An Improvement of Augmented Implicitly Restarted Lanczos Bidiagonalization Method

Yuya Ishida[1], Masami Takata[2], Kinji Kimura[1], and Yoshimasa Nakamura[1]
[1]Graduate School of Informatics, Kyoto University, Kyoto, Kyoto, JAPAN
[2]Research Group of Information and Communication Technology for Life, Nara Women's University, Nara, Nara, JAPAN

Abstract— *Efficient processing for big data is attracting increased attention in many scientific problems. In particular, singular value decomposition (SVD) of matrices is one of the most significant operations in linear algebra. For example, the truncated SVD is used for principal component analysis of large-scale document-term matrices. In this paper, we improve the augmented implicitly restarted Lanczos bidiagonalization (AIRLB) method for the truncated SVD of large-scale sparse matrices. Instead of using both left and right singular vectors, we obtain right singular vectors and use the QR decomposition for SVD of an inner matrix to obtain left singular vectors. As a result, several numerical experiments show that our improvements increase the accuracy of truncated SVD compared with a conventional algorithm.*

Keywords: truncated SVD, large-scale sparse matrices, Lanczos algorithm

1. Introduction

The singular value decomposition (SVD) of the matrices is one of the most significant operations in numerical linear algebra and scientific computing. In some applications of SVD, a part of the singular values and singular vectors of the input matrix may be required. Such decomposition is called a truncated SVD. For example, in the principal component analysis of a large-scale sparse matrix, only some singular values and singular vectors corresponding to larger singular values are required. We call a triplet of a singular value and its left and right singular vectors a singular triplet.

The QR algorithm [5], the Jacobi algorithm [6], the divide-and-conquer algorithm [6], and the bisection and inverse iteration algorithm [13] are the best known SVD algorithms in LAPACK [1]. Like the SVD algorithms, there are some algorithms to compute truncated SVD. The Golub–Kahan–Lanczos (GKL) algorithm [7], the Jacobi–Davidson algorithm [14], the randomized algorithm [8], and the augmented implicitly restarted Lanczos bidiagonalization (AIRLB) algorithm [3], [2] are the best known truncated SVD algorithms. The GKL algorithm is a classical algorithm. The Jacobi–Davidson algorithm is suitable for the largest singular value and its singular vectors. The randomized algorithm is suitable for a truncated SVD whose singular values are not clustered. The AIRLB algorithm is appropriate for use as a computation library since it has low dependency on input matrices and can output solutions stably.

We have developed a truncated SVD library that can be downloaded from [10]. Thus, in this paper, we make the AIRLB algorithm more accurate.

In Section 2, we introduce algorithms for solving truncated SVD problems. In Section 3, we improve the AIRLB algorithm. In Section 4, we evaluate the accuracy of the improved algorithm.

2. Algorithms for Solving Truncated SVD Problems

2.1 Singular Value Decomposition

Let A be an $m \times n$ $(m \geq n)$ real matrix with rank r. The SVD of A is $A = U\Sigma V^\top$ and is also described as

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

where

$$U := \left[\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r\right] \in \mathbb{R}^{m\times r}, \tag{3}$$
$$V := \left[\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r\right] \in \mathbb{R}^{n\times r}, \tag{4}$$

are column orthogonal matrices and

$$\Sigma := \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}, \tag{5}$$

is a nonsingular diagonal matrix. Without loss of generality, we can assume that the decomposition satisfies $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. We denote by σ_i the ith singular value, $\boldsymbol{u}_i$ as the corresponding left singular vector, and $\boldsymbol{v}_i$ as the corresponding right singular vector.

For the truncated SVD of matrix A,

$$\sqrt{\|A\boldsymbol{v}_i - \sigma_i \boldsymbol{u}_i\|^2 + \|A^\top \boldsymbol{u}_i - \sigma_i \boldsymbol{v}_i\|^2} \quad (i = 1, \ldots, l) \tag{6}$$

is called the SVD error. If the SVD error is small, the matrix $U_l \Sigma_l V_l^\top$ with rank l is closely approximating the singular triplets of the input matrix A. Computation accuracy of SVD is estimated by these errors.

Algorithm 1 GKL algorithm

1: Set an n-dimensional unit vector $\boldsymbol{p}_1$
2: $\boldsymbol{q} \leftarrow A\boldsymbol{p}_1,\ \alpha_1 \leftarrow \|\boldsymbol{q}\|,\ \boldsymbol{q}_1 \leftarrow \boldsymbol{q}/\alpha_1$
3: $P_1 \leftarrow \left[\boldsymbol{p}_1\right], Q_1 \leftarrow \left[\boldsymbol{q}_1\right]$
4: **for** $k = 1,\ 2,\ \ldots$ **do**
5: $\quad \boldsymbol{p} \leftarrow A^\top \boldsymbol{q}_k$
6: $\quad \tilde{\boldsymbol{p}} \leftarrow$ Reorthogonalization$(P_k, \boldsymbol{p})$
7: $\quad \beta_k \leftarrow \|\tilde{\boldsymbol{p}}\|,\ \boldsymbol{p}_{k+1} \leftarrow \tilde{\boldsymbol{p}}/\beta_k$
8: $\quad$ Compute the SVD of $B_k = U_k \Sigma_k {V_k}^\top$
9: $\quad$ **if** $\max_{1 \le i \le l} \dfrac{|\beta_k \boldsymbol{u}_i(k)|}{\sqrt{2}} < \delta$ (threshold value) **then**
10: $\quad\quad \hat{\sigma}_i \leftarrow \sigma_i, \hat{\boldsymbol{u}}_i \leftarrow Q_k \boldsymbol{u}_i, \hat{\boldsymbol{v}}_i \leftarrow P_k \boldsymbol{v}_i$
11: $\quad\quad$ Stop algorithm and output $(\hat{\sigma}_i, \hat{\boldsymbol{u}}_i, \hat{\boldsymbol{v}}_i)$ as ith triplets of A
12: $\quad$ **end if**
13: $\quad \boldsymbol{q} \leftarrow A\boldsymbol{p}_{k+1}$
14: $\quad \tilde{\boldsymbol{q}} \leftarrow$ Reorthogonalization$(Q_k, \boldsymbol{q})$
15: $\quad \alpha_{k+1} \leftarrow \|\tilde{\boldsymbol{q}}\|,\ \boldsymbol{q}_{k+1} \leftarrow \tilde{\boldsymbol{q}}/\alpha_{k+1}$
16: $\quad P_{k+1} \leftarrow \left[P_k \quad \boldsymbol{p}_{k+1}\right],\ Q_{k+1} \leftarrow \left[Q_k \quad \boldsymbol{q}_{k+1}\right]$
17: **end for**

2.2 GKL Algorithm

The GKL algorithm outputs l singular triplets corresponding to large singular values of an input matrix $A \in \mathbb{R}^{m \times n}$ $(m \ge n)$ with rank r. We show the pseudocode of the GKL algorithm in Algorithm 1. This algorithm iterates the bidiagonalization of the input matrix to $B_k \in \mathbb{R}^{k \times k}$ and the SVD of the generated bidiagonalized matrix.

First, we set a suitable unit vector $\boldsymbol{p}_1 \in \mathbb{R}^n$. In the kth steps, we generate $\boldsymbol{p}_k \in \mathbb{R}^n$ and $\boldsymbol{q}_k \in \mathbb{R}^m$. These vectors are generated according to following two Krylov subspaces:

$$\begin{aligned} &\mathcal{K}(A^\top A, \boldsymbol{p}_1, k) \\ &= \mathrm{span}\{\boldsymbol{p}_1, (A^\top A)\boldsymbol{p}_1, \ldots, (A^\top A)^{k-1}\boldsymbol{p}_1\}, \qquad (7) \\ &\mathcal{K}(AA^\top, A\boldsymbol{p}_1, k) \\ &= \mathrm{span}\{A\boldsymbol{p}_1, (AA^\top)A\boldsymbol{p}_1, \ldots, (AA^\top)^{k-1}A\boldsymbol{p}_1\}. \qquad (8) \end{aligned}$$

Following bidiagonalization by the Krylov subspace, the singular values of B_k well approximate the large singular values of A. Moreover, the approximated singular vectors can be expressed by the product of the singular vectors of B_k and the transformation matrix $Q_k \in \mathbb{R}^{m \times k}$ and $P_k \in \mathbb{R}^{n \times k}$ used for bidiagonalization. Each vector generated according to the Krylov subspace is orthogonalized to be an orthogonal basis by applying the complete classical Gram–Schmidt algorithm [4] two times (CGS2) for high accuracy. By using level 1 Basic Linear Algebra Subprograms (BLAS) [11], reorthogonalization of $\boldsymbol{p}$ with P_k means applying the following equation twice:

$$\boldsymbol{p} \ \leftarrow \ \boldsymbol{p} - \sum_{j=1}^{k} \langle \boldsymbol{p}_j, \boldsymbol{p} \rangle \boldsymbol{p}_j. \qquad (9)$$

To improve computation speed, Expression (9) is implemented by using matrix–vector multiplication using the level 2 BLAS as

$$\boldsymbol{p}' \leftarrow P_k^\top \boldsymbol{p}, \quad \boldsymbol{p} \leftarrow \boldsymbol{p} - P_k \boldsymbol{p}'. \qquad (10)$$

By using the column orthogonal matrices P_k and Q_k, A is bidiagonalized to B_k. The form of B_k is

$$B_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{k-1} & \beta_{k-1} \\ & & & & \alpha_k \end{bmatrix} \qquad (11)$$

and the following equations holds

$$\begin{aligned} AP_k &= Q_k B_k, \qquad (12) \\ A^\top Q_k &= P_k B_k^\top + \beta_k \boldsymbol{p}_{k+1} \boldsymbol{e}_k^\top, \qquad (13) \end{aligned}$$

where $\boldsymbol{e}_k$ is the kth column of the $k \times k$ identity matrix.

The matrix size of B_k is smaller than the size of A, so executing SVD for B_k is easier than A. By executing SVD of B_k, we obtain $U_k = \left[\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k\right] \in \mathbb{R}^{k \times k}$, $V_k = \left[\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_i\right] \in \mathbb{R}^{k \times k}$ and $\Sigma_k = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_i) \in \mathbb{R}^{k \times k}$ where $B_k = U_k \Sigma_k V_k^\top$ and

$$B_k \boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i, \quad B_k^\top \boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i, \qquad (14)$$

for $i = 1, \ldots, k$. Using Eqs. (12), (13), and (14), we obtain

$$\begin{aligned} AP_k \boldsymbol{v}_i &= Q_k B_k \boldsymbol{v}_i \\ &= \sigma_i Q_k \boldsymbol{u}_i, \qquad (15) \\ A^\top Q_k \boldsymbol{u}_i &= P_k B_k^\top \boldsymbol{u}_i + \beta_k \boldsymbol{p}_{k+1} \boldsymbol{e}_k^\top \boldsymbol{u}_i \\ &= \sigma_i P_k \boldsymbol{v}_i + \beta_k \boldsymbol{p}_{k+1} \boldsymbol{e}_k^\top \boldsymbol{u}_i. \qquad (16) \end{aligned}$$

By defining $\hat{\sigma}_i := \sigma_i, \hat{\boldsymbol{v}}_i := P_k \boldsymbol{v}_i$ and $\hat{\boldsymbol{u}}_i := Q_k \boldsymbol{u}_i$, Eqs. (15) and (16) are described as

$$\begin{aligned} A\hat{\boldsymbol{v}}_i &= \hat{\sigma}_i \hat{\boldsymbol{u}}_i, \qquad (17) \\ A^\top \hat{\boldsymbol{u}}_i &= \hat{\sigma}_i \hat{\boldsymbol{v}}_i + \beta_k \boldsymbol{p}_{k+1} \boldsymbol{e}_k^\top \boldsymbol{u}_i. \qquad (18) \end{aligned}$$

If second term of Eq. (18) is zero, then Eqs. (17) and (18) are equal to Eq. (1). Therefore, the truncated SVD is complete.

We estimate the error of $\hat{\sigma}_i$ as the singular value of matrix A. The following theorem provides an upper bound of the singular value error.

Theorem 1 (Wilkinson's theorem [15]): Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of an $n \times n$ real symmetric matrix A. If $\|\hat{\boldsymbol{x}}\| = 1$, then

$$\min_j |\hat{\lambda} - \lambda_j| \le \|M\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}\|.$$

Let the true values of singular values of the matrix A be $\sigma_1, \sigma_2, \ldots, \sigma_n$. The ith singular triplets $(\hat{\sigma}_i, \hat{\boldsymbol{u}}_i, \hat{\boldsymbol{v}}_i)$ obtained by the GKL algorithm for the matrix A correspond to the eigenvalue $\hat{\lambda}_i := \hat{\sigma}_i$ and the eigenvector

$$\hat{\boldsymbol{x}}_i := \frac{1}{\sqrt{\|\hat{\boldsymbol{u}}_i\|^2 + \|\hat{\boldsymbol{v}}_i\|^2}} \begin{bmatrix} \hat{\boldsymbol{u}}_i \\ \hat{\boldsymbol{v}}_i \end{bmatrix} \in \mathbb{R}^{m+n} \qquad (19)$$

of the expansion matrix

$$M := \begin{bmatrix} & A \\ A^\top & \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)} \tag{20}$$

of A. Thus,

$$\begin{aligned} &\min_j |\hat{\sigma}_i - \sigma_j| = \min_j |\hat{\lambda}_i - \lambda_j| \\ &\leq \|M\hat{\boldsymbol{x}}_i - \hat{\lambda}_i\hat{\boldsymbol{x}}_i\| = \sqrt{\|M\hat{\boldsymbol{x}}_i - \hat{\lambda}_i\hat{\boldsymbol{x}}_i\|^2} \\ &= \frac{\sqrt{\|A\hat{\boldsymbol{v}}_i - \hat{\sigma}_i\hat{\boldsymbol{u}}_i\|^2 + \|A^\top\hat{\boldsymbol{u}}_i - \hat{\sigma}_i\hat{\boldsymbol{v}}_i\|^2}}{\sqrt{2}}. \end{aligned} \tag{21}$$

The right-hand side of Eq. (21) is regarded as the upper bound of the singular value error and this is used as a singular value error afterwards. From Eqs. (17) and (18), the singular value error of A is as follows,

$$\begin{aligned} &\frac{\sqrt{\|A\hat{\boldsymbol{v}}_i - \hat{\sigma}_i\hat{\boldsymbol{u}}_i\|^2 + \|A^\top\hat{\boldsymbol{u}}_i - \hat{\sigma}_i\hat{\boldsymbol{v}}_i\|^2}}{\sqrt{2}} \\ &= \frac{\|\beta_k\boldsymbol{p}_{k+1}\boldsymbol{e}_k^\top\boldsymbol{u}_i\|}{\sqrt{2}} = \frac{\|\beta_k\boldsymbol{p}_{k+1}\boldsymbol{u}_i(k)\|}{\sqrt{2}} \\ &= \frac{\|\boldsymbol{p}_{k+1}\| \cdot |\beta_k\boldsymbol{u}_i(k)|}{\sqrt{2}} = \frac{|\beta_k\boldsymbol{u}_i(k)|}{\sqrt{2}}. \end{aligned} \tag{22}$$

Since error can be estimated by $|\beta_k\boldsymbol{u}_i(k)|/\sqrt{2}$ with a small amount of computation, $\max_{1\leq i\leq l}(|\beta_k\boldsymbol{u}_i(k)|/\sqrt{2})$ can be used for the stopping criterion.

As P_k and Q_k are enlarged as the number of iterations increases, the GKL algorithm will use more spaces in computation. The algorithm uses a space for P_k and Q_k at most $mn + n^2$ on the computer.

2.3 AIRLB Algorithm

The AIRLB algorithm is one of the Krylov subspace algorithms, and it can obtain the singular triplets corresponding to large singular values of large-scale sparse matrices faster than the GKL algorithm with a smaller memory space. In the GKL algorithm, the memory space to use and the amount of computation increase as the number of iterations increases. The AIRLB algorithm overcomes the problem. This algorithm is given in Algorithm 2.

Assume that we need l singular triplets of matrix A. First, take the same procedure as the GKL algorithm and obtain a $k \times k$ $(l < k)$ bidiagonal matrix $\tilde{B}_k$. In general, twice the value of l is used for k. Next, the SVD is performed on the obtained matrix $\tilde{B}_k = \tilde{U}_k\tilde{\Sigma}_k\tilde{V}_k^\top$. Then, we continue with the GKL algorithm leaving only necessary l singular triplets corresponding to large l singular values. The triplets are sorted in order and the remaining $k - l$ triplets are discarded. Next, reuse the remaining l singular triplets to obtain a $k \times k$ pseudo-diagonal matrix $\tilde{B}'_k \in \mathbb{R}^{k\times k}$. Creating new matrix requires $k - l$ iterations of the GKL steps. In reorthogonalization of the vectors, the CGS2 is applied.

Algorithm 2 AIRLB algorithm

1: Set an n-dimensional unit vector $\tilde{\boldsymbol{v}}_1$, $i \leftarrow 1$
2: **repeat**
3: $\tilde{P}_i \leftarrow \left[\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, \ldots, \tilde{\boldsymbol{v}}_i\right]$
4: **while** $i \leq k$ **do**
5: $\boldsymbol{u} \leftarrow A\tilde{\boldsymbol{v}}_i$, Reorthogonalization($\tilde{Q}_i, \boldsymbol{u}$)
6: $\tilde{\alpha}_i \leftarrow \|\boldsymbol{u}\|$, $\tilde{\boldsymbol{u}}_i \leftarrow \boldsymbol{u}/\tilde{\alpha}_i$
7: $\tilde{Q}_i \leftarrow \left[\tilde{\boldsymbol{u}}_1, \tilde{\boldsymbol{u}}_2, \ldots, \tilde{\boldsymbol{u}}_i\right]$
8: $\boldsymbol{v} \leftarrow A^\top\tilde{\boldsymbol{u}}_i$, Reorthogonalization($\tilde{P}_i, \boldsymbol{v}$)
9: $\tilde{\beta}_i \leftarrow \|\boldsymbol{v}\|$, $\tilde{\boldsymbol{v}}_{i+1} \leftarrow \boldsymbol{v}/\tilde{\beta}_i$
10: $\tilde{P}_{i+1} \leftarrow \left[\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, \ldots, \tilde{\boldsymbol{v}}_{i+1}\right]$
11: $i \leftarrow i + 1$
12: **end while**
13: $\tilde{\boldsymbol{v}}_{l+1} \leftarrow \tilde{\boldsymbol{v}}_{k+1}$
14: **Compute the SVD of** $\tilde{B}_k = \tilde{U}_k\tilde{\Sigma}_k\tilde{V}_k^\top$
15: **for** $i = 1, \ldots, l$ **do**
16: $\tilde{\rho}_i \leftarrow \tilde{\beta}_k\tilde{\boldsymbol{u}}_i(k)$
17: **end for**
18: $\tilde{B}_k(1:l, 1:l) \leftarrow \tilde{\Sigma}_k(1:l, 1:l)$, $\tilde{Q}_k \leftarrow \tilde{Q}_k\tilde{U}_k(:, 1:l)$, $\tilde{P}_k \leftarrow \tilde{P}_k\tilde{V}_k(:, 1:l)$
19: $i \leftarrow l + 1$
20: **until** $\max_{1\leq i\leq l} \frac{|\tilde{\rho}_i|}{\sqrt{2}} \leq \delta$ (threshold value)
21: $\tilde{\boldsymbol{u}}_i \leftarrow \tilde{Q}_k(:, i)$, $\tilde{\boldsymbol{v}}_i \leftarrow \tilde{P}_k(:, i)$
22: Output $(\tilde{\sigma}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i)$ for $i = 1, \ldots, l$

Then, the pseudo-bidiagonal matrix $\tilde{B}'_k$ is given as

$$\tilde{B}'_k = \begin{bmatrix} \tilde{\sigma}_1 & & & \tilde{\rho}_1 & & & & \\ & \ddots & & \vdots & & & & \\ & & \tilde{\sigma}_l & \tilde{\rho}_l & & & & \\ & & & \tilde{\alpha}_{l+1} & \tilde{\beta}_{l+1} & & & \\ & & & & \ddots & \ddots & & \\ & & & & & \tilde{\alpha}_{k-1} & \tilde{\beta}_{k-1} & \\ & & & & & & \tilde{\alpha}_k & \end{bmatrix}, \tag{23}$$

where $\tilde{\sigma}_i$ is the ith largest singular value of $\tilde{B}'_k$, and $\tilde{\rho}_i := \beta_k\tilde{\boldsymbol{u}}_i(k)$. The lower right bidiagonal part is composed of bidiagonalization by the GKL algorithm with $\tilde{\boldsymbol{v}}_{k+1}$ as the initial vector. Here $\tilde{B}'_k$ is treated as a new $\tilde{B}_k$ and the SVD is continued to confirm the accuracy of the singular triplets as the input matrix A. Iterate these restart procedures until the singular value error is small enough in the sense of Wilkinson's theorem (Theorem 1).

In the SVD of the matrix $\tilde{B}'_k$ in the algorithm, it is desirable to not process $\tilde{B}'_k$ directly, but to bidiagonalize as preprocessing and apply the algorithm afterwards from the viewpoint of reducing the amount of computation. Actually, we make a bidiagonal matrix $G_L\tilde{B}'_kG_R$ from $\tilde{B}'_k$ by using the Givens transformation [6] of rotation matrices $G_L \in \mathbb{R}^{k\times k}$ and $G_R \in \mathbb{R}^{k\times k}$. In this paper, from the viewpoint of computational complexity and computational accuracy,

we use orthogonal transformation by the Givens transformation, not the Householder transformation. For the Givens transformation, we use LAPACK DLARTG [1] instead of BLAS DROTG [11] to achieve high accuracy. The rotation matrices are orthogonal matrices, the singular values are invariant under the rotation, and the SVD holds $G_L \tilde{B}'_k G_R = \tilde{U}' \tilde{\Sigma}' \tilde{V}'^{\top}$. Therefore, the column orthogonal matrices $G_L^{\top}\tilde{U}'$ and $G_R\tilde{V}'$ are the singular vectors of $\tilde{B}'_k$.

Let us prepare column orthogonal matrices $\tilde{Q}_k \in \mathbb{R}^{m\times k}$ and $\tilde{P}_k \in \mathbb{R}^{n\times k}$ to generate $\tilde{B}_k$ in Algorithm 2. Equations (12) and (13) of the GKL algorithm also lead to the following equations:

$$A\tilde{P}_k = \tilde{Q}_k\tilde{B}_k, \quad (24)$$

$$A^{\top}\tilde{Q}_k = \tilde{P}_k\tilde{B}_k^{\top} + \tilde{\beta}_k\tilde{\boldsymbol{p}}_{k+1}\boldsymbol{e}_k^{\top}, \quad (25)$$

where the n-dimensional vector $\tilde{\boldsymbol{p}}_{k+1}$ is the $(k+1)$th column of $\tilde{P}_{k+1}$. By multiplying $\tilde{U}_l$ and $\tilde{V}_l$, which are singular vectors of $\tilde{B}_k$, we obtain

$$A\tilde{P}_l = \tilde{Q}_l\tilde{\Sigma}_l, \quad (26)$$

$$A^{\top}\tilde{Q}_l = \tilde{P}_l\tilde{\Sigma}_l^{\top} + \tilde{\beta}_k\tilde{\boldsymbol{p}}_{k+1}\boldsymbol{e}_k^{\top}\tilde{U}_l, \quad (27)$$

where $\tilde{Q}_l$ is substituted by $\tilde{Q}_k\tilde{U}_l$ and $\tilde{P}_l$ is substituted by $\tilde{P}_k\tilde{V}_l$. At the next restart of the algorithm, $\tilde{\Sigma}_l$, $\tilde{Q}_l$, and $\tilde{P}_l$ are adopted as new initial matrices at line 18 of Algorithm 2.

From Eqs. (26) and (27), the upper bound of the singular value error is described as

$$\min_j |\tilde{\sigma}_i - \sigma_j| \le \frac{\sqrt{\|A\tilde{\boldsymbol{v}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{u}}_i\|^2 + \|A^{\top}\tilde{\boldsymbol{u}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{v}}_i\|^2}}{\sqrt{2}} = \frac{|\tilde{\rho}_i|}{\sqrt{2}}, \quad (28)$$

where $\tilde{\boldsymbol{u}}_i$ is the ith column of $\tilde{Q}_l$, $\tilde{\boldsymbol{v}}_i$ is the ith column of $\tilde{P}_l$, and $\tilde{\rho}_i$ is the element of $\tilde{B}'_k$ at $(i, l+1)$. Similarly to the GKL algorithm, $\max_{1\le i\le l}(|\tilde{\rho}_i|/\sqrt{2})$ is used as a stopping criterion.

In the AIRLB algorithm, $\tilde{P}_i$ and $\tilde{Q}_i$ are not enlarged over k. The algorithm uses a maximum memory space for $\tilde{P}_i$ and $\tilde{Q}_i$ of $mk + nk$.

3. New Restart Strategy

In the AIRLB algorithm, the SVD of the small matrix $\tilde{B}_k$ is performed internally and the result is used at the restarting point of the algorithm. Unless computation errors are considered, the singular vectors obtained by SVD are orthogonal matrices. The GKL algorithm is known to be unstable. Thus, the orthogonality becomes worse because of the rounding error. To avoid this problem, we propose one-sided restart strategy. We introduce a method to obtain singular vectors of $\tilde{B}_k$ with maximum orthogonality by decomposing the right side of the singular vectors into a column orthogonal matrix and an upper triangular matrix using the QR decomposition [6].

Algorithm 3 AIRLB algorithm (proposal algorithm)

1: Set an n-dimensional unit vector $\tilde{\boldsymbol{v}}_1$, $i \leftarrow 1$
2: **repeat**
3: $\tilde{P}_i \leftarrow \left[\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, \ldots, \tilde{\boldsymbol{v}}_i\right]$
4: **while** $i \le k$ **do**
5: $\boldsymbol{u} \leftarrow A\tilde{\boldsymbol{v}}_i$, Reorthogonalization($\tilde{Q}_i, \boldsymbol{u}$)
6: $\tilde{\alpha}_i \leftarrow \|\boldsymbol{u}\|$, $\tilde{\boldsymbol{u}}_i \leftarrow \boldsymbol{u}/\tilde{\alpha}_i$
7: $\tilde{Q}_i \leftarrow \left[\tilde{\boldsymbol{u}}_1, \tilde{\boldsymbol{u}}_2, \ldots, \tilde{\boldsymbol{u}}_i\right]$
8: $\boldsymbol{v} \leftarrow A^{\top}\tilde{\boldsymbol{u}}_i$, Reorthogonalization($\tilde{P}_i, \boldsymbol{v}$)
9: $\tilde{\beta}_i \leftarrow \|\boldsymbol{v}\|$, $\tilde{\boldsymbol{v}}_{i+1} \leftarrow \boldsymbol{v}/\tilde{\beta}_i$
10: $\tilde{P}_{i+1} \leftarrow \left[\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, \ldots, \tilde{\boldsymbol{v}}_{i+1}\right]$
11: $i \leftarrow i + 1$
12: **end while**
13: $\tilde{\boldsymbol{v}}_{l+1} \leftarrow \tilde{\boldsymbol{v}}_{k+1}$
14: **Compute the SVD of** $\tilde{B}_k = \tilde{U}_k\tilde{\Sigma}_k\tilde{V}_k^{\top}$
15: **Compute the QR Decomposition of** $\tilde{V}_l = Q_1R_1$
16: $\tilde{V}_l \leftarrow Q_1$
17: **Compute the QR Decomposition of** $\tilde{B}_k\tilde{V}_l = Q_2R_2$
18: $\tilde{U}_l \leftarrow Q_2$, $\tilde{\Sigma}_l \leftarrow R_2$
19: **for** $i = 1, \ldots, l$ **do**
20: $\tilde{\rho}_i \leftarrow \tilde{\beta}_k\tilde{\boldsymbol{u}}_i(k)$
21: **end for**
22: $\tilde{B}_k(1:l, 1:l) \leftarrow \tilde{\Sigma}_l$, $\tilde{P}_k \leftarrow \tilde{P}_k\tilde{V}_l$, $\tilde{Q}_k \leftarrow \tilde{Q}_k\tilde{U}_l$
23: $i \leftarrow l + 1$
24: **until** $\max_{1\le i\le l} \frac{|\tilde{\rho}_i|}{\sqrt{2}} \le \delta$ (threshold value)
25: $\tilde{\boldsymbol{u}}_i \leftarrow \tilde{Q}_k(:, i)$, $\tilde{\boldsymbol{v}}_i \leftarrow \tilde{P}_k(:, i)$
26: Output $(\tilde{\sigma}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i)$ for $i = 1, \ldots, l$

The whole algorithm is described in Algorithm 3.

In the conventional algorithm, l vectors are extracted from right singular vectors $\tilde{V}_k$ and set as new $\tilde{V}_l$. Our new algorithm uses the QR decomposition with $\tilde{V}_l = Q_1R_1$ for reorthogonalizing $\tilde{V}_l$. Let the orthogonal matrix Q_1 be a new $\tilde{V}_l$:

$$\tilde{V}_l \leftarrow \left[\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, \ldots, \tilde{\boldsymbol{v}}_l\right], \quad (29)$$

$$\tilde{V}_l = Q_1R_1, \quad (30)$$

$$\tilde{V}_l \leftarrow Q_1. \quad (31)$$

Since $\tilde{B}_k\tilde{V}_l = \tilde{U}_l\tilde{\Sigma}_l$ is established, to obtain reorthogonalized $\tilde{U}_l$, the QR decomposition is made with $\tilde{B}_k\tilde{V}_l = Q_2R_2$. We adopt the obtained column orthogonal matrix Q_2 as a new $\tilde{U}_l$:

$$\tilde{B}_k\tilde{V}_l = Q_2R_2, \quad (32)$$

$$\tilde{U}_l \leftarrow Q_2. \quad (33)$$

Through this procedure, singular vectors are reorthogonalized, but to reduce the residual $\|\tilde{B}_k\tilde{V}_l - \tilde{U}_l\tilde{\Sigma}_l\|$, $\tilde{\Sigma}_l$ also needs to be replaced with those corresponding to the new $\tilde{V}_l$ and $\tilde{U}_l$. We adopt the upper triangular matrix R_2 obtained in reorthogonalization of $\tilde{U}_l$ as $\tilde{\Sigma}_l \leftarrow R_2$.

We consider restarting the algorithm corresponding to the new $\tilde{U}_l$, $\tilde{V}_l$, and $\tilde{\Sigma}_l$. By multiplying $\tilde{V}_l$ by Eq. (24), we obtain

$$A\tilde{P}_k\tilde{V}_l = \tilde{Q}_k\tilde{B}_k\tilde{V}_l = \tilde{Q}_k\tilde{U}_l\tilde{\Sigma}_l.$$

By substituting $\tilde{P}_k\tilde{V}_l$ into $\tilde{P}_l$ and $\tilde{Q}_k\tilde{U}_l$ into $\tilde{Q}_l$, we obtain

$$A\tilde{P}_l = \tilde{Q}_l\tilde{\Sigma}_l. \tag{34}$$

Equation (34) is isomorphic to Eq. (26).

Next, by multiplying $\tilde{U}_l$ by Eq. (25) and using Eq. (32), we obtain the following equation:

$$A^\top\tilde{Q}_k\tilde{U}_l = \tilde{P}_k\tilde{B}_k^\top\tilde{U}_l + \tilde{\beta}_k\tilde{\boldsymbol{p}}_{k+1}\boldsymbol{e}_k^\top\tilde{U}_l. \tag{35}$$

By using right singular vectors $\tilde{V}_k = \begin{bmatrix}\tilde{V}_l & \tilde{V}_{l+1:k}\end{bmatrix}$, we can rewrite $\tilde{P}_k\tilde{B}_k^\top\tilde{U}_l$ as

$$\begin{aligned}\tilde{P}_k\tilde{B}_k^\top\tilde{U}_l &= \tilde{P}_k\tilde{V}_k\tilde{V}_k^\top\tilde{B}_k^\top\tilde{U}_l \\ &= \tilde{P}_k\tilde{V}_k\begin{bmatrix}\tilde{V}_l^\top\tilde{B}_k^\top \\ \tilde{V}_{l+1:k}^\top\tilde{B}_k^\top\end{bmatrix}\tilde{U}_l. \end{aligned}\tag{36}$$

Using Eq. (32), Eq. (36) is described as

$$\begin{aligned}&\tilde{P}_k\tilde{V}_k\begin{bmatrix}\tilde{V}_l^\top\tilde{B}_k^\top \\ \tilde{V}_{l+1:k}^\top\tilde{B}_k^\top\end{bmatrix}\tilde{U}_l \\ &= \tilde{P}_k\tilde{V}_k\left[R_2^\top\tilde{U}_l^\top\tilde{V}_{l+1:k}^\top\tilde{B}_k^\top\right]\tilde{U}_l = \tilde{P}_k\tilde{V}_k\begin{bmatrix}R_2^\top \\ \tilde{V}_{l+1:k}^\top\tilde{B}_k^\top\tilde{U}_l\end{bmatrix} \\ &= P_k(\tilde{V}_lR_2^\top + \tilde{V}_{l+1:k}\tilde{V}_{l+1:k}^\top\tilde{B}_k^\top\tilde{U}_l).\end{aligned}\tag{37}$$

By the QR decomposition of matrix $\tilde{B}_k\tilde{V}_k$, it holds

$$\tilde{B}_k\tilde{V}_k = \tilde{U}_k\begin{bmatrix}R_2 & \Gamma \\ & R_3\end{bmatrix}, \tag{38}$$

where R_2 is an upper triangular matrix obtained as in Eq. (32), R_3 is an upper triangular matrix and Γ is an $l \times (k-l)$ matrix. By multiplying $\tilde{U}_l^\top$ by Eq. (38),

$$\begin{aligned}\tilde{U}_l^\top\tilde{B}_k\tilde{V}_k &= \tilde{U}_l^\top\tilde{U}_k\begin{bmatrix}R_2 & \Gamma \\ & R_3\end{bmatrix} = \begin{bmatrix}I & 0\end{bmatrix}\begin{bmatrix}R_2 & \Gamma \\ & R_3\end{bmatrix} \\ &= \begin{bmatrix}R_2 & \Gamma\end{bmatrix}.\end{aligned}\tag{39}$$

Therefore, $\tilde{U}_l^\top\tilde{B}_k\tilde{V}_{l+1:k} = \Gamma$. By using the equation $\tilde{V}_{l+1:k}^\top\tilde{B}_k^\top\tilde{U}_l = \Gamma^\top$, Eq. (37) is described as

$$\tilde{P}_k(\tilde{V}_lR_2^\top + \tilde{V}_{l+1:k}\Gamma^\top). \tag{40}$$

Since we have confirmed that $\|\tilde{V}_{l+1:k}\Gamma^\top\|/\|\tilde{V}_lR_2^\top\| \ll 1$ in our experiments, the following approximation holds in the computation:

$$\tilde{V}_lR_2^\top + \tilde{V}_{l+1:k}\Gamma^\top \simeq \tilde{V}_lR_2^\top. \tag{41}$$

Let us assume here that $\tilde{V}_{l+1:k}\Gamma^\top = O$ where O is a zero matrix. Then Eq. (35) is described as

$$A^\top\tilde{Q}_k\tilde{U}_l = \tilde{P}_k\tilde{V}_lR_2^\top + \tilde{\beta}_k\tilde{\boldsymbol{p}}_{k+1}\boldsymbol{e}_k^\top\tilde{U}_l. \tag{42}$$

By substituting $\tilde{P}_k\tilde{V}_l$ into $\tilde{P}_l$ and $\tilde{Q}_k\tilde{U}_l$ into $\tilde{Q}_l$, we obtain

$$A^\top\tilde{Q}_l = \tilde{P}_lR_2^\top + \tilde{\beta}_k\tilde{\boldsymbol{p}}_{k+1}\boldsymbol{e}_k^\top\tilde{U}_l. \tag{43}$$

Therefore, Eq. (43) is isomorphic to Eq. (27). From Eqs. (26), (27), (34), and (43), the AIRLB algorithm adopting one-sided restart can continue in the same manner as the conventional algorithm.

In the computation, instead of generating a column orthogonal matrix and an upper triangular matrix by the direct QR decomposition and matrix multiplication using BLAS DGEMM, we implement our method with computational routines that prioritize accuracy by using the Householder reflectors [12] such as LAPACK DGEQRF and LAPACK DORMQR.

In this algorithm, $\tilde{B}_k$ is not a diagonal matrix but an upper triangular matrix

$$\begin{aligned}\tilde{B}_k &= \begin{bmatrix}R_2 & \tilde{\boldsymbol{\rho}} & & & \\ & \tilde{\alpha}_{l+1} & \tilde{\beta}_{l+1} & & \\ & & \ddots & \ddots & \\ & & & \tilde{\alpha}_{k-1} & \tilde{\beta}_{k-1} \\ & & & & \tilde{\alpha}_k\end{bmatrix} \\ &= \begin{bmatrix}\tilde{\sigma}_1 & \epsilon_{1,2} & \epsilon_{1,3} & \cdots & \tilde{\rho}_1 & & & & \\ & \tilde{\sigma}_2 & \epsilon_{2,3} & \cdots & \tilde{\rho}_2 & & & & \\ & & \ddots & \vdots & \vdots & & & & \\ & & & \tilde{\sigma}_l & \tilde{\rho}_l & & & & \\ & & & & \tilde{\alpha}_{l+1} & \tilde{\beta}_{l+1} & & & \\ & & & & & \ddots & \ddots & & \\ & & & & & & \tilde{\alpha}_{k-1} & \tilde{\beta}_{k-1} \\ & & & & & & & \tilde{\alpha}_k\end{bmatrix}.\end{aligned}\tag{44}$$

From the form changing of $\tilde{B}_k$, the form of the stopping criterion changes, as defined in the AIRLB algorithm. For the singular triplets $(\tilde{\sigma}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i)$ of the proposes algorithm,

$$\|A\tilde{\boldsymbol{v}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{u}}_i\| = \|\sum_{j=1}^{i-1}\epsilon_{j,i}\tilde{\boldsymbol{u}}_j\| \le \sum_{j=1}^{i-1}|\epsilon_{j,i}|. \tag{45}$$

Moreover,

$$\begin{aligned}\|A^\top\tilde{\boldsymbol{u}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{v}}_i\| &= \|\sum_{j=i+1}^{l}\epsilon_{i,j}\tilde{\boldsymbol{v}}_j + \tilde{\rho}_i\tilde{\boldsymbol{p}}_{k+1}\| \\ &\le \sum_{j=i+1}^{l}|\epsilon_{i,j}| + |\tilde{\rho}_i|.\end{aligned}\tag{46}$$

Although R_2 is upper triangular, it can be approximated to a diagonal matrix. Therefore, the upper bound of the singular value error is described as

$$\begin{aligned}&\min_j|\tilde{\sigma}_i - \sigma_j| \le \frac{\sqrt{\|A\tilde{\boldsymbol{v}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{u}}_i\|^2 + \|A^\top\tilde{\boldsymbol{u}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{v}}_i\|^2}}{\sqrt{2}} \\ &\le \frac{\sqrt{(\sum_{j=1}^{i-1}|\epsilon_{j,i}|)^2 + (\sum_{j=i+1}^{l}|\epsilon_{i,j}| + |\tilde{\rho}_i|)^2}}{\sqrt{2}} \\ &\simeq \frac{|\tilde{\rho}_i|}{\sqrt{2}}.\end{aligned}\tag{47}$$

Then, the same error criterion as the AIRLB algorithm can be used.

By using the Givens transformation, we create a bidiagonal matrix from $\tilde{B}_k$. Unlike in the conventional algorithm, dense bidiagonalization is required; however, from the aspect of precision, orthogonal transformation by the Givens rotation is used. In that case, LAPACK DLARTG is used instead of BLAS DROTG to achieve high accuracy.

4. Numerical Experiments

In this section, numerical experiments are performed to evaluate the proposed algorithm. To show the improvement by adopting a one-sided restarting strategy, we compare the implementation adopting the new restart strategy and the conventional implementation restarting with singular vectors on both sides.

4.1 Experiment Environment

For the experimental environment, we use a computer (ACCMS, Kyoto University) equipped with Intel Xeon Phi KNL CPU (1.4 GHz $\times$ 68 cores) and DDR4-2133 memory (90 GB). Each program is compiled using Intel C++ and Fortran Compilers 16.0.2 and Intel Math Kernel Library [9] as a computation library.

As a numerical experiment, we compare the AIRLB algorithms. Implementation of the QR algorithm uses DBDSQR on LAPACK 1.0 (SIAM SIAG/LA, 1991) [5].

For these numerical experiments, we prepare two types of matrices. First, we use real sparse matrices $A_1 \in \mathbb{R}^{1,000,000\times 1,000,000}$ and $A_2 \in \mathbb{R}^{1,800,000\times 1,800,000}$ as input. There are 1,000 elements consisting of uniform random numbers of $[0, 1)$ in each row. Here A_1 and A_2 are examples of large-scale sparse matrices, which are similar in data to real problems assuming large-scale document-term matrices. By performing SVD for these matrices, we show that our new implementation can solve the actual problems more accurately. Second, we use real bidiagonal matrices $A_3 \in \mathbb{R}^{10,000\times 10,000}$ and $A_4 \in \mathbb{R}^{50,000\times 50,000}$, all diagonal and off-diagonal elements are 1. The ith singular value of A_3 and A_4 is $1 - \cos(\frac{-2i+2n+1}{2n+1}\pi)$ where n is the matrix size. Therefore, large singular values of these matrices are quite clustered around 2. Thus, these matrices are difficult problems to solve. By solving SVD for these matrices, we show that our new implementation can solve difficult problems with high accuracy. The output is l $(l = 10, 20, 30)$ singular triplets corresponding to the larger singular values of the input matrices.

From Eq. (21), we adopt

$$\frac{1}{l}\sum_{1\le i\le l}\frac{1}{\sqrt{2}}\sqrt{\|A\tilde{\boldsymbol{v}}_i-\tilde{\sigma}_i\tilde{\boldsymbol{u}}_i\|^2+\|A^\top\tilde{\boldsymbol{u}}_i-\tilde{\sigma}_i\tilde{\boldsymbol{v}}_i\|^2} \quad (48)$$

as the average error value and

$$\max_{1\le i\le l}\frac{1}{\sqrt{2}}\sqrt{\|A\tilde{\boldsymbol{v}}_i-\tilde{\sigma}_i\tilde{\boldsymbol{u}}_i\|^2+\|A^\top\tilde{\boldsymbol{u}}_i-\tilde{\sigma}_i\tilde{\boldsymbol{v}}_i\|^2} \quad (49)$$

as the maximum error value for machine computed singular triplets $(\tilde{\sigma}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i)$ of A. Moreover, we use the orthogonal errors

$$\|\tilde{U}_l^\top\tilde{U}_l - I\|,\ \|\tilde{V}_l^\top\tilde{V}_l - I\| \quad (50)$$

to check orthogonality of $\tilde{U}_l = \left[\tilde{\boldsymbol{u}}_1, \tilde{\boldsymbol{u}}_2, \ldots, \tilde{\boldsymbol{u}}_l\right]$ and $\tilde{V}_l = \left[\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, \ldots, \tilde{\boldsymbol{v}}_l\right]$.

4.2 Discussion of Numerical Experiment

Figure 1 shows the computational results for performing truncated SVD. As a result, when restarting with only singular vectors on the right side, the average and the maximum value of the singular value error, orthogonal errors of $\tilde{U}_l$ and $\tilde{V}_l$ are decreased as compared with the case of restarting using the singular vectors on both sides. The reduction in error is thus established. Regarding the computation time, the computation time of the matrix–vector operation is dominant over the computation time required for the SVD of the internal small matrix $\tilde{B}_k$. Therefore, no matter what inner routine is used, there is only a small difference in computation time. As a result, it is verified that our improvement is effective for the truncated SVD of large-scale sparse matrices on real and difficult problems, and it is desirable for highly accurate computation to adopt one-sided restart for implementation of the AIRLB algorithm.

5. Conclusions

In this paper, we have improved the AIRLB algorithm to compute truncated SVD of the input large-scale sparse matrix.

We have proposed an algorithm that restarts only right singular vectors without using singular vectors on both sides of the singular vectors of the small matrix $\tilde{B}_k$ generated inside the AIRLB algorithm. At restarting, our improved implementation executes the QR decomposition for reorthogonalizing the matrix composed of right singular vectors.

Using numerical experiments, we have verified that the average and the maximum singular value errors are reduced compared with a conventional algorithm. With respect to computation time, the matrix–vector operation of a large-scale sparse matrix is dominant, so that only a small difference is produced in any test case.

As future research, we expect to use the bisection and inverse iteration algorithm [13] for SVD of the inner matrix $\tilde{B}_k$ in the AIRLB algorithm.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 17H02858.

References

[1] E. Anderson, et al. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, 1999.

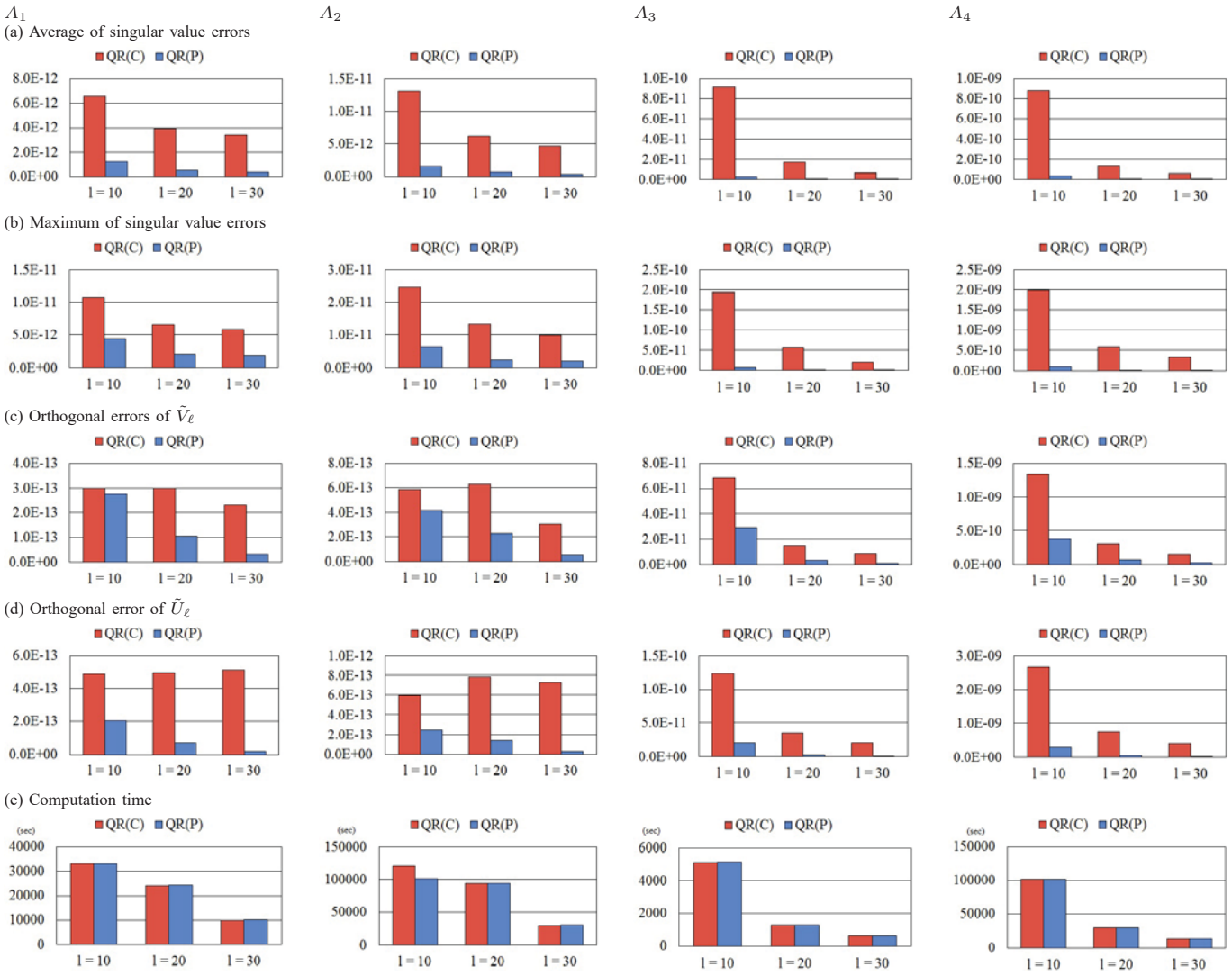

Fig. 1: Performance of truncated SVD. (QR(C) denotes the conventional algorithm restarting with both sides, and QR(P) denotes the proposed algorithm restarting with one side.)

[2] J. Baglama and L. Reichel. Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing* 27(1): 19–42, 2005.

[3] D. Calvetti, et al. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis* 2(1): 1–21, 1994.

[4] J. W. Daniel, et al. Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. *Mathematics of Computation* 30(136): 772–795, 1976.

[5] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific and Statistical Computing* 11(5): 873–912, 1990.

[6] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 4th edition, 2012.

[7] G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis* 2(2): 205–224, 1965.

[8] N. Halko, et al. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Reviews*, 53(2): 217–288, 2011.

[9] Intel Math Kernel Library: Available at https://software.intel.com/en-us/intel-mkl/

[10] (2015). LAPROGNC(Linear Algebra PROGrams in Numerical computation). [Online]. Available: http://www-is.amp.i.kyoto-u.ac.jp/kkimur/LAPROGNC/LAPROGNC.html

[11] C. L. Lawson, et al. Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software* 5(3): 308–323, 1979.

[12] R. B. Lehoucq. The computation of elementary unitary matrices. *ACM Transactions on Mathematical Software* 22(4): 393–400, 1996.

[13] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Society for Industrial and Applied Mathematics, 1998.

[14] G. L. Sleijpen and H. A. Van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM Review*, 42(2): 267–293, 2000.

[15] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, 1965.