively, in Z_t. In addition, $U_t^{(X,r)} = \left(\boldsymbol{u}_{1,t}^{(X)}, \cdots, \boldsymbol{u}_{r,t}^{(X)}\right)$ and $U_t^{(Z,m)} = \left(\boldsymbol{u}_{1,t}^{(Z)}, \cdots, \boldsymbol{u}_{m,t}^{(Z)}\right)$ are determined using the r left singular vectors of $U_t^{(X)} = \left(\boldsymbol{u}_{1,t}^{(X)}, \cdots, \boldsymbol{u}_{n,t}^{(X)}\right)$ and the m left singular vectors of $U_t^{(Z)} = \left(\boldsymbol{u}_{1,t}^{(Z)}, \cdots, \boldsymbol{u}_{k,t}^{(Z)}\right)$, respectively. The degree of change in two different locations is defined as $a_{(t)} = 1 - \left\|\left(U_t^{(X,r)}\right)^\top U_t^{(Z,m)}\right\|_2$. The 2 norm of a matrix means the largest singular value. Here, the arbitrarily selectable parameters are M, L, n, k, r, and m. Generally, M, n, and k should be set equal to or smaller than 100.

3. Target Matrices

The two-sided Jacobi method for eigenvalue decomposition can be used to compute the eigenvalues and eigenvectors of a real symmetric matrix. Precisely, it is also possible to extend the target matrix to a Hermitian matrix. Notably, the two-sided Jacobi method for singular value decomposition can even be designed to perform computations on complex matrices of any size. However, in this paper, we have considered only real upper triangular matrices. By preprocessing using the *QR* and *LQ* decompositions in the case of rectangular matrices, the singular value decomposition of rectangular matrices can be reduced to that of upper triangular matrices. Moreover, because we can easily extend our method to be used on an upper complex matrix, the singular value decomposition using the two-sided Jacobi method is designed for enabling computations on real upper triangular matrices.

4. Singular Value Decomposition Using the Two-sided Jacobi Method

4.1 Outline

Let $J^{(i)}$, $K^{(i)}$, $N^{(i)}$, and $M^{(i)}$ be the products of rotation matrices. Let $R^{(i)}$ and $L^{(i)}$ be real upper and lower triangular matrices, respectively. In a singular value decomposition using the two-sided Jacobi method, Eqs. (4) and (5) are repeatedly computed as follows:

$$K^{(i)} R^{(i)} J^{(i)} = L^{(i)}, \quad (4)$$

$$N^{(i)} L^{(i)} M^{(i)} = R^{(i+1)}, \quad i = 0, 1, \cdots \quad (5)$$

By these iterative computations, $R^{(i)}$ and $L^{(i)}$ converge into a diagonal matrix. In the case of convergence, the left singular vector U and right singular vector V can be computed as follows:

$$U = \left(K^{(0)}\right)^\top \left(N^{(0)}\right)^\top \left(K^{(1)}\right)^\top \left(N^{(1)}\right)^\top \cdots \left(K^{(m-1)}\right)^\top \left(N^{(m-1)}\right)^\top, \quad (6)$$

$$V = J^{(0)} M^{(0)} J^{(1)} M^{(1)} \cdots J^{(m-1)} M^{(m-1)}, \quad (7)$$

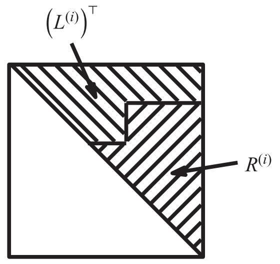

Fig. 1 Space sharing of the upper triangular matrix.

where m denotes the iteration number in the case of convergence. Here, the matrix multiplication in Eqs. (6) and (7) is accomplished using the Givens rotations. From **Fig. 1**, it is shown that $R^{(i)}$ and $L^{(i)}$ are stored together in the upper triangular matrix. In the case of Fig. 1, memory allocation is not required for $R^{(i)}$ and $L^{(i)}$ as separate matrices. Therefore, $R^{(i)}$ and $L^{(i)}$ can be computed using the same memory area.

As shown in Eq. (10), $R_{j,k}$ is converted to 0 by using the rotation matrices P and Q. Here, I denotes an identity matrix.

$$P = \begin{pmatrix} I & 0 & \cdots & \cdots & 0 \\ 0 & c_1 & \cdots & s_1 & \vdots \\ \vdots & \vdots & I & \vdots & \vdots \\ \vdots & -s_1 & \cdots & c_1 & 0 \\ 0 & \cdots & \cdots & 0 & I \end{pmatrix}, \quad (8)$$

$$Q = \begin{pmatrix} I & 0 & \cdots & \cdots & 0 \\ 0 & c_2 & \cdots & -s_2 & \vdots \\ \vdots & \vdots & I & \vdots & \vdots \\ \vdots & s_2 & \cdots & c_2 & 0 \\ 0 & \cdots & \cdots & 0 & I \end{pmatrix}, \quad (9)$$

$$P \times \begin{pmatrix} \ddots & \cdots & \cdots & \cdots & \cdots \\ \vdots & R_{j,j} & \cdots & R_{j,k} & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & 0 & \cdots & R_{k,k} & \vdots \\ \cdots & \cdots & \cdots & \cdots & \ddots \end{pmatrix} \times Q = \begin{pmatrix} \ddots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \hat{R}_{j,j} & \cdots & 0 & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & 0 & \cdots & \hat{R}_{k,k} & \vdots \\ \cdots & \cdots & \cdots & \cdots & \ddots \end{pmatrix}. \quad (10)$$

Repeating Eq. (10), $R^{(i)}$ can be transformed to $L^{(i)}$. However, Eq. (10) is not the computation of $L^{(i)}$ from $R^{(i)}$. Therefore, Eq. (10) is not expressed using $R_{j,j}^{(i)}$, $R_{j,k}^{(i)}$, $R_{k,k}^{(i)}$, $L_{j,j}^{(i)}$, and $L_{k,k}^{(i)}$. Because P and Q are rotation matrices, θ_1 and θ_2 satisfy $c_1 = \cos\theta_1$, $s_1 = \sin\theta_1$, $c_2 = \cos\theta_2$, and $s_2 = \sin\theta_2$. Hereafter, we will discuss only those element parts whose values change.

$$\begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix}\begin{pmatrix} R_{j,j} & R_{j,k} \\ 0 & R_{k,k} \end{pmatrix}\begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix} = \begin{pmatrix} \hat{R}_{j,j} & 0 \\ 0 & \hat{R}_{k,k} \end{pmatrix}. \tag{11}$$

To compute $L^{(i)}$ from $R^{(i)}$, Eq. (11) must be repeated many times. In iterative procedures, an ordering strategy for erasing the off-diagonal elements, as explained in Section 4.2, is adopted. We used the same procedure to obtain $R^{(i+1)}$ from $L^{(i)}$.

The computation of c_1, s_1, c_2, s_2, $\hat{R}_{j,j}$, and $\hat{R}_{k,k}$ from $R_{j,j}, R_{j,k}, R_{k,k}$ is explained in Sections 4.3, 4.4, and 4.5.

4.2 Ordering Strategy and Convergence Criterion

In the ordering strategy, the off-diagonal elements in an upper triangular matrix $R^{(i)}$ are reduced to 0. Because the nonzero elements appear in the lower triangular part, that part is set as the lower triangular matrix $L^{(i)}$. The details are as follows: If $\left|R^{(0)}_{1,1}\right| \geq \left|R^{(0)}_{n,n}\right|$, we use the following strategy. The off-diagonal elements are reduced to 0 in the following order of elements: $(1,2), (1,3), \cdots, (1,n), (2,3), (2,4), \cdots, (n-2, n-1), (n-2, n), (n-1, n)$. Subsequently, the off-diagonal elements in the lower triangular matrix $L^{(i)}$ are reduced to 0 in the following order of elements: $(2,1), (3,1), \cdots, (n,1), (3,2), (4,2), \cdots, (n-1, n-2), (n, n-2), (n, n-1)$. If $\left|R^{(0)}_{1,1}\right| < \left|R^{(0)}_{n,n}\right|$, we use the following strategy. The off-diagonal elements are reduced to 0 in the following order of elements: $(n-1, n), (n-2, n), \cdots, (1, n), (n-2, n-1), (n-3, n-1), \cdots, (1,3), (1,2)$. Subsequently, the off-diagonal elements in the lower triangular matrix $L^{(i)}$ are reduced to 0 in the following order of elements: $(n, n-1), (n, n-2), \cdots, (n, 1), (n-1, n-2), (n-1, n-3), \cdots, (3,1), (2,1)$. Using the two-sided Jacobi method, all the off-diagonal elements converge to 0. The conditional instruction means that larger edge of diagonal element and its subsidiary off-diagonal elements are processed at first in order to help sorting of diagonal elements which introduced in Section 4.8. Computationally, because the number of iterations is finite, the off-diagonal elements may not be an exact 0. Therefore, if Eq. (12) is satisfied, the element is set to $R_{j,k} \leftarrow 0$.

$$\left|R_{j,k}\right| \leq \varepsilon \sqrt{\left|R_{j,j}\right|} \times \sqrt{\left|R_{k,k}\right|}. \tag{12}$$

Once all the off-diagonal elements converge to 0, the iteration is terminated.

4.3 Implementation Method Using the Arctangent Function

Unlike in the one-sided Jacobi method, the singular value decomposition using the two-sided Jacobi method requires many operations to calculate c_1, s_1, c_2, and s_2.

4.3.1 Conventional Implementation

We introduce the Upper-triangular, Left transformation first (UL) algorithm and the Upper-triangular, Right transformation first (UR) algorithm for computing c_1, s_1, c_2, and s_2 as a conventional method [8]. If $\left(R_{j,j} - R_{k,k}\right)\left(R_{j,j} + R_{k,k}\right)$ is greater than or equal to 0, the UL algorithm should be adopted as follows.

$$\theta_1 = \frac{1}{2}\tan^{-1}\left(\frac{2R_{k,k}}{\left(R_{j,j} - R_{k,k}\right)\left(R_{j,j} + R_{k,k}\right)/R_{j,k} + R_{j,k}}\right), \tag{13}$$

$$\theta_2 = \tan^{-1}\left(\frac{\underline{\underline{R_{j,k} + R_{k,k}\tan(\theta_1)}}}{R_{j,j}}\right). \tag{14}$$

Here, $-\frac{\pi}{4} \leq \theta_1 \leq \frac{\pi}{4}$ and $-\frac{\pi}{2} \leq \theta_2 \leq \frac{\pi}{2}$. Conversely, if $\left(R_{j,j} - R_{k,k}\right)\left(R_{j,j} + R_{k,k}\right)$ is negative, the UR algorithm should be adopted as follows.

$$\theta_2 = \frac{1}{2}\tan^{-1}\left(\frac{2R_{j,j}}{\left(R_{j,j} - R_{k,k}\right)\left(R_{j,j} + R_{k,k}\right)/R_{j,k} - R_{j,k}}\right), \tag{15}$$

$$\theta_1 = \tan^{-1}\left(\frac{\underline{\underline{R_{j,j}\tan(\theta_2) - R_{j,k}}}}{R_{k,k}}\right). \tag{16}$$

Here, $-\frac{\pi}{2} \leq \theta_1 \leq \frac{\pi}{2}$ and $-\frac{\pi}{4} \leq \theta_2 \leq \frac{\pi}{4}$. Then, we compute c_1, s_1, c_2, and s_2.

$$c_1 = \cos(\theta_1),\ s_1 = \sin(\theta_1), \tag{17}$$

$$c_2 = \cos(\theta_2),\ s_2 = \sin(\theta_2). \tag{18}$$

Then, the computed c_1, s_1, c_2, and s_2 are substituted in Eqs. (19) and (20);

$$u = c_1 + c_2, \tag{19}$$

$$\hat{R}_{j,j} = \underline{\underline{R_{j,j} + \frac{s_2}{u} \times R_{j,k}}},\ \hat{R}_{k,k} = \underline{\underline{R_{k,k} - \frac{s_1}{u} \times R_{j,k}}}. \tag{20}$$

The fused multiply-accumulate operation can be adopted in the double underlined part of the equations. It reduces the error of the final operation result by performing a product–sum operation in one instruction without rounding up the integration in the middle and is important for the achievement of high accuracy.

4.3.2 Proposed Implementation

In numerical computations, performing such a large number of operations introduces numerous errors into the variables under computation. Therefore, we propose to implement a method using the arctangent function. In the proposed implementation, the number of operations for computing c_1, s_1, c_2, and s_2 is decreased by using α and β.

Here, c_1, s_1, c_2, and s_2 are computed using $\tan^{-1}$, θ_1, and θ_2 as follows:

$$\alpha = \tan^{-1}\left(\frac{R_{j,k}}{R_{j,j} - R_{k,k}}\right), \tag{21}$$

$$\beta = \tan^{-1}\left(\frac{-R_{j,k}}{R_{j,j} + R_{k,k}}\right), \tag{22}$$

$$\theta_1 = \frac{1}{2}(\alpha + \beta),\ \theta_2 = \frac{1}{2}(\alpha - \beta), \tag{23}$$

$$c_1 = \cos(\theta_1),\ s_1 = \sin(\theta_1), \tag{24}$$

$$c_2 = \cos(\theta_2),\ s_2 = \sin(\theta_2). \tag{25}$$

Here, $-\frac{\pi}{2} \leq \theta_1 \leq \frac{\pi}{2}$ and $-\frac{\pi}{2} \leq \theta_2 \leq \frac{\pi}{2}$. Then, the computed c_1, s_1, c_2, and s_2 are substituted in Eqs. (26) and (27);

$$u = c_1 + c_2, \tag{26}$$

$$\hat{R}_{j,j} = \underline{\underline{R_{j,j} + \frac{s_2}{u} \times R_{j,k}}},\ \hat{R}_{k,k} = \underline{\underline{R_{k,k} - \frac{s_1}{u} \times R_{j,k}}}. \tag{27}$$

We define the upper computation method as the faster version of the proposed implementation. The fused multiply-accumulate operation can be adopted in the double underlined part of the equations.

In order to reduce the error of the decomposition $\|A - U\Sigma V^{\top}\|_F$, we use the accurate version of the proposed implementation as follows:

$$\alpha = \tan^{-1}\left(\frac{R_{j,k}}{R_{j,j} - R_{k,k}}\right), \tag{28}$$

$$\beta = \tan^{-1}\left(\frac{-R_{j,k}}{R_{j,j} + R_{k,k}}\right), \tag{29}$$

$$\theta_1 = \frac{1}{2}(\alpha + \beta),\ \theta_2 = \frac{1}{2}(\alpha - \beta), \tag{30}$$

$$v_1 = \cos(\theta_1),\ w_1 = \sin(\theta_1), \tag{31}$$

$$v_2 = \cos(\theta_2),\ w_2 = \sin(\theta_2). \tag{32}$$

We compute c_1 and s_1 using the Givens rotation for $x \leftarrow v_1$ and $y \leftarrow w_1$ and compute c_2 and s_2 using the Givens rotation for $x \leftarrow v_2$ and $y \leftarrow w_2$ in Section 4.5. Finally, Eqs. (33) and (34) can be computed using c_1, s_1, c_2, and s_2 as follows:

$$u = c_1 + c_2, \tag{33}$$

$$\hat{R}_{j,j} = \underline{\underline{R_{j,j} + \frac{s_2}{u} \times R_{j,k}}},\ \hat{R}_{k,k} = \underline{\underline{R_{k,k} - \frac{s_1}{u} \times R_{j,k}}}. \tag{34}$$

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

4.4 Implementation Method by Rutishauser

Integrating the implementation method by Rutishauser [11] with the two-sided Jacobi method for eigenvalue decomposition, the fused multiply-accumulate, which can achieve high accuracy, prevents the introduction of errors in c_1, s_1, c_2, and s_2.

In addition, t_1 and t_2 are decided as follows:

$$\gamma_1 = \frac{R_{j,j} - R_{k,k}}{R_{j,k}},\ t_1 = \frac{1}{\gamma_1 + \mathrm{SIGN}\left(\sqrt{\underline{\underline{1 + (\gamma_1)^2}}}, \gamma_1\right)}, \tag{35}$$

$$\gamma_2 = -\frac{R_{j,j} + R_{k,k}}{R_{j,k}},\ t_2 = \frac{1}{\gamma_2 + \mathrm{SIGN}\left(\sqrt{\underline{\underline{1 + (\gamma_2)^2}}}, \gamma_2\right)}, \tag{36}$$

where the function SIGN(A, B) returns the value of A with the sign of B. Subsequently, using t_1 and t_2,

$$\hat{v}_1 = \underline{\underline{1 - t_1 \times t_2}},\ \hat{w}_1 = t_1 + t_2, \tag{37}$$

and

$$\hat{v}_2 = \underline{\underline{1 + t_1 \times t_2}},\ \hat{w}_2 = t_1 - t_2, \tag{38}$$

are computed. We compute c_1 and s_1 using the Givens rotation for $x \leftarrow \hat{v}_1$ and $y \leftarrow \hat{w}_1$ and compute c_2 and s_2 using the Givens rotation for $x \leftarrow \hat{v}_2$ and $y \leftarrow \hat{w}_2$ in Section 4.5. Finally, Eqs. (39) and (40) can be computed using c_1, s_1, c_2, and s_2 as follows:

$$u = c_1 + c_2, \tag{39}$$

$$\hat{R}_{j,j} = \underline{\underline{R_{j,j} + \frac{s_2}{u} \times R_{j,k}}},\ \hat{R}_{k,k} = \underline{\underline{R_{k,k} - \frac{s_1}{u} \times R_{j,k}}}. \tag{40}$$

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

Algorithm 1 Implementation of the Givens rotation

```
 1: if $x = 0$ and $y = 0$ then
 2:     $\cos(\theta) \leftarrow 1$
 3:     $\sin(\theta) \leftarrow 0$
 4:     $\sqrt{x^2 + y^2} \leftarrow 0$
 5: else
 6:     $f \leftarrow |x|$
 7:     $g \leftarrow |y|$
 8:     if $f \geq g$ then
 9:         $v \leftarrow y/f$
10:         $r \leftarrow \sqrt{\underline{\underline{1 + v^2}}}$
11:         $\cos(\theta) \leftarrow \mathrm{SIGN}(1/r, x)$
12:         $\sin(\theta) \leftarrow v/r$
13:         $\sqrt{x^2 + y^2} \leftarrow r \times f$
14:     else
15:         $u \leftarrow x/g$
16:         $r \leftarrow \sqrt{\underline{\underline{1 + u^2}}}$
17:         $\cos(\theta) \leftarrow u/r$
18:         $\sin(\theta) \leftarrow \mathrm{SIGN}(1/r, y)$
19:         $\sqrt{x^2 + y^2} \leftarrow r \times g$
20:     end if
21: end if
```

4.5 Implementation Method Using the Givens Rotation

4.5.1 Implementation of the Givens Rotation

Consider the Givens rotation as follows:

$$\cos(\theta) = \frac{x}{\sqrt{x^2 + y^2}}, \quad \sin(\theta) = \frac{y}{\sqrt{x^2 + y^2}}. \tag{41}$$

Here, we executed Algorithm 1 to compute $\cos(\theta)$, $\sin(\theta)$, and $\sqrt{x^2 + y^2}$. To avoid overflow and underflow, the Givens rotation should be implemented as Algorithm 1. When $\sqrt{x^2 + y^2}$ is not required, $\sqrt{x^2 + y^2} \leftarrow r \times f$ and $\sqrt{x^2 + y^2} \leftarrow r \times g$ are not computed. The fused multiply-accumulate can be adopted in the double-underlined part of lines 1 and 1 of Algorithm 1.

4.5.2 Implementation Details

Algorithm 2 was executed to compute c_1, s_1, c_2, and s_2. Here, the function SIGN(A, B) returns the value of A with the sign of B. Subsequently, Eqs. (42) and (43) are computed using c_1, s_1, c_2, and s_2,

$$u \leftarrow c_1 + c_2, \tag{42}$$

$$\hat{R}_{j,j} \leftarrow \underline{\underline{R_{j,j} + \frac{s_2}{u} \times R_{j,k}}},\ \hat{R}_{k,k} \leftarrow \underline{\underline{R_{k,k} - \frac{s_1}{u} \times R_{j,k}}}. \tag{43}$$

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

4.6 Comparing the Number of Operations

We compare the number of operations for computing c_1, s_1, c_2, and s_2 in **Table 1**.

4.7 Implementation Technique for a Summation

If $|x_0|$ is considerably greater than $|x_i| (i = 1, \cdots, q)$, the method based on $T_q = \sum_{i=1}^{q} x_i$ and $S_q = x_0 + T_q$ is appropriate to compute $S_q = \sum_{i=0}^{q} x_i$. The computation process is adopted in Eqs. (20), (27), (34), (40), and (43).

Table 1 Comparing the number of operations.

	conventional (arctan)	proposed (faster, arctan)	proposed (accurate, arctan)	proposed Rutishauser	proposed Givens rotation
addition and subtraction	4	5	5	7	5
multiplication	3	2	2	0	2
division	5	4	10	12	11
The fused multiply-accumulate	3	2	4	8	10
Square root	0	0	2	4	4
$\tan^{-1}$	2	2	2	0	0
cos	2	2	2	0	0
sin	2	2	2	0	0

Algorithm 2 Implementation method using the Givens rotation

```
1: $f_1 \leftarrow R_{j,j} - R_{k,k}$
2: $f_1 \leftarrow f_1 + \mathrm{SIGN}\left(\underline{\sqrt{f_1^2 + R_{j,k}^2}}, f_1\right)$ The Givens rotation is adopted in the underlined part
3: if $f_1 \geq 0$ then
4:     $g_1 \leftarrow R_{j,k}$
5: else
6:     $g_1 \leftarrow -R_{j,k}$
7:     $f_1 \leftarrow -f_1$
8: end if
9: $f_2 \leftarrow R_{j,j} + R_{k,k}$
10: $f_2 \leftarrow f_2 + \mathrm{SIGN}\left(\underline{\sqrt{f_2^2 + R_{j,k}^2}}, f_2\right)$ The Givens rotation is adopted in the underlined part
11: if $f_2 \geq 0$ then
12:     $g_2 \leftarrow -R_{j,k}$
13: else
14:     $g_2 \leftarrow R_{j,k}$
15:     $f_2 \leftarrow -f_2$
16: end if
17: if $f_1 \geq f_2$ then
18:     $t_1 \leftarrow g_1/f_1$
19:     $\hat{c}_1 \leftarrow \underline{\underline{-t_1 \times g_2 + f_2}}$
20:     $\hat{s}_1 \leftarrow \underline{\underline{t_1 \times f_2 + g_2}}$
21:     $\hat{c}_2 \leftarrow \underline{\underline{t_1 \times g_2 + f_2}}$
22:     $\hat{s}_2 \leftarrow \underline{\underline{t_1 \times f_2 - g_2}}$
23: else
24:     $t_2 \leftarrow g_2/f_2$
25:     $\hat{c}_1 \leftarrow \underline{\underline{-g_1 \times t_2 + f_1}}$
26:     $\hat{s}_1 \leftarrow \underline{\underline{f_1 \times t_2 + g_1}}$
27:     $\hat{c}_2 \leftarrow \underline{\underline{g_1 \times t_2 + f_1}}$
28:     $\hat{s}_2 \leftarrow \underline{\underline{-f_1 \times t_2 + g_1}}$
29: end if
30: Compute $c_1$ and $s_1$ using the Givens rotation for $x \leftarrow \hat{c}_1$ and $y \leftarrow \hat{s}_1$
31: Compute $c_2$ and $s_2$ using the Givens rotation for $x \leftarrow \hat{c}_2$ and $y \leftarrow \hat{s}_2$
```

4.8 Addition of Sorting Function to the Two-sided Jacobi Method

From the Eq. (12), if both diagonal elements of the 2×2 matrix are large, the non-diagonal elements are considered to be converging even if they are somewhat large. Thus, when the large diagonal elements are collected in the upper left corner, the singular values in the diagonal elements converge from the upper left to the lower right. As a result, it is possible to separate the locations that converge at the beginning from those that converge later. Therefore, the convergence is improved. For these reasons, in order to collect the large diagonal elements in the upper part of the left and the smaller elements in the lower right, we add the function of sorting to the two-sided Jacobi method as **Table 2**.

Table 2 Case list for function of sorting.

cases			setting
$\left\|R_{1,1}^{(0)}\right\| \geq \left\|R_{n,n}^{(0)}\right\|$	$\left\|\hat{R}_{j,j}\right\| < \left\|\hat{R}_{k,k}\right\|$	$s_1 > 0$	$c_1 \leftarrow s_1$ and $s_1 \leftarrow -c_1$
$\left\|R_{1,1}^{(0)}\right\| \geq \left\|R_{n,n}^{(0)}\right\|$	$\left\|\hat{R}_{j,j}\right\| < \left\|\hat{R}_{k,k}\right\|$	$s_1 \leq 0$	$c_1 \leftarrow -s_1$ and $s_1 \leftarrow c_1$
$\left\|R_{1,1}^{(0)}\right\| \geq \left\|R_{n,n}^{(0)}\right\|$	$\left\|\hat{R}_{j,j}\right\| < \left\|\hat{R}_{k,k}\right\|$	$s_2 > 0$	$c_2 \leftarrow s_2$ and $s_2 \leftarrow -c_2$
$\left\|R_{1,1}^{(0)}\right\| \geq \left\|R_{n,n}^{(0)}\right\|$	$\left\|\hat{R}_{j,j}\right\| < \left\|\hat{R}_{k,k}\right\|$	$s_2 \leq 0$	$c_2 \leftarrow -s_2$ and $s_2 \leftarrow c_2$
$\left\|R_{1,1}^{(0)}\right\| < \left\|R_{n,n}^{(0)}\right\|$	$\left\|\hat{R}_{j,j}\right\| \geq \left\|\hat{R}_{k,k}\right\|$	$s_1 > 0$	$c_1 \leftarrow s_1$ and $s_1 \leftarrow -c_1$
$\left\|R_{1,1}^{(0)}\right\| < \left\|R_{n,n}^{(0)}\right\|$	$\left\|\hat{R}_{j,j}\right\| \geq \left\|\hat{R}_{k,k}\right\|$	$s_1 \leq 0$	$c_1 \leftarrow -s_1$ and $s_1 \leftarrow c_1$
$\left\|R_{1,1}^{(0)}\right\| < \left\|R_{n,n}^{(0)}\right\|$	$\left\|\hat{R}_{j,j}\right\| \geq \left\|\hat{R}_{k,k}\right\|$	$s_2 > 0$	$c_2 \leftarrow s_2$ and $s_2 \leftarrow -c_2$
$\left\|R_{1,1}^{(0)}\right\| < \left\|R_{n,n}^{(0)}\right\|$	$\left\|\hat{R}_{j,j}\right\| \geq \left\|\hat{R}_{k,k}\right\|$	$s_2 \leq 0$	$c_2 \leftarrow -s_2$ and $s_2 \leftarrow c_2$

In the case $\left|R_{1,1}^{(0)}\right| \geq \left|R_{n,n}^{(0)}\right|$, if $\left|\hat{R}_{j,j}\right| < \left|\hat{R}_{k,k}\right|$ and $s_1 > 0$ are satisfied, we set $c_1 \leftarrow s_1$ and $s_1 \leftarrow -c_1$, which means $\theta_1 \leftarrow \theta_1 - \frac{\pi}{2}$, and, if $\left|\hat{R}_{j,j}\right| < \left|\hat{R}_{k,k}\right|$ and $s_1 \leq 0$ are satisfied, we set $c_1 \leftarrow -s_1$ and $s_1 \leftarrow c_1$, which means $\theta_1 \leftarrow \theta_1 + \frac{\pi}{2}$. If $\left|\hat{R}_{j,j}\right| < \left|\hat{R}_{k,k}\right|$ and $s_2 > 0$ are satisfied, we set $c_2 \leftarrow s_2$ and $s_2 \leftarrow -c_2$, which means $\theta_2 \leftarrow \theta_2 - \frac{\pi}{2}$, and, if $\left|\hat{R}_{j,j}\right| < \left|\hat{R}_{k,k}\right|$ and $s_2 \leq 0$ are satisfied, we set $c_2 \leftarrow -s_2$ and $s_2 \leftarrow c_2$, which means $\theta_2 \leftarrow \theta_2 + \frac{\pi}{2}$. If $\frac{\pi}{2}$ is subtracted from or added to both θ_1 and θ_2, we set $\hat{R}_{j,j} \leftarrow \hat{R}_{k,k}$, $\hat{R}_{k,k} \leftarrow \hat{R}_{j,j}$. Otherwise, we set $\hat{R}_{j,j} \leftarrow -\hat{R}_{k,k}$, $\hat{R}_{k,k} \leftarrow -\hat{R}_{j,j}$.

In the case $\left|R_{1,1}^{(0)}\right| < \left|R_{n,n}^{(0)}\right|$, if $\left|\hat{R}_{j,j}\right| \geq \left|\hat{R}_{k,k}\right|$ and $s_1 > 0$ are satisfied, we set $c_1 \leftarrow s_1$ and $s_1 \leftarrow -c_1$, which means $\theta_1 \leftarrow \theta_1 - \frac{\pi}{2}$, and, if $\left|\hat{R}_{j,j}\right| \geq \left|\hat{R}_{k,k}\right|$ and $s_1 \leq 0$ are satisfied, we set $c_1 \leftarrow -s_1$ and $s_1 \leftarrow c_1$, which means $\theta_1 \leftarrow \theta_1 + \frac{\pi}{2}$. If $\left|\hat{R}_{j,j}\right| \geq \left|\hat{R}_{k,k}\right|$ and $s_2 > 0$ are satisfied, we set $c_2 \leftarrow s_2$ and $s_2 \leftarrow -c_2$, which means $\theta_2 \leftarrow \theta_2 - \frac{\pi}{2}$, and, if $\left|\hat{R}_{j,j}\right| \geq \left|\hat{R}_{k,k}\right|$ and $s_2 \leq 0$ are satisfied, we set $c_2 \leftarrow -s_2$ and $s_2 \leftarrow c_2$, which means $\theta_2 \leftarrow \theta_2 + \frac{\pi}{2}$. If $\frac{\pi}{2}$ is subtracted from or added to both θ_1 and θ_2, we set $\hat{R}_{j,j} \leftarrow \hat{R}_{k,k}$, $\hat{R}_{k,k} \leftarrow \hat{R}_{j,j}$. Otherwise, we set $\hat{R}_{j,j} \leftarrow -\hat{R}_{k,k}$, $\hat{R}_{k,k} \leftarrow -\hat{R}_{j,j}$.

Furthermore, by performing the above operation, the two-sided Jacobi method becomes a sorting function from larger singular values to smaller singular values. Notably, after performing the above operation, c_1 and c_2 are still nonnegative.

5. Correction of c_1, s_1, c_2, or s_2

Using the implementation method by Rutishauser [11], we can correct c_1, s_1, c_2, or s_2. Let $\tilde{c}$, $\tilde{s}$, $\hat{c}$, and $\hat{s}$ be four variables that represent c_1 and c_2, s_1 and s_2, correction result for c_1 and c_2, and correction result for s_1 and s_2, respectively.

5.1 False Position Method

The false position method is depicted in **Fig. 2**. 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. (44) is set to a new posi-

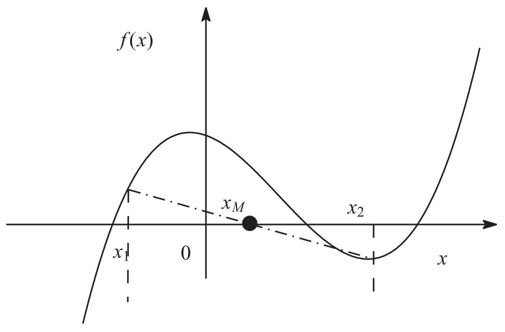

Fig. 2 False position method.

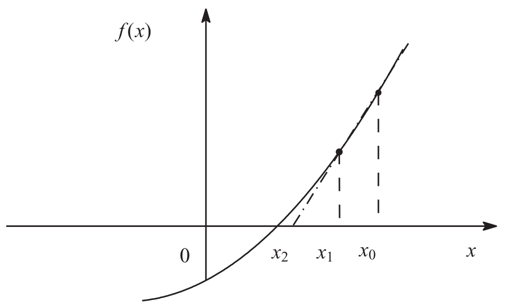

Fig. 3 Secant method.

tion to compute the real root x in $f(x) = 0$. One has the following:

$$x_M = \frac{x_1 \times f(x_2) - x_2 \times f(x_1)}{f(x_2) - f(x_1)}. \tag{44}$$

Here, if the sign of $f(x_1)$ is the same as that of $f(x_M)$, then $x_1 \leftarrow x_M$. However, 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. 2, x_M is set to a new x_1.

5.2 Secant Method

The secant method is depicted in **Fig. 3**.

In the secant method, the following recurrence relation is adopted to compute the real root x in $f(x) = 0$:

$$\begin{aligned} x_{n+1} &= x_n - f(x_n) \times \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \\ &= \frac{x_{n-1} f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}. \end{aligned} \tag{45}$$

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. (45), Eq. (44) is obtained.

5.3 Correction Method

Theoretically, the equation $\tilde{c}^2 + \tilde{s}^2 = 1$ is satisfied. However, computationally, it is not satisfied because of a rounding error. Therefore, we propose a correction method for $\tilde{c}$ and $\tilde{s}$.

Assuming that $\tilde{s}$ is correct, $\tilde{c}$ is calculated as follows:

$$x^2 + \tilde{s}^2 = 1. \tag{46}$$

Assuming that $\tilde{c}$ is correct, $\tilde{s}$ is computed as follows:

$$\tilde{c}^2 + x^2 = 1. \tag{47}$$

Equations (46) and (47) can be appropriately used by introducing $\tilde{c} = \cos\theta$ and $\tilde{s} = \sin\theta$. For the case where $-\frac{\pi}{4} \le \theta \le \frac{\pi}{4}$, Eq. (46) is used, whereas Eq. (47) is adopted if $\frac{\pi}{4} < \theta \le \frac{\pi}{2}$ or $-\frac{\pi}{2} \le \theta < -\frac{\pi}{4}$. For the singular value decomposition using the two-sided Jacobi method, $\tilde{c} \ge 0$ is satisfied. Therefore, we can assume that $-\frac{\pi}{2} \le \theta \le \frac{\pi}{2}$.

When both the nonlinear single equation $f(x) = 0$ and initial numbers x_0 and x_1 are given, then x_2, which is the result of one iteration of the secant method, is equal to x_M achieved using the false position method. One has the following relation:

$$x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}, \tag{48}$$

In the case where $-\frac{\pi}{4} \le \theta \le \frac{\pi}{4}$, $\tilde{c}$ is recomputed using eq.(46). The case where $-\frac{\pi}{4} \le \theta \le \frac{\pi}{4}$ can be considered equivalent to $\tilde{c} \ge |\tilde{s}|$. To compute $\tilde{c}$, the initial numbers are set to $x_0 = 1$ and $x_1 = \tilde{c}$. When $f(x) = x^2 + \tilde{s}^2 - 1$,

$$\hat{c} = \frac{(\tilde{c}^2 + \tilde{s}^2 - 1) - \tilde{c}\tilde{s}^2}{(\tilde{c}^2 + \tilde{s}^2 - 1) - (\tilde{s}^2)} = 1 - \tilde{s} \times \frac{\tilde{s}}{1 + \tilde{c}}, \tag{49}$$

is obtained, where $\hat{c}$ is more appropriate for satisfying $f(x) = x^2 + \tilde{s}^2 - 1$. The Givens rotation for vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as follows:

$$\boldsymbol{x} \leftarrow \tilde{c}\boldsymbol{x} + \tilde{s}\boldsymbol{y}, \ \boldsymbol{y} \leftarrow -\tilde{s}\boldsymbol{x} + \tilde{c}\boldsymbol{y}. \tag{50}$$

When not using $\tilde{c}$ but $\hat{c}$,

$$z_1 = \frac{\tilde{s}}{1 + \tilde{c}}, \tag{51}$$

$$\boldsymbol{x} \leftarrow (1 - \tilde{s} \times z_1)\boldsymbol{x} + \tilde{s}\boldsymbol{y} = \underline{\underline{\tilde{s}\left(\underline{\underline{-z_1\boldsymbol{x} + \boldsymbol{y}}}\right) + \boldsymbol{x}}}, \tag{52}$$

$$\boldsymbol{y} \leftarrow -\tilde{s}\boldsymbol{x} + (1 - \tilde{s} \times z_1)\boldsymbol{y} = \underline{\underline{-\tilde{s}\left(\underline{\underline{z_1\boldsymbol{y} + \boldsymbol{x}}}\right) + \boldsymbol{y}}}, \tag{53}$$

is obtained.

The case wherein $\frac{\pi}{4} < \theta \le \frac{\pi}{2}$ can be regarded as equivalent to $\tilde{c} < |\tilde{s}|$ and $\tilde{s} \ge 0$. To compute $\tilde{s}$, the initial numbers are set to $x_0 = 1$ and $x_1 = \tilde{s}$. When $f(x) = \tilde{c}^2 + x^2 - 1$,

$$\hat{s} = \frac{(\tilde{c}^2 + \tilde{s}^2 - 1) - \tilde{s}\tilde{c}^2}{(\tilde{c}^2 + \tilde{s}^2 - 1) - (\tilde{c}^2)} = 1 - \tilde{c} \times \frac{\tilde{c}}{1 + \tilde{s}}, \tag{54}$$

is obtained, where $\hat{s}$ is more appropriate for satisfying $f(x) = \tilde{c}^2 + x^2 - 1$. When not using $\tilde{s}$ but $\hat{s}$, the Givens rotation for vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as follows:

$$z_2 = \frac{\tilde{c}}{1 + \tilde{s}}, \tag{55}$$

$$\boldsymbol{x} \leftarrow \tilde{c}\boldsymbol{x} + (1 - \tilde{c} \times z_2)\boldsymbol{y} = \underline{\underline{\tilde{c}\left(\underline{\underline{-z_2\boldsymbol{y} + \boldsymbol{x}}}\right) + \boldsymbol{y}}}, \tag{56}$$

$$\boldsymbol{y} \leftarrow -(1 - \tilde{c} \times z_2)\boldsymbol{x} + \tilde{c}\boldsymbol{y} = \underline{\underline{\tilde{c}\left(\underline{\underline{z_2\boldsymbol{x} + \boldsymbol{y}}}\right) - \boldsymbol{x}}}. \tag{57}$$

The case where $-\frac{\pi}{2} \le \theta < -\frac{\pi}{4}$ can be regarded as equivalent to $\tilde{c} < |\tilde{s}|$ and $\tilde{s} \le 0$. To compute $\tilde{s}$, the initial numbers are set to $x_0 = -1$ and $x_1 = \tilde{s}$. When $f(x) = \tilde{c}^2 + x^2 - 1$,

$$\hat{s} = \frac{-(\tilde{c}^2 + \tilde{s}^2 - 1) - \tilde{s}\tilde{c}^2}{(\tilde{c}^2 + \tilde{s}^2 - 1) - (\tilde{c}^2)} = -1 + \tilde{c} \times \frac{\tilde{c}}{1 - \tilde{s}}, \tag{58}$$

is obtained, where $\hat{s}$ is more appropriate for satisfying $f(x) =$

Table 4 Comparison of Jacobi SVD algorithms (without the incorporation of our implementation technique).

	One-sided Jacobi	Two-sided Jacobi Conventional (arctan)	Two-sided Jacobi Proposed (faster, arctan)
A_1			
$\|\|U^\top U - I\|\|_F$	$1.91 * 10^{-4}$	$1.11 * 10^{-4}$	$1.12 * 10^{-4}$
$\|\|V^\top V - I\|\|_F$	$8.20 * 10^{-5}$	$1.03 * 10^{-4}$	$1.03 * 10^{-4}$
$\|\|A - U\Sigma V^\top\|\|_F$	$6.33 * 10^{-4}$	$5.72 * 10^{-4}$	$5.69 * 10^{-4}$
Computation time [s]	1.140	0.882	0.741
A_2			
$\|\|U^\top U - I\|\|_F$	$5.51 * 10^{-4}$	$3.18 * 10^{-4}$	$3.21 * 10^{-4}$
$\|\|V^\top V - I\|\|_F$	$1.78 * 10^{-4}$	$2.87 * 10^{-4}$	$2.95 * 10^{-4}$
$\|\|A - U\Sigma V^\top\|\|_F$	$2.54 * 10^{-3}$	$2.27 * 10^{-3}$	$2.53 * 10^{-3}$
Computation time [s]	10.246	8.160	6.358
A_3			
$\|\|U^\top U - I\|\|_F$	$1.00 * 10^{-3}$	$6.25 * 10^{-4}$	$6.11 * 10^{-4}$
$\|\|V^\top V - I\|\|_F$	$2.81 * 10^{-4}$	$5.62 * 10^{-4}$	$5.54 * 10^{-4}$
$\|\|A - U\Sigma V^\top\|\|_F$	$4.12 * 10^{-3}$	$4.34 * 10^{-3}$	$4.96 * 10^{-3}$
Computation time [s]	38.645	35.668	28.868
A_4			
$\|\|U^\top U - I\|\|_F$	$1.50 * 10^{-3}$	$9.70 * 10^{-4}$	$9.64 * 10^{-4}$
$\|\|V^\top V - I\|\|_F$	$3.91 * 10^{-4}$	$9.20 * 10^{-4}$	$9.11 * 10^{-4}$
$\|\|A - U\Sigma V^\top\|\|_F$	$6.77 * 10^{-3}$	$7.13 * 10^{-3}$	$7.15 * 10^{-3}$
Computation time [s]	92.253	133.499	113.791
A_5			
$\|\|U^\top U - I\|\|_F$	$2.01 * 10^{-4}$	$1.47 * 10^{-4}$	$1.50 * 10^{-4}$
$\|\|V^\top V - I\|\|_F$	$9.40 * 10^{-5}$	$1.66 * 10^{-4}$	$1.68 * 10^{-4}$
$\|\|A - U\Sigma V^\top\|\|_F$	$9.22 * 10^{-4}$	$7.34 * 10^{-4}$	$9.32 * 10^{-4}$
Computation time [s]	0.989	0.993	0.922
A_6			
$\|\|U^\top U - I\|\|_F$	$5.29 * 10^{-4}$	$4.67 * 10^{-4}$	$4.71 * 10^{-4}$
$\|\|V^\top V - I\|\|_F$	$2.18 * 10^{-4}$	$5.18 * 10^{-4}$	$5.25 * 10^{-4}$
$\|\|A - U\Sigma V^\top\|\|_F$	$2.03 * 10^{-3}$	$2.56 * 10^{-3}$	$2.80 * 10^{-3}$
Computation time [s]	8.402	8.556	8.253
A_7			
$\|\|U^\top U - I\|\|_F$	$1.00 * 10^{-3}$	$9.38 * 10^{-4}$	$9.30 * 10^{-4}$
$\|\|V^\top V - I\|\|_F$	$3.84 * 10^{-4}$	$1.05 * 10^{-3}$	$1.04 * 10^{-3}$
$\|\|A - U\Sigma V^\top\|\|_F$	$3.46 * 10^{-3}$	$4.31 * 10^{-3}$	$4.94 * 10^{-3}$
Computation time [s]	30.260	39.966	39.056
A_8			
$\|\|U^\top U - I\|\|_F$	$1.59 * 10^{-3}$	$1.54 * 10^{-3}$	$1.54 * 10^{-3}$
$\|\|V^\top V - I\|\|_F$	$5.20 * 10^{-4}$	$1.69 * 10^{-3}$	$1.70 * 10^{-3}$
$\|\|A - U\Sigma V^\top\|\|_F$	$5.95 * 10^{-3}$	$7.51 * 10^{-3}$	$7.15 * 10^{-3}$
Computation time [s]	78.082	154.763	150.870

$\tilde{c}^2 + x^2 - 1$. When not using $\tilde{s}$ but $\hat{s}$, the Givens rotation for vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as follows:

$$z_3 = \frac{\tilde{c}}{1 - \tilde{s}}, \tag{59}$$

$$\boldsymbol{x} \leftarrow \tilde{c}\boldsymbol{x} + (-1 + \tilde{c} \times z_3)\,\boldsymbol{y} = \tilde{c}\left(\underline{\underline{z_3\boldsymbol{y} + \boldsymbol{x}}}\right) - \boldsymbol{y}, \tag{60}$$

$$\boldsymbol{y} \leftarrow -(-1 + \tilde{c} \times z_3)\,\boldsymbol{x} + \tilde{c}\boldsymbol{y} = \tilde{c}\left(\underline{\underline{-z_3\boldsymbol{x} + \boldsymbol{y}}}\right) + \boldsymbol{x}. \tag{61}$$

Notably, the fused multiply-accumulate can be adopted in the double underlined part.

6. Experiments

We checked whether the two-sided Jacobi method (arctan and Rutishauser versions) had shorter computation time and higher accuracy than those of the one-sided Jacobi method implemented in LAPACK [10], for small matrices. The experimental environment is mentioned in **Table 3**. We used the following eight matrices for the comparison:

- A_1 (dimension size: 500×500, an upper triangular matrix)

Table 3 Experimental environment.

CPU	Intel(R) Xeon(R) Silver 4116 @ 2.10 GHz (2 CPUs)
RAM	192 GB
OS	Ubuntu 20.04.1 LTS
Compiler	gfortran 9.3.0
Options	-O3 -mtune=native -march=native
Software	Lapack 3.9.0
Precision	single precision

- A_2 (dimension size: 1000×1000, an upper triangular matrix)
- A_3 (dimension size: 1500×1500, an upper triangular matrix)
- A_4 (dimension size: 2000×2000, an upper triangular matrix)

and

- A_5 (dimension size: 500×500, an upper triangular matrix)
- A_6 (dimension size: 1000×1000, an upper triangular matrix)
- A_7 (dimension size: 1500×1500, an upper triangular matrix)
- A_8 (dimension size: 2000×2000, an upper triangular matrix)

In A_1, A_2, A_3, and A_4, all the elements are set to random numbers $\in [0, 1]$ generated using a uniform random number generator. However, in A_5, A_6, A_7, and A_8, all the elements are set to 1. Generally, the dimension size in the SST method is set to the small

Table 5 Comparison of Jacobi SVD algorithms (with the incorporation of our implementation technique).

	One-sided Jacobi	Two-sided Jacobi Conventional (arctan)	Two-sided Jacobi Proposed (faster, arctan)	Two-sided Jacobi Proposed (accurate, arctan)	Two-sided Jacobi Proposed Rutishauser	Two-sided Jacobi Proposed Givens rotation
A_1						
$\|U^\top U - I\|_F$	$1.91 * 10^{-4}$	$4.36 * 10^{-5}$	$4.30 * 10^{-5}$	$4.33 * 10^{-5}$	$4.32 * 10^{-5}$	$4.34 * 10^{-5}$
$\|V^\top V - I\|_F$	$8.20 * 10^{-5}$	$4.32 * 10^{-5}$	$4.32 * 10^{-5}$	$4.31 * 10^{-5}$	$4.30 * 10^{-5}$	$4.33 * 10^{-5}$
$\|A - U\Sigma V^\top\|_F$	$6.33 * 10^{-4}$	$3.74 * 10^{-4}$	$3.69 * 10^{-4}$	$3.70 * 10^{-4}$	$3.77 * 10^{-4}$	$3.60 * 10^{-4}$
Computation time [s]	1.140	0.819	0.712	0.698	0.683	0.694
A_2						
$\|U^\top U - I\|_F$	$5.51 * 10^{-4}$	$8.90 * 10^{-5}$	$8.87 * 10^{-5}$	$8.92 * 10^{-5}$	$8.93 * 10^{-5}$	$8.86 * 10^{-5}$
$\|V^\top V - I\|_F$	$1.78 * 10^{-4}$	$8.65 * 10^{-5}$	$8.65 * 10^{-5}$	$8.65 * 10^{-5}$	$8.59 * 10^{-5}$	$8.61 * 10^{-5}$
$\|A - U\Sigma V^\top\|_F$	$2.54 * 10^{-3}$	$1.07 * 10^{-3}$	$1.06 * 10^{-3}$	$1.08 * 10^{-3}$	$1.07 * 10^{-3}$	$1.06 * 10^{-3}$
Computation time [s]	10.246	6.325	6.191	6.030	6.032	6.265
A_3						
$\|U^\top U - I\|_F$	$1.00 * 10^{-3}$	$1.41 * 10^{-4}$	$1.40 * 10^{-4}$	$1.41 * 10^{-4}$	$1.39 * 10^{-4}$	$1.40 * 10^{-4}$
$\|V^\top V - I\|_F$	$2.81 * 10^{-4}$	$1.30 * 10^{-4}$	$1.30 * 10^{-4}$	$1.30 * 10^{-4}$	$1.29 * 10^{-4}$	$1.29 * 10^{-4}$
$\|A - U\Sigma V^\top\|_F$	$4.12 * 10^{-3}$	$2.02 * 10^{-3}$	$2.04 * 10^{-3}$	$2.07 * 10^{-3}$	$2.01 * 10^{-3}$	$2.11 * 10^{-3}$
Computation time [s]	38.645	28.324	27.150	26.822	27.387	28.528
A_4						
$\|U^\top U - I\|_F$	$1.50 * 10^{-3}$	$1.96 * 10^{-4}$	$1.96 * 10^{-4}$	$1.96 * 10^{-4}$	$1.95 * 10^{-4}$	$1.95 * 10^{-4}$
$\|V^\top V - I\|_F$	$3.91 * 10^{-4}$	$1.73 * 10^{-4}$	$1.73 * 10^{-4}$	$1.74 * 10^{-4}$	$1.71 * 10^{-4}$	$1.72 * 10^{-4}$
$\|A - U\Sigma V^\top\|_F$	$6.77 * 10^{-3}$	$3.16 * 10^{-3}$	$3.18 * 10^{-3}$	$3.08 * 10^{-3}$	$3.17 * 10^{-3}$	$3.14 * 10^{-3}$
Computation time [s]	92.253	104.210	102.440	99.877	104.007	104.847
A_5						
$\|U^\top U - I\|_F$	$2.01 * 10^{-4}$	$4.48 * 10^{-5}$	$4.45 * 10^{-5}$	$4.51 * 10^{-5}$	$4.52 * 10^{-5}$	$4.48 * 10^{-5}$
$\|V^\top V - I\|_F$	$9.40 * 10^{-5}$	$4.51 * 10^{-5}$	$4.47 * 10^{-5}$	$4.48 * 10^{-5}$	$4.52 * 10^{-5}$	$4.51 * 10^{-5}$
$\|A - U\Sigma V^\top\|_F$	$9.22 * 10^{-4}$	$5.52 * 10^{-4}$	$5.87 * 10^{-4}$	$5.55 * 10^{-4}$	$5.71 * 10^{-4}$	$5.72 * 10^{-4}$
Computation time [s]	0.989	0.789	0.757	0.737	0.721	0.747
A_6						
$\|U^\top U - I\|_F$	$5.29 * 10^{-4}$	$9.12 * 10^{-5}$	$9.11 * 10^{-5}$	$9.18 * 10^{-5}$	$9.16 * 10^{-5}$	$9.16 * 10^{-5}$
$\|V^\top V - I\|_F$	$2.18 * 10^{-4}$	$9.13 * 10^{-5}$	$9.15 * 10^{-5}$	$9.16 * 10^{-5}$	$9.14 * 10^{-5}$	$9.13 * 10^{-5}$
$\|A - U\Sigma V^\top\|_F$	$2.03 * 10^{-3}$	$1.77 * 10^{-3}$	$1.74 * 10^{-3}$	$1.72 * 10^{-3}$	$1.77 * 10^{-3}$	$1.77 * 10^{-3}$
Computation time [s]	8.402	6.533	6.469	6.224	6.250	6.226
A_7						
$\|U^\top U - I\|_F$	$1.00 * 10^{-3}$	$1.39 * 10^{-4}$	$1.39 * 10^{-4}$	$1.39 * 10^{-4}$	$1.39 * 10^{-4}$	$1.39 * 10^{-4}$
$\|V^\top V - I\|_F$	$3.84 * 10^{-4}$	$1.38 * 10^{-4}$	$1.38 * 10^{-4}$	$1.38 * 10^{-4}$	$1.38 * 10^{-4}$	$1.38 * 10^{-4}$
$\|A - U\Sigma V^\top\|_F$	$3.46 * 10^{-3}$	$3.52 * 10^{-3}$	$3.93 * 10^{-3}$	$3.51 * 10^{-3}$	$3.52 * 10^{-3}$	$3.40 * 10^{-3}$
Computation time [s]	30.260	29.807	29.257	28.888	28.822	28.928
A_8						
$\|U^\top U - I\|_F$	$1.59 * 10^{-3}$	$1.86 * 10^{-4}$	$1.85 * 10^{-4}$	$1.85 * 10^{-4}$	$1.85 * 10^{-4}$	$1.86 * 10^{-4}$
$\|V^\top V - I\|_F$	$5.20 * 10^{-4}$	$1.84 * 10^{-4}$	$1.84 * 10^{-4}$	$1.84 * 10^{-4}$	$1.84 * 10^{-4}$	$1.84 * 10^{-4}$
$\|A - U\Sigma V^\top\|_F$	$5.95 * 10^{-3}$	$5.91 * 10^{-3}$	$5.88 * 10^{-3}$	$5.59 * 10^{-3}$	$5.06 * 10^{-3}$	$4.96 * 10^{-3}$
Computation time [s]	78.082	111.501	110.080	108.564	108.220	109.734

size of 100×100 or less. The dimension size of each of these test matrices is greater than that used in the SST method. However, if the dimension size is too small, it is difficult to perform computation-time comparisons. Therefore, in the experiments, the dimension size of each test matrix was selected as sufficiently small and not too small.

The performance results are presented in **Table 4**. The comparison is made without the incorporation of our implementation technique, which is described in Sections 4.7, 4.8, and 5, in Table 4. In terms of the computation speed, the conventional method is not better than the proposed method. If we do not incorporate our implementation technique in Sections 4.7, 4.8, and 5, the performance of the two-sided Jacobi method is not better than that of the one-sided Jacobi method implemented in LAPACK [10] in terms of computation time and accuracy.

The performance results are presented in **Table 5**. The comparison is made with the incorporation of our implementation technique in Sections 4.7, 4.8, and 5 in Table 5. The two-sided Jacobi method (Givens rotation) has shorter computation time and higher accuracy than those of the one-sided Jacobi method implemented in LAPACK for both the 500×500, 1000×1000, and 1500×1500 upper triangular matrices whose elements are generated using a uniform random number generator and also, for the 500×500, 1000×1000, and the 1500×1500 upper triangular matrix whose all elements are 1.

The accuracy of the implemented two-sided Jacobi method, which is applied using the arctangent function, the implementation procedure by Rutishauser, or Givens rotation, is higher than that of the one-sided Jacobi method, for small matrices. Particularly, the orthogonality in the implemented two-sided Jacobi method is better than that of the one-sided Jacobi method. The smaller is the matrix size, the higher is the accuracy of the implemented two-sided Jacobi method.

In test matrices A_4 and A_8, each of whose dimension size is 2000×2000, the computation time of the implemented two-sided Jacobi method is slightly longer than that of the one-sided Jacobi method. In the other test matrices, the computation time of the implemented two-sided Jacobi method is shorter than that of the one-sided Jacobi method.

The dimension size in the SST method is set to the small size of 100×100 or less. Therefore, using the implemented two-sided Jacobi method, the SST method can be performed with high speed and high accuracy. Consequently, the implemented two-sided Jacobi method is appropriate for the SST method.

7. Conclusion

To perform the singular value decomposition of small matrices, we improved the two-sided Jacobi method. Particularly, we proposed the three implementation methods that used the arctangent function, implementation procedure by Rutishauser, and Givens rotation, respectively. We confirmed experimentally that for small matrices, the implemented two-sided Jacobi method had shorter computation time and higher accuracy than those of the one-sided Jacobi method. The matrix size for the SST method was smaller than that of the test matrices. Therefore, the implemented two-sided Jacobi method is appropriate to apply the SST method.

For future work, we plan to apply our two-sided Jacobi method to implement the SST method for solving real problems.

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

References

[1] Brent, R.P., Luk, F.T. and van Loan, C.: Computation of the singular value decomposition using mesh-connected processors, *Journal of VLSI and Computer Systems*, Vol.1, pp.242–270 (1985).

[2] 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).

[3] Drmac, Z. and Veselic, K.: New fast and accurate Jacobi SVD algorithm: I., *SIAM J. Matrix Anal. Appl.*, Vol.29, pp.1322–1342 (2008).

[4] Drmac, Z. and Veselic, K.: New fast and accurate Jacobi SVD algorithm: II., *SIAM J. Matrix Anal. Appl.*, Vol.29, pp.1343–1362 (2008).

[5] Elsner, J. and Tsonis, A.: *Singular Spectrum Analysis: A New Tool in Time Series Analysis*, Plenum Press, New York (1996).

[6] Forsythe, G.E. and Henrici, P.: The cyclic Jacobi method for computing the principal values of a complex matrix, *Transactions of the American Mathematical Society*, Vol.94, pp.1–23 (1960).

[7] Golyandina, N., Nekrutkin, V. and Zhigljavsky, A.: *Analysis of Time Series Structure: SSA and Related Techniques*, Chapman And Hall, United Kingdom (2000).

[8] Hari, V. and Zadelj-Martić, V.: Parallelizing the Kogbetliantz Method: A First Attempt, *Journal of Numerical Analysis, Industrial and Applied Mathematics* (*JNAIAM*), Vol.2, No.1-2, pp.49–66 (2007).

[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 2019-12-19).

[11] Rutishauser, H.: The Jacobi method for real symmetric matrices, *Numerische Mathematik*, Vol.9, No.1, pp.1–10 (1966).

Sho Araki received his B.E. and M.I. degrees from Kyoto University in 2012 and 2014. His research interests include parallel algorithms for eigenvalue and singular value decomposition.

Masana Aoki received her B.E. and M.I. degrees from Kyoto University in 2018 and 2020. She has operated assets at Nomura Asset Management Co., Ltd. since 2020. Her research interests include parallel algorithms for eigenvalue and singular value decomposition.

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 for distributed memory systems.

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 has been a professor of Graduate School of Informatics, Kyoto University from 2001. His research interests includes integrable dynamical systems which originally appear in classical mechanics. But integrable systems have a rich mathematical structure. His recent subject is to design new numerical algorithms such as the mdLVs and I-SVD for singular value decomposition by using discrete-time integrable systems. He is a member of JSIAM, SIAM, MSJ and AMS.