JSIAM Letters Vol.2 (2010) pp.91–94 ©2010 Japan Society for Industrial and Applied Mathematics JSIAM Letters

Proposal and efficient implementation of multiple division divide-and-conquer algorithm for SVD

Yutaka Kuwajima[1], Youichiro Shimizu[1] and Takaomi Shigehara[1]

[1] Graduate School of Science and Engineering, Saitama University, 255 Shimo-Okubo, Sakura-ku, Saitama City, Saitama 338-8570, Japan

E-mail *kuwa@mail.saitama-u.ac.jp*

Received March 31, 2010, Accepted May 15, 2010

Abstract

We propose a divide-and-conquer algorithm with multiple division for singular value decomposition (SVD). The algorithm turns out to be efficient for reducing the execution time in the case that the deflation occurrence rate of the input matrix is low, which is exactly the case that the standard divide-and-conquer algorithm (DC2-SVD) with division number two requires $O(n^3)$ arithmetic operations. Here n is the size of the input matrix. The comparison with DC2-SVD as well as another up-to-date algorithm I-SVD is made through numerical experiment.

Keywords singular value decomposition, divide-and-conquer method, multiple division

Research Activity Group Algorithms for Matrix / Eigenvalue Problems and their Applications

1. Introduction

Consider the singular value decomposition (SVD) of an upper-bidiagonal matrix $B \in \mathbf{R}^{n\times(n+1)}$ with diagonals a_j and subdiagonals b_j $(j = 1, \ldots, n)$. The divide-and-conquer algorithm with division number $k = 2$ (DC2-SVD) [1] is a well-established numerical algorithm for this purpose. Although DC2-SVD keeps the numerical accuracy at high level, the numerical cost of DC2-SVD is not necessarily low, and it requires $O(n^3)$ arithmetic operations in a general case such that the deflation occurrence rate is low. To overcome this, we propose, in this paper, the multiple division divide-and-conquer algorithm (DCK-SVD) for SVD. DCK-SVD is an application of multiple division divide-and-conquer algorithm for a real symmetric tridiagonal eigenproblem [2,3], and it can reduce the operation count to $3k/[2(k^2-1)]$, compared to DC2-SVD, where k ($\ll n$) is the division number in DCK-SVD. In numerical experiment, DCK-SVD is compared with DC2-SVD in LAPACK [4] as well as I-SVD [5], which is an up-to-date algorithm for SVD of upper-bidiagonal matrices, introduced in a broad view of applying the integrable systems to numerical computation.

2. Framework of DCK-SVD

Denote the set of l-dimensional real vectors and the set of $l_1 \times l_2$ real matrices by $\mathbf{R}^l$ and $\mathbf{R}^{l_1\times l_2}$, respectively. For simplicity, we assume that the column number $n+1$ of B is a multiple of the division number k ($n+1 = mk, m \in \mathbf{N}$). To save the space, we write down some equations for $k = 3$. Generalization to general k is obvious. Let I_l, $\boldsymbol{e}_j^{(l)}$ and $\mathbf{0}_l$ be the l-dimensional unit matrix, its j-th column vector and l-dimensional zero vector, respectively. Without loss of generality, we assume that B is irreducible ($b_j \neq 0$; $j = 1, \ldots, n$) in the following. We also assume that the singular values (SVLs) of B are labeled in ascending order $0 < \sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_n$.

We first divide B into k blocks in the obvious way as

$$B \equiv \begin{pmatrix} B_1 & & \\ \boldsymbol{\alpha}_1^T & \boldsymbol{\beta}_1^T & \\ & B_2 & \\ & \boldsymbol{\alpha}_2^T & \boldsymbol{\beta}_2^T \\ & & B_3 \end{pmatrix} \in \mathbf{R}^{n\times(n+1)}, \tag{1}$$

where $\boldsymbol{\alpha}_i = a_{mi}\boldsymbol{e}_m^{(m)}$, $\boldsymbol{\beta}_i = b_{mi}\boldsymbol{e}_1^{(m)}$ $(i = 1, \ldots, k-1)$ and $B_i \in \mathbf{R}^{(m-1)\times m}$ $(i = 1, \ldots, k)$. Suppose that we have the SVD $B_i = U_i\Sigma_i V_i^T$ of B_i, where $\Sigma_i = (D_i, \mathbf{0}_{m-1}) \in \mathbf{R}^{(m-1)\times m}$ with a diagonal matrix $D_i \in \mathbf{R}^{(m-1)\times(m-1)}$, and $U_i \in \mathbf{R}^{(m-1)\times(m-1)}$, $V_i \in \mathbf{R}^{m\times m}$ are orthogonal matrices. Then, we have $B = U'MV'^T$, where

$$M \equiv \begin{pmatrix} \Sigma_1 & & \\ \boldsymbol{\alpha}_1'^T & \boldsymbol{\beta}_1'^T & \\ & \Sigma_2 & \\ & \boldsymbol{\alpha}_2'^T & \boldsymbol{\beta}_2'^T \\ & & \Sigma_3 \end{pmatrix} \in \mathbf{R}^{n\times(n+1)} \tag{2}$$

with $\boldsymbol{\alpha}_i' = V_i^T\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i' = V_{i+1}^T\boldsymbol{\beta}_i$ $(i = 1, \ldots, k-1)$, and

$$\begin{cases} U' \equiv \{\bigoplus_{i=1}^{k-1}[U_i \oplus (1)]\} \oplus U_k \in \mathbf{R}^{n\times n}, \\ V' \equiv \bigoplus_{i=1}^{k} V_i \in \mathbf{R}^{(n+1)\times(n+1)} \end{cases} \tag{3}$$

are orthogonal matrices. So, if we find the SVD $M = \hat{U}\Sigma\hat{V}^T$ of M, we obtain the SVD $B = U\Sigma V^T$ of B, where $U \equiv U'\hat{U}$, $V \equiv V'\hat{V}$ are orthogonal. The method for computing the SVD of M is separately discussed later in this section.

Since B_i $(i = 1, \ldots, k)$ has a form similar to B, the SVD of B_i is obtained by a recursive usage of DCK-SVD, in principle. In the present implementation, however, we use the DC2-SVD routine `DBDSDC` in LAPACK for this purpose.

The problem of finding the SVD of M in (2) is equivalent to the positive semidefinite real symmetric eigen-

problem of

$$M^T M = D^2 + SS^T \in \mathbf{R}^{(n+1)\times(n+1)}, \tag{4}$$

where $D \equiv \bigoplus_{i=1}^{k}[D_i \oplus (0)]$ is diagonal and $S \in \mathbf{R}^{(n+1)\times(k-1)}$ is determined from (2) in the obvious way. The matrix in (4) has such a form as a diagonal matrix plus low-rank perturbation, and its spectral decomposition can be obtained within $O(n^2)$ arithmetic operations by the method in [3].

2.1 Singular values of M

If the perturbation is of rank one ($k = 2$), then the eigenproblem for (4) is easily solved with high numerical accuracy [6]. By repeating this procedure $k-1$ times in a general case, we first obtain the eigenvalues $\lambda_j \geq 0$ of $M^T M$, leading to the SVLs $\sigma_j = \sqrt{\lambda_j}$ of M ($j = 1, \ldots, n$).

2.2 Singular vectors of M

To obtain the eigenvectors of $M^T M$, define

$$F(\lambda) \equiv \begin{pmatrix} \tilde{F}(\lambda) & S^T W \\ W^T S & W^T(\lambda I_{n+1} - D^2)W \end{pmatrix}, \quad \lambda \in \mathbf{R} \tag{5}$$

with $\tilde{F}(\lambda) \equiv I_{k-1} - S^T(I_{n+1} - WW^T)(\lambda I_{n+1} - D^2)^{-1}S$, where $W \equiv (\boldsymbol{e}_{i(1)}^{(n+1)}, \ldots, \boldsymbol{e}_{i(s)}^{(n+1)})$ with s distinct positive integers $i(1), \ldots, i(s)$ equal to or less than $n+1$. If λ is an eigenvalue of $M^T M$ ($\lambda = \lambda_j$), the matrix $F(\lambda_j)$ becomes singular, and the eigenvector corresponding to the eigenvalue λ_j of $M^T M$ (the right singular vector (SVC) corresponding to the SVL σ_j of M) is given by

$$\boldsymbol{v}_j = [(I_{n+1} - WW^T)(\lambda_j I_{n+1} - D^2)^{-1}S, -W]\boldsymbol{x}_j \tag{6}$$

with $\boldsymbol{x}_j \in \mathrm{Ker}\, F(\lambda_j)$ ($\boldsymbol{x}_j \neq \mathbf{0}_{k+s-1}$). The computation of $\boldsymbol{x}_j$ is carried out by the inverse power method. On the method for a choice of W in $F(\lambda_j)$ for keeping the numerical accuracy, see [3]. The left SVC corresponding to the SVL σ_j is given by

$$\boldsymbol{u}_j = \sigma_j^{-1} M \boldsymbol{v}_j = \sigma_j^{-1}[\tilde{D}\boldsymbol{v}_j + (\boldsymbol{e}_m^{(n)}, \ldots, \boldsymbol{e}_{(k-1)m}^{(n)}, O)\boldsymbol{x}_j] \tag{7}$$

with $\tilde{D} \equiv \{\bigoplus_{i=1}^{k-1}[D_i \oplus (0)]\} \oplus D_k$, where the second equality follows from (5) and (6).

3. Technical notes on implementation

3.1 Removal of SVLs $\sigma \simeq 0$ of B

Since B is irreducible, B does not have zero SVL. However, it might have an arbitrarily tiny SVL in general. Such tiny SVLs are largely removed by the following procedure, where c_1 is a small cut-off parameter.

1) By applying successive Givens transformations from the right, B is reduced to an upper-bidiagonal form with the zero vector in the last column. This corresponds to RQ decomposition of B; $B = RQ_r$ with upper-bidiagonal matrix $R \in \mathbf{R}^{n\times(n+1)}$ and orthogonal matrix $Q_r \in \mathbf{R}^{(n+1)\times(n+1)}$.
2) Define $R' \in \mathbf{R}^{n\times(n-1)}$ by removing the first and the last columns from R.
3) By applying successive Givens transformations from the left, R' is reduced to a lower-bidiagonal form with the zero vector in the last row. This corresponds to QL decomposition of R'; $R' = Q_l L'$ with orthogonal matrix $Q_l \in \mathbf{R}^{n\times n}$ and lower-bidiagonal matrix $L' \in \mathbf{R}^{n\times(n-1)}$.
4) Define

$$\tilde{B} \equiv Q_l^T B Q_r^T = \begin{pmatrix} B' & \mathbf{0}_{n-1} \\ \tilde{b}_{n,1}\boldsymbol{e}_1^{(n)T} & 0 \end{pmatrix} \in \mathbf{R}^{n\times(n+1)},$$

where $B' \in \mathbf{R}^{(n-1)\times n}$ is upper-bidiagonal. In general, fill-in might occur only in the bottom-left element of $\tilde{B}$ by this procedure.
5) If $|\tilde{b}_{n,1}| < c_1\|B\|_F$, set $\tilde{b}_{n,1} = 0$. Here $\|\cdot\|_F$ is the Frobenius norm.

Repeating 1)–5) for the output B' successively, we can further remove tiny SVLs from B. Note that though the smallness of the fill-in indicates the existence of a small SVL for B, the converse is not true. As a result, all tiny SVLs cannot be removed by the above procedure in general.

3.2 Deflation

By using suitable permutation matrices $P_u \in \mathbf{R}^{n\times n}$ and $P_v \in \mathbf{R}^{(n+1)\times(n+1)}$, M in (2) is reduced to

$$M_1 = P_u^T M P_v = \begin{pmatrix} D' & O \\ S_1^T & S_2^T \end{pmatrix} \in \mathbf{R}^{n\times(n+1)}, \tag{8}$$

where $S_1 \equiv (s_{i,j}^{(1)}) \in \mathbf{R}^{n_1\times(k-1)}$, $S_2 \in \mathbf{R}^{k\times(k-1)}$ and $D' \equiv \mathrm{diag}(d_1', \ldots, d_{n_1}') \in \mathbf{R}^{n_1\times n_1}$ ($0 < d_1' \leq \cdots \leq d_{n_1}'$) with $n_1 = n - k + 1$. The SVD of M_1 is essentially the same as that of M. So we consider M_1 in the following.

It is called *deflation* to remove the trivial solutions in the SVD of M_1 in (8) (equivalently the eigenproblem of $M_1^T M_1 \in \mathbf{R}^{(n+1)\times(n+1)}$). First of all, $M_1^T M_1$ has eigenvalue zero, that is removed as follows: By using QR decomposition $S_2 = Q_2 R_2$ with orthogonal matrix $Q_2 \in \mathbf{R}^{k\times k}$ and upper-triangular matrix $R_2 = (R_2'^T, \mathbf{0}_{k-1})^T \in \mathbf{R}^{k\times(k-1)}$, we have

$$M_1(I_{n_1} \oplus Q_2) = \begin{pmatrix} D' & O & \mathbf{0}_{n_1} \\ S_1^T & R_2'^T & \mathbf{0}_{k-1} \end{pmatrix} \equiv (M_2, \mathbf{0}_n).$$

Hence the (right) SVC corresponding to zero SVL of M_1 is $(\mathbf{0}_{n_1}^T, \boldsymbol{q}_{2;k}^T)^T$ with the last column $\boldsymbol{q}_{2;k}$ of Q_2, and we are left with M_2. Deflation for M_2 occurs when there exists l ($1 \leq l \leq n_1$) such that $s_{l,1}^{(1)} = \cdots = s_{l,k-1}^{(1)} = 0$. In this case, M_2 has a SVL d_l' with the corresponding left and right SVCs $\boldsymbol{e}_l^{(n)}$ for both. If this condition is satisfied $n-r$ times, then by using suitable permutation matrices $Q_u, Q_v \in \mathbf{R}^{n\times n}$, M_2 is transformed to

$$Q_u^T M_2 Q_v = \begin{pmatrix} D_1' & O \\ S_1'^T & R_2'^T \end{pmatrix} \oplus D_2' \equiv M_3 \oplus D_2',$$

where $D_1' \in \mathbf{R}^{n_2\times n_2}$, $D_2' \in \mathbf{R}^{(n-r)\times(n-r)}$ are positive definite diagonal matrices with $n_2 = r - k + 1$. Thus, after the deflation, we are left with the $r \times r$ nontrivial part M_3. We call $\delta \equiv 1 - r/n$ the *deflation occurrence rate* (DOR) in the following.

Note that the SVD of M_1 also has a trivial solution in case of $d_j' = d_{j+1}' = \cdots = d_{j+k-1}'$ for some j ($j = 1, \ldots, n_1 - k + 1$). However, since it is rare that this

condition is satisfied, we do not take into account this type of deflation in the present implementation.

3.3 Left SVCs corresponding to $\sigma \simeq 0$

A direct usage of (7) for computing left SVCs for tiny SVLs causes loss of numerical accuracy. In general, the left SVC for a SVL σ of M_3 is equivalent to the eigenvector associated with the eigenvalue σ^2 of

$$M_3 M_3^T = \begin{pmatrix} D_1'^2 & D_1' S_1' \\ S_1'^T D_1' & S_1'^T S_1' + R_2'^T R_2' \end{pmatrix} \in \mathbf{R}^{r \times r}.$$

Hence the left SVC of a tiny SVL σ can be found from Ker $(\sigma^2 I_r - M_3 M_3^T)$, that can be computed by a method analogous to the method in Subsec. 2.2 for computing the eigenvectors of a real diagonal matrix plus low-rank perturbation.

3.4 SVCs corresponding to multiple SVLs

If l neighboring SVLs $\sigma_j, \sigma_{j+1}, \ldots, \sigma_{j+l-1}$ for some j ($l = 2$ in most cases) are close to each other, the computation of kernels of $F(\sigma_j^2), F(\sigma_{j+1}^2), \ldots, F(\sigma_{j+l-1}^2)$ causes a serious loss of numerical accuracy. As a result, the corresponding SVCs become nearly parallel to each other. In this case, together with the average σ_j' of the neighboring SVLs, these kernels are numerically approximated by computing the eigenvectors corresponding to the l smallest eigenvalues of $F(\sigma_j'^2)$, that are obtained by applying the simultaneous inverse iteration with orthogonalization to $F(\sigma_j'^2)$.

3.5 Reorthogonalization

In the final stage in the computation of SVCs, we need a reorthogonalization process, if necessary. When we compute the inner products among SVCs, we use the technique in [3], which makes it possible to complete the computation of the inner products for all pairs of SVCs within $O(n^2)$ arithmetic operations. According to the results of the inner products, we divide the SVCs into several groups, following the criteria:

- The number of groups is maximal under the condition that the following two conditions are satisfied.
- Both for left and right, the SVCs in each group are orthogonal to those in other groups.
- For any pair of left and right SVCs in each group, at least one of them is not orthogonal to some SVC in the same group.

After grouping, we find the SVD of M_3 restricted to the subspaces spanned by the SVCs belonging to each group. More precisely, let $\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_l$ and $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_l$ be the left and right SVCs belonging to one of the groups. Together with orthonormal bases $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_l$ of span$(\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_l)$ and $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l$ of span$(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_l)$, define

$$L \equiv (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_l)^T M_3 (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l),$$

and compute the SVD

$$L = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_l) \Sigma' (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_l)^T$$

of L. Then

$$\begin{cases} \boldsymbol{u}_j = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_l) \boldsymbol{p}_j, \\ \boldsymbol{v}_j = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l) \boldsymbol{q}_j \end{cases} \quad (j = 1, \ldots, l)$$

give the reorthogonalized SVCs with high numerical precision.

4. DCK-SVD algorithm

From Secs. 2 and 3, we obtain the DCK-SVD algorithm. In the main algorithm (proc dcksvd) below, proc rksvd is a subroutine to compute the SVD $M = \hat{U} \Sigma \hat{V}^T$ of M.

proc dcksvd(k, B, Σ, U, V)
Remove SVLs $\sigma \simeq 0$ from B by using the cut-off parameter c_1. (Subsec. 3.1)
Divide B as in (1).
for $i = 1$ **to** k
  call dc2svd(B_i, Σ_i, U_i, V_i), where dc2svd is the `DBDSDC` routine in LAPACK to compute the SVD $B_i = U_i \Sigma_i V_i^T$.
end for
call rksvd(k, M, Σ, $\hat{U}$, $\hat{V}$).
Define U' and V' by (3).
Compute $U \equiv U' \hat{U}$ and $V \equiv V' \hat{V}$.
return Σ, U, V

proc rksvd(k, M, Σ, $\hat{U}$, $\hat{V}$)
Perform deflation. (Subsec. 3.2)
Compute σ_j ($j = 1, \ldots, n$). (Subsec. 2.1)
Compute $\hat{\boldsymbol{u}}_j$, $\hat{\boldsymbol{v}}_j$ ($j = 1, \ldots, n$). (Subsec. 2.2)
(If $\sigma_j / \sigma_n < c_2$, then compute $\hat{\boldsymbol{u}}_j$ following Subsec. 3.3.)
for $j = 1$ **to** $n - 1$
  $l = 1$
  while $j + l \leq n$ **and** $|(\hat{\boldsymbol{u}}_j, \hat{\boldsymbol{u}}_{j+l})| > c_3$ **do**
    $l = l + 1$
  end while
  if $l \geq 2$ **then**
    Compute the left SVCs corresponding to $\sigma_j, \ldots, \sigma_{j+l-1}$. (Subsec. 3.4)
  end if
end for
for $j = 1$ **to** $n - 1$
  $l = 1$
  while $j + l \leq n$ **and** $|(\hat{\boldsymbol{v}}_j, \hat{\boldsymbol{v}}_{j+l})| > c_3$ **do**
    $l = l + 1$
  end while
  if $l \geq 2$ **then**
    Compute the right SVCs corresponding to $\sigma_j, \ldots, \sigma_{j+l-1}$. (Subsec. 3.4)
  end if
end for
Perform reorthogonalization, if necessary. (Subsec. 3.5)
return Σ, $\hat{U}$, $\hat{V}$

5. Numerical experiment

Numerical environment is as follows: CPU: Intel Core i7-960 3.2GHz, memory: 12GB, OS: Ubuntu Linux 9.10 (kernel 2.6.31), LAPACK: version 3.2.1, BLAS: ATLAS version 3.6.0-22 Ubuntu2, and I-SVD: version 0.4.5. We set the cut-off parameters to $c_1 = 8 \times 10^{-16}$, $c_2 = 10^{-6}$ and $c_3 = 0.9$ in DCK-SVD. We use `DBDSDC` routine in LAPACK for DC2-SVD. For I-SVD, we use GotoBLAS2, and `DBDSLV` routine is performed on a single thread.

Test matrices are the following three types:

- **L-DOR**: Upper-bidiagonal matrix $B = (\mathbf{0}_n, B_L) \in \mathbf{R}^{n\times(n+1)}$ with $B_L \in \mathbf{R}^{n\times n}$ such that $T - \lambda_{\min}(T)\, I_n = B_L B_L^T$, where T is the real symmetric tridiagonal matrix obtained by tridiagonalizing a real symmetric dense matrix with normal random numbers of mean 0 and variance 1 in elements, and $\lambda_{\min}(T)$ is the minimum eigenvalue of T. L-DOR is based on a physical model which shows so-called quantum chaos [7].
- **H-DOR1**: a_j $(j = 1, \ldots, n)$ are uniform random numbers in the interval $(-2, 2]$, while b_j $(j = 1, \ldots, n)$ are uniform random numbers in the interval $(-1, 1]$.
- **H-DOR2**: a_j and b_j $(j = 1, \ldots, n)$ are uniform random numbers in the interval $(-1, 1]$.

Matrix size is $n = 5000$. On the execution time as well as numerical errors, we take an average of 16 examples for each type. The DOR is low ($\delta = 0$) for L-DOR, while the other two have high DORs ($\delta = 0.92$ for H-DOR1 and $\delta = 0.95$ for H-DOR2, on average).

Figs. 1 and 2 show the dependence of execution time by DCK-SVD on the division number k for L-DOR and H-DOR1, respectively. For comparison, the execution times by DC2-SVD and I-SVD are also shown. For L-DOR, the DOR is low and as a result, a large division number is preferable in DCK-SVD. With the optimal division number $k = 32$, DCK-SVD (11.4 sec) is faster than DC2-SVD (32.5 sec) and I-SVD (15.8 sec). For H-DOR1, the DOR is high and as a result, the small division number is preferable in DCK-SVD. The optimal division number is $k = 8$, with which the execution time (2.35 sec) by DCK-SVD is comparable to that (2.74 sec) by DC2-SVD. The execution time (13.8 sec) by I-SVD does not depend on the DOR largely. H-DOR2 has also a high DOR, and the dependence of execution time by DCK-SVD on the division number k is the same as in Fig. 2. The execution time by DCK-SVD is 4.30 sec with the optimal division number $k = 4$, while the execution times by DC2-SVD and I-SVD are 2.86 sec and 14.6 sec, respectively. Table 1 shows orthogonal errors

$$\epsilon_{OU} = \max_{1\le i\le n} \|U^T \boldsymbol{u}_i - \boldsymbol{e}_i^{(n)}\|_2,$$

$$\epsilon_{OV} = \max_{1\le i\le n+1} \|V^T \boldsymbol{v}_i - \boldsymbol{e}_i^{(n+1)}\|_2$$

for left and right SVCs as well as the residual

$$\epsilon_R = \max_{1\le i\le n} \frac{\|B\boldsymbol{u}_i - \sigma_i \boldsymbol{v}_i\|_2}{\|B\|_2},$$

when L-DOR, H-DOR1 and H-DOR2 are solved by DCK-SVD, DC2-SVD and I-SVD, respectively. For DCK-SVD, we use the optimal division number for each type. Table 1 shows that DCK-SVD has almost the same numerical accuracy as I-SVD.

In case of H-DOR2 of matrix size $n = 3000, 4000$, we observed several cases that tiny SVLs induce the orthogonal errors beyond $O(10^{-11})$. In all cases, however, they are suppressed below $O(10^{-11})$ by adjusting the value of c_1 in the range $c_1 \in [8\times10^{-16}, 5\times10^{-11}]$, according to the input matrices.

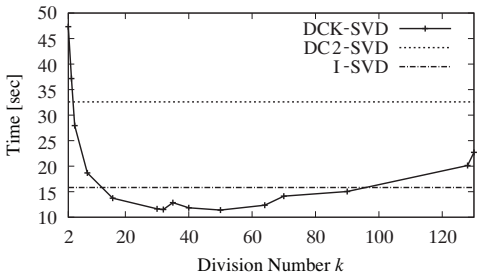

Fig. 1. Dependence of execution time for L-DOR on the division number k in DCK-SVD.

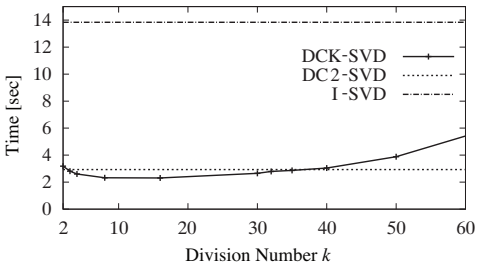

Fig. 2. Dependence of execution time for H-DOR1 on the division number k in DCK-SVD.

Table 1. Numerical errors.

L-DOR	ϵ_{OU}	ϵ_{OV}	ϵ_R
DCK-SVD	1.17E−12	1.17E−12	1.27E−13
DC2-SVD	1.13E−14	1.55E−14	5.92E−15
I-SVD	2.54E−12	2.54E−12	1.26E−14
H-DOR1	ϵ_{OU}	ϵ_{OV}	ϵ_R
DCK-SVD	8.60E−13	4.17E−13	1.03E−13
DC2-SVD	4.55E−15	4.00E−15	9.14E−15
I-SVD	8.25E−13	8.25E−13	8.47E−15
H-DOR2	ϵ_{OU}	ϵ_{OV}	ϵ_R
DCK-SVD	2.74E−13	2.23E−13	3.10E−14
DC2-SVD	4.07E−15	4.14E−15	9.00E−15
I-SVD	1.68E−12	1.60E−12	9.71E−13

Acknowledgments

We are grateful to the anonymous referee for helpful comments, which served to improve the quality of this paper. This work was partially supported by Grant-in-Aid for Scientific Research (C) No.19560058.

References

[1] M. Gu, J. Demmel and I. Dhillon, Efficient computation of the singular value decomposition with applications to least squares problems, Tech. Rep., UT-CS-94-257, Univ. of Tennessee, 1994.
[2] Y. Kuwajima and T. Shigehara, An extension of divide-and-conquer for real symmetric tridiagonal eigenproblem (in Japanese), Trans. JSIAM, **15** (2005) 89–115.
[3] Y. Kuwajima and T. Shigehara, An improvement of multiple division divide-and-conquer for real symmetric tridiagonal eigenproblem (in Japanese), Trans. JSIAM, **16** (2006) 453–480.
[4] E. Anderson et al., LAPACK Users' Guide, Third Edition, SIAM, Philadelphia, 1999.
[5] H. Toyokawa, K. Kimura, M. Takata and Y. Nakamura, On parallelism of the I-SVD algorithm with a multi-core processor, JSIAM Letters, **1** (2009) 48–51.
[6] M. Gu and S. C. Eisenstat: A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, SIAM J. Matrix Anal. Appl., **16** (1995) 172–191.
[7] M. L. Mehta, Random Matrices, Third Edition, Pure and Applied Mathematics, 142, Elsevier/Academic Press, 2004.