数理解析研究所講究録
第2005巻 2016年 21-31

An Algorithm for Simultaneous Band Reduction of Two Dense Symmetric Matrices

Lei Du[1),2)] Akira Imakura[1)] Tetsuya Sakurai[1),2)]
1) Faculty of Engineering, Information and Systems, University of Tsukuba, 305–8573, Japan
2)JST CREST, 4-1-8 Hon-cho, Kawaguchi-shi, Saitama 332-0012, Japan

Abstract

In this paper, we propose an algorithm for simultaneously reducing two dense symmetric matrices to band form with the same bandwidth by congruent transformations. The simultaneous band reduction can be considered as an extension of the simultaneous tridiagonalization of two dense symmetric matrices. In contrast to algorithms of simultaneous tridiagonalization that are based on Level-2 BLAS (Basic Linear Algebra Subroutine) operations, our band reduction algorithm is devised to take full advantage of Level-3 BLAS operations for better performance. Numerical results are presented to illustrate the effectiveness of our algorithm.

1 Introduction

Given two real n-by-n dense symmetric matrices A, B, we consider the simultaneous band reduction of A and B via congruent transformations with respect to a matrix $Q \in \mathbb{R}^{n\times n}$ as follows

$$K = Q^T AQ, \quad M = Q^T BQ, \tag{1}$$

where K and M are band matrices with the same odd-numbered bandwidth s.

It is well known that the band reduction of a single dense symmetric matrix, as a pre-processing step, is widely applied to compute the spectral decomposition, which can be also used to solve shifted linear systems for example. Please refer to [3, 17] and references therein for more details.

For a pair of nonsymmetric matrices A and B, algorithms for different condensed forms have also been proposed, for example algorithms for the Hessenberg-triangular form [1, 4, 7, 12].

Considering the symmetry, algorithms for the tridiagonal-diagonal form have been proposed in [2, 11, 16]. Recently, methods for the tridiagonal-tridiagonal form, simultaneous tridiagonalization, have been proposed in [6, 15] by congruent transformations. Compared with other condensed forms, the tridiagonal-tridiagonal can be obtained under a weak condition that the matrix pencil (A, B) is regular. The complexity of simultaneous tridiagonalization is $\mathcal{O}(n^3)$ FLOPs. In addition, the computations are based on the Level-2 BLAS operations.

In this paper, we propose an algorithm for simultaneously reducing two dense symmetric matrices to band form with the same bandwidth by congruent transformations. In contrast to the algorithms of simultaneous tridiagonalization that are based on Level-2 BLAS operations, our band reduction algorithm is devised to take full advantage of

Level-3 BLAS operations and able to achieve better performance. The band reduction can be also used as a pre-processing step for solving problems such as the generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ and the generalized shifted linear systems $(A+\sigma_i B)\boldsymbol{x} = \boldsymbol{b}$ etc. Please refer to [5, 8, 9, 10, 13] for algorithms of solving the band (tridiagonal) generalized eigenvalue problems.

The paper is organized as follows. In the Section 2, we briefly study the existing methods for simultaneous tridiagonalization. In Section 3, we empoly the tridiagonalization ideas and propose an algorithm for the simultaneous band reduction. Implementation details will be discussed. In Section 4, we give some numerical results to illustrate the effectiveness of our algorithm. Finally, we make some concluding remarks and point our future work in Section 5.

Throughout this paper the following notation is used. If the size of a matrix or a vector is apparent from the context without confusion, we will drop the index, e.g., denote an m-by-n matrix $A_{m\times n}$ by A and an n-dimensional vector $\boldsymbol{x}_n$ by $\boldsymbol{x}$. I will always represent the identify matrix, A^T denotes the transpose of A. The Matlab colon notation is used. For example, the entry of A at the ith row and jth column is $A(i,j)$, the kth column of A is $A(:,k)$ and $A(:,i:j) = [A(:,i), A(:,i+1), \ldots, A(:,j)]$ is a sub-matrix with $j-i+1$ columns.

2 A brief review of the simultaneous tridiagonalization

The simultaneous tridiagonalization of two symmetric matrices is firstly discussed by Garvey et al in [6]. Then Sidje [15] gave more details on simultaneous tridiagonalization under a unified framework. In this section, we briefly study their methods.

For simplicity, we let $n = 8$ and assume that two iteration steps of tridiagonalization of A and B have been complete, which gives $A^{(2)}$ and $B^{(2)}$ as follows,

$$A^{(2)} = \left(\begin{array}{cc|cccccc} * & * & & & & & & \\ * & * & * & & & & & \\ \hline & * & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \end{array}\right) \text{ and } B^{(2)} = \left(\begin{array}{cc|cccccc} * & * & & & & & & \\ * & * & * & & & & & \\ \hline & * & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \end{array}\right),$$

where $*$ denotes the nonzero entries.

The third iteration step for $A^{(3)}$ and $B^{(3)}$ can be described by the following two-stage computations.

- Stage 1: if $A^{(2)}(4:8,3) \neq \alpha B^{(2)}(4:8,3)$ for the given scalar α, then construct a matrix L_3 and compute $\bar{A}^{(2)} = L_3^T A^{(2)} L_3$, $\bar{B}^{(2)} = L_3^T B^{(2)} L_3$ to make $\bar{A}^{(2)}(4:8,3) = \alpha \bar{B}^{(2)}(4:8,3)$.

- Stage 2: construct a matrix H_3 and let $A^{(3)} = H_3^T \bar{A}^{(2)} H_3, B^{(3)} = H_3^T \bar{B}^{(2)} H_3$ to eliminate nonzero entires of $\bar{A}^{(2)}(5:8,3)$ and $\bar{B}^{(2)}(5:8,3)$.

Meanwhile, there should be no fill-in for all the zero entries (i,j), when $|i-j|>1$, during both stages. In the following subsections, we briefly describe how to construct L_3 and H_3. Please refer to [6, 15] for more details.

2.1 Stage 1: construct matrix L_3

It is easy to check that L_3 with the following pattern can always avoid fill-in for the computations $\bar{A}^{(2)} = L_3^T A^{(2)} L_3$ and $\bar{B}^{(2)} = L_3^T B^{(2)} L_3$,

$$L_3 = \left(\begin{array}{cc|cccccc} 1 & & & & & & & \\ & 1 & & & & & & \\ \hline & & 1 & & & & & \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \end{array}\right).$$

Theoretically, all nonsingular matrices could be chosen for the sub-block $L_3(4:8,4:8)$. In practice, various rank-one updates have been given in [6, 15], where $L_3(3:8,3:8) = I_6 + [0, \boldsymbol{x}^T]^T[1, \boldsymbol{y}^T]$. In order to make $\bar{A}^{(2)}(4:8,3)$ and $\bar{B}^{(2)}(4:8,3)$ be collinear (the corresponding $\otimes$ entires), i.e., $\bar{A}^{(2)}(4:8,3) = \alpha \bar{B}^{(2)}(4:8,3)$,

$$\bar{A}^{(2)} = \left(\begin{array}{cc|cccccc} * & * & & & & & & \\ * & * & * & & & & & \\ \hline & * & * & \otimes & \otimes & \otimes & \otimes & \otimes \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \end{array}\right) \text{ and } \bar{B}^{(2)} = \left(\begin{array}{cc|cccccc} * & * & & & & & & \\ * & * & * & & & & & \\ \hline & * & * & \otimes & \otimes & \otimes & \otimes & \otimes \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \end{array}\right).$$

the unknown vector $\boldsymbol{x}$ can be efficiently determined as follows

$$[1, \boldsymbol{x}^T]^T = \frac{(A^{(2)}(3:8,3:8) - \alpha B^{(2)}(3:8,3:8))^{-1}\boldsymbol{e}_1}{\boldsymbol{e}_1^T (A^{(2)}(3:8,3:8) - \alpha B^{(2)}(3:8,3:8))^{-1}\boldsymbol{e}_1}. \tag{2}$$

Moreover the solution is also unique. To avoid solving the linear system for $\boldsymbol{x}$ per iteration and reduce the total computation cost, a practical approach is given in [6]. Only the LDL^T decomposition of $A - \alpha B$ or its inverse needs to be computed in $\mathcal{O}(n^3)$ operations. In the follow-up steps, $\boldsymbol{x}$ can be efficiently computed by using the LDL^T or its inverse in $\mathcal{O}(n^2)$ operations.

Although $\boldsymbol{x}$ is determined uniquely, there are different choices for $\boldsymbol{y}$. Here we recall some results in [6, 15],

(i) $\boldsymbol{y} = \mathbf{0}$ corresponding to $L_3(4:8, 4:8) = I$;

(ii) Determine $\boldsymbol{y}$ by letting $\bar{A}^{(2)}(4:8, 3) = \sigma \boldsymbol{e}_1$;

(iii) $\boldsymbol{y} = -(1 + \sqrt{1 + \|\boldsymbol{x}\|_2^2})\boldsymbol{x}/\|\boldsymbol{x}\|_2^2$ by minimizing the condition number of L_3;

(iv) $\boldsymbol{y} = -2\boldsymbol{x}/\|\boldsymbol{x}\|_2^2$;

Remark 2.1. For the case (ii), the computation of stage 2 will not be needed for tridiagonalization.

2.2 Stage 2: construct matrix H_3

When the stage 1 is complete, it is easy to simultaneously eliminate the nonzero entires of $\bar{A}^{(2)}(5:8, 3)$ and $\bar{B}^{(2)}(5:8, 3)$. For example, a Householder transformation H_3 as follows can be applied to both $\bar{A}^{(2)}$ and $\bar{B}^{(2)}$.

$$H_3 = \left(\begin{array}{cc|cc} 1 & & & \\ & 1 & & \\ \hline & & 1 & \\ & & & I - 2\boldsymbol{u}\boldsymbol{u}^T \end{array}\right),$$

where $\boldsymbol{u}$ is an unit vector and is determined by $\bar{A}^{(2)}(4:8, 3)$. Then we can obtain $A^{(3)} = H_3^T \bar{A}^{(2)} H_3$ and $B^{(3)} = H_3^T \bar{B}^{(2)} H_3$ as follows,

$$A^{(3)} = \left(\begin{array}{cc|cccccc} * & * & & & & & & \\ * & * & * & & & & & \\ \hline & * & * & * & & & & \\ & & * & * & * & * & * & * \\ & & & * & * & * & * & * \\ & & & * & * & * & * & * \\ & & & * & * & * & * & * \\ & & & * & * & * & * & * \end{array}\right) \text{ and } B^{(3)} = \left(\begin{array}{cc|cccccc} * & * & & & & & & \\ * & * & * & & & & & \\ \hline & * & * & * & & & & \\ & & * & * & * & * & * & * \\ & & & * & * & * & * & * \\ & & & * & * & * & * & * \\ & & & * & * & * & * & * \\ & & & * & * & * & * & * \end{array}\right).$$

We can continue the two-stage computations recursively until two tridiagonal matrices are obtained. For the general case, we describe the simultaneous tridiagonalization procedure in Algorithm 1.

Remark 2.2. In steps 5, 7 and 8, matrix updates like $M = (I + \boldsymbol{u}\boldsymbol{v}^T)^T M (I + \boldsymbol{u}\boldsymbol{v}^T)$ are needed, where M is a matrix, $\boldsymbol{u}$ and $\boldsymbol{v}$ are vectors. Level-2 BLAS such as DSYR, DSYR2, DSYMV can be used for updating M.

From discussions above, we see the algorithm of simultaneous tridiagonalization has the following disadvantages.

- Complexity of the algorithm is order of n^3 flops ;
- Implementations of the algorithm are mainly based on Level-2 BLAS operations.

Algorithm 1 Pseudocode of simultaneous tridiagonalization

1: Given two n-by-n symmetric matrices A, B, and scalar α; let $Q = I$;
2: Compute the LDL^T of $A - \alpha B$ or inverse $N := (A - \alpha B)^{-1}$;
3: **for** $k = 1 : n - 2$ **do**
4: Compute $\boldsymbol{x}$ and $\boldsymbol{y}$ for L_k; ▷ **Level-2 BLAS**
5: Compute $\bar{A}^{(k-1)}$ and $\bar{B}^{(k-1)}$; ▷ **Level-2 BLAS**
6: Compute $\boldsymbol{u}$ for H_k; ▷ **Level-2 BLAS**
7: Compute $A^{(k)}$ and $B^{(k)}$; ▷ **Level-2 BLAS**
8: Update N and $Q = QL_kH_k$; (if necessary) ▷ **Level-2 BLAS**
9: **end for**

We know that almost all modern computers have the structure of memory hierarchy. Computation time of an algorithm mainly depends on the arithmetic operations (FLOPs) and data movement (memory access). A basic rule for devising fast algorithms is to reduce memory access as more as possible. The ratios of FLOPs to memory access corresponding to different level operations are given in Table 1.

Table 1: Ratios of arithmetic operations to memory access. Denote $\alpha, \beta \in \mathbb{R}$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $A, B, C \in \mathbb{R}^{n \times n}$.

		FLOPs	memory access	Ratio
$\boldsymbol{y} = \alpha\boldsymbol{x} + \boldsymbol{y}$	(Level-1)	$2n$	$3n$	$2/3$
$\boldsymbol{y} = \alpha A\boldsymbol{x} + \beta\boldsymbol{y}$	(Level-2)	$2n^2$	n^2	2
$C = \alpha AB + \beta C$	(Level-3)	$2n^3$	$4n^2$	$n/2$

From Table 1, we see that an algorithm that is able to implemented by higher level BLAS operations maybe show better performance, which inspires us to extend the tridiagonalization algorithm and devise a new algorithm that can take advantage of the Level-3 BLAS operations.

3 An algorithm for simultaneous band reduction

In this section, we extend the ideas in Section 2 and propose an algorithm for simultaneous band reduction. Similar to the simultaneous tridiagonalization, the computations of band reduction will be also divided two stages. We first process t ($t := (s-1)/2$ hereafter) steps like the stage 1 of Algorithm 1 to make t pairs of vectors collinear. Then we continue to do t steps like the stage 2 of Algorithm 1 to eliminate the off-diagonal nonzero entries for all $|i - j| > t$.

As there are different variants for simultaneous tridiagonalization, in what follows, our strategy of band reduction is mainly based on the first variant that $\boldsymbol{y} = 0$. For simplicity, we take $t = 2$ and discuss the two stages of simultaneous band reduction as follows.

Assuming that two steps of band reduction of A and B have been complete. Matrices $A^{(1)}$ and $B^{(1)}$ obtained are as follows,

$$A^{(1)} = \begin{pmatrix} * & * & * & & & & & \\ * & * & * & * & & & & \\ \hline * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \end{pmatrix} \text{ and } B^{(1)} = \begin{pmatrix} * & * & * & & & & & \\ * & * & * & * & & & & \\ \hline * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \\ & & * & * & * & * & * & * \end{pmatrix}.$$

3.1 Stage 1: construct matrix L_2

The aim of this stage is to construct a matrix L_2 which makes t pairs of vectors corresponding to $\bar{A}^{(1)} := L_2^T A^{(1)} L_2$ and $\bar{B}^{(1)} := L_2^T B^{(1)} L_2$ are collinear. When $t = 2$, we can construct L_2 as follows,

$$L_2 = \underbrace{\begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ \hline & & 1 & & & & & \\ & & * & 1 & & & & \\ & & * & & 1 & & & \\ & & * & & & 1 & & \\ & & * & & & & 1 & \\ & & * & & & & & 1 \end{pmatrix}}_{L_2^{(1)}} \times \underbrace{\begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ \hline & & 1 & & & & & \\ & & & 1 & & & & \\ & & & * & 1 & & & \\ & & & * & & 1 & & \\ & & & * & & & 1 & \\ & & & * & & & & 1 \end{pmatrix}}_{L_2^{(2)}}.$$

The unknown entries of $L_2^{(1)}$ and $L_2^{(2)}$ are determined by the similar strategy in section 2.1. We note that the unknown entries of $L_2^{(2)}$ will be determined after that of $L_2^{(1)}$.

Applying $L_2^{(1)}$ and $L_2^{(2)}$ to $A^{(1)}$ successively, we could obtain the following two matrices with the same pattern, respectively. The nonzero entries denoted by $\otimes$ are what we care about.

$$\bar{A}^{(1)} = L_2^{(1)T} A^{(1)} L_2^{(1)} = \begin{pmatrix} * & * & * & & & & & \\ * & * & * & * & & & & \\ \hline * & * & * & \otimes & \otimes & \otimes & \otimes & \otimes \\ & * & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \\ & & \otimes & * & * & * & * & * \end{pmatrix},$$

and

$$\bar{A}^{(1)} = L_2^{(2)T}\bar{A}^{(1)}L_2^{(2)} = \left(\begin{array}{cc|cccccc} * & * & * & & & & & \\ * & * & * & * & & & & \\ \hline * & * & * & \otimes & \otimes & \otimes & \otimes & \otimes \\ & * & \otimes & * & \otimes & \otimes & \otimes & \otimes \\ & & \otimes & \otimes & * & * & * & * \\ & & \otimes & \otimes & * & * & * & * \\ & & \otimes & \otimes & * & * & * & * \\ & & \otimes & \otimes & * & * & * & * \end{array}\right).$$

Similar to $\bar{A}^{(1)}$, we can also obtain $\bar{B}^{(1)}$. Congruent transformations by $L_2^{(1)}$ make $\bar{A}^{(1)}(4:8,3)$ and $\bar{B}^{(1)}(4:8,3)$ be collinear. Congruent transformations by $L_2^{(2)}$ make $\bar{A}^{(1)}(5:8,4)$ and $\bar{B}^{(1)}(5:8,4)$ be collinear and keep the collinearity of $\bar{A}^{(1)}(4:8,3)$ and $\bar{B}^{(1)}(4:8,3)$.

3.2 Stage 2: construct matrix H_2

In this stage, we eliminate nonzero entries of $\bar{A}^{(1)}$ and $\bar{B}^{(1)}$ in columns 3 and 4 for the band form. We construct the matrix H_2 as follows, a product of two Householder matrices $H_2^{(1)}$ and $H_2^{(2)}$,

$$H_2 = \underbrace{\left(\begin{array}{cc|ccc} 1 & & & & \\ & 1 & & & \\ \hline & & 1 & & \\ & & & 1 & \\ & & & & I - 2\boldsymbol{u}_4\boldsymbol{u}_4^T \end{array}\right)}_{H_2^{(1)}} \times \underbrace{\left(\begin{array}{cc|ccc} 1 & & & & \\ & 1 & & & \\ \hline & & 1 & & \\ & & & 1 & \\ & & & & I - 2\boldsymbol{u}_3\boldsymbol{u}_3^T \end{array}\right)}_{H_2^{(2)}},$$

where $\boldsymbol{u}_4$ and $\boldsymbol{u}_3$ can be determined by $\bar{A}^{(1)}(5:8,3:4)$. Then we can obtain $A^{(2)} = H_2^T\bar{A}^{(1)}H_2$ and $B^{(2)} = H_2^T\bar{B}^{(1)}H_2$.

As the product of $H_2^{(1)}$ and $H_2^{(2)}$ can be represented as follows by the compact WY representation [14],

$$H_2 = \left(\begin{array}{cc|ccc} 1 & & & & \\ & 1 & & & \\ \hline & & 1 & & \\ & & & 1 & \\ & & & & I - VTV^T \end{array}\right),$$

where V is a 4×2 matrix, and T denotes a 2×2 upper triangular matrix. In practice, sub-matrices $A^{(2)}(5:8,5:8)$ and $B^{(2)}(5:8,5:8)$ can be effectively computed by this representation, . Level-3 BLAS such as DSYR2K, DSYMM can be employed.

We summarize discussion above and give the pseudocode of band reduction in Algorithm 2.

Remark 3.1. If $s = 3$, Algorithm 2 is consistent with the Algorithm 1 corresponding to variant (i). We note that the strategy discussed in this section can be also applied to variants (iii) and (iv) of tridiagonalization.

Algorithm 2 Pseudocode of band reduction

1: Given symmetric matrices A, B, $Q = I$ and scalar α;
2: Given bandwidth s, let $t = (s-1)/2$;
3: Compute the LDL^T of $A - \alpha B$ or inverse $N := (A - \alpha B)^{-1}$; ▷ **Inverse**
4: **for** $k = 1 : \lfloor \frac{n-t}{t} \rfloor$ **do**
5: **for** $i = (k-1)t + 1 : kt$ **do**
6: Compute $\boldsymbol{x}_{n-i}$ for $L_k^{(i-(k-1)t)}$; ▷ **Level-2 BLAS**
7: Compute $\bar{A}^{(k)}(:, i)$ and $\bar{B}^{(k)}(:, i)$; ▷ **Level-2 BLAS**
8: Update N and $Q = QL_k^{(i-(k-1)t)}$; (if necessary) ▷ **Level-2 BLAS**
9: **end for**
10: Compute the QR decomposition of $\bar{A}(kt + 1 : n, (k-1)t + 1 : kt)$ and compact WY-representation of H_k; ▷ **Level-2 BLAS**
11: Compute $A^{(k)} = H_k^T A^{(k-1)} H_k$; ▷ **Level-3 BLAS**
12: Compute $B^{(k)} = H_k^T B^{(k-1)} H_k$; ▷ **Level-3 BLAS**
13: Update N and $Q = QH_k$; (if necessary) ▷ **Level-3 BLAS**
14: **end for**

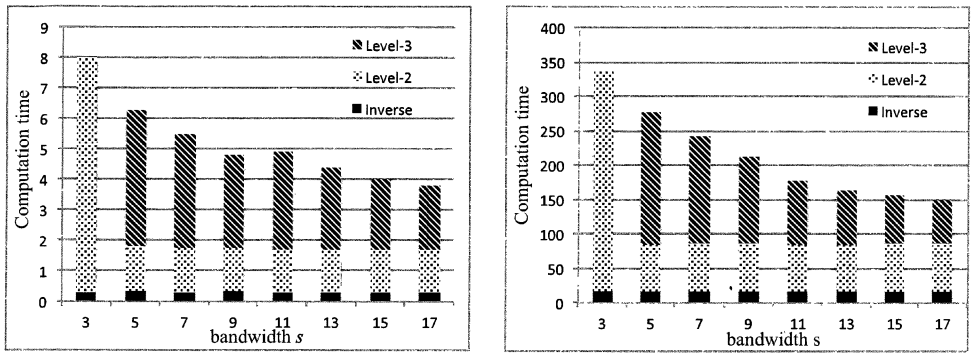
4 Numerical experiments

In this section, we give some numerical results to show the performance of the proposed algorithm. We also apply the algorithm for solving generalized shifted linear systems. Test matrices were initialized by random values and two different sizes ($n = 1000, 3000$). The computer specifications are Red Hat Linux, AMD Opteron(tm) processor, 2.5GHz (1 core) with 32GB of RAM. The algorithm was implemented in the Fortran 90 language and compiled with ifort (ver. 13.1.1) using the Intel MKL for LAPACK and BLAS. In the implementation, we did not compute the Q explicitly, but the inverse of $A - \alpha B$.

In Figure 1, we compare the computation time of band reduction corresponding to Algorithm 2 with different bandwidth. We see that the total computation time is reduced with greater bandwidth. Compared with $s = 3$, the algorithm took less than half time when $s = 17$. In Figure 2, symmetric band linear systems $K\boldsymbol{x} = \boldsymbol{b}$ were solved by calling the Lapack function **DGBSV**. The computation time almost increased linearly with the increasing of bandwidth. We also computed the solution of generalized shifted linear systems $(A + \sigma_i B)\boldsymbol{x} = \boldsymbol{b}$ for $i = 1, 2, \ldots, L$ by using band reduction and **DGBSV**. The total computation time is denoted by $T_{total} := T_{band\,reduction} + L \cdot T_{DGBSV}$. In Figure 3, we show the average computation time for one linear system, i.e., T_{total}/L. Compared to **DSYSV**, band reduction shows its advantages when the bandwidth s and number of shifted linear systems L are greater.

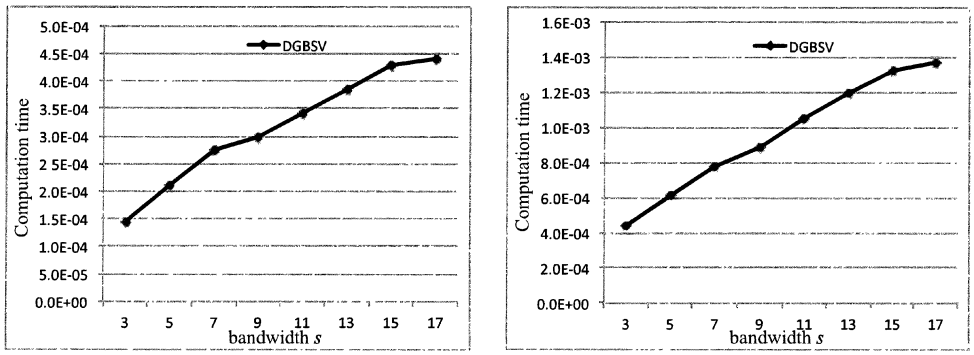
5 Conclusions and future work

In this paper, we proposed an algorithm for simultaneous band reduction of two dense symmetric matrices. Although our discussions are based on real-valued matrices, the algorithm can be easily extended to complex-valued Hermitian matrices. We also gave some numerical results to show the performance of band reduction with different band-



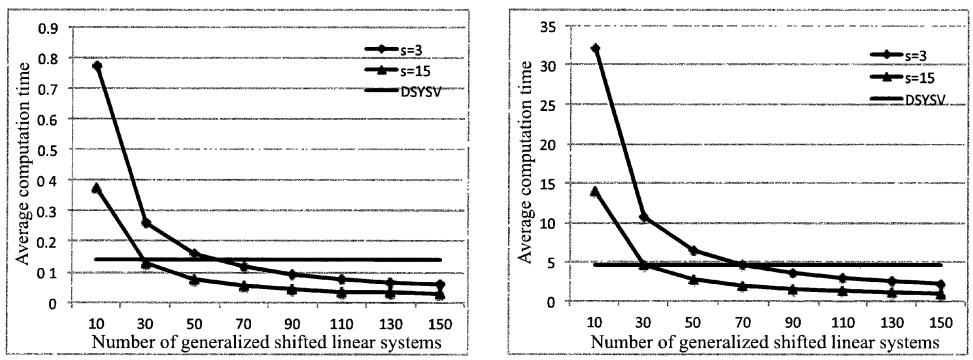
(a) $n = 1000$. (b) $n = 3000$.

Figure 1: Computation time of band reduction (in seconds) versus matrix bandwidth.



(a) $n = 1000$. (b) $n = 3000$.

Figure 2: Computation time of band linear systems (in seconds) versus matrix bandwidth.



(a) $n = 1000$. (b) $n = 3000$.

Figure 3: Average computation time per linear system (in seconds) versus the number of generalized shifted linear systems.

width and an application of solving generalized shifted linear systems.

The topics, such as

- other algorithms for band reduction,
- simultaneously reduce band form to tridiagonal form,
- accelerate the computing using GPU etc.,

will be considered as our future work.

Acknowledgments

This research was partially supported by Strategic Programs for Innovative Research (SPIRE) Field 5 "The origin of matter and the universe", JST/CREST project "Development of an Eigen-Supercomputing Engine using a Post-Petascale Hierarchical Model" and KAKENHI (Nos. 25870099, 25104701, 25286097).

References

[1] B. Adlerborn, K. Dackland and B. Kågström, Parallel and Blocked Algorithms for Reduction of a Regular Matrix Pair to Hessenberg-Triangular and Generalized Schur Forms, in Applied Parallel Computing, Lecture Notes in Comput. Sci. 2367, Springer, Berlin, Heidelberg, 757–767(2006).

[2] M.A. Brebner, and J. Grad, Eigenvalues of $Ax = \lambda Bx$ for Real Symmetric Matrices A and B Computed by Reduction to a Pseudosymmetric Form and the HR process, Linear Algebra Appl., **43**, 99–118(1982).

[3] C.H. Bischof, B. Lang and X.B. Sun, A framework for symmetric band reduction, ACM Trans. Math. Software, **26**(4), 581–601(2000).

[4] K. Dackland and B. Kågström, Block algorithms and software for reduction of a regular matrix pair to generalized Schur form, ACM Trans. Math. Software, **25**(4), 425–454(1999).

[5] L. Elsner, A. Fasse and E. Langmann, A divide-and-conquer method for the tridiagonal generalized eigenvalue problem, J. Comput. Appl. Math., **86**(1), 141–148(1997).

[6] S.D. Garvey, F. Tisseur, M.I. Friswell, J.E.T. Penny and U. Prells, Simultaneous tridiagonalization of two symmetric matrices, Int. J. Numer. Meth. Eng., **57**(12), 1643–1660(2003).

[7] B. Kågström, D. Kressner, E.S. Quintana-ortí and G. Quintana-ortí, Block algorithms for the reduction to Hessenberg-triangular form revisited, BIT Numer. Math., **48**(3), 563–584(2008).

[8] L. Kaufman, An Algorithm for the Banded Symmetric Generalized Matrix Eigenvalue Problem, SIAM J. Matrix Anal. Appl., **14**(2), 372–389(1993).

[9] K. Li, T.Y. Li and Z. Zeng, An algorithm for the generalized symmetric tridiagonal eigenvalue problem, Numer. Algorithms, **8**(2), 269–291(1994).

[10] K. Li, Durand-Kerner root-finding method for the generalized tridiagonal eigenproblem, Missouri J. Math. Sci., 33–43(1999).

[11] R.S. Martin and J.H. Wilkinson, Reduction of the symmetric eigenproblem $Ax = \lambda Bx$ and related problems to standard form, Numer. Math., **11**(2), 99–110(1968).

[12] C.B. Moler and G.W. Stewart, An algorithm for generalized matrix eigenvalue problems, SIAM J. Numer. Anal., **10**(2), 241–256(1973).

[13] G. Peters and J.H. Wilkinson, Eigenvalues of $Ax = \lambda Bx$ with band symmetric A and B, Comput. J., **12**(4), 398–404(1969).

[14] R. Schreiber and C. van Loan, A storage-efficient WY representation for products of householder transformations, SIAM J. Sci. Stat. Comput., **10**(1), 52–57(1989).

[15] R.B. Sidje, On the simultaneous tridiagonalization of two symmetric matrices, Numer. Math., **118**(3), 549–566(2011).

[16] F. Tisseur, Tridiagonal-diagonal reduction of symmetric indefinite pairs, SIAM J. Matrix Anal. Appl., **26**(1), 215–232(2004).

[17] F.G. van Zee, R.A. van de Geijn, G. Quintana-ortí and G.J. Elizondo, Families of algorithms for reducing a matrix to condensed form, ACM Trans. Math. Software, **39**(1), 2:1–2:32(2012).