mathematics

MDPI

Article

Efficient Reduction Algorithms for Banded Symmetric Generalized Eigenproblems via Sequentially Semiseparable (SSS) Matrices

Fan Yuan [1,†], Shengguo Li [1,†], Hao Jiang [1], Hongxia Wang [2], Cheng Chen [3], Lei Du [4] and Bo Yang [1,*]

1 College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China; yuanfan.nudt@gmail.com (F.Y.); nudtlsg@nudt.edu.cn (S.L.); haojiang@nudt.edu.cn (H.J.)
2 College of Liberal Arts and Sciences, National University of Defense Technology, Changsha 410073, China; hxwang@nudt.edu.cn
3 Computational Aerodynamics Institute, China Aerodynamics Research and Development Center, Mianyang 621000, China; chencheng@nudt.edu.cn
4 School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China; dulei@dlut.edu.cn
* Correspondence: yangbo78@nudt.edu.cn
† These authors contributed equally to this work.

Abstract: In this paper, a novel algorithm is proposed for reducing a banded symmetric generalized eigenvalue problem to a banded symmetric standard eigenvalue problem, based on the sequentially semiseparable (SSS) matrix techniques. It is the first time that the SSS matrix techniques are used in such eigenvalue problems. The newly proposed algorithm only requires linear storage cost and $O(n^2)$ computation cost for matrices with dimension n, and is also potentially good for parallelism. Some experiments have been performed by using Matlab, and the accuracy and stability of algorithm are verified.

Keywords: generalized eigenvalue problems; rank-structured matrix; SSS matrix; banded reduction

MSC: 6505; 65F30; 65R20; 68W40; 68P05

Citation: Yuan, F.; Li, S.; Wang, H.; Jiang, H.; Wang, H.; Chen, C.; Du, L.; and Yang, B. Efficient Reduction Algorithms for Banded Symmetric Generalized Eigenproblems via Sequentially Semiseparable (SSS) Matrices. *Mathematics* **2022**, *10*, 1676. https://doi.org/10.3390/math10101676

Academic Editor: Juan Ramón Torregrosa Sánchez

Received: 30 March 2022
Accepted: 9 May 2022
Published: 13 May 2022

1. Introduction

In this paper, we consider how to reduce the following generalized eigenvalue problem (GEP) to a standard eigenvalue problem,

$$AQ = BQ\Lambda, \tag{1}$$

with banded Hermitian matrices $A, B \in \mathbb{C}^{n\times n}$ and B positive definite. In the real case, A and B would be symmetric instead of Hermitian. The classical way is first to compute the Cholesky factorization of $B = LL^H$ with a lower triangular matrix L, and then multiply (1) with L^{-1}, and it yields a standard eigenvalue problem,

$$L^{-1}AL^{-H}\cdot L^HQ = L^HQ\Lambda. \tag{2}$$

The problem with this approach is that $C := L^{-1}AL^{-H}$ is dense though A and L are banded, since L^{-1} is fully triangular in general. In this paper, we consider how to reduce matrix C to a symmetric banded form efficiently.

The LAPACK library [1] includes some routines for reducing the banded GEP to banded SEP, named `XHBGST` and `XSBGST`, which `X` denotes different precision, `S` (single precision), `D` (double precision), `C` (single complex precision) and `Z` (double complex precision). We use the real double case to introduce the main process. First, it (`DPBSTF`) computes a split Cholesky factorization [2] of the real symmetric banded positive definite matrix B, $B = S^TS$, where the leading $p \times p$ submatrix of S an upper banded matrix with bandwidth b_B and

the lowers $n - p$ rows form a lower banded matrix with bandwidth b_B. This factorization is also called the 'twisted factorization' [3]. Then, the routine `PSBGST` updates $A = X^T AX$, where $X = S^{-1}Q$ and Q is an orthogonal matrix chosen to preserve the bandwidth of A. The matrix S is treated as a product of elementary matrices:

$$S = S_m S_{m-1} \dots S_2 S_1 S_{m+1} \dots S_{n-1} S_n,$$

where S_i is determined by the i-th row of S. For each value of i, the current matrix A is updated by forming $S_i^{-1} A S_i^{-T}$, and the introduced bulge is chased out by applying plane rotations.

Some approaches have been proposed to reduce the generalized eigenvalue problem to the tridiagonal-diagonal form [4,5], which directly reduce B to the diaognal form and A to the tridiagonal form, respectively. In 1973, Crawford [6] proposed a new reduction method for the real symmetric case with $b_A = b_B$. Different from LAPACK, the method is based on a decomposition of the matrices into $b_B \times b_B$ blocks and the bulge is removed immediately by using matrix-matrix multiplications. Lang [7] combined the good features of Crawford's scheme with LAPACK routines, which proceeds by blocks and can also handle different bandwidths $b_B < b_A$. A distributed parallel version [8] is included in ELPA [9].

In this work we present a new reduction algorithm that is completely different from previous algorithm, and the tool that we use is the sequentially semiseparable matrix (SSS) techniques. The SSS matrix was introduced in [10,11], which is a kind of rank-structured matrices, see the next section. Since a symmetric banded matrix can be seen as a special block tridiagonal matrix, L is a block bidiagonal matrix and it is well-known that the inverse of L is a lower triangular SSS matrix, see the next section and [12]. The matrix $C = L^{-1} A L^{-T}$ can be proved to be an SSS matrix, see the next section. Different from Crawford's method [6] and LAPACK [1], we compute C explicitly, but express it in SSS form. The computation and storage do not increase much. Like Crawford's method, the advantage of our approach is that the matrices are partitioned into blocks, almost all operations are (small) matrix-matrix multiplications, and some of these small matrix-matrix multiplications can be computed in parallel by using dynamic modeling. For the task-based implementation of matrix operations, we can leverage the `CHAMELEON` library [13,14] to implement a parallel version of our algorithm, which is one potential advantage of our algorithm. This will be our future work.

In this paper we reduce the original problem to the block tridiagonalization problem of an SSS matrix. Some fast algorithms for tridiagonalizing a diagonal plus semiseparable matrix was introduced in [15,16] which costs $O(n^2)$ flops. We generalize the tridiagonalization approach in [15] to the (block) SSS matrix case, and show how to further get its banded form, and the complexity is $O(nr^2)$ flops, where n is the dimension of matrix, and $r = b_A = b_B$. The disadvantage of our algorithm proposed in this work is that it requires the semi-bandwidths of A and B to be equal, $b_A = b_B$. The procedure is shown in Algorithm 1, and it works on the SSS generators of matrix $C = L^{-1} A L^{-H}$ and the outputs are also some small matrices. The memory and computation costs are in the same order as the algorithms in LAPACK, and our algorithm is easy to be implemented in parallel.

The following sections of this paper are organized as follows. Section 2 gives a brief introduction to semiseparable and SSS matrices, and fast matrix multiplication of two SSS matrices is also included. Section 3 describes how to express matrix C into an SSS matrix and how to recompress its generators. The banded reduction process for symmetric SSS matrix is shown in Section 4 and the complexity analysis is included. All the performance results are summarized in Section 5. Conclusions are drawn in Section 6.

2. Semiseparable and SSS Matrices

Rank structured matrices have attracted much attention in recent years. In [17], Raf Vandevril, Marc Van Barel, and Nicola, Mastronardi present a comprehensive overview of the mathematical and numerical properites of one class of these matrices: semiseparable ma-

trices, which is the simplest case. The rank-structured matrices include $\mathscr{H}$-matrices [18–21], $\mathscr{H}^2$-matrices [22,23], quasiseparable matrices [24,25], semiseparable matrices [17,26], sequentially semiseparable matrices [11], hierarchically semiseparable matrices [27–29], etc. Current machine learning and big data analysis are research hotspots [30], and rank-structured matrix techniques can also be used in these areas [31–34].

The semiseparable structure is a matrix analog of the semiseparable integral kernels as described by Kailath in [35]. The semiseparable matrix has been referred as the inverse of unreducible tridiagonal matrix, and Green matrix, one-pair matrix, and single-pair matrix, see [36–38]. Semiseparable matrices appear in several types of applications, e.g., the field of integral equations, boundary value problems, Gauss-Markov process, time-varying linear systems, statistics, acoustic and electromagnetic scattering theory, rational interpolation, and so on.

The sequentially semiseparable matrices (SSS) matrices exploit the off-diagonal low-rank property: the off-diagonal blocks are represented as product of a sequence low-rank matrices. For an $n \times n$ matrix A with block partitioning

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & & \vdots \\ A_{N1} & A_{N2} & \cdots & A_{NN} \end{pmatrix}, \tag{3}$$

where $A_{ij} \in \mathbb{R}^{m_i \times m_j}$ and $n = m_1 + \cdots + m_N$, it can be represented by

$$A_{ij} = \begin{cases} A_{ii} & \text{if } i = j, \\ U_i W_{i+1} \cdots W_{j-1} V_j^T & \text{if } j > i, \\ P_i R_{i-1} \cdots R_{j+1} Q_j^T & \text{if } j < i. \end{cases} \tag{4}$$

For a symmetric matrix A, $P_k = V_k$, $R_k = W_k^T$ and $Q_k = U_k$ for each k. The dimensiosn of these generator matrices $\{U_i\}_{i=1}^{N-1}$, $\{V_i\}_{i=2}^{N}$, $\{W_i\}_{i=2}^{N-1}$, $\{P_i\}_{i=2}^{N}$, $\{Q_i\}_{i=1}^{N-1}$, $\{R_i\}_{i=2}^{N-1}$ and $\{D_i\}_{i=1}^{N}$ are shown in Table 1. The empty products are defined to be the identity matrix. For $N = 4$, the matrix A has the following form,

$$A = \begin{bmatrix} D_1 & U_1 V_2^T & U_1 W_2 V_3^T & U_1 W_2 W_3 V_4^T \\ P_2 Q_1^T & D_2 & U_2 V_3^T & U_2 W_3 V_4^T \\ P_3 R_2 Q_1^T & P_3 Q_2^T & D_3 & U_3 V_4^T \\ P_4 R_3 R_2 Q_1^T & P_4 R_3 Q_2^T & P_4 Q_3^T & D_4 \end{bmatrix}. \tag{5}$$

Table 1. Dimensions of the generators of the SSS matrix shown in Equation (4), k_i and l_i are column dimensions of U_i and P_i, respectively.

Matrix	U_i	V_i	W_i	P_i	Q_i	R_i
Dimension	$m_i \times k_i$	$m_i \times k_{i-1}$	$k_{i-1} \times k_i$	$m_i \times l_i$	$m_i \times l_{i+1}$	$l_{i+1} \times l_i$

Fast Matrix-Matrix Multiplication

A fast algorithm for multiplying an SSS matrix (4) with any given vector or matrix has been presented in [10,39]. This subsection only introduces the case that both A and B are SSS matrices [10]. Let A and B be matrices in SSS form that are conformally partitioned. The forward and backward recursions are defined as

$$\begin{aligned} G_1 &= 0, \quad G_{i+1} = Q_i^T(A) U_i(B) + R_i(A) G_i W_i(B), \quad i = 1, \ldots, n-1, \\ H_n &= 0, \quad H_{i-1} = V_i^T(A) P_i(B) + W_i(A) H_i R_i(B), \quad i = n, \ldots, 2. \end{aligned}$$

We have the following theorem.

Theorem 1 (see [40]). *The SSS form of matrix $C = AB$ can be computed through the following recursions:*

$$\begin{aligned}
D_i(C) &= D_i(A)D_i(B) + P_i(A)G_iV_i^T(B) + U_i(A)H_iQ_i^T(B),\\
P_i(C) &= \begin{bmatrix} D_i(A)P_i(B) + U_i(A)H_iR_i(B) & P_i(A)\end{bmatrix}\\
R_i(C) &= \begin{bmatrix} R_i(B)\\ Q_i^T(A)P_i(B) & R_i(A)\end{bmatrix}\\
Q_i(C) &= \begin{bmatrix} Q_i(B) & D_i^T(B)Q_i(A) + V_i(B)G_i^TR_i^T(A)\end{bmatrix}\\
U_i(C) &= \begin{bmatrix} D_i(A)U_i(B) + P_i(A)G_iW_i(B) & U_i(A)\end{bmatrix}\\
W_i(C) &= \begin{bmatrix} W_i(B)\\ V_i^T(A)U_i(B) & W_i(A)\end{bmatrix}\\
V_i(C) &= \begin{bmatrix} V_i(B) & D_i^T(B)V_i(A) + Q_i(B)H_i^TW_i^T(A)\end{bmatrix}.
\end{aligned} \tag{6}$$

This algorithm is an order of magnitude faster than the general matrix-matrix multiplication algorithms. Notice that after multiplication the ranks of generators will increase. Dewilde and van der veen [39] present a technique to compress the generators. A simple, efficient and numerically stable method is further proposed in [11] to compress a given SSS representation to a predefined tolerance τ, and this method is further introduced in Section 3.2 for completeness.

3. The Reduction Algorithm

Assume the bandwidth of matrix B is b_B, $N = \lceil n/b_B \rceil$, and L can be seen as a lower block bidiagonal matrix and each block is a $b_B \times b_B$ small matrix. Without loss of any generality, we assume $n = N \cdot b_B$. From Gaussian elimination, we know

$$L = L_1 \cdots L_N, \tag{7}$$

where L_i is an identity matrix except for the $b_B \times b_B$ diagonal block L_{ii} and the $b_B \times b_B$ subdiagonal block $L_{i,i-1}$ or $L_{i+1,i}$ from L. There are two form of L: the 'row-wise' and 'column-wise' forms. If $N = 3$, the row-wise form, i.e., $L_{i,i-1}$ is nonzero, is written as

$$L = \begin{bmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ & L_{32} & L_{33}\end{bmatrix} = \begin{bmatrix} L_{11} & & \\ & I & \\ & & I\end{bmatrix}\begin{bmatrix} I & & \\ L_{21} & L_{22} & \\ & & I\end{bmatrix}\begin{bmatrix} I & & \\ & I & \\ & L_{32} & L_{33}\end{bmatrix} \equiv L_1L_2L_3.$$

Then, its inverse yields

$$\begin{aligned}
L^{-1} &= \begin{bmatrix} I & & \\ & I & \\ & -L_{33}^{-1}L_{32} & L_{33}^{-1}\end{bmatrix}\begin{bmatrix} I & & \\ -L_{22}^{-1}L_{21} & L_{22}^{-1} & \\ & & I\end{bmatrix}\begin{bmatrix} L_{11}^{-1} & & \\ & I & \\ & & I\end{bmatrix}\\
&\equiv \begin{bmatrix} \widetilde{L}_{11} & & \\ \widetilde{L}_{21}\widetilde{L}_{11} & \widetilde{L}_{22} & \\ \widetilde{L}_{32}\widetilde{L}_{21}\widetilde{L}_{11} & \widetilde{L}_{32}\widetilde{L}_{22} & \widetilde{L}_{33}\end{bmatrix}, \text{ where } \widetilde{L}_{i,i-1} = -L_{ii}^{-1}L_{i,i-1}.
\end{aligned}$$

If $N = 3$ and the column-wise form is written as

$$L = \begin{bmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ & L_{32} & L_{33}\end{bmatrix} = \begin{bmatrix} L_{11} & & \\ L_{21} & I & \\ & & I\end{bmatrix}\begin{bmatrix} I & & \\ & L_{22} & \\ & L_{32} & I\end{bmatrix}\begin{bmatrix} I & & \\ & I & \\ & & L_{33}\end{bmatrix} \equiv L_1L_2L_3.$$

Then, the inverse yields

$$L^{-1} = \begin{bmatrix} I & & \\ & I & \\ & & L_{33}^{-1} \end{bmatrix} \begin{bmatrix} I & & \\ & L_{22}^{-1} & \\ & -L_{32}L_{22}^{-1} & I \end{bmatrix} \begin{bmatrix} L_{11}^{-1} & & \\ -L_{21}L_{11}^{-1} & I & \\ & & I \end{bmatrix}$$
$$\equiv \begin{bmatrix} \widetilde{L}_{11} & & \\ \widetilde{L}_{22}\widetilde{L}_{21} & \widetilde{L}_{22} & \\ \widetilde{L}_{33}\widetilde{L}_{32}\widetilde{L}_{21} & \widetilde{L}_{33}\widetilde{L}_{32} & \widetilde{L}_{33} \end{bmatrix}, \text{ where } \widetilde{L}_{i,i-1} = -L_{i,i-1}L_{ii}^{-1}.$$

It can be seen that L^{-1} is exactly the sequentially semiseparable (SSS) matrix defined in [11]. The exact formulae for the entries of L^{-1} are given as, see also [12],

$$\left(L^{-1}\right)_{ij} = (-1)^{i+j}\left[\prod_{k=i}^{j+1}\left(L_{kk}^{-1}L_{k+1,k}\right)\right]L_{jj}^{-1},$$

for $i = 2, \cdots, N$ and $j = 1, \cdots, i-1$. It is easy to see that the off-diagonal blocks of L^{-1} are low-rank, and its rank is b_B, see [11,17]. The SSS generators of L^{-1} are

$$D_i(L^{-1}) = L_{ii}^{-1}, P_i = -D_i(L^{-1})L_{i,i-1}, R_i(L^{-1}) = P_{i-1}(L^{-1}), Q_i(L^{-1}) = D_i(L^{-1}).$$

The complexity of computing these generators is $N\cdot(r^3 + \frac{2}{3}r^3) = O(\frac{5}{3}nr^2)$, where L_{ii} and $L_{i,i-1}$ are upper and lower triangular of dimension $r = b_B$, respectively. To represent L^{-1} in SSS form, we only need $\{D_i\}$ and $\{P_i\}$, and their numbers are N and $N-1$, respectively. For more complex operations, we introduce the other two generators $\{R_i\}$ and $\{Q_i\}$.

3.1. The SSS Representation of C

In this subsection, we show that matrix $C = L^{-1}AL^{-H}$ is also an SSS matrix. It is well-known that L^{-1} is an SSS matrix. If A is a symmetric block tridiagonal matrix, it is also an SSS matrix, see Equation (5), with generators,

$$D_i(A) = A_{ii}, P_i(A) = V_i(A) = A_{i+1,i}, R_i(A) = W_i(i)^T = 0, Q_i(A) = U_i(A) = I.$$

According to Theorem 1, if we multiply L^{-1} with A, the forward and backward recursions, $\{G_i\}$ and $\{H_i\}$, are

$$\begin{aligned} &G_1 = 0, \quad G_{i+1} = Q_i^T(L^{-1})U_i(A) = Q_i^T(L^{-1}), && i = 1, \ldots, N-1, \\ &H_n = 0, \quad H_{i-1} = V_i^T(L^{-1})P_i(A) = V_i^T(L^{-1})A_{i+1,i} = 0, && i = N, \ldots, 2. \end{aligned}$$

It is because $R_i(A) = W_i(A) = 0$, and $V_i(L^{-1}) = 0$ (L is lower triangular). Then, the generators of $\bar{C} = L^{-1}A$ are

$$\begin{aligned} D_i(\bar{C}) &= D_i(L^{-1})D_i(A) + P_i(L^{-1})G_iV_i^T(A), \\ P_i(\bar{C}) &= \begin{bmatrix} D_i(L^{-1})P_i(A) & P_i(L^{-1}) \end{bmatrix} \\ R_i(\bar{C}) &= \begin{bmatrix} 0 & \\ Q_i^T(L^{-1})P_i(A) & R_i(L^{-1}) \end{bmatrix} \\ Q_i(\bar{C}) &= \begin{bmatrix} Q_i(A) & D_i^T(A)Q_i(L^{-1}) + V_i(A)G_i^TR_i^T(L^{-1}) \end{bmatrix} \\ U_i(\bar{C}) &= \begin{bmatrix} D_i(L^{-1}) & 0 \end{bmatrix} \equiv D_i(L^{-1}) \\ W_i(\bar{C}) &= \begin{bmatrix} 0 & \\ & 0 \end{bmatrix} \equiv 0 \\ V_i(\bar{C}) &= \begin{bmatrix} V_i(A) & 0 \end{bmatrix} \equiv V_i(A). \end{aligned} \tag{8}$$

The ranks of $P_i(\bar{C}), R_i(\bar{C})$ and $Q_i(\bar{C})$ are all b_B. We can use the recompression techniques [10] to compress them into compact form, which is also introduced in Section 3.2. After obtaining the compact form of $\bar{C}$, we can further compute $C = \bar{C} \cdot L^{-H}$ and express it in SSS form. The forward and backward recursions are computed as

$$\begin{aligned} &\bar{G}_1 = 0, \quad \bar{G}_{i+1} = Q_i^T(\bar{C})U_i(L^{-H}) + R_i(\bar{C})\bar{G}_i W_i(L^{-H}), \qquad i = 1, \ldots, N-1, \\ &\bar{H}_n = 0, \quad \bar{H}_{i-1} = V_i^T(\bar{C})P_i(L^{-H}) + W_i(\bar{C})\bar{H}_i R_i(L^{-H}) \equiv 0, \quad i = N, \ldots, 2. \end{aligned}$$

Since the generators of L^{-T} are the same as those of L^{-1}, i.e., $D_i(L^{-T}) = D_i(L^{-1})^T$, $V_i(L^{-T}) = P_i(L^{-1}), W_i(L^{-T}) = R_i(L^{-1})^T, U_i(L^{-T}) = Q_i(L^{-1})$, $\bar{G}_i$ can be computed as

$$\bar{G}_1 = 0, \bar{G}_{i+1} = Q_i^T(\bar{C})Q_i(L^{-1}) + R_i(\bar{C})\bar{G}_i R_i(L^{-1})^T, \quad i = 1, \ldots, N-1,$$

and the generators of C are computed as

$$\begin{aligned} D_i(C) &= D_i(\bar{C})D_i^T(L^{-1}) + P_i(\bar{C})\bar{G}_i P_i^T(L^{-1}), \\ P_i(C) &= P_i(\bar{C}), \\ R_i(C) &= R_i(\bar{C}), \\ Q_i(C) &= D_i(L^{-1})Q_i(\bar{C}) + P_i(L^{-1})\bar{G}_i^T R_i^T(\bar{C}). \end{aligned} \tag{9}$$

All the generators of C are computed and their ranks are also b_B. Since C is symmetric, only the generators of the diagonals and lower triangular part are needed. The generators of C are already in compact form. To summarize, we have the following proposition.

Proposition 1. *Assume that A and B are Hermitian matrices with bandwidth $b_A = b_B$, and B is further positive definite, and its Cholesky factorization is $B = LL^H$. Then, the matrix $C = L^{-1}AL^{-H}$ is an SSS matrix and the ranks of its off-diagonal generators are all b_B.*

It is easy to see that the complexity of computing the generators of $\bar{C}$ is $O(\frac{31}{3}nr^2)$ floating point operations (flops), and similarly, the complexity of computing the generators of C from $\bar{C}$ is another $O(10nr^2)$ flops.

3.2. Recompression of C

From Equation (8), we know that the SSS representation of $\bar{C}$ is not compact. We can use the techniques proposed in Section 3.7 of [11] and Section 10.6 of [10], to compress them into compact forms. Furthermore, since $C = L^{-1}AL^{-T}$ is symmetric, we only need to compress the generators of the lower triangular part, $\{P_i\}$, $\{R_i\}$ and $\{Q_i\}$. For a symmetric SSS matrix, its generators satisfy $V_i = P_i, U_i = Q_i$ and $W_i = R_i^T$. Therefore, we only need to consider the lower triangular part of $\bar{C}$.

For completeness, we recall the recompression process in [11]. To be consistent with the context of this paper, we introduce the process by using the generators of the lower triangular part. The recompression method is split into two stages in [11]. In this paper, we reverse the stages. In the first stage, the representation is converted into the right proper form; that is, now all the row bases G_i of the Hankel-blocks (The term Hankel-block is taken from [39]. In this paper it denotes the off-diagonal blocks that extend from the diagonal to the southwest corner), where

$$\begin{aligned} G_N &= P_N, \\ G_i &= \begin{pmatrix} P_i \\ G_{i+1}R_i \end{pmatrix}, \quad \text{for } i = N-1, \cdots, 2. \end{aligned}$$

will have orthonormal columns. In the second stage, the representation is converted into the left proper form. By left proper form they mean that all the column bases C_i of the Hankel-blocks, where

$$\begin{aligned} C_1 &= Q_1^T, \\ C_i &= \begin{pmatrix} R_i C_{i-1} & Q_i^T \end{pmatrix}, \end{aligned}$$

should have orthonormal columns. The second stage recursions will essentially be first-stage recursions in the opposite order. Note that the Hankel-block $H_i = G_{i+1} C_i$.

We follow the notation used in [11], and use hats to denote the representation in right proper form. Consider the following recursions:

$$\begin{aligned} \begin{pmatrix} P_i \\ \widetilde{R}_i \end{pmatrix} &\approx \begin{pmatrix} \widehat{P}_i \\ \widehat{R}_i \end{pmatrix} \Sigma_i F_i^H, \quad \tau\text{-accurate SVD}, \\ \widetilde{R}_{i-1} &= \Sigma_i F_i^H R_{i-1}, \\ \widehat{Q}_{i-1} &= Q_{i-1} F_i \Sigma_i^H, \end{aligned} \tag{10}$$

with the understanding that $\widetilde{R}_N$ and $\widehat{R}_N$ are empty matrices. Then it is easy to check that the new row bases

$$\begin{aligned} \widehat{G}_n &= \widehat{P}_n, \\ \widehat{G}_i &= \begin{pmatrix} \widehat{P}_i & \\ & \widehat{G}_{i+1} \widehat{R}_i \end{pmatrix}, \end{aligned}$$

have orthonormal columns and that the hatted sequences form a valid SSS representation for the given matrix. The generators $\{\widehat{P}_i\}$, $\{\widehat{R}_i\}$ and $\{\widehat{Q}_i\}$ are all $r \times r$ matrices. For our problem, our main goal is to have a compact form of generators and the orthonormality does not matter much. Therefore, we only need the first stage. We do not introduce the next stage, and the interested readers can refer to Section 3.7 of [11].

In Equation (10), it uses truncated SVD to compute low-rank approximations of generators $\{P_i\}$. For accuracy, we can let τ be small or zero. We can also use RRQR [41] or interpolative decomposition (ID) [42] to find a low-rank approximation. In our implementation, we used ID and the computed generators $\{\widehat{P}_i\}$ are not orthonormal.

Complexity

The recompression consists of three steps:

1. Compute a low-rank approximation of $\begin{pmatrix} P_i \\ \widetilde{R}_i \end{pmatrix}$ of dimensions $r \times 2r$ or $2r \times 2r$. If using ID, it costs $2r^3 + (N-2)4r^3 = O(4nr^2)$ flops.
2. Compute $\widetilde{R}_{i-1} = X_i R_{i-1}$, where X is an $r \times 2r$ matrix and R_{i-1} is of dimension $2r \times 2r$. It costs $(N-2) \cdot 2r \times 2r^2 = O(4nr^2)$ flops.
3. Compute $\widehat{Q}_{i-1} = Q_{i-1} \cdot X^T$, where X^T is of dimension $2r \times r$ and Q_{i-1} is of dimension $r \times 2r$. It costs $(N-1) \cdot 2r \times 2r^2 = O(4nr^2)$ flops.

We use the fact that ID costs $O(mnk)$ flops for computing a rank k approximation of a $m \times n$ matrix, Therefore, the recompression of matrix $\bar{C}$ costs $O(12nr^2)$ flops in total. Thus, computing the SSS representation of $C = L^{-1} A L^{-T}$ totally costs $(\frac{5}{3} + \frac{31}{3} + 10 + 12)nr^2 = O(34nr^2)$ flops.

4. Banded Reduction for Symmetric SSS Matrix

We know that matrix C is an SSS matrix and we can use the method in Section 3.2 to get its compact SSS representation. In this section, we introduce how to tridiagonalize C by using its SSS representations.

For clearness, we assume that C is a 4×4 SSS matrix, the same as Equation (5),

$$C = \begin{bmatrix} D_1 & U_1V_2^T & U_1W_2V_3^T & U_1W_2W_3V_4^T \\ P_2Q_1^T & D_2 & U_2V_3^T & U_2W_3V_4^T \\ P_3R_2Q_1^T & P_3Q_2^T & D_3 & U_3V_4^T \\ P_4R_3R_2Q_1^T & P_4R_3Q_2^T & P_4Q_3^T & D_4 \end{bmatrix}. \tag{11}$$

Since matrix C is symmetric, we have $P_i = V_i$, $Q_i = U_i$ and $R_i = W_i^T$.

We perform the following steps and see how to convert C into a block tridiagonal form by working on the generators of C.

1. Work on the last two block rows. The off-diagonal block is

$$\begin{bmatrix} P_3R_2Q_1^T & P_3Q_2^T \\ P_4R_3R_2Q_1^T & P_4R_3Q_2^T \end{bmatrix} = \begin{bmatrix} P_3 \\ P_4R_3 \end{bmatrix} \odot \begin{bmatrix} R_2Q_1^T & Q_2^T \\ R_2Q_1^T & Q_2^T \end{bmatrix},$$

where $\odot$ means to multiply in the same block row. Find an orthogonal matrix $Q \in \mathbb{C}^{2r\times 2r}$ such that $Q \cdot \begin{bmatrix} P_3 \\ P_4R_3 \end{bmatrix} = \begin{bmatrix} \hat{P}_3 \\ 0 \end{bmatrix}$. Compute $Q \cdot \begin{bmatrix} D_3 & Q_3P_4^T \\ P_4Q_3^T & D_4 \end{bmatrix} \cdot Q^T = \begin{bmatrix} \hat{D}_3 & \hat{Q}_3^T \\ \hat{Q}_3 & \hat{D}_4 \end{bmatrix}$, and define $\hat{R}_3 = 0, \hat{P}_4 = I$, and update $\hat{D}_3, \hat{Q}_3$. Now, we have

$$C = \begin{bmatrix} D_1 & U_1V_2^T & U_1W_2\hat{V}_3^T & 0 \\ P_2Q_1^T & D_2 & U_2\hat{V}_3^T & 0 \\ \hat{P}_3R_2Q_1^T & \hat{P}_3Q_2^T & \hat{D}_3 & \hat{U}_3 \\ 0 & 0 & \hat{Q}_3^T & D_4 \end{bmatrix}.$$

2. Work on the 2-nd and 3-rd block rows. The off-diagonal block is

$$\begin{bmatrix} P_2Q_1^T \\ \hat{P}_3R_2Q_1^T \end{bmatrix} = \begin{bmatrix} P_2 \\ \hat{P}_3R_2 \end{bmatrix} \odot \begin{bmatrix} Q_1^T \\ Q_1^T \end{bmatrix}.$$

We can compute an orthogonal matrix with dimension $2r$ such that $Q \cdot \begin{bmatrix} P_2 \\ \hat{P}_3R_2 \end{bmatrix} = \begin{bmatrix} \hat{P}_2 \\ 0 \end{bmatrix}$. Compute $Q \cdot \begin{bmatrix} D_2 & Q_2\hat{P}_3^T \\ \hat{P}_3Q_2^T & \hat{D}_3 \end{bmatrix} \cdot Q^T = \begin{bmatrix} \hat{D}_2 & \hat{Q}_2^T \\ \hat{Q}_2 & \hat{D}_3 \end{bmatrix}$, and define $\hat{R}_2 = 0, \hat{P}_3 = I_r$, and update $\hat{D}_2, \hat{Q}_2$. Now, we will introduce a bulge at positions $(2,4)$ and $(4,2)$, and C looks like

$$C = \begin{bmatrix} D_1 & U_1\hat{V}_2^T & 0 & 0 \\ \hat{P}_2Q_1^T & \hat{D}_2 & \hat{U}_2 & X \\ 0 & \hat{Q}_2^T & \hat{D}_3 & \hat{U}_3 \\ 0 & X^T & \hat{Q}_3^T & D_4 \end{bmatrix}.$$

The bulge is computed as $\begin{bmatrix} 0 & \hat{Q}_3^T \end{bmatrix} \cdot Q^T = \begin{bmatrix} X & \hat{Q}_3^T \end{bmatrix}$, and $\hat{Q}_3$ is updated. We can use the standard chasing algorithm [43] to eliminate the bulge. The bulge chasing process does not affect the top-left part of matrix C which is represented in SSS form. We draw attention to the fact that $P_i = V_i \Rightarrow I_r$, $R_i = W_i^T \Rightarrow 0$, for $i = 3, 4$, and the block tridiagonal matrix is defined by $\hat{D}_i$ and $\hat{Q}_i$ or $\hat{U}_i$. Finally, matrix C has the following form after bulge chasing,

$$C = \begin{bmatrix} D_1 & U_1\hat{V}_2^T & 0 & 0 \\ \hat{P}_2Q_1^T & \hat{D}_2 & \hat{U}_2 & 0 \\ 0 & \hat{Q}_2^T & \hat{D}_3 & \hat{U}_3 \\ 0 & 0 & \hat{Q}_3^T & \hat{D}_4 \end{bmatrix}.$$

3. Define $\hat{Q}_1^T = \hat{P}_2 Q_1^T$, $\hat{D}_1 = D_1$ and $\hat{P}_2 = I_r$. Finally, we get the block tridiagonal matrix

$$C = \begin{bmatrix} \hat{D}_1 & \hat{U}_1 & 0 & 0 \\ \hat{Q}_1^T & \hat{D}_2 & \hat{U}_2 & 0 \\ 0 & \hat{Q}_2^T & \hat{D}_3 & \hat{U}_3 \\ 0 & 0 & \hat{Q}_3^T & \hat{D}_4 \end{bmatrix}.$$

The whole procedure is summarized in Algorithm 1. The procedure starts from the last row ($k = N$) and ends at the second row ($k = 2$), i.e., Step 11 computes the off-diagonal block $\hat{Q}_1$ of matrix C above.

Algorithm 1: (Symmetric banded reduction algorithm for symmetric SSS matrix) Assume that C is an $N \times N$ block symmetric SSS matrix, and its generators are $\{P_i\}, \{R_i\}, \{Q_i\}$ and $\{D_i\}$, for $i = 1, \cdots, N$ and each generator is a $r \times r$ matrix.

Inputs: generators $\{P_i\}, \{R_i\}, \{Q_i\}$ and $\{D_i\}$, for $i = 1, \cdots, N$

Outputs: a block tridiagonal matrix defined in $\{D_i\}$ and $\{Q_i\}$.

1. `DO` $k = N : -1 : 3$
2. Compute orthogonal matrix H_k such that $H_k \cdot \begin{bmatrix} P_{k-1} \\ P_k R_{k-1} \end{bmatrix} = \begin{bmatrix} \hat{P}_{k-1} \\ 0 \end{bmatrix}$, and update $P_{k-1} = \hat{P}_{k-1}$.
3. Compute $H_k \cdot \begin{bmatrix} D_{k-1} & Q_{k-1} P_k^T \\ P_k Q_{k-1}^T & D_k \end{bmatrix} H_k^T = \begin{bmatrix} \hat{D}_{k-1} & \hat{U}_{k-1} \\ \hat{Q}_{k-1} & \hat{D}_k \end{bmatrix}$, and define $R_{k-1} = 0$, $P_k = I_r$, and update $D_{k-1} = \hat{D}_{k-1}$, $Q_{k-1} = \hat{Q}_{k-1}$ and $D_k = \hat{D}_k$.
4. `if` $k < N$ %(it will introduce a bulge.)
5. Compute the bulge X by computing $H_k \cdot \begin{bmatrix} 0 \\ Q_k \end{bmatrix} = \begin{bmatrix} X \\ \hat{Q}_k \end{bmatrix}$, and update $Q_k = \hat{Q}_k$;
6. `for` $i = k, N-1$
7. Apply the standard chasing procedure and chase the bulge down;
8. `end for`
9. `end if`
10. `END DO`
11. when $k = 2$, compute $\hat{Q}_1^T = \hat{P}_2 Q_1^T$ and update $Q_1 = \hat{Q}_1$.

Remark 1. *By further computing the QR factorization of $Q_i = \widehat{Q}\widehat{R}_i$ for $i = 2, \cdots, N$, we can get an symmetric banded matrix, and its off-diagonal blocks are $\widehat{R}_i$ and its diagonal blocks are $\widehat{D}_i = \widehat{Q}_i^T D_i \widehat{Q}_i$.*

Remark 2. *For an $N \times N$ block SSS matrix, the number of D_i is N, the number of P_i is N-1, Q_i is N-1, and R_i is N-2, and the total number of generators is $4(N-1)$. The lower triangular part of C has $\frac{N(N+1)}{2}$ blocks. When $N \geq 6$, the generators require less storage than storing C explicitly as a dense matrix.*

Remark 3. *Algorithm 1 is based on eliminating block rows and columns one by one from the bottom. We can get a similar algorithm by reducing the block rows and columns from the top.*

Complexity

For a symmetric SSS matrix, we assume that all its generators are $r \times r$ matrices. We follow the steps of Algorithm 1 to estimate its computational cost.

1. Step 2: QR factorization of a $2r \times r$ tall matrix costs $2r^2(2r - \frac{r}{3}) = \frac{10}{3}r^3$. It totally executes $N - 2$ times, and thus it costs $O(\frac{10}{3}nr^2)$ flops.

2. Step 3: Computing $P_kQ_{k-1}^T$ costs $2r^3$ and the product of H_k and H_k^T with $\begin{bmatrix} D_{k-1} & Q_{k-1}P_k^T \\ P_kQ_{k-1}^T & D_k \end{bmatrix}$ costs $2\cdot 2(2r)^3 = 32r^3$ flops. Since it executes $N-2$ times, it costs $O(34nr^2)$ flops in total.
3. Step 4:
   - Computing the bulge costs $2\cdot 2r \times 2r \times r = 8r^3$. There are $N-2$ times, and it costs $8nr^2$ flops in total.
   - Each bulge chasing costs $\frac{10}{3}r^3 + 2\times(2r)\times(2r)\times(3r) + 2\times(2r)\times(2r)\times(2r) = (\frac{10}{3}+24+16)r^3 = (\frac{10}{3}+40)r^3$.
   - For k-th step, it requires $N-k$ bulge chasing steps. Since $\sum_{k=N-1}^{3}(N-k) = \sum_{\ell=1}^{N-3}\ell = \frac{(N-3)(N-2)}{2}$, it totally costs $O((\frac{10}{3}+40)\frac{1}{2}(N^2-5N)r^3 = O(\frac{65}{3}(n^2r-5nr^2))$ flops.
4. Step 11: it costs $2r^3$ flops.

Therefore, the block tridiagonalization of a symmetric SSS matrix totally costs $O(\frac{65}{3}n^2r + (34+\frac{10}{3}-\frac{325}{3}+8)nr^2) = O(\frac{65}{3}n^2r - \frac{188}{3}nr^2)$ flops.

5. Numerical Results

In this section we test the accuracy of the banded matrix obtained by using SSS matrix techniques. We further compare the accuracy of the proposed algorithms in computing the eigenvalues of symmetric banded positive generalized eigenvalue matrices. All the numerical results are obtained by using Matlab 2017(b) on a laptop with 16GB memory.

Example 1. *Assume that A and B are two randomized symmetric banded matrices and B is further positive definite, which are constructed by using the following Matlab codes*

- *A = rand(n); B = rand(n);*
- *A = (A + A')/2; B = (B + B')/2 + α∗eye(n);*
- *A = triu(tril(A,r), −r); B = triu(tril(B,r), −r);*

where n is the dimension of matrix and r is the semi-bandwidth, α is a constant to make sure B positive, which is 10 *in our experiments. We compute matrix* $C = L^{-1}AL^{-T}$ *explicitly and L is the lower Cholesky factor of matrix B. Then, we compute* $\|C-\widehat{C}\|_F$ *where* $\widehat{C} = Q_S'\hat{T}Q_S$, $\hat{T}$ *is the symmetric banded matrix computed by Algorithm 1, and* Q_S *is accumulated orthogonal matrix in Algorithm 1. The results are shown in Table 2. For simplicity we assume* $n = N\cdot r$, *i.e., n is divisible by r.*

We let $N = 16, 64, 256, 512$ and $r = 8, 16, 32$, respectively. The backward errors of Algorithm 1 are shown in Table 2, and the times cost by Algorithm 1 are shown in Table 3. When $N = 512$ and $r = 32$, it is out of memory on the computer used for experiments with 16GB memory, and the result is not included in the following tables. The results show that the proposed algorithm is numerically stable.

Table 2. Backward errors of the computed banded matrix by Algorithm 1.

	$r = 8$	$r = 16$	$r = 32$
$N = 16$	2.23×10^{-15}	4.74×10^{-15}	1.39×10^{-14}
$N = 64$	8.83×10^{-15}	1.89×10^{-14}	5.82×10^{-14}
$N = 256$	3.40×10^{-14}	7.44×10^{-14}	2.27×10^{-13}
$N = 512$	6.69×10^{-14}	1.46×10^{-13}	-

Table 3. The times in second cost by Algorithm 1.

	$r = 8$	$r = 16$	$r = 32$
$N = 16$	5.33×10^{-2}	2.78×10^{-2}	4.04×10^{-2}
$N = 64$	1.34×10^{-1}	3.00×10^{-1}	2.18×10^{0}
$N = 256$	4.87×10^{0}	3.37×10^{1}	1.40×10^{2}
$N = 512$	3.91×10^{1}	2.77×10^{2}	-

Example 2. *In this example we compare the accuracy of the computed generalized eigenvalues by the SSS approach. After obtaining the symmetric banded matrix $\hat{T}$ from Algorithm 1, we call the Matlab routine* `eig` *to compute the eigenvalues of $\hat{T}$, and then compare with the eigenvalues computed directly from A and B (by using* `eig(A,B,'chol')`*). The relative errors are measured as* $\max_{i=1}^{n} \frac{|\hat{\lambda}_i - \lambda_i|}{\lambda_i}$*, where $\hat{\lambda}_i$ is the eigenvalue computed by using Algorithm 1, and λ_i is the eigenvalue computed directly from A and B. The maximum errors and maximum relative errors are shown in Tables 4 and 5, respectively. Matrices A and B are constructed as in Example 1, and the meanings of parameters are the same.*

Table 4. The maximum errors of eigenvalues computed by Algorithm 1.

	$r = 8$	$r = 16$	$r = 32$
$N = 16$	2.55×10^{-15}	4.44×10^{-15}	1.73×10^{-14}
$N = 64$	3.11×10^{-15}	9.99×10^{-15}	3.71×10^{-14}
$N = 256$	1.76×10^{-14}	1.87×10^{-14}	1.51×10^{-13}
$N = 512$	2.93×10^{-14}	7.79×10^{-14}	-

Table 5. The maximum relative errors of eigenvalues computed by Algorithm 1.

	$r = 8$	$r = 16$	$r = 32$
$N = 16$	7.29×10^{-14}	6.25×10^{-14}	5.15×10^{-14}
$N = 64$	9.15×10^{-14}	3.89×10^{-11}	3.80×10^{-13}
$N = 256$	3.92×10^{-13}	5.36×10^{-11}	2.73×10^{-12}
$N = 512$	1.12×10^{-12}	1.70×10^{-12}	-

6. Conclusions

In this paper, the rank-structured matrix techniques are first used to reduce a banded generalized eigenvalue problem to a banded standard eigenvalue problem, and it is the first time that rank-structured matrix is used for such eigenvalue problems. Note that there are some works for reducing a rank-structured matrix to its tridiagonal or Hessenberg forms [25], which is different from this work. We in this work focus on the symmetric banded matrices which are sparse, not one particular rank-structured matrix such as quasiseparable, semiseparable and so on. In particular, we use the fast algorithms based on the sequentially semiseparable (SSS) matrix to reduce the banded symmetric generalized eigenvalue problems. The whole process of the proposed algorithm is shown in Algorithm 1, and the complexity analysis is also included. Comparing with the classical algorithms in LAPACK, the algorithm proposed in this paper requires the same order of storage and computation cost. The newly proposed algorithm consists of many small matrix-matrix multiplications which can be potentially executed in parallel. We plan to implement our algorithm by leveraging `CHAMELEON` library [13,14] and even extend it to the distributed parallel computing case by combining some data redistribution techniques such as [44] in near future.

Author Contributions: Conceptualization, S.L.; Data curation, F.Y. and L.D.; Formal analysis, F.Y., C.C. and B.Y.; Writing—review & editing, H.J., H.W. and L.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by NSFC (No. 2021YFB0300101, 62073333, 61902411, 62032023, 12002382, 11275269, 42104078), 173 Program of China (2020-JCJQ-ZD-029), Open Research Fund from State Key Laboratory of High Performance Computing of China (HPCL) (No. 202101-01), Guangdong Natural Science Foundation (2018B030312002), and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant (No. 2016ZT06D211).

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Not Applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Anderson, E.; Bai, Z.; Bischof, C.; Blackford, S.; Demmel, J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A.; et al. *LAPACK Users' Guide*, 3rd ed.; SIAM: Philadelphia, PA, USA, 1999.
2. Wilkinson, J. Some recent advances in numerical linear algebra. In *The State of the Art in Numerical*; Academic Press: New York, NY, USA, 1977.
3. Dhillon, I.S.; Parlett, B.N. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra Appl.* **2004**, *387*, 1–28.
4. Brebner, M.; Grad, J. Eigenvalues of Ax= λBx for real symmetric matrices A and B computed by reduction to a pseudosymmetric form and the HR process. *Linear Algebra Appl.* **1982**, *43*, 99–118.
5. Tisseur, F. Tridiagonal-diagonal reduction of symmetric indefinite pairs. *SIAM J. Matrix Anal. Appl.* **2004**, *26*, 215–232.
6. Crawford, C. Reduction of a band-symmetric generalized eigenvalue problem. *Commun. ACM* **1973**, *16*, 41–44.
7. Lang, B. Efficient reduction of banded hermitian positive definite generalized eigenvalue problems to banded standard eigenvalue problems. *SIAM J. Sci. Comput.* **2019**, *41*, C52–C72.
8. Rippl, M.; Lang, B.; Huckle, T. Parallel eigenvalue computation for banded generalized eigenvalue problems. *Parallel Comput.* **2019**, *88*, 102542.
9. Marek, A.; Blum, V.; Johanni, R.; Havu, V.; Lang, B.; Auckenthaler, T.; Heinecke, A.; Bungartz, H.; Lederer, H. The ELPA library: scalable parallel eigenvalue solutions for electronic structure theory and computational science. *J. Phys. Condens. Matter* **2014**, *26*, 1–15.
10. Chandrasekaran, S.; Dewilde, P.; Gu, M.; Pals, T.; Sun, X.; van der Veen, A.J.; White, D. *Fast Stable Solvers for Sequentially Semi-Separable Linear Systems of Equations*; Technical Report; University of California: Berkeley, CA, USA, 2003.
11. Chandrasekaran, S.; Dewilde, P.; Gu, M.; Pals, T.; Sun, X.; van der Veen, A.J.; White, D. Some fast algorithms for sequentially semiseparable representation. *SIAM J. Matrix Anal. Appl.* **2005**, *27*, 341–364.
12. Singh, V. The inverse of a certain block matrix. *Bull. Aust. Math. Soc.* **1979**, *20*, 161–163.
13. Chameleon Software Homepage. Available online: https://solverstack.gitlabpages.inria.fr/chameleon/ (accessed on the 22 April 2022).
14. Agullo, E.; Augonnet, C.; Dongarra, J.; Ltaief, H.; Namyst, R.; Thibault, S.; Tomov, S. A hybridization methodology for high-performance linear algebra software for GPUs. In *GPU Computing Gems Jade Edition*; Elsevier: Amsterdam, The Netherlands, 2012; pp. 473–484.
15. Chandrasekaran, S.; Gu, M. A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices. *Numer. Math.* **2004**, *96*, 723–731.
16. Chandrasekaran, S.; Gu, M. Fast and stable eigendecomposition of symmetric banded plus semi-separable matrices. *Linear Algebra Appl.* **2000**, *313*, 107–114.
17. Vandebril, R.; Van Barel, M.; Mastronardi, N. *Matrix Computations and Semiseparable Matrices, Volume I: Linear Systems*; Johns Hopkins University Press: Baltimore, MD, USA, 2008.
18. Hackbusch, W. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing* **1999**, *62*, 89–108.
19. Hackbusch, W.; Grasedyck, L.; Börm, S. An introduction to hierarchical matrices. *Math. Bohem.* **2002**, *127*, 229–241.
20. Hackbusch, W.; Khoromskij, B. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part II: Application to multi-dimensional problems. *Computing* **2000**, *64*, 21–47.
21. Börm, S.; Grasedyck, L.; Hackbusch, W. Introduction to hierarchical matrices with applications. *Eng. Anal. Bound. Elem.* **2003**, *27*, 405–422.
22. Hackbusch, W.; Khoromskij, B.; Sauter, S. On $\mathcal{H}^2$-matrices. In *Proceedings of the Lecture on Applied Mathematics*; Bungartz, H., Hope, R.H.W., Zenger, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 9–29.
23. Hackbusch, W.; Börm, S. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing* **2002**, *69*, 1–35.

24. Eidelman, Y.; Gohberg, I. On a new class of structured matrices. *Integral Equ. Oper. Theory* **1999**, *34*, 293–324.
25. Eidelman, Y.; Gohberg, I.; Gemignani, L. On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms. *Linear Algebra Appl.* **2007**, *420*, 86–101.
26. Vandebril, R.; Van Barel, M.; Mastronardi, N. *Matrix Computations and Semiseparable Matrices, Volume II: Eigenvalue and Singular value Methods*; Johns Hopkins University Press: Baltimore, MD, USA, 2008.
27. Chandrasekaran, S.; Dewilde, P.; Gu, M.; Lyons, W.; Pals, T. A fast solver for HSS representations via sparse matrices. *SIAM J. Matrix Anal. Appl.* **2006**, *29*, 67–81.
28. Li, S.; Gu, M.; Cheng, L.; Chi, X.; Sun, M. An accelerated divide-and-conquer algorithm for the bidiagonal SVD problem. *SIAM J. Matrix Anal. Appl.* **2014**, *35*, 1038–1057.
29. Liao, X.; Li, S.; Lu, Y.; Roman, J.E. A parallel structured divide-and-conquer algorithm for symmetric tridiagonal eigenvalue problems. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *32*, 367–378.
30. Zhang, J.; Su, Q.; Tang, B.; Wang, C.; Li, Y. DPSNet: Multitask Learning Using Geometry Reasoning for Scene Depth and Semantics. *IEEE Trans. Neural Networks Learn. Syst.* **2021**, 1–12. https://doi.org/10.1109/TNNLS.2021.3107362.
31. Rebrova, E.; Chávez, G.; Liu, Y.; Ghysels, P.; Li, X.S. A study of clustering techniques and hierarchical matrix formats for kernel ridge regression. In Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Vancouver, BC, Canada, 21–25 May 2018; pp. 883–892.
32. Chávez, G.; Liu, Y.; Ghysels, P.; Li, X.S.; Rebrova, E. Scalable and memory-efficient kernel ridge regression. In Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, 18–22 May 2020; pp. 956–965.
33. Erlandson, L.; Cai, D.; Xi, Y.; Chow, E. Accelerating parallel hierarchical matrix-vector products via data-driven sampling. In Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, 18–22 May 2020; pp. 749–758.
34. Cai, D.; Chow, E.; Erlandson, L.; Saad, Y.; Xi, Y. SMASH: Structured matrix approximation by separation and hierarchy. *Numer. Linear Algebra Appl.* **2018**, *25*, e2204.
35. Kailath, T. Fredholm resolvents, Wiener-Hopf equations, and Riccati differential equations. *IEEE Trans. Inf. Theory* **1969**, *15*, 665–672.
36. Asplund, E. Inverses of matrices aij which satisfy a_{ij} =0 for $j > i + p$. *Math. Scand.* **1959**, *7*, 57–60.
37. Barrett, W.; Feinsilver, P. Inverses of banded matrices. *Linear Algebra Appl.* **1981**, *4*, 111–130.
38. Vanderbril, R.; Barel, M.V.; Mastronardi, N. A note on the representation and definition of semiseparable matrices. *Numer. Linear Algebra Appl.* **2005**, *12*, 839–858.
39. Dewilde, P.; van der Veen, A. *Time-Varying Systems and Computations*; Kluwer Academic Publishers: Amsterdam, The Netherlands, 1998.
40. Chandrasekaran, S.; Dewilde, P.; Gu, M.; Pals, T.; van der Veen, A.J. Fast stable solver for sequentially semi-separable linear systems of equations. In Proceedings of the International Conference on High-Performance Computing, Bangalore, India, 18–21 December 2002; pp. 545–554.
41. Gu, M.; Eisenstat, S.C. Efficient algorithms for computing a strong-rank revealing QR factorization. *SIAM J. Sci. Comput.* **1996**, *17*, 848–869.
42. Cheng, H.; Gimbutas, Z.; Martinsson, P.; Rokhlin, V. On the compression of low rank matrices. *SIAM J. Sci. Comput.* **2005**, *26*, 1389–1404.
43. Schwarz, H.R. Tridiagonalization of a symmetric band matrix. *Numer. Math.* **1968**, *12*, 231–241.
44. Li, S.; Jiang, H.; Dong, D.; Huang, C.; Liu, J.; Liao, X.; Chen, X. Efficient data redistribution algorithms from irregular to block cyclic data distribution. *IEEE Trans. Parallel Distrib. Syst.* **2022**. https://doi.org/10.1109/TPDS.2022.3166484.