Abstract—The orthogonal qr algorithm with shifts (oqds algorithm), proposed by von Matt, is an algorithm for computing the singular values of bidiagonal matrices. This algorithm is accurate in terms of relative error, and it is also applicable to general triangular matrices. In particular, for lower tridiagonal matrices, BLAS Level 2.5 routines are available in preprocessing stage for this algorithm. BLAS Level 2.5 routines are faster than BLAS Level 2 routines widely used in preprocessing for bidiagonalization. Generally, it takes $O(n^3)$ operations to reduce a full $n$-by-$n$ matrix to a band matrix such as bidiagonal or lower tridiagonal matrix. On the other hand, computing the singular values of a bidiagonal or lower tridiagonal matrices takes only $O(n^2)$ operations. Consequently, if we have an algorithm for computing the singular values of lower tridiagonal matrices, we can expect that the total computation time including preprocessing to obtain the singular values is reduced.

In this paper, we consider the oqds algorithm for lower tridiagonal matrices. We propose a shift strategy for lower tridiagonal matrices to accelerate convergence and derive criteria for deflation or splitting.

1. Introduction

In 1997, von Matt proposed an algorithm, based on Rutishauser’s qr algorithm [1], called orthogonal qr algorithm with shifts (oqds algorithm) for computing the singular values of bidiagonal matrices in which all the transformations consist of Givens rotations [2]. It is shown that the oqds algorithm is also applicable to general triangular matrices [3].

In this paper, we shall consider the application of the oqds algorithm to lower tridiagonal matrices. It allows us to use lower-tridiagonalization as pre-processing instead of bidiagonalization. The lower-tridiagonalization is less computational complexity than the bidiagonalization. Further, we can adopt BLAS Level 2.5 routines with efficient cache reuse which are faster than BLAS Level 2 routines for implementation of the lower-tridiagonalization. The oqds algorithm for lower tridiagonal matrices thus enables us to reduce the total computation time to obtain the singular values of general triangular matrices.

For practical use, we should design good shift strategies for convergence acceleration and good convergence criteria for accurate computation. However, appropriate shift strategies and convergence criteria for lower tridiagonal matrices have not been proposed yet. In this paper, we propose a shift strategy consisting of the generalized Newton shift and associated two methods, Laguerre shift and Kato-Temple shift, and the well known Gerschgorin shift. Moreover, we design new convergence criteria for deflation and splitting required for the implementation of the oqds algorithm. By the criteria, we can do the convergence test for lower tridiagonal matrices. At the end, we show some results of numerical experiments to compare the oqds algorithms for bidiagonal matrices and for lower tridiagonal matrices.

2. Orthogonal QR Algorithm for Lower Tridiagonal Matrix

Let

$$L = \begin{bmatrix} \alpha_1 & & & & \\ \beta_1 & \alpha_2 & & & \\ & \gamma_1 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \gamma_{n-2} & \beta_{n-1} \alpha_n \end{bmatrix}$$

be an $n$-by-$n$ lower tridiagonal matrix. One step of Cholesky LR method [4] with shift $\sigma^2$ transforms the lower tridiagonal matrix $L$ into the upper tridiagonal matrix $U$ by

$$L^T L - \sigma^2 I = U^T U.$$
the generalized Givens transformation which is numerically stable. The qd’s algorithm is formulated as the iteration of the implicit Cholesky LR step.

### 2.1 Implicit Cholesky Decomposition

The implicit Cholesky decomposition computes an upper tridiagonal matrix $U$ from $L$ and $\sigma$ by an orthogonal transformation

$$Q \begin{bmatrix} L \\ 0 \end{bmatrix} = \begin{bmatrix} U \\ \sigma I \end{bmatrix},$$

where $Q$ is a $2n$-by-$2n$ orthogonal matrix. It is readily verified that, for the same $L$ and $\sigma$, the same $U$ is obtained by (3) as by the Cholesky LR method (2). The orthogonal matrix $Q$ is given by superposition of the Givens and the generalized Givens transformations on $\mathbb{R}^2$.

**Definition 2.1** (Generalized Givens transformation [2]). Let $\sigma$ be a real. The transformation on $\mathbb{R}^2$

$$G \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} r \\ \sigma \end{bmatrix}$$

by a 2-by-2 orthogonal matrix $G$ is called the generalized Givens transformation if $r = \pm \sqrt{x_1^2 + x_2^2 - \sigma^2}$ and $\sigma^2 < x_1^2 + x_2^2$.

Such a matrix $G$ is uniquely determined by

$$G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

and

$$\begin{bmatrix} c \\ s \end{bmatrix} = \frac{1}{x_1^2 + x_2^2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} r \\ \sigma \end{bmatrix}.$$  

It should be noted that the generalized Givens transformation is equal to the ordinary Givens transformation if $\sigma = 0$.

The procedure of the generalized Givens transformation is shown in Algorithm 1. The first step of the implicit Cholesky decomposition is a series of three orthogonal transformations:

$$G_1 \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \tilde{\alpha}_1 \\ \tilde{\beta}_1 \end{bmatrix},$$
$$G_2 \begin{bmatrix} \gamma_1 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \tilde{\gamma}_1 \\ \tilde{\beta}_1 \end{bmatrix},$$
$$G_3 \begin{bmatrix} \sigma \end{bmatrix} = \begin{bmatrix} \tilde{\sigma} \end{bmatrix}.$$  

In the first column, the first transformation (7) by $G_1$ generates the lower diagonal element $\sigma$. The second (8), the third (9) by $G_2$, $G_3$ vanish $\beta_1$, $\gamma_1$, respectively. Here, $G_1$ is the generalized Givens transformation for the first and the $(n + 1)$th rows and $G_2$, $G_3$ are Givens transformations for the first and the second rows, the first and the third rows, respectively. After the three transformations, we obtain the first column of the matrix on the right hand side of equation (3). Applying similar operations for second to $n$th columns successively, we obtain the upper tridiagonal matrix $U$ in the equation (3) and let the next $L$ be $U^T$ to continue the algorithm. The operations are numerically stable.

Algorithm 2 summarizes the above procedures. The subroutines “rotg” and “rot” appeared in the Algorithm 2 are basic BLAS routines. If we call the “rotg($x_1$, $x_2$, $c$, $s$)”, the rotation angle of the Givens transformation is stored in the arguments $c$ and $s$ with cos and sin forms. The subroutine “rot($x_1$, $x_2$, $c$, $s$)”
applies the Givens transformation defined by the arguments $c$ and $s$.

**Algorithm 2 Implicit Cholesky Decomposition for an $n$-by-$n$ lower tridiagonal matrix $L$ (icds($L$))**

\[
U := 0
\]

for $i = 1$ to $n$

\[
\tilde{a}_i := a_i
\]

\[
\text{rotg}(\tilde{a}_i, 0, 0, c, s)
\]

*eliminate subdiagonal element

\[
\text{rotg}(\tilde{a}_i, \gamma_i, c, s)
\]

\[
\text{rot}(\tilde{b}_i, \beta_i, c, s)
\]

*eliminate subsubdiagonal element

\[
\text{rot}(\tilde{c}_i, \gamma_i, c, s)
\]

end for

return $L$


### 3. Shift Strategy

In the implicit Cholesky decomposition, proper choice of the shift value $\sigma$ significantly accelerates convergence of the oqs algorithm. The shift value $\sigma$ must be smaller than the minimum singular value of the matrix $L$ to keep the positive-definiteness of $U^T U$. Therefore, we need a method to estimate the lower bound of the minimum singular value of the lower tridiagonal matrix $L$ or the minimum eigenvalue of $L^T L$.

In this section, we discuss four types of lower bounds of the minimum singular value or eigenvalue and design shift method using them.

#### 3.1 Gerschgorin Shift

**Theorem 3.1** (Gerschgorin [5]). For an $n$-by-$n$ matrix $A = (a_{ij})$, let us define

\[
R_i := \sum_{k \neq i} |a_{ik}|.
\]

Then, for any eigenvalue $\lambda$ of $A$, there exists an integer $i$ such as

\[
|\lambda - a_{ii}| \leq R_i.
\]

If the matrix $A$ is positive-definite symmetric, $\min(a_{ii} - R_i)$ gives a lower bound of the eigenvalues since all the eigenvalues of $A$ are positive real number.

#### 3.2 Generalized Newton Shift

For a positive-definite symmetric matrix $A$ and an arbitrary positive integer $p$, the value of $(\text{Tr}(A^{-p}))^{-1/p}$ is a lower bound of the eigenvalues of $A$. Then, finding the value of $\text{Tr}(L^T L)^{-p}$, we get a lower bound of the singular values of $L$. We consider a method of computing the value of $\text{Tr}(L^T L)^{-p}$ in this subsection.

#### Algorithm 3 Gerschgorin shift (gerschgorin($L$))

\[
\sigma := \alpha_{n-1}^2 + \beta_{n-2}^2 + \gamma_{n-3}^2 - |\alpha_{n-1} \gamma_{n-3} - \beta_{n-2} \gamma_{n-3}| - |\alpha_{n-2} \beta_{n-2}|
\]

if $\sigma \leq 0$ then

return 0
end if

\[
tmp := \alpha_{n-1}^2 + \beta_{n-2}^2 + \gamma_{n-3}^2 - |\alpha_{n-1} \gamma_{n-3} - \beta_{n-2} \gamma_{n-3}| - |\alpha_{n-2} \beta_{n-2}|
\]

if $\text{tmp} \leq 0$ then

return 0
else if $\text{tmp} < \sigma$ then

\[
\sigma := \text{tmp}
\]
end if

end for

\[
\sigma := \text{tmp}
\]

end if


Let $L$ be an $n$-by-$n$ lower tridiagonal matrix,

\[
\tilde{L} = \begin{bmatrix}
\tilde{a}_1 & \tilde{b}_1 & \tilde{a}_2 & \tilde{b}_2 & \tilde{a}_3 & \cdots & \tilde{a}_{n-2} & \tilde{b}_{n-1} & \tilde{a}_n
\end{bmatrix}
\]

(12)

determined from $L$ with shift $s$ by

\[
\tilde{L} \tilde{L}^T = LL^T - sl.
\]

(13)

The relationships among elements are given by

\[
\tilde{a}_i^2 + \tilde{b}_{i-1}^2 + \tilde{\gamma}_{i-2}^2 = \alpha_i^2 + \beta_{i-1}^2 + \gamma_{i-2}^2 - s,
\]

(14)

\[
\tilde{b}_{i-2} \tilde{\gamma}_{i-2} + \tilde{a}_{i-1} \beta_{i-1} = \beta_{i-2} \gamma_{i-2} + \alpha_{i-1} \beta_{i-1},
\]

(15)

\[
\tilde{a}_{i-2} \gamma_{i-2} = \alpha_{i-2} \gamma_{i-2}.
\]

(16)

Differentiating equations (14)–(16) with respect to $s$, we obtain

\[
2\tilde{a}_i \tilde{a}_i' + 2\tilde{b}_{i-1} \tilde{b}_{i-1}' + 2\tilde{\gamma}_{i-2} \tilde{\gamma}_{i-2}' = -1,
\]

(17)

\[
\tilde{b}_{i-2} \tilde{\gamma}_{i-2} + \tilde{b}_{i-2} \tilde{\gamma}_{i-2}' + \tilde{a}_{i-1} \beta_{i-1} + \tilde{a}_{i-1} \beta_{i-1}' = 0,
\]

(18)
\[ \ddot{a}_{i-2} \ddot{y}_{i-2} + \ddot{a}_{i-2} \ddot{y}_{i-2} = 0. \] (19)

Note that the \( \alpha_i, \beta_i, \gamma_i \) are independent of \( s \) but the \( \ddot{a}_i, \ddot{b}_i, \ddot{y}_i \) are not. Differentiating once more, we get

\[
2\dddot{a}_i^2 + 2\dddot{a}_i \dddot{a}_i' + 2\dddot{b}_i^2 \\
+ 2\dddot{b}_i \dddot{b}_i' + 2\dddot{y}_i^2 + 2\dddot{y}_i \dddot{y}_i' = 0.
\] (20)

\[
\dddot{a}_{i-2}^2 + 2\dddot{a}_{i-2} \dddot{a}_{i-2}' + \dddot{a}_{i-2} \dddot{a}_{i-2}'' = 0,
\] (21)

\[
\dddot{b}_{i-2}^2 + 2\dddot{b}_{i-2} \dddot{b}_{i-2}' + \dddot{b}_{i-2} \dddot{b}_{i-2}'' = 0,
\] (22)

Let us write the eigenvalues of the matrix \( LL^T \) by \( \lambda_1, \lambda_2, \ldots, \lambda_n \). Then, the characteristic polynomial of the matrix \( LL^T \)

\[
f(s) = \det(LL^T - sI) \\
= (\lambda_1 - s)(\lambda_2 - s) \cdots (\lambda_n - s),
\] (23)

because of the triangularity of the matrix \( L \), is expressed by

\[
f(s) = \ddot{a}_1 \ddot{a}_2 \cdots \ddot{a}_n.
\] (24)

Let us define

\[
g(s) := -\frac{f'(s)}{f(s)} = -\frac{\dddot{a}_1 \dddot{a}_2 - \dddot{a}_2 \dddot{a}_1 - \cdots - \dddot{a}_n \dddot{a}_1 \dddot{a}_n}{\dddot{a}_1^2} \tag{25}
\]

\[
h(s) := g'(s) = \dddot{a}_1 \dddot{a}_2 \cdots \dddot{a}_n - \dddot{a}_2 \dddot{a}_1 \dddot{a}_n - \cdots - \dddot{a}_n \dddot{a}_1 \dddot{a}_2 \dddot{a}_n \tag{26}
\]

so that \( g(0) = \text{Tr}(LL^T)^{-1} \) and \( h(0) = \text{Tr}((LL^T)^{-2}) \). Each \( \dddot{a}_i \) tends to \( \alpha_i \) as \( s \to 0 \). Hence, we can calculate the value of \( \dddot{a}_1', \dddot{b}_1', \dddot{y}_1', \dddot{a}_2', \dddot{b}_2', \dddot{y}_2' \) at \( s = 0 \) from \( \alpha_i, \beta_i, \gamma_i \) by using (17)–(22), and then \( g(0) \) and \( h(0) \) by (25) and (26). It is clear by the definition of \( g(0) \) and \( h(0) \) that the values are always nonnegative without numerical error (in infinite-precision arithmetic). The procedure of computation for the traces of lower tridiagonal matrix \( L \) is shown in Algorithm 4. The generalized Newton shift is value of \( 1/\sqrt{\text{tr}(2)} \).

### 3.3 Laguerre Shift

If we already have the value of \( \text{Tr}((LL^T)^{-1}) \) and \( \text{Tr}((LL^T)^{-2}) \), we could improve the sharpness of the shift by \( O(1) \) operation. Laguerre shift is one of the methods to improve the shift value.

**Theorem 3.2 (Laguerre [7]).** For an \( n \times n \) positive-definite symmetric penta-diagonal matrix \( B = LL^T \), let \( \theta \) be the following:

\[
\theta := \frac{n}{\text{Tr}(B^{-1}) + \sqrt{(n-1)(n\text{Tr}(B^{-2}) - \text{Tr}(B^{-1})^2)}}.
\]

Then, the \( \theta \) is a lower bound of the eigenvalues of \( B \) which is greater than \( \text{Tr}(B^{-1}) \) and \( \text{Tr}(B^{-2})^{1/2} \).

If the value \( n\text{Tr}(B^{-2}) - \text{Tr}(B^{-1})^2 \) is negative, Laguerre shift is useless. In that case, we adopt the generalized Newton shift. Algorithm 5 shows a procedure of Laguerre method.

### 3.4 Kato-Temple Shift

There is another lowerbound, Kato-temple shift.

**Theorem 3.3 (Kato-Temple [8]).** For an \( n \times n \) symmetric matrix \( A_n \), let \( A_{n-1} \) denote the submatrix of \( A_n \) obtained by deleting the last row and column. For any lower bound \( \lambda' \) of the eigenvalues of \( A_{n-1} \), and for any \( x \in \mathbb{R}^n, ||x|| = 1 \), let \( \rho = x^TAx \). Then, if \( \rho < \lambda' \), the value

\[
\rho - \frac{||Ax - \rho x||^2}{\lambda' - \rho} \leq \lambda_{\min}(A_n)
\]

gives a lower bound of the eigenvalues of \( A_n \).

We choose \( x = (0, \ldots, 0, 1)^T \). The method requires \( \lambda' \) which is a lower bound for the submatrix \( A_{n-1} \), but the generalized Newton method enables us to find the lower bound of \( A_{n-1} \).

---

**Algorithm 4 Computation for the traces (\text{trace}(L))**

\[
\alpha_1' := -1/(2\alpha_1) \\
\beta_1' := -\alpha_2'\beta_1/\alpha_1 \\
\gamma_1' := -\alpha_1'\gamma_1/\alpha_1 \\
\alpha_1 := (-\beta_1'\beta_1 - 0.5)/\alpha_2 \\
\beta_2 := -(\gamma_1'\beta_1 + \gamma_1\gamma_1' + \alpha_1'\beta_2)/\alpha_2 \\
\gamma_1 := (1 + 2 \times \gamma_1\gamma_1' + 2\beta_2\beta_2)/(2\alpha_3) \\
\alpha_2 := -\alpha_1^2/\alpha_3 \\
\beta_2' := -\alpha_1'\beta_2 + 2\alpha_1'\beta_2' - \alpha_1'\beta_3/\alpha_3 \\
\gamma_1' := -(\alpha_2'\gamma_1 + 2\gamma_1\gamma_1')/\alpha_1 \\
\alpha_2 := -\beta_2^2 + \beta_2\beta_3 + \alpha_2'^2/\alpha_2 \\
\beta_3 := -(\gamma_1'\beta_1 + 2\gamma_1\beta_1 + \gamma_1\beta_1' + \alpha_1\beta_2 + 2\alpha_1\beta_3)/\alpha_2 \\
\alpha_2 := -(\gamma_1^2 + \gamma_1\gamma_1' + \beta_2^2 + \beta_3\beta_3' + \alpha_3'^2)/\alpha_3 \\
\text{for } i = 4 \text{ to } N \text{ do} \\
\gamma_1 := -(\alpha_1'\gamma_1 + \gamma_1\gamma_1')/\alpha_1 \\
\beta_2 := -(\gamma_1'\beta_1 + 2\gamma_1\beta_1 + \gamma_1\beta_1' + \alpha_1'\beta_2 + 2\alpha_1'\beta_2)/\alpha_2 \\
\gamma_1 := -(\alpha_2'\gamma_1 + 2\gamma_1\gamma_1' + \beta_2^2 + \beta_3\beta_3' + \alpha_3'^2)/\alpha_3 \\
\text{end for} \\
\text{for } i = 1 \text{ to } N \text{ do} \\
\gamma_1 := -(\alpha_1'\gamma_1 + \gamma_1\gamma_1')/\alpha_1 \\
\beta_2 := -(\gamma_1'\beta_1 + 2\gamma_1\beta_1 + \gamma_1\beta_1' + \alpha_1'\beta_2 + 2\alpha_1'\beta_2)/\alpha_2 \\
\gamma_1 := -(\alpha_2'\gamma_1 + 2\gamma_1\gamma_1' + \beta_2^2 + \beta_3\beta_3' + \alpha_3'^2)/\alpha_3 \\
\text{end for} \\
\text{return } (\text{trace}(1), \text{trace}(2))
\]

**Algorithm 5 Laguerre shift (laguerre(try, tr))**

\[
(\text{try}, \text{tr}) := \text{trace}(L) \\
\text{tmp} := n \times \text{tr} - \text{tr}^2 \\
\text{if } \text{tmp} > 0 \text{ then} \\
\text{return } n/(\text{tr} + \sqrt{(n-1) \times \text{tmp}}) \\
\text{else return } 0 \\
\text{end if}
\]
in computation of the lower bound of \( A_n \). Consequently, we obtain one more improved shift value by \( O(1) \) operation. Algorithm 6 shows a procedure of Kato-Temple method. The procedure of the proposed shift composed by the generalized Newton, Laguerre and Kato-Temple is shown in Algorithm 7. We adopt the largest value of them.

### 3.5 Applying Shift

Among the shifts discussed in this section, we cannot determine which is the most effective. The sharpness of each shift depends on the type of matrix, and the type of matrix is unknown before computing. Laguerre shift often gives sharp shift but if the value of \( n \text{Tr}(B^{-2}) - \text{Tr}(B^{-1})^2 \) is negative, we cannot adopt the shift. Besides, even if a shift value is smaller than minimum singular value, iteration of the qogs algorithm might fail. For example, if \( \sigma > x_1^2 + x_2^2 \), we could not apply the generalized Givens transformation. Gerschgorin shift gives a sharp value if the subdiagonal and second-subdiagonal elements are small. On the other hand, if non-diagonal elements are too large, Gerschgorin shift gives useless value such as zero or negative value. In such a case, we should choose another shift. Generalized Newton shift always gives usable value in the case other shifts failed.

For those reasons, we should design a proper shift strategy. Generally, non-diagonal elements converge to zero in the qogs algorithm, and after deflation or splitting, the eigenvalues of the matrix \( A \) become more clustered. Therefore, we adopt the largest value of generalized Newton, Laguerre, Kato-Temple shift first, and if the generalized Givens transformation failed, then we move to the Gerschgorin shift. Then, one step of the qogs algorithm works as Algorithm 8. The subroutine “gerschgorin(L)” returns the value of Gerschgorin shift of matrix \( L \).

### 4. Convergence Criteria

It is nontrivial how to assess a series of matrices generated by the iterative process of the qogs algorithm converges sufficiently. Besides, in the implementation of this algorithm, deflation and splitting are required for activating the shift method. In this section, we consider the situation that deflation or splitting is available where the values of subdiagonal and second-subdiagonal elements are so small.

Let us write

\[ L := L - \beta_k e_{k+1} e_k^T \]
Similarly, we get the numerical criterion for neglecting a second-subdiagonal element $\gamma_k$. On the setting of

$$
\hat{L} := L - \gamma_k \mathbf{e}_k \mathbf{e}_k^T,
$$

the perturbation matrices are given by

$$
E'_1 := \gamma^2 \mathbf{e}_k \mathbf{e}_k^T + \alpha_{k+2} \mathbf{e}_{k+2} \mathbf{e}_{k+2}^T + \alpha_{k+1} \mathbf{e}_{k+1} \mathbf{e}_{k+1}^T + \beta \mathbf{e}_k \mathbf{e}_k^T + \mathbf{e}_{k+1} \mathbf{e}_{k+1}^T, \\
E'_2 := \gamma^2 \mathbf{e}_{k+2} \mathbf{e}_{k+2}^T + \alpha_{k+2} \mathbf{e}_{k+2} \mathbf{e}_{k+2}^T + \alpha_{k+1} \mathbf{e}_{k+1} \mathbf{e}_{k+1}^T + \beta \mathbf{e}_k \mathbf{e}_k^T + \mathbf{e}_{k+1} \mathbf{e}_{k+1}^T.
$$

Then, by evaluating the $1$- and $\infty$-norms of these matrices, we obtain the criterion for neglecting a second-subdiagonal element $\gamma_k$ as follows:

$$
\sigma^2 + \|\gamma_k\| \|\gamma_k\| + \min(\|\alpha_{k+2}\| + |\beta_k|, |\alpha_k| + |\beta_k|) = \sigma^2. \quad (36)
$$

For the matrices in iteration, we perform deflation and splitting as follows:

1) If $\beta_{k-1}$ and $\gamma_{k-2}$ in the last row satisfy the criteria (35) and (36), then we deflate the matrix by deleting the last row and column.
2) If $\beta_{k-1}$, $\gamma_{k-1}$ and $\gamma_{k-2}$ satisfy the criteria (35) and (36), then we split the matrix into two submatrices formed by rows and columns 1 to $k-1$ and $k$ to $n$, respectively.

5. Numerical Experiments

Some numerical experiments were performed for the oqds algorithms for bidiagonal matrices and for lower tridiagonal matrices. The singular values of square random matrices were computed by the oqds algorithm for bidiagonal matrices by von Matt and by the oqds algorithm for lower tridiagonal matrices which we propose. It should be noted that: The oqds for bidiagonal matrices were applied to random bidiagonal matrices and the proposed oqds algorithm for lower tridiagonal matrices were applied to random lower tridiagonal matrix. The numerical experiments were performed on a Linux PC with Intel Core i7 920 (Nehalem) 2.66GHz and DDR3-1066 12GB memory. Table 1 shows the computation time of each algorithm. The first row shows the size of matrices. The second and the third rows show the computation time taken by the oqds algorithm for bidiagonal matrices and for lower tridiagonal matrices, respectively.

<table>
<thead>
<tr>
<th>matrix size</th>
<th>10000</th>
<th>20000</th>
<th>30000</th>
</tr>
</thead>
<tbody>
<tr>
<td>oqds for bidiagonal</td>
<td>11.764</td>
<td>43.243</td>
<td>93.080</td>
</tr>
<tr>
<td>proposed oqds for lower tridiagonal</td>
<td>27.089</td>
<td>100.013</td>
<td>210.225</td>
</tr>
</tbody>
</table>

5.1 Discussion

Hence, in order to compute the eigenvalues of matrices of the same size, the oqds algorithm for lower tridiagonal matrices is expected to take a longer computation time than the oqds for bidiagonal matrices. From Table 1, we observe that the computation time in the former algorithm is not extremely
longer than the latter algorithm: the former is two or three times slower than the latter.

This observation demonstrates that the oqds algorithm for lower tridiagonal matrices is practically useful for the general dense matrices. Commonly, the computation of the singular values of a dense matrix is twofold:

1) preprocess of reducing into a sparse band matrix.
2) singular computation of the sparse band matrix.

The computation time for preprocess is estimated $O(n^3)$ while for the singular value computation $O(n^2)$. Hence, a vast amount of the computation time is consumed by the preprocess. On the preprocess for dense matrices, it is reported in [11] that the reduction into a lower tridiagonal matrix is about 50% faster than that into bidiagonal matrices. Therefore, the total time of preprocess into a lower tridiagonal matrix and the oqds for lower tridiagonal matrices is much faster than the time of preprocess into bidiagonal matrices and the oqds for bidiagonal matrices.

6. Conclusions

We proposed the oqds algorithm for lower tridiagonal matrices. Though computing singular values of lower tridiagonal matrices takes longer time than bidiagonal matrices, preprocess reducing dense matrices into lower tridiagonal matrices takes less time than into bidiagonal matrices. Not only simple reduction of computational complexity, we can apply the BLAS Level 2.5 routines to lower tridiagonalization. The BLAS Level 2.5 routines are more cache efficient than BLAS Level 2 routines commonly applied to bidiagonalization. A cache efficient algorithm saves a number of memory accesses which waste a big time. The computation time for preprocess is estimated $O(n^3)$ while for the singular value computation $O(n^2)$, hence, a vast amount of the computation time is consumed by the preprocess. Therefore, if we can compute the singular values of lower tridiagonal matrices not so longer than for bidiagonal matrices, it is expected that total computation time decreases extremely.

For an implementation of this algorithm, we proposed a new shift strategy consisting of the generalized Newton shift and associated two methods, Laguerre shift and Kato-Temple shift, and the well known Gerschgorin shift. Moreover, we design new convergence criteria for deflation and splitting required for the implementation of the oqds algorithm. By the criteria, we can do the convergence test for lower tridiagonal matrices.

As a result, the algorithm computes the singular values of a lower tridiagonal matrix within $O(n^2)$ computation time. Although it takes about two or three times as long time for tridiagonal matrices as for bidiagonal matrices, proposed algorithm is expected to be faster than the conventional methods since the preprocessing requires $O(n^3)$ operations and takes much larger time than the oqds algorithm.

As a future work, we have to perform more experiment to compare the computation time including preprocessing. Furthermore, exact error analysis should be made and we ought to check out the accuracy of the algorithm after improving the implementation and setting proper test matrices which have known eigenvalues.

Acknowledgments

The authors would like to express a sincere gratitude to Professor Shuhei Kamioka and Mr. Kazuki Maeda for valuable discussions and comments. This work was supported by JSPS KAKENHI Grant Number 24360038.

References