MATHICSE Technical Report
Nr. 01.2018
Janvier 2018

A fast spectral divide-and-conquer method for banded matrices

Ana Susnjara, Daniel Kressner

A fast spectral divide-and-conquer method for banded matrices

Ana Šušnjara*   Daniel Kressner†

Abstract

Based on the spectral divide-and-conquer algorithm by Nakatsukasa and Higham [SIAM J. Sci. Comput., 35(3):A1325–A1349, 2013], we propose a new algorithm for computing all the eigenvalues and eigenvectors of a symmetric banded matrix. For this purpose, we combine our previous work on the fast computation of spectral projectors in the so called HODLR format, with a novel technique for extracting a basis for the range of such a HODLR matrix. The numerical experiments demonstrate that our algorithm exhibits quasilinear complexity and allows for conveniently dealing with large-scale matrices.

1 Introduction

Given a large symmetric banded matrix $A \in \mathbb{R}^{n\times n}$, we consider the computation of its *complete* spectral decomposition

$$A = Q\Lambda Q^T, \quad \Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \dots, \lambda_n), \tag{1}$$

where $\lambda_i, i = 1, \dots, n$ are the eigenvalues of A and the columns of the orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ the corresponding eigenvectors. This problem has attracted quite some attention from the early days of numerical linear algebra until today, particularly when A is a a tridiagonal matrix.

A number of applications give rise to banded eigenvalue problems. For example, they constitute a critical step in solvers for *general* dense symmetric eigenvalue problems. Nearly all existing approaches, with the notable exception of [22], first reduce a given dense symmetric matrix to tridiagonal form. This is followed by a method for determining the spectral decomposition of a tridiagonal matrix, such as the QR algorithm, the classical divide-and-conquer (D&C) method or the algorithm of multiple relatively robust representations (MRRR). All these methods have complexity $\mathcal{O}(n^2)$ or higher; simply because all n eigenvectors are computed and stored explicitly.

On a modern computing architecture with a memory hierarchy, it turns out to be advantageous to perform the tridiagonalization based on successive band reduction [6], with a symmetric banded matrix as an intermediate step [3, 5, 12, 14, 23]. In this context, it would be preferable to design an eigenvalue solver that works directly with banded matrices, therefore avoiding the reduction from banded to tridiagonal form. Such a possibility has been explored for classical D&C in [2, 13]. However, the proposed methods seem to suffer from numerical instability or an unsatisfactory complexity growth as the bandwidth increases.

*MATH-ANCHP, École Polytechnique Fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland. E-mail: ana.susnjara@epfl.ch. The work of Ana Šušnjara has been supported by the SNSF research project *Low-rank updates of matrix functions and fast eigenvalue solvers.*

†MATH-ANCHP, École Polytechnique Fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland. E-mail: daniel.kressner@epfl.ch.

In this paper we propose a new and fast approach to computing the spectral decomposition of a symmetric banded matrix. This is based on the spectral D&C method from [22], which recursively splits the spectrum using invariant subspaces extracted from spectral projectors associated with roughly half of the spectrum. In previous work [17], we have developed a fast method for approximating such spectral projectors in a hierarchical low-rank format, the so called HODLR (hierarchically off-diagonal low-rank) format [1]. However, the extraction of the invariant subspace, requires to determine a basis for the range of the spectral projector. This represents a major challenge. We present an efficient algorithm for computing an orthonormal basis of an invariant subspace in the HODLR format, which heavily exploits properties of spectral projectors. The matrix of eigenvectors is stored implicitly, via orthonormal factors, where each factor is an orthonormal basis for an invariant subspace. Our approach extends to general symmetric HODLR matrices.

Several existing approaches that use hierarchical low-rank formats for the fast solution of eigenvalue problems are based on computing (inexact) LDL^T decompositions in such a format, see [11, sec. 13.5] for an overview. These decompositions allow to slice the spectrum of a symmetric matrix into smaller chunks and are particularly well suited when only the eigenvalues and a few eigenvectors are needed.

To the best our knowledge, the only existing fast methods suitable for the complete spectral decomposition of a large symmetric matrix are based on variations the classical D&C method by Cuppen for a symmetric tridiagonal matrix [7]. One recursion of the method divides, after a rank-one perturbation, the matrix into a 2×2 block diagonal matrix. In the conquer phase the rank-one perturbation is incorporated by solving a secular equation for the eigenvalues and applying a Cauchy-like matrix to the matrix of eigenvectors. Gu and Eisenstat [10] not only stabilized Cuppen's method but also observed that the use of the fast multipole method for the Cauchy-like matrix multiplication reduced its complexity to $\mathcal{O}(n^2)$ for computing all eigenvectors. Vogel et al. [24] extended these ideas beyond tridiagonal matrices, to general symmetric HSS (hierarchically semiseparable) matrices. Moreover, by representing the matrix of eigenvectors in factored form, the overall cost reduces to $\mathcal{O}(n \log^2 n)$. While our work bears similarities with [24], such as the storage of eigenvectors in factored form, it differs in several key aspects. First, our developments use the HODLR format while [24] uses the HSS format. The later format stores the low-rank factors of off-diagonal blocks in a nested manner and thus reduces the memory requirements by a factor $\log n$ *if* the involved ranks stay on the same level. However, one may need to work with rather large values of n in order to gain significant computational savings from working with HSS instead of HODLR. A second major difference is that the spectral D&C method used in this paper has, despite the similarity in name, little in common with Cuppen's D&C. One advantage of using spectral D&C is that it conveniently allows to compute only parts of the spectrum. A third major difference is that [24] incorporates a perturbation of rank $r > 1$, as it is needed to process matrices of bandwidth larger than one by sequentially splitting it up into r rank-one perturbations. The method presented in this paper processes higher ranks directly, avoiding the need for splitting and leveraging the performance of level 3 BLAS operations. While the timings reported in [24] cover matrices of size up to 10 240 and appear to be comparable with the timings presented in this paper, our experiments additionally demonstrate that our newly proposed method allows for conveniently dealing with large-scale matrices.

The rest of the paper is organized as follows. In section 2, we recall the spectral divide-and-conquer algorithm for computing the spectral decomposition of a symmetric matrix. Section 3 gives a brief overview of the HODLR format and of a fast method for computing spectral projectors of HODLR matrices. In section 4 we discuss the fast extraction of invariant subspaces from a spectral projector given in the HODLR format. Section 5 presents the overall spectral D&C algorithm in the HODLR format for computing the spectral decomposition of a banded matrix.

Numerical experiments are presented in section 6.

2 Spectral divide-and-conquer

In this section we recall the spectral D&C method by Nakatsukasa and Higham [22] for a symmetric $n \times n$ matrix A with spectral decomposition (1). We assume that the eigenvalues are sorted in ascending order and choose a shift $\mu \in \mathbb{R}$ such that

$$\lambda_1 \le \cdots \le \lambda_\nu < \mu < \lambda_{\nu+1} \le \cdots \le \lambda_n, \qquad \nu \approx n/2.$$

The relative spectral gap associated with this splitting of eigenvalues is defined as

$$\text{gap} = \frac{\lambda_{\nu+1} - \lambda_\nu}{\lambda_n - \lambda_1}.$$

The spectral projector associated with the first ν eigenvalues is the orthogonal projector onto the subspace spanned by the corresponding eigenvectors. Given (1), it takes the form

$$\Pi_{<\mu} = Q \begin{bmatrix} I_\nu & 0 \\ 0 & 0 \end{bmatrix} Q^T.$$

Note that

$$\Pi_{<\mu}^T = \Pi_{<\mu}^2 = \Pi_{<\mu}, \qquad \text{trace}(\Pi_{<\mu}) = \text{rank}(\Pi_{<\mu}) = \nu.$$

The spectral projector associated with the other $n - \nu$ eigenvalues is given by

$$\Pi_{>\mu} = Q \begin{bmatrix} 0 & 0 \\ 0 & I_{n-\nu} \end{bmatrix} Q^T$$

and satisifies analogous properties.

The method from [22] first computes the matrix sign function

$$\text{sign}(A - \mu I) = Q \begin{bmatrix} -I_\nu & 0 \\ 0 & I_{n-\nu} \end{bmatrix} Q^T$$

and then extracts the spectral projectors via the relations

$$\Pi_{<\mu} = \frac{1}{2}(I - \text{sign}(A - \mu I)), \qquad \Pi_{>\mu} = I - \Pi_{<\mu}.$$

The ranges of these spectral projector are invariant subspaces of $A - \mu I$ and, in turn, of A. Letting $Q_{<\mu} \in \mathbb{R}^{n\times\nu}$ and $Q_{>\mu} \in \mathbb{R}^{n\times(n-\nu)}$ denote arbitrary orthonormal bases for Range($\Pi_{<\mu}$) and Range($\Pi_{>\mu}$), respectively, we therefore obtain

$$\begin{bmatrix} Q_{<\mu} & Q_{>\mu} \end{bmatrix}^T A \begin{bmatrix} Q_{<\mu} & Q_{>\mu} \end{bmatrix} = \begin{bmatrix} A_{<\mu} & 0 \\ 0 & A_{>\mu} \end{bmatrix}, \tag{2}$$

where the eigenvalues of $A_{<\mu} = Q_{<\mu}^T A Q_{<\mu}$ are $\lambda_1, \ldots, \lambda_\nu$ and the eigenvalues of $A_{>\mu} = Q_{>\mu}^T A Q_{>\mu}$ are $\lambda_{\nu+1}, \ldots, \lambda_n$. Applying the described procedure recursively to $A_{<\mu}$ and $A_{>\mu}$ leads to Algorithm 1. When the size of the matrix is below a user-prescribed minimal size n_{stop}, the recursion is stopped and a standard method for computing spectral decompositions is used, denoted by `eig`.

In the following sections, we discuss how Algorithm 1 can be implemented efficiently in the HODLR format.

Algorithm 1 Spectral D&C method

Input: Symmetric matrix $A \in \mathbb{R}^{n \times n}$.
Output: Spectral decomposition $A = Q\Lambda Q^T$.

```
1:  function $[Q, \Lambda] = \texttt{sdc}(A)$
2:      if $n \le n_{\text{stop}}$ then
3:          Return $[Q, \Lambda] = \texttt{eig}(A)$.
4:      else
5:          Choose shift $\mu$.
6:          Compute sign function of $A - \mu I$ and extract spectral projectors $\Pi_{<\mu}$ and $\Pi_{>\mu}$.
7:          Compute orthonormal bases $Q_{<\mu}, Q_{>\mu}$ of Range($\Pi_{<\mu}$), Range($\Pi_{>\mu}$).
8:          Compute $A_{<\mu} = Q_{<\mu}^T A Q_{<\mu}$ and $A_{>\mu} = Q_{>\mu}^T A Q_{>\mu}$.
9:          Call recursively $[Q_1, \Lambda_1] = \texttt{sdc}(A_{<\mu})$ and $[Q_2, \Lambda_2] = \texttt{sdc}(A_{>\mu})$.
10:         Set $Q \leftarrow \begin{bmatrix} Q_{<\mu}Q_1 & Q_{>\mu}Q_2 \end{bmatrix}$, $\Lambda = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}$.
11:     end if
12: end function
```

3 Computation of spectral projectors in HODLR format

In this section, we briefly recall the HODLR format and the algorithm from [17] for computing spectral projectors in the HODLR format.

3.1 HODLR format

Given an $n \times m$ matrix M let us consider a block matrix partitioning of the form

$$M = \left[\begin{array}{c|c} M_{11}^{(1)} & M_{12}^{(1)} \\ \hline M_{21}^{(1)} & M_{22}^{(1)} \end{array} \right]. \tag{3}$$

This partitioning is applied recursively, p times, to the diagonal blocks $M_{11}^{(1)}$, $M_{22}^{(1)}$, leading to the hierarchical partitioning shown in Figure 1. We say that M is a *HODLR matrix* of level p and HODLR rank k if all off-diagonal blocks seen during this process have rank at most k. In the HODLR format, these blocks are stored, more efficiently, in terms of their low-rank factors. For example, for $p = 2$, the HODLR format takes the form

$$M = \left[\begin{array}{c|c} \begin{array}{c|c} M_{11}^{(2)} & U_1^{(2)}\big(V_2^{(2)}\big)^T \\ \hline U_2^{(2)}\big(V_1^{(2)}\big)^T & M_{22}^{(2)} \end{array} & U_1^{(1)}\big(V_2^{(1)}\big)^T \\ \hline U_2^{(1)}\big(V_1^{(1)}\big)^T & \begin{array}{c|c} M_{33}^{(2)} & U_3^{(2)}\big(V_4^{(2)}\big)^T \\ \hline U_4^{(2)}\big(V_3^{(2)}\big)^T & M_{44}^{(2)} \end{array} \end{array} \right].$$

The definition of a HODLR matrix of course depends on how the partitioning (3) is chosen on each level of the recursion. This choice is completely determined by the integer partitions

$$n = n_1 + n_2 + \cdots n_{2^p}, \qquad m = m_1 + m_2 + \cdots + m_{2^p}, \tag{4}$$

corresponding to the sizes $n_j \times m_j$, $j = 1, \dots, 2^p$, of the diagonal blocks $M_{11}^{(p)}, \dots, M_{2^p,2^p}^{(p)}$ on the lowest level of the recursion. Given specific integer partitions (4), we denote the set of HODLR matrices of rank k by $\mathcal{H}_{n\times m}(k)$.

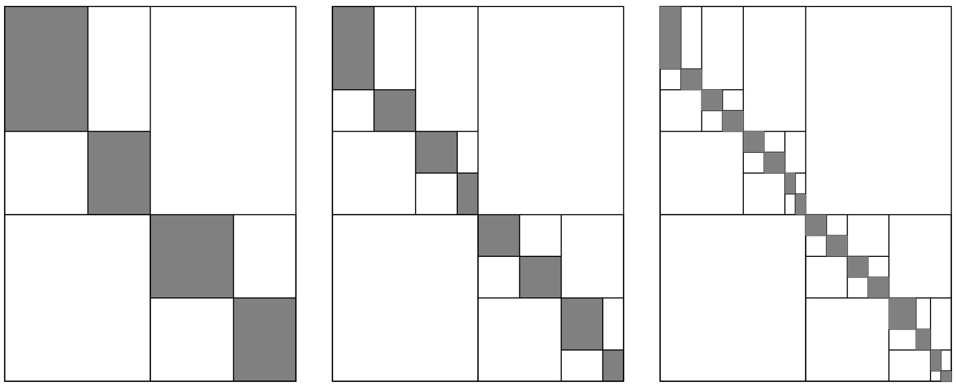

Figure 1: Illustration of HODLR matrices for $p = 2$, $p = 3$, and $p = 4$. The diagonal blocks (grey) are stored as dense matrices, while the off-diagonal blocks (white) are stored in terms of their low-rank factors.

3.2 Operations in the HODLR format

Assuming that the integer partitions (4) are balanced, $p = \mathcal{O}(\log \tilde{n})$ with $\tilde{n} = \max\{n, m\}$, and $k = \mathcal{O}(1)$, the storage of $M \in \mathcal{H}_{n\times m}(k)$ in the HODLR format requires $\mathcal{O}(\tilde{n} \log \tilde{n})$ memory. Various matrix operations with HOLDR matrices can be preformed with linear-polylogarithmic complexity. Table 1 summarizes the operations needed in this work; we refer to, e.g., [4, 11] for more details. In order to perform operations including two HODLR matrices, the corresponding partitions ought to be compatible.

The operations listed in Table 1 with subscript $\mathcal{H}$ employ recompression in order to limit the increase of off-diagonal ranks. In this paper recompression is done adaptively, such that the 2-norm approximation error in each off-diagonal block is bounded by a prescribed truncation tolerance ϵ. For matrix addition, recompression is done only after adding two off-diagonal blocks, whereas multiplying HOLDR matrices and computing the Cholesky decomposition requires recompression in intermediate steps.

Table 1: Complexity of operations involving HODLR matrices: $M \in \mathcal{H}_{n\times n}(k)$ symmetric positive definite, $T \in \mathcal{H}_{m\times m}(k)$ triangular and invertible, $M_1, M_2 \in \mathcal{H}_{n\times m}(k)$, $M_3 \in \mathcal{H}_{m\times p}(k)$, $B \in \mathbb{R}^{m\times p}$, $v \in \mathbb{R}^m$.

Operation	Computational complexity
Matrix-vector multiplication: $M_1 v$	$\mathcal{O}(k\tilde{n}\log\tilde{n})$, with $\tilde{n} = \max\{n, m\}$
Matrix addition: $M_1 +_{\mathcal{H}} M_2$	$\mathcal{O}(k^2\tilde{n}\log\tilde{n})$, with $\tilde{n} = \max\{n, m\}$
Matrix-matrix multiplication: $M_2 *_{\mathcal{H}} M_3$	$\mathcal{O}(k^2\tilde{m}\log^2\tilde{m})$, with $\tilde{m} = \max\{n, m, p\}$
Cholesky decomposition: $\mathcal{H}$-Cholesky(M)	$\mathcal{O}(k^2 n\log^2 n)$
Solution of triangular system: $T^{-1}B$	$\mathcal{O}(km\log m)$
Multiplication with (triangular)$^{-1}$: $M_1 *_{\mathcal{H}} T^{-1}$	$\mathcal{O}(k^2\tilde{n}\log^2\tilde{n})$, with $\tilde{n} = \max\{n, m\}$

In this work we also need to extract submatrices of HODLR matrices. Let $M \in \mathcal{H}_{n\times n}(k)$, associated with an integer partition $n = n_1 + \cdots + n_{2^p}$, and consider a subset of indices $C \subset \{1, \dots, n\}$. Then the submatrix $M(C, C)$ is again a HODLR matrix. To see this, consider the partitioning (3) and let $C = C_1 \cup C_2$ with $C_1 = C \cap [1, n_1^{(1)}]$ and $C_2 = C \cap [n_1^{(1)} + 1, n]$, where

$n_1^{(1)}$ is the size of $M_{11}^{(1)}$. Then

$$\begin{aligned} M(C,C) &= \left[\begin{array}{c|c} M_{11}^{(1)}(C_1,C_1) & M_{12}^{(1)}(C_1,C_2) \\ \hline M_{21}^{(1)}(C_2,C_1) & M_{22}^{(1)}(C_2,C_2) \end{array}\right] \\ &= \left[\begin{array}{c|c} M_{11}^{(1)}(C_1,C_1) & U_1^{(2)}(C_1,:)\big(V_2^{(2)}(C_2,:)\big)^T \\ \hline U_2^{(2)}(C_2,:)\big(V_1^{(2)}(C_1,:)\big)^T & M_{22}^{(1)}(C_2,C_2) \end{array}\right]. \end{aligned}$$

Hence, the off-diagonal blocks again have rank at most k. Applying this argument recursively to $M_{11}^{(1)}(C_1,C_1)$, $M_{22}^{(1)}(C_2,C_2)$ establishes $M(C,C) \in \mathcal{H}_{m\times m}(k)$, associated with the integer partition

$$|C| =: m = m_1 + m_2 + \cdots m_{2^p},$$

where m_1 is the cardinality of $C \cap [1, n_1]$, m_2 is the cardinality of $C \cap [n_1+1, n_1+n_2]$, and so on. Note that it may happen that some $m_j = 0$, in which case the corresponding blocks in the HODLR format vanish. Formally, this poses no problem in the definition and operations with HOLDR matrices. In practice, these blocks are removed to reduce overhead.

3.3 Computation of spectral projectors in the HODLR format

The method presented in [17] for computing spectral projectors of banded matrices is based on the dynamically weighted Halley iteration from [21, 22] for computing the matrix sign function. In this work, we also need a slight variation of that method for dealing with HODLR matrices.

Given a symmetric non-singular matrix A, the method from [21] uses an iteration

$$X_{k+1} = \frac{b_k}{c_k} X_k + \Big(a_k - \frac{b_k}{c_k}\Big) X_k (I + c_k X_k^T X_k)^{-1}, \quad X_0 = A/\alpha, \tag{5}$$

that converges globally cubically to $\operatorname{sign}(A)$. The parameter $\alpha > 0$ is such that $\alpha \gtrsim \|A\|_2$. The parameters a_k, b_k, c_k are computed by

$$a_k = h(l_k), \quad b_k = (a_k - 1)^2/4, \quad c_k = a_k + b_k - 1, \tag{6}$$

where l_k is determined by the recurrence

$$l_k = l_{k-1}(a_{k-1} + b_{k-1} l_{k-1}^2)/(1 + c_{k-1} l_{k-1}^2), \quad k \geq 1, \tag{7}$$

with a lower bound l_0 for $\sigma_{\min}(X_0)$, and the function h is given by

$$h(l) = \sqrt{1+\gamma} + \frac{1}{2}\sqrt{8 - 4\gamma + \frac{8(2-l^2)}{l^2\sqrt{1+\gamma}}}, \quad \gamma = \sqrt[3]{\frac{4(1-l^2)}{l^4}}.$$

In summary, except for α and l_0 the parameters determining (5) are simple and cheap to compute.

The algorithm `hQDWH` presented in [17] for banded A is essentially an implementation of (5) in the HODLR matrix arithmetic, with one major difference. Following [22], the first iteration of `hQDWH` avoids the computation of the Cholesky factorization for the evaluation of $(I + c_0 X_0^T X_0)^{-1} = (I + c_0/\alpha^2 A^2)^{-1}$ in the first iteration. Instead, a QR decomposition of a $2n \times n$ matrix $\begin{bmatrix} \sqrt{c_0} X_0 \\ I \end{bmatrix}$ is computed. This improves numerical stability and allows us to safely determine spectral projectors even for relatives gaps of order 10^{-16}. For reasons explained in [17, Remark 3.1], existing algorithms for performing QR decompositions of HODLR matrices come with various drawbacks. When A is a HODLR matrix, Algorithm 2 therefore uses a Cholesky decomposition (instead of a QR decomposition) in the first step as well. In turn, as will be demonstrated by numerical experiments in section 6, this restricts the application of the algorithm to matrices with relative spectral gaps of order 10^{-8} or larger. We do not see this as a major disadvantage in the setting under consideration. The relative gap is controlled by the choice of the shift μ in our D&C method and tiny relative spectral gaps can be easily avoided by adjusting μ.

The inexpensive estimation of α, l_0 for banded A is discussed in [17]. For a HODLR matrix A, we determine α and l_0 by applying a few steps of the (inverse) power method to A^2 and A^2/α^2, respectively.

Algorithm 2 hDWH algorithm

Input: Symmetric HODLR matrix $A \in \mathbb{R}^{n\times n}$, truncation tolerance $\epsilon > 0$, stopping tolerance $\varepsilon > 0$.

Output: Approximate spectral projectors $\Pi_{<0}$ and $\Pi_{>0}$ in the HODLR format.

```
1: if $A$ is banded then
2:     Compute $\Pi_{<0}$ and $\Pi_{>0}$ using the hQDWH algorithm [17].
3: else
4:     Compute initial parameters $\alpha \gtrsim \|A\|_2$ via power iteration on $A^2$ and $l_0 \lesssim \sigma_{\min}(A/\alpha)$ via
   inverse power iteration on $A^2/\alpha^2$.
5:     $X_0 = A/\alpha$.
6:     $k = 0$.
7:     while $|1 - l_k| > \varepsilon$ do
8:         Compute $a_k$, $b_k$, $c_k$ according to the recurrence (6).
9:         $W_k = \mathcal{H}\text{-Cholesky}(I + c_k X_k^T *_{\mathcal{H}} X_k)$.
10:        $Y_k = X_k *_{\mathcal{H}} W_k^{-1}$.
11:        $V_k = Y_k *_{\mathcal{H}} W_k^{-T}$.
12:        $X_{k+1} = \frac{b_k}{c_k} X_k +_{\mathcal{H}} \left(a_k - \frac{b_k}{c_k}\right) V_k$.
13:        $k = k + 1$.
14:        Compute $l_k$ according to the recurrence (7).
15:    end while
16:    Set $U = X_k$.
17:    Return $\Pi_{<0} = \frac{1}{2}(I - U)$ and $\Pi_{>0} = \frac{1}{2}(I + U)$.
18: end if
```

Assuming that a constant number of iterations is needed and that the HOLDR ranks of all intermediate quantities are bounded by k, the complexity of Algorithm 2 is $\mathcal{O}(k^2 n \log^2 n)$.

4 Computation of invariant subspace basis in the HODLR format

This section addresses the efficient extraction of a basis for the range of a spectral projector $\Pi_{<\mu}$ given in the HODLR format.

Assuming that $\text{rank}(\Pi_{<\mu}) = \nu$, the most straightforward approach to obtain a basis for $\text{Range}(\Pi_{<\mu})$ is to simply take its first ν columns. Numerically, this turns out to be a terrible idea, especially when A is banded.

Example 1. Let n be even and let $A \in \mathbb{R}^{n\times n}$ be a symmetric banded matrix with bandwidth b and eigenvalues distributed uniformly in $[-1,\ -10^{-1}]\cup[10^{-1},\ 1]$. In particular, $\operatorname{rank}(\Pi_{<0}) = n/2$. Figure 2 shows that the condition number of the first $n/2$ columns of $\Pi_{<0}$ grows dramatically as n increases. By computing a QR decomposition of these columns, we obtain an orthonormal basis $Q_1 \in \mathbb{R}^{n\times n/2}$. This basis has perfect condition number but, as Table 2 shows, it represents a r ather poor approximation of $\operatorname{Range}(\Pi_{<0})$.

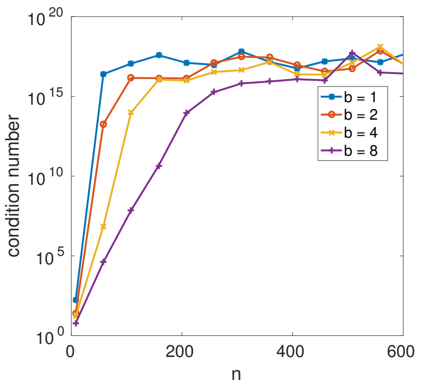

Figure 2: Condition number of the first $n/2$ columns of the spectral projector $\Pi_{<0}$ for the matrix described in Example 1 with bandwidths $b = 1, 2, 4, 8$.

Table 2: Angles (in radians) between $\operatorname{Range}(\Pi_{<0})$ and $\operatorname{Range}(Q_1)$, with Q_1 an orthonormal basis for $\operatorname{Range}(\Pi_{<0}(:, 1 : \frac{n}{2}))$.

n	$\measuredangle(\operatorname{Range}(\Pi_{<0}), \operatorname{Range}(Q_1)))$
64	$4.4916e-02$
256	$1.5692e+00$
1024	$1.5700e+00$
4096	$1.5707e+00$

There exist a number of approaches that potentially avoid the problems observed in Example 1, such as applying a QR factorization with pivoting [9, Chapter 5.4] to $\Pi_{<0}$. None of these approaches has been realized in the HODLR format. In fact, techniques like pivoting across blocks appear to be incompatible with the format.

In the following, we develop a new algorithm for computing a basis for $\operatorname{Range}(\Pi_{<\mu})$ in the HODLR format, which consists of two steps: (1) We first determine a set of well-conditioned columns of $\Pi_{<\mu}$ by performing a Cholesky factorization with *local* pivoting. As we will see below, the number of obtained columns is generally less but not much less than ν. (2) A randomized algorithm is applied to complete the columns to a basis of $\operatorname{Range}(\Pi_{<\mu})$.

4.1 Column selection by block Cholesky with local pivoting

The spectral projector $\Pi_{<\mu}$ is not only symmetric positive semidefinite but it is also idempotent. The (pivoted) Cholesky factorization of such matrices has particular properties.

Theorem 2 ([16, Theorem 10.9])**.** *Let $B \in \mathbb{R}^{n\times n}$ be a symmetric positive semidefinite matrix of rank r. Then there is a permutation matrix P such that P^TBP admits a Cholesky factorization:*

$$P^TBP = R^TR, \quad R = \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix},$$

where R_1 is a $r \times r$ upper triangular matrix with positive diagonal elements.

Note that, by the invertibility of R_1, the first r columns of BP as well as $[R_1 \quad R_2]^T$ form a basis for Range(B). The latter turns out to be orthonormal if B is idempotent.

Lemma 3 ([20, Corollary 1.2.])**.** *Suppose, in addition to the hypotheses of Theorem 2, that $B^2 = B$. Then*

$$[R_1 \quad R_2] \begin{bmatrix} R_1^T \\ R_2^T \end{bmatrix} = I_r.$$

The algorithm described in [16, Chapter 10] for realizing Theorem 2 chooses the maximal diagonal element as the pivot in every step of the standard Cholesky factorization algorithm. In turn, the diagonal elements of the Cholesky factor are monotonically decreasing and it is safe to decide which ones are considered zero numerically. Unfortunately, this algorithm, which will be denoted by `cholp` in the following, cannot be applied to $\Pi_{<\mu}$ because the diagonal pivoting strategy destroys the HODLR format. Instead, we use `cholp` only for the (dense) diagonal blocks of $\Pi_{<\mu}$.

To illustrate the idea of our algorithm, we first consider a general symmetric positive semidefinite HODLR matrix M of level 1, which takes the form

$$M = \begin{pmatrix} M_{11} & U_1V_2^T \\ V_2U_1^T & M_{22} \end{pmatrix}$$

with dense diagonal blocks M_{11}, M_{22}. Applying `cholp` to M_{11} gives a decomposition $P_1^TM_{11}P_1 = R_{11}^TR_{11}$, with the diagonal elements of R_{11} decreasing monotonically. As M, and in turn also M_{11}, will be chosen as a principal submatrix of $\Pi_{<\mu}$, Lemma 3 implies that $\|R_{11}\|_2 \le 1$. In particular, the diagonal elements of R_{11} are bounded by 1. Let s_1 denote the number of diagonal elements not smaller than a prescribed threshold δ. As will be shown in Lemma 4 below, choosing δ sufficiently close to 1 ensures that $R_{11}(1:s_1, 1:s_1)$ is well-conditioned. Letting π_1 denote the permutation associated with P_1 and setting $C_1 = \pi_1(1:s_1)$, we have

$$M_{11}(C_1, C_1) = R_{11}(1:s, 1:s)^TR_{11}(1:s, 1:s).$$

The Schur complement of this matrix in M (without considering the rows and columns neglected in the first block) is given by

$$S = M_{22} - V_2U_1(C_1,:)^TM_{11}(C_1,C_1)^{-1}U_1(C_1,:)V_2^T = M_{22} - \tilde{R}_{12}^T\tilde{R}_{12}, \tag{8}$$

where the rank of $\tilde{R}_{12} := R_{11}(1:s_1, 1:s_1)^{-T}U_1(C_1,:)V_2^T$ is not larger than the rank of $U_1V_2^T$. We again apply `cholp` to S and only retain diagonal elements of the Cholesky factor R_{22} larger or equal than δ. Letting C_2 denote the corresponding indices and setting $s_2 = |C_2|$, $C = C_1 \cup (n_1 + C_2)$, where n_1 is the size of M_{11}, we obtain the factorization

$$M(C,C) = \tilde{R}^T\tilde{R} \quad \text{with} \quad \begin{bmatrix} R_{11}(1:s_1, 1:s_1) & \tilde{R}_{12}(:, C_2) \\ 0 & R_{22}(1:s_2, 1:s_2) \end{bmatrix}.$$

For a general HODLR matrix, we proceed recursively in an analogous fashion, with the difference that we now form submatrices of HODLR matrices (see section 3.2) and the operations in (8) are executed in the HODLR arithmetic.

The procedure described above leads to Algorithm 3. Based on the complexity of operations stated in Table 1, the cost of the algorithm applied to an $n \times n$ spectral projector $\Pi_{<\mu} \in \mathcal{H}_{n\times n}(k)$ is $\mathcal{O}(k^2 n \log^2 n)$. In Line 10 we update a HODLR matrix with a matrix given by its low-rank representation. This operation essentially corresponds to the addition of two HODLR matrices. Recompression is performed when computing all off-diagonal blocks, while dense diagonal blocks are updated by a dense matrix of the corresponding size. In Line 10 we also enforce symmetry in the Schur complement.

Algorithm 3 Incomplete Cholesky factorization with local pivoting for HODLR matrices

Input: Positive semidefinite HODLR matrix $M \in \mathcal{H}_{n\times n}(k)$ of level p, tolerance $\delta > 0$.
Output: Indices $C \subset [1, n]$ and upper triangular HODLR matrix $\tilde{R}$ such that $M(C, C) \approx \tilde{R}^T \tilde{R}$, with $\tilde{r}_{ii} \geq \delta$ for $i = 1, \dots, |C|$.

```
1: function $[C, \tilde{R}]$ = hcholp_inc($M$)
2:     if $p = 0$ then
3:         Compute $[R, \pi]$ = cholp($M$) such that $M(\pi, \pi) = R^T R$.
4:         Set $s$ such that $r_{11} \geq \delta, \dots, r_{ss} \geq \delta$ and $r_{s+1,s+1} < \delta$ (or $s = n$).
5:         Return $C = \pi(1 : s)$ and $\tilde{R} = R(1 : s, 1 : s)$.
6:     else
7:         Partition $M = \begin{pmatrix} M_{11} & U_1 V_2^T \\ V_2 U_1^T & M_{22} \end{pmatrix}$.
8:         Call recursively $[C_1, \tilde{R}_{11}]$ = hcholp_inc($M_{11}$).
9:         Compute $\tilde{U}_1 = \tilde{R}_{11}^{-T} U_1(C_1, :)$.
10:        Compute $S = M_{22} -_{\mathcal{H}} V_2 \tilde{U}_1^T \tilde{U}_1 V_2^T$.
11:        Call recursively $[C_2, \tilde{R}_{22}]$ = hcholp_inc($S$).
12:        Return $C = C_1 \cup (n_1 + C_2)$ and HODLR matrix $\tilde{R} = \begin{bmatrix} \tilde{R}_{11} & \tilde{U}_1 V_2(C_2, :)^T \\ 0 & \tilde{R}_{22} \end{bmatrix}$.
13:    end if
14: end function
```

4.1.1 Analysis of Algorithm 3

The indices C selected by Algorithm 3 applied to $\Pi_{<\mu}$ need to attain two goals: (1) $\Pi_{<\mu}(:, C)$ has moderate condition number, (2) $|C|$ is not much smaller than the rank of $\Pi_{<\mu}$. In the following analysis, we show that the first goal is met when choosing δ sufficiently close to 1. The attainment of the second goal is demonstrated by the numerical experiments in section 6.

Our analysis needs to take into account that Algorithm 3 is affected by error due to truncation in the HODLR arithmetic. On the one hand, the input matrix, the spectral projector $\Pi_{<\mu}$ computed by Algorithm 2, is not exactly idempotent:

$$\Pi_{<\mu}^2 = \Pi_{<\mu} + F, \tag{9}$$

with a symmetric perturbation matrix F of small norm. On the other hand, the incomplete Cholesky factor $\tilde{R}$ returned by Algorithm 3 is inexact as well:

$$\Pi_{<\mu}(C, C) = \tilde{R}^T \tilde{R} + E, \tag{10}$$

with another symmetric perturbation matrix E of small norm. For a symmetric matrix $\Pi_{<\mu}$ satisfying (9), Theorem 2.1 in [20] shows that

$$\|\Pi_{<\mu}\|_2 \leq 1 + \|F\|_2. \tag{11}$$

The following lemma establishes a bound on the norm of the inverse of $\Pi_{<\mu}(C, C)$.

Lemma 4. *With the notation introduced above, set $\varepsilon_{\mathcal{H}} = \|E\|_2 + \|F\|_2$ and $r = |C|$, and suppose that $1 - \delta^2 + \varepsilon_{\mathcal{H}} < 1/r$. Then $\|\Pi_{<\mu}(C, C)^{-1}\|_2 \leq \frac{1}{r}\frac{1}{\delta^2 - 1 + 1/r - \varepsilon_{\mathcal{H}}}$.*

Proof. Using (10) and (11), we obtain

$$\|\tilde{R}^T\tilde{R}\|_2 \leq \|\Pi_{<\mu}(C, C)\|_2 + \|E\|_2 \leq \|\Pi_{<\mu}\|_2 + \|E\|_2 \leq 1 + \varepsilon_{\mathcal{H}}.$$

We now decompose $\tilde{R} = D + T$, such that D is diagonal with $d_{ii} = \tilde{r}_{ii} \geq \delta$ and T is strictly upper triangular. Then

$$\|D^2 + T^T D + DT + T^T T\|_2 \leq 1 + \varepsilon_{\mathcal{H}}.$$

Because the matrix on the left is symmetric, this implies

$$\lambda_{\max}(D^2 + T^T D + DT + T^T T) \leq 1 + \varepsilon_{\mathcal{H}} \Rightarrow \lambda_{\max}(T^T D + DT + T^T T) \leq 1 - \delta^2 + \varepsilon_{\mathcal{H}}.$$

On the other hand,

$$\lambda_{\min}(T^T D + DT + T^T T) \geq \lambda_{\min}(T^T D + DT) \geq -(r-1)\lambda_{\max}(T^T D + DT) \geq -(r-1)(1 - \delta^2 + \varepsilon_{\mathcal{H}}),$$

where the second inequality uses that the trace of $T^T D + DT$ is zero and hence its eigenvalues sum up to zero. In summary,

$$\|T^T D + DT + T^T T\|_2 \leq (r-1)(1 - \delta^2 + \varepsilon_{\mathcal{H}}),$$

and $\Pi_{<\mu}(C, C) = D^2 + \tilde{E}$ with $\|\tilde{E}\|_2 \leq (r-1)(1 - \delta^2) + r\varepsilon_{\mathcal{H}}$. This completes the proof because

$$\begin{aligned}\|\Pi_{<\mu}(C, C)^{-1}\|_2 &\leq \|D^{-2}\|_2\|(I + D^{-2}\tilde{E})^{-1}\|_2 \leq \frac{1}{\delta^2 - (r-1)(1-\delta^2) - r\varepsilon_{\mathcal{H}}} \\ &= \frac{1}{r}\frac{1}{\delta^2 - 1 + 1/r - \varepsilon_{\mathcal{H}}},\end{aligned}$$

where the inverse exists under the conditions of the lemma. □

The following theorem shows that the columns $\Pi_{<\mu}(:, C)$ selected by Algorithm 3 have an excellent condition number if δ is sufficiently close to one and the perturbations introduced by the HODLR arithmetic remain small.

Theorem 5. *Let C denote the set of r indices returned by Algorithm 3 and suppose that the conditions (9) and (10) as well as the condition of Lemma 4 are satisfied. Then it holds for the 2-norm condition number of $\Pi_{<\mu}(:, C)$ that*

$$\kappa(\Pi_{<\mu}(:, C)) \leq \frac{1}{r}\frac{1 + \varepsilon_{\mathcal{H}}}{\delta^2 - 1 + 1/r - \varepsilon_{\mathcal{H}}} = 1 + 2r(1-\delta) + \mathcal{O}((1-\delta)^2 + \varepsilon_{\mathcal{H}}).$$

Proof. By definition, $\kappa(\Pi_{<\mu}(:, C)) = \|\Pi_{<\mu}(:, C)\|_2\|\Pi_{<\mu}(:, C)^\dagger\|_2$. From (11) we get

$$\|\Pi_{<\mu}(:, C)\|_2 \leq \|\Pi_\mu\|_2 \leq 1 + \|F\|_2. \tag{12}$$

To bound the second factor, we note that $\|\Pi_{<\mu}(:, C)^\dagger\|_2 \leq \|\Pi_{<\mu}(C, C)^{-1}\|_2$ and apply Lemma 4. Using the two bounds, the statement follows. □

The condition of Lemma 4, $1 - \delta^2 \lesssim 1/r$, requires δ to be very close to 1. We conjecture that this condition can be improved to a distance that is proportional to $1/\log_2 r$ or even a constant independent of r. The latter is what we observe in the numerical experiments; choosing δ constant and letting r grow does not lead to a deterioration of the condition number.

4.2 Range correction

As earlier, let C denote a set of indices obtained by Algorithm 3 for a threshold δ, and $r = |C|$. We recall that the dimension of the column space of $\Pi_{<\mu}$ can be easily computed knowing that $\operatorname{trace}(\Pi_{<\mu}) = \operatorname{rank}(\Pi_{<\mu}) = \nu$. If $r = \nu$, then it only remains to perform the orthogonalization to get an orthonormal basis of $\operatorname{Range}(\Pi_{<\mu})$. However, depending on the choice of δ, in practice it can occur that the cardinality of C is smaller than ν, which implies that the selected columns cannot span the column space of $\Pi_{<\mu}$. In this case additional vectors need to be computed to get a complete orthonormal basis for $\operatorname{Range}(\Pi_{<\mu})$.

An orthonormal basis for $\operatorname{Range}(\Pi_{<\mu}(:,C))$ in the HODLR format can be computed using a method suggested in [18], and it is given as

$$\Pi_{<\mu}(:,C) *_{\mathcal{H}} \tilde{R}^{-1}. \tag{13}$$

The biggest disadvantage of the method in [18] is the loss the orthogonality in badly conditioned problems, caused by squaring of the condition number when computing $\tilde{R}$. However, choosing only well-conditioned subset of columns of $\Pi_{<\mu}$ allows us to avoid dealing with badly conditioned problems, and thus prevents potential loss of orthogonality in (13).

In case $r < \nu$, we complete the basis (13) to an orthonormal basis for $\operatorname{Range}(\Pi_{<\mu})$, by computing an orthonormal basis of the orthogonal complement of $\operatorname{Range}(\Pi_{<\mu}(:,C))$ in $\operatorname{Range}(\Pi_{<\mu})$. First we detect the orthogonal complement of $\operatorname{Range}(\Pi_{<\mu}(:,C))$ in $\operatorname{Range}(\Pi_{<\mu})$.

Lemma 6. *If* $(\operatorname{Range}(\Pi_{<\mu}(:,C)))^{\perp}$ *is the orthogonal complement of* $\operatorname{Range}(\Pi_{<\mu}(:,C))$*, then*

$$R^{\perp}_{\Pi_{<\mu},C} := (\operatorname{Range}(\Pi_{<\mu}(:,C)))^{\perp} \cap \operatorname{Range}(\Pi_{<\mu})$$

is the orthogonal complement of $\operatorname{Range}(\Pi_{<\mu}(:,C))$ *in the vector space* $\operatorname{Range}(\Pi_{<\mu})$*. Moreover,* $\dim(R^{\perp}_{\Pi_{<\mu},C}) = \operatorname{rank}(\Pi_{<\mu}) - r$.

Proof. The statements follow directly from the definition of $R^{\perp}_{\Pi_{<\mu},C}$. □

Using (13) we construct an orthogonal projector

$$P_{C^{\perp}} = I - \Pi_{<\mu}(:,C) *_{\mathcal{H}} \tilde{R}^{-1} *_{\mathcal{H}} (\Pi_{<\mu}(:,C) *_{\mathcal{H}} \tilde{R}^{-1})^{T} \tag{14}$$

onto $(\operatorname{Range}(\Pi_{<\mu}(:,C)))^{\perp}$. From (13) it steadily follows that $\operatorname{Range}(P_{C^{\perp}}\Pi_{<\mu}) = R^{\perp}_{\Pi_{<\mu},C}$. Thus computing an orthonormal basis for $P_{C^{\perp}}\Pi_{<\mu}$ will allow us to obtain a complete orthonormal basis for $\operatorname{Range}(\Pi_{<\mu})$.

To this end, we employ a randomized algorithm [15] to compute an orthonormal basis of dimension $\nu - r$ for $\operatorname{Range}(P_{C^{\perp}}\Pi_{<\mu})$: (1) we first multiply $P_{C^{\perp}}\Pi_{<\mu}$ with a random matrix $X \in \mathbb{R}^{n\times(\nu-r+p)}$, where p is an oversampling parameter; (2) we compute its QR decomposition. As singular values of $\Pi_{<\mu}$ are either unity or zero, multiplication with the orthogonal projector $P_{C^{\perp}}$, generated by the linearly independent columns C, gives a matrix whose singular values are well-separated as well. In particular, the resulting matrix has $\nu - r$ singular values equal to 1, and the others equal to zero. Indeed, in exact arithmetics $P_{C^{\perp}}\Pi_{<\mu}$ has the exact rank $\nu - r$, and then oversampling is not required [15]. However, due to the formatted arithmetics, we use a small oversampling parameter p to improve accuracy. As we require only $\nu - r$ columns to complete the basis for $\operatorname{Range}(\Pi_{<\mu})$, finally we keep only the first $\nu - r$ columns of the orthonormal factor.

A pseudo-code for computing a complete orthonormal basis for $\operatorname{Range}(\Pi_{<\mu})$ is given in Algorithm 4. Note that $\Pi_{<\mu}(:,C)$ is a rectangular HODLR matrix, obtained by extracting columns with indices C of a HODLR matrix, as explained in section 3. This implies that the complexity of operations stated in Table 1 carries over for the operations involving HODLR matrices in

Algorithm 4. The complexity of the algorithm also depends on the number of the missing basis vectors. However, in our experiments we observe that $\nu - r$ is very small with respect to ν and n for choice of δ we use, which makes the cost of operations in Line 3 and Line 4 negligible. In the setup when $\nu \approx n/2$, the overall complexity of Algorithm 4 is governed by solving a triangular system in Line 5 or Line 7, i.e. it is $\mathcal{O}(k^2 n \log^2 n)$.

Algorithm 4 Computation of a complete orthonormal basis for Range($\Pi_{<\mu}$)

Input: Spectral projector $\Pi_{<\mu} \in \mathbb{R}^{n\times n}$ in the HODLR format with rank($\Pi_{<\mu}$) $= \nu$, column indices C and the Cholesky factor $\tilde{R}$ returned by Algorithm 3, an oversampling parameter p.
Output: Orthonormal matrix $Q_{<\mu} \in \mathbb{R}^{n\times\nu}$ such that Range($Q_{<\mu}$) = Range($\Pi_{<\mu}$).

```
1: if $|C| < \nu$ then
2:     Generate a random matrix $X \in \mathbb{R}^{n\times(\nu-r+p)}$, for $r = |C|$.
3:     $Z = \Pi_{<\mu}X - \Pi_{<\mu}(:,C)(\tilde{R}^{-1}(\tilde{R}^{-T}(\Pi_{<\mu}(C,:)(\Pi_{<\mu}X))))$.
4:     Compute $[Q_c, \sim, \sim] = \texttt{qr}(Z, 0)$.
5:     Return $Q_{<\mu} = [\Pi_{<\mu}(:,C) *_{\mathcal{H}} \tilde{R}^{-1} \quad Q_c(:, 1:\nu - r)]$.
6: else
7:     Return $Q_{<\mu} = [\Pi_{<\mu}(:,C) *_{\mathcal{H}} \tilde{R}^{-1}]$.
8: end if
```

4.2.1 Storing additional columns

When range correction is performed, we additionally need to store tall-and-skinny matrix Q_c from Algorithm 4. The idea is to incorporate columns of Q_c into an existing HODLR matrix $\Pi_{<\mu}(:,C) *_{\mathcal{H}} \tilde{R}^{-1}$ of size $n \times r$ to get a HODLR matrix of size $n \times \nu$. More specifically, we append $\nu - r$ columns after the last column of $\Pi_{<\mu}(:,C) *_{\mathcal{H}} \tilde{R}^{-1}$, by enlarging all blocks of $\Pi_{<\mu}(:,C) *_{\mathcal{H}} \tilde{R}^{-1}$ that contain the last column. Recompression is performed when updating the off-diagonal blocks. It is expected that the off-diagonal ranks in the updated blocks grow, however, numerical experiments in section 6 demonstrate that the increase is not significant.

5 Divide-and-conquer method in the HODLR format

In this section we give the overall spectral divide-and-conquer method for computing the eigenvalue decomposition of a symmetric banded matrix $A \in \mathbb{R}^{n\times n}$. For completeness, we also include a pseudocode given in Algorithm 5. In the following we discuss several details related to its implementation and provide the structure of the eigenvectors matrix.

5.1 Computing the shift

The purpose of computing shift μ is to split a problem of size n into a two smaller subproblems of roughly the same size. In this work, the computation of a shift is performed by computing the median of diag(A), as proposed in [22]. Although this way of estimating the median of eigenvalues may not be optimal, it is a cheap method and gives reasonably good results. For more details regarding the shift computation, we refer the reader to a discussion in [22]. Moreover, we note that it remains an open problem to develop a better strategy for splitting the spectrum.

Algorithm 5 Spectral divide-and-conquer algorithm in the HODLR format (`hSDC`)

Input: A symmetric banded or HODLR matrix $A \in \mathbb{R}^{n\times n}$.
Output: A structured matrix Q containing the eigenvectors of A and a diagonal matrix Λ containing the eigenvalues of A.

```
1:  function $[Q, \Lambda]$ =hsdc($A$)
2:      if $n \leq n_{\text{stop}}$ then
3:          $[Q, \Lambda] =$ eig($A$).
4:      else
5:          Compute $\mu =$ median(diag($A$)).
6:          Compute $\Pi_{<\mu}$ and $\Pi_{>\mu}$ in the HODLR format by applying Algorithm 2 to $A - \mu I$.
7:          Compute column indices $C_{<\mu}$ and $C_{>\mu}$ by applying Algorithm 3 to $\Pi_{<\mu}$ and $\Pi_{>\mu}$.
8:          Compute $Q_{<\mu}$ and $Q_{>\mu}$ by applying Algorithm 4 to $\Pi_{<\mu}, C_{<\mu}$, and $\Pi_{>\mu}, C_{>\mu}$.
9:          Form $A_{<\mu} = Q_{<\mu}^T *_{\mathcal{H}} A *_{\mathcal{H}} Q_{<\mu}$ and $A_{>\mu} = Q_{>\mu}^T *_{\mathcal{H}} A *_{\mathcal{H}} Q_{>\mu}$.
10:         Call recursively $[Q_1, \Lambda_1] =$ hsdc($A_{<\mu}$) and $[Q_2, \Lambda_2] =$ hsdc($A_{>\mu}$).
11:         Set $Q \leftarrow \begin{bmatrix} Q_{<\mu} & Q_{>\mu} \end{bmatrix} *_{\mathcal{H}} \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}$ and $\Lambda = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}$.
12:     end if
13: end function
```

5.2 Terminating the recursion

We stop the recursion when the matrix attains the minimal prescribed size n_{stop}, and use MATLAB built-in function `eig` to perform the final step of diagonalization. In a practical implementation, we set n_{stop} depending on the breakeven point of `hQDWH` relative to `eig` obtained in [17].

5.3 Matrix of eigenvectors

For simplicity, without loss of generality we assume that for size of a given matrix A holds $n = 2^s n_{\text{stop}}$, for $s \in \mathbb{N}$. We say that Algorithm 5 performed level l divide step, with $0 \leq l < s$, if all matrices of size $n/2^l$ had been subdivided.

The eigenvectors matrix is given as an implicit product of orthonormal HODLR matrices. After level l divide step of Algorithm 5, structured matrix Q has the form

$$Q = Q^{(0)} *_{\mathcal{H}} Q^{(1)} *_{\mathcal{H}} \cdots *_{\mathcal{H}} Q^{(l)}.$$

$Q^{(i)} \in \mathbb{R}^{n\times n}$, $0 \leq i \leq l$, is a block-diagonal matrix with 2^i diagonal blocks, where each diagonal block is an orthogonal matrix of the form $[H_1\, H_2]$, with H_1, H_2 orthonormal HODLR matrices computed in Line 8 of Algorithm 5. The computation of the eigenvectors matrix is completed by computing $Q^{(s)}$, a block-diagonal orthogonal matrix with 2^s orthogonal dense diagonal blocks that are computed in Line 3 of Algorithm 5.

The overall storage required to store Q equals to the sum of memory requirements for matrices $Q^{(i)}, 0 \leq i \leq s$. Assume that the off-diagonal ranks occurring in matrices $Q^{(i)}$, $0 \leq i < s$, are bounded by $\tilde{k}$. To determine the storage, we use that $Q^{(i)}$, for $0 \leq i < s$, has 2^i diagonal blocks of the form $[H_1\, H_2]$, where the storage of both H_1 and H_2 requires $\mathcal{O}(\tilde{k}\frac{n}{2^i}\log_2\frac{n}{2^i})$ memory. Hence we get that the storage for matrices $Q^{(i)}$, $0 \leq i < s$, adds up to

$$\sum_{l=0}^{s-1} \tilde{k} 2^{l+1} \frac{n}{2^l} \log_2 \frac{n}{2^l} = 2\tilde{k}n \sum_{l=0}^{s-1} \log_2 \frac{n}{2^l} = 2\tilde{k}n \left(s \log_2 n - \frac{(s-1)s}{2} \right)$$
$$= \tilde{k}n \log_2^2 \frac{n}{n_{\text{stop}}} + \log_2 \frac{n}{n_{\text{stop}}} (\log_2 n_{\text{stop}} + 1/2). \tag{15}$$

Moreover, the storage of $Q^{(s)}$ requires $2^s n_{\text{stop}}^2 = nn_{\text{stop}}$ units of memory. Hence, from the latter and (15) follows that the overall memory needed for storing Q is $\mathcal{O}(\tilde{k}n \log^2 n)$.

5.4 Computational complexity

Now we derive the theoretical complexity of Algorithm 5, based on the complexity of operations given in Table 1. The numerical results in section 6 give an insight how the algorithm behaves in practice, and confirm theoretical results.

We first note that for a HODLR matrix of size m and rank k the complexity of one divide step, computed in Line 5–Line 9, is $\mathcal{O}(k^2 m \log_2^2 m)$. When performing level l divide step, the computation involves 2^l HODLR matrices of size $n/2^l \times n/2^l$. Denoting with $\tilde{k}$ an upper bound for the off-diagonal ranks appearing in the process, similarly as in the previous section we derive the complexity of our algorithm:

$$\sum_{l=0}^{s-1} \tilde{k}^2 2^l \frac{n}{2^l} \log_2^2 \frac{n}{2^l} = \tilde{k}^2 n \sum_{l=0}^{s-1} \log_2^2 \frac{n}{2^l} = \tilde{k}^2 n \left(s \log_2 n (\log_2 n_{\text{stop}} + 1) + \frac{(s-1)s(2s-1)}{6} \right)$$
$$\approx \mathcal{O}(\tilde{k}^2 n \log_2^3 n).$$

At the final level of recursive application of Algorithm 5, when the algorithm is applied to matrices of size not larger than n_{stop}, the complexity comes from diagonalizing 2^s dense matrices, i.e., equals to $\mathcal{O}(nn_{\text{stop}}^2)$. Thus the overall complexity of Algorithm 5 is $\mathcal{O}(\tilde{k}^2 n \log^3 n)$.

6 Numerical experiments

In this section, we show the performance of our MATLAB implementation of the spectral divide-and-conquer method in the HODLR format for various matrices. All computations were performed in MATLAB version R2016b on a machine with the dual Intel Core i7-5600U 2.60GHz CPU, 256 KByte of level 2 cache and 12 GByte of RAM.

In order to draw a fair comparison with respect to highly optimized MATLAB built-in functions, all experiments were carried out on a single core. The memory requirements shown in Example 10 are obtained experimentally, using MATLAB built-in functions.

In all experiments, we set the truncation tolerance to $\epsilon = 10^{-10}$, the minimal block-size $n_{\min} = 250$ for tridiagonal matrices and $n_{\min} = 500$ for b-banded matrices with $b > 1$. Moreover, the stopping tolerance in the `hDWH` algorithm is set to $\varepsilon = 10^{-15}$. In Algorithm 4 we use the oversampling parameter $p = 10$. We use breakeven points in [17] to set the termination criterion in Algorithm 5. For tridiagonal matrices we use $n_{\text{stop}} = 3250$, for 2-banded matrices $n_{\text{stop}} = 1750$ and $n_{\text{stop}} = 2500$ for b-banded with $b > 2$.

The efficiency of our algorithm is tested on a set of matrices coming from applications, as well as on various synthetic matrices.

6.1 Generation of test matrices

To generate synthetic matrices, we employ the procedure explained in [17, section 6] that uses a sequence of Givens rotations to obtain a symmetric banded matrix with a prescribed bandwidth and spectrum, starting from a diagonal matrix containing n eigenvalues. As the accuracy of computed spectral projectors depends on the relative spectral gap, we generate matrices such that `gap` is constant whenever the spectrum is split in half. We generate such a spectrum by first dividing the interval $[-1, 1]$ into $[-1, -\texttt{gap}] \cup [\texttt{gap}, 1]$ and then recursively applying the same procedure to both subintervals. In particular, interval $[c, d]$ is split into $[c, \frac{c+d}{2} - \frac{d-c}{2}\,\texttt{gap}] \cup [\frac{c+d}{2} + \frac{d-c}{2}\,\texttt{gap}, d]$. The recursive division stops when the number of subintervals is $\leq n/n_{\text{stop}}$. To each subinterval we assign equal number of eigenvalues coming from a uniform distribution. We observed similar results for eigenvalues coming from a geometric distribution, but we omit them to avoid redundancy.

Example 7 (**Percentage and conditioning of selected columns**)**.** We first investigate the percentage of selected columns throughout Algorithm 5 depending on a given threshold δ, together with the condition number of the selected columns. We show results for matrices of size $n = 10240$, with bandwidths $b = 1$ and $b = 8$, and spectral gaps $\texttt{gap} = 10^{-2}$ and $\texttt{gap} = 10^{-6}$, generated as described above. In this example we ensure that in all divide steps of Algorithm 5 the gap between separated parts of the spectrum corresponds to `gap`, by computing the shift μ as the median of eigenvalues of a considered matrix. In each divide step in Algorithm 5 we compute the percentage of selected columns, and finally we show their average for each δ. Moreover, we present a maximal condition number of the selected columns in the whole divide-and-conquer process for a given δ. As expected, smaller values of δ lead to a higher percentage of selected columns, but this leads to a higher condition number as well. Figure 3 and Figure 4 show that already for $\delta \geq 0.4$ we get a good trade-off between the percentage of selected columns and the condition number. This also implies that the off-diagonal ranks in the eigenvectors matrix remain low.

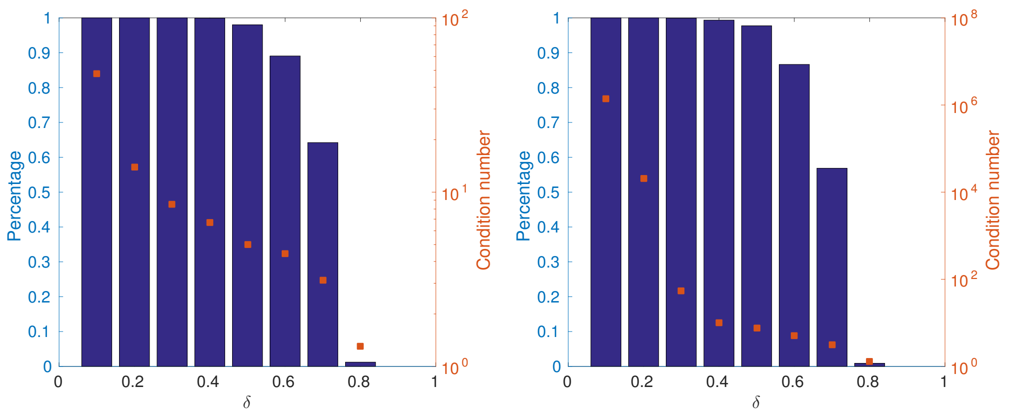

Figure 3: Example 7. Percentage of selected columns and their condition number for a tridiagonal matrix with eigenvalues in $[-1, 1]$ *with relative spectral gap* $\texttt{gap} = 10^{-2}$ *(left) and* $\texttt{gap} = 10^{-6}$ *(right).*

Example 8 (**Breakeven point relative to `eig`**)**.** Our runtime comparisons are performed on generated $n \times n$ banded matrices. We examine for which values of n Algorithm 5 outperforms `eig`. In Table 3 we show breakeven points for banded matrices constructed as in section 6.1,

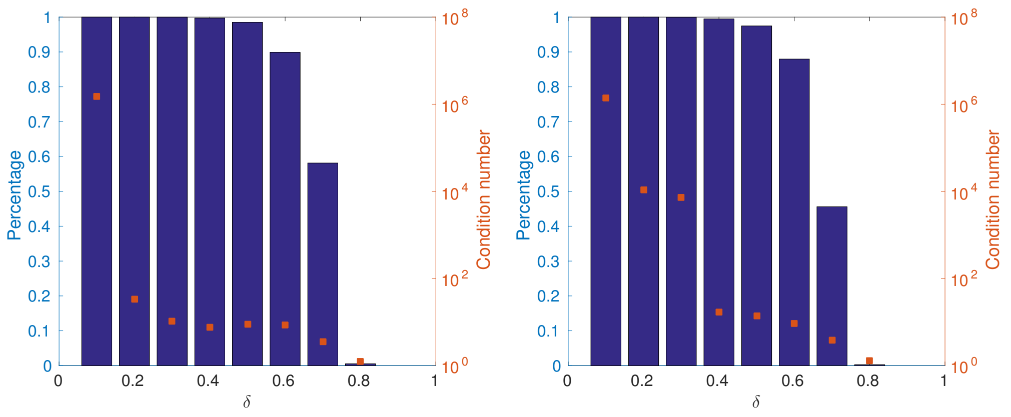

Figure 4: Example 7. Percentage of selected columns and their condition number for a 8-banded matrix with eigenvalues in $[-1,1]$ *with relative spectral gap* $\texttt{gap} = 10^{-2}$ *(left) and* $\texttt{gap} = 10^{-6}$ *(right).*

with `gap` $\in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. We use the threshold parameter $\delta = 0.4$. For $b = 2$ and $b = 4$ banded matrices our algorithm becomes faster than `eig` for relatively small n. This is due to the fact that MATLAB's `eig` first performs tridiagonal reduction. Our results show the benefit of avoiding the reduction of a banded matrix to a tridiagonal form, especially when the bandwidth is small. However, for tridiagonal matrices the breakeven point is relatively high.

Table 3: Breakeven point of hSDC relative to `eig` *applied for banded matrices with various bandwidths and spectral gaps.*

b / gap	1	2	4	8
10^{-1}	$n = 10000$	$n = 2100$	$n = 3000$	$n = 5300$
10^{-2}	$n = 14000$	$n = 2500$	$n = 3800$	$n = 7400$
10^{-3}	$n = 16000$	$n = 2800$	$n = 4900$	$n = 8000$
10^{-4}	$n = 17000$	$n = 3000$	$n = 5200$	$n = 8600$

Example 9 (**Accuracy for various matrices**). In this example, we test the accuracy of the computed spectral decomposition. Denoting with $Q = [q_1, \dots, q_n]$ and $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ the output of Algorithm 5, and with $\tilde{Q} = [\tilde{q}_1, \dots, \tilde{q}_n]$ and $\tilde{\Lambda} = \operatorname{diag}(\tilde{\lambda}_1, \dots, \tilde{\lambda}_n)$ the eigenvalue decomposition obtained using MATLAB's `eig`, we consider four different error metrics:

- the largest relative error in the computed eigenvalues: $e_\lambda = \max_i |\lambda_i - \tilde{\lambda}_i| / \|A\|_2$,
- the largest relative residual norm: $e_{\text{res}} = \max_i \|Aq_i - \lambda_i q_i\|_2 / \|A\|_2$,
- the loss of orthogonality: $e_{\text{orth}} = \max_i \|Q^T q_i - e_i\|_2$,
- the largest error in the computed eigenvectors : $e_Q = \max_i |1 - \cos \measuredangle(q_i, \tilde{q}_i)|$.

In the subsequent experiments we set $\delta = 0.4$.

1. First we show the accuracy of the newly proposed algorithm for tridiagonal matrices. For matrices of size smaller than 3250, we use $n_{\text{stop}} = 500$, which allows us to perform at least one divide step in Algorithm 5. We consider some of the matrices suggested in [19]:

   - the BCSSTRUC1 set in the Harwell-Boeing Collection [8]. Considered problems are in fact generalized eigenvalue problems, with M a mass matrix and K a stiffnes matrix. Each problem is transformed into an equivalent standard eigenvalue problem $L^{-1}KL^{-T}x = \lambda x$, where L denotes the Cholesky factor of M. Finally, matrices are reduced to tridiagonal form via Matlab function `hess`.
   - The symmetric Alemdar and Cannizzo matrices, and matrices from the NASA set [8]. Considered matrices are reduced to tridiagonal form using Matlab function `hess`.
   - The $(1, 2, 1)$ symmetric tridiagonal Toeplitz matrix.
   - The Legendre-type tridiagonal matrix.
   - The Laguerre-type tridiagonal matrix.
   - The Hermite-type tridiagonal matrix.
   - Symmetric tridiagonal matrices with eigenvalues coming from a random $(0, 1)$ distribution and a uniform distribution on $[-1, 1]$.

   In Table 4 we report the observed accuracies. The results are satisfactory, and the errors are roughly of order of the truncation tolerance $\epsilon = 10^{-10}$. We also mention that the percentage of selected columns, as well as the condition number of selected columns throughout Algorithm 5 were along the lines the results presented in Example 7.

Table 4: Accuracy of hSDC for tridiagonal matrices from Example 9.

	matrix	n	e_λ	e_{res}	e_{orth}	e_Q
BCSSTRUC1	bcsst08	1074	$4.4 \cdot 10^{-14}$	$2.2 \cdot 10^{-12}$	$5.6 \cdot 10^{-10}$	$5.6 \cdot 10^{-12}$
	bcsst09	1083	$5.2 \cdot 10^{-11}$	$2.3 \cdot 10^{-11}$	$7.8 \cdot 10^{-10}$	$6.3 \cdot 10^{-12}$
	bcsst11	1474	$1.8 \cdot 10^{-11}$	$1.4 \cdot 10^{-10}$	$2.6 \cdot 10^{-9}$	$7.3 \cdot 10^{-11}$
NASA	nasa1824	1824	$3.2 \cdot 10^{-12}$	$6.8 \cdot 10^{-9}$	$2.5 \cdot 10^{-9}$	$1.5 \cdot 10^{-10}$
	nasa2146	2146	$1.5 \cdot 10^{-10}$	$2.9 \cdot 10^{-9}$	$1.8 \cdot 10^{-9}$	$7.1 \cdot 10^{-11}$
	nasa2190	2190	$1.5 \cdot 10^{-12}$	$2.1 \cdot 10^{-12}$	$6.7 \cdot 10^{-10}$	$4.1 \cdot 10^{-11}$
	nasa4704	4704	$8.9 \cdot 10^{-12}$	$2.9 \cdot 10^{-10}$	$9.8 \cdot 10^{-9}$	$5.3 \cdot 10^{-10}$
	Cannizzo matrix	4098	$3.4 \cdot 10^{-11}$	$8.3 \cdot 10^{-10}$	$1.5 \cdot 10^{-9}$	$2.6 \cdot 10^{-10}$
	Alemdar matrix	6245	$1.25 \cdot 10^{-9}$	$7.4 \cdot 10^{-8}$	$2.5 \cdot 10^{-9}$	$2.8 \cdot 10^{-10}$
	(1,2,1) matrix	10000	$2.4 \cdot 10^{-10}$	$1.6 \cdot 10^{-9}$	$2.5 \cdot 10^{-11}$	$1.3 \cdot 10^{-11}$
	Clement-type	10000	$1.75 \cdot 10^{-10}$	$2.3 \cdot 10^{-9}$	$2.6 \cdot 10^{-9}$	$1.4 \cdot 10^{-10}$
	Legendre-type	10000	$2.4 \cdot 10^{-11}$	$2.7 \cdot 10^{-10}$	$9.6 \cdot 10^{-11}$	$2.7 \cdot 10^{-11}$
	Laguerre-type	10000	$3.4 \cdot 10^{-11}$	$1.9 \cdot 10^{-10}$	$3.7 \cdot 10^{-9}$	$4.3 \cdot 10^{-11}$
	Hermite-type	10000	$9.8 \cdot 10^{-11}$	$3.1 \cdot 10^{-9}$	$9.3 \cdot 10^{-10}$	$3.2 \cdot 10^{-11}$
	Random normal $(0, 1)$	10000	$6.9 \cdot 10^{-10}$	$4.5 \cdot 10^{-8}$	$7.5 \cdot 10^{-9}$	$1.1 \cdot 10^{-10}$
	Uniform in $[-1, 1]$	10000	$1.1 \cdot 10^{-11}$	$2.1 \cdot 10^{-9}$	$3.5 \cdot 10^{-10}$	$1.8 \cdot 10^{-11}$

2. Now we examine the dependency of the error measures on the decreasing spectral gap. We construct 8-banded matrices of size $n = 10240$ with $\texttt{gap} = 10^{-i}, i = 1, \ldots, 10$ using a method from section 6.1. As in Example 7, we ensure that the gap between two separated parts of spectrum in all divide steps of Algorithm 5 is the prescribed `gap`. Figure 5 shows

that our algorithm preforms well for matrices with larger spectral gaps, and confirms the expected behaviour of errors. The error growth is a result of the decreasing accuracy when computing spectral projectors associated with decreasing spectral gaps. For gaps of order $\leq 10^{-9}$ the algorithm breaks down due to indefinitness of a matrix in Line 9 of Algorithm 2.

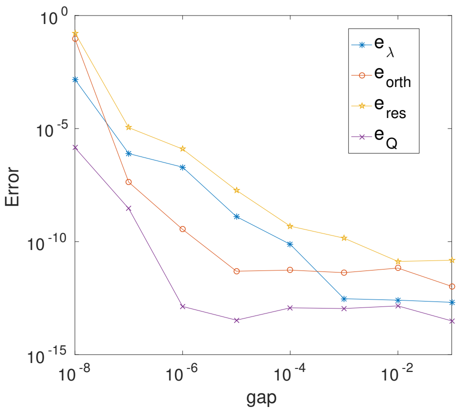

Figure 5: Example 9. Behaviour of the errors with respect to a decreasing spectral gap for banded matrices.

Example 10 (Scalability). For $n \times n$ tridiagonal matrices, generated as in section 6.1 with $\mathsf{gap} = 10^{-2}$, we demonstrate the performance of our algorithm with respect to n. Again we use the threshold parameter $\delta = 0.4$. We show that the asymptotic behaviour of our algorithm matches the theoretical bounds both for the computational time and storage requirements. Figure 6 (left) shows that time needed to compute the complete spectral decomposition follows the expected $\mathcal{O}(n \log^3(n))$ reference line, whereas Figure 6 (right) demonstrates that the memory required to store the matrix of eigenvectors is $\mathcal{O}(n \log^2(n))$.

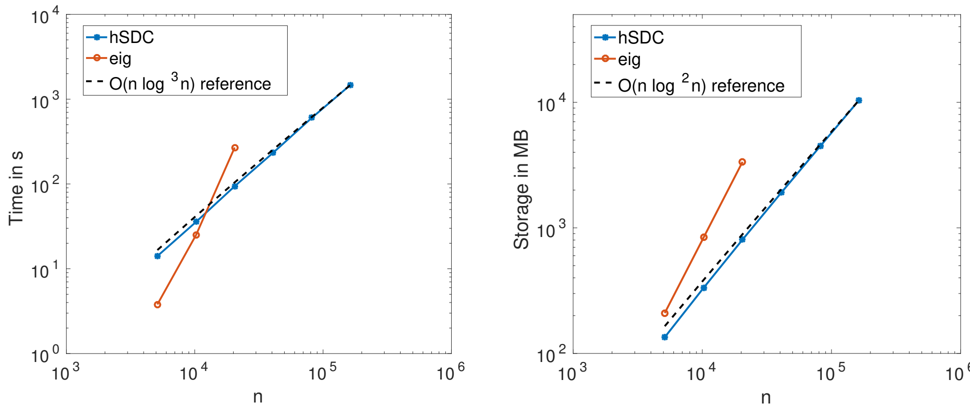

Figure 6: Example 10. Performance of the hSDC algorithm with respect to n for tridiagonal matrices. Left: Computational time. Right: Memory requirements.

7 Conclusion

In this work we have proposed a new fast spectral divide-and-conquer algorithm for computing the complete spectral decomposition of symmetric banded matrices. The algorithm exploits the fact that spectral projectors of banded matrices can be efficiently computed and stored in the HODLR format. We have presented a fast novel method for selecting well-conditioned columns of a spectral projector based on a Cholesky decomposition with pivoting, and provided a theoretical justification for the method. This method enables us to efficiently split the computation of the spectral decomposition of a symmetric HODLR matrix into two smaller subproblems.

The new spectral D&C method is implemented in the HODLR format and has a linear-polylogarithmic complexity. In the numerical experiments, performed both on synthetic matrices and matrices coming from applications, we have verified the efficiency of our method, and have shown that it is a competitive alternative to the state-of-the-art methods for some classes of banded matrices.

Acknowledgements. The authors would like to thank Stefano Massei for helpful discussions on this paper.

References

[1] S. Ambikasaran and E. Darve. An $\mathcal{O}(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices: with application to radial basis function interpolation. *J. Sci. Comput.*, 57(3):477–501, 2013.

[2] P. Arbenz. Divide and conquer algorithms for the bandsymmetric eigenvalue problem. *Parallel Comput.*, 18(10):1105–1128, 1992.

[3] T. Auckenthaler, V. Blum, H.-J. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, and P. R. Willems. Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations. *Parallel Computing*, 37(12):783–794, 2011.

[4] J. Ballani and D. Kressner. *Matrices with hierarchical low-rank structure.* CIME Summer School on Exploiting Hidden Structure in Matrix Computations, 2016.

[5] P. Bientinesi, F. D Igual, D. Kressner, M. Petschow, and E. S. Quintana-Ortí. Condensed forms for the symmetric eigenvalue problem on multi-threaded architectures. *Concurrency and Computation: Practice and Experience*, 23(7):694–707, 2011.

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

[7] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36(2):177–195, 1980/81.

[8] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Software*, 38(1):1–25, 2011.

[9] G. H. Golub and C. F. Van Loan. *Matrix computations.* Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013.

[10] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix Anal. Appl.*, 16(1):172–191, 1995.

[11] W. Hackbusch. *Hierarchical matrices: algorithms and analysis*, volume 49 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2015.

[12] A. Haidar, H. Ltaief, and J. Dongarra. Parallel Reduction to Condensed Forms for Symmetric Eigenvalue Problems Using Aggregated Fine-grained and Memory-aware Kernels. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 8:1–8:11. ACM, 2011.

[13] A. Haidar, H. Ltaief, and J. Dongarra. Toward a high performance tile divide and conquer algorithm for the dense symmetric eigenvalue problem. *SIAM J. Sci. Comput.*, 34(6):C249–C274, 2012.

[14] A. Haidar, R. Solcà, M. Gates, S. Tomov, T. Schulthess, and J. Dongarra. Leading Edge Hybrid Multi-GPU Algorithms for Generalized Eigenproblems in Electronic Structure Calculations. In *Supercomputing*, volume 7905 of *Lecture Notes in Computer Science*, pages 67–80. Springer Berlin Heidelberg, 2013.

[15] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.

[16] N. J. Higham. *Accuracy and stability of numerical algorithms.* SIAM, Philadelphia, PA, 1996.

[17] D. Kressner and A. Šušnjara. Fast computation of spectral projectors of banded batrices. *SIAM J. Matrix Anal. Appl.*, 38(3):984–1009, 2017.

[18] M. Lintner. *Lösung der 2D Wellengleichung mittels hierarchischer Matrizen.* Doctoral thesis, TU München, 2002.

[19] O. A. Marques, C. Vömel, J. W. Demmel, and B. N. Parlett. Algorithm 880: a testing infrastructure for symmetric tridiagonal eigensolvers. *ACM Trans. Math. Software*, 35(1):Art. 8, 13, 2009.

[20] C. B. Moler and G. W. Stewart. On the Householder-Fox algorithm for decomposing a projection. *J. Comput. Phys.*, 28(1):82–91, 1978.

[21] Y. Nakatsukasa, Z. Bai, and F. Gygi. Optimizing Halley's iteration for computing the matrix polar decomposition. *SIAM J. Matrix Anal. Appl.*, 31(5):2700–2720, 2010.

[22] Y. Nakatsukasa and N. J. Higham. Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD. *SIAM J. Sci. Comput.*, 35(3):A1325–A1349, 2013.

[23] E. Solomonik, G. Ballard, J. Demmel, and T. Hoefler. A communication-avoiding parallel algorithm for the symmetric eigenvalue problem. arXiv:1604.03703, 2016.

[24] J. Vogel, J. Xia, S. Cauley, and V. Balakrishnan. Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions. *SIAM J. Sci. Comput.*, 38(3):A1358–A1382, 2016.