Fast Computation Method of Column Space by using the DQDS Method and the OQDS Method

Sho Araki[1], **Hiroki Tanaka**[2], **Masami Takata**[3], **Kinji Kimura**[4], **and Yoshimasa Nakamura**[1]
[1]Graduate School of Informatics, Kyoto University, Kyoto, Kyoto, JAPAN
[2]Research Organization of Science and Technology, Ritsumeikan University, Kusatsu, Shiga, JAPAN
[3]Research Group of Information and Communication Technology for Life,
Nara Women's University, Nara, Nara, JAPAN
[4]Department of Computer Science and Technology, Salesian Polytechnic, Machida, Tokyo, JAPAN

Abstract— *In this paper, we propose a computational method that combines the DQDS (differential qd with shift) method and the OQDS (orthogonal qd with shift) method to compute the column space of a rectangular matrix. Once a rectangular matrix is decomposed into the product of two orthogonal matrices and a bidiagonal matrix using the Householder transformation, the methods for computing the column space can be applied to the bidiagonal matrix. In the proposed method, the DQDS method is adopted to investigate the distribution of all singular values, which is then used to determine the numerical rank. Using the OQDS method, we can obtain the row space in the lower bidiagonal matrix, which is equal to the column space in the upper bidiagonal matrix. In a numerical experiment, we compare the orthogonality of the computed row space and the computation time in the conventional method, which computes all right singular vectors, and those in the proposed method.*

Keywords: the column space, the DQDS method, the OQDS method, Singular value decomposition

1. Introduction

Generalized eigenvalue problems composed of symmetric matrices and positive definite symmetric matrices arise in applications such as molecular orbital methods of quantum chemistry and data clustering. In these applications, only some eigenvalues, and the eigenvectors corresponding to them, are required. The Sakurai-Sugiura method [12] is known for computing truncated eigenvalue decomposition. By using the Sakurai-Sugiura method, eigenvalues and eigenvectors in a specified circular disk can be obtained. When the upper bound of the number of eigenvalues contained within the circular disk is given, a rectangular matrix for computing the column space, which is needed in the Sakurai-Sugiura method, can be obtained. The column space is spanned by the left singular vectors, which can be obtained by singular value decomposition of the rectangular matrix, except for the null space. Usually, singular value decomposition methods have a long computation time and a large accumulation of rounding error. The column space, which is needed in the Sakurai-Sugiura method, is not the left singular vectors, but the column space expressed as a linear combination of the left singular vectors. Consequently, for computing only the column space, a method without singular value decomposition should be developed. Once the rectangular matrix is decomposed into a product of two orthogonal matrices and a bidiagonal matrix using the Householder transformation [4], the method for computing the column space can be applied to the bidiagonal matrix.

In this paper, we propose a computation method, which combines the DQDS (differential qd with shift) method [5], [11] and the OQDS (orthogonal qd with shift) method [10], for computing the column space of an upper bidiagonal matrix. In the proposed method, the DQDS method is adopted to investigate the distribution of all singular values. From the distribution of singular values, the numerical rank is determined. Using the OQDS method, we can obtain the row space in the lower bidiagonal matrix, which is equal to the column space in the upper bidiagonal matrix.

In a numerical experiment, we compare the orthogonality of the computed row space and the computation time in the conventional method, which computes all right singular vectors, and those in the proposed method.

In sec.2, we introduce the DQDS method. In sec.3, we explain the OQDS method. In sec.4, we propose a method for the computation of column space. In sec.5, we evaluate the proposed method with respect to computation time and accuracy.

2. DQDS Method for Computing Singular Values

2.1 Outline

Let B be an $n \times n$ bidiagonal matrix:

$$B = \begin{pmatrix} \sqrt{q_1} & \sqrt{e_1} & & \\ & \sqrt{q_2} & \ddots & \\ & & \ddots & \sqrt{e_{n-1}} \\ & & & \sqrt{q_n} \end{pmatrix}. \quad (1)$$

Algorithm 1 DQDS method

Set $d_1 := q_1 - s$;
for $k := 1, 2, \cdots, n-1$ **do**
  Set $\hat{q}_k := d_k + e_k$;
  if $SM \times \hat{q}_k < q_{k+1}$ and $SM \times q_{k+1} < \hat{q}_k$ **then**
    Set $\hat{e}_k := (q_{k+1}/\hat{q}_k)\, e_k$;
    Set $d_{k+1} := (q_{k+1}/\hat{q}_k)\, d_k - s$;
  else
    Set $\hat{e}_k := (e_k/\hat{q}_k)\, q_{k+1}$;
    Set $d_{k+1} := (d_k/\hat{q}_k)\, q_{k+1} - s$;
  end if
end for
Set $\hat{q}_n := d_n$;

Using the DQDS method, the bidiagonal matrix B is transformed into the $n \times n$ upper bidiagonal matrix,

$$\hat{B} = \begin{pmatrix} \sqrt{\hat{q}_1} & \sqrt{\hat{e}_1} & & \\ & \sqrt{\hat{q}_2} & \ddots & \\ & & \ddots & \sqrt{\hat{e}_{n-1}} \\ & & & \sqrt{\hat{q}_n} \end{pmatrix}. \tag{2}$$

Algorithm 1 shows the DQDS method. Here, a variable s means a value of shift and SM is the smallest positive number. $1/SM$ does not overflow. Let Σ be a diagonal matrix arranged in descending order of singular values. After the repetition of the above operation in the DQDS method, $\hat{B}$ converges to a diagonal matrix D. Then $\Sigma_{kk} = \sqrt{D_{kk} + S}$ $(k = 1, \cdots, n)$, where S is the sum of the value of the shift s, is satisfied. In the DQDS method, singular values can be obtained at high speed by introducing shift.

In the DQDS method, all variables must be non-negative numbers and d_k satisfies $d_k (k = 1, \cdots, n-1) > 0$. If $d_k = 0$, then $d_{k+1} < 0$ and the DQDS method fails because $s > 0$. However, in the case where $\hat{q}_n = d_n = 0$, one iteration in the DQDS method can be terminated correctly. Consequently, the value of the shift s can be set to not only less than the minimum eigenvalue $\lambda_{\min}\left(B^\top B\right)$ in $B^\top B$, but also equal to the minimum eigenvalue.

The variable d_k in the DQDS method is computed using a special format $f \times g + h$. Almost all computers have the fused multiply-add operation, which enables high precision computation. The special format utilizes the fused multiply-add operation. Thus, the DQDS method achieves high accuracy.

When the DQDS method is terminated, the diagonal elements are arranged in descending order of their singular values. Considering the method's implementation, in the case that $q_1 < q_n$, the elements in B are replaced as follows:

$$B_r = \begin{pmatrix} \sqrt{q_n} & \sqrt{e_{n-1}} & & \\ & \sqrt{q_{n-1}} & \ddots & \\ & & \ddots & \sqrt{e_1} \\ & & & \sqrt{q_1} \end{pmatrix}. \tag{3}$$

Even with this replacement, singular values of B_r are equal to that of B. Thus, after the replacement where $B \leftarrow B_r$, the DQDS method is adopted to B_r.

In a singular value computation with shift, the sum of the value of shift is added to the obtained diagonal elements. The strategy to give the value of shift is expressed in sec.2.2. Let $s^{(i)}$ set the value of shift s in the i-th iteration. The sum of the value of shift S is computed as follows:

$$S = \sum_{i=1}^{i_0} s^{(i)}. \tag{4}$$

In eq. (4), a small value is added to a large value. In this case, loss of trailing digits occurs. To avoid this problem, the double-double arithmetic[7], which expresses one number using two double precision floating point numbers, should be adopted.

2.2 Strategy to give the value of shift

2.2.1 Appropriate update procedure of shift

s is set to an arbitrary positive number. The DQDS method can be transformed into eq. (5), in which $\hat{q}_k$ and $\hat{e}_k$ are not used.

$$\begin{aligned} d_1 &= q_1 - s, \\ d_k &= q_k \frac{d_{k-1}}{(d_{k-1} + e_{k-1})} - s, \quad (k = 2, \cdots, n). \end{aligned} \tag{5}$$

The following operation is repeated for s.

1) If $d_k \leq 0$ for some $1 \leq k \leq n$, $s' \leftarrow \max(d_k + s, 0)$. If s' is equal to s in floating-point computation, $s' \leftarrow 0, d_k \leftarrow 0$.
2) After the above operation, if $d_n \geq 0$, then we end the repetition.
3) Moreover, in order to prevent s from becoming too small, $s \leftarrow \max(s', 0.75s)$
4) Recompute eq. (5) using the above s.

Even if s is not a lower bound of $\lambda_{\min}\left(B^\top B\right)$, by repeating the above operation multiple times, s is often transformed into a positive number, which is a lower bound of $\lambda_{\min}\left(B^\top B\right)$. The above appropriate update procedure of shift can be utilized not only to obtain the generalized Rutishauser bound [1] described in sec.2.2.2 with eq. (5), but also to deal with the case that $d_k \leq 0$ for some $1 \leq k \leq n$ in the DQDS method.

2.2.2 Shift strategy

The design of the shift strategy, which gives the value of the shift s in the DQDS method, is extremely difficult, and it is still in the process of development. In this section, we explain not the aggressive shift [11], which is implemented in the routine `DLASQ` of LAPACK [9], but our shift strategy proposed in this paper.

In our proposed shift strategy, the following lower bounds are combined.

- generalized Rutishauser bound [1]
- Maximum value of Laguerre bound [10], Newton bound, generalized Newton bound [13]
- bounds based on the Collatz inequality [3]
- Johnson bound [8]

We introduce mathematical definitions and the combination.

The generalized Rutishauser bound is expressed as follows. Let s be set $\lambda_{\min}\left(F^\top F\right)$, where

$$F = \begin{pmatrix} \sqrt{q_{n-1}} & \sqrt{e_{n-1}} \\ 0 & \sqrt{q_n} \end{pmatrix}. \tag{6}$$

$\lambda_{\min}\left(F^\top F\right)$ is an upper bound of $\lambda_{\min}\left(B^\top B\right)$. Through appropriate update procedure in sec.2.2.1, s becomes a positive number and a tight lower bound of $\lambda_{\min}\left(B^\top B\right)$. In the generalized Rutishauser bound, only in the case where $d_k > 0\,(k = 1, \ldots, n-1)$ and $d_n < 0$, s is adopted as the lower bound of $\lambda_{\min}\left(B^\top B\right)$. Of course, if $d_k > 0\,(k = 1, \ldots, n-1)$ and $d_n \geq 0$, $\lambda_{\min}\left(F^\top F\right)$ can be regarded as the suitable amount of shift. When $d_k \leq 0$ for some $1 \leq k \leq n-1$, the other bounds are used as the value of shift.

Using

$$a = \mathrm{Tr}\left(\left(BB^\top\right)^{-1}\right), \tag{7}$$

and

$$b = \mathrm{Tr}\left(\left(BB^\top\right)^{-2}\right), \tag{8}$$

the Laguerre bound, the Newton bound, and the generalized Newton bound are defined as follows:

- Laguerre bound

$$L = \frac{n}{a + \sqrt{(n-1)\,(nb - a^2)}} \tag{9}$$

- Newton bound

$$N = a^{-1} \tag{10}$$

- Generalized Newton bound

$$GN = b^{-\frac{1}{2}} \tag{11}$$

Here, a and b are computed by the recurrence formula(12) and (13) [13].

$$a = \sum_{k=1}^{n} f_k, \tag{12}$$
$$\begin{cases} f_1 = \dfrac{1}{q_1}, \\ f_k = \dfrac{1}{q_k} + \dfrac{e_{k-1}}{q_k} f_{k-1}, \quad k = 2, \cdots, n, \end{cases}$$

$$b = \sum_{k=1}^{n} g_k, \tag{13}$$
$$\begin{cases} g_1 = f_1^2, \\ g_k = f_k^2 + \dfrac{e_{k-1}}{q_k}\left(g_{k-1} + f_{k-1}^2\right), \quad k = 2, \cdots, n, \end{cases}$$

In theory,

$$N \leq GN \leq L, \tag{14}$$

is always satisfied. In a numerical computation, eq. (14) may not be satisfied because of rounding error. Therefore, $X = \max(N, GN, L)$ is adopted.

In the Collatz inequality, if all elements in an $n \times n$ matrix A are positive numbers and $\boldsymbol{v}$ is a vector, which consists of n positive elements, then,

$$\lambda_{\max}(A) \leq \max_k \frac{(A\boldsymbol{v})_k}{v_k}. \tag{15}$$

If all elements in B are positive numbers and K is defined as follows,

$$K = \begin{pmatrix} \sqrt{q_1} & & & \\ -\sqrt{e_1} & \sqrt{q_2} & & \\ & \ddots & \ddots & \\ & & -\sqrt{e_{n-1}} & \sqrt{q_n} \end{pmatrix}, \tag{16}$$

then the lower bound of $\lambda_{\min}\left(B^\top B\right)$ is obtained using the following inequality:

$$\begin{aligned} \min_k \frac{\boldsymbol{v}_k}{\left(\left(K^\top K\right)^{-1}\boldsymbol{v}\right)_k} &\leq \lambda_{\min}\left(K^\top K\right) \\ &= \lambda_{\min}\left(B^\top B\right), \end{aligned} \tag{17}$$

since all elements of $\left(K^\top K\right)^{-1}$ are positive numbers. $\boldsymbol{v}$ is generated by using the inverse iteration method [4]:

$$\boldsymbol{x} = \left(K^\top K\right)^{-1}(1, 1, \cdots, 1)^\top, \tag{18}$$
$$\boldsymbol{v} = \frac{\boldsymbol{x}}{\max_k \boldsymbol{x}_k}. \tag{19}$$

Moreover,

$$\frac{1}{\max_k \boldsymbol{x}_k} \leq \lambda_{\min}\left(K^\top K\right) = \lambda_{\min}\left(B^\top B\right), \tag{20}$$

can be adopted as the further lower bound.

Johnson bound is the lower bound based on the following theorem. Let us set $C = \left(B^\top + B\right)/2$.

$$\lambda_{\min}(C) \leq \sqrt{\lambda_{\min}(B^\top B)}, \tag{21}$$

is satisfied. Therefore, by adopting Gerschgorin bound [6] to C, the lower bound $\lambda_{\min}\left(B^\top B\right)$ is obtained.

In our proposed shift strategy, at fast, the generalized Rutishauser bound is computed. When the positive lower bound of $\lambda_{\min}\left(B^\top B\right)$ can be obtained, the proposed shift strategy is terminated and the value of the shift s sets to the lower bound. Otherwise, the Laguerre bound, the Newton bound, and the generalized Newton bound are computed. When

$$Z < 2 \times X, \tag{22}$$

is satisfied, X is adopted as the value of shift s. Here, Z means the upper bound of $\lambda_{\min}\left(B^\top B\right)$. Z is defined as $Z = \min(z_1, z_2, z_3)$:

- By using the small matrix F,

$$z_1 = \lambda_{\min}\left(F^\top F\right). \tag{23}$$

- By using the approximate amount g_k for Z,

$$z_2 = \frac{1}{\sqrt{\max_{k=1}^{n} g_k}}. \tag{24}$$

- The upper bound is obtained by using an optimization problem [14]. If, using a and b, an integer number m is determined within $m - 1 < a^2/b \leq m$, then,

$$z_3 = \frac{m}{a + \sqrt{(mb - a^2)/(m-1)}}, \tag{25}$$

is the value for Z.

When eq. (22) is not satisfied, s is computed by using the bounds based on the Collatz inequality. However, in the case where the value of the shift s is not a positive number, the value of the shift s is reset by using the Johnson bound.

2.3 Convergence criteria of the DQDS method

There are three methods that serve as convergence criteria.

In the first method, when the element $e_k (k = 1, \ldots, n-2)$ is extremely smaller than the sum of the value of shift S in eq. (4), the e_k can be regarded as 0. In the DQDS method, if e_k is equal to 0, the split operation [11] occurs and the given matrix is divided into two matrices. Moreover, when the element e_{n-1} is sufficiently smaller than the element $S + q_n$, the dimension size decreases by one. Thus, by checking e_k $(k = 1, \ldots, n-2)$ and e_{n-1}, the DQDS method is terminated. However, only by using the first method, the number of iterations increases and rounding error accumulates.

Algorithm 2 shows the DQD method [5], [11], which is based on the DQDS method without a shift, with the second convergence method. In the second method, when the element d_k is extremely smaller than the sum of the value of the shift S, the d_k can be regarded as 0. By using the algorithm 2 and $d_k = 0$, $\hat{q}_n$ becomes 0. Thereafter, if $\hat{q}_n = 0$, then $\hat{e}_{n-1} = 0$ in the next iteration. Thus, the confirmation that d_k is extremely smaller than the sum of the value of the shift S, leads to convergence.

Algorithm 2 DQD method

```
1: Set d_1 := q_1;
2: if d_1 + S = S then
3:     Set d_1 := 0;
4: end if
5: for k := 1, 2, ..., n − 1 do
6:     Set q̂_k := d_k + e_k;
7:     if q̂_k = 0 then
8:         Set ê_k := 0, d_{k+1} := q_{k+1};
9:     else if SM × q̂_k < q_{k+1} and SM × q_{k+1} < q̂_k then
10:        Set ê_k := (q_{k+1}/q̂_k) e_k;
11:        Set d_{k+1} := (q_{k+1}/q̂_k) d_k;
12:    else
13:        Set ê_k := (e_k/q̂_k) q_{k+1};
14:        Set d_{k+1} := (d_k/q̂_k) q_{k+1};
15:    end if
16:    if d_{k+1} + S = S then
17:        Set d_{k+1} := 0;
18:    end if
19: end for
20: Set q̂_n := d_n;
```

In the third method, we use the following recurrence relation,

$$\begin{aligned} &d_1 = q_1, \\ &d_k = q_k \left(d_{k-1}/\left(d_{k-1} + e_{k-1}\right)\right), \quad k = 2, \cdots, n, \end{aligned} \tag{26}$$

where the d_k in eq. (26) completely matches the d_k in the DQD method. When the element e_{k-1} is extremely smaller than the element d_{k-1}, e_{k-1} can be regarded as 0. However, the implementation of the iteration in the DQD method is not efficient in terms of computation time. We notice $\hat{q}_k = d_k + e_k$. If $e_k \leq \varepsilon d_k = \varepsilon\left(\hat{q}_k - e_k\right)$, then e_{k-1} can be regarded as 0. Thus, we check,

$$e_k \leq \frac{\varepsilon}{1+\varepsilon}\hat{q}_k, \tag{27}$$

where ε is extremely smaller than 1. Instead of eq. (27), for checking whether e_k is sufficiently small or not, we should use $e_k \leq \varepsilon \hat{q}_k$. When e_k is extremely small, $\hat{e}_k$ is sufficiently small. Therefore, the given matrix can be divided.

3. OQDS Method for Computing Singular Vectors

3.1 Outline

In singular value computation, the DQDS methods and the OQDS methods are mathematically equivalent. The OQDS

method is theoretically suitable for computation of singular values, with high accuracy in terms of relative errors. However, the OQDS method needs square root computation. On the other hand, in the DQDS method, square root computation is deleted by variable transformation. Thus, the DQDS method is more advantageous than the OQDS method, in terms of high speed. However, the DQDS method cannot compute singular vectors. In contrast, the OQDS method can compute singular vectors. Thus, in this paper, we use the OQDS method to compute singular vectors.

Let $L^{(i)}$ be an $n \times n$ lower bidiagonal matrix,

$$L^{(i)} = \begin{pmatrix} \alpha_1^{(i)} & & & \\ \beta_1^{(i)} & \alpha_2^{(i)} & & \\ & \ddots & \ddots & \\ & & \beta_{n-1}^{(i)} & \alpha_n^{(i)} \end{pmatrix}, \tag{28}$$

and $U^{(i)}$ be an $n \times n$ upper bidiagonal matrix,

$$U^{(i)} = \begin{pmatrix} \gamma_1^{(i)} & \zeta_1^{(i)} & & \\ & \gamma_2^{(i)} & \ddots & \\ & & \ddots & \zeta_{n-1}^{(i)} \\ & & & \gamma_n^{(i)} \end{pmatrix}, \tag{29}$$

respectively.

In the OQDS method, the following three operations are repeated for $L^{(i)}$ and $U^{(i)}$.

1) Compute a shift $u^{(i)}$ satisfying:

$$0 \leq u^{(i)} \leq \sigma_{\min}\left(L^{(i)}\right). \tag{30}$$

2) LU step

$$P^{(i)} \begin{pmatrix} L^{(i)} \\ t^{(i)} I_n \end{pmatrix} = \begin{pmatrix} U^{(i)} \\ t^{(i+1)} I_n \end{pmatrix}, \tag{31}$$

$$t^{(i+1)} = \sqrt{\left(t^{(i)}\right)^2 + \left(u^{(i)}\right)^2} \tag{32}$$

3) UL step

$$\begin{pmatrix} I_n & O \\ O & \left(Q^{(i)}\right)^\top \end{pmatrix} \begin{pmatrix} U^{(i)} \\ t^{(i+1)} I_n \end{pmatrix} Q^{(i)} = \begin{pmatrix} L^{(i+1)} \\ t^{(i+1)} I_n \end{pmatrix}. \tag{33}$$

The lower half of the matrix is trivial. Thus, only $U^{(i)}Q^{(i)} = L^{(i+1)}$ should be considered.

$P^{(i)}$ and $Q^{(i)}$ are a $2n \times 2n$ orthogonal matrix and an $n \times n$ orthogonal matrix, respectively. $P^{(i)}$ is computed by using the Givens rotation and the generalized Givens rotation [10]. $Q^{(i)}$ consists of Givens rotation. $t^{(i_0)}I_n$ expresses a diagonal matrix having the same value. Let Σ be a diagonal matrix arranged in descending order of singular values. When $L^{(i_0)}$ and $t^{(i_0)}I_n$ converge to a diagonal matrix E and tI_n, if the split operation does not occur, $\Sigma_{kk} = \sqrt{E_{kk}^2 + t^2}$ $(k = 1, \cdots, n)$. To obtain right singular vectors, an orthogonal matrix V is computed with $V = Q^{(0)} \cdots Q^{(i_0-1)}$. Using Givens rotation, we add tI_n to $E = L^{(i_0)}$. Then, $L^{(i_0)}$ and tI_n become Σ and the zero matrix, respectively. The orthogonal matrix U, which consists of left singular vectors, are not required in the Sakurai-Sugiura method. Thus, in this paper, we do not introduce $P^{(i)}$, which is adopted to compute U.

The sum of the value of shift is computed as,

$$t^{(i+1)} = \sqrt{\left(t^{(i)}\right)^2 + \left(u^{(i)}\right)^2}. \tag{34}$$

To avoid loss of trailing digits, the double-double arithmetic should be adopted into the OQDS method.

When the OQDS method is terminated, if the split operation does not occur, the diagonal elements of E are arranged in descending order. Thus, in the case that $\alpha_1^{(i)} < \alpha_n^{(i)}$, by using the matrix Y,

$$Y = \begin{pmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{pmatrix}, \tag{35}$$

and

$$\begin{pmatrix} U^{(i)} \\ t^{(i)} I_n \end{pmatrix} \leftarrow \begin{pmatrix} Y & O \\ O & Y \end{pmatrix} \begin{pmatrix} L^{(i)} \\ t^{(i)} I_n \end{pmatrix} Y, \tag{36}$$

all elements are rearranged in reverse order. $L^{(i)}$ and $U^{(i)}$ are an $n \times n$ lower bidiagonal matrix and an $n \times n$ upper bidiagonal matrix, respectively.

3.2 LU and UL steps

The LU step is an operation that transforms $L^{(i)}$ to $U^{(i)}$.

$$\begin{pmatrix} \rho_1^{(i)} & & \\ \beta_1^{(i)} & \alpha_2^{(i)} & \\ & \beta_2^{(i)} & \ddots \\ & & \ddots \\ t^{(i)} & t^{(i)} & \\ & & \ddots \end{pmatrix} \xrightarrow{\text{O1}} \begin{pmatrix} \rho_1^{(i)} & & & \\ \beta_1^{(i)} & \alpha_2^{(i)} & & \\ & \beta_2^{(i)} & \ddots & \\ & & \ddots & \\ \sqrt{\left(t^{(i)}\right)^2 + \left(u^{(i)}\right)^2} & & & \\ & t^{(i)} & & \\ & & \ddots & \end{pmatrix}$$

$$\xrightarrow{\text{O2}} \begin{pmatrix} & \gamma_1^{(i)} & \zeta_1^{(i)} & \\ & & \rho_2^{(i)} & \\ & & \beta_2^{(i)} & \ddots \\ & & & \ddots \\ \sqrt{\left(t^{(i)}\right)^2 + \left(u^{(i)}\right)^2} & & t^{(i)} & \\ & & & \ddots \end{pmatrix} \xrightarrow{\text{O3}} \tag{37}$$

Algorithm 3 shows the LU step with the value of the shift $u^{(i)}$ and elements of $L^{(i)}$ and $U^{(i)}$. In the computation of $\rho_{k+1}^{(i)}$, the fused multiply-add operation can be adopted effectively. Algorithm 4 shows the LU step in the case where $u^{(i)} = 0$.

Algorithm 3 The LU step in the case of the value of shift $u^{(i)} > 0$

Set $\rho_1^{(i)} := \sqrt{\alpha_1^{(i)} - u^{(i)}}\sqrt{\alpha_1^{(i)} + u^{(i)}}$;
for $k := 1, 2, \cdots, n-1$ **do**
  Set $\gamma_k^{(i)} := \sqrt{\left(\rho_k^{(i)}\right)^2 + \left(\beta_k^{(i)}\right)^2}$, $\zeta_k^{(i)} := \frac{\beta_k^{(i)}}{\gamma_k^{(i)}}\alpha_{k+1}^{(i)}$;
  Set $\rho_{k+1}^{(i)} := \sqrt{\frac{\rho_k^{(i)}}{\gamma_k^{(i)}}\alpha_{k+1}^{(i)} - u^{(i)}}\sqrt{\frac{\rho_k^{(i)}}{\gamma_k^{(i)}}\alpha_{k+1}^{(i)} + u^{(i)}}$;
end for
Set $\gamma_n^{(i)} := \rho_n^{(i)}$;

Algorithm 4 The LU step in the case of the value of shift $u^{(i)} = 0$

Set $\rho_1^{(i)} := \alpha_1^{(i)}$;
for $k := 1, 2, \cdots, n-1$ **do**
  Set $\gamma_k^{(i)} := \sqrt{\left(\rho_k^{(i)}\right)^2 + \left(\beta_k^{(i)}\right)^2}$;
  if $\gamma_k^{(i)} = 0$ **then**
    Set $\zeta_k^{(i)} := 0$, $\rho_{k+1}^{(i)} := \alpha_{k+1}^{(i)}$;
  else
    Set $\zeta_k^{(i)} := \left(\beta_k^{(i)}/\gamma_k^{(i)}\right)\alpha_{k+1}^{(i)}$;
    Set $\rho_{k+1}^{(i)} := \left(\rho_k^{(i)}/\gamma_k^{(i)}\right)\alpha_{k+1}^{(i)}$;
  end if
end for
Set $\gamma_n^{(i)} := \rho_n^{(i)}$;

The UL step is an operation that transforms $U^{(i)}$ to $L^{(i+1)}$ as following:

$$\begin{pmatrix} \gamma_1^{(i)} & \zeta_1^{(i)} & & & \\ & \gamma_2^{(i)} & \zeta_2^{(i)} & & \\ & & \ddots & \ddots & \\ & & & \ddots & \\ t^{(i)} & & & & \\ & t^{(i)} & & & \\ & & \ddots & & \\ & & & \ddots & \end{pmatrix} \rightarrow \begin{pmatrix} \alpha_1^{(i+1)} & & & & \\ \beta_1^{(i+1)} & \eta_2^{(i+1)} & \zeta_2^{(i)} & & \\ & & \ddots & \ddots & \\ & & & \ddots & \\ t^{(i)} & & & & \\ & t^{(i)} & & & \\ & & \ddots & & \\ & & & \ddots & \end{pmatrix} \tag{38}$$

The eq. (38) can be carried out using Givens rotation from the right hand side. Algorithm 5 shows the UL step with elements of $U^{(i)}$ and $L^{(i+1)}$.

In the OQDS method, the Givens rotation is adopted to the LU and UL steps. Usually, in implementation, the Givens rotation can be computed by using `DROTG`, which is a level-1 routine in BLAS [2]. However, computation of Givens rotation with high precision, which is required from the OQDS method, should be performed by using `DLARTG` in LAPACK.

3.3 Shift strategy for the OQDS method

In the OQDS method, the value of shift $u^{(i)}$ is computed using the follows:

Algorithm 5 UL step

Set $\eta_1^{(i)} := \gamma_1^{(i)}$;
for $k := 1, 2, \cdots, n-1$ **do**
  Set $\alpha_k^{(i+1)} := \sqrt{\left(\eta_k^{(i)}\right)^2 + \left(\zeta_k^{(i)}\right)^2}$;
  if $\alpha_k^{(i+1)} = 0$ **then**
    Set $\beta_k^{(i+1)} := 0$, $\eta_{k+1}^{(i)} := \gamma_{k+1}^{(i)}$;
  else
    Set $\beta_k^{(i+1)} := \left(\zeta_k^{(i)}/\alpha_k^{(i+1)}\right)\gamma_{k+1}^{(i)}$;
    Set $\eta_{k+1}^{(i)} := \left(\eta_k^{(i)}/\alpha_k^{(i+1)}\right)\gamma_{k+1}^{(i)}$;
  end if
end for
Set $\alpha_n^{(i)} := \eta_n^{(i)}$;

1) generalized Rutishauser bound
2) bounds based on the Collatz inequality
3) Johnson bound

Unlike the shift strategy for the DQDS method in sec.2.2, square root computations are not required in the bounds based on the Collatz inequality. Thus, since the bounds based on the Collatz inequality can be computed at high speed, the Laguerre bound, the Newton bound, and the generalized Newton bound can be excluded. The strategy of the combination of the lower bounds in the OQDS method is the same as that in the DQDS method.

3.4 Convergence criteria of the OQDS method

The first convergence criteria of the DQDS method in sec.2.3 can be used as the convergence criteria of the OQDS method through change of variables.

In terms of the 2 norm, the third convergence criteria of the DQDS method in sec.2.3 is defined. In the OQDS method, in terms of the 1 norm, the convergence criteria are redefined as follows. If, in the following recurrence relation,

$$\begin{aligned} &\mu_1 = \alpha_1^{(i)}, \\ &\mu_k = \alpha_k^{(i)} \frac{\mu_{k-1}}{\left(\mu_{k-1} + \beta_{k-1}^{(i)}\right)}, \quad k = 2, \cdots, n, \end{aligned} \tag{39}$$

$\beta_{k-1}^{(i)}$ is extremely smaller than μ_{k-1}, then $\beta_{k-1}^{(i)}$ can be regarded as 0. The convergence criteria in terms of the 1 norm is more accurate than those in terms of the 2 norm.

4. Proposed Method

In this section, we propose a fast computational method for column space of the upper bidiagonal matrix. The DQDS method is known as a fast computational method only for singular values. Thus, the DQDS method is adopted to investigate the distribution of all singular values. From the distribution of singular values, the numerical rank is determined. After excluding the number K of singular values,

Table 1: Experimental Environment

CPU	Intel(R) Core(TM) CPU i3-7100 @ 3.90GHz
RAM	4 GB
OS	Ubuntu 16.04.3 LTS
Compiler	gcc version 5.4.0, gfortran version 5.4.0
Options	-O3 -mtune=native -march=native -fopenmp
Software	Intel Math Kernel Library 2018

which are very close to 0, from the size n of the matrix, the numerical rank is defined as $n - K$. If the split operation does not occur, the diagonal elements of E computed in the OQDS method are arranged in descending order. Here, if non-diagonal elements do not become completely 0 in iterations, the OQDS method may compute some right singular vectors corresponding to the range from the smallest singular value to K-th singular value. The OQDS method can compute the null space of the lower bidiagonal matrix $L^{(0)}$ as K right singular vectors. In the OQDS method, the computational cost of the null space is cheaper than that of all the right singular vectors. While singular vectors corresponding to the null space are computed, the row space can be obtained as the complementary space of the null space. Here, the row space in the lower bidiagonal matrix $L^{(0)}$ is equal to the column space in the upper bidiagonal matrix $\left(L^{(0)}\right)^\top$. Therefore, the column space is obtained from the complementary space of K vectors corresponding to the range from the smallest singular value to K-th singular value in $V = Q^{(0)} \cdots Q^{(i_0-1)}$ of $L^{(0)}$. In other words, the column space is obtained from the first $n - K$ vectors in V. Finally, singular values computed in the OQDS method should be compared with those computed in the DQDS method.

5. Numerical Experiment

To confirm the effectiveness of the proposed method, we compare orthogonality of the computed row space and the computation time in the conventional method, which computes all right singular vectors with the OQDS method, with that of the proposed method. Table 1 shows the experimental environment. The following bidiagonal matrix is used for comparison:

$$\sigma_i := \varepsilon^{\frac{i-1}{n-1}}, \quad i = 1, \cdots, n - t_0 \tag{40}$$

$$\sigma_i := \varepsilon^{2\frac{i-1}{n-1}}, \quad i = n - t_0 + 1, \cdots, n \tag{41}$$

where σ_i means the value of singular-value. ε is set to the machine epsilon$\left(2.22044604925031 \times 10^{-16}\right)$ and t_0 is set to 20. The dimension n of the matrix is 128. For this matrix, we can obtain the number K of singular values, which are very close to 0, from the result computed with the DQDS method. From the relative gap $\hat{\sigma}_{i+1}/\hat{\sigma}_i$, where $\hat{\sigma}_i$ is computed as singular values with the DQDS method, K is found to be t_0. Table 2 shows orthogonality $||V^\top V - I||_F$ of the computed row space and the computation time.

Table 2: Orthogonality and Computation Time

	Conventional method	Proposed method
Orthogonality	1.23×10^{-14}	4.76×10^{-15}
Computation Time	5.27×10^{-4}	1.03×10^{-4}

6. Conclusions

In this paper, we have proposed a fast computation method, which combines the DQDS methods and the OQDS methods, for computing the column space of the rectangular matrix. In the proposed method, the DQDS method is adopted to investigate the distribution of all singular values. Using the OQDS method, we can obtain the row space in the lower bidiagonal matrix. In the numerical experiment, we have confirmed the effectiveness of the proposed method.

As part of future work, the proposed method is expected to be incorporated into the Sakurai-Sugiura method.

Acknowledgment

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

References

[1] K. Aishima, T. Matsuo, and K. Murota. Rigorous proof of cubic convergence for the dqds algorithm for singular values. *Japan J. Indust. Appl. Math.* , Vol.25, pp.63–81, 2008.

[2] (2018). Basic Linear Algebra Subprograms, Netlib. [online]. http://netlib.org/blas/index.html

[3] L. Collatz. Einschliessungssatz fuer die charakteristischen Zahlen von Matrizen. *Mathemathische Zeitschrift*, Vol.48, pp.221-226, 1942.

[4] J. Demmel. Applied Numerical Linear Algebra. *SIAM*, Philadelphia, 1997.

[5] K. V. Fernando and B. N. Parlett. Accurate singular values and differential qd algorithms. *Numer. Math.*, Vol.67, pp.191–229, 1994.

[6] S. Gerschgorin. Über die Abgrenzung der Eigenwerteeiner Matrix. *Izv. Akad.Nauk. USSR Otd. Fiz.-Mat. Nauk*, Vol. 7, pp. 749–754, 1931.

[7] Y. Hida, X. S. Li, D. H. Bailey. Library for Double-Double and Quad-Double Arithmetic. *Proc. 15th Symposium on Computer Algorithmetic*, pp.155–162, 2007.

[8] C. R. Johnson. A Gersgorin-type lower bound for the smallest singular value. *Lin. Alg. Appl.*, Vol. 112, pp. 1–7, 1989.

[9] (2018) LAPACK-Linear Algebra PACKage [online]. http://www.netlib.org/lapack/

[10] U. von Matt. The orthogonal qd-algorithm. *SIAM J. Sci. Comput.*, Vol. 18, pp. 1163–1186, 1997.

[11] B. N. Parlett and O. A. Marques. An Implementation of the dqds Algorithm (Positive Case). *Lin. Alg. Appl*, Vol. 309, No. 1-3, pp. 217–259, 2000.

[12] T. Sakurai and H. Tadano. CIRR: a Rayleigh-Ritz type method with counter integral for generalized eigenvalue problems. *Hokkaido Math. J.*, Vol. 36, pp. 745–757, 2007.

[13] T. Yamashita, K. Kimura and Y. Yamamoto. A new subtraction-free formula for lower bounds of the minimal singular value of an upper bidiagonal matrix. *Numerical Algorithms*, Vol. 69, Issue 4, pp. 893–912, 2015.

[14] Y. Yamamoto. On the optimality and sharpness of Laguerre ' s lower bound on the smallest eigenvalue of a symmetric positive definite matrix. *Applications of Mathematics*, Vol. 62, Issue 4, pp. 319–331, 2017.