# AN EXTENSION OF THE DIVIDE-AND-CONQUER METHOD FOR A CLASS OF SYMMETRIC BLOCK-TRIDIAGONAL EIGENPROBLEMS

WILFRIED N. GANSTERER*§, ROBERT C. WARD*§, AND RICHARD P. MULLER†§

Technical Report UT-CS-00-447 [1]

University of Tennessee

September 2000

**Abstract.** A divide-and-conquer method for computing eigenvalues and eigenvectors of a block-tridiagonal matrix with rank-one off-diagonal blocks is presented. The implications of unbalanced merging operations due to unequal block sizes are analyzed and illustrated with numerical examples. It is shown that an unfavorable order for merging blocks in the synthesis phase of the algorithm may lead to a significant increase of the arithmetic complexity. A strategy to determine a good merging order which is at least close to optimal in all cases is given.

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

$$M_p := \begin{pmatrix} B_1 & E_1^\top & & & \\ E_1 & B_2 & E_2^\top & & \\ & E_2 & B_3 & \ddots & \\ & & \ddots & \ddots & E_{p-1}^\top \\ & & & E_{p-1} & B_p \end{pmatrix} \in \mathbb{R}^{n\times n}, \tag{1.1}$$

where 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 $E_i \in \mathbb{R}^{k_{i+1}\times k_i}$ $(i = 1, 2, \ldots, p-1)$ have rank one:

$$E_i = \sigma_i u_i v_i^\top$$

with $\|u_i\|_2 = \|v_i\|_2 = 1$ and $\sigma_i > 0$. The block sizes $k_i$ have to satisfy $1 \le k_i < n$ and $\sum_{i=1}^p k_i = n$.

Many applications, e.g., the self-consistent-field procedure in Quantum Chemistry [20], yield computational matrix eigenproblems with the property that the magnitudes of the matrix elements rapidly decrease as they move away from the diagonal; thus, they can be approximated by matrices of the form (1.1). Although the off-diagonal blocks $E_i$ of the matrices arising in these problems are in general not rank-one matrices, it is possible to approximate them with rank-one matrices, and in many applications, the approximations may be sufficient for the desired accuracy.

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

†Materials and Process Simulation Center, Beckman Institute, California Institute of Technology 139-74, Pasadena, California, 91125

§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

**Related Work.** 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 computing the spectral decomposition of a rank-one modification of a diagonal matrix which has been given in [12, 6]. Over time numerically stable and efficient implementations of Cuppen's method were developed [9, 19, 14, 15, 18].

The divide-and-conquer approach not only has attractive parallelization properties [21, 11], 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].

Several variants of generalizing the divide-and-conquer approach to banded matrices have been investigated ([2] based on [4, 3]; more recently [10]). One of the central questions remains numerical stability and although promising advances have been made no final method has been established (yet).

**2. Mathematical Concept.** The algorithm presented here is an extension of Cuppen's tridiagonal divide-and-conquer method for computing eigenvalues and eigenvectors of the symmetric *block-tridiagonal* matrix (1.1). Three phases can be distinguished: (i) subdivision, (ii) solution of subproblems, and (iii) synthesis of the solutions of the subproblems. These phases are illustrated in more detail in Sections 2.1 to 2.3. In order to simplify the presentation of the basic concepts, the special case $p = 2$ (two diagonal blocks) is considered first. The general case $p > 2$ is discussed in Section 2.4.

**2.1. Subdivision.** Firstly, the matrix

$$M_2 := \begin{pmatrix} B_1 & E_1^\top \\ E_1 & B_2 \end{pmatrix} = \begin{pmatrix} B_1 & \sigma_1 v_1 u_1^\top \\ \sigma_1 u_1 v_1^\top & B_2 \end{pmatrix}$$

is represented as a rank-one modification of a block diagonal matrix $\tilde{M}_2$:

$$\begin{aligned} M_2 &= \begin{pmatrix} B_1 - \sigma_1 v_1 v_1^\top & \mathbf{0} \\ \mathbf{0} & B_2 - \sigma_1 u_1 u_1^\top \end{pmatrix} + \sigma_1 \begin{pmatrix} v_1 \\ u_1 \end{pmatrix} \begin{pmatrix} v_1^\top & u_1^\top \end{pmatrix} \\ &= \tilde{M}_2 + \sigma_1 \begin{pmatrix} v_1 \\ u_1 \end{pmatrix} \begin{pmatrix} v_1^\top & u_1^\top \end{pmatrix}. \end{aligned} \tag{2.1}$$

**2.2. Solution of Subproblems.** Secondly, the spectral decompositions of the decoupled diagonal blocks

$$\tilde{B}_1 := B_1 - \sigma_1 v_1 v_1^\top, \quad \tilde{B}_2 := B_2 - \sigma_1 v_1 v_1^\top$$

of $\tilde{M}_2$ can be computed independently:

$$\tilde{B}_1 = Q_1 D_1 Q_1^\top \quad \text{and} \quad \tilde{B}_2 = Q_2 D_2 Q_2^\top. \tag{2.2}$$

Substituting (2.2) into (2.1) yields

$$M_2 = \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} \left[ \begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix} + \sigma_1 z z^\top \right] \begin{pmatrix} Q_1^\top & \\ & Q_2^\top \end{pmatrix} \tag{2.3}$$

where

$$z := \begin{pmatrix} Q_1^\top & \\ & Q_2^\top \end{pmatrix} \begin{pmatrix} v_1 \\ u_1 \end{pmatrix}. \tag{2.4}$$

(2.3) shows that $M_2$ is orthogonally similar to a rank-one modification of the diagonal matrix

$$D := \begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix}.$$

**2.3. Synthesis of the Solution of the Subproblems.** In the third phase of the algorithm, the spectral decomposition

$$D + \sigma_1 z z^\top = Q \Lambda Q^\top \tag{2.5}$$

is computed (i. e., the corresponding diagonal blocks $D_1$ and $D_2$ are "merged"). The method used is identical to the one in the standard divide-and-conquer algorithm for a tridiagonal matrix.

**2.3.1. Eigenvalues of the Synthesis Problem.** After deflation [6, 7, 9] it is guaranteed that the vector $z$ has $k \le n$ nonzero components $\zeta_i$ and that the corresponding $k$ entries $d_i$ of the diagonal matrix $D$ are all distinct. The eigenvalues $\lambda_i$ in (2.5) can now be computed as the solutions of the secular equation [12]

$$f(\lambda) := 1 + \sigma_1 \sum_{i=1}^{k} \frac{\zeta_i^2}{d_i - \lambda} = 0. \tag{2.6}$$

Just like in the tridiagonal divide-and-conquer method, the $d_i$ are poles of $f(\lambda)$, and $f(\lambda)$ is strictly increasing between these poles. Moreover, its zeros, the eigenvalues $\lambda_i$, interlace the $d_i$. Equation (2.6) is solved with a usually very fast converging version of Newton's method which uses a rational approximation of $f(\lambda)$ between two poles $d_i$ and $d_{i+1}$ ([16, 6], also see the routine `LAPACK/*laed4`).

**2.3.2. Eigenvectors of the Synthesis Problem.** As shown in [6], the eigenvector $q_i$ of (2.5) corresponding to the eigenvalue $\lambda_i$ is given by

$$q_i = (D - \lambda_i I)^{-1} z. \tag{2.7}$$

However, using (2.7) for computing the eigenvectors of (2.5) is not numerically stable and can lead to a loss of numerical orthogonality if the corresponding eigenvalues $\lambda_i$ and $\lambda_j$ are very close to each other [7, 9, 19]. Therefore, a slightly modified, numerically stable method introduced by Gu and Eisenstat [14] has to be used, exactly as in the tridiagonal divide-and-conquer method. The central idea is to apply formula (2.7) to a nearby problem $D + \bar{z}\bar{z}^\top$ with a modification vector $\bar{z}$ which can be computed efficiently to componentwise high relative accuracy and for which the computed eigenvalues are "exact". A theorem by Löwner [17] shows how to compute the vector $\bar{z}$ from the $d_i$ and the $\lambda_i$.

Substituting the spectral decomposition (2.5) into (2.3) reveals that the diagonal matrix $\Lambda$ contains the desired eigenvalues of $M_2$ and that its eigenvectors $V$ can be computed according to

$$V = \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} Q. \tag{2.8}$$

**2.4. The General Case.** The main differences between the algorithm for $p = 2$ illustrated in Sections 2.1 to 2.3 and the general case $p > 2$ lie in the subdivision and synthesis phases. After a discussion of the three phases for the general case, the complete block-tridiagonal divide-and-conquer method is summarized in Algorithm 2.1.

**2.4.1. Subdivision.** With the corrections

$$\begin{aligned}\tilde{B}_1 &:= B_1 - \sigma_1 v_1 v_1^\top\\ \tilde{B}_i &:= B_i - \sigma_{i-1} u_{i-1} u_{i-1}^\top - \sigma_i v_i v_i^\top, \quad i = 2, 3, \ldots, p-1\\ \tilde{B}_p &:= B_p - \sigma_{p-1} u_{p-1} u_{p-1}^\top,\end{aligned}$$

$M_p$ can be represented as a series of $p-1$ rank-one modifications of a block-diagonal matrix $\tilde{M}_p$, with the $\tilde{B}_i$ $(i = 1, 2, \ldots, p)$ along its diagonal.

**2.4.2. Solution of Subproblems.** 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 $\tilde{M}_p$ make it possible to construct a diagonal matrix

$$D := \begin{pmatrix} D_1 & & \\ & \ddots & \\ & & D_p \end{pmatrix}$$

and a block-diagonal matrix

$$X := \begin{pmatrix} Q_1 & & \\ & \ddots & \\ & & Q_p \end{pmatrix}$$

such that

$$D + \sigma_1 X^\top \begin{pmatrix} v_1 \\ u_1 \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} v_1^\top & u_1^\top & \mathbf{0} \end{pmatrix} X + \ldots + \sigma_i X^\top \begin{pmatrix} \mathbf{0} \\ v_i \\ u_i \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{0} & v_i^\top & u_i^\top & \mathbf{0} \end{pmatrix} X + \ldots$$

$$\ldots + \sigma_{p-1} X^\top \begin{pmatrix} \mathbf{0} \\ v_{p-1} \\ u_{p-1} \end{pmatrix} \begin{pmatrix} \mathbf{0} & v_{p-1}^\top & u_{p-1}^\top \end{pmatrix} X$$

is orthogonally similar to $M_p$.

**2.4.3. Synthesis of the Solution of the Subproblems.** The eigenvalues and eigenvectors of

$$D + \sum_{i=1}^{p-1} \sigma_i X^\top \begin{pmatrix} \mathbf{0} \\ v_i \\ u_i \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{0} & v_i^\top & u_i^\top & \mathbf{0} \end{pmatrix} X, \tag{2.9}$$

where the norm of each modification vector equals $\sqrt{2}$, have to be computed. This can be done by successively performing rank-one modifications, which corresponds to merging neighboring diagonal blocks of $D$. In total, $p-1$ merging operations are required. Each of them combines two diagonal blocks of $D$, accounting for one rank-one modification in (2.9), and involves the update (2.4) of the modification vector, the solution of the corresponding secular equation (2.6), the computation of the corresponding eigenvectors and the update (2.8) of the eigenvector matrix. In the course of this process, the blocks to be merged grow bigger and bigger.

The synthesis process can be pictured as a binary tree, the *merging tree*, with the blocks $\tilde{B}_1, \tilde{B}_2, \ldots, \tilde{B}_p$ as leaves. If $p$ is a power of two, this tree is balanced. If $p$ is not a power of two, a merging order can be determined which, in a "preprocessing phase", reduces the number of blocks to the closest power of two less than $p$.

A balanced merging tree is obviously attractive for a parallel implementation since it provides natural task parallelism (see [9, 13]). However, for unequally sized blocks, the best order for merging blocks (in terms of efficiency) does not necessarily correspond to a balanced tree. The reason is that even if the tree is balanced with respect to the number of nodes (possibly after the "preprocessing" mentioned above) unequal sizes of the submatrices in the nodes may make the tree highly unbalanced workwise. For a *sequential* algorithm, it is irrelevant whether the merging tree is balanced or not because the independence of different merging operations cannot be taken advantage of. Aspects related to the merging order in the synthesis phase are discussed in more detail in Section 3.

ALGORITHM 2.1. **Block-Tridiagonal Divide-and-Conquer**
**Input:** *$M_p$ as in (1.1)*
**(A) Subdivision**

$$\tilde{B}_1 := B_1 - \sigma_1 v_1 v_1^\top,\ \tilde{B}_p := B_p - \sigma_{p-1} u_{p-1} u_{p-1}^\top,\ z_1 := \begin{pmatrix} v_1 \\ u_1 \end{pmatrix}$$

**do** $i = 2, 3, \ldots, p-1$

$$\tilde{B}_i := B_i - \sigma_{i-1} u_{i-1} u_{i-1}^\top - \sigma_i v_i v_i^\top,\ z_i := \begin{pmatrix} v_i \\ u_i \end{pmatrix}$$

**end do**

**(B) Solution of Subproblems**

**do** $i = 1, 2, \ldots, p$

*compute the spectral decomposition* $\tilde{B}_i = Q_i D_i Q_i^\top$

**end do**

$X :=$ *block-diag* $(Q_1, Q_2, \ldots, Q_p)$

**(C) Synthesis**

*determine a good merging order (see Section 3.4);*
*denote the ordered modification vectors by* $z_{(i)}$,
*the corresponding parts of* $X$ *and* $D$ *by* $X_{(i)}$ *and* $D_{(i)}$

**do** $i = 1, 2, \ldots, p-1$

*(i) update the modification vector* $z_{(i)}$

$$\tilde{z}_{(i)} := X_{(i)}^\top \begin{pmatrix} \mathbf{0} \\ z_{(i)} \\ \mathbf{0} \end{pmatrix}$$

*(ii) decompose* $\quad D_{(i)} + \sigma_{(i)} \tilde{z}_{(i)} \tilde{z}_{(i)}^\top = Q \Lambda_{(i)} Q^\top$

*(iii) update the eigenvectors* $\quad X := X \begin{pmatrix} I & & \\ & Q & \\ & & I \end{pmatrix}$

**end do**

**Output:** *spectral decomposition* $\quad M_p = V \Lambda_{(p-1)} V^\top$ *with* $V := X$

FIG. 2.1. *The divide-and-conquer algorithm for block-tridiagonal matrices with rank-one off-diagonal blocks.*

**3. Unbalanced Merging Operations.** In the standard divide-and-conquer method for a given irreducible tridiagonal matrix the block sizes $k_i$ can be chosen by the user or by the algorithm. Usually this is done by breaking the original matrix in two and then repeatedly halving the sizes of the submatrices. Except for potential differences in the early stages of the synthesis phase, the block sizes are therefore essentially equal [18, 21]. The situation considered in this paper is slightly more complicated because the sizes $k_i$ of the diagonal blocks $B_i$ are assumed to be determined a priori (as indicated in (1.1)) and may be unequal.

Unequal block sizes influence two important aspects of the algorithm presented: its arithmetic complexity and its workspace requirements. Although the sequence of block sizes $\{k_i\}_{i=1}^{p}$ is given a priori, there is some freedom in choosing the *order* in which (neighboring) blocks are to be merged. It will be illustrated in Sections 3.1 and 3.2 that for a sequential algorithm it is beneficial to choose the merging order such that the merging operations are as "balanced" as possible.

In Sections 3.1 and 3.2, we investigate the arithmetic complexity of a single merging operation involving a $c \times c$ block $\tilde{A}_1$ and an $(l-c) \times (l-c)$ block $\tilde{A}_2$ where $1 \le c \le l-1$. $c$ will also be called the *cut point* between the two blocks. If $c \approx [l/2]$ the merging operation is called *balanced*, if $c \ll [l/2]$ or $c \gg [l/2]$ it is called *unbalanced*. The nonzero portions $u$ and $v$ of the associated correction vector $z$ are assumed to be of size $b_1$ and $b_2$, respectively (thus, $b_1 = k_{i+1}$ and $b_2 = k_i$ for some $i$).

If the merging operation occurs at the beginning of the merging process, i. e., if it involves two original (neighboring) blocks of (1.1) which are leaves of the merging tree, then $b_1 = c$ and $b_2 = l - c$. If the merge occurs at a later stage of the merging process, i. e., if it involves two blocks which are themselves the result of at least one merging operation, then $1 \le b_1 < c$ and $1 \le b_2 < l-c$. In the final merging operation, $l = n$.

The effects of the merging order in the context of the full synthesis phase for $p > 2$ are illustrated in Section 3.3 by comparing a very bad merging order (unbalanced merges) with a very good merging order (balanced merges) for an example problem. In Section 3.4 an approach for determining a good merging order will be described.

**3.1. Arithmetic Complexity.** For a given sequence $\{k_i\}_{i=1}^{p}$ of block sizes the flop counts of several operations of Algorithm 2.1 are independent of the cut points $c$. This includes the correction operations in Step A and the solution of the subproblems in Step B. The flop count for solving a secular equation (2.6) and for computing the eigenvectors of the synthesis problem in Step C.ii can also be considered independent of the cut point $c$. There are, however, two operations in the synthesis phase, whose flop counts depend on $c$.

1. Update of the modification vector $z$ (cf. Step C.i in Algorithm 2.1): The matrix-vector products

$$Q_1^\top \begin{pmatrix} \mathbf{0} \\ v \end{pmatrix} \quad \text{and} \quad Q_2^\top \begin{pmatrix} u \\ \mathbf{0} \end{pmatrix}$$

require

$$c(2b_2 - 1) + (l-c)(2b_1 - 1) = 2c(b_2 - b_1) + l(2b_1 - 1) \quad \text{flops.} \tag{3.1}$$

Considered separately, expression (3.1) would favor unbalanced merging operations, i. e., $c$ as small or as large as possible, depending on whether $b_2 - b_1$ is greater or less than zero.

2. Multiplication of the intermediate eigenvectors $X$ with the eigenvectors $Q$ of the synthesis problem (cf. Step C.iii in Algorithm 2.1):
Assuming for simplicity that no deflation has occurred (which implies that we compute an upper bound for the arithmetic complexity of this operation), the $c \times c$ block of eigenvectors of $\tilde{A}_1$ has to be multiplied with the upper $c \times l$ part of the eigenvector matrix $Q$ of the synthesis problem which requires

$$cl(2c-1) \quad \text{flops.} \tag{3.2}$$

Multiplication of the $(l-c) \times (l-c)$ block of eigenvectors of $\tilde{A}_2$ with the lower $(l-c) \times l$ part of the eigenvector matrix $Q$ of the synthesis problem requires

$$(l-c)l\,(2\,(l-c)-1) \quad \text{flops.} \tag{3.3}$$

Adding up (3.1), (3.2), and (3.3) yields $W(c)$, an upper bound for the portion of the arithmetic complexity of a single merging operation depending on the cut point $c$:

$$W(c) = 4c^2 l + c\left(2\,(b_2 - b_1) - 4l^2\right) + 2l^3 - l^2 + l\,(2b_1 - 1) \quad \text{flops.} \tag{3.4}$$

Asymptotically, the updates of the eigenvectors (Step C.iii) are the most expensive parts of the synthesis phase and also of the entire divide-and-conquer algorithm 2.1 (cf. [8, 10]). Therefore, the dominating terms of the arithmetic complexity of the full algorithm can be computed based on $W(c)$ (see Section 3.3.1).

Investigation of the first and second derivative of $W(c)$ in terms of $c$ reveals that $W(c)$ has a minimum at

$$c^* = \frac{l}{2} - \frac{b_2 - b_1}{4l} \tag{3.5}$$

with the value

$$W(c^*) = l^3 - l^2 + l\,(b_1 + b_2 - 1) - \frac{(b_2 - b_1)^2}{4l}. \tag{3.6}$$

Note that $|b_2 - b_1| < l$, thus

$$\left|\frac{b_2 - b_1}{4l}\right| < \frac{1}{4},$$

and therefore (3.5) implies that the closest integer value to $c^*$ is $l/2$ if $l$ is even and either $(l+1)/2$ or $(l-1)/2$ if $l$ is odd. This means that the *optimum cut point* for a single merging operation in terms of arithmetic complexity is either $\lfloor l/2 \rfloor$ or $\lfloor (l+1)/2 \rfloor$.

Comparison of the highest order terms in (3.4) and (3.6),

$$W(c) \approx 2l^3 - (4c+1)\,l^2 + 4c^2 l \tag{3.7}$$

$$W(c^*) \approx l^3 - l^2 \tag{3.8}$$

shows that asymptotically the "penalty" to be paid for an unbalanced merging operation in which the cut point deviates from $c^*$ is

$$\frac{W(c)}{W(c^*)} \approx \frac{2l}{l-1} - \frac{4c+1}{l-1} + \frac{4c^2}{l(l-1)}.$$

This quadratic function in $c$ has the minimum value 1 at $c = c^* = l/2$. Its maximum value occurs at the boundaries, i.e., for $c = 1$ or $c = l - 1$:

$$\frac{W(1)}{W(c^*)} = \frac{W(l-1)}{W(c^*)} = \frac{2l}{l-1} - \frac{5}{l-1} + \frac{4}{l(l-1)}. \tag{3.9}$$

For large $l$, (3.9) approaches the value 2. Thus,

$$W(c^*) \leq W(c) \leq 2W(c^*).$$

**3.2. Workspace Requirements.** The implementation of the block-tridiagonal divide-and-conquer algorithm is based on the LAPACK [1] implementation of the tridiagonal divide-and-conquer algorithm (`LAPACK/*stedc` and dependencies). Several routines had to be modified, some others could be used directly. New aspects arise from unequal block sizes. The routine `LAPACK/*laed1` formally requires $\min(1, l) \leq c \leq l/2$ which cannot be guaranteed in the block-tridiagonal divide-and-conquer algorithm (it may well happen that the first of two blocks involved in a merging operation is larger than the second one).

In the following, we give an analysis of the total workspace requirements of our implementation of the block-tridiagonal divide-and-conquer algorithm, which are strongly influenced by the design of the LAPACK routines (see [18]). It is shown that highly unbalanced merging operations may significantly increase the workspace requirements. This observation again underlines the benefits of a merging order where merging operations are as balanced as possible (see Section 3.4).

**3.2.1. Workspace for Vectors.** Workspace of $S_v := 3n$ floating point numbers is required for storing intermediate vectors, just like in the original LAPACK routines.

**3.2.2. Eigenvectors Before the Merging Operation.** As in Algorithm 2.1 let $X$ denote the matrix containing the eigenvectors of the two subproblems to be merged. The workspace $S_X$ required for storing $X$ is

$$S_X := c(c_1 + c_2) + (l - c)(c_2 + c_3) \tag{3.10}$$

floating point numbers. Here, $c_1$ denotes the number of columns in $X$ which are of "type 1" (nonzero in the upper part only), $c_2$ denotes the number of columns in $X$ which are of "type 2" (dense), and $c_3$ denotes the number of columns in $X$ which are of "type 3" (nonzero in the lower part only).

Before deflation, $c_1 = c$, $c_2 = 0$, and $c_3 = l - c$. Columns of type 2 are created when (almost) equal eigenvalues are deflated which appear in the upper *and* in the lower block: The Householder/Givens transformations $G$ with $GG^\top = I$ used to zero out corresponding entries in the $\tilde{z}$-vector

$$X\left(D + \tilde{z}\tilde{z}^\top\right)X^\top = XG^\top\left(GDG^\top + G\tilde{z}\tilde{z}^\top G^\top\right)GX^\top$$

may cause a fill-in in the upper/lower part of $X$, because the operation $XG^\top$ forms linear combinations of the columns of $X$ which correspond to (almost) equal eigenvalues. If these happen to be in different blocks of $D$, then the lower/upper parts of the corresponding columns of $X$ fill up.

This implies that an upper bound on $c_2$ is given by the total number of eigenvalues deflated:

$$c_2 \leq l - k, \quad 1 \leq k \leq l,$$

where $k$ denotes the number of non-deflated eigenvalues. Every deflated eigenvalue may potentially add two columns to $c_2$ (by filling up the upper part and the lower part of two columns of $X$) while taking away one column from $c_1$ and $c_3$ each. Thus,

$$c_1 + c_2 \leq c + \frac{l-k}{2} \quad \text{and} \quad c_2 + c_3 \leq l - c + \frac{l-k}{2}. \tag{3.11}$$

Substituting (3.11) into (3.10) yields

$$S_X \leq -\frac{kl}{2} + \frac{l^2}{2} + c^2 + (l-c)^2. \tag{3.12}$$

Note that less deflation (larger $k$) implies a lower bound for the workspace requirements $S_X$.

**3.2.3. Multiplication of the Eigenvector Matrices.** The matrix multiplication in the routine `LAPACK/*laed3` for establishing the updated (intermediate) eigenvector matrix $X$ (Step C.iii in Algorithm 2.1) requires a workspace of

$$S_S := \max\left\{(c_1 + c_2)\, k, (c_2 + c_3)\, k\right\}$$

floating point numbers for copying row blocks of $Q$ consisting of the first $c_1 + c_2$ and the last $c_2 + c_3$ rows, respectively, into a temporary array $S$ (see also [18]). The bounds (3.11) imply

$$S_S \leq k \left( \max\{c, l-c\} + \frac{l-k}{2} \right).$$

**Case I:** $c \leq l - c \quad \Leftrightarrow \quad c \leq l/2$ leads to

$$S_S \leq -\frac{k^2}{2} + k\left(\frac{3l}{2} - c\right). \tag{3.13}$$

**Case II:** $c \geq l - c \quad \Leftrightarrow \quad c \geq l/2$ leads to

$$S_S \leq -\frac{k^2}{2} + k\left(\frac{l}{2} + c\right). \tag{3.14}$$

Note that in both cases, less deflation (larger $k$) also implies a lower bound on the workspace requirements $S_S$.

**3.2.4. Total Workspace Requirements.**
**Case I:** $1 \leq c \leq l/2$

(3.12) and (3.13) imply

$$S_X + S_S \leq -\frac{k^2}{2} + k\,(l-c) + \frac{l^2}{2} + c^2 + (l-c)^2.$$

Investigation of the first and second derivative of this upper bound in terms of $k$ reveals that it achieves a maximum if $c$ eigenvalues are deflated, i. e. for

$$k^* = l - c.$$

Therefore, a sharp upper bound for $S_X + S_S$ independent of how much deflation occurs is

$$S_X + S_S \leq \frac{l^2}{2} + \frac{3\,(l-c)^2}{2} + c^2. \tag{3.15}$$

Investigation of the first and second derivative in terms of $c$ reveals that the quadratic function on the right hand side of (3.15) has a minimum for $c = 3l/5$. This value is not within the range considered, and therefore the actual minimum occurs at the upper boundary, for

$$c^* = \frac{l}{2} \tag{3.16}$$

with the function value

$$\frac{9l^2}{8}. \tag{3.17}$$

At the lower boundary (for $c = 1$), the upper bound (3.15) achieves its maximum $2l^2 - 3l + 5/2$, and therefore the best general upper bound for $S_X + S_S$ independent of $k$ and $c$ is given as

$$S_X + S_S \leq 2l^2 - 3l + 3. \tag{3.18}$$

**Case II:** $l/2 \leq c \leq l - 1$
(3.12) and (3.14) imply

$$S_X + S_S \leq -\frac{k^2}{2} + kc + \frac{l^2}{2} + c^2 + (l-c)^2 .$$

Investigation of the first and second derivative of this upper bound in terms of $k$ reveals that it achieves a maximum if $l - c$ eigenvalues are deflated, i. e. for

$$k^* = c.$$

Therefore, a sharp upper bound for $S_X + S_S$ independent of how much deflation occurs is

$$S_X + S_S \leq \frac{l^2}{2} + (l-c)^2 + \frac{3c^2}{2}. \tag{3.19}$$

Investigation of the first and second derivative in terms of $c$ reveals that the quadratic function on the right hand side of (3.19) has a minimum for $c = 2l/5$. This value is not within the range considered, and therefore the actual minimum occurs at the lower boundary, for

$$c^* = \frac{l}{2} \tag{3.20}$$

with the function value

$$\frac{9l^2}{8}. \tag{3.21}$$

At the upper boundary (for $c = l - 1$), the upper bound (3.19) achieves its maximum $2l^2 - 3l + 5/2$, and therefore also in this case the best general upper bound $S_X + S_S$ independent of $k$ and $c$ is given as

$$S_X + S_S \leq 2l^2 - 3l + 3. \tag{3.22}$$

(3.16) and (3.20) show that the cut point $c^*$ which yields the lowest workspace requirements over all possible deflation scenarios is essentially the same which minimizes the arithmetic complexity of the merging operation as discussed in Section 3.1.

**3.3. The General Case.** After having investigated the influence of the cut point $c$ on arithmetic complexity and workspace requirements of a single merging operation we now turn to a discussion of the effects in the framework of a general synthesis phase consisting of $p-1$ merging operations.

**3.3.1. Arithmetic Complexity.** Expression (3.4) shows how the arithmetic complexity of updating modification vector and eigenvectors in one merging operation depends on the cut point $c$, and expression (3.6) gives its minimum value for the optimal $c^*$. Utilizing these expressions allows one to illustrate the potentially significant increase of the arithmetic complexity caused by a bad choice of the merging order in the synthesis phase of the full block-tridiagonal divide-and-conquer algorithm.

We consider the following example: $n = 2^q$ and $p = n/2+1 = 2^{q-1}+1$. The block sizes are $k_1 = 2^{q-1}$, $k_2 = k_3 = \dots k_{2^{q-1}+1} = 1$. Two different merging strategies will be compared: in the first one all the merging operations are as unbalanced as possible, whereas in the second one all the merging operations are perfectly balanced. In the analysis, the leading-term approximations (3.7) and (3.8) for the expressions (3.4) and (3.6) are used.

**Unbalanced Merges:** Merge $\tilde{B}_1$ with $\tilde{B}_2$, the resulting block with $\tilde{B}_3$, the resulting block with $\tilde{B}_4$, etc.

This may be considered an unnatural and somewhat artificial merging order for the matrix considered. Nevertheless, the intention is to illustrate the effects of unbalanced merges in a "worst case" scenario. The natural approach for this problem, subdividing between the first large block and the tridiagonal part, then treating the tridiagonal part using the standard divide-and-conquer algorithm for a tridiagonal matrix, and eventually performing one final merging operation with the first block, is investigated later as the alternative strategy called "balanced merges".

For the strategy of unbalanced merges, the intermediate block sizes $l_i$ in the synthesis phase are given as

$$l_i = 2^{q-1} + i, \quad i = 1, \dots, 2^{q-1}, \tag{3.23}$$

and the cut points $c_i$ as

$$c_i = l_i - 1, \quad i = 1, \dots, 2^{q-1}, \tag{3.24}$$

i. e., every single merging operation is as unbalanced as possible.

Substituting (3.23) for $l$ and (3.24) for $c$ into (3.7) and summing over all the merging operations yields an upper bound $W_U$ (because no deflation was assumed) for the total arithmetic complexity of the updates of modification vectors and eigenvectors in the synthesis phase for unbalanced merges:

$$\begin{aligned} W_U &= \sum_{i=1}^{2^{q-1}} \left( 2\left(2^{q-1}+i\right)^3 - 5\left(2^{q-1}+i\right)^2 + 4\left(2^{q-1}+i\right) \right) \\ &= \sum_{i=1}^{n/2} \left( 2i^3 + i^2\,(3n-5) + i\left(\frac{3n^2}{2} - 5n + 4\right) + \frac{n^3 - 5n^2 + 8n}{4} \right) \\ &= \frac{15}{32}n^4 - \frac{7}{12}n^3 + \frac{7}{12}n \end{aligned}$$

It is worth noting that this unfavorable merging order leads to an $O(n^4)$ (!) arithmetic complexity for the special problem considered.

**Balanced Merges:** Merge the small blocks first. Start with $2^{q-2}$ merges of neigboring blocks of size 1 ($\tilde{B}_{2i}$ with $\tilde{B}_{2i+1}$, $i = 1, 2, \dots 2^{q-2}$), then perform $2^{q-3}$ merges of the resulting blocks of size 2, then perform $2^{q-4}$ merges of the resulting blocks of size 4, etc. After $2^{q-1} - 1$ merging operations, all original blocks of size one have been merged into a single block of size $2^{q-1}$. In the final merging operation, this block is merged with $\tilde{B}_1$.

For this merging order, the intermediate block sizes $l_i$ in the synthesis phase are given as

$$l_i = 2^i, \quad i = 1, 2, \dots, q, \tag{3.25}$$

and the cut points $c_i$ as

$$c_i = l_i/2, \quad i = 1, 2, \dots, q,$$

i. e., every single merging operation is perfectly balanced.

Substituting (3.25) for $l$ into (3.8) and summing over all merging operations yields an upper bound $W_B$ (because no deflation was assumed) for the total arithmetic complexity of updates of the modification vectors and eigenvectors in the synthesis phase for balanced merges:

$$\begin{aligned}
W_B &= 2^{q-2}\left(\left(2^1\right)^3 - \left(2^1\right)^2\right) + 2^{q-3}\left(\left(2^2\right)^3 - \left(2^2\right)^2\right) + \dots \\
&\dots + 2\left(\left(2^{q-2}\right)^3 - \left(2^{q-2}\right)^2\right) + \left(2^{q-1}\right)^3 - \left(2^{q-1}\right)^2 + \left(2^q\right)^3 - \left(2^q\right)^2 \\
&= n^3 - n^2 + \sum_{i=1}^{q-1}\left(2^{q-1+2i} - 2^{q-1+i}\right) \\
&= n^3 - n^2 + \frac{n}{2}\sum_{i=1}^{q-1} 4^i - \frac{n}{2}\sum_{i=1}^{q-1} 2^i \\
&= \frac{7}{6}n^3 - \frac{3}{2}n^2 + \frac{1}{3}n
\end{aligned}$$

Thus, the favorable balanced merging operations lead to an $O(n^3)$ arithmetic complexity for the special problem considered, which is an order of magnitude lower than the unbalanced merges.

This example illustrated "worst" and "best" case scenarios. In most practical problems the arithmetic complexities corresponding to different merging orders will usually not differ that much. Nevertheless, for a given sequence of block sizes the arithmetic complexities associated with different merging strategies may still vary significantly. Therefore, the choice of a good merging order in the synthesis phase (see Section 3.4) is crucial for efficiency.

**3.3.2. Workspace Requirements.** The discussion in Section 3.2 was applicable to any single merging operation occurring in the block-tridiagonal divide-and-conquer algorithm. Obviously, the workspace requirements for the entire algorithm are determined by the maximum over all merging operations. This maximum is achieved in the final merging operation where $l = n$.

Accounting for $S_v = 3n$ in (3.18) or (3.22) yields an upper bound for the total floating-point workspace required by the block-tridiagonal divide-and-conquer algorithm:

$$S_v + S_X + S_S \le 2n^2 + 3.$$

The minimum workspace requirements allowing for all possible deflation scenarios may be as low as

$$\frac{9}{8}n^2 + 3n$$

floating point numbers (cf. (3.17) and (3.21)). In the best case—if no deflation happens at all in the last merging operation ($c_2 = 0$), and if $c = n/2$—a workspace of only $n^2 + 3n$ floating point numbers suffices.

**3.4. Determination of a Merging Order.** It has been shown in Sections 3.1, 3.2, and 3.3 that *equally sized* blocks in a merging operation lead to the lowest arithmetic complexity and therefore to the highest efficiency, as well as to the lowest workspace requirements. Therefore, we determine a merging order in the sequential block-tridiagonal divide-and-conquer algorithm such that blocks to be merged differ as little as possible in size.

Merging operations and their eigenvector updates towards the end of the synthesis phase dominate the arithmetic complexity of the entire algorithm (because $l$ is close to $n$). Consequently, highest priority is assigned to minimizing the difference in block sizes in the final merging operation. The cut point which corresponds to the minimum complexity of this final merging operation is determined by finding the index $j$, $1 \leq j \leq p-1$, such that

$$\sum_{i=1}^{j} k_i \leq \frac{n}{2} \quad \text{and} \quad \sum_{i=1}^{j+1} k_i > \frac{n}{2}.$$

The cut points for the previous merges in each of the two parts are determined by continuing this process recursively above and below the final cut point.

Note that this recursive strategy is not optimal in a rigorous sense because it does not minimize the overall arithmetic complexity of the synthesis phase for all possible sequences of block sizes. A simple generic example illustrates this: For a sequence $\{m, 1, 1, m\}$ of block sizes the strategy used would merge the first with the second and the third with the fourth block, resulting in the sequence $\{m+1, m+1\}$, and finally merge two equally sized blocks. Using the leading term approximations (3.7) and (3.8) this amounts to $12m^3 + O(m^2)$ flops. A different merging order would lead to a lower flop count: First merging the two small blocks in the middle, resulting in the sequence $\{m, 2, m\}$, then the first two blocks, resulting in the sequence $\{m+2, m\}$, and finally two blocks of slightly different sizes requires only $10m^3 + O(m^2)$ flops.

However, even in these cases the strategy used tends to be not much more expensive than the theoretical optimum. This fact and the significantly larger overhead for determining an optimal merging order in the rigorous sense led to the decision to use the recursive strategy described above.

**4. Numerical Aspects.** An error analysis of Cuppen's divide-and-conquer algorithm for tridiagonal matrices was given by Barlow [5]. The numerical stability of a slightly modified divide-and-conquer algorithm for tridiagonal matrices was shown by Gu and Eisenstat [14, 15]. Many elements of these analyses can be carried over directly to the algorithm discussed in this paper. It can be seen that the block-tridiagonal divide-and-conquer algorithm computes a numerical spectral decomposition $\hat{V}\hat{\Lambda}\hat{V}^\top$ such that

$$M_p = \hat{V}\hat{\Lambda}\hat{V}^\top + O\left(\varepsilon n \left(\max_{i=1,2,\ldots,p} \|B_i\|_2 + \max_{i=1,2,\ldots,p} \sigma_i\right)\right),$$

where quantities superscripted with a hat distinguish computed quantities from exact quantities and $\varepsilon$ denotes the machine precision (unit roundoff).

The norm of the floating-point error $\delta\tilde{B}_i := \hat{\tilde{B}}_i - \tilde{B}_i$ of each of the correction operations in Step A of Algorithm 2.1 can be bounded by

$$\begin{aligned}
&\|\delta\tilde{B}_1\|_2 \le \varepsilon k_1 \left(2\|B_1\|_2 + 5\sigma_1\right) + O(\varepsilon^2),\\
&\|\delta\tilde{B}_i\|_2 \le \varepsilon k_i \left(2\|B_i\|_2 + 5\sigma_{i-1} + 5\sigma_i\right) + O(\varepsilon^2), \quad i = 2, 3, \ldots, p-1,\\
&\|\delta\tilde{B}_p\|_2 \le \varepsilon k_p \left(2\|B_p\|_2 + 5\sigma_{p-1}\right) + O(\varepsilon^2),
\end{aligned}$$

and therefore the entire floating-point error made in the subdivision phase can be bounded by

$$10\varepsilon n \left( \max_{i=1,2,\ldots,p} \|B_i\|_2 + \max_{i=1,2,\ldots,p} \sigma_i \right) + O(\varepsilon^2).$$

Provided a backward stable algorithm is used to compute the spectral decompositions in Step B in Algorithm 2.1, the errors made when computing the eigenvalues $d_i$ of the subproblems can be bounded as

$$|d_i - \hat{d}_i| \le O(\varepsilon) \max_{j=1,2,\ldots,p} \|\tilde{B}_j\|_2, \quad i = 1, 2, \ldots, n$$

according to Weyl's theorem (see, for example, [8]).

The method used for computing the spectral decomposition of the rank-one modification problems arising in Step C.ii of Algorithm 2.1 is the one developed by Gu and Eisenstat [14]. Therefore, for the synthesis part of the algorithm described in Section 2.3 their analysis can be directly applied, showing that the eigenvalues are computed to high absolute accuracy. They also showed that the vector $\bar{z}$ is close to $z$ in high absolute accuracy and that these properties suffice to compute the spectral decomposition of a rank-one modification of a diagonal matrix numerically stably. Therefore each numerically computed eigendecomposition of a rank-one modification problem satisfies

$$D_{(i)} + \sigma_{(i)} \tilde{z}_{(i)} \tilde{z}_{(i)}^\top = \hat{Q} \hat{\Lambda}_{(i)} \hat{Q}^\top + O\left(\varepsilon \left(\|D_{(i)}\|_2 + \sigma_{(i)} \|\tilde{z}_{(i)}\|_2^2\right)\right)$$

where $\hat{Q}$ is numerically orthogonal.

**5. Experiments.** The block-tridiagonal divide-and-conquer method has been implemented in Fortran 77 (`dsbtdc`) and evaluated experimentally. In Section 5.1 it is compared to corresponding LAPACK routines, and in Section 5.2 the effects of the merging order for unequal block sizes are illustrated.

Experiments were performed on a number of test matrices. Symmetric blocks $B_i$ as well as vectors $u_i$ and $v_i$, which determine the rank-one off-diagonal blocks $E_i$, were created randomly using Matlab. $\sigma_i = 1$ ($i = 1, 2, \ldots, p-1$) was chosen for all 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 $\varepsilon = 1.1 \cdot 10^{-16}$.

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

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

**5.1. Equally Sized Blocks.** `dsbtdc` is compared with the routines

- `LAPACK/dsbevd` for a banded symmetric matrix, which performs tridiagonalization, the tridiagonal divide-and-conquer method, and finally the backtransformation of the eigenvectors, as well as
- `LAPACK/dsyev` for a general symmetric matrix, which performs tridiagonalization, the QR-algorithm on the tridiagonal matrix, and finally the backtransformation of the eigenvectors.

Results are shown for the following three matrices:

- $M^e_{124}$: $n = 620$; $p = 124$; block sizes $k_i = 5$, $i = 1, 2, \ldots, p$
- $M^e_{62}$: $n = 620$; $p = 62$; block sizes $k_i = 10$, $i = 1, 2, \ldots, p$
- $M^e_{31}$: $n = 620$; $p = 31$; block sizes $k_i = 20$, $i = 1, 2, \ldots, p$

The narrowest band matrix which fully contains the corresponding block-tridiagonal matrix was used as input for `LAPACK/dsbevd`. This matrix contains $2(p-2)$ zero $n/p \times n/p$ triangles in addition to the block-tridiagonal matrix. These triangles fill up during the tridiagonalization performed by `LAPACK/dsbevd`. However, a direct comparison for block-tridiagonal matrices is not possible, and especially for large values of $p$ the difference is not very big. For `LAPACK/dsyev`, the block-tridiagonal matrix was completed to a full matrix by zero entries.

The experiments are summarized in Table 5.1. $T_{BT}, T_{LB}$ and $T_{LF}$ denote the runtimes of `dsbtdc`, `LAPACK/dsbevd` and `LAPACK/dsyev`, respectively. The results show that due to being able to take advantage of the special structure of the off-diagonal blocks as well as due to improved data-locality which is important for the memory hierarchies of modern computer systems the block-tridiagonal divide-and-conquer algorithm is more efficient than the standard method for banded eigenvalue problems while achieving the same level of accuracy.

TABLE 5.1
*Comparison for block-tridiagonal matrices with equally sized blocks.*

| Routine | | $M^e_{124}$ | $M^e_{62}$ | $M^e_{31}$ |
|---|---|---|---|---|
| `dsbtdc` | | | | |
| | $T_{BT}$ [s] | 1.6 | 2.5 | 3.1 |
| | $\mathcal{R}$ | $1.4 \cdot 10^{-15}$ | $1.6 \cdot 10^{-15}$ | $1.2 \cdot 10^{-15}$ |
| | $\frac{\mathcal{R}\ \|M_p\|_2}{\max \|B_i\|_2 + \max \sigma_i}$ | $1.4 \cdot 10^{-15}$ | $1.6 \cdot 10^{-15}$ | $1.2 \cdot 10^{-15}$ |
| | $\mathcal{O}$ | $3.9 \cdot 10^{-15}$ | $4.9 \cdot 10^{-15}$ | $6.5 \cdot 10^{-15}$ |
| `LAPACK/dsbevd` | | | | |
| | $T_{LB}/T_{BT}$ | 5.0 | 3.5 | 3.4 |
| | $\mathcal{R}$ | $2.3 \cdot 10^{-14}$ | $6.8 \cdot 10^{-15}$ | $6.3 \cdot 10^{-15}$ |
| | $\mathcal{O}$ | $2.6 \cdot 10^{-14}$ | $8.4 \cdot 10^{-15}$ | $8.0 \cdot 10^{-15}$ |
| `LAPACK/dsyev` | | | | |
| | $T_{LF}/T_{BT}$ | 12.1 | 7.1 | 6.5 |
| | $\mathcal{R}$ | $4.9 \cdot 10^{-15}$ | $5.6 \cdot 10^{-15}$ | $3.7 \cdot 10^{-15}$ |
| | $\mathcal{O}$ | $1.4 \cdot 10^{-14}$ | $1.3 \cdot 10^{-14}$ | $1.1 \cdot 10^{-14}$ |

**5.2. General Blocks.** The two testmatrices

- $M^b_8$: $n = 1500$; $p = 8$; block sizes $\{k_i\}_{i=1}^8 = \{5, 180, 190, 375, 5, 180, 190, 375\}$
- $M^u_8$: $n = 1500$; $p = 8$; block sizes $\{k_i\}_{i=1}^8 = \{375, 190, 375, 190, 180, 180, 5, 5\}$

differ only in the order of their diagonal blocks. However, the merging operations for $M^b_8$ are quite nicely balanced (except for two initial merges), whereas most of the merging operations for $M_8^u$ (in particular the ones towards the end of the synthesis phase) are unbalanced.

Table 5.2 shows the corresponding experimental data.

TABLE 5.2
*Influence of unequal block sizes on execution time.*

| Routine | | $M_8^b$ | $M_8^u$ |
|---|---|---|---|
| `dsbtdc` | | | |
| | $T_{BT}$ $[s]$ | 39.2 | 46.2 |
| | $\mathcal{R}$ | $2.5 \cdot 10^{-15}$ | $3.6 \cdot 10^{-15}$ |
| | $\frac{\mathcal{R}\ \Vert M_p\Vert_2}{\max \Vert B_i\Vert_2 + \max \sigma_i}$ | $2.4 \cdot 10^{-15}$ | $3.3 \cdot 10^{-15}$ |
| | $\mathcal{O}$ | $1.8 \cdot 10^{-14}$ | $1.7 \cdot 10^{-14}$ |
| `LAPACK/dsbevd` | | | |
| | $T_{LB}/T_{BT}$ | 5.3 | 4.7 |
| | $\mathcal{R}$ | $7.7 \cdot 10^{-15}$ | $7.6 \cdot 10^{-15}$ |
| | $\mathcal{O}$ | $1.5 \cdot 10^{-14}$ | $1.7 \cdot 10^{-14}$ |
| `LAPACK/dsyevd` | | | |
| | $T_{LF}T_{BT}$ | 6.9 | 7.6 |
| | $\mathcal{R}$ | $9.3 \cdot 10^{-15}$ | $5.6 \cdot 10^{-15}$ |
| | $\mathcal{O}$ | $1.9 \cdot 10^{-14}$ | $1.4 \cdot 10^{-14}$ |

**6. Conclusion.** It has been shown that for a special class of block-tridiagonal matrices the divide-and-conquer approach for computing the spectral decomposition can be directly extended and yields very attractive results in terms of efficiency and accuracy.

The implications of unequally sized blocks have been studied. A reliable method for determining a merging order whose associated arithmetic complexity is at least close to optimal for any given sequence of block sizes has been proposed.

The algorithm presented in this paper provides *approximate* eigenpairs if the off-diagonal blocks of a general block-tridiagonal matrix, which in general have a rank higher than one, are approximated by rank-one matrices. In some instances this approximation may not be accurate enough. However, the algorithm will be extended to allow for higher rank approximations of the off-diagonal blocks $E_i$. If they are not of rank one, then the singular value decompositions (see [13])

$$E_i = \sum_{i=1}^{k_i} \sigma_i u_i v_i^\top, \quad i = 1, 2, \ldots, p-1,$$

can be used for constructing approximations of arbitrary rank corresponding to the largest singular values $\sigma_i$. The authors are currently developing these ideas in order to utilize higher rank approximations of the off-diagonal blocks if required. This approximative approach will be illustrated in more detail in a separate forthcoming paper.

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

[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, 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.

[11] 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.

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

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

[14] 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.

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

[16] R.-C. Li, *Solving the secular equations stably and efficiently*, Lapack Working Note 89, University of Tennessee, Knoxville, TN, Nov. 1994.

[17] K. Löwner, *Über monotone Matrixfunktionen*, Math. Z., 38 (1934), pp. 177–216.

[18] 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.

[19] 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.

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

[21] 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.