SIAM J. SCI. COMPUT.
Vol. 25, No. 1, pp. 65–85

COMPUTING APPROXIMATE EIGENPAIRS OF SYMMETRIC BLOCK TRIDIAGONAL MATRICES*

WILFRIED N. GANSTERER†, ROBERT C. WARD†, RICHARD P. MULLER‡, AND WILLIAM A. GODDARD III‡

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 [W. N. Gansterer, R. C. Ward, and R. P. Muller, *ACM Trans. Math. Software*, 28 (2002), pp. 45–58], 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 cost compared to standard methods such as, for example, implemented in LAPACK.

Key words. block tridiagonal matrix, eigenvalue problem, divide-and-conquer method, approximate eigenpairs

AMS subject classifications. 15A18, 65F15, 65G50, 65Y20

DOI. 10.1137/S1064827501399432

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 $B_i \in \mathbb{R}^{k_i \times k_i}$ $(i = 1, 2, \ldots, p)$ along the diagonal are symmetric, and the off-diagonal blocks $C_i \in \mathbb{R}^{k_{i+1} \times k_i}$ $(i = 1, 2, \ldots, p-1)$ 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.

*Received by the editors December 10, 2001; accepted for publication (in revised form) January 17, 2003; published electronically September 9, 2003. This work was supported by the Academic Strategic Alliances Program of the Accelerated Strategic Computing Initiative (ASCI/ASAP) under subcontract B341492 of DOE contract W-7405-ENG-48. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/25-1/39943.html

†Department of Computer Science, University of Tennessee, 203 Claxton Complex, 1122 Volunteer Blvd., Knoxville, TN 37996-3450 (wilfried@cs.utk.edu, ward@cs.utk.edu).

‡Materials and Process Simulation Center, Beckman Institute, California Institute of Technology 139-74, Pasadena, CA 91125 (rpm@wag.caltech.edu, wag@wag.caltech.edu).

1.1. Objectives. Given a (variable) accuracy parameter $\tau \geq \varepsilon$ ($\tau < 0.1$), where ε denotes the machine precision (unit roundoff), the goal is to find an *approximate numerical spectral decomposition*

$$M_p = \hat{V}\hat{\Lambda}\hat{V}^\top + E \quad \text{with} \quad \|E\|_2 = O(\tau\|M_p\|_2). \tag{1.2}$$

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

$$\mathcal{O} := \max_{i=1,2,\ldots,n} \left\| \left(\hat{V}^\top \hat{V} - I \right) e_i \right\|_2 = O(\varepsilon n), \tag{1.3}$$

where $e_i \in \mathbb{R}^n$ has components δ_{ij} (Kronecker delta). An algorithm which produces a decomposition with properties (1.2) and (1.3) is for reduced accuracy requirements the equivalent of a *stable* algorithm (cf. section 2.4).

The method discussed in this paper for computing (1.2) has the attractive feature that lower accuracy requirements lead to a proportionate 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 [32, Chap. 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, and higher accuracy usually becomes more important as it proceeds. The matrices arising in quantum chemistry 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 for performing such a block tridiagonal approximation of a general matrix are investigated in [5].

Since the method developed in this paper has a variable accuracy parameter τ, it is nicely applicable to the SCF method. More generally, it can be used in many other areas of electronic structure computations, for example, when modeling photoisomerization in retinal molecules, photosynthesis, charge transfer in chemical reactions, or in material design. Moreover, we anticipate applications in the context of preconditioning (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 [30, 22, 7], 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 [9]. 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 [18, 8]. Over time, numerically stable and efficient implementations of Cuppen's method were developed [11, 31, 20, 21, 29].

The divide-and-conquer approach not only has attractive parallelization properties [33, 17], but 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 [10, Chap. 5].

In some situations, tridiagonalization of a band matrix can be comparatively expensive *relative* to the total cost of the calculation of eigenpairs [23, Chap. 7]. Moreover, unfavorable data access patterns and bad data locality may cause inefficiencies in a tridiagonalization process [12]. 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. (See [2], based on [4, 3], and more recently [13, 14].) One of the central questions remaining is that of numerical stability.

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 [16]. 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 approximating them with rank-one matrices is not always sufficiently accurate. The algorithm discussed in this paper can 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 of investigating 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'. Analogous 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). Approximations may be introduced into the synthesis step of our method, which can lead to additional time savings if the accuracy requirements are low (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 [16] is to allow for *higher rank approximations* of the off-diagonal blocks C_i. The SVDs (see [19, Chap. 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 $m_i := \min(k_i, k_{i+1})$, $\sigma_1^i \geq \sigma_2^i \geq \cdots \geq \sigma_{m_i}^i \geq 0$, 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 (largest) r_i 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 $V_i := \left(v_1^i | v_2^i | \ldots | v_{r_i}^i\right) \in \mathbb{R}^{k_i \times r_i}$. Denoting the remaining terms in the SVDs by

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

yields the representations

$$C_i = C_i' + C_i'', \quad i = 1, 2, \ldots, p-1.$$

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

$$E^{(1)} := \begin{pmatrix} \mathbf{0} & {C_1''}^\top & & & \\ C_1'' & \mathbf{0} & {C_2''}^\top & & \\ & C_2'' & \mathbf{0} & \ddots & \\ & & \ddots & \ddots & {C_{p-1}''}^\top \\ & & & C_{p-1}'' & \mathbf{0} \end{pmatrix}.$$

Invoking Weyl's theorem (see, for example, [10, Chap. 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

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

Using MATLAB notation, i.e., denoting the lower triangular part of a matrix A by $\operatorname{tril}(A)$ and the upper triangular part of A by $\operatorname{triu}(A)$, we have

$$\begin{aligned} \|E^{(1)}\|_2 &= \|\operatorname{tril}(E^{(1)}) + \operatorname{triu}(E^{(1)})\|_2 \leq 2\|\operatorname{tril}(E^{(1)})\|_2 \\ &= 2 \max_{i=1,2,\ldots,p-1} \sqrt{\max_{j=1,2,\ldots,k_i} \left|\lambda_j\left({C_i''}^\top C_i''\right)\right|} = 2 \max_{i=1,2,\ldots,p-1} \sigma_{r_i+1}^i. \end{aligned} \tag{2.1}$$

(Note that $\operatorname{tril}(E^{(1)})^\top \operatorname{tril}(E^{(1)})$ is a block diagonal matrix.) This leads to the error bound

$$|\lambda - \lambda'| \leq 2 \max_{i=1,2,\ldots,p-1} \sigma_{r_i+1}^i =: \tau_1 \|M_p\|_2 \tag{2.2}$$

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 sections.

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 $\tilde{M'}_p := \text{block-diag}(\tilde{B}_1, \tilde{B}_2, \ldots, \tilde{B}_p)$:

$$M'_p = \tilde{M'}_p + \sum_{i=1}^{p-1} W_i W_i^\top. \tag{2.3}$$

The matrices $W_i \in \mathbb{R}^{n\times r_i}$ in (2.3) are given as

$$W_1 := \begin{pmatrix} V_1\Sigma_1^{1/2} \\ U_1\Sigma_1^{1/2} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \quad W_{p-1} := \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ V_{p-1}\Sigma_{p-1}^{1/2} \\ U_{p-1}\Sigma_{p-1}^{1/2} \end{pmatrix}, \tag{2.4}$$

$$W_i := \begin{pmatrix} \mathbf{0} \\ V_i\Sigma_i^{1/2} \\ U_i\Sigma_i^{1/2} \\ \mathbf{0} \end{pmatrix}, \quad i = 2,3,\ldots,p-2.$$

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

$$\tilde{M'}_p = QDQ^\top. \tag{2.5}$$

2.2.3. Synthesis of the solutions of the subproblems. Substituting (2.5) into (2.3) and denoting $Y_i := Q^\top W_i$ yields the representation

$$M'_p = Q \left(D + \sum_{i=1}^{p-1} Y_i Y_i^\top \right) Q^\top, \tag{2.6}$$

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

$$S := D + \sum_{i=1}^{p-1} Y_i Y_i^\top. \tag{2.7}$$

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] and Arbenz and Golub [4] have developed a method for performing an eigenanalysis of the entire rank-r modification (2.7) 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 Figure 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.

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 [20] can be utilized in each rank-one modification and therefore numerical stability and numerical orthogonality of the computed eigenvectors can be guaranteed (see section 2.4). 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 (analogous 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

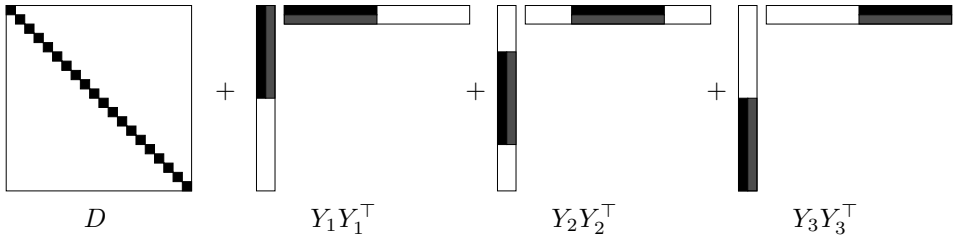

FIG. 2.1. *Sparsity structure of the matrices* Y_i ($n = 20$, $p = 4$, $r_i = 2$ *for* $i = 1, 2, 3$).

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 [8, 9, 11] 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:

Type I. 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 $D + xx^\top$.

Type II. 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 described for Type I.

This process is called *deflation*. It not only reduces the problem size for the eigenvalue computation but it also introduces a block structure in the eigenvector matrices which reduces the work required for accumulating them (see Figure 3.1).

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* δ_2 is chosen as the product of the *deflation parameter* τ_2 and the norm of the matrix of the eigenproblem. For accuracy at the level of the machine precision ε, *standard deflation* has to be used, where τ_2 is a moderate multiple of ε. For example, in LAPACK [1], the deflation tolerance is set to

$$\delta_2^L := \tau_2^L \max\left(\max_{i=1,2,\ldots,n} |d_i|, \max_{i=1,2,\ldots,n} |x_i|\right) \quad \text{with} \quad \tau_2^L = 8\varepsilon.$$

For lower accuracy requirements, as considered in this paper, it is possible to relax the deflation criteria accordingly by increasing the deflation parameter 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 one corresponding component of the modification vector x in the Type II deflation be denoted by γ and σ.

Deflating all components x_i of x for which

$$|x_i| \leq \tau_2 \left(\|D\|_2 + \|x\|_2^2\right) \quad \text{(Type I)}$$

and considering diagonal entries d_i and d_j as equal if

$$|\,(d_{i+1} - d_i)\,\gamma\sigma| \leq \tau_2 \left(\|D\|_2 + \|x\|_2^2\right) \quad \text{(Type II)}$$

corresponds to computing the eigenpairs of a rank-one modification problem whose difference from the original rank-one modification problem can be bounded normwise by $O(\tau_2 \left(\|D\|_2 + \|x\|_2^2\right))$. Moreover, it has been shown [11, 6] that the application of the standard deflation tolerance throughout the synthesis step of a *tridiagonal* divide-and-conquer method results in the computation of an eigendecomposition $\hat{V}\hat{\Lambda}\hat{V}^\top$ which differs from the original tridiagonal matrix T by an error matrix $E^{(2)}$, for which

$$\|E^{(2)}\|_2 \leq \tau_2 c_1(n) \|T\|_2,$$

where $c_1(n)$ grows moderately in n.

In our block tridiagonal method this error bound depends on the ranks r_i of the off-diagonal blocks (see also [6]), since the errors are accumulated additively. Applying this result to our algorithm and again using Weyl's theorem therefore shows that the deviation of the computed eigenvalues $\hat{\lambda}$ from the exact eigenvalues λ' of M_p' can be (crudely) bounded as

$$|\lambda' - \hat{\lambda}| \leq \sqrt{r_{\max}}\tau_2 c_1(n) \|M_p'\|_2 \leq \sqrt{r_{\max}}\tau_2 (1+\tau_1) c_1(n) \|M_p\|_2, \tag{2.8}$$

where $r_{\max} = \max_{i=1,2,\ldots,p-1}\{r_i\}$, the maximum rank of the off-diagonal blocks.

2.4. Numerical properties of the block tridiagonal eigensolver. At this point, we discuss the numerical properties of the algorithm proposed. Whenever possible, we follow the arguments given in [20, 21], adapting or extending them if required.

2.4.1. Eigenvalue errors. Putting together the error bounds (2.2) and (2.8) 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

$$|\lambda - \hat{\lambda}| \leq |\lambda - \lambda'| + |\lambda' - \hat{\lambda}| \leq \tau_1 \|M_p\|_2 + \sqrt{r_{\max}}\tau_2 (1+\tau_1) c_1(n) \|M_p\|_2.$$

Given a block tridiagonal matrix M_p and an accuracy parameter τ, this implies that choosing (for example) the ranks of the approximations of the off-diagonal blocks in (2.2) such that $\tau_1 \leq \tau/2$ and setting the deflation parameter τ_2 such that $\sqrt{r_{\max}}\tau_2(1+\tau_1)c_1(n) \leq \tau/2$ makes it possible to compute the eigenvalues to the required absolute accuracy:

$$|\hat{\lambda} - \lambda| \leq \tau \|M_p\|_2.$$

2.4.2. Relaxed deflation. In this section it is shown that applying Gu and Eisenstat's method [20] in combination with *relaxed* deflation to a rank-one modification problem $D + xx^\top$ is a numerically stable way for computing an eigendecomposition $\hat{Q}^{(1)}\hat{D}^{(1)}\hat{Q}^{(1)\top}$ at the accuracy level of the relaxed deflation, i.e.,

$$D + xx^\top = \hat{Q}^{(1)}\hat{D}^{(1)}\hat{Q}^{(1)\top} + O(\tau_2 \left(\|D\|_2 + \|x\|_2^2\right)). \tag{2.9}$$

Gu and Eisenstat [20] base their stability analysis on the following assumptions:

- Standard deflation is used ($\tau_2 = \tau_2^S := 2\eta n^2$, where η is a small multiple of ε that is independent of n).
- The stopping criterion for computing the eigenvalues is essentially $\eta O(n)$.
- $\eta n < 1/100$.

They show that

- the eigenvalues are computed to high absolute accuracy (the error is $O(\eta n)$);
- the difference between x and the constructed vector $\tilde{x}$ of the rank-one modification problem $D + \tilde{x}\tilde{x}^\top$, for which the computed eigenvalues are *exact*, can be bounded by $4\eta n^2 \|x\|_2$ componentwise, which implies stability with ε-accuracy; and
- the computed eigenvectors are numerically orthogonal.

The method discussed in this paper provides the possibility to relax deflation in each rank-one modification problem $D + xx^\top$ of the synthesis phase by choosing $\tau_2 = \tau_2^R := 2\alpha n^2$ with $\alpha \geq \eta$. This leads to a problem $D_R + x_R {x_R}^\top$ which differs from the original one by $O(\tau_2^R \left(\|D\|_2 + \|x\|_2^2\right))$.

Apart from that, our method utilizes the strategy developed in [20] for solving each rank-one modification problem: (i) the eigenvalues of $D_R + x_R {x_R}^\top$ are computed with the same stopping criterion as proposed in [20], (ii) the vector $\tilde{x}_R$ is constructed such that these eigenvalues are exact for the modified problem $D_R + \tilde{x}_R \tilde{x}_R^\top$, and (iii) the eigenvectors are computed from this modified problem.

With the assumption $\eta n < 1/100$, completely analogous to [20], it follows that the difference between x_R and $\tilde{x}_R$ is bounded by $5\eta n^2 \|x_R\|_2$ componentwise, which implies stability with ε-accuracy for computing eigenvalues and eigenvectors of the problem $D_R + x_R {x_R}^\top$ resulting from relaxed deflation.

Putting everything together, we have

$$\begin{aligned} D + xx^\top &= D_R + x_R {x_R}^\top + O(\tau_2^R \left(\|D\|_2 + \|x\|_2^2\right)) \\ &= \hat{Q}^{(1)} \hat{D}^{(1)} \hat{Q}^{(1)\top} + O(\tau_2^S \|x_R\|_2) + O(\tau_2^R \left(\|D\|_2 + \|x\|_2^2\right)) \\ &= \hat{Q}^{(1)} \hat{D}^{(1)} \hat{Q}^{(1)\top} + O(\tau_2^R \left(\|D\|_2 + \|x\|_2^2\right)) \end{aligned}$$

since $\alpha \geq \eta$.

2.4.3. Stability. Condition (1.2) follows from stability with τ_2-accuracy of the eigendecomposition of each rank-one modification problem, from the additivity of the errors from consecutive rank-one modification problems, and from the fact that stable algorithms and orthogonal similarity transformations are used to transform the eigendecomposition of the synthesis matrix (2.7) into the desired spectral decomposition (1.2).

Condition (1.3) is satisfied because we use the stable method developed by Gu and Eisenstat [20] for computing numerically orthogonal eigenvectors of each rank-one modification problem.

2.4.4. Residuals. Having established (1.2), it is easy to see that the residuals s_i are of the same order as the eigenvalue error. Denote

$$s_i := M_p \hat{v}_i - \hat{\lambda}_i \hat{v}_i \quad \text{and} \quad R := M_p \hat{V} - \hat{V} \hat{\Lambda}.$$

From (1.3),

$$\|\hat{V}\|_2 = 1 + O(\varepsilon^{1/2} n^{3/4})$$

and

$$R = \left(M_p - \hat{V}\hat{\Lambda}\hat{V}^\top\right)\hat{V} + E^{(3)} \quad \text{with} \quad \|E^{(3)}\|_2 = O(\tau c_2(\varepsilon, n)\|M_p\|_2),$$

where $c_2(\varepsilon, n)$ grows moderately in n. We have

$$\begin{aligned}\|s_i\|_2 \leq \|R\|_2 &\leq \|M_p - \hat{V}\hat{\Lambda}\hat{V}^\top\|_2\|\hat{V}\|_2 + \|E^{(3)}\|_2 \\ &\leq (\beta_1\tau\|M_p\|_2)\left(1 + \beta_2\varepsilon^{1/2}n^{3/4}\right) + \beta_3\tau c_2(\varepsilon, n)\|M_p\|_2 \\ &\leq \beta_4\tau\left(1 + \varepsilon^{1/2}n^{3/4} + c_2(\varepsilon, n)\right)\|M_p\|_2\end{aligned}$$

with some constants $\beta_1, \beta_2, \beta_3, \beta_4$, and therefore

$$\|s_i\|_2 = O\left(\tau c_3(\varepsilon, n)\|M_p\|_2\right), \tag{2.10}$$

where $c_3(\varepsilon, n)$ grows moderately in n.

2.4.5. Eigenvector error. It is well known that the eigenvalues of a real symmetric matrix A are always well conditioned with respect to a symmetric perturbation, while the sensitivity to perturbation of an invariant subspace depends on the separation of the associated eigenvalues from the rest of the spectrum (see, for example, [34] or [19, Chap. 8]). In particular, this implies that *componentwise* a computed eigenvector approximation $\hat{v}_i$ may differ significantly from the exact v_i.

In order to quantify this error, we first use (1.2) and (1.3) to conclude that the differences between the computed eigenvalue approximations $\hat{\lambda}_i$ and the corresponding Rayleigh quotients of the computed eigenvector approximations $\hat{v}_i$ are of the same order τ as the accuracy of the eigenvalue approximations:

$$\|\hat{v}_i^\top M_p\hat{v}_i - \hat{\lambda}_i\|_2 = \|\hat{v}_i^\top E\hat{v}_i\|_2 \leq \|E\|_2 = O(\tau\|M_p\|_2).$$

Therefore, we can apply Theorem 11.7.1 from [23] in order to bound the eigenvector error. The theorem tells us that the quality of the pair $(\hat{\lambda}_i, \hat{v}_i)$ as an approximation for (λ_i, v_i) is related to the *gap* of $\hat{\lambda}_i$, i.e., the distance of $\hat{\lambda}_i$ to the nearest eigenvalue other than λ_i. More specifically, a bound on the sine of the angle between v_i and $\hat{v}_i$ is less than $\tau/gap(\hat{\lambda}_i)$.

Obviously, each eigenvalue needs to be approximated at a higher resolution than the corresponding gap in order to ensure a one-to-one correspondence between exact eigenvalues and eigenvalue approximations. This is consistent with the information of the theorem mentioned before, which implies that the accuracy parameter τ at the very least has to be *smaller* than the minimum gap in the part of the spectrum which is of interest. However, this requirement may be too weak, since it only yields a bound of the sine of the angle between computed and exact eigenvector less than one. The precise requirement on the relationship between the minimum gap and the parameter τ depends on the accuracy requirements on individual eigenvector approximations. If the corresponding gap is too small, it may be impossible to satisfy certain accuracy requirements on an eigenvector approximation.

Clearly, information about the gap is not always readily available. Our code, which is described in the following sections, monitors the differences between the eigenvalue approximations in order to estimate the gap. If this estimate is too small (relative to τ), it issues a warning about potential inaccuracies in individual eigenvector approximations.

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

3.1. Arithmetic complexity. First, 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

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

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 transformation matrices $T^{(1)}$, $T^{(2)}$, ..., $T^{(r_i)}$ and the r_i eigenvector matrices $Q^{(1)}$, $Q^{(2)}$, ..., $Q^{(r_i)}$ of the rank-one modification problems are multiplied onto the block diagonal eigenvector matrix Q of the two subproblems to be merged as illustrated in Figure 3.1. Each transformation matrix $T^{(i)} = G^{(i)} P^{(i)}$ is the product of a matrix $P^{(i)}$ combining all column permutations resulting from Type I deflation and a matrix $G^{(i)}$ combining all Givens rotations resulting from Type II deflation. When applied, it tends to make only minor changes to the zero-nonzero structure of the resulting matrix. After that, the eigenvector matrix $Q^{(i)}$ of the rank-one modification problem is computed and multiplied onto this resulting matrix. Asymptotically, the eigenvector accumulation over all rank-one modification problems tends to be the most expensive part of the entire divide-and-conquer algorithm described in this paper (cf. [10, 13, 16]). However, the special structure resulting from deflation leads to substantial computational savings in this operation.

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. Figure 3.1). Additionally, it also depends on whether more or less deflation tends to happen in later rank-one modifications of a single merging operation. In Figure 3.1 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. (2.4)), which in turn leads 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

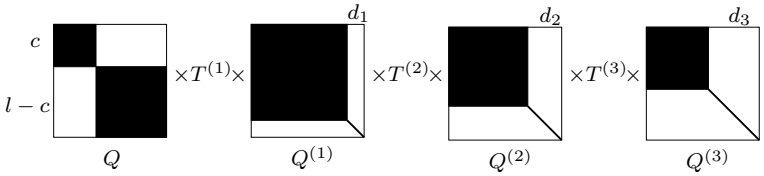

FIG. 3.1. *Sparsity structure in the eigenvector accumulation for a merging operation with rank $r_i = 3$; d_i eigenvalues are deflated in rank-one modification i; $T^{(i)}$ denotes the transformation matrix resulting from Type* I *and Type* II *deflation; multiplication proceeds from left to right.*

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 than the situation depicted in Figure 3.1. For details, see [15].

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 Figure 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.

The highest order terms of the arithmetic complexity of this operation are (cf. [16])

$$2l^3 - 4c\, l^2 + 4c^2\, l \quad \text{flops.} \tag{3.2}$$

For a perfectly balanced merging operation ($c = l/2$), this reaches its minimum at

$$l^3 \quad \text{flops.} \tag{3.3}$$

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.

Considering highest order terms, each such operation requires

$$2l^3 \quad \text{flops.} \tag{3.4}$$

3.1.3. Comparison to a rank-one modification problem. Flop counts (3.3) and (3.4) allow us to quantify the work increase due to a multiple rank modification compared to a rank-one modification as discussed in [16]. In a perfectly balanced merging operation, the ratio of highest order terms of the flop count for the accumulation strategy for computing the eigenvectors corresponding to a rank-r_i modification and the flop count for computing the eigenvectors corresponding to a rank-*one* modification is

$$\frac{l^3 + (r_i - 1)\, 2l^3}{l^3} = 2r_i - 1. \tag{3.5}$$

Consequently, for large l, the solution of a rank-r_i modification problem with an accumulation strategy may require $2r_i - 1$ times as many flops as a rank-one modification

problem. This 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. For example, comparison with the estimated flop count of $9n^3$ for the symmetric QR algorithm applied to a full matrix [19, Chap. 8] indicates that *without deflation* our divide-and-conquer algorithm would only be able to compete for block tridiagonal matrices with *low* off-diagonal ranks.

Fortunately, in most practical situations this worst-case scenario is too pessimistic, because deflation 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 a large decrease in the size of the matrices to be multiplied (see section 4.1).

Still, the work is strongly influenced by the rank r_f in the final merging operation. This illustrates that it is important to choose a proper merging order and also indicates that the merging operation with the *minimum rank* should be performed last. This aspect 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 [16] 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 this situation, it is possible to determine the optimal merging order by solving a corresponding minimization problem exactly (using dynamic programming), but there is also a simple heuristic that usually provides a sufficiently accurate solution for purposes of this algorithm [16].

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 (3.5) illustrates.

Using the flop counts derived in section 3.1, it is possible to justify putting the 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_f modification and is as unbalanced as possible ($c = 1$). Flop counts (3.2) and (3.4) yield

$$2n^3 - 4n^2 + 4n + (r_f - 1)2n^3 = 2r_f n^3 - 4n^2 + 4n \tag{3.6}$$

  flops. Note that this is an *overestimation* of the actual flop count since even in an unbalanced merging operation the cut point is always greater than or equal to the rank r_i. More precisely, $\min(c, l - c) \geq r_i \geq 1$ holds for all i.
- If the same merging operation was perfectly balanced ($c = n/2$), but involved a modification with higher rank ($r_f + x$) ($x = 1, 2, \ldots$), its flop count according to (3.3) and (3.4) would be

$$n^3 + (r_f + x - 1)2n^3 = (2r_f + 2x - 1)n^3. \tag{3.7}$$

The difference between (3.7) and (3.6) is positive for all $x \geq 1$ 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 $x = 0$ and sufficiently large n the difference is negative, because for the same number r_f of rank-one modifications a perfectly balanced merging operation is less expensive than any unbalanced one, as has been shown in [16].)

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 apply 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 and the new routine is compared with corresponding LAPACK routines. In section 4.3 it is illustrated how the new method performs in an application problem from quantum chemistry.

For the experiments summarized in section 4.1, test matrices with specified eigenvalue distributions were created. For the experiments summarized in section 4.2, random block tridiagonal matrices with *prescribed* ranks $r_i \leq \min(k_i, k_{i+1})$ 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 $\sigma_i^j = 1/j$ ($j = 1, 2, \ldots, r_i$; $i = 1, 2, \ldots, p-1$) for these test matrices.

All computations were done on a SUN Ultra 5 workstation with a 400 MHz UltraSPARC-IIi processor, which has 512 MB RAM, 2 MB L2 cache, 16 KB L1 data cache, and 16 KB L1 instruction cache, using double precision with a machine precision $\varepsilon \approx 1.1 \cdot 10^{-16}$.

4.1. Relaxing deflation. In order to illustrate how much deflation can be expected, `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*: $\lambda_i = \text{rand}[-1, 1]$; 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. Figures 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 than or equal to $n = 180$. The

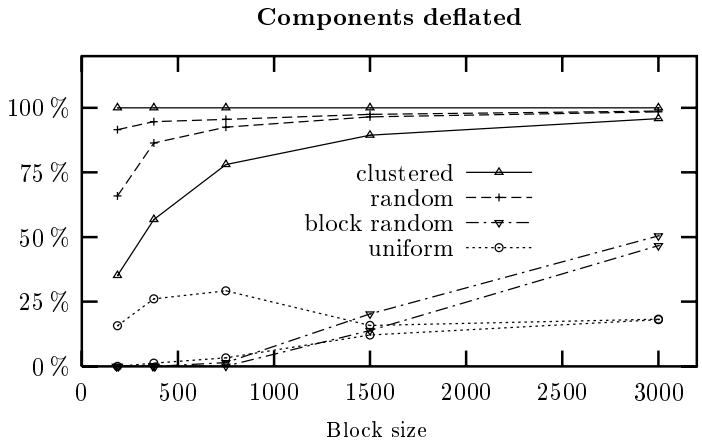

FIG. 4.1. *Lower and upper bounds for the deflation observed in each merging operation for* $\tau_2 = 10^{-10}$.

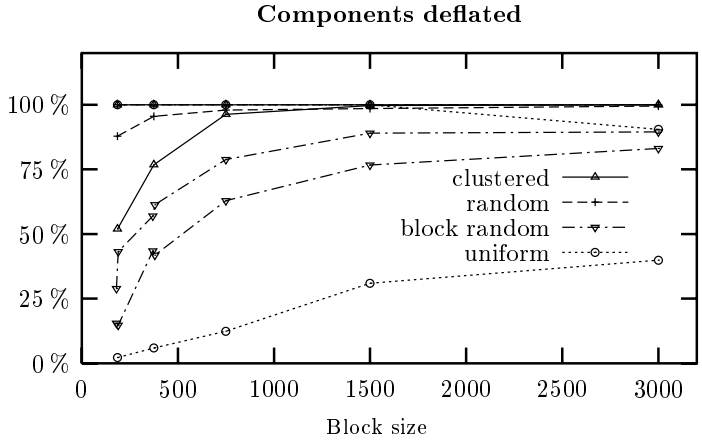

FIG. 4.2. *Lower and upper bounds for the deflation observed in each merging operation for* $\tau_2 = 10^{-4}$.

actual deflation values for all the rank-one modification problems lie between these bounds.

Figure 4.1 illustrates that for a small deflation parameter $\tau_2 = 10^{-10}$ 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 determine the most time consuming merging operations. 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 larger deflation parameter $\tau_2 = 10^{-4}$ much more deflation occurs for all four matrices, as Figure 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

TABLE 4.1
Runtimes (in seconds) for increasing deflation tolerances; τ_2^L denotes the deflation parameter used in LAPACK (*see section* 2.3).

τ_2	M_{300}^1	M_{300}^2	M_{300}^5	M_{300}^6	M_{300}^7	M_{300}^{10}
`dsbtdc`						
τ_2^L (full accuracy)	25.6	143.2	983.6	1249.6	1604.2	2524.4
10^{-14}	24.6	133.3	921.8	1165.6	1496.3	2351.3
10^{-10}	19.1	86.6	556.4	703.4	909.5	1394.0
10^{-6}	13.7	43.8	212.5	269.6	325.4	467.5
10^{-4}	10.6	25.1	70.2	76.9	91.4	108.5
10^{-2}	7.0	10.6	21.7	25.2	29.1	39.7
`LAPACK/dsbevd`	1117.9	1127.7	1156.7	1159.3	1161.8	1167.0
`LAPACK/dsyevd`	705.4	693.8	695.9	692.2	695.5	690.8

matrix with a uniform eigenvalue distribution. The lower bound for this matrix is much lower than that for the other three matrices, but it is at a significantly higher level than the corresponding one in Figure 4.1.

4.2. Comparison with other methods. In this section, the numerical behavior of our method is illustrated by the absolute error $\mathcal{E}$ in the eigenvalues computed by `dsbtdc` (compared to eigenvalues computed with MATLAB), by the scaled residual error $\mathcal{R}$, and by the departure from orthogonality $\mathcal{O}$ of the eigenvectors, defined by

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

There is no standard routine for computing eigenpairs of a block tridiagonal matrix. We show comparisons of `dsbtdc` with two tridiagonalization-based routines from LAPACK [1]:

- `LAPACK/dsbevd`, which computes eigenvalues and eigenvectors of a banded symmetric matrix, and
- `LAPACK/dsyevd`, which computes eigenpairs of a general symmetric matrix.

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. As input for `LAPACK/dsyevd`, the block tridiagonal matrix was completed to a *full* matrix by zero entries. This routine has the highest memory requirements since the entire matrix is filled up by the first Householder transformation of the tridiagonalization process.

Table 4.1 illustrates the effect of relaxing the deflation parameter τ_2 on the runtimes of `dsbtdc`. Results are shown for the matrices M_{300}^r with $n = 3000$, $p = 300$, and block sizes $k_i = 10$ $(i = 1, 2, \ldots, p)$. The superscript r indicates the ranks of the off-diagonal blocks, which were all chosen to be equal ($r_i = r$ for $i = 1, 2, \ldots, p-1$). By constructing off-diagonal blocks with prescribed rank r (no approximation of the off-diagonal blocks, $\tau_1 = 0$) we are able to isolate the influence of the deflation parameter τ_2 on the runtimes. Figure 4.3 shows the runtimes from Table 4.1 as ratios $T_{\mathrm{BT}}/T_{\mathrm{LF}}$ of the runtimes T_{BT} for `dsbtdc` and T_{LF} for the routine `LAPACK/dsyevd`.

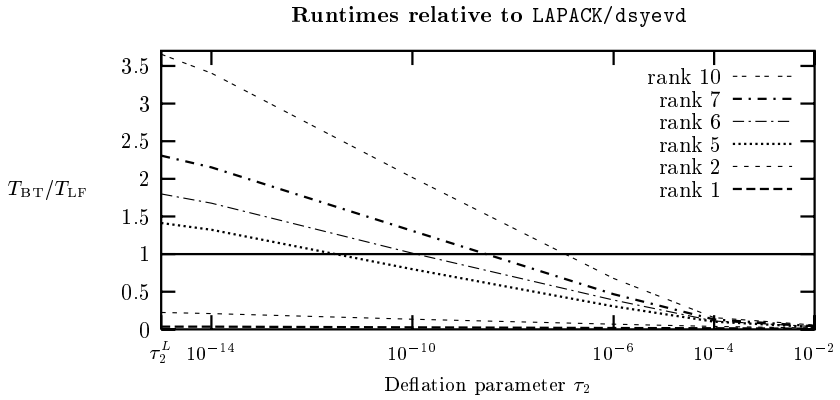

FIG. 4.3. *Runtimes of* `dsbtdc` *relative to* `LAPACK/dsyevd` *for different ranks of the off-diagonal blocks and for varying deflation parameter.*

TABLE 4.2

Eigenvalue errors $\mathcal{E}$, scaled residuals $\mathcal{R}$, and departure from orthogonality $\mathcal{O}$ of the eigenpairs which were computed with reduced accuracy for each matrix M_{300}^r, $r = 1, 2, 5, 6, 7, 10$ (no rank approximation, deflation parameter $\tau_2 = 10^{-6}$).

$\tau_2 = 10^{-6}$	M_{300}^1	M_{300}^2	M_{300}^5	M_{300}^6	M_{300}^7	M_{300}^{10}
`dsbtdc`						
$\mathcal{E}$	$2.6 \cdot 10^{-7}$	$7.7 \cdot 10^{-7}$	$1.7 \cdot 10^{-6}$	$2.1 \cdot 10^{-6}$	$2.3 \cdot 10^{-6}$	$5.0 \cdot 10^{-6}$
$\mathcal{R}$	$8.2 \cdot 10^{-7}$	$1.5 \cdot 10^{-6}$	$2.3 \cdot 10^{-6}$	$2.0 \cdot 10^{-6}$	$2.4 \cdot 10^{-6}$	$2.5 \cdot 10^{-6}$
$\mathcal{O}$	$2.6 \cdot 10^{-15}$	$3.8 \cdot 10^{-15}$	$5.1 \cdot 10^{-15}$	$6.0 \cdot 10^{-15}$	$8.2 \cdot 10^{-15}$	$9.3 \cdot 10^{-15}$

Table 4.2 shows $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{O}$ for the eigenpairs computed by `dsbtdc` for $\tau_1 = 0$, $\tau_2 = 10^{-6}$, and Table 4.3 shows $\mathcal{R}$ and $\mathcal{O}$ for full accuracy computations ($\tau_1 = 0$, $\tau_2 = \tau_2^L$).

Especially for *low rank* off-diagonal blocks `dsbtdc` is significantly faster than the standard methods for banded or general eigenvalue problems, even if the eigensystem is computed to full accuracy. Although `LAPACK/dsyevd` tends to be faster for medium rank off-diagonal blocks and higher accuracy requirements for the problem sizes shown so far, it has higher memory requirements, and, even more important, a higher asymptotic complexity, which is illustrated in Figure 4.4.

4.3. Full SCF method. In each iteration of the SCF method a linear eigenproblem must be solved [32, Chap. 3]. The relevant eigenvectors are the part of the spectrum which corresponds to *occupied* orbitals, and they are denoted by $V_1 \in \mathbb{R}^{n \times o}$. V_1 represents the coefficients of a basis expansion of the unknown wave functions. In the type of application we consider, a sizable portion of the orbitals is occupied ($o \approx n/2$). It is not required to compute individual eigenvectors (columns of V_1) to the prescribed accuracy. It suffices if the *density matrix*

$$P = V_1 V_1^\top \tag{4.1}$$

is approximated accurately enough in each iteration. Either P itself or the total system energy is used in the convergence criterion of the SCF method [32, Chap. 3].

To determine the efficiency of the algorithm described in this paper inside the SCF method, we use as a test case a linear $C_{322}H_{646}$ alkane molecule. The complete

TABLE 4.3

Scaled residuals $\mathcal{R}$ and departure from orthogonality $\mathcal{O}$ of the eigenpairs which were computed to full accuracy for each matrix M_{300}^r, $r = 1, 2, 5, 6, 7, 10$ (no rank approximation, deflation parameter $\tau_2 = \tau_2^L$).

$\tau_2 = \tau_2^L$	M_{300}^1	M_{300}^2	M_{300}^5	M_{300}^6	M_{300}^7	M_{300}^{10}
`dsbtdc`						
$\mathcal{R}$	$4.0 \cdot 10^{-15}$	$6.5 \cdot 10^{-15}$	$7.4 \cdot 10^{-15}$	$1.1 \cdot 10^{-14}$	$1.3 \cdot 10^{-14}$	$1.5 \cdot 10^{-14}$
$\mathcal{O}$	$3.6 \cdot 10^{-15}$	$4.5 \cdot 10^{-15}$	$7.4 \cdot 10^{-15}$	$6.0 \cdot 10^{-15}$	$6.2 \cdot 10^{-15}$	$4.6 \cdot 10^{-15}$
`dsbevd`						
$\mathcal{R}$	$8.4 \cdot 10^{-15}$	$9.8 \cdot 10^{-15}$	$7.6 \cdot 10^{-15}$	$6.7 \cdot 10^{-15}$	$7.6 \cdot 10^{-15}$	$7.3 \cdot 10^{-15}$
$\mathcal{O}$	$5.4 \cdot 10^{-15}$	$5.9 \cdot 10^{-15}$	$5.2 \cdot 10^{-15}$	$4.5 \cdot 10^{-15}$	$5.4 \cdot 10^{-15}$	$4.4 \cdot 10^{-15}$
`dsyevd`						
$\mathcal{R}$	$4.8 \cdot 10^{-15}$	$6.1 \cdot 10^{-15}$	$4.5 \cdot 10^{-15}$	$4.5 \cdot 10^{-15}$	$4.7 \cdot 10^{-15}$	$5.7 \cdot 10^{-15}$
$\mathcal{O}$	$6.3 \cdot 10^{-15}$	$4.8 \cdot 10^{-15}$	$5.8 \cdot 10^{-15}$	$6.2 \cdot 10^{-15}$	$5.6 \cdot 10^{-15}$	$4.8 \cdot 10^{-15}$

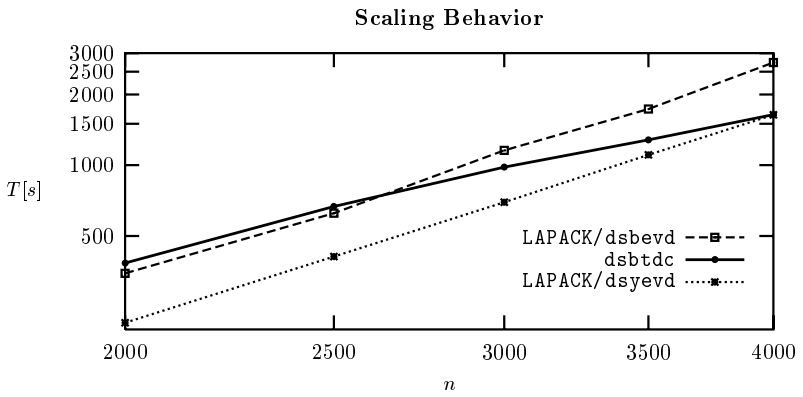

FIG. 4.4. *Runtimes (in seconds) of* `dsbtdc`, `LAPACK/dsbevd`, *and* `LAPACK/dsyevd` *for M_n^5 with $n = 2000, 2500, 3000, 3500, 4000$ ($k_i = 10$, $i = 1, 2, \ldots, p$, $p = 200, 250, 300, 350, 400$), full accuracy (no rank approximation, $\tau_2 = \tau_2^L$), both axes on a logarithmic scale.*

neglect of differential overlap (CNDO) method [26, 27, 28, 25, 24] leads to a Fock matrix of dimension $n = 1934$ each iteration, whose eigenvalues and eigenvectors need to be computed. Figure 4.5 shows a comparison of runtimes for the full SCF method using four different routines for solving all the eigenproblems (`LAPACK/dsyev`, `LAPACK/dsyevd`, `LAPACK/dsbevd`, and `dsbtdc`).

For `dsbtdc`, the same accuracy parameter τ was used in all iterations. Although no comprehensive investigations of different strategies for varying τ with the iterations have been completed so far, the experiments performed seem to indicate that choosing the same value for τ in all iterations gives the shortest runtimes for the specific test case considered.

The band matrix used as input to `LAPACK/dsbevd` was determined by choosing an upper/lower bandwidth b and dropping all the entries outside the resulting band. Clearly, reducing the bandwidth reduces the time required for a single iteration, but increases the approximation error and therefore potentially increases the total number of iterations required (eventually causing loss of convergence of the SCF method). Experimentally, a value of $b = 45$ was determined to yield the approximation quality which results in the best overall runtime for the given problem.

The SCF method was considered converged when the energy difference between

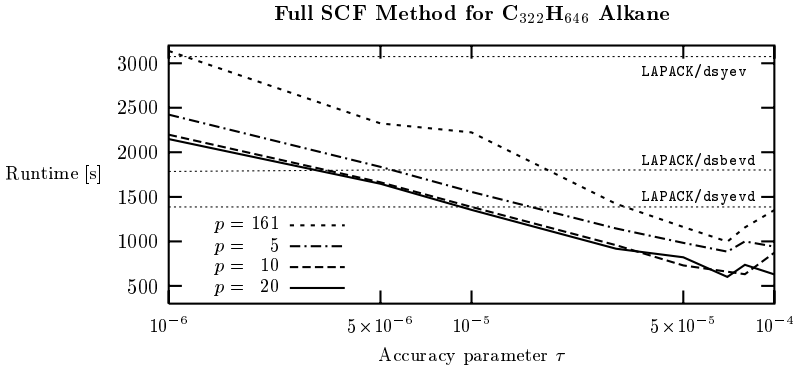

FIG. 4.5. *Runtimes for the full SCF method using different eigensolvers* ($n = 1934$, *runtimes of* `dsbtdc` *shown for different values of* p).

successive iterations was less than 10^{-6}, a commonly used convergence criterion in quantum chemistry. Figure 4.5 shows that using `dsbtdc` with a properly chosen accuracy parameter τ can lead to a significant reduction in the computing time (more than a factor of 2). For larger problems, even higher runtime reductions are to be expected (cf. Figure 4.4).

The fact that the convergence criterion was satisfied even for some values $\tau < 10^{-6}$ indicates that individual eigenpairs, in particular the ones entering the convergence criterion via (4.1), were approximated even better than the general error bounds indicate.

For accuracy parameters $\tau > 10^{-4}$, even shorter runtimes were observed in a few cases. However, in this region convergence behavior was irregular. In many cases, the SCF method did not converge at all within 25 iterations. In some cases, convergence occurred for a certain value $\tau^* > 10^{-4}$, while the iteration did not converge for some neighboring values $10^{-4} < \tau < \tau^*$. This effect is most likely due to the fact that individual eigenvector approximations cannot be guaranteed to have any accuracy if τ is larger than some gap in the spectrum (see section 2.4.5). In that case, the corresponding eigenvalues are considered identical, and a basis of the eigenspace is approximated instead of certain individual eigenvectors. If these eigenvectors do not correspond to occupied orbitals, then the relevant wave functions and the density matrix P may still be approximated sufficiently accurately (cf. (4.1)), and the SCF method still converges.

5. Conclusion. A divide-and-conquer based method for approximating eigenpairs of symmetric block tridiagonal matrices has been proposed. The central ideas which allow one 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 the proposed method is numerically stable and competitive. Especially for medium and low accuracy requirements it is more efficient than the standard methods used in LAPACK. These performance improvements are due to the ability of taking full advantage of lower accuracy requirements and also are due to improved data-locality, which is important for the memory hierarchies of modern computer systems and also extremely important for any implementation on a parallel

computer.

Future work. In order to complete a framework for approximating eigenpairs of arbitrary symmetric matrices, we are investigating several approaches for approximating full matrices by block tridiagonal matrices of the general form (1.1).

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

Acknowledgments. We would like to thank Professor Stanley Eisenstat for pointing out how the optimal merging order for off-diagonal blocks with equal ranks can be determined relatively easily as well as Robert Day for his help in performing some of the experiments. Also, we would like to express our gratitude to the anonymous referees, whose comments and suggestions greatly helped us to improve the quality of this paper, in particular the discussion of numerical stability.

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*, 3rd ed., SIAM, Philadelphia, 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] Y. Bai, W. N. Gansterer, and R. C. Ward, *Block Tridiagonalization of "Effectively" Sparse Symmetric Matrices*, Technical report UT-CS-02-492, Department of Computer Science, University of Tennessee, Knoxville, TN, 2002; also available online from http://www.cs.utk.edu/~library/techreportsmain.php; ACM Trans. Math. Software, submitted.

[6] J. L. Barlow, *Error analysis of update methods for the symmetric eigenvalue problem*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 598–618.

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

[8] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, *Rank-one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

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

[10] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

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

[12] 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, Lecture Notes in Computer Science 1573, J. M. L. M. Palma, J. J. Dongarra, and V. Hernandez, eds., Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1998, pp. 20–28.

[13] 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, Vienna, Austria, 1999; also available online from http://www.vcpc.univie.ac.at/aurora/publications/.

[14] W. N. Gansterer, J. Schneid, and C. W. Ueberhuber, *A low-complexity divide-and-conquer method for computing eigenvalues and eigenvectors of symmetric band matrices*, BIT, 41 (2001), pp. 967–976.

[15] W. N. Gansterer and R. C. Ward, *Computing Approximate Eigenpairs of Symmetric Block Tridiagonal Matrices*, Technical report UT-CS-01-463, Department of Computer Science, University of Tennessee, Knoxville, TN, 2001; also available online from http://www.cs.utk.edu/~library/techreportsmain.php.

[16] 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*, ACM Trans. Math. Software, 28 (2002), pp. 45–58.

[17] K. Gates and P. Arbenz, *Parallel Divide and Conquer Algorithms for the Symmetric Tridiagonal Eigenproblem*, Technical report 222, Institut für Wissenschaftliches Rechnen, ETH Zürich, Zürich, Switzerland, 1994; also available online from http://www.inf.ethz.ch/research/publications/.

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

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

[20] 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.

[21] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.

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

[23] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1997.

[24] J. A. Pople and D. L. Beveridge, *Approximate Molecular Orbital Theory*, 1st ed., McGraw–Hill, New York, 1970.

[25] J. A. Pople, D. L. Beveridge, and P. A. Dobosh, *Approximate self-consistent molecular orbital theory.* V. *Intermediate neglect of differential overlap.*, J. Chem. Phys., 47 (1967), p. 2026.

[26] J. A. Pople, D. P. Santry, and G. A. Segal, *Approximate self-consistent molecular orbital theory.* I. *Invariant procedures.*, J. Chem. Phys., 43 (1965), p. S129.

[27] J. A. Pople and G. A. Segal, *Approximate self-consistent molecular orbital theory.* II. *Calculations with complete neglect of differential overlap.*, J. Chem. Phys., 43 (1965), p. S136.

[28] J. A. Pople and G. A. Segal, *Approximate self-consistent molecular orbital theory.* III. *CNDO results for ab*2 *and ab*3 *systems.*, J. Chem. Phys., 44 (1966), p. 3289.

[29] 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; also available online from http://www.netlib.org/lapack/lawns/downloads/.

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

[31] 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.

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

[33] F. Tisseur and 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.

[34] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.