Implementation of the Orthogonal QD Algorithm for Lower Tridiagonal Matrices

Sho ARAKI*[1], Hiroki TANAKA*[2], Kinji KIMURA*[3] and Yoshimasa NAKAMURA*[4]

*Graduate School of Informatics, Kyoto University, Yoshidahonmachi, Sakyo-ku, Kyoto 606-8501, JAPAN
[1]contact author:araki@amp.i.kyoto-u.ac.jp
[2]hirotnk@amp.i.kyoto-u.ac.jp
[3]kkimur@i.kyoto-u.ac.jp
[4]ynaka@i.kyoto-u.ac.jp

Abstract— *The orthogonal qd 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.

(This paper is submitted to PDPTA'13)

Keywords

singular value computation, orthogonal qd algorithm, lower tridiagonal matrix, shift strategy, deflation, splitting.

1. Introduction

In 1997, von Matt proposed an algorithm, based on Rutishauser's qd algorithm [1], called orthogonal qd 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 QD Algorithm for Lower Tridiagonal Matrix

Let

$$L = \begin{bmatrix} \alpha_1 & & & & \\ \beta_1 & \alpha_2 & & & \\ \gamma_1 & \beta_2 & \alpha_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-2} & \beta_{n-1} & \alpha_n \end{bmatrix} \tag{1}$$

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

$$L^T L - \sigma^2 I = U^T U. \tag{2}$$

Then, we set $L := U^T$. By repeating this procedure iteratively, the diagonal elements of the matrix L converge to the singular values of the matrix L and the non-diagonal elements get into zero. It is known that the Cholesky decomposition is numerically unstable; the Cholesky decomposition may collapse even if the shift value σ is zero. For resolving this problem, we use the *implicit* Cholesky decomposition [2]. The implicit Cholesky decomposition is designed by using

Algorithm 1 Generalized Givens transformation (rotg2(x_1, x_2, σ, c, s))

```
$scale := \max(|x_1|, |x_2|)$
if $scale = 0$ then
  $c := 1$
  $s := 0$
else
  $x_1 := x_1/scale$
  $x_2 := x_2/scale$
  $sig := \sigma/scale$
  $norm2 := x_1^2 + x_2^2$
  $r := \sqrt{norm2 - sig^2}$
  $c := (x_1 \times r + x_2 \times sig)/norm2$
  $s := (x_2 \times r - x_1 \times sig)/norm2$
  $x_1 := scale \times r$
  $x_2 := \sigma$
end if
```

the generalized Givens transformation which is numerically stable. The oqds 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 σ by an orthogonal transformation

$$Q\begin{bmatrix} L \\ 0 \end{bmatrix} = \begin{bmatrix} U \\ \sigma I \end{bmatrix}, \tag{3}$$

where Q is a $2n$-by-$2n$ orthogonal matrix. It is readily verified that, for the same L and σ, 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 σ be a real. The transformation on $\mathbb{R}^2$

$$G\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} r \\ \sigma \end{bmatrix} \tag{4}$$

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

$$G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \tag{5}$$

$$\begin{bmatrix} c \\ s \end{bmatrix} = \frac{1}{x_1^2 + x_2^2}\begin{bmatrix} x_1 & x_2 \\ x_2 & -x_1 \end{bmatrix}\begin{bmatrix} r \\ \sigma \end{bmatrix}. \tag{6}$$

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:

$$G_1\begin{bmatrix} \alpha_1 & & & \\ \beta_1 & \alpha_2 & & \\ \gamma_1 & \beta_2 & \ddots & \\ & \gamma_2 & \ddots & \\ & & \ddots & \\ & & & \\ 0 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} = \begin{bmatrix} \tilde{\alpha}_1 & & & \\ \beta_1 & \alpha_2 & & \\ \gamma_1 & \beta_2 & \ddots & \\ & \gamma_2 & \ddots & \\ & & \ddots & \\ & & & \\ \sigma & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}, \tag{7}$$

$$G_2\begin{bmatrix} \tilde{\alpha}_1 & & & \\ \beta_1 & \alpha_2 & & \\ \gamma_1 & \beta_2 & \ddots & \\ & \gamma_2 & \ddots & \\ & & \ddots & \\ & & & \\ \sigma & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} = \begin{bmatrix} \tilde{\tilde{\alpha}}_1 & \tilde{\beta}_1 & 0 & \\ 0 & \tilde{\alpha}_2 & & \\ \gamma_1 & \beta_2 & \ddots & \\ & \gamma_2 & \ddots & \\ & & \ddots & \\ & & & \\ \sigma & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}, \tag{8}$$

and then

$$G_3\begin{bmatrix} \tilde{\tilde{\alpha}}_1 & \tilde{\beta}_1 & 0 & \\ 0 & \tilde{\alpha}_2 & & \\ \gamma_1 & \beta_2 & \ddots & \\ & \gamma_2 & \ddots & \\ & & \ddots & \\ & & & \\ \sigma & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} = \begin{bmatrix} \check{\alpha}_1 & \check{\beta}_1 & \check{\gamma}_1 & \\ 0 & \tilde{\alpha}_2 & & \\ 0 & \tilde{\beta}_2 & \ddots & \\ & \gamma_2 & \ddots & \\ & & \ddots & \\ & & & \\ \sigma & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}. \tag{9}$$

In the first column, the first transformation (7) by G_1 generates the lower diagonal element σ. The second (8), the third (9) by G_2, G_3 vanish β_1, γ_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 nth 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 $n$ do
    $\check{\alpha}_i := \alpha_i$
    rotg2($\check{\alpha}_i, 0, \sigma, c, s$)

    *eliminate subdiagonal element
    rotg($\check{\alpha}_i, \beta_i, c, s$)
    rot($\check{\beta}_i, \beta_i, c, s$)

    *eliminate subsubdiagonal element
    rotg($\check{\alpha}_i, \gamma_i, c, s$)
    rot($\check{\beta}_i, \beta_i, c, s$)
    rot($\check{\gamma}_i, \gamma_i, c, s$)
end for
return $\check{L}$
```

3. Shift Strategy

In the implicit Cholesky decomposition, proper choice of the shift value σ significantly accelerates convergence of the oqds algorithm. The shift value σ 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*

$$R_i := \sum_{k \neq i} |a_{ik}|. \tag{10}$$

Then, for any eigenvalue λ of A, there exists an integer i such as

$$|\lambda - a_{ii}| \leq R_i. \tag{11}$$

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 $(\mathrm{Tr}(A^{-p}))^{-1/p}$ is a lower bound of the eigenvalues of A. Then, finding the value of $\mathrm{Tr}\{(L^T L)^{-p}\}$, we get a lower bound of the singular values of L. We consider a method of computing the value of $\mathrm{Tr}\{(L^T L)^{-p}\}$ in this subsection.

Algorithm 3 Gerschgorin shift (gerschgorin(L))

```
$\sigma := \alpha_{n-1}^2 + \beta_{n-2}^2 + \gamma_{n-3}^2 - |\alpha_{n-3}\gamma_{n-3}| - |\beta_{n-3}\gamma_{n-3} + \alpha_{n-2}\beta_{n-2}|$
if $\sigma \leq 0$ then
    return 0
end if
$tmp := \alpha_{n-2}^2 + \beta_{n-3}^2 + \gamma_{n-4}^2 - |\alpha_{n-4}\gamma_{n-4}| - |\beta_{n-4}\gamma_{n-4} + \alpha_{n-3}\beta_{n-3}| - |\beta_{n-3}\gamma_{n-3} + \alpha_{n-2}\beta_{n-2}|$
if $tmp \leq 0$ then
    return 0
else if $tmp < \sigma$ then
    $\sigma := tmp$
end if
for i = N - 2 to 3 do
    $tmp := \alpha_i^2 + \beta_{i-1}^2 + \gamma_{i-2}^2 - |\alpha_{i-2}\gamma_{i-2}| - |\beta_{i-1}\gamma_{i-1} + \alpha_i\beta_i| - |\beta_{i-2}\gamma_{i-2} + \alpha_{i-1}\beta_{i-1}| - |\alpha_i\gamma_i|$
    if $tmp \leq 0$ then
        return 0
    else if $tmp < \sigma$ then
        $\sigma := tmp$
    end if
end for
$tmp := \alpha_2^2 + \beta_1^2 - |\alpha_1\beta_1| - |\beta_1\gamma_1 + \alpha_2\beta_2| - |\alpha_2\gamma_2|$
if $tmp \leq 0$ then
    return 0
else if $tmp < \sigma$ then
    $\sigma := tmp$
end if
$tmp := \alpha_1^2 - |\alpha_1\beta_1| - |\alpha_1\gamma_1|$
if $tmp \leq 0$ then
    return 0
else if $tmp < \sigma$ then
    $\sigma := tmp$
end if
```

Let $\bar{L}$ be an n-by-n lower tridiagonal matrix,

$$\bar{L} = \begin{bmatrix} \bar{\alpha}_1 & & & & \\ \bar{\beta}_1 & \bar{\alpha}_2 & & & \\ \bar{\gamma}_1 & \bar{\beta}_2 & \bar{\alpha}_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \bar{\gamma}_{n-2} & \bar{\beta}_{n-1} & \bar{\alpha}_n \end{bmatrix} \tag{12}$$

determined from L with shift s by

$$\bar{L}\bar{L}^T = LL^T - sI. \tag{13}$$

The relationships among elements are given by

$$\bar{\alpha}_i^2 + \bar{\beta}_{i-1}^2 + \bar{\gamma}_{i-2}^2 = \alpha_i^2 + \beta_{i-1}^2 + \gamma_{i-2}^2 - s, \tag{14}$$

$$\bar{\beta}_{i-2}\bar{\gamma}_{i-2} + \bar{\alpha}_{i-1}\bar{\beta}_{i-1} = \beta_{i-2}\gamma_{i-2} + \alpha_{i-1}\beta_{i-1}, \tag{15}$$

$$\bar{\alpha}_{i-2}\bar{\gamma}_{i-2} = \alpha_{i-2}\gamma_{i-2}. \tag{16}$$

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

$$2\bar{\alpha}_i\bar{\alpha}'_i + 2\bar{\beta}_{i-1}\bar{\beta}'_{i-1} + 2\bar{\gamma}_{i-2}\bar{\gamma}'_{i-2} = -1, \tag{17}$$

$$\bar{\beta}'_{i-2}\bar{\gamma}_{i-2} + \bar{\beta}_{i-2}\bar{\gamma}'_{i-2} + \bar{\alpha}'_{i-1}\bar{\beta}_{i-1} + \bar{\alpha}_{i-1}\bar{\beta}'_{i-1} = 0, \tag{18}$$

$$\bar{\alpha}'_{i-2}\bar{\gamma}_{i-2} + \bar{\alpha}_{i-2}\bar{\gamma}'_{i-2} = 0. \tag{19}$$

Note that the α_i, β_i, γ_i are independent of s but the $\bar{\alpha}_i$, $\bar{\beta}_i$, $\bar{\gamma}_i$ are not. Differentiating once more, we get

$$\begin{aligned}2\bar{\alpha}'^2_i + 2\bar{\alpha}_i\bar{\alpha}''_i + 2\bar{\beta}'^2_{i-1}& \\ + 2\bar{\beta}_{i-1}\bar{\beta}''_{i-1} + 2\bar{\gamma}'^2_{i-2} + 2\bar{\gamma}_{i-2}\bar{\gamma}''_{i-2} &= 0,\end{aligned} \tag{20}$$

$$\bar{\alpha}''_{i-2}\bar{\gamma}_{i-2} + 2\bar{\alpha}'_{i-2}\bar{\gamma}'_{i-2} + \bar{\alpha}_{i-2}\bar{\gamma}''_{i-2} = 0, \tag{21}$$

$$\begin{aligned}\bar{\beta}''_{i-2}\bar{\gamma}_{i-2} + 2\bar{\beta}'_{i-2}\bar{\gamma}'_{i-2} + \bar{\beta}_{i-2}\bar{\gamma}''_{i-2}& \\ + \bar{\alpha}''_{i-1}\bar{\beta}_{i-1} + 2\bar{\alpha}'_{i-1}\bar{\beta}'_{i-1} + \bar{\alpha}_{i-1}\bar{\beta}''_{i-1} &= 0.\end{aligned} \tag{22}$$

Let us write the eigenvalues of the matrix LL^T by $\lambda_1, \lambda_2, \cdots, \lambda_n$. Then, the characteristic polynomial of the matrix $\bar{L}\bar{L}^T$

$$\begin{aligned}f(s) &= \det(LL^T - sI) \\ &= (\lambda_1 - s)(\lambda_2 - s)\cdots(\lambda_n - s),\end{aligned} \tag{23}$$

because of the triangularity of the matrix L, is expressed by

$$f(s) = \bar{\alpha}_1\bar{\alpha}_2\cdots\bar{\alpha}_n. \tag{24}$$

Let us define

$$g(s) := -\frac{f'(s)}{f(s)} = -2\frac{\bar{\alpha}'_1}{\bar{\alpha}_1} - 2\frac{\bar{\alpha}'_2}{\bar{\alpha}_2} - \cdots - 2\frac{\bar{\alpha}'_n}{\bar{\alpha}_n}, \tag{25}$$

$$\begin{aligned}h(s) &:= g'(s) \\ &= -2\frac{\bar{\alpha}''_1\bar{\alpha}_1 - \bar{\alpha}'^2_1}{\bar{\alpha}^2_1} - \cdots - 2\frac{\bar{\alpha}''_n\bar{\alpha}_n - \bar{\alpha}'^2_n}{\bar{\alpha}^2_n}\end{aligned} \tag{26}$$

so that $g(0) = \mathrm{Tr}\{(L^TL)^{-1}\}$ and $h(0) = \mathrm{Tr}\{(L^TL)^{-2}\}$. Each $\bar{\alpha}_i$ tends to α_i as $s \to 0$. Hence, we can calculate the value of $\bar{\alpha}'_i$, $\bar{\beta}'_i$, $\bar{\gamma}'_i$, $\bar{\alpha}''_i$, $\bar{\beta}''_i$, $\bar{\gamma}''_i$ at $s = 0$ from α_i, β_i, γ_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 $1/\sqrt{tr2}$.

Algorithm 4 Computation for the traces (trace(L))

```
$\alpha'_1 := -1/(2\alpha_1)$
$\beta'_1 := -\alpha'_1\beta_1/\alpha_1$
$\gamma'_1 := -\alpha'_1\gamma_1/\alpha_1$
$\alpha'_2 := (-\beta_1\beta'_1 - 0.5)/\alpha_2$
$\beta'_2 := -(\gamma'_1\beta_1 + \gamma_1\gamma'_1 + \alpha'_2\beta_2)/\alpha_2$
$\alpha'_3 := -(1 + 2 \times \gamma_1\gamma'_1 + 2\beta_2\beta'_2)/(2\alpha_3)$
$\alpha''_1 := -{\alpha'_1}^2/\alpha_1$
$\beta''_1 := -(\alpha''_1\beta_1 + 2\alpha'_1\beta'_1)/\alpha_1$
$\gamma''_1 := -(\alpha''_1\gamma_1 + 2\alpha'_1\gamma'_1)/\alpha_1$
$\alpha''_2 := -({\beta'_1}^2 + \beta_1\beta''_1 + {\alpha'_2}^2)/\alpha_2$
$\beta''_2 := -(\gamma''_1\beta_1 + 2\gamma'_1\beta'_1 + \gamma_1\beta''_1 + \alpha''_2\beta_2 + 2\alpha'_2\beta'_2)/\alpha_2$
$\alpha''_3 := -({\gamma'_1}^2 + \gamma_1\gamma''_1 + {\beta'_2}^2 + \beta_2\beta''_2 + {\alpha'_3}^2)/\alpha_3$
for $i = 4$ to $N$ do
    $\gamma'_{i-2} := -\alpha'_{i-2}\gamma_{i-2}/\alpha_{i-2}$
    $\beta'_{i-1} := -(\beta'_{i-2}\gamma_{i-2} + \beta_{i-2}\gamma'_{i-2} + \alpha'_{i-1}\beta_{i-1})/\alpha_{i-1}$
    $\alpha'_i := -(1 + 2\beta_{i-1}\beta'_{i-1} + 2\gamma_{i-2}\gamma'_{i-2})/(2\alpha_i)$
    $\gamma''_{i-2} := -(\alpha''_{i-2}\gamma_{i-2} + 2\alpha'_{i-2}\gamma'_{i-2})/\alpha_{i-2}$
    $\beta''_{i-1} := -(\beta''_{i-2}\gamma_{i-2} + 2\beta'_{i-2}\gamma'_{i-2}$
                $+\beta_{i-2}\gamma''_{i-2} + \alpha''_{i-1}\beta_{i-1} + 2\alpha'_{i-1}\beta'_{i-1})/\alpha_{i-1}$
    $\alpha''_i := -({\alpha'_i}^2 + {\beta'_{i-1}}^2 + \beta_{i-1}\beta''_{i-1} + {\gamma'_{i-2}}^2 + \gamma_{i-2}\gamma''_{i-2})/\alpha_i$
end for
$tr1 := 0$
for $i = 1$ to $N$ do
    $tr1 := tr1 - (2\alpha'_i/\alpha_i)$
end for
$tr2 := 0$
for $i = 1$ to $N$ do
    $tr2 := tr2 - 2(\alpha''_i\alpha_i - {\alpha'_i}^2)/\alpha_i^2$
end for
return $(tr1, tr2)$
```

Algorithm 5 Laguerre shift (laguerre($tr1, tr2$))

```
$(tr1, tr2) :=$ trace($L$)
$tmp := n \times tr2 - tr1^2$
if $tmp > 0$ then
    return $n/(tr1 + \sqrt{(n-1) \times tmp})$
else
    return 0
end if
```

3.3 Laguerre Shift

If we already have the value of $\mathrm{Tr}\{(LL^T)^{-1}\}$ and $\mathrm{Tr}\{(LL^T)^{-2}\}$, 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-by-n positive-definite symmetric penta-diagonal matrix $B = LL^T$, let θ be the following value:*

$$\theta := \frac{n}{\mathrm{Tr}\,(B^{-1}) + \sqrt{(n-1)\left(n\mathrm{Tr}\,(B^{-2}) - \mathrm{Tr}\,(B^{-1})^2\right)}}.$$

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

If the value $n\mathrm{Tr}\left(B^{-2}\right) - \mathrm{Tr}\left(B^{-1}\right)^2$ 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-by-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 λ^* 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*

$$\rho - \frac{\|A_nx - \rho x\|^2}{\lambda^* - \rho} \le \lambda_{\min}(A_n)$$

gives a lower bound of the eigenvalues of A_n.

We choose $x = (0, \ldots, 0, 1)^T$. The method requires λ^* 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}

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.

Algorithm 6 Kato-Temple method

```
  $x := (0, \dots, 0, 1)^T$
  $(tr1, tr2) := \mathrm{trace}(L_{n-1})$
  $\lambda^* := \mathrm{laguerre}(tr1, tr2)$
  $\rho := x^T L_{n-1} x$
  if $\rho < \lambda^*$ then
    return $\rho - \|A_n x - \rho x\|^2 / (\lambda^* - \rho)$
  else
    return $0$
  end if
```

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 $n\mathrm{Tr}\left(B^{-2}\right) - \mathrm{Tr}\left(B^{-1}\right)^2$ is negative, we cannot adopt the shift. Besides, even if a shift value is smaller than minimum singular value, iteration of the oqds 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 oqds 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 oqds 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 oqds 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

$$\hat{L} := L - \beta_k \mathbf{e}_{k+1} \mathbf{e}_k{}^T$$

Algorithm 7 Proposed shift (algshift(L))

```
  $\alpha'_1 := -1/(2\alpha_1)$
  $\beta'_1 := -\alpha'_1\beta_1/\alpha_1$
  $\gamma'_1 := -\alpha'_1\gamma_1/\alpha_1$
  $\alpha'_2 := (-\beta_1\beta'_1 - 0.5)/\alpha_2$
  $\beta'_2 := -(\gamma'_1\beta_1 + \gamma_1\gamma'_1 + \alpha'_2\beta_2)/\alpha_2$
  $\alpha'_3 := -(1 + 2 \times \gamma_1\gamma'_1 + 2\beta_2\beta'_2)/(2\alpha_3)$
  $\alpha''_1 := -{\alpha'_1}^2/\alpha_1$
  $\beta''_1 := -(\alpha''_1\beta_1 + 2\alpha'_1\beta'_1)/\alpha_1$
  $\gamma''_1 := -(\alpha''_1\gamma_1 + 2\alpha'_1\gamma'_1)/\alpha_1$
  $\alpha''_2 := -({\beta'_1}^2 + \beta_1\beta''_1 + {\alpha'_2}^2)/\alpha_2$
  $\beta''_2 := -(\gamma''_1\beta_1 + 2\gamma'_1\beta'_1 + \gamma_1\beta''_1 + \alpha''_2\beta_2 + 2\alpha'_2\beta'_2)/\alpha_2$
  $\alpha''_3 := -({\gamma'_1}^2 + \gamma_1\gamma''_1 + {\beta'_2}^2 + \beta_2\beta''_2 + {\alpha'_3}^2)/\alpha_3$
  for $i = 4$ to $N$ do
    $\gamma'_{i-2} := -\alpha'_{i-2}\gamma_{i-2}/\alpha_{i-2}$
    $\beta'_{i-1} := -(\beta'_{i-2}\gamma_{i-2} + \beta_{i-2}\gamma'_{i-2} + \alpha'_{i-1}\beta_{i-1})/\alpha_{i-1}$
    $\alpha'_i := -(1 + 2\beta_{i-1}\beta'_{i-1} + 2\gamma_{i-2}\gamma'_{i-2})/(2\alpha_i)$
    $\gamma''_{i-2} := -(\alpha''_{i-2}\gamma_{i-2} + 2\alpha'_{i-2}\gamma'_{i-2})/\alpha_{i-2}$
    $\beta''_{i-1} := -(\beta''_{i-2}\gamma_{i-2} + 2\beta'_{i-2}\gamma'_{i-2}$
                  $+\beta_{i-2}\gamma''_{i-2} + \alpha''_{i-1}\beta_{i-1} + 2\alpha'_{i-1}\beta'_{i-1})/\alpha_{i-1}$
    $\alpha''_i := -({\alpha'_i}^2 + {\beta'_{i-1}}^2 + \beta_{i-1}\beta''_{i-1} + {\gamma'_{i-2}}^2 + \gamma_{i-2}\gamma''_{i-2})/\alpha_i$
  end for
  $tr1 := 0$
  for $i = 1$ to $N - 1$ do
    $tr1 := tr1 - (2\alpha'_i/\alpha_i)$
  end for
  $tr2 := 0$
  for $i = 1$ to $N - 1$ do
    $tr2 := tr2 - 2(\alpha''_i\alpha_i - {\alpha'_i}^2)/\alpha_i^2$
  end for
  $\lambda^* := 1/sqrt(tr2)$
  $tmp := n \times tr2 - tr1^2$
  if $tmp > 0$ then
    $\lambda^* := \max(\lambda^*, n/(tr1 + \sqrt{(n-1) \times tmp}))$
  end if
  $tr1 := tr1 - (2\alpha'_N/\alpha_N)$
  $tr2 := tr2 - 2(\alpha''_N\alpha_N - {\alpha'_N}^2)/\alpha_N^2$
  $shift := 1/sqrt(tr2)$
  $x := (0, \dots, 0, 1)^T$
  $\rho := x^T L_{n-1} x$
  if $\rho < \lambda^*$ then
    $shift := \max(shift, \rho - \|A_n x - \rho x\|^2 / (\lambda^* - \rho))$
  end if
  $tmp := n \times tr2 - tr1^2$
  if $tmp > 0$ then
    $shift := \max(shift, n/(tr1 + \sqrt{(n-1) \times tmp}))$
  end if
  return $shift$
```

Algorithm 8 oqds step(oqds($L, shift$))

```
$flag := 0$
if $flag = 0$ then
    $\sigma :=$ algshift($L$)
else if $flag := 1$ then
    $\sigma :=$ gerschgorin($L$)
else
    $\sigma := 0$
end if
if $\sigma + shift = shift$ then
    $\breve{L} :=$ icds($L, 0$)
    $L := \breve{L}$
else
    $\breve{L} :=$ icds($L, \sigma$)
    if $\breve{\alpha} \neq \breve{\alpha}$ then
        $flag := flag + 1$
    else
        $shift := shift + \sigma$
        $L := \breve{L}$
    end if
end if
```

which is the matrix equal to L except for zero at $(k+1, k)$-entry. Then

$$L^T L = \hat{L}^T \hat{L} + E_1, \tag{27}$$
$$LL^T = \hat{L}\hat{L}^T + E_2 \tag{28}$$

hold, where

$$E_1 := \beta^2 \mathbf{e}_k \mathbf{e}_k{}^T + \alpha_{k+1}\beta_k \left(\mathbf{e}_k \mathbf{e}_{k+1}^T + \mathbf{e}_{k+1}\mathbf{e}_k^T\right) + \beta_k \gamma_{k-1} \left(\mathbf{e}_{k-1}\mathbf{e}_k^T + \mathbf{e}_k \mathbf{e}_{k-1}^T\right), \tag{29}$$

$$E_2 := \beta^2 \mathbf{e}_{k+1}\mathbf{e}_{k+1}{}^T + \alpha_k \beta_k \left(\mathbf{e}_{k-1}\mathbf{e}_k^T + \mathbf{e}_k \mathbf{e}_{k-1}^T\right) + \beta_k \gamma_k \left(\mathbf{e}_{k+1}\mathbf{e}_k^T + \mathbf{e}_k \mathbf{e}_{k+1}^T\right). \tag{30}$$

Theorem 4.1 (Weyl's monotonicity theorem [9], [10]). *For an n-by-n positive-definite matrix A, let $\lambda_i(A)$ denote the i-th largest eigenvalue of A. Then, there exist reals u_i and v_i such that*

$$\lambda_i\left(L^T L\right) = \lambda_i\left(\hat{L}^T \hat{L}\right) + u_i \|E_1\|_1, \tag{31}$$
$$\lambda_i\left(LL^T\right) = \lambda_i\left(\hat{L}\hat{L}^T\right) + v_i \|E_2\|_1 \tag{32}$$

where $|u_i| \leq 1$, $|v_i| \leq 1$.

From the definitions (29) and (30) of E_1 and E_2, we have

$$\|E_1\|_1 = \|E_1\|_\infty = |\beta_k| \left(|\alpha_{k+1}| + |\beta_k| + |\gamma_{k-1}|\right), \tag{33}$$
$$\|E_2\|_1 = \|E_2\|_\infty = |\beta_k| \left(|\alpha_k| + |\beta_k| + |\gamma_k|\right). \tag{34}$$

By Weyl's monotonicity theorem, we thus get the numerical deflation or splitting criterion to neglect a subdiagonal element β_k:

$$\sigma^2 + |\beta_k| \left(|\beta_k| + \min\left(|\alpha_{k+1}| + |\gamma_{k-1}|, |\alpha_k| + |\gamma_k|\right)\right) \simeq \sigma^2, \tag{35}$$

where '$\simeq$' means that the left-hand side and the right-hand side are numerically equal. We assume that β_k is so small and negligible provided that (35) holds numerically.

Similarly, we get the numerical criterion for neglecting a second-subdiagonal element γ_k. On the setting of

$$\hat{L} := L - \gamma_k \mathbf{e}_{k+2}\mathbf{e}_k{}^T,$$

the perturbation matrices are given by

$$E_1' := \gamma^2 \mathbf{e}_k \mathbf{e}_k{}^T + \alpha_{k+2}\gamma_k \left(\mathbf{e}_{k+2}\mathbf{e}_k^T + \mathbf{e}_k \mathbf{e}_{k+2}^T\right) + \beta_{k+1}\gamma_k \left(\mathbf{e}_{k+1}\mathbf{e}_k^T + \mathbf{e}_k \mathbf{e}_{k+1}^T\right),$$
$$E_2' := \gamma^2 \mathbf{e}_{k+2}\mathbf{e}_{k+2}{}^T + \alpha_k \gamma_k \left(\mathbf{e}_{k-2}\mathbf{e}_k^T + \mathbf{e}_k \mathbf{e}_{k-2}^T\right) + \beta_k \gamma_k \left(\mathbf{e}_{k-1}\mathbf{e}_k^T + \mathbf{e}_k \mathbf{e}_{k-1}^T\right).$$

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

$$\sigma^2 + |\gamma_k| \left(|\gamma_k| + \min\left(|\alpha_{k+2}| + |\beta_{k+1}|, |\alpha_k| + |\beta_k|\right)\right) \simeq \sigma^2. \tag{36}$$

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

1) If β_{n-1} and γ_{n-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 β_{k-1}, γ_{k-1} and γ_{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 1
Computation time (seconds)

matrix size	10000	20000	30000
oqds for bidiagonal	11.764	43.243	93.080
proposed oqds for lower tridiagonal	27.089	100.013	210.225

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

[1] H. Rutishauser, "Der Quotienten-Differenzen-Algorithms," in *ZAMP*, 5 (1954), pp. 233–251.
[2] U. von Matt, "The orthogonal QD algorithm," *SIAM J. Sci. Comput.*, vol.18, Issue:4, (1997), pp.1163–1186.
[3] U. von Matt, "The orthogonal QD algorithm," *Technical Reports of the Computer Science Department*, University of Maryland, 1994.
[4] H. Rutishauser, "Uber eine kubisch konvergente Variante der LR-Thransformation," *Zeitschrift fur Angewandte Mathematik und Mechanik*, vol. 40 (1960), pp. 49–54.
[5] S. Gerschgorin, "Uber die Abgrenzung der Eigenwerte einer Matrix," *Izv. Akad. Nauk.*, USSR Otd. Fiz.-Mat. Nauk 7,(1931), pp.749–754.
[6] T. Yamashita, Y. Yamamoto and K. Kimura, "A new method for computation of conserved quantities of the discrete finite Toda equation," unpublished.
[7] B. N. Parlett, "Laguerre's Method Applied to the Matrix Eigenvalue Problem," *Math. Comp.*, 18 (1964), pp. 464–485.
[8] F. Chatelin, *Valeurs propres de matrices*, Masson, Paris, 1988, ISBN 2-225-80968-2.
[9] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, 1980.
[10] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1995.
[11] T. Imamura, S. Yamada and M. Machida, "Narrow-band reduction approach of a DRSM eigensolver on a multicore-based cluster system," *Advances in Parallel Computing*, vol.19, *Parallel Computing: From Multicores and GPU's to Petascale*, IOS Press, 2010, pp.91–98.