COMPUTING APPROXIMATE EIGENPAIRS OF SYMMETRIC BLOCK TRIDIAGONAL MATRICES

WILFRIED N. GANSTERER*§ AND ROBERT C. WARD*§

Technical Report UT-CS-01-463 [1]

University of Tennessee

July 19, 2001

Abstract. A divide-and-conquer method for computing approximate eigenvalues and eigenvectors of a block-tridiagonal matrix is presented. In contrast to a method described earlier [13] the off-diagonal blocks can have arbitrary ranks. It is shown that lower rank approximations of the off-diagonal blocks as well as relaxation of deflation criteria permit the computation of approximate eigenpairs with prescribed accuracy at significantly reduced computational costs compared to standard methods as, for example, implemented in LAPACK.

1. Introduction. We consider the problem of computing approximate eigenvalues and eigenvectors of an irreducible symmetric block tridiagonal matrix

$$M_p := \begin{pmatrix} B_1 & C_1^\top & & & \\ C_1 & B_2 & C_2^\top & & \\ & C_2 & B_3 & \ddots & \\ & & \ddots & \ddots & C_{p-1}^\top \\ & & & C_{p-1} & B_p \end{pmatrix} \in \mathbb{R}^{n\times n} \tag{1.1}$$

with $p > 1$. The blocks $B_i \in \mathbb{R}^{k_i\times k_i}$ $(i = 1, 2, \dots, p)$ along the diagonal are symmetric, and the off-diagonal blocks $C_i \in \mathbb{R}^{k_{i+1}\times k_i}$ $(i = 1, 2, \dots, p-1)$ are arbitrary. The block sizes k_i have to satisfy $1 \le k_i < n$ and $\sum_{i=1}^p k_i = n$, but are otherwise arbitrary.

It should be emphasized that the class of matrices of the form (1.1) comprises *banded* symmetric matrices, a very important type of matrices arising in numerous applications. For banded matrices with upper and lower bandwidth b, a block-tridiagonal structure can be chosen, for example, by setting $k_i = b+1$ for all i with all the subdiagonal blocks C_i being upper triangular. However, other possibilities for imposing a block-tridiagonal structure on a banded matrix exist, which may be more appropriate in some situations.

1.1. Objectives. Given a (variable) accuracy parameter τ, the goal is to find an *approximate spectral decomposition*

$$M_p \approx \hat{V}\hat{\Lambda}\hat{V}^\top. \tag{1.2}$$

The diagonal matrix $\hat{\Lambda}$ contains the approximations $\hat{\lambda}_i$ to the eigenvalues λ_i of M_p and the column vectors $\hat{v}_i$ of $\hat{V}$ are the approximations to the eigenvectors v_i of M_p. The computed approximate eigenpairs $(\hat{\lambda}_i, \hat{v}_i)$ have to satisfy that

*Department of Computer Science, University of Tennessee, 203 Claxton Complex, 1122 Volunteer Blvd., Knoxville, TN 37996-3450

§This work was supported by the Academic Strategic Alliances Program of the Accelerated Strategic Computing Initiative (ASCI/ASAP) under subcontract number B341492 of DOE contract W-7405-ENG-48.

[1]Available from: http://www.cs.utk.edu/~library/TechReports.html

- the residuals are bounded by τ, i. e.,

$$\mathcal{R} := \max_{i=1,2,\ldots,n} \left\| M_p \hat{v}_i - \hat{\lambda}_i \hat{v}_i \right\|_2 \leq \tau; \tag{1.3}$$

- the matrix $\hat{V}$ is numerically orthogonal, i. e.,

$$\mathcal{O} := \max_{i=1,2,\ldots,n} \left\| \left(\hat{V}^\top \hat{V} - I \right) e_i \right\|_2 = O(\varepsilon n), \tag{1.4}$$

  where $e_i \in \mathbb{R}^n$ has components δ_i^j and ε denotes the machine precision (unit roundoff).

Moreover, the method developed for computing (1.2) should have the feature that lower accuracy requirements lead to a higher reduction of the computing time compared to computing the spectral decomposition to full accuracy (only limited by the rounding error and by the condition of the problem).

1.2. Motivation. The *self-consistent-field* (*SCF*) method, which is used for solving the *Hartree-Fock Equations* in Quantum Chemistry [24, ch. 3], involves the full-spectrum solution of a sequence of eigenvalue problems with very large and in general *dense* matrices. It has an *inner-outer iterative* structure. Very low accuracy may be sufficient in early iterations of the SCF-method, and higher accuracy usually becomes more important as it proceeds.

The matrices arising in these eigenvalue problems are in general not block-tridiagonal, but they often have the property that the magnitudes of their elements rapidly decrease when moving away from the diagonal, and therefore they can be approximated by matrices of the form (1.1). Various approaches to performing such a block-tridiagonal approximation of a general matrix will be summarized in a forthcoming paper.

Since the method developed in this paper has a variable accuracy parameter τ, it will be nicely applicable to the SCF method, and, more general, to many other problems with similar properties. Moreover, we anticipate applications in the context of preconditioning (for example, for approximating the spectrum of the inverse of a given matrix).

1.3. Related Work. The standard method for computing eigenpairs of a symmetric band matrix, as, for example, implemented in LAPACK [1], is to tridiagonalize it [22, 19, 5], to compute eigenvalues and eigenvectors of the similar tridiagonal matrix, and finally to transform the eigenvectors.

The standard *divide-and-conquer method* for computing eigenvalues and eigenvectors of a *tridiagonal* symmetric matrix has been developed by Cuppen [7]. The core of this algorithm is a method for efficiently finding the spectral decomposition of a rank-one modification of a diagonal matrix which has been given in [15, 6]. Over time numerically stable and efficient implementations of Cuppen's method were developed [9, 23, 17, 18, 21]. The routines `LAPACK/dsyevd` and `LAPACK/dsbevd` tridiagonalize a given symmetric general or banded matrix, respectively, and, if eigenvectors are desired, then apply a divide-and-conquer algorithm to the resulting tridiagonal matrix.

The divide-and-conquer approach not only has attractive parallelization properties [25, 14], in combination with tridiagonalization it is even sequentially one of the fastest methods currently available if all eigenvalues and eigenvectors of a large dense or banded symmetric matrix are to be computed [8, ch. 5].

In some situations, tridiagonalization of a band matrix can be comparatively expensive *relative* to the total cost of the calculation of eigenpairs [20, ch. 7]. Moreover, unfavorable data access patterns and bad data locality may cause inefficiencies in a tridiagonalization process [10]. This fact motivates attempts to compute eigenpairs of a band matrix without tridiagonalizing the entire matrix. One possible approach seems to be a generalization of the divide-and-conquer method to band matrices. Several variants of such a generalization have been investigated ([2] based on [4, 3]; more recently [11, 12]). One of the central questions remains numerical stability and although promising advances have been made no final method has been established (yet).

A divide-and-conquer method for a special case of (1.1), i. e., for block-tridiagonal matrices with *rank-one* off-diagonal blocks C_i, has been investigated in [13]. The result is a very efficient and numerically stable algorithm for this special problem class. In general, the off-diagonal blocks C_i of the matrices arising in application problems are not rank-one matrices, and there are instances where approximating them with rank-one matrices is not sufficiently accurate. The algorithm discussed in this paper is able to handle off-diagonal blocks with arbitrary ranks and therefore is able to achieve full accuracy.

So far, the divide-and-conquer approach for eigenproblems has been used exclusively for computing *full accuracy* solutions of the symmetric *tridiagonal* eigenproblem. The major innovation of the algorithm proposed here is the idea to investigate the potential of methods based on the divide-and-conquer approach for computing *approximate* eigenpairs of a more *general* class of matrices. Our experiments, summarized in Section 4, indicate that the resulting method is highly competitive compared to other methods for computing eigenpairs of symmetric matrices, especially if *low accuracy* eigenpair approximations are sufficient and if the *full* spectrum of M_p needs to be approximated.

2. Concept. The algorithm presented in this paper involves two main phases:

1. Approximation of M_p by another block-tridiagonal matrix $M'_p \in \mathbb{R}^{n\times n}$, whose off-diagonal blocks C'_i are approximations of the original C_i. This phase is outlined in Section 2.1.
2. Application of a block-tridiagonal divide-and-conquer method to compute eigenvalues and eigenvectors of M'_p. In analogy to Cuppen's tridiagonal divide-and-conquer method, this phase consists of (i) subdivision (see Section 2.2.1), (ii) solution of the subproblems (see Section 2.2.2), and (iii) synthesis of the solutions of the subproblems (see Section 2.2.3). Depending on the accuracy requirements of the application context, the synthesis step may or may not be approximative (see Section 2.3).

Another variant of this algorithm, in which the computation of the eigenvectors is performed separately in a third phase, will be discussed in a forthcoming paper.

2.1. Approximation of the Off-Diagonal Blocks. The natural extension of the algorithm discussed in [13] is to allow for *higher rank approximations* of the off-diagonal blocks C_i. The singular value decompositions (see [16, ch. 2])

$$C_i = \sum_{j=1}^{m_i} \sigma_j^i u_j^i {v_j^i}^\top, \quad i = 1, 2, \ldots, p-1,$$

with $m_i := \min(k_i, k_{i+1})$, $\sigma_1^i \geq \sigma_2^i \geq \ldots \geq \sigma_{m_i}^i \geq 0$ and $\|u_j^i\|_2 = \|v_j^i\|_2 = 1$ for all i and j can be used for constructing approximations C'_i of rank r_i $(1 \leq r_i \leq m_i)$

corresponding to the first r_i (largest) singular values:

$$C_i' := \sum_{j=1}^{r_i} \sigma_j^i u_j^i {v_j^i}^\top = U_i \Sigma_i V_i^\top, \quad i = 1, 2, \ldots, p-1,$$

using the notation $U_i := \left(u_1^i | u_2^i | \ldots | u_{r_i}^i\right) \in \mathbb{R}^{k_{i+1} \times r_i}$, $\Sigma_i := \operatorname{diag}\left(\sigma_1^i, \sigma_2^i, \ldots, \sigma_{r_i}^i\right)$ and $V_i := \left(v_1^i | v_2^i | \ldots | v_{r_i}^i\right) \in \mathbb{R}^{k_i \times r_i}$.

Approximation Error. The rank-r_i approximations of the off-diagonal blocks C_i of M_p result in a matrix

$$M_p' := \begin{pmatrix} B_1 & {C_1'}^\top & & & \\ C_1' & B_2 & {C_2'}^\top & & \\ & C_2' & B_3 & \ddots & \\ & & \ddots & \ddots & {C_{p-1}'}^\top \\ & & & C_{p-1}' & B_p \end{pmatrix} \in \mathbb{R}^{n \times n},$$

which is related to M_p according to

$$M_p = M_p' + E^{(1)},$$

where $E^{(1)}$ is a block-tridiagonal matrix with the entries

$$\left(\mathbf{0}, \sum_{j=r_1+1}^{m_1} \sigma_j^1 v_j^1 {u_j^1}^\top\right)$$

in the first block row, the entries

$$\left(\sum_{j=r_{i-1}+1}^{m_{i-1}} \sigma_j^{i-1} u_j^{i-1} {v_j^{i-1}}^\top, \mathbf{0}, \sum_{j=r_i+1}^{m_i} \sigma_j^i v_j^i {u_j^i}^\top\right)$$

in block rows $i = 2, 3, \ldots, p-1$, and the entries

$$\left(\sum_{j=r_{p-1}+1}^{m_{p-1}} \sigma_j^{p-1} u_j^{p-1} {v_j^{p-1}}^\top, \mathbf{0}\right)$$

in the last (pth) block row.

Invoking Weyl's theorem (see, for example, [8, ch. 5]), we can see that the absolute difference between the eigenvalues λ of M_p and the eigenvalues λ' of M_p' can be bounded according to

$$|\lambda - \lambda'| \leq \|E^{(1)}\|_2.$$

Keeping in mind that for n-vectors $\|\cdot\|_1 \leq \sqrt{n}\|\cdot\|_2$ and that $\|uv^\top\|_1 \leq \|u\|_1 \|v\|_\infty$, we have

$$\|E^{(1)}\|_1 \leq \max_{i=2,3,\ldots,p-1} \left(\left\| \sum_{j=r_1+1}^{m_1} \sigma_j^1 u_j^1 {v_j^1}^\top \right\|_1, \left\| \sum_{j=r_{p-1}+1}^{m_{p-1}} \sigma_j^{p-1} v_j^{p-1} {u_j^{p-1}}^\top \right\|_1, \right.$$

$$\left\| \sum_{j=r_{i-1}+1}^{m_{i-1}} \sigma_j^{i-1} v_j^{i-1} u_j^{i-1\top} \right\|_1 + \left\| \sum_{j=r_i+1}^{m_i} \sigma_j^i u_j^i v_j^{i\top} \right\|_1 \Bigg)$$
$$\leq 2\sqrt{n} \max_{i=1,2,\ldots,p-1} \sum_{j=r_i+1}^{m_i} \sigma_j^i .$$

Since $E^{(1)}$ is symmetric, its 2-norm equals the maximum of the absolute values of its eigenvalues, which is smaller than any matrix norm induced by a vector norm. In particular, it is smaller than its 1-norm. Thus, $\|E^{(1)}\|_2 \leq \|E^{(1)}\|_1$, which leads to the error bound

$$|\lambda - \lambda'| \leq 2\sqrt{n} \max_{i=1,2,\ldots,p-1} \sum_{j=r_i+1}^{m_i} \sigma_j^i =: \tau_1 \tag{2.1}$$

for the eigenvalues λ' of M_p' with respect to the eigenvalues λ of M_p.

2.2. Divide-and-Conquer Solution. The eigenpairs of M_p' can be computed using a divide-and-conquer approach outlined in the following.

2.2.1. Subdivision. With the corrections

$$\begin{aligned} \tilde{B}_1 &:= B_1 - V_1 \Sigma_1 V_1^\top, \\ \tilde{B}_i &:= B_i - U_{i-1}\Sigma_{i-1}U_{i-1}^\top - V_i \Sigma_i V_i^\top, \quad i = 2,3,\ldots,p-1, \\ \tilde{B}_p &:= B_p - U_{p-1}\Sigma_{p-1}U_{p-1}^\top, \end{aligned}$$

M_p' can be represented as a series of rank-r_i modifications of the block-diagonal matrix $\tilde{M}'_p :=$ block-diag $\left(\tilde{B}_1, \tilde{B}_2, \ldots, \tilde{B}_p\right)$:

$$M_p' = \tilde{M}'_p + \sum_{i=1}^{p-1} W_i W_i^\top . \tag{2.2}$$

The matrices $W_i \in \mathbb{R}^{n \times r_i}$ in (2.2) are given as

$$W_1 := \begin{pmatrix} V_1 \Sigma_1^{1/2} \\ U_1 \Sigma_1^{1/2} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \quad W_{p-1} := \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ V_{p-1}\Sigma_{p-1}^{1/2} \\ U_{p-1}\Sigma_{p-1}^{1/2} \end{pmatrix}, \tag{2.3}$$
$$W_i := \begin{pmatrix} \mathbf{0} \\ V_i \Sigma_i^{1/2} \\ U_i \Sigma_i^{1/2} \\ \mathbf{0} \end{pmatrix}, \quad i = 2,3,\ldots,p-2.$$

2.2.2. Solution of the Subproblems. Next, the spectral decompositions

$$\tilde{B}_i = Q_i D_i Q_i^\top, \quad i = 1,2,\ldots,p,$$

of the p diagonal blocks of $\tilde{M}'_p$ have to be computed using the method which is most efficient for the size, structure and sparsity pattern of each matrix $\tilde{B}_i$. In the following,

$$D := \begin{pmatrix} D_1 & & \\ & \ddots & \\ & & D_p \end{pmatrix} \in \mathbb{R}^{n \times n}$$

denotes the diagonal matrix consisting of the eigenvalues of the diagonal blocks,

$$Q := \begin{pmatrix} Q_1 & & \\ & \ddots & \\ & & Q_p \end{pmatrix} \in \mathbb{R}^{n \times n}$$

is a block-diagonal matrix which contains the eigenvector matrices of the diagonal blocks, and thus

$$\tilde{M'}_p = QDQ^\top. \tag{2.4}$$

2.2.3. Synthesis of the Solutions of the Subproblems. Substituting (2.4) into (2.2) and denoting $Y_i := Q^\top W_i$ yields the representation

$$M'_p = Q \left(D + \sum_{i=1}^{p-1} Y_i Y_i^\top \right) Q^\top,$$

which implies that M'_p is orthogonally similar to the *synthesis matrix*

$$S := D + \sum_{i=1}^{p-1} Y_i Y_i^\top. \tag{2.5}$$

Denoting $r := \sum_{i=1}^{p-1} r_i$, the synthesis matrix S is a rank-r modification of a diagonal matrix. Computation of the eigendecomposition of S reveals the eigenvalues of M'_p and its eigenvectors in a factored form.

2.2.4. Eigenpairs of the Synthesis Matrix. There are several approaches for computing eigenvalues and eigenvectors of S.

Arbenz, Gander and Golub [3, 4] have developed a method for performing an eigenanalysis of the entire rank-r modification (2.5) at once. We decided not to use their approach in our context for several reasons:

- Its main advantage is the transformation of an $n \times n$ to an $r \times r$ problem. In our case, r is often not significantly smaller than n, and therefore this problem transformation is not very beneficial.
- It is yet unclear how to generalize deflation, which often leads to significant reductions of the computing time (see Sections 2.3 and 4), for a rank-r modification.
- Unsatisfactory numerical accuracy has been observed by Arbenz [2], in particular, a loss of numerical orthogonality of the computed eigenvectors. A variant which combined divide-and-conquer for the eigenvalue computation with inverse iteration for the eigenvector computation showed improved accuracy, but turned out to be less efficient [2].

Instead, we represent S as a sequence of r rank-*one* modifications of D. In principle, these rank-one modifications can be performed in any order. However, due to the sparsity structure in the modification vectors (the columns of the matrices Y_i, see Fig. 2.1) it is preferable to complete all rank-one modifications corresponding to the same off-diagonal block C_i' (represented by the columns of the matrix Y_i) before starting with a different one. We will refer to the process of performing the r_i rank-one modifications Y_i corresponding to one off-diagonal block C_i' as one *merging operation*, because it accounts for the dependencies represented by the off-diagonal block C_i' and therefore "merges" two diagonal blocks. It will be discussed in Section 3.2 how to determine a good order for performing the individual merging operations.

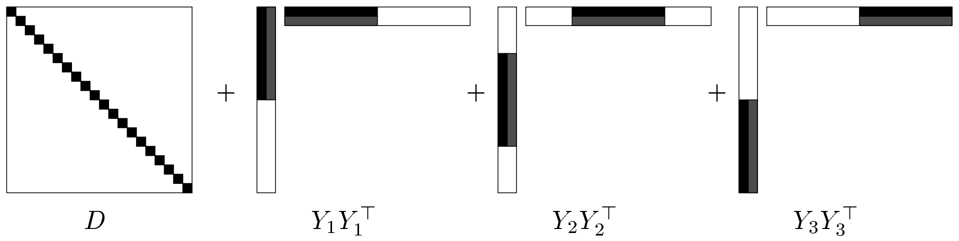

FIG. 2.1. *Sparsity structure of the matrices* Y_i ($n = 20$, $p = 4$, $r_i = 2$ *for* $i = 1, 2, 3$)

A major advantage of our approach is that it is possible to utilize the technology for rank-one modifications developed for the tridiagonal divide-and-conquer method. In particular, the concepts developed in [17] can be utilized in each rank-one modification and therefore numerical stability and numerical orthogonality of the computed eigenvectors can be guaranteed. A potential disadvantage, however, lies in the arithmetic complexity of the eigenvector computation. Accumulation of the r eigenvector matrices for the rank-one modification problems (analogously to the tridiagonal divide-and-conquer method) requires $O(n^3)$ flops in the worst case (see Section 3.1). *Deflation* (see Section 2.3) may, however, significantly reduce the actual flop count. In particular, if only approximate eigenpairs are needed, relaxing the deflation criteria can lead to a significant reduction of computing times (cf. Sections 4.1 and 4.2).

Another algorithmic variant for computing the eigenvectors of S which has the potential of reducing the order of the arithmetic complexity to $O(n^2)$ is currently under investigation and will be discussed in a forthcoming paper.

2.3. Relaxing Deflation. It has been shown in [6, 9] that there are two special situations in which eigenpairs of a rank-one modification problem $D + xx^\top$ with a diagonal matrix D can be found very efficiently:

- If there is a zero component x_i in x then the corresponding entry d_i of D is an eigenvalue and the vector e_i is an eigenvector of $D + xx^\top$.
- If there are two equal entries in D then one of the corresponding components of x can be eliminated using Givens rotations. After this transformation the corresponding eigenpairs are given as in the previous case.

This process is called *deflation*. It not only reduces the problem size for the eigenvalue computation, it also introduces a block structure in the eigenvector matrices which reduces the work required for accumulating them (see Fig. 3.2).

So far, the divide-and-conquer approach has only been used for computing eigenpairs to full accuracy. In this case, only "nearly zero" components of x or "nearly equal" entries of D may be deflated. Typically, the *deflation tolerance* τ_2 is chosen as a moderate multiple of the machine precision ε times the norm of the matrix of the eigenproblem. For example, in LAPACK [1], it is set to

$$\tau_2 = \tau_2^L := 8\varepsilon \max\left(\max_{i=1,2,\ldots,n} |d_i|, \max_{i=1,2,\ldots,n} |x_i| \right).$$

For lower accuracy requirements, as considered in this paper, it is possible to relax the deflation criteria accordingly by increasing the deflation tolerance τ_2 (*relaxed deflation*). This establishes an approximate synthesis step. In most situations the amount of deflation is significantly increased and therefore the computational effort for accumulating the eigenvector matrices of the rank-one modification problems is significantly reduced (cf. Sections 4.1 and 4.2).

Approximation Error. Let the parameters of a Givens rotation used for eliminating a component of a modification vector x corresponding to two entries d_i and d_{i+1} of D for which $|d_i - d_{i+1}| = \delta$ be denoted by γ and σ. It has been shown in [9], that deflating components x_i of modification vectors x if

$$|x_i| \leq \tau_2$$

and considering diagonal entries d_i and d_{i+1} as equal if

$$|\delta\gamma\sigma| \leq \tau_2$$

throughout the synthesis step of a divide-and-conquer method results in the computation of an eigendecomposition $\hat{V}\hat{\Lambda}\hat{V}^\top$ which differs from the original matrix M_p' by an error matrix $E^{(2)}$, for which

$$\|E^{(2)}\|_2 \leq \eta\tau_2$$

where η is a constant of order unity.

Applying this result to our algorithm and again using Weyl's theorem shows that the deviation of the computed eigenvalues $\hat{\lambda}$ from the exact eigenvalues λ' of M_p' can be bounded as

$$|\lambda' - \hat{\lambda}| \leq \eta\tau_2. \tag{2.6}$$

2.4. Numerical Properties of the Algorithm. At this point, we are able to show that the first two objectives stated in Section 1.1, which relate to the numerical accuracy of the method, are achieved.

Putting together the error bounds (2.1) and (2.6) shows that the distance of the computed eigenvalues $\hat{\lambda}$ of M_p' from the exact eigenvalues λ of M_p, which is due to lower rank approximation of the off-diagonal blocks and relaxed deflation, can be bounded as

$$|\lambda - \hat{\lambda}| \leq |\lambda - \lambda'| + |\lambda' - \hat{\lambda}| \leq \tau_1 + \eta\tau_2.$$

Given a block-tridiagonal matrix M_p and an accuracy parameter τ, this implies that (for example) choosing lower rank approximations of the off-diagonal blocks in (2.1) such that $\tau_1 \leq \tau/2$ and setting the deflation tolerance τ_2 such that $\eta\tau_2 \leq \tau/2$ makes it possible to satisfy condition (1.3). Condition (1.4) is satisfied by utilizing the stable method for computing numerically orthogonal eigenvectors of a rank-one modification problem developed by Gu and Eisenstat [17].

3. Implementation. In this section, we will discuss the arithmetic complexity of the algorithm presented and related implementation aspects. The achieved efficiency may strongly depend on these aspects.

3.1. Arithmetic Complexity. In this section, we analyze the dominating terms of the arithmetic complexity of a single merging operation with a cut point c, where a $c \times c$ and an $(l-c) \times (l-c)$ diagonal block are to be connected by a rank-r_i off-diagonal block. In such a rank-r_i merging operation, the eigendecomposition of

$$D + y_i^{(1)} y_i^{(1)\top} + y_i^{(2)} y_i^{(2)\top} + \ldots + y_i^{(r_i)} y_i^{(r_i)\top} \tag{3.1}$$

with a diagonal matrix D has to be computed. As discussed in Section 2.2.4, this rank-r_i modification is handled as a sequence of r_i rank-one modifications according to

```
$D^{(0)} := D$
do  $j = 1, 2, \ldots, r_i$
    factorize  $D^{(j-1)} + y_i^{(j)} y_i^{(j)\top} = Q^{(j)} D^{(j)} Q^{(j)\top}$
    do  $k = j+1, j+2, \ldots, r_i$
        update  $y_i^{(k)} := Q^{(j)\top} y_i^{(k)}$
    end do
end do
```

For each rank-r_i merging operation, the r_i eigenvector matrices $Q^{(1)}$, $Q^{(2)}$, ..., $Q^{(r_i)}$ of the rank-one modification problems have to be multiplied onto the block-diagonal eigenvector matrix Q of the two subproblems to be merged as illustrated in Figs. 3.1 and 3.2. Asymptotically, the eigenvector accumulation over all rank-one modifica-

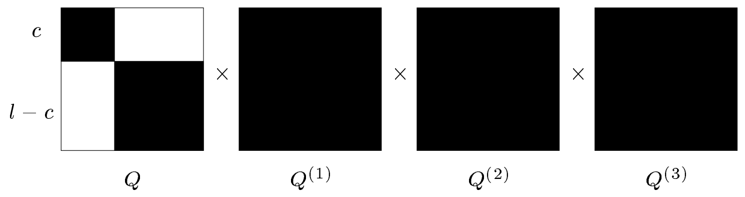

FIG. 3.1. *Sparsity structure in the eigenvector accumulation for a merging operation with rank* $r_i = 3$ (*no deflation*)

tion problems tends to be the most expensive part of the entire divide-and-conquer algorithm described in this paper (cf. [8, 11, 13]).

Order of Rank-One Modifications. The actual flop count of the eigenvector accumulation for a single merging operation depends on how much deflation occurs in each rank-one modification (cf. Fig. 3.2). Additionally, it also depends on whether more or less deflation tends to happen in later rank-one modifications of a single merging operation. In Fig. 3.2 we chose to depict a case where more deflation occurs in *later* rank-one modifications. There are two reasons for that:

- In our experiments the rank-one modifications of (3.1) corresponding to *larger* singular values of the off-diagonal block are performed first. This tends to cause smaller entries in the modification vectors corresponding to later rank-one modifications with smaller singular values (cf. (2.3)), which in turn leads

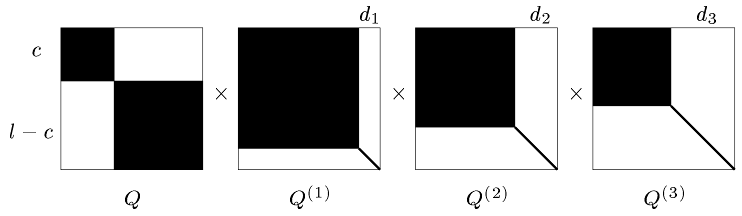

FIG. 3.2. *Sparsity structure in the eigenvector accumulation for a merging operation with rank $r_i = 3$; d_i eigenvalues are deflated in rank-one modification i*

to to more deflation (cf. Section 2.3). In general, we observed more deflation in later rank-one modifications of a single merging operation.

- Obviously, one might consider going through the singular values in reverse order (from the smallest to the largest) and thereby performing rank-one modifications corresponding to smaller singular values first. This could lead to more deflation earlier and less deflation later in a merging operation. However, investigation of a special case indicates that in most situations this is *not* beneficial, because it tends to require slightly more floating point operations for the eigenvector accumulation than the situation depicted in Fig. 3.2. This is summarized in the following.

First it should be pointed out that the $r_i + 1$ eigenvector matrices to be accumulated in a single merging operation have to be multiplied *from left to right*, because $Q^{(i+1)}$ can only be computed *after* $Q^{(i)}$ is known and usually it is not feasible to provide storage for all the intermediate matrices instead of accumulating them immediately.

Multiplying a block-diagonal matrix Q as shown in Figs. 3.1 and 3.2 with a matrix $Q^{(1)}$ and the result with a matrix $Q^{(2)}$, where $d_1 < d_2$ (more deflation occurs later) and $c < n - d_1$, requires

$$2l^3 - 2l^2\,(c+d_1+d_2) + l\left(2c^2+3cd_1+d_1^2+d_2^2\right) - d_1c\,(2c+d_1) \quad \text{flops.} \tag{3.2}$$

When more deflation occurs earlier, i.e., Q is first multiplied with a matrix $Q^{(2)}$ and the result with a matrix $Q^{(1)}$, where $d_1 < d_2$ and $c < n - d_2$, the flop count is

$$\begin{aligned} 2l^3 - 2l^2\,(c+d_1+d_2-1/2) + l\left(2c^2+cd_1+2cd_2-c+d_1^2+d_2^2-d_1\right)& \\ -d_2c\,(2c+d_2) + d_1c\,(d_2-d_1+1)&\,. \end{aligned} \tag{3.3}$$

Subtracting (3.3) from (3.2) yields

$$-l^2 + l\,(2cd_1 - 2cd_2 + c + d_1) + c\,((2c+d_2)\,(d_2-d_1) - d_1)\,. \tag{3.4}$$

Since $d_2 > d_1$, the constant term in (3.4) is always positive, but the coefficient of the linear term is negative whenever $d_1 < c$. Moreover, the dominating quadratic term is always negative. Therefore, in most situations more deflation later in each merging operation tends to lead to slightly fewer floating point operations in the eigenvector accumulation.

In the following we count the floating point operations required for the eigenvector accumulation in the worst case where no deflation occurs.

3.1.1. Eigenvector Update for the First Rank Modification. The block-diagonal eigenvector matrix of the two subproblems (containing a full $c \times c$ block and a full $(l-c) \times (l-c)$ block, see Fig. 3.1) has to be multiplied from the right with the eigenvector matrix of the first rank modification, which is a full $l \times l$ matrix if no deflation occurs. The result of this operation is also a full $l \times l$ matrix.

This operation requires (cf. [13])

$$(2c-1)cl + (2(l-c)-1)(l-c)l = 2l^3 - l^2(4c+1) + 4c^2l \quad \text{flops}. \tag{3.5}$$

In case the merging operation is perfectly balanced ($c = l/2$) this yields a count of

$$2l^3 - l^2(2l+1) + 4l^2/4l = l^3 - l^2 \quad \text{flops}. \tag{3.6}$$

3.1.2. Eigenvector Updates for Later Rank Modifications. In the multiple rank case under consideration, the eigenvector updates corresponding to the following $r_i - 1$ rank modifications involve the multiplication of two full $l \times l$ matrices (the matrix accumulated so far is multiplied with the eigenvector matrix of the current rank modification), *independently* of the cut point c (see Fig. 3.1).

Each such operation requires

$$(2l-1)ll = 2l^3 - l^2 \quad \text{flops}. \tag{3.7}$$

3.1.3. Comparison to a Rank-One Modification Problem. Flop counts (3.6) and (3.7) allow us to quantify the work increase due to a multiple rank modification compared to a rank-one modification as discussed in [13]. If the merging operation is perfectly balanced, the accumulation strategy for computing the eigenvectors corresponding to a rank-r_i modification leads to an increase of the required flops compared to a rank-one modification by a factor of

$$\frac{l^3 - l^2 + (r_i - 1)\left(2l^3 - l^2\right)}{l^3 - l^2} = 1 + (r_i - 1)\left(1 + \frac{l^3}{l^3 - l^2}\right), \tag{3.8}$$

For large l, (3.8) approaches

$$2r_i - 1, \tag{3.9}$$

which indicates that high ranks r_i of the off-diagonal blocks C_i' may become the limiting factor in the efficiency of the method presented here. Obviously, the final merging operations of the synthesis step dominate the work, because they involve the largest matrices. In particular, if the rank of the off-diagonal block corresponding to the final merging operation is r_f, then up to $(r_f - 1)\,2n^3$ flops may be required for the corresponding eigenvector accumulations. This is clearly not attractive for large values of r_f.

Fortunately, there are many situations where this worst-case scenario is too pessimistic. Firstly, deflation often greatly improves the situation. In particular, if accuracy requirements are low, deflation tolerances may be relaxed strongly, as shown in Section 2.3. In most cases this will lead to a large amount of deflation and therefore decrease the size of the matrices to be multiplied (see Section 4.1).

Secondly, if a proper merging order is chosen, the work is dominated by the minimum of the ranks r_i, which will be discussed in more detail in the following.

3.2. Merging Order. When all the off-diagonal blocks are approximated with the same rank, then the analysis given in [13] is applicable and shows that the merging order should be determined such that the merging operations, in particular the late(r) ones, are as balanced as possible.

In the most general situation, where the ranks of the off-diagonal blocks C_i' differ, the r_i have to be taken into account when determining the merging order, since a higher rank implies significantly more arithmetic work for performing the merging operation, as (3.9) illustrates.

Using the flop counts derived in Section 3.1 it is possible to justify putting highest priority on choosing lower rank modifications for later merging operations, *independently* of how unbalanced they may be. In particular, it can be shown that the dominating final merging operation ($l = n$) should correspond to the off-diagonal block C_i' with the lowest rank:

- Let us consider a final merging operation which involves a rank-r_i modification and is as unbalanced as possible ($c = 1$). Flop counts (3.5) and (3.7) yield

  $$2n^3 - 5n^2 + 4n + (r_i - 1)(2n^3 - n^2) = 2r_i n^3 - (r_i + 4)n^2 + 4n \tag{3.10}$$

  flops. Note that this is an *overestimation* of the actual flop count since even in an unbalanced merging operation the cut point always has to be greater than or equal to the rank r_i. More precisely, $\min(c, l - c) \geq r_i \geq 1$ always holds.
- If the same merging operation was perfectly balanced ($c = n/2$), but involved a modification with higher rank $(r_i + x)$ $(x = 1, 2, \ldots)$, its flop count according to (3.6) and (3.7) would be

  $$n^3 - n^2 + (r_i + x - 1)(2n^3 - n^2) = (2r_i + 2x - 1)n^3 - (r_i + x)n^2. \tag{3.11}$$

The difference between (3.11) and (3.10)

$$(2x - 1)n^3 + (4 - x)n^2 - 4n$$

is positive for all $x = 1, 2, \ldots$ and for sufficiently large n. This shows that a modification with higher rank implies more floating point operations, even if it is completely balanced. (For $x = 0$ the difference is negative, because for the same number r_i of rank-one modifications a perfectly balanced merging operation is less expensive than any unbalanced one, as has been shown in [13].)

In our code, we use the following strategy for determining the merging order: First, we determine all the cut points which correspond to the off-diagonal blocks with the minimum rank $r_{\min} := \min_{i=1,2,\ldots,p-1}\{r_i\}$. Among these, we select the final cut point as the one with the least imbalance in the merging operation. Then we continue this strategy recursively for determining the previous cut points in the parts above and below the final cut point.

4. Experiments.

The block-tridiagonal divide-and-conquer method has been implemented in Fortran (`dsbtdc`) and evaluated experimentally. In Section 4.1 it is illustrated that in most cases a significant amount of deflation can be expected, which increases with increasing deflation tolerances. In Section 4.2 the reduction of runtimes for decreasing accuracy requirements is illustrated. In Section 4.3 runtimes and numerical results of the new routine are compared with a corresponding LAPACK routine when full accuracy is required.

For the experiments summarized in Section 4.1, test matrices with specified eigenvalue distributions were created. For the experiments summarized in Sections 4.2 and 4.3, random block-tridiagonal matrices with prescribed ranks of the off-diagonal blocks were generated by creating random symmetric blocks B_i ($i = 1, 2, \ldots, p$) as well as r_i ($i = 1, 2, \ldots, p-1$) random vectors u_i and v_i, which determine the rank-r_i off-diagonal blocks C_i'. The singular values of the off-diagonal blocks were chosen as $\sigma_i^j = 1.\mathrm{D}0/j$ ($j = 1, 2, \ldots, r_i$, $i = 1, 2, \ldots, p-1$) for these test matrices.

The computations were done on a SUN Ultra 5 Workstation with a 400 MHz UltraSPARC-IIi processor in double precision with a machine precision $\varepsilon \approx 1.1 \cdot 10^{-16}$.

The accuracy of each method is measured by the scaled residual error $\mathcal{R}$ and by the departure from orthogonality $\mathcal{O}$ of the eigenvectors, defined by

$$\mathcal{R} := \max_{i=1,2,\ldots,n} \frac{\left\| M_p \hat{v}_i - \hat{\lambda}_i \hat{v}_i \right\|_2}{\|M_p\|_2} \qquad \text{and}$$
$$\mathcal{O} := \max_{i=1,2,\ldots,n} \left\| \left(\hat{V}^\top \hat{V} - I \right) e_i \right\|_2 .$$

4.1. Relaxing Deflation. In order to illustrate how much deflation can be expected on average, `dsbtdc` was run on randomly created block-tridiagonal test matrices with three different prescribed eigenvalue distributions:

1. *uniform*: $\lambda_i = 1 - (i-1)\frac{2}{n-1}$, $i = 1, 2, \ldots, n$;
2. *random*: $\lambda_i = \mathrm{rand}[-1, 1]$; and
3. *clustered* around 0: $\lambda_i = \pm \frac{1}{2^{(i-1)/k}}$, $i = 1, 2, \ldots, n$, where $k = \frac{n}{80}$ was chosen in order to guarantee some minimum distance between clustered eigenvalues.

Results are shown for one matrix for each eigenvalue distribution, each with $n = 3000$, $p = 600$, block sizes $k_i = 5$ ($i = 1, 2, \ldots, p$), and all off-diagonal blocks with full rank. All three matrices generated had the characteristic that the magnitude of their elements decreased when moving away from the diagonal. For comparison, we also show the amount of deflation which occurred for the matrix M_{300}^5 (one of the matrices used in the runtime comparisons of Section 4.2) with the label "block random".

We recorded the deflation for each rank-one modification problem in the synthesis step, which gives five numbers for the last merging operation ($n = 3000$), ten numbers for the two merging operations before that ($n = 1500$), etc. Figs. 4.1 and 4.2 show *two* graphs for each of the four matrices: They are lower and upper bounds of deflation for the rank-one modifications of block sizes greater or equal to $n = 180$. The actual deflation values for all the rank-one modification problems lie between these bounds.

Fig. 4.1 illustrates that for a small deflation tolerance $\tau_2 = 10^{-10}$ the test matrices with clustered eigenvalues and also those with random eigenvalues showed very high amounts of deflation (the upper bounds are at or close to 100%). As expected, significantly less deflation occurred for the matrix with a uniform eigenvalue distribution. However, even in that case almost 25% of the eigenvalues could be deflated for large blocks, which are the most time consuming merges. The amount of deflation occurring for M_{300}^5 tends to lie between the bounds of the other three matrices with known eigenvalue distribution, which was also to be expected.

With a higher deflation tolerance $\tau_2 = 10^{-4}$ much more deflation occurs for all four matrices, as Fig. 4.2 illustrates. In this case the upper bounds for all three matrices with prescribed eigenvalue distributions are at or very close to 100% and therefore cannot be distinguished in the picture. Again, least deflation occurs for the matrix with a uniform eigenvalue distribution. The lower bound for this matrix is

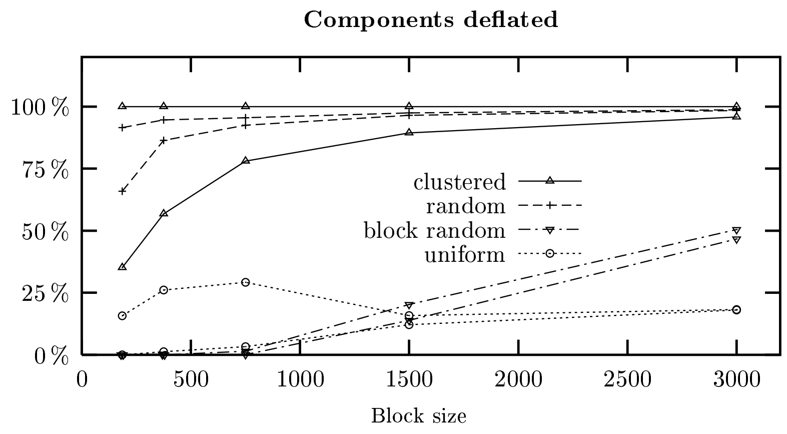

FIG. 4.1. *Lower and upper bounds for the deflation observed in each merging operation for* $\tau_2 = 10^{-10}$

much lower than that for the other three matrices, but it is at a significantly higher level than the corresponding one in Fig. 4.1.

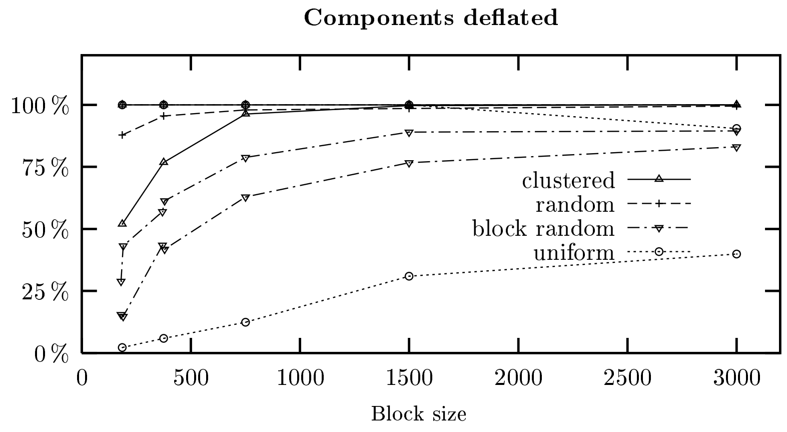

FIG. 4.2. *Lower and upper bounds for the deflation observed in each merging operation for* $\tau_2 = 10^{-4}$

4.2. Variable Accuracy Requirements. Table 4.1 illustrates the effect of relaxing the deflation tolerance τ_2 on the runtimes of `dsbtdc`. Results are shown for the matrices M_{300}^r with $n = 3000$, $p = 300$, and the block sizes $k_i = 10$ ($i = 1, 2, \ldots, p$). The ranks of the off-diagonal blocks r_i were all chosen equal ($r_i = r$ for $i = 1, 2, \ldots, p-1$) and indicated in the superscript. Constructing off-diagonal blocks with prescribed rank r (no approximation of the off-diagonal blocks, $\tau_1 = 0$) made it possible to isolate the influence of the deflation tolerance τ_2 on the runtimes.

Fig. 4.3 shows the same data as ratios $T_{\mathrm{BT}}/T_{\mathrm{LB}}$ of the runtimes T_{BT} for `dsbtdc` and T_{LB} for the routine `LAPACK/dsbevd`, which computes eigenpairs of a banded symmetric matrix by performing tridiagonalization, the tridiagonal divide-and-conquer method, and finally the backtransformation of the eigenvectors.

TABLE 4.1
Runtimes (in seconds) for increasing deflation tolerances.

τ_2	M^1_{300}	M^2_{300}	M^5_{300}	M^6_{300}	M^7_{300}	M^{10}_{300}
`dsbtdc`						
τ_2^L (full accuracy)	30.2	154.3	942.6	1239.6	1557.2	2344.6
10^{-14}	29.4	142.2	931.5	1142.6	1472.9	2224.0
10^{-10}	23.1	95.3	582.6	732.3	922.5	1429.6
10^{-6}	17.5	50.9	229.4	279.8	359.5	498.8
10^{-4}	14.5	31.6	83.2	95.8	103.9	130.1
10^{-2}	11.0	17.0	34.8	40.7	46.5	64.1
`LAPACK/dsbevd`	1501.7	1539.4	1529.7	1566.4	1550.9	1551.4

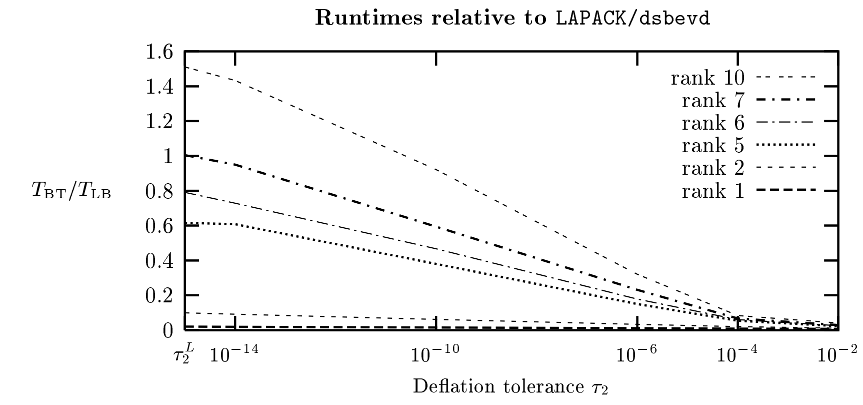

FIG. 4.3. *Runtimes of* `dsbtdc` *relative to* `LAPACK/dsbevd` *for different ranks of the off-diagonal blocks and for varying deflation tolerances*

4.3. Comparison with Other Approaches. There is no standard routine for computing eigenpairs of a block-tridiagonal matrix and therefore a direct comparison is not possible. However, it is possible to compare `dsbtdc` with a routine for computing eigenvalues and eigenvectors of a *banded* symmetric matrix, such as the routine `LAPACK/dsbevd` mentioned before.

As input for `LAPACK/dsbevd` the narrowest band matrix which fully contains the respective block-tridiagonal matrix M_p was used. This matrix contains $2(p-2)$ zero $n/p \times n/p$ triangles in addition to the block-tridiagonal matrix M_p. These triangles fill up during the tridiagonalization performed by `LAPACK/dsbevd`. However, especially for large values of p the overhead is negligible.

The experiments are summarized in Table 4.2. It can be seen that due to improved data-locality, which is important for the memory hierarchies of modern computer systems and also expected to be extremely important for any implementation on a parallel computer, for low and medium rank off-diagonal blocks the block-tridiagonal divide-and-conquer algorithm is more efficient than the standard method for banded eigenvalue problems, even if the eigensystem of the approximate matrix M_p' is computed to full accuracy.

5. Conclusion. A divide-and-conquer based method for approximating eigenpairs of symmetric block-tridiagonal matrices has been proposed. The central ideas

TABLE 4.2
Comparison with standard LAPACK*-routine (full accuracy).*

Routine	M^1_{300}	M^2_{300}	M^5_{300}	M^6_{300}	M^7_{300}	M^{10}_{300}
`dsbtdc`						
$T_{\rm BT}$ $[s]$	30.2	154.3	942.6	1239.6	1557.2	2344.6
$\mathcal{R}$	$9.0 \cdot 10^{-15}$	$6.7 \cdot 10^{-15}$	$8.7 \cdot 10^{-15}$	$1.3 \cdot 10^{-14}$	$1.2 \cdot 10^{-14}$	$1.5 \cdot 10^{-14}$
$\mathcal{O}$	$2.5 \cdot 10^{-15}$	$4.9 \cdot 10^{-15}$	$4.2 \cdot 10^{-15}$	$6.7 \cdot 10^{-15}$	$5.2 \cdot 10^{-15}$	$3.7 \cdot 10^{-15}$
`dsbevd`						
$T_{\rm LB}$ $[s]$	1501.7	1548.0	1529.7	1566.4	1550.9	1551.4
$\mathcal{R}$	$7.8 \cdot 10^{-15}$	$7.4 \cdot 10^{-15}$	$7.2 \cdot 10^{-15}$	$8.0 \cdot 10^{-15}$	$7.5 \cdot 10^{-15}$	$7.3 \cdot 10^{-15}$
$\mathcal{O}$	$5.7 \cdot 10^{-15}$	$7.6 \cdot 10^{-15}$	$6.4 \cdot 10^{-15}$	$5.8 \cdot 10^{-15}$	$5.3 \cdot 10^{-15}$	$5.6 \cdot 10^{-15}$

which allow to reduce computing times at the cost of gradually reduced accuracy are (i) lower rank approximation of the off-diagonal blocks, (ii) a generalized divide-and-conquer method for block-tridiagonal matrices, and (iii) relaxing the deflation tolerance in the synthesis step of this divide-and-conquer method. It has been shown that especially for medium and low accuracy requirements the proposed method is very efficient compared to the standard method for band matrices used in LAPACK.

Future Work. In order to complete a framework for approximating eigenpairs of arbitrary symmetric matrices, we will investigate several alternatives for approximating full matrices by block-tridiagonal matrices of the general form (1.1).

Not in all situations the full approximate spectral decomposition (1.2) is needed. In many important applications only $k < n$ eigenpairs are to be computed. For such cases it would be desirable to have an efficient method with a proportionally reduced computational effort. We are investigating alternative approaches for computing the eigenvectors of a block-tridiagonal matrix given its eigenvalues with this feature and comparing them to competing methods, such as Krylov subspace methods.

Acknowlegdements. We would like to thank Robert Day for his help in performing the experiments.

References.

[1] E. ANDERSON, Z. BAI, C. H. BISCHOF, S. BLACKFORD, J. W. DEMMEL, J. J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. C. SORENSEN, LAPACK *Users' Guide*, SIAM Press, Philadelphia, PA, 3rd ed., 1999.

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

[3] P. ARBENZ, W. GANDER, AND G. H. GOLUB, *Restricted rank modification of the symmetric eigenvalue problem: Theoretical considerations*, Linear Algebra Appl., 104 (1988), pp. 75–95.

[4] P. ARBENZ AND G. H. GOLUB, *On the spectral decomposition of Hermitian matrices modified by low rank perturbations with applications*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 40–58.

[5] C. H. BISCHOF, B. LANG, AND X. SUN, *A framework for symmetric band reduction*, ACM Trans. Math. Software, 26 (2000), pp. 581–601.

[6] J. R. BUNCH, C. P. NIELSEN, AND D. C. SORENSEN, *Rank-one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

[7] J. J. M. CUPPEN, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[8] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM Press, Philadelphia, PA, 1997.

[9] J. J. Dongarra and D. C. Sorensen, *A fully parallel algorithm for the symmetric eigenproblem*, SIAM J. Sci. Comput., 8 (1987), pp. s139–s154.

[10] W. N. Gansterer, D. F. Kvasnicka, and C. W. Ueberhuber, *Multi-sweep algorithms for the symmetric eigenproblem*, in VECPAR'98—Third International Conference for Vector and Parallel Processing, J. M. L. M. Palma, J. J. Dongarra, and V. Hernandez, eds., vol. 1573 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg New York Tokyo, June 1998, pp. 20–28.

[11] W. N. Gansterer, J. Schneid, and C. W. Ueberhuber, *A divide-and-conquer method for symmetric banded eigenproblems. Part II: Complexity analysis*, Technical Report AURORA TR1999-14, Vienna University of Technology, 1999.

[12] ——, *A low-complexity divide-and-conquer method for computing eigenvalues and eigenvectors of symmetric band matrices*, (2000). submitted.

[13] W. N. Gansterer, R. C. Ward, and R. P. Muller, *An extension of the divide-and-conquer method for a class of symmetric block-tridiagonal eigenproblems*, Technical Report UT-CS-00-447, Department of Computer Science, University of Tennessee, Knoxville, TN, 2000. submitted.

[14] K. Gates and P. Arbenz, *Parallel divide and conquer algorithms for the symmetric tridiagonal eigenproblem*, Technical Report 222, Institut für Wissenschaftliches Rechnen, ETH Zürich, 1994.

[15] G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 3rd ed., 1996.

[17] M. Gu and S. C. Eisenstat, *A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1266–1276.

[18] ——, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.

[19] L. Kaufman, *Band reduction algorithms revisited*, ACM Trans. Math. Software, 26 (2000), pp. 551–567.

[20] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM Press, Philadelphia, PA, 1998.

[21] J. Rutter, *A serial implementation of cuppen's divide and conquer algorithm for the symmetric eigenvalue problem*, Lapack Working Note 69, Computer Science Division (EECS), University of California at Berkeley, Berkeley, CA, 1994.

[22] H. R. Schwarz, *Tridiagonalization of a symmetric band matrix*, Numer. Math., 12 (1968), pp. 231–241.

[23] D. C. Sorensen and P. T. P. Tang, *On the orthogonality of eigenvectors computed by divide-and-conquer techniques*, SIAM J. Numer. Anal., 28 (1991), pp. 1752–1775.

[24] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry*, Dover Publications, Mineola, NY, 1996.

[25] F. Tisseur and J. J. Dongarra, *A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures*, SIAM J. Sci. Comput., 20 (1999), pp. 2223–2236.