Research Article **Open Access**

A. Ohashi* and T. Sogabe

On computing the minimum singular value of a tensor sum

https://doi.org/110.1515/spma-2019-0009
Received November 22, 2018; accepted July 17, 2019

Abstract: Recently, the Lanczos bidiagonalization method over tensor space has been proposed for computing the maximum and minimum singular values of a tensor sum T. The method over tensor space is practical in memory and has a simple implementation due to recent developments in tensor computations; however, there is still room for improvement in the convergence to the minimum singular value. This study reconstructed an invert Lanczos bidiagonalization method from vector space to tensor space. The resulting algorithm requires solving linear systems at each iteration step. Using standard direct methods, such as the LU decomposition for solving the linear systems requires a huge memory of $O(n^6)$. Therefore, this paper proposes a tensor-structure-preserving direct methods of T whose memory requirements are of $O(n^3)$, which is equivalent to the order of iterative methods. Numerical examples indicate that the number of iterations tends to be much smaller than that of the conventional method.

Keywords: tensor sum; invert Lanczos bidiagonalization; singular value; third-order tensor

1 Introduction

Consider computing the minimum singular value of the following generalized tensor sum [1]:

$$T := I_n \otimes I_m \otimes A + I_n \otimes B \otimes I_\ell + C \otimes I_m \otimes I_\ell \in \mathbb{R}^{\ell mn \times \ell mn}, \tag{1}$$

where $A \in \mathbb{R}^{\ell\times\ell}$, $B \in \mathbb{R}^{m\times m}$, $C \in \mathbb{R}^{n\times n}$, I_m is the $m \times m$ identity matrix, the symbol "$\otimes$" denotes the Kronecker product, and matrix T is assumed to be large, sparse, and nonsingular.

The matrix T arises in a finite difference discretization of the following three-dimensional constant-coefficient partial differential equation:

$$\left[-\boldsymbol{a} \cdot (\nabla \star \nabla) + \boldsymbol{b} \cdot \nabla + c\right] u(x, y, z) = g(x, y, z) \text{ in } \Omega, \quad u(x, y, z) = 0 \text{ on } \partial\Omega, \tag{2}$$

where $\Omega = (0, 1) \times (0, 1) \times (0, 1)$, $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, $c \in \mathbb{R}$, and the symbol "$\star$" denotes elementwise products. If $\boldsymbol{a} = (1, 1, 1)$, then $\boldsymbol{a} \cdot (\nabla \star \nabla) = \Delta$. With regard to efficient numerical methods for linear systems $T\boldsymbol{x} = \boldsymbol{f}$, see [2, 3].

Recently, Ohashi and Sogabe [1] proposed the Lanczos bidiagonalization method over tensor space to compute the maximum/minimum singular values of T in (1). The method was obtained from reconstructing the Lanczos bidiagonalization method, which is widely known as the Golub-Kahan bidiagonalization method [5], over tensor space. The method over tensor space succeeded in avoiding large memory usage due to the two-fold Kronecker product structure of T, and was thus found to be practical. However, there is still room for improvement in terms of the number of iterations. In fact, as shown by the numerical result of Ohashi and Sogabe [1] —see Fig. 5.2(a)— there is case in which the number of iterations required for the minimum singular value is about 5 times greater than that for the maximum singular value.

***Corresponding Author: A. Ohashi:** College of Science & Engineering, Ritsumeikan University, Japan, E-mail: a-ohashi@fc.ritsumei.ac.jp
T. Sogabe: Graduate School of Engineering, Nagoya University, Japan, E-mail: sogabe@na.nuap.nagoya-u.ac.jp

This paper proposes an efficient, with respect to the number of iterations, computation of the minimum singular value of the tensor sum $\mathcal{T}$. The key approach is to reconstruct the Lanczos bidiagonalization method over tensor space for $\mathcal{T}^{-1}$ not $\mathcal{T}$. The resulting algorithm requires two linear systems of the form $\mathcal{T}\boldsymbol{x}_k = \boldsymbol{b}_k$ and $\mathcal{T}^{\mathrm{T}}\boldsymbol{y}_k = \boldsymbol{c}_k$ at each iteration step k, where the superscript T represents the transpose. In order to solve such linear systems with iteration-dependent multiple right-hand-sides, some direct method, e.g., the LU decomposition, may be a suitable method. However, a direct method requires a huge memory of $O(n^6)$ when $\ell = m = n$ in (1), due to loss of the Kronecker product structure. This motivates the consideration of a structure-preserving direct approach to solving $\mathcal{T}\boldsymbol{x}_k = \boldsymbol{b}_k$ and $\mathcal{T}^{\mathrm{T}}\boldsymbol{y}_k = \boldsymbol{c}_k$ using the eigenvalue decomposition and the Schur decomposition of small matrices A, B, and C. This new structure-preserving approach will reduce $O(n^6)$ to $O(n^3)$.

The rest of this paper is organized as follows. Section 2 describes the Lanczos bidiagonalization method over tensor space for $\mathcal{T}$. Section 3 describes the reconstruction of a Lanczos bidiagonalization method over tensor space for $\mathcal{T}^{-1}$ and proposes two implementations of matrix-vector multiplications of $\mathcal{T}^{-1}$ and $(\mathcal{T}^{-1})^{\mathrm{T}}$. Section 4 presents the numerical experiments, while Section 5 provides the concluding remarks. Throughout of this paper, notations of tensor operations, e.g., n-mode products and tensor norms, are the same as in the paper by Kolda and Bader [4]. Moreover, small bold letters $\boldsymbol{w}$, $\boldsymbol{x}$, $\boldsymbol{y}$ are used for vectors, capital letters A, B, C are used for matrices, and calligraphic bold letters $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{R}$ are used for third-order tensors.

2 The Lanczos bidiagonalization method over tensor space for $\mathcal{T}$

In this section, the Lanczos bidiagonalization method over tensor space [1] is briefly explained. The algorithm is the reconstruction of the Lanczos bidiagonalization method [5] from vector space to tensor space and is shown in Algorithm 1.

Algorithm 1 The Lanczos bidiagonalization method over tensor space for $\mathcal{T}$

Choose an initial tensor $\mathcal{P}_0 \in \mathbb{R}^{\ell\times m\times n}$ such that $\|\mathcal{P}_0\| = 1$.
$\mathcal{Q}_0 := \mathcal{P}_0 \times_1 A + \mathcal{P}_0 \times_2 B + \mathcal{P}_0 \times_3 C$;
$\alpha_0 := \|\mathcal{Q}_0\|$;
$\mathcal{Q}_0 := \mathcal{Q}_0/\alpha_0$;
for $i = 0, 1, \ldots, k$ **do**
    $\mathcal{R}_i := \mathcal{Q}_i \times_1 A^{\mathrm{T}} + \mathcal{Q}_i \times_2 B^{\mathrm{T}} + \mathcal{Q}_i \times_3 C^{\mathrm{T}} - \alpha_i\mathcal{P}_i$;
    $\beta_i := \|\mathcal{R}_i\|$;
    $\mathcal{P}_{i+1} := \mathcal{R}_i/\beta_i$;
    $\mathcal{Q}_{i+1} := \mathcal{P}_{i+1} \times_1 A + \mathcal{P}_{i+1} \times_2 B + \mathcal{P}_{i+1} \times_3 C - \beta_i\mathcal{Q}_i$;
    $\alpha_{i+1} := \|\mathcal{Q}_{i+1}\|$;
    $\mathcal{Q}_{i+1} := \mathcal{Q}_{i+1}/\alpha_{i+1}$;
end for

Using α_i and β_i obtained from the k steps of Algorithm 1, the upper bidiagonal matrix D_k is generated as follows:

$$D_k := \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{k-1} & \beta_{k-1} \\ & & & & \alpha_k \end{pmatrix} \in \mathbb{R}^{k\times k}. \tag{3}$$

If the relative residual norms $\|\mathcal{R}_k\| \cdot |\boldsymbol{e}_k^{\mathrm{T}} \boldsymbol{u}_{\mathrm{M}}^{(D_k)}|$ and $\|\mathcal{R}_k\| \cdot |\boldsymbol{e}_k^{\mathrm{T}} \boldsymbol{u}_{\mathrm{m}}^{(D_k)}|$ are sufficiently small, the maximum/minimum singular values of D_k, referred to as $\sigma_{\mathrm{M}}^{(D_k)}$ and $\sigma_{\mathrm{m}}^{(D_k)}$, approximate the maximum/minimum singular values of T, respectively. Here $\boldsymbol{e}_k^{\mathrm{T}}$ denotes the transpose of the k-dimensional k-th canonical vector, and $\boldsymbol{u}_{\mathrm{M}}^{(D_k)}$ and $\boldsymbol{u}_{\mathrm{m}}^{(D_k)}$ are the left singular vectors of D_k corresponding to $\sigma_{\mathrm{M}}^{(D_k)}$ and $\sigma_{\mathrm{m}}^{(D_k)}$, respectively.

3 The Lanczos bidiagonalization method over tensor space for T^{-1}

Here, the reconstruction of a Lanczos bidiagonalization method for T^{-1} over tensor space is considered. The reconstructed algorithm is shown in Algorithm 2, which is obtained from the modifications of lines 2,6, and 9 in Algorithm 1.

Algorithm 2 A Lanczos bidiagonalization method over tensor space for T^{-1}

1: Choose an initial tensor $\mathcal{P}_0 \in \mathbb{R}^{\ell \times m \times n}$ such that $\|\mathcal{P}_0\| = 1$.
2: $\mathcal{Q}_0 := \mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_0)$;
3: $\alpha_0 := \|\mathcal{Q}_0\|$;
4: $\mathcal{Q}_0 := \mathcal{Q}_0/\alpha_0$;
5: **for** $i = 0, 1, \ldots, k$ **do**
6: $\quad \mathcal{R}_i := \mathrm{vec}^{-1}((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i) - \alpha_i \mathcal{P}_i$;
7: $\quad \beta_i := \|\mathcal{R}_i\|$;
8: $\quad \mathcal{P}_{i+1} := \mathcal{R}_i/\beta_i$;
9: $\quad \mathcal{Q}_{i+1} := \mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_{i+1}) - \beta_i \mathcal{Q}_i$;
10: $\quad \alpha_{i+1} := \|\mathcal{Q}_{i+1}\|$;
11: $\quad \mathcal{Q}_{i+1} := \mathcal{Q}_{i+1}/\alpha_{i+1}$;
12: **end for**

Note that the approximate minimum singular value of T is computed from the bidiagonal matrix D_k in (3) generated in Algorithm 2. In what follows, we give the detail. Using α_i and β_i obtained from Algorithm 2, D_k in (3) is generated. Let $\sigma_{\mathrm{M}}^{(D_k)}$ and $\boldsymbol{u}_{\mathrm{M}}^{(D_k)}$ be the maximum singular value of D_k and the corresponding left singular vector, respectively. If the relative residual norm $\|\mathcal{R}_k\| \cdot |\boldsymbol{e}_k^{\mathrm{T}} \boldsymbol{u}_{\mathrm{M}}^{(D_k)}|$ is sufficiently small, then $1/\sigma_{\mathrm{M}}^{(D_k)}$ approximates the minimum singular value of T, i.e., $\sigma_{\mathrm{m}}^{(T)} \approx 1/\sigma_{\mathrm{M}}^{(D_k)}$.

In order to run Algorithm 2 efficiently, with respect to memory usage, implementations of $\mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_{i+1})$ and $\mathrm{vec}^{-1}((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i)$ are required. Therefore, T^{-1} is implicitly generated from decompositions of much smaller matrices A, B, and C, rather than using T to compute the matrix-vector multiplications. In order to implicitly generate T^{-1}, the following two decompositions are considered: first, the eigenvalue decomposition shown in Section 3.1; second, the Schur decomposition shown in Section 3.2.

3.1 On the computations of $\mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_{i+1})$ and $\mathrm{vec}^{-1}((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i)$ using the right and left eigenvectors of matrices A, B, and C

In this subsection, an implementation over tensor space using the eigenvalues and the right and left eigenvectors of A, B, and C is described. Here, we assume that A, B, and C are diagonalizable since T must have the eigenvalue decomposition.

Let Λ be the diagonal matrix whose diagonal elements are the eigenvalues of T, let X and Y be the matrices whose column vectors are the right and left vectors of T, respectively, and let $D := Y^{\mathrm{H}}X$, where the superscript H represents the conjugate transpose. Then, T^{-1} can be decomposed into

$$T^{-1} = X\Lambda^{-1}D^{-1}Y^{\mathrm{H}}. \tag{4}$$

X, Y, D, and Λ are obtained from computations for A, B, and C below. First, using the definition of T, we have

$$X = X_{(C)} \otimes X_{(B)} \otimes X_{(A)}, \quad Y = Y_{(C)} \otimes Y_{(B)} \otimes Y_{(A)}, \tag{5}$$

where $X_{(A)}$ and $Y_{(A)}$ are the matrices whose column vectors are the right and left eigenvectors of A, respectively, and $X_{(B)}$, $X_{(C)}$, $Y_{(B)}$, and $Y_{(C)}$ are also the matrices defined in the same way as $X_{(A)}$ and $Y_{(A)}$. Second, from (5) and the definition of D, it follows that

$$D = Y_{(C)}^{\mathrm{H}} X_{(C)} \otimes Y_{(B)}^{\mathrm{H}} X_{(B)} \otimes Y_{(A)}^{\mathrm{H}} X_{(A)}.$$

Here, let $D_{(A)} := Y_{(A)}^{\mathrm{H}} X_{(A)}$, $D_{(B)} := Y_{(B)}^{\mathrm{H}} X_{(B)}$, and $D_{(C)} := Y_{(C)}^{\mathrm{H}} X_{(C)}$. Then, we have

$$D^{-1} = D_{(C)}^{-1} \otimes D_{(B)}^{-1} \otimes D_{(A)}^{-1}. \tag{6}$$

This means that the inverse of the large matrix D^{-1} is computed by the inverses of the small matrices $D_{(A)}$, $D_{(B)}$, and $D_{(C)}$. Moreover, it can be seen that $D_{(A)}$, $D_{(B)}$, and $D_{(C)}$ are block diagonal matrices from the properties of the right and left eigenvectors. Finally, let $\lambda_i^{(A)}$, $\lambda_j^{(B)}$, and $\lambda_k^{(C)}$ be the eigenvalues of A, B, and C, respectively, where $i = 1, 2, \ldots, \ell$, $j = 1, 2, \ldots, m$, and $k = 1, 2, \ldots, n$. Then, $\Lambda = \mathrm{diag}(\mathrm{vec}(\hat{\mathcal{L}}))$, where $(\hat{\mathcal{L}})_{ijk} = \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_k^{(C)}$ and $\mathrm{diag}(\boldsymbol{d})$ denotes a diagonal matrix whose diagonals are elements of a vector $\boldsymbol{d}$.

Using (4)–(6),

$$\begin{aligned} T^{-1} &= \left(X_{(C)} \otimes X_{(B)} \otimes X_{(A)}\right) \Lambda^{-1} \left(D_{(C)}^{-1} \otimes D_{(B)}^{-1} \otimes D_{(A)}^{-1}\right) \left(Y_{(C)}^{\mathrm{H}} \otimes Y_{(B)}^{\mathrm{H}} \otimes Y_{(A)}^{\mathrm{H}}\right) \\ &= \left(X_{(C)} \otimes X_{(B)} \otimes X_{(A)}\right) \Lambda^{-1} \left(D_{(C)}^{-1} Y_{(C)}^{\mathrm{H}} \otimes D_{(B)}^{-1} Y_{(B)}^{\mathrm{H}} \otimes D_{(A)}^{-1} Y_{(A)}^{\mathrm{H}}\right), \end{aligned} \tag{7}$$

$$\begin{aligned} (T^{-1})^{\mathrm{T}} &= \left(Y_{(C)} \otimes Y_{(B)} \otimes Y_{(A)}\right) \left((D_{(C)}^{-1})^{\mathrm{H}} \otimes (D_{(B)}^{-1})^{\mathrm{H}} \otimes (D_{(A)}^{-1})^{\mathrm{H}}\right) (\Lambda^{-1})^{\mathrm{H}} \left(X_{(C)}^{\mathrm{H}} \otimes X_{(B)}^{\mathrm{H}} \otimes X_{(A)}^{\mathrm{H}}\right) \\ &= \left(Y_{(C)} (D_{(C)}^{-1})^{\mathrm{H}} \otimes Y_{(B)} (D_{(B)}^{-1})^{\mathrm{H}} \otimes Y_{(A)} (D_{(A)}^{-1})^{\mathrm{H}}\right) (\Lambda^{-1})^{\mathrm{H}} \left(X_{(C)}^{\mathrm{H}} \otimes X_{(B)}^{\mathrm{H}} \otimes X_{(A)}^{\mathrm{H}}\right). \end{aligned} \tag{8}$$

Here, let $\mathcal{L} := \mathrm{vec}^{-1}(\mathrm{diag}(\Lambda^{-1}))$, where $\mathrm{diag}(\Lambda^{-1})$ is the vector whose i-th element is $(\Lambda^{-1})_{i,i}$, i.e., $(\mathcal{L})_{ijk} = 1/(\lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_k^{(C)})$. From the definition of $\mathcal{L}$, the mode products, (7), and (8), the following implementations of $\mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_{i+1})$ and $\mathrm{vec}^{-1}((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i)$ are obtained:

Algorithm 3-1 Computation of $\mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_{i+1})$ using the eigendecomposition

1: $\mathcal{Z} = \mathcal{L} * \left(\mathcal{P}_{i+1} \times_1 D_{(A)}^{-1} Y_{(A)}^{\mathrm{H}} \times_2 D_{(B)}^{-1} Y_{(B)}^{\mathrm{H}} \times_3 D_{(C)}^{-1} Y_{(C)}^{\mathrm{H}}\right)$
2: $\mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_{i+1}) = \mathcal{Z} \times_1 X_{(A)} \times_2 X_{(B)} \times_3 X_{(C)}$

Algorithm 3-2 Computation of $\mathrm{vec}^{-1}((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i)$ using the eigendecomposition

1: $\mathcal{Z} = \overline{\mathcal{L}} * \left(\mathcal{Q}_i \times_1 X_{(A)}^{\mathrm{H}} \times_2 X_{(B)}^{\mathrm{H}} \times_3 X_{(C)}^{\mathrm{H}}\right)$
2: $\mathrm{vec}^{-1}((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i) = \mathcal{Z} \times_1 Y_{(A)} (D_{(A)}^{-1})^{\mathrm{H}} \times_2 Y_{(B)} (D_{(B)}^{-1})^{\mathrm{H}} \times_3 Y_{(C)} (D_{(C)}^{-1})^{\mathrm{H}}$

The symbol "$*$" is the elementwise product, i.e., $(\mathcal{A} * \mathcal{B})_{ijk} = a_{ijk} b_{ijk}$ for $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{\ell \times m \times n}$, and $\overline{\mathcal{L}}$ denotes the elementwise complex conjugate of $\mathcal{L}$, i.e., $(\overline{\mathcal{A}})_{ijk} = \overline{a}_{ijk}$.

Using Algorithm 2 with Algorithms 3-1 and 3-2, only matrices much smaller than T, $X_{(A)}$, $X_{(B)}$, $X_{(C)}$, $Y_{(A)}$, $Y_{(B)}$, $Y_{(C)}$, $D_{(A)}$, $D_{(B)}$, and $D_{(C)}$, and $\ell \times m \times n$ tensors, $\mathcal{P}_i$, $\mathcal{Q}_i$, $\mathcal{L}$, and $\mathcal{Z}$ are required. Therefore, the memory requirement is of $O(n^3)$ when $\ell = m = n$.

3.2 On the computations of $\mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_{i+1})$ and $\mathrm{vec}^{-1}((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i)$ using the Schur decomposition of matrices A, B, and C

In this subsection, another implementation over tensor space using the complex Schur decomposition of A, B, and C is described. Here, Li et al. [6] proposed the solver of $T\boldsymbol{x} = \boldsymbol{b}$ based on the Schur decomposition, where $\boldsymbol{x}$ and $\boldsymbol{b}$ are unknown and given vectors, respectively. Note that in order to implement Algorithm 2, solvers of the forms $T\boldsymbol{x}_k = \boldsymbol{b}_k$ and $T^{\mathrm{T}}\boldsymbol{y}_k = \boldsymbol{c}_k$ are required at each iteration step k.

Using the complex Schur decomposition, T can be decomposed into $T = Q_{(T)}R_{(T)}Q_{(T)}^{\mathrm{H}}$, where $Q_{(T)}$ is the unitary matrix and $R_{(T)}$ is the upper triangular matrix. Furthermore, A, B, and C are also decomposed into $A = Q_{(A)}R_{(A)}Q_{(A)}^{\mathrm{H}}$, $B = Q_{(B)}R_{(B)}Q_{(B)}^{\mathrm{H}}$, and $C = Q_{(C)}R_{(C)}Q_{(C)}^{\mathrm{H}}$ via the complex Schur decomposition. Then, $Q_{(T)}$ and $R_{(T)}$ are obtained from $Q_{(A)}$, $Q_{(B)}$, $Q_{(C)}$, $R_{(A)}$, $R_{(B)}$, and $R_{(C)}$ as follows:

$$\begin{cases} Q_{(T)} = Q_{(C)} \otimes Q_{(B)} \otimes Q_{(A)}, \\ R_{(T)} = I_n \otimes I_m \otimes R_{(A)} + I_n \otimes R_{(B)} \otimes I_\ell + R_{(C)} \otimes I_m \otimes I_\ell. \end{cases} \tag{9}$$

Substituting (9) into $T^{-1}\boldsymbol{p}_{i+1} = (Q_{(T)}R_{(T)}^{-1}Q_{(T)}^{\mathrm{H}})\boldsymbol{p}_{i+1}$ yields

$$\boldsymbol{w} = \left(Q_{(C)}^{\mathrm{H}} \otimes Q_{(B)}^{\mathrm{H}} \otimes Q_{(A)}^{\mathrm{H}}\right)\boldsymbol{p}_{i+1}, \tag{10}$$

$$\boldsymbol{x} = R_{(T)}^{-1}\boldsymbol{w}, \tag{11}$$

$$T^{-1}\boldsymbol{p}_{i+1} = \left(Q_{(C)} \otimes Q_{(B)} \otimes Q_{(A)}\right)\boldsymbol{x}. \tag{12}$$

Here, let $\mathcal{W} = \mathrm{vec}^{-1}(\boldsymbol{w})$ and $\mathcal{X} = \mathrm{vec}^{-1}(\boldsymbol{x})$. Then, applying vec^{-1} operator to (10)–(12), the following algorithm is obtained from the relations between the tensor product and n-mode products.

Algorithm 4-1(A) Computation of $\mathrm{vec}^{-1}\left(T^{-1}\boldsymbol{p}_{i+1}\right)$ using the Schur decomposition

1: $\mathcal{W} = \mathcal{P}_{i+1} \times_1 Q_{(A)}^{\mathrm{H}} \times_2 Q_{(B)}^{\mathrm{H}} \times_3 Q_{(C)}^{\mathrm{H}}$
2: $\mathcal{X} = \mathrm{vec}^{-1}\left(R_{(T)}^{-1}\boldsymbol{w}\right)$
3: $\mathrm{vec}^{-1}\left(T^{-1}\boldsymbol{p}_{i+1}\right) = \mathcal{X} \times_1 Q_{(A)} \times_2 Q_{(B)} \times_3 Q_{(C)}$

As can be seen from Algorithm 4-1(A), $\mathcal{X}$ is computed by utilizing the vec^{-1} operator for the solution of the linear system of the form $R_{(T)}\boldsymbol{x} = \boldsymbol{w}$. Assuming that $\ell = m = n$ and A, B, and C are dense, the memory requirement of $R_{(T)}$ is of $O(n^6)$.

Therefore, it is necessary to implement the second line of Algorithm 4-1(A) using not the large matrix $R_{(T)}$, but the smaller matrices $R_{(A)}$, $R_{(B)}$, and $R_{(C)}$ to decrease the memory requirement. Now, that the focus is placed on the structure of $R_{(T)}$, i.e., the $mn \times mn$ block upper triangular matrix when $R_{(T)}$ is partitioned into $\ell \times \ell$ blocks. Using this structure, an implementation that is based on the backward substitution is described.

In the beginning, $\mathcal{X}$ is set to the zero tensor $O_{\ell \times m \times n}$. Second, $\boldsymbol{x}_{:mn}$, which denotes the (m, n) mode-1 (column) fibers of $\mathcal{X}$ is computed by solving the following linear system via backward substitution:

$$\boldsymbol{x}_{:mn} = \left(R_{(A)} + r_{mm}^{(B)}I_\ell + r_{nn}^{(C)}I_\ell\right)^{-1}\boldsymbol{w}_{:mn},$$

where $r_{mm}^{(B)}$ represents the (m, m) element of $R_{(B)}$. Third, compute

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{w}_{:(m-1)n} - \left(\boldsymbol{e}_n^{(n)} \otimes \boldsymbol{r}_{m-1}^{(B)} \otimes I_\ell + \boldsymbol{r}_n^{(C)} \otimes \boldsymbol{e}_{m-1}^{(m)} \otimes I_\ell\right)\boldsymbol{x}, \\ \boldsymbol{x}_{:(m-1)n} &= \left(R_{(A)} + r_{(m-1)(m-1)}^{(B)}I_\ell + r_{nn}^{(C)}I_\ell\right)^{-1}\boldsymbol{y}, \end{aligned} \tag{13}$$

where $\boldsymbol{e}_{m-1}^{(m)}$ and $\boldsymbol{r}_{m-1}^{(B)}$ denote the $(m-1)$-th canonical vector of $1 \times m$ and the $(m-1)$-th row vector of $R_{(B)}$, respectively. Using the mode products, (13) is rewritten as

$$\boldsymbol{y} = \boldsymbol{w}_{:j_2j_3} - \left(\mathcal{X} \times_2 \boldsymbol{r}_{j_2}^{(B)} \times_3 \boldsymbol{e}_{j_3}^{(n)} + \mathcal{X} \times_2 \boldsymbol{e}_{j_2}^{(m)} \times_3 \boldsymbol{r}_{j_3}^{(C)}\right),$$

since the $\ell \times 1 \times n$ tensor is obtained from the 2-mode product of the $\mathcal{X}$ and $m \times 1$ vectors, and the $\ell \times 1 \times 1$ tensor (vector) is obtained from the 3-mode product of the obtained tensor and $n \times 1$ vectors. Finally, the rest of fibers $\boldsymbol{x}_{:(m-2)n}, \boldsymbol{x}_{:(m-3)n}, \ldots, \boldsymbol{x}_{:1n}, \boldsymbol{x}_{:m(n-1)}, \boldsymbol{x}_{:(m-1)(n-1)}, \ldots, \boldsymbol{x}_{:11}$ are similarly computed in order.

The backward substitution over the tensor space for $\mathcal{X} = \mathrm{vec}^{-1}(R_{(T)}^{-1}\boldsymbol{w})$ is summarized in Algorithm 4-1(B). Here, the function $\tilde{\boldsymbol{x}} = \mathrm{backwardsubstitution}(U, \boldsymbol{b})$ of the 5-th line in Algorithm 4-1(B) denotes the backward substitution of $U\tilde{\boldsymbol{x}} = \boldsymbol{b}$ with an upper triangular matrix U and a vector $\boldsymbol{b}$. Moreover, $n : -1 : 1$ means to decrement a variable by 1 from n to 1.

Algorithm 4-1(B) The backward substitution over tensor space for $\mathcal{X} = \mathrm{vec}^{-1}(R_{(T)}^{-1}\boldsymbol{w})$

1: $\mathcal{X} := O_{\ell \times m \times n}$;
2: **for** $j_3 = n : -1 : 1$ **do**
3: **for** $j_2 = m : -1 : 1$ **do**
4: $\boldsymbol{y} = \boldsymbol{w}_{:j_2 j_3} - \left(\mathcal{X} \times_2 \boldsymbol{r}_{j_2}^{(B)} \times_3 \boldsymbol{e}_{j_3}^{(n)} + \mathcal{X} \times_2 \boldsymbol{e}_{j_2}^{(m)} \times_3 \boldsymbol{r}_{j_3}^{(C)}\right)$;
5: $\boldsymbol{x}_{:j_2 j_3} = \mathrm{backwardsubstitution}\left(R_{(A)} + r_{j_2,j_2}^{(B)} I_\ell + r_{j_3,j_3}^{(C)} I_\ell, \boldsymbol{y}\right)$;
6: **end for**
7: **end for**

Now, $\mathrm{vec}^{-1}\left(T^{-1}\boldsymbol{p}_{i+1}\right)$ can be implemented using Algorithm 4-1(A) with Algorithm 4-1(B).

From here, $(T^{-1})^{\mathrm{T}}\boldsymbol{q}_i = (Q_{(T)}R_{(T)}^{-1}Q_{(T)}^{\mathrm{H}})^{\mathrm{H}}\boldsymbol{q}_i$ can be considered. Substituting (9) into $(T^{-1})^{\mathrm{T}}\boldsymbol{q}_i = (Q_{(T)}R_{(T)}^{-1}Q_{(T)}^{\mathrm{H}})^{\mathrm{H}}\boldsymbol{q}_i$ yields

$$\boldsymbol{w} = \left(Q_{(C)}^{\mathrm{H}} \otimes Q_{(B)}^{\mathrm{H}} \otimes Q_{(A)}^{\mathrm{H}}\right)\boldsymbol{q}_i, \tag{14}$$

$$\boldsymbol{x} = \left(R_{(T)}^{-1}\right)^{\mathrm{H}}\boldsymbol{w}, \tag{15}$$

$$\left(T^{-1}\right)^{\mathrm{T}}\boldsymbol{q}_i = \left(Q_{(C)} \otimes Q_{(B)} \otimes Q_{(A)}\right)\boldsymbol{x}. \tag{16}$$

Applying vec^{-1} operator to (14)–(16), the following algorithm is obtained.

Algorithm 4-2(A) Computation of $\mathrm{vec}^{-1}\left(\left(T^{-1}\right)^{\mathrm{T}}\boldsymbol{q}_i\right)$ using the Schur decomposition

1: $\mathcal{W} = \Omega_i \times_1 Q_{(A)}^{\mathrm{H}} \times_2 Q_{(B)}^{\mathrm{H}} \times_3 Q_{(C)}^{\mathrm{H}}$
2: $\mathcal{X} = \mathrm{vec}^{-1}\left(\left(R_{(T)}^{-1}\right)^{\mathrm{H}}\boldsymbol{w}\right)$
3: $\mathrm{vec}^{-1}\left(\left(T^{-1}\right)^{\mathrm{T}}\boldsymbol{q}_i\right) = \mathcal{X} \times_1 Q_{(A)} \times_2 Q_{(B)} \times_3 Q_{(C)}$

$\mathcal{X}$ is computed by applying the vec^{-1} operator to the solution of the linear system with the lower triangular matrix $R_{(T)}^{\mathrm{H}}$. Here, let $\tilde{\boldsymbol{x}} = \mathrm{forwardsubstitution}(L, \boldsymbol{b})$ be the function of forward substitution for a linear system $L\tilde{\boldsymbol{x}} = \boldsymbol{b}$ with a lower triangular matrix L and vector $\boldsymbol{b}$. The forward substitution over tensor space for $\mathcal{X} = \mathrm{vec}^{-1}((R_{(T)}^{-1})^{\mathrm{H}}\boldsymbol{w})$ is displayed in Algorithm 4-2(B).

Now, $\mathrm{vec}^{-1}\left((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i\right)$ can be implemented using Algorithm 4-2(A) with Algorithm 4-2(B).

Therefore, $\mathrm{vec}^{-1}(T^{-1}\boldsymbol{p}_{i+1})$ and $\mathrm{vec}^{-1}((T^{-1})^{\mathrm{T}}\boldsymbol{q}_i)$ are computed employing only smaller matrices $R_{(A)}$, $R_{(B)}$, $R_{(C)}$, $Q_{(A)}$, $Q_{(B)}$, $Q_{(C)}$ and tensors $\mathcal{W}$ and $\mathcal{X}$, using Algorithms 4-1(A) and 4-2(A) with Algorithms 4-1(B) and 4-2(B), respectively. Supposing that $\ell = m = n$, the memory requirement is of $O(n^3)$.

If the real Schur decomposition is used instead of the complex Schur decomposition, the diagonal blocks of upper triangular matrices $R_{(A)}$, $R_{(B)}$, and $R_{(C)}$ are either 1×1 or 2×2 matrices. Therefore, using the real Schur decomposition, the ordinary back substitution and forward substitution are modified in the 5-th lines of Algorithms 4-1(B) and 4-2(B) to solve the linear systems for the above upper triangular matrices.

Algorithm 4-2(B) The forward substitution over tensor space for $\mathcal{X} = \mathrm{vec}^{-1}((R_{(T)}^{-1})^{\mathrm{H}}\boldsymbol{w})$

1: $\mathcal{X} := O_{\ell\times m\times n}$;
2: **for** $j_3 = 1 : n$ **do**
3: **for** $j_2 = 1 : m$ **do**
4: $\boldsymbol{y} = \boldsymbol{w}_{:j_2j_3} - \left(\mathcal{X} \times_2 \boldsymbol{r}_{j_2}^{(B)} \times_3 \boldsymbol{e}_{j_3}^{(n)} + \mathcal{X} \times_2 \boldsymbol{e}_{j_2}^{(m)} \times_3 \boldsymbol{r}_{j_3}^{(C)}\right)$;
5: $\boldsymbol{x}_{:j_2j_3}$:= forwardsubstitution $\left(R_{(A)}^{\mathrm{H}} + r_{j_2,j_2}^{(B)} I_\ell + r_{j_3,j_3}^{(C)} I_\ell, \boldsymbol{y}\right)$;
6: **end for**
7: **end for**

4 Numerical examples

This section gives the results of the numerical experiments using Algorithm 2 (shown in Section 3) to compute the minimum singular values of the test matrices T. For comparison, the results of the numerical experiments using Algorithm 1 [1], which is the conventional method of Algorithm 2, are also given. All computations were carried out using MATLAB R2018a version on Windows 7 on a workstation with two Xeon processors (2.4GHz). All the initial guesses $\mathcal{P}_0$, which are used for Algorithms 1 and 2, are tensors with random numbers which are uniformly distributed in the interval (0,1). The stopping criteria of Algorithms 1 and 2 is $\|\mathcal{R}_k\| \cdot |\boldsymbol{e}_k^{\mathrm{T}}\boldsymbol{u}_{\mathrm{M}}^{(D_k)}| < 10^{-10}$.

4.1 Numerical results

In this subsection, the average numbers of iterations, the average required times, and the accuracies of the approximate minimum singular values of T by Algorithms 1 and 2 are compared. Algorithm 2 was implemented using Algorithms 3-1 and 3-2 or Algorithms 4-1(A) and 4-2(A) with Algorithms 4-1(B) and 4-2(B).

In Section 4.1.1, the first test matrices T in (1) are the same as the test matrices of the numerical examples by Ohashi and Sogabe [1], which were obtained from a 7-point central difference discretization of the PDE in (2) over an $(n+1)\times(n+1)\times(n+1)$ grid. The test matrices T in (1) are generated from

$$A = \frac{1}{h^2}a_1M_1 + \frac{1}{2h}b_1M_2 + \frac{1}{3}cI_n, \tag{17}$$

$$B = \frac{1}{h^2}a_2M_1 + \frac{1}{2h}b_2M_2 + \frac{1}{3}cI_n, \tag{18}$$

$$C = \frac{1}{h^2}a_3M_1 + \frac{1}{2h}b_3M_2 + \frac{1}{3}cI_n, \tag{19}$$

where a_i and b_i $(i = 1, 2, 3)$ correspond to the i-th elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ in (2), respectively. Here, $h = 1/(n+1)$, and M_1 and M_2 are given below.

$$M_1 = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{n\times n}, M_2 = \begin{pmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 0 \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

In Section 4.1.2, the second test matrices T in (1) are made from random matrices A, B, and C. The random numbers in the matrices are uniformly distributed in the interval (0,1).

In all of the tables in Section 4.1.1, we show the average number of iterations and the average required CPU time computed by 10 runs of our algorithms and the conventional algorithm using 10 types of random initial tensors. In Section 4.1.2, we use 10 types of random matrices A, B, and C as test matrices, and we show the average number of iterations and the average required CPU time computed by 10 runs of our algorithms and the conventional algorithm using 10 types of random initial tensors.

4.1.1 Numerical results for T composed of A, B, C in (17)–(19)

The first test matrices T were obtained from the following two sets of parameters: first, $\boldsymbol{a} = (100, 100, 100)$, $\boldsymbol{b} = (1, 1, 1)$, and $c = 1$; second, $\boldsymbol{a} = (1, 1, 1)$, $\boldsymbol{b} = (100, 100, 100)$, and $c = 1$. Note that T with the first set of parameters has high symmetry from (17)–(19) and that T with the second set has low symmetry, as can be seen from the definitions of A, B, and C in (17)–(19).

First, in Tables 1–3, numerical results for T with high symmetry, which are obtained from the first set of parameters, are displayed. Tables 1 and 2 summarize the average number of iterations and the average required time by Algorithms 1 and 2. Table 3 summarizes the following relative errors of the approximate minimum singular values $1/\sigma_{\mathrm{M}}^{(D_k)}$ by Algorithms 1 and 2:

$$\frac{\left|\sigma_{\mathrm{m}}^{(T)} - (1/\sigma_{\mathrm{M}}^{(D_k)})\right|}{\sigma_{\mathrm{m}}^{(T)}}, \tag{20}$$

where $\sigma_{\mathrm{m}}^{(T)}$ is computed by the MATLAB function `svd`.

Table 1: Numbers of iterations of Algorithms 1 and 2 for high symmetry T in (1). T is generated from $\boldsymbol{a} = (100, 100, 100)$, $\boldsymbol{b} = (1, 1, 1)$, and $c = 1$ in (17)–(19), and the size of T is $n^3 \times n^3$. The symbol "—" means that the algorithm did not converge to the minimum singular values of T.

n	15	20	25	30	35
Alg. 1 [1]	410.5	713.4	1094.3	1545.6	—
Alg. 2 with Algs. 3-1 and 3-2	6.0	6.0	6.0	6.0	6.0
Alg. 2 with Algs. 4-1(A) and 4-2(A)	6.0	6.0	6.0	6.0	6.0

Table 2: Required time (sec) of Algorithms 1 and 2 for high symmetry T in (1). T is generated from $\boldsymbol{a} = (100, 100, 100)$, $\boldsymbol{b} = (1, 1, 1)$, and $c = 1$ in (17)–(19), and the size of T is $n^3 \times n^3$. The symbol "—" means that the algorithm did not converge to the minimum singular values of T.

n	15	20	25	30	35
Alg. 1 [1]	7.9	36.3	125.8	391.1	—
Alg. 2 with Algs. 3-1 and 3-2	0.1	0.2	0.4	0.6	1.0
Alg. 2 with Algs. 4-1(A) and 4-2(A)	4.2	12.6	30.1	62.6	116.6

Table 3: Relative errors between the approximate minimum singular value of T in Table 1 and the minimum singular value of T computed by `svd`. The superscript † indicates that the relative error in (20) equals 0. The symbol "—" means that the algorithm did not converge to the minimum singular values of T.

n	15	20	25	30	35
Alg. 1 [1]	8.2e-12	7.1e-15	1.3e-14	9.5e-15	—
Alg. 2 with Algs. 3-1 and 3-2	1.0e-14	1.0e-16†	6.8e-15	3.4e-14	8.8e-14
Alg. 2 with Algs. 4-1(A) and 4-2(A)	1.0e-16†	1.0e-16†	1.0e-16†	1.0e-16†	1.0e-16†

From Table 1, both Algorithm 2 with Algorithms 3-1 and 3-2 and Algorithm 2 with Algorithms 4-1(A) and 4-2(A) required about 6 iterations to satisfy the stopping criteria. Moreover, the accuracies of the obtained approximate minimum singular values of T were about 10^{-14} from Table 3. Namely, for T with high symmetry,

both Algorithm 2 with Algorithms 3-1 and 3-2 and Algorithm 2 with Algorithms 4-1(A) and 4-2(A) seem stable, since the numbers of iterations of Algorithm 2 tend to not depend on the size of T from Table 1. Comparing the required time in Table 2, Algorithm 2 with Algorithms 3-1 and 3-2 required the least time among all of the algorithms.

Next, the average number of iterations and the average required time by Algorithms 1 and 2 for T with low symmetry are displayed in Tables 4 and 5, respectively. The relative errors between the approximate minimum singular values of T in Table 4 and the minimum singular values of T computed by the MATLAB function `svd` are shown in Table 6.

Table 4: Numbers of iterations of Algorithms 1 and 2 for T with low symmetry. T were generated from $\boldsymbol{a} = (1, 1, 1)$, $\boldsymbol{b} = (100, 100, 100)$, and $c = 1$ in (17)–(19), and the size of T is $n^3 \times n^3$. The symbol "—" means that the algorithm did not converge to the minimum singular values of T.

n	15	20	25	30	35
Alg. 1 [1]	293.2	375.1	488.9	600.3	780.5
Alg. 2 with Algs. 3-1 and 3-2	11.7	11.0	—	—	—
Alg. 2 with Algs. 4-1(A) and 4-2(A)	11.7	11.0	10.0	10.0	10.0

Table 5: Required times (sec) of Algorithms 1 and 2 for T with low symmetry. T were generated from $\boldsymbol{a} = (1, 1, 1)$, $\boldsymbol{b} = (100, 100, 100)$, and $c = 1$ in (17)–(19), and the size of T is $n^3 \times n^3$. The symbol "—" means that the algorithm did not converge to the minimum singular values of T.

n	15	20	25	30	35
Alg. 1 [1]	4.8	9.2	20.9	40.6	83.5
Alg. 2 with Algs. 3-1 and 3-2	0.3	0.5	—	—	—
Alg. 2 with Algs. 4-1(A) and 4-2(A)	9.9	29.0	64.8	134.9	262.2

Table 6: Relative errors between the approximate minimum singular value of T in Table 4 and the minimum singular value of T computed by `svd`. The superscript † indicates that the relative error in (20) equals 0.

n	15	20	25	30	35
Alg. 1 [1]	1.4e-15	1.0e-16†	1.2e-15	3.6e-15	4.0e-15
Alg. 2 with Algs. 3-1 and 3-2	7.3e-14	7.8e-10	—	—	—
Alg. 2 with Algs. 4-1(A) and 4-2(A)	4.8e-15	1.0e-16†	9.1e-15	1.4e-14	9.1e-15

In Table 4, barring the results for which Algorithm 2 did not converge, both Algorithm 2 with Algorithms 3-1 and 3-2 and Algorithm 2 with Algorithms 4-1(A) and 4-2(A) required about 10 iterations to compute the approximate minimum singular values of T, with an accuracy of about 10^{-14} from Table 6. Thus, for matrix T with low symmetry, Algorithm 2 for T^{-1} also converged faster than Algorithm 1 for T concerning the number of iterations. On the other hand, regarding the required time in Table 5, Algorithm 1 converged the most rapidly among all of the algorithms. Focusing on the cases $n = 25, 30, 35$, Algorithm 2 with Algorithms 3-1 and 3-2 did not converge to the minimum singular value of T. These results may be caused by the condition numbers of $X_{(A)}$, $X_{(B)}$, $X_{(C)}$, $Y_{(A)}$, $Y_{(B)}$, and $Y_{(C)}$ in Algorithms 3-1 and 3-2, where the condition numbers are shown in Table 7, i.e., computed T^{-1} may completely differ from original T^{-1}.

Table 7: Condition numbers of $X_{(A)}, X_{(B)}, X_{(C)}, Y_{(A)}, Y_{(B)}, Y_{(C)}$ of T used in Table 4.

n	15	20	25	30	35
$X_{(A)}, X_{(B)}, X_{(C)}$	1.1e+2	5.1e+3	1.1e+6	1.5e+9	2.8e+13
$Y_{(A)}, Y_{(B)}, Y_{(C)}$	1.1e+2	5.1e+3	1.1e+6	1.5e+9	2.8e+13

From Tables 4 and 7, it seems that Algorithm 2 with Algorithms 3-1 and 3-2 did not converge when the condition numbers of $X_{(A)}, X_{(B)}, X_{(C)}, Y_{(A)}, Y_{(B)}$, and $Y_{(C)}$ are too large. On the other hand, Algorithm 2 with Algorithms 4-1(A) and 4-2(A) stably converged, since the condition numbers of the obtained matrices from the Schur decomposition are close to those of the original matrices.

According to all of the above results, Algorithm 2 requires much fewer iterations than Algorithm 1 when the minimum singular value of T is computed. Moreover, the implementation of Algorithm 2 with Algorithms 4-1(A) and 4-2(A) using the Schur decomposition is more robust than that of Algorithm 2 with Algorithms 3-1 and 3-2 using the eigendecomposition. Especially, Algorithm 2 with Algorithms 4-1(A) and 4-2(A) is the stablest method regardless of the condition number of T. However, you can choose Algorithm 2 with Algorithms 3-1 and 3-2 if you know that the matrices which have the eigenvectors of T are not ill-conditioned since Algorithm 2 with Algorithms 4-1(A) and 4-2(A) is the method which requires the more extended time than Algorithm 2 with Algorithms 3-1 and 3-2.

4.1.2 Numerical results for T composed of random matrices A, B, C

Here, we present Tables 8–10 which are results for T composed of random matrices A, B, C. Tables 8 and 9 show the average number of iterations and the average required time, respectively. Table 10 shows the relative errors between the approximate minimum singular value of T in Table 8.

Table 8: Numbers of iterations of Algorithms 1 and 2 for T with random matrices A, B, C and the size of T is $n^3 \times n^3$. The symbol "—" means that the algorithm did not converge to the minimum singular values of T.

n	15	20	25	30	35
Alg. 1 [1]	—	—	—	—	—
Alg. 2 with Algs. 3-1 and 3-2	13.5	13.2	13.4	14.9	15.4
Alg. 2 with Algs. 4-1(A) and 4-2(A)	13.5	13.2	13.4	15.3	15.3

Table 9: Required times (sec) of Algorithms 1 and 2 for T with random matrices A, B, C and the size of T is $n^3 \times n^3$. The symbol "—" means that the algorithm did not converge to the minimum singular values of T.

n	15	20	25	30	35
Alg. 1 [1]	—	—	—	—	—
Alg. 2 with Algs. 3-1 and 3-2	0.3	0.5	0.9	1.6	2.7
Alg. 2 with Algs. 4-1(A) and 4-2(A)	10.7	33.3	79.6	187.9	355.5

Table 10: Relative errors in (20) between the approximate minimum singular value of T in Table 8 and the minimum singular value of T computed by `svd`. The superscript † indicates that the relative error in (20) equals 0.

n	15	20	25	30	35
Alg. 1 [1]	—	—	—	—	—
Alg. 2 with Algs. 3-1 and 3-2	6.4e-14	3.9e-13	9.3e-13	8.8e-13	8.5e-13
Alg. 2 with Algs. 4-1(A) and 4-2(A)	6.4e-14	3.9e-13	9.3e-13	8.8e-13	8.5e-13

In Tables 8–10, Algorithm 1 did not converge for all T composed of random numbers. However, Algorithm 2 with Algorithms 3-1 and 3-2 and Algorithm 2 with Algorithms 4-1(A) and 4-2(A) converged to the approximate minimum singular values whose accuracies are 10^{-13}. Namely, Algorithm 2 is hardly affected by the structure of T. From Table 9, the required times by Algorithm 2 with Algorithms 3-1 and 3-2 were much less than Algorithm 2 with Algorithms 4-1(A) and 4-2(A). Therefore, for T composed of random matrices, it seems that Algorithm 2 with Algorithms 3-1 and 3-2 using the eigendecomposition is the most suitable method with regard to the stability and the rapidity.

5 Concluding remarks

This paper presented an invert Lanczos bidiagonalization method over tensor space with tensor-structure-preserving direct methods. From the results of the numerical examples obtained from PDE, it was found that the number of iterations by the proposed algorithm was at most 260 times lower than that of the conventional method [1]. Furthermore, when T has low symmetry, the proposed method using the Schur decomposition was more robust in terms of the number of iterations and the accuracy, than that using the eigendecomposition. From the results of the numerical examples of random matrices, we confirmed that the proposed algorithm implementing either the Schur decomposition or the eigendecomposition was similarly more robust than the conventional method. Comparing the proposed algorithms, the implementation by the eigendecomposition converged more rapidly than the implementation by the Schur decomposition. However, the implementation by the Schur decomposition was more robust concerning the condition number of the matrices with the eigenvectors of T than the implementation by the eigendecomposition.

Our future work will be as follows: 1) the result in this paper will be extended to a shift-and-invert Lanczos bidiagonalization over tensor space for computing singular triplets close to a given target point. Up to now, there has been a certain freedom over tensor space, yielding several algorithms. Therefore, finding the best algorithm among these will be required. Furthermore, combining successful restarting techniques and shift strategies, see, e.g., excellent work [7–11] will also be required; 2) when matrices A, B, and C are large and sparse, the proposed algorithm may require Krylov subspace methods [3]. In this case, it is of prime importance to find memory-efficient preconditioners, e.g., some tensor-structure-preserving preconditioner based on implicit wavelet preconditioners [12, 13].

Acknowledgments The authors wish to express our gratitude to Dr. Volker Mehrmann (TU Berlin) and Dr. Dmitry Savostyanov (University of Southampton at the time) for their fruitful comments regarding a very early version of this research during the SLA2014 conference in Greece. This work has been supported in part by JSPS KAKENHI Grant Number JP18H05392. The authors would like to thank Enago (www.enago.jp) for the English language review.

References

[1] A. Ohashi, T. Sogabe, On computing maximum/minimum singular values of a generalized tensor sum, Electron. T. Numer. Ana., 43(1), 244–254 (2015)
[2] J. Ballani, L. Grasedyck, A projection method to solve linear systems in tensor format, Numer. Linear Algebr., 20(1), 27–43 (2013)
[3] D. Kressner, C. Tobler, Krylov subspace methods for linear systems with tensor product structure, SIAM J. Matrix Anal. A., 31(4), 1688–1714 (2010)
[4] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev., 51(3), 455–500 (2009)
[5] G. Golub, W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, SIAM J. Numer. Anal., 2(2), 205–224 (1965)
[6] B.W. Li, S.Tian, Y.S. Sun, Z.M. Hu, Schur-decomposition for 3D matrix equations and its application in solving radiative discrete ordinates equations discretized by Chebyshev collocation spectral method, J. Comput. Phys., 229(4), 1198–1212 (2010)
[7] E. Kokiopoulou, C. Bekas, E. Gallopoulos, Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization, Appl. Numer. Math., 49(1), 39–61 (2004)
[8] Z. Jia, D. Niu, Calculating the singular values and pseudo-inverse of a matrix, SIAM J. Sci. Comput., 32(2), 714–744 (2010)
[9] D. Niu, X. Yuan, A harmonic Lanczos bidiagonalization method for computing interior singular triplets of large matrices, Appl. Math. Comput., 218(14), 7459–7467 (2012)
[10] J. Baglama, L. Reichel, An implicitly restarted block Lanczos bidiagonalization method using Leja shifts, BIT, 53(2), 285–310 (2013)
[11] D. Niu, X. Yuan, An implicitly restarted Lanczos bidiagonalization method with refined harmonic shifts for computing smallest singular triplets, J. Comput. Appl. Math., 260(14), 208-217 (2014)
[12] S.C. Hawkins, K. Chen, An implicit wavelet sparse approximate inverse preconditioner, SIAM J. Sci. Comput., 27(2), 667-686 (2005)
[13] A. Imakura, T. Sogabe, S.L. Zhang, An implicit wavelet sparse approximate inverse preconditioner using block finger pattern, Numer. Linear Algebr., 16(11–12), 915-928 (2009)