SIAM J. SCI. STAT. COMPUT.
Vol. 11, No. 1, pp. 161-173, January 1990

AN $O(N^2)$ METHOD FOR COMPUTING THE EIGENSYSTEM OF $N \times N$ SYMMETRIC TRIDIAGONAL MATRICES BY THE DIVIDE AND CONQUER APPROACH*

DORON GILL† AND EITAN TADMOR‡

To Eugene Isaacson on his 70th birthday

Abstract. An efficient method to solve the eigenproblem of $N \times N$ symmetric tridiagonal matrices is proposed. Unlike the standard eigensolvers that necessitate $O(N^3)$ operations to compute the eigenvectors of such matrices, the proposed method computes both the eigenvalues and eigenvectors with only $O(N^2)$ operations. The method is based on *serial* implementation of the recently introduced Divide and Conquer algorithm [3], [1], [4]. It exploits the fact that by $O(N^2)$ Divide and Conquer operations one can compute the eigenvalues of an $N \times N$ symmetric tridiagonal matrix *and* a small number of pairs of successive rows of its eigenvector matrix. The rest of the eigenvectors (either all together or one at a time) are computed by linear three-term recurrence relations. The paper is concluded with numerical examples that demonstrate the superiority of the proposed method for a special class of symmetric tridiagonal matrices, by saving an *order* of magnitude in execution time at the expense of sacrificing a *few* orders of accuracy, although for symmetric tridiagonal matrices in general, the method appears to be unstable.

Key words. symmetric eigenvalue problem, divide and conquer, updating problem

AMS(MOS) subject classification. 65F15

1. Introduction. The QR algorithm computes the eigenvalues of an $N \times N$ Symmetric Tridiagonal (ST) matrix with $O(N^2)$ operations, while the corresponding eigenvector matrix is accumulated during the algorithm at the expense of $O(N^3)$ operations. The additional order of magnitude required to compute the eigenvectors is typical of serial algorithms. A complete $O(N^2)$ eigensolver can be obtained by appending such serial algorithms with the Inverse Iteration (INVIT) method. Indeed, $O(N)$ operations of only *one* INVIT will suffice to accurately compute each eigenvector corresponding to an *isolated* eigenvalue [8, Chap. 4]. In case of clustered eigenvalues, however, the INVIT requires a more carefully chosen initialization, to avoid the loss of mutual orthogonality between the corresponding, closely "related" eigenvectors.

Recently, a parallel Divide and Conquer (DC) algorithm was introduced for computing the spectral decomposition of ST matrices [3], [1], [4]. A *serial* implementation of this algorithm, described in § 2, requires the same number of operations. Namely, the eigenvalues, which coincide with the roots of the so-called secular equation [6], are computed at the expense of no more than $O(N^2)$ sequential operations, while the associated eigenvectors necessitate $O(N^3)$ sequential operations. As before, the INVIT, taken with the necessary precautions, is available here as an $O(N^2)$ method to compute these eigenvectors. In §§ 3-4, we propose an alternative efficient method, derived from

* Received by the editors March 2, 1988; accepted for publication (in revised form) January 31, 1989.

† School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Ramat Aviv, Tel Aviv 69 978, Israel.

‡ School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Ramat Aviv, Tel Aviv 69 978, Israel. Present address, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, Virginia 23665. The research of this author was partially supported by the National Aeronautics and Space Administration under NASA contract NAS1-18107. Additional support was provided by the U.S.-Israel Binational Science Foundation grant 85-00346, and by National Science Foundation grant DMS85-03294, while this author was in residence at the University of California at Los Angeles, Los Angeles, California 90024.

(and therefore better suited to) the DC algorithm, which computes the eigensystem of $N \times N$ ST matrices with only $O(N^2)$ sequential operations. The method employs linear three-term recurrence relations that successively compute the *rows* of the eigenvector matrix (or the *components* of each of the desired eigenvectors). The coefficients of these relations depend on the already computed eigenvalues, and the method hinges on the fact that the initial first two rows (or components) for the recurrence relations *emerge naturally* from the DC computation of these eigenvalues. Thus, the input data for the recurrence relations depends solely on the $O(N^2)$ operations for the DC calculation of the eigenvalues. Together with the additional $O(N^2)$ operations required to carry out these relations, we end up with an efficient $O(N^2)$ method to compute the whole eigensystem of *ST* matrices. It should be emphasized that the advantages of the DC algorithm are retained in our case. That is, we have a method which on the one hand is well suited to exploit parallelism; on the other hand, even when run in serial mode on large problems, the method is faster than the previously best sequential algorithms, e.g., [3], [4].

The main limitation of the proposed method lies in the possible instability of the three term recurrence relations mentioned above. In § 4, we identify a useful class of ST matrices for which the corresponding recurrence relations are stable. In such stable cases, the numerical results of our method are almost as accurate as the standard DC algorithm. In the general case, however, the accuracy of our method may deteriorate for large N, $N \gtrsim 100$, due to the instability of the corresponding recurrence relations. To overcome the unstable error accumulation in such cases, one may *restart* the recurrence relations at any stage of the recursive iterations with two new successive rows of the eigenvector matrix. In § 4, we show how to obtain two such successive rows for restarting, at the expense of $O(N^2)$ DC operations.

Due to the sensitivity of the three-term recurrence relations, their input data should be provided with high accuracy. To achieve this, we employ in § 5 an improved root finder—interesting for its own sake—in order to solve the secular equation mentioned above. Numerical examples that demonstrate the efficiency as well as the limitations of the proposed method are presented in § 6.

2. The Divide and Conquer algorithm—An overview. Let D_N be an $N \times N$ diagonal matrix and let $D_N + \sigma z_N z_N'$ be a Rank One Modification (ROM) of this matrix by a unit N-vector z_N.[1] The spectral decomposition of such ROM matrices is the heart of the Divide and Conquer (DC) algorithm. Here we note that the problem of finding the spectral decomposition of an N-dimensional ROM matrix, the so-called *updating problem*, can be solved at the expense of no more than Const. N^2 operations [1], [3], [4]. Details of this solution are discussed in § 4.

With this in mind we now turn to consider the eigenproblem of general $N \times N$ Symmetric Tridiagonal (ST) matrices

$$T_N = \left[\begin{array}{cccc:cccc}
t_{11} & t_{12} & & & & & & \\
t_{21} & t_{22} & & & & & & \\
 & & \ddots & t_{mm} & t_{mm+1} & & & \\
\hdashline
 & & & t_{m+1,m} & t_{m+1,m+1} & \ddots & & \\
 & & & & & & & t_{N-1,N} \\
 & & & & & & t_{N,N-1} & t_{NN}
\end{array}\right], \quad t_{ij} = t_{ji}.$$

[1] Throughout the paper, vectors and matrices will be used with a subscript index denoting their dimension.

We can assume without restriction that N is even, $N=2m$, and that T_N is already given in its unreduced form, i.e., $t_{i,i+1}\neq 0$, $1\leqq i\leqq N-1$; otherwise, T_N is decoupled into smaller unreduced ST matrices. Then, we can *split* T_N into the sum of

$$T_N=\left[\begin{array}{cccc:cccc} t_{11} & t_{12} & & & & & & \\ & t_{22} & & & & & & \\ & & t_{mm}-\beta & & & 0 & & \\ \hdashline & & 0 & & t_{m+1,m+1}-\beta & & & \\ & & & & & & & t_{N,N} \end{array}\right]+\beta\left[\begin{array}{ccc:ccc} 0 & & & & & 0 \\ & & 1 & 1 & & \\ \hdashline & & 1 & 1 & & \\ 0 & & & & & 0 \end{array}\right],\qquad \beta=t_{m,m+1},$$

i.e.,

$$T_N=\begin{bmatrix} T^{(1)}_{N/2} & \\ & T^{(2)}_{N/2}\end{bmatrix}+\beta b_N b'_N,\qquad b_N=e^{(m)}_N+e^{(m+1)}_N, \tag{2.1}$$

where the blocks $T^{(1)}_{N/2}$ and $T^{(2)}_{N/2}$ are $N/2\times N/2$ ST matrices and $\beta\equiv t_{m,m+1}\neq 0$ is the coupling term of these two blocks.

The DC algorithm [3], [1], [4] is based on the fact that in order to solve the eigenproblem of N-dimensional ST matrices, it is sufficient to solve this problem for $(N/2)$-dimensional ST matrices. Specifically, if

$$\begin{aligned} T^{(1)}_{N/2}&=P^{(1)}_{N/2}\Lambda^{(1)}_{N/2}P^{(1)'}_{N/2}, \qquad P^{(1)}_{N/2}P^{(1)'}_{N/2}=I_{N/2}\\ T^{(2)}_{N/2}&=P^{(2)}_{N/2}\Lambda^{(2)}_{N/2}P^{(2)'}_{N/2}, \qquad P^{(2)}_{N/2}P^{(2)'}_{N/2}=I_{N/2}, \end{aligned}\tag{2.2}$$

are the spectral decompositions of the $N/2\times N/2$ ST matrices $T^{(1)}_{N/2}$ and $T^{(2)}_{N/2}$, respectively, then we can compute the spectral decomposition of the $N\times N$ ST matrix T_N by the following procedure:

I. First, we evaluate the unit N-vector z_N,

$$z_N=\frac{1}{\sqrt{2}}\begin{bmatrix} P^{(1)}_{N/2} & \\ & P^{(2)}_{N/2}\end{bmatrix}^t b_N,\qquad b_N=\begin{bmatrix}0\\ \cdot\\ 1\\ \cdots\\ 1\\ \cdot\\ 0\end{bmatrix} \tag{2.3a}$$

so that by (2.1), (2.2), and (2.3a), T_N is unitarily similar to the ROM matrix

$$\begin{aligned} T_N&=\begin{bmatrix} P^{(1)}_{N/2}\Lambda^{(1)}_{N/2}P^{(1)t}_{N/2} & \\ & P^{(2)}_{N/2}\Lambda^{(2)}_{N/2}P^{(2)t}_{N/2}\end{bmatrix}+\beta b_N b'_N\\ &=\begin{bmatrix} P^{(1)}_{N/2} & \\ & P^{(2)}_{N/2}\end{bmatrix}\left(\begin{bmatrix} \Lambda^{(1)}_{N/2} & \\ & \Lambda^{(2)}_{N/2}\end{bmatrix}+2\beta z_N z'_N\right)\begin{bmatrix} P^{(1)t}_{N/2} & \\ & P^{(2)t}_{N/2}\end{bmatrix}\\ &=\begin{bmatrix} P^{(1)}_{N/2} & \\ & P^{(2)}_{N/2}\end{bmatrix}(D_N+2\beta z_N z'_N)\begin{bmatrix} P^{(1)}_{N/2} & \\ & P^{(2)}_{N/2}\end{bmatrix}^t,\\ D_N&=\begin{bmatrix} \Lambda^{(1)}_{N/2} & \\ & \Lambda^{(2)}_{N/2}\end{bmatrix}. \end{aligned}$$

II. Second, we solve the updating problem by finding the spectral decomposition of the ROM matrix

$$D_N+\sigma z_N z'_N=Q_N\Lambda_N Q'_N,\qquad Q_N Q'_N=I_N,\quad \sigma=2\beta. \tag{2.3b}$$

III. Finally, we compute the unitary matrix

$$P_N = \begin{bmatrix} P^{(1)}_{N/2} & \\ & P^{(2)}_{N/2} \end{bmatrix} Q_N \tag{2.3c}$$

and obtain, by (2.3b) and (2.3c), the spectral decomposition of T_N as

$$T_N = \begin{bmatrix} P^{(1)}_{N/2} & \\ & P^{(2)}_{N/2} \end{bmatrix} Q_N \Lambda_N Q_N^t \begin{bmatrix} P^{(1)}_{N/2} & \\ & P^{(2)}_{N/2} \end{bmatrix}^t = P_N \Lambda_N P_N^t, \qquad P_N P_N^t = I_N.$$

This process can be applied recursively: the N-dimensional eigenproblem of T_N is solved in terms of two independent $(N/2)$-dimensional eigenproblems of $T^{(1)}_{N/2}$ and $T^{(2)}_{N/2}$, which in turn are solved in terms of four independent $(N/4)$-dimensional eigenproblems of $T^{(1)}_{N/4}$, $T^{(2)}_{N/4}$, $T^{(3)}_{N/4}$, $T^{(4)}_{N/4}$, etc. Thus, the DC algorithm for an $N=2^n$-dimensional ST matrix T_N is organized as follows. After $n-1$ splitting steps we are left with 2^{n-2} pairs of 2×2 ST matrices. In the first iteration they are used to construct, with the help of (2.3a)–(2.3c), the eigensystem of 2^{n-3} pairs of 4×4 ST matrices; in the second iteration, one constructs the eigensystem of 2^{n-4} pairs of 8×8 ST matrices, etc.; after $n-2$ such iterations we end up with the eigensystems of the pair $T^{(1)}_{N/2}$, $T^{(2)}_{N/2}$, and the last $n-1$ iteration solves the eigenproblem of T_N. A sequential implementation of a typical kth iteration consists of 2^{n-k-1} times, evaluating the 2^{k+1}-dimensional unit vectors z in the first stage (2.3a), solving 2^{k+1}-dimensional ROM eigenproblems in the second stage (2.3b), and computing 2^{k+1}-dimensional products of unitary matrices in the third stage (2.3c).

The total amount of work spent on the first two stages, (2.3a) and (2.3b), of all iterations, does not exceed 2 Const. N^2; the total work required for computing the eigenvectors in (2.3c) is $\sum_{k=1}^{n-1} 2^{n-k-1}\cdot 4(2^k)^3 \lesssim \frac{2}{3}N^3$. Thus, the total operations cost of the DC algorithm for finding the eigensystem (both the eigenvalues and eigenvectors) of an $N\times N$ ST matrix is $\frac{2}{3}N^3 + 2$ Const. N^2.

If only the eigenvalues are required, then we can do better by saving the $O(N^3)$ operations required to compute the eigenvectors in the third stage (2.3c). Instead, the first stage of a typical kth iteration, which requires 2^{n-k-1} different evaluations of 2^{k+1}-dimensional unit vectors of the form

$$z_{2^{k+1}} = \frac{1}{\sqrt{2}} \begin{bmatrix} P^{(1)}_{2^k} & \\ & P^{(2)}_{2^k} \end{bmatrix}^t b_{2^{k+1}},$$

can be efficiently implemented as follows: According to (2.3c), $P^{(1)}_{2^k}$ is represented by a successive product of

$$\begin{bmatrix} Q^{(1)}_{2^j} & & \\ & \ddots & \\ & & Q^{(2^{k-j})}_{2^j} \end{bmatrix} \qquad j=k, k-1, \cdots, 1,$$

where $Q^{(\cdot)}_{2^j}$ were found by spectral decompositions of ROM matrices in previous iterations; similarly, $P^{(2)}_{2^k}$ is represented by a successive product of

$$\begin{bmatrix} Q^{(2^{k-j}+1)}_{2^j} & & \\ & \ddots & \\ & & Q^{(2^{k-j+1})}_{2^j} \end{bmatrix}, \qquad j=k, k-1, \cdots, 1.$$

Hence, we can evaluate each of the 2^{n-k-1} different vectors, $z_{2^{k+1}}$, as $z_{2^{k+1}} = z^{(k)}_{2^{k+1}}$, where

$$z^{(j)}_{2^{k+1}} = \begin{bmatrix} Q^{(1)}_{2^j} & & \\ & \ddots & \\ & & Q^{(2^{k-j+1})}_{2^j} \end{bmatrix}^t z^{(j-1)}_{2^{k+1}}, \qquad j=1,2,\cdots,k, \qquad z^{(0)}_{2^{k+1}} \equiv \frac{1}{\sqrt{2}} b_{2^{k+1}}, \tag{2.4a}$$

at the expense of $\sum_{j=1}^{k} 2^{k-j+1} \cdot 2^{2j} \lesssim 4^{k+1}$ operations. The total work spent on the first stages in all iterations is therefore $\sum_{k=1}^{n-1} 2^{n-k-1} \cdot 4^{k+1} \lesssim 2N^2$. This is complemented with the solution of 2^{n-k-1} different updating problems (see (2.3b))

$$D_{2^{k+1}} + \sigma z_{2^{k+1}} z'_{2^{k+1}} = Q_{2^{k+1}} \Lambda_{2^{k+1}} Q_{2^{k+1}}. \tag{2.4b}$$

The total work spent on the second stage in all iterations amounts to 2 Const. N^2. Consequently, the total operations cost of the DC algorithm, (2.4a), (2.4b), for finding the eigenvalues of an $N \times N$ ST matrix is (2 Const. + 2)N^2.

3. An $O(N^2)$ method for the eigensystem of $N \times N$ ST matrix. Given an $N \times N$ ST matrix T_N, we can compute its eigenvalues by the DC algorithm (2.4a), (2.4b) at the expense of no more than $O(N^2)$ operations.[2] Thus, it remains to compute efficiently, i.e., with $O(N^2)$ operations, the eigenvectors of this matrix. To this end we may proceed as follows.

We seek the unitary matrix P_N, $P_N P'_N = I_N$, which diagonalizes T_N,

$$T_N = P_N \Lambda_N P'_N. \tag{3.1}$$

Let $p^{(i)} \equiv p_N^{(i)}$ denote the ith *row* vector of P_N. Equating the ith rows of

$$T_N P_N = P_N \Lambda_N$$

we obtain, in view of the reduced tridiagonal structure of T_N,

$$t_{i,i-1} p_N^{(i-1)} + t_{i,i} p_N^{(i)} + t_{i,i+1} p_N^{(i+1)} = p_N^{(i)} \Lambda_N, \qquad t_{i,i\pm 1} \neq 0. \tag{3.2}$$

Equation (3.2) is a linear three-term recurrence relation between the rows, $p_N^{(i)}$, of P_N, whose coefficients are determined by the entries of T_N. The input data required to solve these relations uniquely consists of

(1) The eigenvalues $\Lambda_N = \operatorname{diag}(\lambda^{(1)}, \lambda^{(2)}, \cdots, \lambda^{(N)})$ of T_N, which determine the terms $p_N^{(i)} \Lambda_N \equiv (\lambda^{(1)} p_{i1}, \lambda^{(2)} p_{i2}, \cdots, \lambda^{(N)} p_{iN})$ on the right of (3.2). The eigenvalues are computed by the DC algorithm (2.4a), (2.4b) with (2 Const. + 2)N^2 operations.

(2) Two successive *rows* of P_N that will serve as initial data for the recursive three-term relations (3.2). The proposed method hinges on the observation that two such rows *emerge naturally* from that part of the DC algorithm (2.4a), (2.4b) which computes the eigenvalues of T_N. Indeed, from the last $n-1$ iteration of (2.4a) we have at our disposal the unit N-vector z_N, which according to (2.3a) satisfies

$$\begin{bmatrix} z_{N/2}^{(1)} \\ z_{N/2}^{(2)} \end{bmatrix} = \begin{bmatrix} P_{N/2}^{(1)} & \vdots \\ \cdots & \cdots \\ & \vdots\ P_{N/2}^{(2)} \end{bmatrix}' \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \begin{bmatrix} z_{N/2}^{(1)} \\ z_{N/2}^{(2)} \end{bmatrix} \equiv \sqrt{2} \cdot z_N. \tag{3.3}$$

Hence $z_{N/2}^{(1)}$ and $z_{N/2}^{(2)}$ are in fact the last and first column vectors of $P_{N/2}^{(1)\prime}$ and $P_{N/2}^{(2)\prime}$, respectively. Put differently, $(z_{N/2}^{(1)}, O_{N/2})'$ and $(O_{N/2}, z_{N/2}^{(2)})'$ are row numbers $m = N/2$ and $m+1$ of

$$\begin{bmatrix} P_{N/2}^{(1)} & \\ & P_{N/2}^{(2)} \end{bmatrix}.$$

[2] In fact, as observed by Cuppen [3], this number of operations can be substantially reduced by up to $O(N \log N)$ operations, in practical cases which employ sufficiently many deflations.

Consequently, equating the m and $m+1$ rows of (2.3c), we obtain the two initial successive rows as

$$p_N^{(m)}=(z_{N/2}^{(1)}, O_{N/2})'Q_N, \qquad (3.4)$$
$$p_N^{(m+1)}=(O_{N/2}, z_{N/2}^{(2)})'Q_N;$$

the remaining rows of P_N are computed recursively by (3.2)

$$p_N^{(i+1)}=\frac{1}{t_{i,i+1}}[p_N^{(i)}(\Lambda_N-t_{i,i}I_N)-t_{i,i-1}p_N^{(i-1)}], \qquad i=m+1,\cdots,N-1, \qquad (3.5a)$$

$$p_N^{(i-1)}=\frac{1}{t_{i,i-1}}[p_N^{(i)}(\Lambda_N-t_iI_N)-t_{i,i+1}p_N^{(i+1)}], \qquad i=m, m-1,\cdots,2. \qquad (3.5b)$$

The operation cost of (3.4)–(3.5) does not exceed $3N^2$. Thus (2.4a), (2.4b) together with (3.4), (3.5) provide us with an $O(N^2)$ method for computing the whole eigensystem of $N\times N$ ST matrices.

The error analysis of the proposed method depends on two ingredients:

(1) The accuracy of the input data for (3.5a), (3.5b), namely, the errors accumulated in computing the eigenvalues $\Lambda_N=\operatorname{diag}(\lambda^{(1)}, \lambda^{(2)},\cdots,\lambda^{(N)})$ and the two successive rows $p_N^{(m)}, p_N^{(m+1)}$ of P_N. The size of these errors is determined by the stability properties of the DC algorithm (2.4a), (2.4b). In this context, we recall that stable behavior of the DC algorithm hinges on an accurate solution of the ROM problem (2.4b) (see [1], [3], [4]). In § 5, we borrow from [1], [3], and [4], discussing a root finder for an accurate computation of the eigenvalues $\Lambda_{2^{k+1}}$, which are obtained as the roots of the characteristic equation associated with the ROM matrix in (2.4b).

(2) The second source of error is due to accumulation of rounding errors in the recurrence relations (3.5a), (3.5b). In order to examine this error accumulation, we rewrite (3.5) as a one-step iteration

$$[p_N^{(i+1)}, p_N^{(i)}]=[p_N^{(i)}, p_N^{(i-1)}]\begin{bmatrix}1/t_{i,i+1}[\Lambda_N-t_{i,i}I_N] & I_N\\ -(t_{i,i-1}/t_{i,i+1})I_N & O_N\end{bmatrix}, \quad i=m+1,\cdots,N-1, \qquad (3.6a)$$

$$[p_N^{(i-1)}, p_N^{(i)}]=[p_N^{(i)}, p_N^{(i+1)}]\begin{bmatrix}1/t_{i,i-1}[\Lambda_N-t_{i,i}I_N] & I_N\\ -(t_{i,i+1}/t_{i,i-1})I_N & O_N\end{bmatrix}, \quad i=m, m-1,\cdots,2. \qquad (3.6b)$$

An indication of the stability properties of (3.6a), (3.6b) is provided by the eigenvalues $\kappa=\kappa_{ij}^{\pm}$ of the two $2N\times 2N$ matrices on the right-hand sides, i.e., for $i=m+1, m+2,\cdots,N-1$ we have

$$t_{i,i+1}(\kappa_{ij}^{\pm})^2-(\lambda_j-t_{i,i})\kappa_{ij}^{\pm}+t_{i,i-1}=0, \qquad j=1,2,\cdots,N \qquad (3.7a)$$

and for $i=m, m-1,\cdots,2$ we have

$$t_{i,i-1}(\kappa_{ij}^{\pm})^2-(\lambda_j-t_{i,i})\kappa_{ij}^{\pm}+t_{i,i+1}=0, \qquad j=1,2,\cdots,N. \qquad (3.7b)$$

Hence the error in the ith iteration of (3.5) is amplified by a factor of at least

$$g^{(i)}\equiv\max_{1\leqq j\leqq N}(|\kappa_{ij}^{+}|, |\kappa_{ij}^{-}|).$$

Thus, the method is expected to be stable if

$$\prod_{i=2}^{N-1} g^{(i)}=\prod_{i=2}^{N-1}\max_{1\leqq j\leqq N}(|\kappa_{ij}^{+}|, |\kappa_{ij}^{-}|)\leqq \text{Const.} \qquad (3.8)$$

As a canonical example for such stable behavior, let us consider ST matrices whose entries are "slowly varying" along their diagonals, i.e., $t_{ij} \sim t_{i+1,j+1}$. Now, *if* the superdiagonal entries are properly scaled so that also $t_{i,i+1} \sim t_{i,i-1}$, then by Gershgorin estimate we have for *any* $1 \leqq j \leqq N$,

$$|\lambda_j - t_{i,i}|^2 \lesssim (|t_{i,i-1}| + |t_{i,i+1}|)^2 \lesssim 4|t_{i,i-1}| \cdot |t_{i,i+1}|,$$

and hence the product of the characteristic roots $\kappa_{ij}^{\pm}$ is of order unity, for

$$|\kappa_{ij}^{\pm}|^2 = \left|\frac{(\lambda_j - t_{i,i}) \pm \sqrt{(\lambda_j - t_{i,i})^2 - 4t_{i,i+1}t_{i,i-1}}}{2t_{i,i\pm1}}\right|^2 \lesssim \left|\frac{t_{i,i\mp1}}{t_{i,i\pm1}}\right| \sim 1.$$

If, on the other hand, (3.8) fails, we have an unstable error growth at the amount $\prod_{i=2}^{N-1} g^{(i)} \gg 1$, as confirmed by the numerical examples demonstrated in § 6. Typically, such an instability shows up by the loss of orthogonality between the computed rows $p_N^{(i)}$ of P_N. Hence, one approach to solve the stability problem would be to use reorthogonalization, once the instability was detected by the loss of orthogonality; consult [3, § 3]. An alternative approach to overcome the instability problem, which better suits the proposed method, is to *restart* the recurrence relations (3.5) at the current iteration with two new successive rows of P_N. How should we obtain two such successive rows for restarting? Consider, for example, the $N = 4m$-dimensional problem. The iteration before the last of (2.4a) provides us with two $(N/2)$-dimensional unit vectors, say $z_{N/2}$ and $w_{N/2}$, where

$$\text{(3.9a)} \qquad \begin{bmatrix} z_{N/4}^{(1)} \\ z_{N/4}^{(2)} \end{bmatrix} = \begin{bmatrix} P_{N/4}^{(1)} & \vdots & \\ \cdots & \cdots & \cdots \\ & \vdots & P_{N/4}^{(2)} \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} z_{N/4}^{(1)} \\ z_{N/4}^{(2)} \end{bmatrix} \equiv \sqrt{2}\, z_{N/2},$$

$$\text{(3.9b)} \qquad \begin{bmatrix} w_{N/4}^{(1)} \\ w_{N/4}^{(2)} \end{bmatrix} = \begin{bmatrix} P_{N/4}^{(3)} & \vdots & \\ \cdots & \cdots & \cdots \\ & \vdots & P_{N/4}^{(4)} \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} w_{N/4}^{(1)} \\ w_{N/4}^{(2)} \end{bmatrix} \equiv \sqrt{2}\, w_{N/2}.$$

As before, we obtain the m and $m+1$ rows of $P_{N/2}^{(1)}$ as

$$\text{(3.10a)} \qquad \begin{aligned} p_{N/2}^{(1,m)} &= (z_{N/4}^{(1)}, O_{N/4})^t Q_{N/2}^{(1)} \\ p_{N/2}^{(1,m+1)} &= (O_{N/4}, z_{N/4}^{(2)})^t Q_{N/2}^{(1)} \end{aligned}$$

and the m and $m+1$ rows of $P_{N/2}^{(2)}$ as

$$\text{(3.10b)} \qquad \begin{aligned} p_{N/2}^{(2,m)} &= (w_{N/4}^{(1)}, O_{N/4})^t Q_{N/2}^{(2)} \\ p_{N/2}^{(2,m+1)} &= (O_{N/4}, w_{N/4}^{(2)})^t Q_{N/2}^{(2)}. \end{aligned}$$

Consequently, we can compute with $O(N^2)$ operations row numbers m, $m+1$, $3m$, and $3m+1$ of P_N, for by (2.3c) we have

$$\text{(3.11)} \qquad \begin{aligned} p_N^{(m)} &= (p_{N/2}^{(1,m)}, O_{N/2})^t Q_N \\ p_N^{(m+1)} &= (p_{N/2}^{(1,m+1)}, O_{N/2})^t Q_N \\ p_N^{(3m)} &= (O_{N/2}, p_{N/2}^{(2,m)})^t Q_N \\ p_N^{(3m+1)} &= (O_{N/2}, p_{N/2}^{(2,m+1)})^t Q_N. \end{aligned}$$

In a similar manner, one can restart the recurrence relations (3.5) at any desired iteration.

4. The eigenvectors of T_N—One at a time. In the previous section we discussed an $O(N^2)$ method for computing the whole eigensystem—eigenvalues and eigenvectors of an $N\times N$ ST matrix. In several applications one is interested in only a few of the eigenvectors of T_N. We now present a variant of this method that enables us to compute each one of the desired eigenvectors with $O(N)$ operations.

As before, we first prepare, with the help of the DC algorithm (2.4a), (2.4b), (3.4), the eigenvalues $\{\lambda^{(j)}\}_{j=1}^N$ and the two middle successive rows, $p_N^{(m)}$ and $p_N^{(m+1)}$ of P_N. This can be done at the expense of $O(N^2)$ operations, and in many practical cases with even less. Equipped with this we can compute the eigenvector $x_N = (x^{(1)}, x^{(2)}, \cdots, x^{(N)})$ corresponding to the eigenvalue, say, $\lambda^{(j)}$

$$T_N x_N = \lambda^{(j)} x_N. \tag{4.1}$$

Equation (4.1) gives us the linear three-term recurrence relations between the components of x

$$t_{i,i-1}x^{(i-1)} + t_{i,i}x^{(i)} + t_{i,i+1}x^{(i+1)} = \lambda^{(j)}x^{(i)}. \tag{4.2}$$

Since x_N coincides with the jth *column* of P_N, we have its two middle entries $x^{(m)}$ and $x^{(m+1)}$ from the jth entries of $p_N^{(m)}$ and $p_N^{(m+1)}$. The rest of the entries are computed recursively with $3N$ operations by

$$x^{(i+1)} = \frac{1}{t_{i,i+1}}[(\lambda_j - t_{i,i})x^{(i)} - t_{i,i-1}x^{(i-1)}], \qquad i = m+1, \cdots, N-1, \tag{4.3a}$$

$$x^{(i-1)} = \frac{1}{t_{i,i-1}}[(\lambda_j - t_{i,i})x^{(i)} - t_{i,i+1}x^{(i+1)}], \qquad i = m, m-1, \cdots, 2. \tag{4.3b}$$

The computation is stable or unstable depending on whether

$$\prod_{i=2}^{N-1} \max\,(|\kappa_{ij}^+|, |\kappa_{ij}^-|) \tag{4.4}$$

is bounded or $\gg 1$.

5. Solution of the updating problem. In this section we follow [1] and [3] in a discussion of the promised $O(N^2)$ method for solving the updating problem (2.3b), i.e., computing the eigensystem of $D_N + \sigma z_N z_N'$. Without loss of generality we may assume that $\sigma > 0$ and that the problem has been deflated, so that the components of $z_N = (z^{(1)} \cdots z^{(N)})'$, as well as the difference between any two diagonal entries of $D_N = \operatorname{diag}(d_{11} < d_{22} < \cdots < d_{NN})$, are different from zero (in practice we take a neighbourhood of zero with a preassigned tolerance, say ε); consult [1], [3], and [4, § 4]. In this case, it follows that the eigenvalues of the updating problem $\lambda^{(i)}$, $i = 1, 2, \cdots, N$, strictly interlace with those of D_N [1, Thm. 1], [3, Thm. 2.1]

$$d_{11} < \lambda^{(1)} < d_{22} < \lambda^{(2)} < \cdots < \lambda^{(N)} < d_{NN} + \sigma \equiv d_{N+1,N+1}. \tag{5.1}$$

With this in mind we now turn to compute the required eigenvalues $\lambda = \lambda^{(i)}$ as the roots of the so-called secular equation [6]

$$f(\lambda) \equiv 1 + \sigma \sum_{j=1}^{N} \frac{(z^{(j)})^2}{d_{jj} - \lambda} = 0. \tag{5.2}$$

The function $f(\lambda)$ is the rational representation of the characteristic polynomial associated with $D_N + \sigma z_N z_N'$, and the interlacing property ensures that f has N simple roots $\lambda^{(i)}$ lying in the open intervals $(d_{ii}, d_{i+1,i+1})$, $i=1,2,\cdots,N$. We shall mention two zero-finders that have been advocated to find these roots:

(1) The zero-finder proposed by Bunch et al. [1], which is based on rational interpolation, employs the values of $f(\lambda)$ and its first derivative, $f'(\lambda)$. The advantage of this zero-finder (which will be referred to as "zeroinder") is that it produces a *monotonic* sequence of approximations in $(d_{ii}, d_{i+1,i+1})$ that converges quadratically to $\lambda^{(i)}$. However, it is very sensitive near the ends of the intervals $(d_{ii}, d_{i+1,i+1})$, where the derivative involved, $f'(\lambda)$, becomes singular.

(2) Cuppen [3] advocated the "zeroinrat" zero-finder of Bus and Dekker [2], which is based on rational interpolation of three f-values in the interval $(d_{ii}, d_{i+1,i+1})$. This algorithm is more robust than the "zeroinder," for it does not involve $f'(\lambda)$; consequently, it avoids the previous difficulty of singular derivatives near d_{ii} and, moreover, it saves half the operations per iteration. Yet, the current "zeroinrat" algorithm lacks the monotonicity property we had before, and, therefore, it requires safeguarding to ensure that we remain within the desired interval (this decreases the convergence rate to ~ 1.839).

Assuming that either one of these zero-finders requires no more than a constant number of iterations to compute (with some preassigned tolerable accuracy) each root of (5.2), then the required eigenvalues $\lambda^{(i)}$, $i=1,2,\cdots,N$, are obtained by $O(N^2)$ operations. Equipped with these eigenvalues, we now may use the Sherman–Morrison formula to compute the associated normalized eigenvectors, $q_N^{(i)}$, which form the column vectors of Q_N in (2.3b), as [1, § 4).

$$(5.3)\qquad q_N^{(i)}=\frac{(D_N-\lambda^{(i)}I_N)^{-1}z_N}{\|(D_N-\lambda^{(i)}I_N)^{-1}z_N\|},\qquad i=1,\cdots,N,$$

and the total operations cost does not exceed $O(N^2)$, as asserted.

To enhance the stability properties of the whole DC algorithm, the updating problem should be solved with maximum accuracy. To achieve this, we now present an efficient implementation for the solution of this problem, based on the ingredients described above.

As a first step we reformulate (5.2) in a manner suggested in [1]. By the interlacing property (5.1) we have

$$(5.4)\qquad \lambda^{(i)}=d_{ii}+\sigma\mu^{(i)},\qquad 0<\mu^{(i)}<1,\qquad \sum_{i=1}^{N}\mu^{(i)}=1.$$

For $i=1,2,\cdots,N$ we make the change of variables, $\lambda=d_{ii}+\sigma\mu$, so that instead of $f\equiv f(\lambda)$, we now obtain N different rational functions, $W_i(\mu)$,

$$(5.5)\qquad W_i(\mu)=1+\sum_{j=1}^{N}\frac{(z^{(j)})^2}{\delta_{ji}-\mu},\qquad \delta_{ji}\equiv\frac{d_{jj}-d_{ii}}{\sigma},\qquad i=1,2,\cdots,N,$$

each of which has a simple root $\mu=\mu^{(i)}$ in the open interval $(\delta_{ii}\equiv 0, \delta_{i+1,i})$. Computing the root of $W_i(\mu)$ in this interval—rather than the root of $f(\lambda)$ in the $(d_{ii}, d_{i+1,i+1})$ interval—has the advantage that $W_i'(\mu)$ is *uniformly* bounded from below (by 1) rather than having $f'(\lambda)\geqq 1/\sigma$, as in [3, Thm. 3.1]. The computation of the desired root proceeds by carefully monitoring a mixture of the two zero-finders mentioned above. Namely, the "zeroinder" algorithm will be used when we are well inside the interval of interest, $(0, \delta_{i+1,i})$, while we switch to the "zeroinrat" algorithm when we approach either end of this interval. To decide upon the switching policy, we first quote the following.

LEMMA 5.1 [4, Lem. 4.6]. *Assume that the deflation test $|z^{(i)}| > \varepsilon$ is satisfied. Then either we have*

$$\frac{\varepsilon^2}{\sigma^2(2+\delta_{i+1,i})}\,\delta_{i+1,i} \leqq \mu^{(i)} \leqq \frac{1}{2}\,\delta_{i+1,i} \tag{5.6a}$$

or alternatively

$$\frac{1}{2}\,\delta_{i+1,i} \leqq \mu^{(i)} \leqq \left(1-\frac{\varepsilon^2}{\sigma^2(2+\delta_{i+1,i})}\right)\delta_{i+1,i}. \tag{5.6b}$$

The bounds on the left- and right-hand sides of (5.6) yield a closed subinterval $[L^{(i)}, H^{(i)}]$, which encloses $\mu^{(i)}$. (Experiments have shown that these bounds actually may be achieved.)

A more practical indication to the location of $\mu^{(i)}$ is obtained from the following considerations. The rational function

$$V_i(\mu) = \text{Const}_i + \frac{(z^{(i)})^2}{-\mu} + \frac{(z^{(i+1)})^2}{\delta_{i+1,i}-\mu}, \qquad \text{Const}_i \equiv \sum_{j\neq i,i+1} \frac{(z^{(j)})^2}{\delta_{ji}-\delta_{i+1,i}} \tag{5.7a}$$

has a simple root, $l^{(i)}$, in the interval $(0, \delta_{i+1,i})$. Since $V_i(\mu)$ dominates $W_i(\mu)$ in that interval and they are both monotonically increasing, we can use this root (that is found by solving a simple quadratic equation) as a lower bound for $\mu^{(i)}$. Similarly, the function

$$U_i(\mu) = \text{Const}_i + \frac{(z^{(i)})^2}{-\mu} + \frac{(z^{(i+1)})^2}{\delta_{i+1,i}-\mu}, \qquad \text{Const}_i \equiv \sum_{j\neq i,i+1} \frac{(z^{(j)})^2}{\delta_{ji}} \tag{5.7b}$$

has a simple root $h^{(i)}$ in the interval $(0, \delta_{i+1,i})$, which may serve as an upper bound for $\mu^{(i)}$.

Returning to our problem of finding the roots of $W_i(\mu)$ in (5.5), we use the "zeroinder" algorithm when inside the $(0, \delta_{i+1,i})$ interval. This requires us to compute $W_i'(\mu)$, and Lemma 5.1 indicates that as we approach either end of the interval, the computation of $W_i'(\mu)$ involves factors of ε^{-4} that will lead to an underflow problem. To avoid this situation, we use a switching policy, which in each step tests if either one of the following inequalities is satisfied:

$$l^{(i)} \leqq L^{(i)}, \qquad h^{(i)} \geqq H^{(i)}, \qquad h^{(i)} \leqq l^{(i)} \tag{5.8}$$

as an indication that we are in the neighborhood of the singular ends, in which case we use the "zeroinrat" algorithm instead. This "switching" policy enabled us to achieve, with the usual 64-bit arithmetic, more than satisfactory results that otherwise would have required the less attractive extended precision arithmetic.

Concerning the computation of the eigenvectors in (5.3), we note that it is possible to have severe round-off when $\lambda^{(i)}$ is close to d_{ii} or $d_{i+1,i+1}$ [3, § 2]. The reformulation of the eigenvalue problem in (5.5) enables one to avoid half of these round-off problems, namely, when $\lambda^{(i)}$ is close to d_{ii}. Indeed, the normalized eigenvectors, $q_N^{(i)}$, are now given by

$$q_N^{(i)} = \frac{[D_N^{(i)}]^{-1} z_N}{\|[D_N^{(i)}]^{-1} z_N\|}, \qquad D_N^{(i)} = \text{diag}\,(\delta_{1i} \cdots \delta_{N_i}) - \mu^{(i)} I. \tag{5.9}$$

Using (5.9) instead of (5.3) avoids cancellation that arises when $\lambda^{(i)}$ is too close to d_{ii}, i.e., when $\mu^{(i)}$ is too close to zero, for $\delta_{ii} \equiv 0$ in this case. We are still left with the other half of the cancellation problem when $\lambda^{(i)}$ is too close to $d_{i+1,i+1}$. If this is indeed the case (as we can foresee by computing the practical bounds for $\mu^{(i)}$ from (5.7a), (5.7b)), then we propose to perform yet another reformulation of our eigenvalue problem, using

$$\lambda^{(i)} = d_{i+1,i+1} - \sigma\eta^{(i)} \tag{5.10}$$

instead of (5.4). In this case the role of the rational functions $W_i(\mu)$ in (5.5) is played by

$$\bar{W}_i(\eta) = 1 + \sum_{j=1}^{N} \frac{(z^{(j)})^2}{\delta_{j,i+1} - \eta}, \qquad \delta_{j,i+1} = \frac{d_{jj} - d_{i+1,i+1}}{\sigma}, \qquad i = 1, 2, \cdots, n, \tag{5.11}$$

each of which has a simple root $\eta = \eta^{(i)}$ in the open interval $(0, -\delta_{i,i+1})$. The corresponding normalized eigenvectors are given by

$$q_N^{(i)} = \frac{[\bar{D}_N^{(i)}]^{-1} z_N}{\|[\bar{D}_N^{(i)}]^{-1} z_N\|}, \qquad \bar{D}_N^{(i)} = \text{diag}\,(\delta_{1,i+1} \cdots \delta_{N,i+1}) + \eta^{(i)} I_N, \tag{5.12}$$

and cancellation that arises when $\lambda^{(i)}$ is too close to $d_{i+1,i+1}$, i.e., when $\eta^{(i)}$ is too close to zero is avoided for $\delta_{i+1,i+1} \equiv 0$.

6. Numerical experiments. The main object of our experiments was a comparison between the standard $O(N^3)$ DC algorithm for computing the eigensystems of $N \times N$ ST matrices (2.3a)–(2.3c), and the proposed $O(N^2)$ method in (2.4a), (2.4b), and (3.4), which makes use of the three-term recurrence relations (3.5a), (3.5b). The input data for these relations, the eigenvalues $\lambda^{(i)}$ and the two initial successive row vectors $p_N^{(m)}$, $p_N^{(m+1)}$ were supplied with maximum accuracy, with the help of the updating solver described in § 5 that avoids extended precision. Indeed all our calculations, including the pathologically ill-conditioned W_{+N}- Wilkinson's matrices, were carried out with a 64-bit arithmetic.

The first set of results includes "well-behaved" matrices taken from [7, (7.4)–(7.9)]. The entries along the diagonals of these matrices are "slowly varying" and their eigenvalues are equally distributed. The stability analysis in § 3 indicates bounded amplification factors in these cases, and the numerical results confirmed the expected stable behavior of our method. Table 1 summarizes the results for the prototype ST matrix of this group where $T_{ij} = 2 - |i - j|$.

Since the rows of P were constructed by equating to zero rows $2, 3, \cdots, N-1$ of $TP - P\Lambda$, the quantities on the left columns, $\|TP - P\Lambda\|_\infty$, stand for

$$\max\,(\|t_{1,1}p^{(1)} + t_{1,2}p^{(2)} - p^{(1)}\Lambda\|, \|t_{N,N-1}p^{(N-1)} + t_{N,N}p^{(N)} - p^{(N)}\Lambda\|).$$

They may serve us as a quantitive indication of the accumulation of rounding errors in the three-term recursion relations (3.5), which is responsible for the loss of (no more than) two orders of magnitude relative to the standard algorithm. The advantage of the proposed method lies upon the fact that the results on the right columns are

TABLE 1
Results for $T[1, 2, 1]$ matrix of order N.

	Standard DC algorithm		The proposed method	
N	$\|TP-P\Lambda\|_\infty$	$\|P'P-I\|_\infty$	$\|TP-P\Lambda\|_\infty$	$\|P'P-I\|_\infty$
101	2.5E − 15	6.2E − 16	9.5E − 15	7.2E − 15
201	2.6E − 15	2.5E − 15	2.2E − 14	1.5E − 14
301	3.0E − 15	2.8E − 15	2.9E − 14	8.8E − 14
401	4.0E − 15	6.9E − 15	2.5E − 13	1.2E − 13

obtained by saving order of magnitude in execution time relative to the results on the left columns.

Next, we turn to the second group of matrices that consists of Wilkinson's matrices, W_{+N}. The superdiagonals of these matrices are properly scaled to begin with—they all equal one; the entries along the main diagonal, however, diag $((N-1)/2, (N-3)/2, \cdots, 1, 0, 1, \cdots, (N-1)/2)$ are far from being "slowly varying." This leads to amplification factors of the recurrence relations (3.5) of order $\sim(N-1)/2!$, which indicates loss of all (64-bit precision) significant figures in computing the eigenproblem of W_{+N} of order $N \gtrsim 40$. Moreover, the largest eigenvalues of W_{+N} are clustered in pairs, which may be inseparable up to the 14th decimal digit. This then leads to additional inaccuracies in the updating solution (while seeking two extremely close roots of the secular equation) as well as in the deflation process. As a result, the initial input data for the recurrence relation will also suffer from loss of accuracy. These arguments are well reflected in Table 2.

In order to be competitive with the standard algorithm that gave excellent results for W_{+N} up to order $N=201$, an attempt was made to improve the results of our method. To this end we have appended our method with the *restarting* procedure described in § 3. Thus, by computing the row vectors (here $m=(N+1)/4$) $p_N^{(m)}$, $p_N^{(m+1)}$, $p_N^{(3m)}$, and $p_N^{(3m+1)}$ as additional input data to restart the three-term recurrence relations, we were able to get decent results for the W_{+N}-matrices up to order $N \sim 200$. Repeating such restarting procedures would enable us to deal with even larger W_{+N}-matrices, still within the $O(N^2)$ operations limit.

Finally, the last group of matrices that were tested consists of randomly generated entries in $[-1, 1]$. The results obtained are summarized in Table 3.

We observe that excellent results are obtained by our method for such randomly generated matrices of order up to $N \sim 100$. If additional restarting procedures were employed every 100–200 iterations, it would enable us to achieve highly accurate results for matrices of almost any practical size.

TABLE 2

Results for the W_{+N} matrices.

	Standard DC algorithm		The proposed method	
N	$\|TP-P\Lambda\|_\infty$	$\|P'P-I\|_\infty$	$\|TP-P\Lambda\|_\infty$	$\|P'P-I\|_\infty$
21	4.5E−16	2.5E−16	1.2E−12	9.8E−10
41	1.3E−15	9.4E−16	3.7E−8	7.4E−7
47	2.0E−15	9.1E−16	5.3E−3	1.0E−3
49	2.0E−15	9.8E−16	0.12	0.23

TABLE 3

Results for random matrices of order N.

	Standard DC algorithm		The proposed method	
N	$\|TP-P\Lambda\|_\infty$	$\|P'P-I\|_\infty$	$\|TP-P\Lambda\|_\infty$	$\|P'P-I\|_\infty$
100	8.4E−15	9.8E−16	9.5E−15	7.6E−15
200	5.9E−15	3.4E−15	6.2E−9	9.8E−8
300	6.3E−15	5.6E−15	4.2E−4	3.1E−2
400	7.2E−15	6.8E−15	$O(1)$	$O(1)$

In summary, we conclude that the proposed method for solving the eigenproblem of ST matrices provides a competitive alternative to the standard eigensolvers for a certain class of such matrices; by sacrificing a *few* orders of accuracy, the method enables one to save *order of magnitude* in the total execution time. This conclusion was confirmed by further extensive numerical experiments reported in [5].

Acknowledgment. We thank Professor Beresford Parlett for an enlightening discussion on the Inverse Iteration method.

REFERENCES

[1] J. R. BUNCH, C. P. NEILSEN, AND D. C. SORENSEN, *Rank one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.
[2] J. C. BUS AND T. J. DEKKER, *Two efficient algorithms with guaranteed convergence for finding a zero of a function*, TOMS 1, 1975, pp. 330–345.
[3] J. J. M. CUPPEN, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.
[4] J. J. DONGARRA AND D. C. SORENSEN, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 139–154.
[5] D. GILL, *Divide and conquer—A parallel algorithm for computing the spectral decomposition of symmetric matrices*, M.Sc. thesis, Tel-Aviv University, Tel-Aviv, Israel, 1987.
[6] G. H. GOLUB, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.
[7] R. T. GREGORY AND D. L. KARNEY, *A Collection of Matrices for Testing Computational Algorithms*, John Wiley, New York, 1969.
[8] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[9] G. PETERS AND J. H. WILKINSON, *Eigenvalues of $Ax = \lambda Bx$ with band symmetric* A *and* B, Comput. J., 12 (1969), pp. 398–404.
[10] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.