Improvement of Computing Partial Singular Value Decomposition for Dense Matrix using Thick-restart-Lanczos Method

Masana Aoki[1], Masami Takata[2], Kinji Kimura[3], 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
[3]Department of Computer Science and Technology, Salesian Polytechnic, Machida, Tokyo, JAPAN

Abstract— *In this paper, we propose a computation method for partial singular value decomposition of dense matrices. Regression analysis can usually be performed using QR decomposition. However, in cases where multicollinearity exists between some independent variables, instead of regression analysis, principal component regression should be performed using larger singular values, and the singular vectors corresponding to the singular values. Observed data in regression analysis is often a large-scale dense matrix. Therefore, a high-speed computational method of partial singular value decomposition is required. The thick-restart-Lanczos (TRL) algorithm is adopted in our proposed method. The TRL method is a computational method for partial eigenvalue decomposition and is implemented in TRLAN. Partial singular value decomposition for sparse matrices can be transformed into partial eigenvalue decomposition by using Takata et al.'s method. This method was developed for principal component analysis for sparse matrices and is adapted to the Implicitly-restarted-Lanczos (IRL) method. In this study, we improve upon Takata et al.'s method by developing the computational method for dense matrices. Further, we compare the conventional method for partial singular value decomposition and our proposed method in TRLAN.*

Keywords: Principal component regression, dense matrix, Thick-restart-Lanczos method, TRLAN

1. Introduction

Regression analysis is performed using large dense matrices from its property of derivation of the problem. Especially, in the case where multicollinearity exists between some independent variables, instead of regression analysis, principal component regression [11] should be performed, not using QR decomposition, but rather using larger singular values and the singular vectors corresponding to the singular values. Therefore, partial singular value decompositions in large dense matrices are required.

The Implicitly-restarted-Lanczos (IRL) method[13] and the Thick-restart-Lanczos (TRL) method[16] are known as computational methods for absolute larger eigenvalues, and the eigenvectors corresponding to the eigenvalues. The IRL and TRL methods are implemented in ARPACK[9] and TRLAN[17]. Since ARPACK and TRLAN have been used for many years, bugs have been removed. Consequently, ARPACK and TRLAN, which have become reliable software, can suitably perform partial eigenvalue decomposition.

Singular value decomposition can be transformed into eigenvalue decomposition by multiplying the given matrix by its transposed matrix. In the eigenvalue decomposition, all eigenvalues are non-negative numbers. Using this transformation, Takata et al. have proposed partial singular value decomposition for sparse matrices using ARPACK[14]. Takata et al.'s method has been introduced into the IRL method to improve ARPACK. Takata et al.'s method has been developed considering the computational order as well as the caches of shared-memory multi-core processors. Therefore, by using OpenMP directives, the cache of CPUs can be utilized efficiently. Then, Takata et al.'s method can achieve much higher performance in parallel computing. On the other hand, in this paper, we target dense matrices because of computations that are required for principal component regression. Hence, to treat dense matrices, we improve Takata et al.'s method. Moreover, since the TRL method is of a higher speed than the IRL method, the proposed method is based on the TRL method.

In Section 2, we introduce the Lanczos[10] and TRL methods, which are known as the Krylov subspace method. In Section 3, we explain a transformation of singular value decomposition to eigenvalue decomposition. In Section 4, we discuss Takata et al.'s method for sparse matrices. In Section 5, we describe the improvement to compute dense matrices. In Section 6, we compare the conventional method with our proposed method in TRLAN.

2. Krylov Subspace Method

The Krylov subspace is a linear subspace and is composed of a matrix $A \in \mathbb{R}^{n\times n}$, an initial vector $\boldsymbol{q_1} \in \mathbb{R}^n(\boldsymbol{q_1} \neq \boldsymbol{0})$, and an iteration number k. Krylov vectors $\boldsymbol{q_1}, A\boldsymbol{q_1}, A^2\boldsymbol{q_1}, \ldots$ are generated by multiplying A by the power of $\boldsymbol{q_1}$. The order-k Krylov subspace is spanned by k Krylov vectors:

$$\mathcal{K}(A, \boldsymbol{q_1}, k) = \mathrm{span}\{\boldsymbol{q}_1, A\boldsymbol{q}_1, \ldots, A^{k-1}\boldsymbol{q}_1\}. \qquad (1)$$

Algorithm 1 Lanczos method

```
1: Generate an initial vector $\boldsymbol{q}_1$, which is set to random number;
2: $\boldsymbol{v}_1 := \boldsymbol{q}_1/|\boldsymbol{q}_1|$;
3: $\beta_0 := 0$;
4: $\boldsymbol{q}_0 := \boldsymbol{0}$;
5: for $j := 1$ to $k$ do
6:     $\mathbf{r}_j := A\boldsymbol{v}_j$;
7:     $\alpha_j := \boldsymbol{v}_j^\top \mathbf{r}_j$;
8:     $\mathbf{r}_j := \mathbf{r}_j - \alpha_j \boldsymbol{v}_j - \beta_{j-1}\boldsymbol{v}_{j-1}$;
9:     $\beta_j := |\mathbf{r}_j|$;
10:    $\boldsymbol{v}_{j+1} := \mathbf{r}_j/\beta_j$;
11: end for
```

The methods for solving linear problems using the Krylov subspace are called the Krylov subspace methods.

2.1 Lanczos method

The Lanczos method is a Krylov subspace method and it reduces a symmetric matrix to a tridiagonal symmetric matrix of a smaller size. By introducing the modified Gram-Schmidt method to the base vector in the Krylov subspace $\mathcal{K}(A, \boldsymbol{q_1}, k)$, orthonormal bases are generated and

$$\begin{aligned} h_{i,j} &= \boldsymbol{v}_i^\top A \boldsymbol{v}_j = \boldsymbol{v}_i^\top A^\top \boldsymbol{v}_j = (A\boldsymbol{v}_i)^\top \boldsymbol{v}_j = \left(\boldsymbol{v}_j^\top (A\boldsymbol{v}_i)\right)^\top \\ &= h_{j,i}, \end{aligned} \tag{2}$$

is satisfied. Thereafter, A is transformed into the tridiagonal symmetric matrix by the Lanczos method. Algorithm 1 is pseudocode. In the k-th step,

$$AV_k = V_k S_k + \beta_k \boldsymbol{v}_{k+1} \boldsymbol{e}_k^\top, \tag{3}$$

is satisfied. Here,

$$S_k := \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdots & \cdots & 0 \\ \beta_1 & \alpha_2 & \ddots & \ddots & & \vdots \\ 0 & \beta_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \beta_{k-1} \\ 0 & \cdots & \cdots & 0 & \beta_{k-1} & \alpha_k \end{bmatrix}. \tag{4}$$

The stopping criterion of the Lanczos method is decided using the Wilkinson theorem[15]. Let $\lambda_j^{(k)} \in \mathbb{R}$ and $\boldsymbol{y}_j^{(k)} \in \mathbf{r}^n$ be set as an eigenvalue with the corresponding normalized eigenvector in S_k. The Lanczos method is satisfied as follows:

$$S_k \boldsymbol{y}_j^{(k)} = \lambda_j^{(k)} \boldsymbol{y}_j^{(k)}. \tag{5}$$

$$\mathbf{x}_j^{(k)} := V_k \boldsymbol{y}_j^{(k)} \in \mathbb{R}^n. \tag{6}$$

Thus,

$$\|\mathbf{x}_j^{(k)}\|_2 = 1. \tag{7}$$

By eq. (3),

$$\begin{aligned} & A\mathbf{x}_j^{(k)} - \lambda_j^{(k)} \mathbf{x}_j^{(k)} \\ &= AV_k \boldsymbol{y}_j^{(k)} - \lambda_j^{(k)} V_k \boldsymbol{y}_j^{(k)} = AV_k \boldsymbol{y}_j^{(k)} - V_k \lambda_j^{(k)} \boldsymbol{y}_j^{(k)} \\ &= AV_k \boldsymbol{y}_j^{(k)} - V_k S_k \boldsymbol{y}_j^{(k)} = (AV_k - V_k S_k) \boldsymbol{y}_j^{(k)} \\ &= \beta_k \boldsymbol{v}_{k+1} \boldsymbol{e}_k^\top \boldsymbol{y}_j^{(k)} = (\boldsymbol{e}_k^\top \boldsymbol{y}_j^{(k)}) \beta_k \boldsymbol{v}_{k+1}. \end{aligned} \tag{8}$$

Moreover, from $\|\boldsymbol{v}_{k+1}\|_2 = 1$, we obtain,

$$\begin{aligned} \|A\mathbf{x}_j^{(k)} - \lambda_j^{(k)} \mathbf{x}_j^{(k)}\|_2 &= \|(\boldsymbol{e}_k^\top \boldsymbol{y}_j^{(k)}) \beta_k \boldsymbol{v}_{k+1}\|_2 \\ &= (\boldsymbol{e}_k^\top \boldsymbol{y}_j^{(k)}) \beta_k \|\boldsymbol{v}_{k+1}\|_2 = (\boldsymbol{e}_k^\top \boldsymbol{y}_j^{(k)}) \beta_k. \end{aligned} \tag{9}$$

Therefore, when $\lambda_i(A)$ $(i = 1, \ldots, n)$ is an eigenvalue in A,

$$\min_i |\lambda_j^{(k)} - \lambda_i(A)| \le \|A\mathbf{x}_j^{(k)} - \lambda_j^{(k)} \mathbf{x}_j^{(k)}\|_2 = (\boldsymbol{e}_k^\top \boldsymbol{y}_j^{(k)}) \beta_k, \tag{10}$$

is satisfied by the Wilkinson theorem. Consequently, by using $(\boldsymbol{e}_k^\top \boldsymbol{y}_j^{(k)})\beta_k$, the stopping criterion can be implemented.

2.2 Thick-restart-Lanczos method

In this section, we introduce the TRL method, which is an improved Lanczos method that uses the restart strategy. In the Lanczos method, until an approximate matrix is obtained, the Krylov subspace is expanded and a new basis vector is added at each iteration. Therefore, as the cost of reorthogonalization increases, it takes a lot more memory and a longer computation time. To decrease the cost of reorthogonalization, the base number of the Krylov subspace is limited. Using the limited subspace, an initial vector of the Krylov subspace is decided at the subsequent time. This operation is called as restart.

The TRL method, which is implemented in TRLAN, is developed as a restart strategy only for real symmetric matrices. The number of desired eigen pairs and the number of bases in the Krylov subspace are set to l and m $(l < m \ll n)$, respectively. In the TRL method, after the Lanczos method performs m iterations, we restart the iteration with a new initial vector $\tilde{\boldsymbol{v}}_{l+1} \in \mathbb{R}^n$. After m iterations in the Lanczos method,

$$AV_m = V_m S_m + \beta_m \boldsymbol{v}_{m+1} \boldsymbol{e}_m^\top, \tag{11}$$

is satisfied by eq. (3). Then, the eigenvalues $\rho_1, \ldots, \rho_m$ and the corresponding eigenvectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$ in S_m are computed as follows:

$$S_m \boldsymbol{y}_i = \rho_i \boldsymbol{y}_i, \quad i = 1, \ldots, m. \tag{12}$$

Among m eigenvalues, approximate eigenvalues and the corresponding normalized eigen vectors are set as $\rho_1, \ldots, \rho_l$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l$, respectively.

$$D_l := \mathrm{diag}(\rho_1, \ldots, \rho_l), \tag{13}$$

$$Y_l := [\boldsymbol{y}_1 \ \ldots \ \boldsymbol{y}_l], \tag{14}$$

are defined. Then, by eq. (12),

$$S_m Y_l = Y_l D_l, \tag{15}$$

is satisfied. By eqs. (11) and (15),

$$\begin{aligned} AV_m Y_l &= V_m S_m Y_l + \beta_m \boldsymbol{v}_{m+1} \boldsymbol{e}_m^\top Y_l \\ &= V_m Y_l D_l + \beta_m \boldsymbol{v}_{m+1} \boldsymbol{e}_m^\top Y_l. \end{aligned} \tag{16}$$

is satisfied. By eq. (16), when we define,

$$\tilde{V}_l := V_m Y_l, \tag{17}$$

$$\boldsymbol{\eta} := \boldsymbol{e}_m^\top Y_l, \tag{18}$$

then

$$A\tilde{V}_l = \tilde{V}_l D_l + \beta_m \boldsymbol{v}_{m+1} \boldsymbol{\eta}. \tag{19}$$

The i-th vector in $\tilde{V}_l$ is set to $\tilde{\boldsymbol{v}}_i$ $(i = 1, \ldots, l)$ and it is satisfied as $\tilde{\boldsymbol{v}}_{l+1} := \boldsymbol{v}_{m+1}$. By the definition of $\tilde{V}_l$, since $\tilde{\boldsymbol{v}}_1, \ldots \tilde{\boldsymbol{v}}_l, \tilde{\boldsymbol{v}}_{l+1}$ are orthogonalized, $\tilde{\boldsymbol{v}}_i$ $(i = l+2, \ldots, m)$, $\tilde{\alpha}_i$ $(i = l+1, \ldots, m)$, $\tilde{\beta}_i$ $(i = l+1, \ldots, m-1)$ can be computed by eq. (19), when the initial vector is set to $\tilde{\boldsymbol{v}}_{l+1}$ and the Lanczos method is restarted. Using,

$$\tilde{V}_m := [\tilde{\boldsymbol{v}}_1 \ \ldots \ \tilde{\boldsymbol{v}}_l \ \tilde{\boldsymbol{v}}_{l+1} \ \ldots \ \tilde{\boldsymbol{v}}_m], \tag{20}$$

$$\tilde{S}_m := \begin{bmatrix} D_l & \beta_m \boldsymbol{\eta}^\top & & & & & O \\ \beta_m \boldsymbol{\eta} & \tilde{\alpha}_{l+1} & \tilde{\beta}_{l+1} & & & & \\ & \tilde{\beta}_{l+1} & \tilde{\alpha}_{l+2} & \tilde{\beta}_{l+2} & & & \\ & & \tilde{\beta}_{l+2} & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & \tilde{\beta}_{m-1} \\ O & & & & & \tilde{\beta}_{m-1} & \tilde{\alpha}_m \end{bmatrix}, \tag{21}$$

$A\tilde{V}_m$ is satisfied as follows:

$$A\tilde{V}_m = \tilde{V}_m \tilde{S}_m + \tilde{\beta}_m \tilde{\boldsymbol{v}}_{m+1} \boldsymbol{e}_m^\top, \tag{22}$$

$$\tilde{V}_m^\top \tilde{V}_m = I. \tag{23}$$

When all elements of $\beta_m \boldsymbol{\eta}$ are sufficient small, the stopping criterion follows the Wilkinson theorem and ρ_i, $\tilde{\boldsymbol{v}}_i$ $(i = 1, \ldots, l)$ are close to the eigenvalue and the eigenvector. Algorithm 2 shows the pseudo code of the TRL Method.

3. Singular Value Decomposition using TRLAN

3.1 Singular value decomposition

A real matrix $B \in \mathbb{R}^{w \times n}$ can decompose using a $w \times w$ orthogonal matrix U and a $n \times n$ orthogonal matrix V as follows:

$$B = U \Sigma V^\top \quad (\sigma_i > 0). \tag{24}$$

Algorithm 2 TRL Method

```
1: Set $m$ and $l$;
2: Input Lanczos decomposition $AV_m = V_m S_m + \beta_m \boldsymbol{v}_{m+1} \boldsymbol{e}_m^\top$;
3: for $i := 1, 2, \cdots$ do
4:     Compute all eigenvalue $\rho_1, \ldots, \rho_m$ and the normalized eigenvectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$ corresponding to the eigenvalue in $S_m$;
5:     Extract the required eigenvalues $\rho_1, \ldots, \rho_l$ and the eigenvectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l$;
6:     $D_l := \mathrm{diag}(\rho_1, \ldots, \rho_l)$;
7:     $Y_l := [\boldsymbol{y}_1 \ \ldots \ \boldsymbol{y}_l]$;
8:     $\tilde{V}_l := V_m Y_l$;
9:     $\boldsymbol{\eta} := \boldsymbol{e}_m^\top Y_l$;
10:    $\tilde{\boldsymbol{v}}_{l+1} := \boldsymbol{v}_{m+1}$;
11:    $\tilde{\mathbf{r}}_{l+1} := A\tilde{\boldsymbol{v}}_{l+1}$;
12:    $\tilde{\alpha}_{l+1} := \tilde{\boldsymbol{v}}_{l+1}^\top \tilde{\mathbf{r}}_{l+1}$;
13:    $\tilde{\mathbf{r}}_{l+1} := \tilde{\mathbf{r}}_{l+1} - \sum_{j=1}^{l} \beta_m \boldsymbol{\eta}(j) \tilde{\boldsymbol{v}}_j - \tilde{\alpha}_{l+1} \tilde{\boldsymbol{v}}_{l+1}$;
14:    $\tilde{\beta}_{l+1} := |\tilde{\mathbf{r}}_{l+1}|$;
15:    $\tilde{\boldsymbol{v}}_{l+2} := \tilde{\mathbf{r}}_{l+1} / \tilde{\beta}_{l+1}$;
16:    $\tilde{V}_{l+1} := [\tilde{V}_l \ \tilde{\boldsymbol{v}}_{l+1}]$;
17:    $\tilde{S}_{l+1} := \begin{bmatrix} D_l & \beta_m \boldsymbol{\eta}^\top \\ \beta_m \boldsymbol{\eta} & \tilde{\alpha}_{l+1} \end{bmatrix}$;
18:    Adopt $A\tilde{V}_{l+1} = \tilde{V}_{l+1} \tilde{S}_{l+1} + \tilde{\beta}_{l+1} \tilde{\boldsymbol{v}}_{l+2} \boldsymbol{e}_{l+1}^\top$ to the Lanczos method at $m - l - 1$ times;
19: end for
```

Here,

$$r_b := \operatorname{rank} B, \tag{25}$$

$$\Sigma := \begin{bmatrix} \sigma_1 & 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \cdots & \cdots & 0 & \sigma_{r_b} & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

$$(\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{r_b} > 0) \tag{26}$$

U and V satisfy

$$U^\top U = UU^\top = I_w, \tag{27}$$

$$V^\top V = VV^\top = I_n. \tag{28}$$

Here, I_w and I_n are $w \times w$ and $n \times n$ identity matrices, respectively. Non-negative real numbers $\sigma_1, \ldots, \sigma_{r_b}$, U, and V mean the singular values, a left singular vector, and a right singular vector, respectively. Eq. (24) is converted to

$$B\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i, \tag{29}$$

$$B^\top \boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i \quad (i = 1, \ldots, r_b), \tag{30}$$

where $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ mean the i-th vector in U and V, respectively.

3.2 Transformation into an eigenvalue problem

By eqs. (29) and (30),

$$B^{(r)\top} B^{(r)} \boldsymbol{v}_i^{(r)} = B^{(r)\top} \left(\sigma_i^{(r)} \boldsymbol{u}_i^{(r)}\right) = \sigma_i^{(r)} \left(B^{(r)\top} \boldsymbol{u}_i^{(r)}\right) = \sigma_i^{(r)2} \boldsymbol{v}_j^{(r)}, \quad (31)$$

$$\boldsymbol{u}_i^{(r)} = \frac{B^{(r)} \boldsymbol{v}_i^{(r)}}{||B^{(r)} \boldsymbol{v}_i^{(r)}||_2} \quad (i = 1, \ \dots, \ r_{b^{(r)}}). \quad (32)$$

are satisfied. Here, $B^{(r)}$, $\sigma_i^{(r)}$, $\boldsymbol{u}_i^{(r)}$, $\boldsymbol{v}_i^{(r)}$, and $r_{b^{(r)}}$ are a real rectangular matrix, the i-th singular value, the i-th left singular vector, the i-th right singular vector, and $\operatorname{rank} B^{(r)}$, respectively. Using eq. (31), a singular value decomposition in $B^{(r)}$ can be transformed into the eigenvalue problem in $B^{(r)\top} B^{(r)}$. $\sigma_i^{(r)2}$ is equal to the eigenvalue in $B^{(r)\top} B^{(r)}$. Therefore, by using TRLAN with $B^{(r)\top} B^{(r)}$, a singular value $\sigma_i^{(r)}$ and a right singular vector $\boldsymbol{v}_i^{(r)}$ in $B^{(r)}$ can be computed. Then, by eq. (32), the corresponding left singular vector $\boldsymbol{u}_i^{(r)}$ is obtained.

In a real rectangular matrix $B^{(c)} \in \mathbb{R}^{w \times n}$ $(w < n)$, $B^{(c)\top}$ is equal to $B^{(r)}$. Let $\sigma_i^{(c)}$, $\boldsymbol{u}_i^{(c)}$, and $\boldsymbol{v}_i^{(c)}$ be set as a singular value, a left singular vector, and a right singular vector in $B^{(c)}$, respectively.

$$B^{(c)} B^{(c)\top} \boldsymbol{u}_i^{(c)} = B^{(c)} \left(\sigma_i^{(c)} \boldsymbol{v}_i^{(c)}\right) = \left(B^{(c)} \boldsymbol{v}_i^{(c)}\right) \sigma_i^{(c)} = \boldsymbol{u}_j^{(r)} \sigma_i^{(c)2}, \quad (33)$$

$$\boldsymbol{v}_i^{(c)} = \frac{B^{(c)\top} \boldsymbol{u}_i^{(c)}}{||B^{(c)\top} \boldsymbol{u}_i^{(c)}||_2} \quad (i = 1, \ \dots, \ r_{b^{(c)}}). \quad (34)$$

are satisfied. Consequently, by eqs. (33) and (34), the singular value decomposition in $B^{(c)}$ can be transformed into the eigenvalue problem in $B^{(c)} B^{(c)\top}$ like $B^{(r)}$.

4. Takata et al.'s Method for Sparse Matrix

4.1 Sparse matrix

In the case of a $w \times n$ ($w \geq n$ and $w < n$) rectangular matrix $B^{(r)}$, these elements should be stored in CRS and CCS formats, respectively. The CRS and CCS formats are the most general [3]. These formats require no assumptions about sparse matrices and do not contain unnecessary elements.

The CRS format stores only non-zero elements of the matrix rows sequentially. If a non-symmetric sparse matrix $B^{(S)}$ is given, we write the matrix as three vectors $\mathbf{val}_R$, **col_ind**, and **row_ptr**. $\mathbf{val}_R$ is a vector of floating-point numbers and stores the value of the non-zero elements of the given matrix $A^{(S)}$ based on a row-wise traversal. **col_ind** is a vector of integers indicating the column indices of the elements in $\mathbf{val}_R$. **row_ptr** is also a vector of integers indicating the locations in $\mathbf{val}_R$ that start a row. The last entry of **row_ptr** indicates the number of non-zero elements in the matrix $B^{(S)}$.

The CCS format is equivalent to the CRS format except that the CCS format traverses the non-zero elements of $B^{(S)}$ column-wise. In other words, the CCS format can be interpreted as the CRS format for $B^{(S)\top}$. The CCS format is composed of the three vectors $\mathbf{val}_R$, **col_ind**, and **row_ptr**. $\mathbf{val}_R$, which is a vector of floating-point numbers, stores the value of the non-zero elements of the given matrix $B^{(S)}$ based on a column-wise traversal. **row_ind**, which is a vector of integers, indicates the row indices of the elements in $\mathbf{val}_R$. **col_ptr**, which is a vector of integers, indicates the locations in $\mathbf{val}_R$ that start a column. The last entry of **col_ptr** indicates the number of non-zero elements in the matrix $B^{(S)}$.

As an example, consider the non-symmetric matrix $B^{(S')}$ defined by

$$B^{(S')} = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 0 \\ 0 & 2 & 0 & -1 & 0 & 3 \\ 2 & 7 & 3 & 2 & 6 & 0 \\ 0 & 3 & 8 & 4 & 0 & 0 \\ 3 & 5 & 0 & 9 & 5 & 9 \\ 0 & 0 & 0 & 0 & 2 & 6 \end{bmatrix}. \quad (35)$$

The CRS format for $B^{(S')}$ is

$$\mathbf{val}_R = \{1, 3, 2, -1, 3, 2, \cdots, 5, 9, 2, 6\}, \quad (36)$$

$$\mathbf{col\_ind} = \{1, 5, 2, 4, 6, 1, \cdots, 5, 6, 5, 6\}, \quad (37)$$

$$\mathbf{row\_ptr} = \{1, 3, 6, 11, 14, 19, 20\}. \quad (38)$$

The CCS format for $B^{(S')}$ is

$$\mathbf{val}_C = \{1, 2, 3, 2, 7, 3, \cdots, 2, 3, 9, 6\}, \quad (39)$$

$$\mathbf{row\_ind} = \{1, 3, 5, 2, 3, 4, \cdots, 6, 2, 5, 6\}, \quad (40)$$

$$\mathbf{row\_ptr} = \{1, 4, 8, 10, 14, 18, 20\}. \quad (41)$$

4.2 Singular value decomposition using ARPACK

The discussion of the data format in $B^{(c)} B^{(c)\top}$ is the same as that of $B^{(r)\top} B^{(r)}$. Hence, we explain only the case of $B^{(r)} \in \mathbb{R}^{w \times n}$ $(w \geq n)$.

In the case that a sparse matrix is decomposed using the IRL method, a computation of $B^{(r)\top} B^{(r)} \mathbf{x}$ is needed. As a reasonable computational method, Takata et al.'s method has been proposed. Algorithm 3 shows the pseudo code of Takata et al.'s method. Here, $B^{(r)}(i, :)$ is the i-th row vector of $B^{(r)}$. Therefore, in algorithm 3, $B^{(r)}(i, :)$ in line 4 is the same data of $B^{(r)\top}(:, i)$ in line 5. Consequently, in the case where n is smaller than cache size, algorithm 3 can be computed efficiently, since $\boldsymbol{b}_i$ can be stored in cache. Thus, in the

Algorithm 3 Takata et al.'s method

```
r = 0;
#omp parallel for private($t$) reduction(+:r)
for $i = 1$ to $n$ do
    $t = \langle \boldsymbol{b}_i, \mathbf{x} \rangle$ ($\boldsymbol{b}_i = B(i,:)$);
    $\mathbf{r} = \mathbf{r} + t\boldsymbol{b}_i^\top$;
end for
#omp end parallel for
```

Algorithm 4 Pseudo code of Takata et al.'s method

```
r = 0;
#omp parallel for private($t$) reduction(+:r)
for $i = 1$ to $n$ do
    $t = 0$;
    for $j$ = row_ptr($i$) to row_ptr($i+1$) − 1 do
        $t = t$ + val$_R(j)$ × x(col_ind($j$));
    end for
    for $j$ = row_ptr($i$) to row_ptr($i+1$) − 1 do
        r(col_ind($j$)) = r(col_ind($j$)) + val$_R(j)$ × $t$;
    end for
end for
#omp end parallel for
```

case of a CPU with a large cache size, ${B^{(r)}}^\top B^{(r)}\mathbf{x}$ can be computed fast. Takata et al.'s method stores a sparse matrix using CRS format and is implemented using Fortran [19]. Algorithm 4 shows the pseudo code in the case of CRS format. For double precision floating point number, Takata et al.'s method is effective in the case that cache size is more $n \times 8$ bytes in theory. On the other hand, in the case that either $B^{(r)}$ can be stored in caches, or $B^{(r)}(i,:)$ cannot be stored in caches, the computation time does not change much.

5. Proposed method

To employ the TRL method, ${B^{(r)}}^\top B^{(r)}\mathbf{x}$ can be computed by using two matrix-vector operations at each iteration. Algorithm 5 shows the pseudo code using `DGEMV` in the Intel Math Kernel Library [8]. In this paper, algorithm 5 is called the conventional method(I). The conventional method(I) takes a long computation time. Therefore, the conventional method(I) is paralleled to the conventional method(II) by using OpenMP. Algorithm 6 shows the pseudo code of the conventional method(II) using `DDOT` and `DAXPY` in the Intel Math Kernel Library. In the conventional method(I) and (II), if the code is implemented according to the description of $\tilde{\mathbf{x}} = B^{(r)}\mathbf{x}$ and $\mathbf{r} = {B^{(r)}}^\top\tilde{\mathbf{x}}$, the two matrix-vector computations are performed separately. In the case that $B^{(r)}$ cannot be stored in caches, the computation time becomes longer.

In this paper, by improving Takata et al.'s method, we aim for improvement of the cache hit ratio. Takata et al.'s method targets a sparse matrix, which is stored using the CRS format. Thus, line 4 and 5 in algorithm 3 is expanded to a block of code that ranges from line 4 to line 10 in algorithm 4. On the other hand, in this paper, since dense matrices are targeted, the expansion in algorithm 4 is not needed. Therefore, we propose an improvement method based on Takata et al.'s method. In the proposed method, by using loop fusion, the conventional method(II) is accelerated. Algorithm 7 shows the pseudo code in the proposed method.

Algorithm 5 Conventional method(I)

```
CALL DGEMV for $\tilde{\mathbf{x}} = B\mathbf{x}$;
CALL DGEMV for $\mathbf{r} = B^\top\tilde{\mathbf{x}}$;
```

Algorithm 6 Conventional method(II)

```
#omp parallel for;
for $i = 1$ to $n$ do
    $\tilde{\mathbf{x}}_i$ = DDOT($\boldsymbol{b}_i, \mathbf{x}$) ($\boldsymbol{b}_i = B(i,:)$);
end for
#omp end parallel for;
r = 0;
#omp parallel for reduction(+:r)
for $i = 1$ to $n$ do
    CALL DAXPY for $\mathbf{r} = \mathbf{r} + \tilde{\mathbf{x}}_i\boldsymbol{b}_i^\top$;
end for
#omp end parallel for
```

Algorithm 7 Proposed method for dense matrix

```
r = 0;
#omp parallel for reduction(+:r)
for $i = 1$ to $n$ do
    $\tilde{\mathbf{x}}_i$ = DDOT($\boldsymbol{b}_i, \mathbf{x}$) ($\boldsymbol{b}_i = B(i,:)$);
    CALL DAXPY for $\mathbf{r} = \mathbf{r} + \tilde{\mathbf{x}}_i\boldsymbol{b}_i^\top$;
end for
#omp end parallel for
```

6. Experiments

6.1 Evaluation of error

Let σ_i $(i = 1, \ldots, r_b)$ be set as a singular value in a matrix $B \in \mathbb{R}^{w\times n}$ $(w \geq n)$, where $r_b = \operatorname{rank} B$. The augmented matrix of B is as follows:

$$M = \begin{bmatrix} O & B \\ B^\top & O \end{bmatrix} \in \mathbb{R}^{(w+n)\times(w+n)}. \tag{42}$$

From the definition, M is $w+n$ square matrix. By using an eigenvalue λ_i $(i = 1, \ldots, w+n)$ in M,

$$\begin{aligned} &\lambda_1 = \sigma_1, \ \ldots, \ \lambda_{r_b} = \sigma_{r_b}, \ \lambda_{r_b+1} = -\sigma_1, \ \ldots, \\ &\lambda_{2r_b} = -\sigma_{r_b}, \ \lambda_{2r_b+1} = 0, \ \ldots, \ \lambda_{w+n} = 0. \end{aligned} \tag{43}$$

Let $(\tilde{\sigma}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i)$ $(i = 1, \ldots, r_b)$ be set as the computation result of singular value decomposition in B. By using a left singular vector $\tilde{\boldsymbol{u}}_i$ and a right singular vector $\tilde{\boldsymbol{v}}_i$, $\tilde{\boldsymbol{x}}_i \in \mathbb{R}^{w+n}$ is determined as follows:

$$\tilde{\boldsymbol{x}}_i := \frac{1}{\sqrt{\|\tilde{\boldsymbol{u}}_i\|^2 + \|\tilde{\boldsymbol{v}}_i\|^2}} \begin{bmatrix} \tilde{\boldsymbol{u}}_i \\ \tilde{\boldsymbol{v}}_i \end{bmatrix}. \tag{44}$$

From the definition, $\|\tilde{\boldsymbol{x}}_i\| = 1$ is satisfied. By the Wilkinson theorem,

$$\begin{aligned} \min_j |\tilde{\sigma}_i - \lambda_j| &\leq \|M\tilde{\boldsymbol{x}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{x}}_i\| \\ &= \sqrt{\|M\tilde{\boldsymbol{x}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{x}}_i\|^2} \\ &= \frac{\sqrt{\|A\tilde{\boldsymbol{v}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{u}}_i\|^2 + \|A^\top\tilde{\boldsymbol{u}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{v}}_i\|^2}}{\sqrt{2}}, \end{aligned} \tag{45}$$

is satisfied. By eq. (43), it is not guaranteed that λ_j is the singular value in B. In principal component regression, since only larger singular values are required, λ_j, which minimizes $|\tilde{\sigma}_i - \lambda_j|$, is regarded as the singular value in B. Hence, by eq. (45), in the evaluation of error, we use $\sqrt{\|A\tilde{\boldsymbol{v}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{u}}_i\|^2 + \|A^\top\tilde{\boldsymbol{u}}_i - \tilde{\sigma}_i\tilde{\boldsymbol{v}}_i\|^2}/\sqrt{2}$.

6.2 Environment

In the numerical experiments with TRLAN, base number m in the Krylov subspace is set to $m := 2l$. Using $100,000 \times 10,000$ real matrices B_1 and B_2, TRLAN with the conventional method(II) and TRLAN with the proposed method are compared with respect to computation time. Singular values in B_1 and B_2 decrease exponentially. Condition numbers in B_1 and B_2 are sufficiently small and large, respectively.

- B_1:

$$\sigma_i := \sqrt{\epsilon^{\frac{i-1}{10000-1}}}. \tag{46}$$

  The condition number in B_1:

$$\frac{\sigma_1}{\sigma_{10000}} = 6.7108864 \times 10^7. \tag{47}$$

- B_2:

$$\sigma_i := \epsilon^{\frac{i-1}{10000-1}}. \tag{48}$$

  The condition number in B_2:

$$\frac{\sigma_1}{\sigma_{10000}} = 4.5035996 \times 10^{15}. \tag{49}$$

Here, the matrices are column full rank. Singular values are represented as σ_i $(i = 1, \ldots, 10,000)$. ϵ is machine epsilon, which is $\epsilon = 2.220 \times 10^{-16}$ in the case of double precision. The condition number (47) is small from the view point of the Cholesky QR decomposition[18].

Experimental environment is shown:

- CPU: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz $\times$ 2
- Mem: 128GB
- OS: Ubuntu 16.04.3 LTS
- Compiler: icc version 18.0.1, ifort version 18.0.1
- Option: -qopenmp -ipo -xHOST -O3 -prec-sqrt -prec-div -fp-model precise
- Library: Intel MKL 2018

Table 1: Computation time in B_1

l	Conventional method(II)	Proposed method
10	55.686	35.819
50	74.437	50.550
100	104.914	72.036
500	281.183	215.116

Table 2: Computation time in B_2

l	Conventional method(II)	Proposed method
10	39.246	25.690
50	56.579	39.150
100	78.7035	56.841
500	268.065	199.513

Table 3: Accuracy in B_1

l	Conventional method(II)	Proposed method
10	5.04E-15	4.37E-15
50	4.07E-15	4.43E-15
100	6.62E-15	6.62E-15
500	4.38E-15	4.38E-15

Table 4: Accuracy in B_2

l	Conventional method(II)	Proposed method
10	3.41E-15	4.41E-15
50	4.13E-15	2.70E-15
100	3.41E-15	3.41E-15
500	5.86E-15	5.72E-15

6.3 Result and consideration

Table 1 and 2 show the computation time, which is taken using TRLAN with the conventional method(II) and the proposed method, in B_1 and B_2, respectively. Table 3 and 4 show the accuracy in B_1 and B_2, respectively.

By tab. 1 and 2, the computation time in TRLAN with the proposed method is 75% of that with the conventional method(II). Although the computational cost of the conventional method(II) is nearly equal to that of the proposed method, the computation time is different. Thus, cache hit ratio can be improved by using the proposed method.

By tab. 3 and 4, the accuracy in both methods is not different. The accuracy is sufficient for principal component regression, in which multicollinearity exists between some independent variables.

7. Conclusion

Principal component regression, which is computed using large dense matrices from its property of derivation of the problem, needs larger singular values and the corresponding singular vectors. We adopted the TRL method, which is implemented in TRLAN, for partial singular value decomposition. TRLAN is a suitable software for partial eigenvalue decomposition. Partial singular value decomposition was performed using the TRL method because the singular value decomposition can be transformed into an eigenvalue problem. In the case that the transformation to an eigenvalue problem is performed using TRLAN, two matrix-vector operations are needed at each iteration. Since the input data for regression analysis is given by a dense matrix, all elements in the matrix cannot be stored in caches when the dimension size is large. To improve the cache hit ratio, we propose a partial singular value decomposition using the TRL method. The proposed method is based on Takata et al.'s method, of which the target is sparse matrices. Numerical experiments show that the computation time in TRLAN with the proposed method is 75% of that with the conventional method(II). The accuracy in TRLAN is sufficient for principal component regression. In a future study, to achieve high-accuracy computations, the proposed method will be adopted as a stopping criterion in iterative methods for the least square problem.

Acknowledgment

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

References

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

[2] Calvetti, D., Reichel, L., and Sorensen, D. C.: *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electronic Transactions on Numerical Analysis, 2(1), pp.1–21 (1994).

[3] Dongarra, J.: Templates for the Solution of Linear Systema: Building Blocks for Iterative Methods, <http://netlib.org/linalg/html_templates/node89.html> (1995).

[4] Duff, I, Grimes, R., and Lewis, J.: *Sparse matrix test problems*, ACM Trans. Math. Soft., vol.15, pp.1-14(1989).

[5] Golub, G. H., and Kahan, W.: *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), pp.205–224 (1965).

[6] Golub G. H., and van Loan, C. F.: Matrix Computations, Baltimore, MD, USA: Johns Hopkins University Press(1996).

[7] Halko, N., Martinsson, P. G., and Tropp, J. A.: *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Reviews, 53(2), pp.217–288 (2011).

[8] Intel Math Kernel Library, <https://software.intel.com/en-us/mkl> (1995).

[9] Lehoucq, R. B., Sorensen, D. C., and Yang, C.: ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods.
<http://www.caam.rice.edu/software/ARPACK> (1998).

[10] Lanczos, C.: *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bureau Standards, Sec, vol.B, no.45, pp.255-282(1950).

[11] Mevik, B.H., and Wehrens, R.: *The pls package: principal component and partial least squares regression in R*, J. Statistical Software, vol.18(2007).

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

[13] Sorensen, D. C., Calvetti, D., and Reichel, L.: *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Elect. Trans. Numer. Anal., vol.2, pp.1-21(1994).

[14] Takata, M., Araki, S., Kimura, K., Fujii, Y., and Nakamura, Y.: *Implementation of computing partial singular value decomposition for principal component analysis using ARPACK*, IPSJ Transactions on Mathematical Modeling and Its Applications, vol.11, no.1, pp.37-44(2018).

[15] Wilkinson, J. H. D.: The Algebraic Eigenvalue Problem, Clarendon Press(1965).

[16] Wu, K., and Simon, H.: *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. and Appl., 22(2), pp.602-616(2000).

[17] Wu, J.: Research on Eigenvalue Computations, <https://sdm.lbl.gov/ kewu/ps/trlan-ug.pdf>, (1999).

[18] Yamamoto, Y., Nakatsukasa, Y., Yanagisawa, Y., and Fukaya, T.: *Roundoff error analysis of the CholeskyQR2 algorithm in an oblique inner product*, JSIAM Letters, vol. 8, pp. 5-8(2015).

[19] <http://www-is.amp.i.kyoto-u.ac.jp/kkimur/LAPROGNC/LAPROGNC-j.html>, Kyoto University, Nakamura-Tsujimoto Laboratory(2014).