NORTH-HOLLAND

A Fast and Stable Parallel *QR* Algorithm for Symmetric Tridiagonal Matrices

Ilan Bar-On
Department of Computer Science
Technion
Haifa 32000, Israel

and

Bruno Codenotti*
IEI-CNR
Via S. Maria 46
56100-Pisa, Italy

Submitted by Daniel Hershkowitz

ABSTRACT

We present a new, fast, and practical parallel algorithm for computing a few eigenvalues of a symmetric tridiagonal matrix by the explicit *QR* method. We present a new divide and conquer parallel algorithm which is fast and numerically stable. The algorithm is work efficient and of low communication overhead, and it can be used to solve very large problems infeasible by sequential methods.

1. INTRODUCTION

Very large band linear systems arise frequently in computational science, directly from finite difference and finite element schemes, and indirectly from applying the symmetric block Lanczos algorithm to sparse systems, or the symmetric block Householder transformation to dense systems. In this paper we present a new divide and conquer parallel algorithm for computing

* Partially supported by ESPRIT B.R.A. III of the EC under contract n. 9072 (project GEPPCOM).

0024-3795/95/$9.50
SSDI 0024-3795(93)00360-C

a few eigenvalues of a symmetric tridiagonal matrix by the explicit *QR* method. Our algorithm is fast and numerically stable. Moreover, the algorithm is work efficient, with low communication overhead, and it can be used in practice to solve very large problems on massively parallel systems, problems infeasible on sequential machines. We conjecture that our method can be generalized to band systems as well.

The *QR* algorithm, developed by Francis [8, 9], is an orthogonal successor to the *LR* algorithm of Rutishauser. We quote here Wilkinson, who said that the *LR* method is "the most significant advance which has been made in connection with the eigenvalue problem since the advent of automatic computers" [23, p. 485]. In fact, the *LR* algorithm for the symmetric tridiagonal case is very efficient, stable, and with cubic convergence rate [18]. An efficient parallel implementation of this method is presented in Bar-On [3]. However, the *LR* method has several limitations. The eigenvalues are found in increasing order, from the smallest to the largest, and it is not possible to locate an intermediate eigenvalue directly. On the other hand, the *QR* algorithm has similar properties but can be applied to compute any eigenvalue, given a sufficiently close initial approximation.

Before discussing this method we would like to mention some other related methods for locating a few eigenvalues. Bisection is a popular sequential method based on the Sturm sequence properties, for locating some ordered eigenvalues or some eigenvalues in a given interval, but its convergence rate is only linear. Hence, it may be used to locate an approximation from which other and faster methods should be used. A new divide and conquer parallel bisection algorithm, both stable and efficient, is given in Bar-On [2] and has the benefits of having the matrix subdivided between the processors through all stages with little communication overhead. This is not the case for other parallel variants such as multisection [12, 11], which is inefficient and requires that the whole matrix be shared by all the processors in the system. Another approach to this problem is a combination of bisection with inverse iteration; see Peters and Wilkinson [16]. A new factorization for the efficient parallel implementation of the inverse iteration method is given in Bar-On and Munk [6].

We turn now our attention to the literature on the *QR* method for symmetric tridiagonal matrices. To our knowledge, the only specific result in this field is by Sameh and Kuck [19] from 1977. In their paper they present a parallel implementation of a variant of the *QR* method of Reinsch [17] which runs in $O(\log n)$ time with $O(n)$ processors, where n is the order of the matrix. Hence, the efficiency of that algorithm is of order $1/\log n$ only. Furthermore, their numerical results are only for very small size matrices (up to order 128), for which the accuracy is not as good as for the sequential method, and the theoretical error analysis implies an exponential error growth. Moreover, their method cannot be applied to general band systems.

Finally, we would like to mention the survey paper of Ipsen and Jessup [11], which concludes that the "shifted QR algorithm alone does not seem to have an efficient parallel implementation." We will rebut this conjecture in this paper.

A very fast and efficient parallel QR algorithm, based on the Cholesky decomposition, was first considered in Bar-On [1, 3]. However, in general, this method seems to be somewhat less accurate than the sequential algorithm. In this paper we would like to present a new divide and conquer parallel QR algorithm which theoretically and experimentally seems to be as stable as the sequential method. For the sake of clarity and to help the reader fix ideas, we will consider models of parallel computation with the following features:

(i) There are p processors, which are connected by a network; the network can at least support fan-in algorithms with optimal speedup.

(ii) The network is synchronous.

(iii) Each processor can access the data without a time penalty.

(iv) Each processor has a local memory.

(v) The processors do not have access to a shared memory.

Note that we do not assume the existence of a common memory, because our algorithm has a very small communication overhead that allows us to transfer messages between pairs of processors with optimal speedup.

2. THE SEQUENTIAL QR ALGORITHM FOR SYMMETRIC TRIDIAGONAL MATRICES

In this section we review the basic features of the sequential QR algorithm, and the method by which it can be adapted to parallel machines. Let A be an $n \times n$ real symmetric tridiagonal matrix, and let us seek some of its eigenvalues close to the real number r.

Let $A_0 = A - rI$, and $z_0 = r$. Repeat the following:

- For $s = 0, 1, \dots,$ do
  1. Choose a shift y_s.
  2. Find the QR decomposition of $A_s - y_s I = Q_s R_s$.
  3. Set $A_{(s+1)} = R_s Q_s$ and $z_{(s+1)} = z_s + y_s$.

  until the last off diagonal element becomes negligible.
- The computed eigenvalue is $z_{(s+1)}$ plus the last diagonal element.
- Deflate the matrix, and proceed as above to get the next eigenvalue.

An implementation of this sort is given in the EISPACK routine BQR [20]. Although the shifting strategy can be quite involved (see Parlett [15, Chapter

8], "Shifts for all seasons"), our main concern in this paper is with the parallelization of the basic transformation,

$$A = QR \quad \Rightarrow \quad \tilde{A} = Q^t A Q = RQ. \tag{1}$$

In practice, we do not actually compute the orthogonal matrix Q, but apply it implicitly by a sequence of Givens rotations as follows:

$$A \equiv A_0 \to A_1 = G_2^t A_0 \tag{2}$$

$$\to A_2 = G_3^t A_1 G_2 \to \cdots$$

$$\to A_{(n-1)} = G_n^t A_{n-2} G_{(n-1)} \tag{3}$$

$$\to A_n = A_{(n-1)} G_n \equiv \tilde{A}, \tag{4}$$

where G_i^t, $i = 2, \dots, n$, is the Givens rotation that annihilates the $(i, i-1)$th element of A. The process looks highly serial, but a closer examination reveals its high potential for parallelization. We need look at a more specific example, such as the following one, for $n = 4$:

$$A_0 = \begin{pmatrix} a_1 & b_1 & & \\ b_1 & a_2 & b_2 & \\ & b_2 & a_3 & b_3 \\ & & b_3 & a_4 \end{pmatrix} \quad \Rightarrow \quad A_1 = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ & x_2 & y_2 & \\ & b_2 & a_3 & b_3 \\ & & b_3 & a_4 \end{pmatrix} \tag{5}$$

$$\Rightarrow \quad A_2 = \begin{pmatrix} d_1 & e_1 & & \\ e_1 & \alpha_2 & \beta_2 & \\ & & x_3 & y_3 \\ & & b_3 & a_4 \end{pmatrix}$$

$$\Rightarrow \quad A_3 = \begin{pmatrix} d_1 & e_1 & & \\ e_1 & d_2 & e_2 & \\ & e_2 & \alpha_3 & \beta_3 \\ & & & x_4 \end{pmatrix} \tag{6}$$

$$\Rightarrow \quad A_4 = \begin{pmatrix} d_1 & e_1 & & \\ e_1 & d_2 & e_2 & \\ & e_2 & d_3 & e_3 \\ & & e_3 & d_4 \end{pmatrix}. \tag{7}$$

Note that in each of the above steps, only the necessary computations are performed, and the upper off-diagonal elements are obtained by symmetry from the lower ones. We conclude that if we were able to know in advance the value of x_i, y_i for some $1 \leqslant i \leqslant n$, we could then have computed the lower off diagonal elements e_j, $j = i, \dots, n-1$, and diagonal elements d_j, $j = i+1, \dots, n$, without any further information. More generally, let A be an $N \times N$ symmetric tridiagonal matrix, where $N = np$ and p is the number of processors available. Consider A as a $p \times p$ block matrix with blocks of order n, namely,

$$A = \begin{pmatrix} A_1 & B_1 & & & \\ C_1 & A_2 & B_2 & & \\ & \ddots & \ddots & \ddots & \\ & & C_{p-2} & A_{p-1} & B_{p-1} \\ & & & C_{p-1} & A_p \end{pmatrix}, \qquad B_i = \begin{pmatrix} 0 & 0 \\ b_{in} & 0 \end{pmatrix}, \tag{8}$$

where $C_i = B_i^t$, and A_i is symmetric and tridiagonal. We could then suggest the following algorithm for computing the transformed matrix $\tilde{A}$:

- *Divide and conquer.* For $i = 1, \dots, p$, compute in advance the pairs $(x_{(i-1)n}, y_{(i-1)n})$. Let T_i be the $(n+2) \times (n+2)$ matrix

$$T_i = \begin{pmatrix} x_{(i-1)n} & y_{(i-1)n} & & & \\ b_{(i-1)n} & a_{(i-1)n+1} & b_{(i-1)n+1} & & \\ & \ddots & \ddots & \ddots & \\ & & b_{in-1} & a_{in} & b_{in} \\ & & & b_{in} & a_{in+1} \end{pmatrix}, \tag{9}$$

  where we assume $x_0 = a_{N+1} = 1$ and $y_0 = b_0 = b_N = 0$.
- *Parallel QR.* Apply the sequential QR algorithm independently and in parallel to each T_i, $i = 1, \dots, p$. We denote by $\tilde{T}_i$ the transformed matrices, and observe that

$$\tilde{T}_i = \begin{pmatrix} * & e_{(i-1)n} & & & \\ e_{(i-1)n} & d_{(i-1)n+1} & * & & \\ \ddots & \ddots & \ddots & & \\ & * & * & e_{in-1} & \\ & & e_{in-1} & d_{in} & * \\ & & & * & * \end{pmatrix}. \tag{10}$$

Hence, the n middle diagonal and off-diagonal elements in each $\tilde{T}_i$ give the corresponding elements of $\tilde{A}$.

The correctness of the above algorithm follows immediately from the above discussion, as the operations on rows below row *in* do not affect the values above it. Note that by the end of each iteration, there is no need to produce the whole matrix $\tilde{A}$, as it should remain subdivided for the following iterations. Only the extreme rows are exchanged between adjacent processors before the next iteration resumes. Hence, in terms of both complexity and communication issues, the parallel QR stage is very efficient. However, in order to devise an efficient parallel algorithm that exploits the above possible parallelization, we need answer the following two questions:

1. Can we compute the above x's and y's efficiently in parallel?
2. What is the effect of rounding errors on the stability of the algorithm?

A positive answer to both of these questions is given in the remainder of this paper. We provide the mathematical and numerical foundations in Section 3, and present and analyze the parallel algorithm in Section 4. An elaborate discussion of the numerical stability of the algorithm is given in Section 5. We finally report on some open related problems in the conclusion, and provide additional details in the appendix.

3. MATHEMATICAL ANALYSIS

We assume that computations are over the set of real numbers, and we view n-vectors as elements of R^n. We denote by $\mathcal{M}(n)$ and by $\mathcal{M}(n, m)$ the classes of $n \times n$ and $n \times m$ matrices respectively. We denote the ith row of $A \in \mathcal{M}(n, m)$ by $A_{[i]}$, its jth column by $A^{[j]}$, and its (i, j)th element by a_{ij}. More generally, we denote rows i_0 to i_1 by $A_{[i_0:i_1]}$, columns j_0 to j_1 by $A^{[j_0:j_1]}$, and the submatrix of rows i_0 to i_1 and columns j_0 to j_1 by $A_{[i_0:i_1]}^{[j_0:j_1]}$. We denote an $N \times N$ symmetric tridiagonal matrix by $A = \operatorname{diag}(b_{i-1}, a_i, b_i)$, $i = 1, \ldots, N$, where it is assumed implicitly that b_0, b_N are missing. We say that A is unreduced if all the off diagonal elements are nonzero, i.e., $b_i \neq 0$, $i = 1, \ldots, N - 1$. We note that if some of the off diagonal elements become zero, the problem decouples into independent subproblems which can then be dealt with in parallel. Henceforth, we will assume that the underlying tridiagonal matrix is unreduced. Note that, in practice, we are dealing with finite precision, and an element becomes zero when it reaches some given threshold.

LEMMA 3.1. *Let A be an unreduced symmetric tridiagonal matrix of order $N > n$, and let $B = A_{[1:n]}^{[1:(n+1)]}$. Let $Q \in \mathscr{M}(n)$ be an orthogonal matrix such that*

$$C = Q^t B = \begin{pmatrix} X & v & w \\ 0 & x & y \end{pmatrix}, \qquad X \in \mathscr{M}(n-1), \quad v, w \in R^{n-1}. \quad (11)$$

Then the last row of C is unique up to a sign change.

Proof. Since A is unreduced, then $B^{[1:(n-1)]} = (B^{[1]} \cdots B^{[(n-1)]}) \in \mathscr{M}(n \times (n-1))$ has full rank, and X is nonsingular. Hence, there is a unique factorization $X = \tilde{Q}\tilde{R}$, where the diagonal of $\tilde{R}$ is taken to be positive. Substituting in (11), we get the QR factorization of B, i.e.,

$$B = \hat{Q}R, \qquad \hat{Q} = QQ', \qquad Q' = \begin{pmatrix} \tilde{Q} & 0 \\ 0 & 1 \end{pmatrix}, \qquad R = \begin{pmatrix} \tilde{R} & v' & w' \\ 0 & x & y \end{pmatrix}, \quad (12)$$

which is unique up to the sign of the last column of $\hat{Q}$, i.e. $\hat{Q}^{[n]} = Q^{[n]}$. Hence, the last row $C_{[n]} = Q_{[n]}^t B$ is unique up to a sign change. ■

From Lemma 3.1 we have the following:

COROLLARY 3.2. *Let A be an unreduced symmetric tridiagonal matrix of order $N = np$, and let $B_i = A_{[1:in]}^{[1:(in+1)]}$ for $i = 1, \dots, p-1$. Let Q_i be an orthogonal matrix such that*

$$C_i = Q_i^t B_i \begin{pmatrix} X_i & v_i & w_i \\ 0 & x_{in} & y_{in} \end{pmatrix}, \qquad X_i \in \mathscr{M}(in-1), \quad v_i, w_i \in R^{in-1}. \quad (13)$$

Then (x_{in}, y_{in}) are unique up to a sign change.

For completeness, we state the following; see Parlett [15, Chapter 7].

PROPOSITION 3.3. *The QR transformation of an unreduced symmetric tridiagonal matrix is unique up to a sign change of its off diagonal elements.*

We therefore call two QR transformations of A *equivalent* if they are the same up to the sign of the off diagonal elements.

LEMMA 3.4. *Let G_1, G_2 be two Given rotations such that*

$$G_1^t A = \begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix} \begin{pmatrix} x_0 & y_0 & 0 \\ b_0 & a_1 & b_1 \end{pmatrix} = \begin{pmatrix} \alpha_1 & \beta_1 & * \\ 0 & x_1 & y_1 \end{pmatrix}, \tag{14}$$

$$G_2^t \hat{A} = \begin{pmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{pmatrix} \begin{pmatrix} -x_0 & -y_0 & 0 \\ b_0 & a_1 & b_1 \end{pmatrix} = \begin{pmatrix} \alpha_2 & \beta_2 & * \\ 0 & x_2 & y_2 \end{pmatrix}. \tag{15}$$

where we assume that $b_0 \neq 0$. Then

$$G^t A = \begin{pmatrix} c_1 & s_1 \\ s_2 & c_2 \end{pmatrix} \begin{pmatrix} x_0 & y_0 & 0 \\ b_0 & a_1 & b_1 \end{pmatrix} = \begin{pmatrix} \alpha_1 & \beta_1 & * \\ 0 & x_2 & y_2 \end{pmatrix}, \tag{16}$$

and G is orthogonal.

Proof. The correctness of Equation (16) is readily verified by inspection. In case $c_2 = c_1$ and $s_2 = -s_1$, we are done. Otherwise, by assumption $b_0 \neq 0$, so that the first column of G_2 is unique up to a sign change. Hence, $c_2 = -c_1$ and $s_2 = s_1$, so G is orthogonal. ■

We will say that the orthogonal matrix G simulates the action of the corresponding two orthogonal matrices G_1 and G_2. We are now ready to present our main theorem.

THEOREM 3.5. *Consider a sequential QR transformation applied to A, and let T_i, $i = 1, \dots, p$, denote the respective blocks as in (9). Let $\hat{T}_i$, $i = 1, \dots, p$, denote the same matrix as T_i except for a possible sign change of the first row, i.e.,*

$$\left(\hat{x}_{(i-1)n}, \hat{y}_{(i-1)n} \right) = \pm \left(x_{(i-1)n}, y_{(i-1)n} \right). \tag{17}$$

Let us apply the parallel QR transformations to $\hat{T}_i$ and denote by

$$\bar{T}_i = \operatorname{diag}\left(\bar{e}_{(j-1)}, \bar{d}_j, \bar{e}_j \right), \qquad j = (i-1)n + 1, \dots, in, \tag{18}$$

the corresponding transformed matrices as in (10). *Then the transformed matrices $\tilde{A}$ and $\bar{A}$ are equivalent.*

Proof. We will present a sequential QR transformation for which the corresponding transformed matrix $\tilde{A}$ is equivalent to the computed one $\bar{A}$. The proof is by induction on the number of blocks. The basis of the induction for $k = 1$ is trivial. We prove the inductive step, showing that the theorem is true for $k > 1$ blocks, provided it is true for $k - 1$ blocks. By induction, the first $k - 1$ parallel transformations can be simulated by an equivalent transformation applied to $A^{[1:(k-1)n+1]}_{[1:(k-1)n+1]}$. Hence, this sequential transformation, together with the kth transformation applied to $\hat{T}_k$, is equivalent to the whole set of k parallel operations. We may therefore assume w.l.o.g. that $p = 2$, and consider the parallel QR transformations applied to $\hat{T}_1$, $\hat{T}_2$. (Note that $\hat{T}_1 = T_1$.) We will then show that these can be simulated by an equivalent transformation applied to A.

We start by applying to A the same sequence of orthogonal transformations applied to $\hat{T}_1$, up to the nth row. At this stage we consider the respective transformed matrices $\bar{T}_{1,(n-1)}$, $\tilde{A}_{(n-1)}$. Let $T_{1,(n-1)}$ denote the last three rows of $\bar{T}_{1,(n-1)}$, and let $T_{2,0}$ denote the first three rows of $\hat{T}_2$, i.e.,

$$T_{1,(n-1)} = \begin{pmatrix} \alpha_{(n-1)} & \beta_{(n-1)} & \\ & x_n & y_n \\ & b_n & a_{(n+1)} \end{pmatrix}, \qquad T_{2,0} = \begin{pmatrix} \hat{x}_n & \hat{y}_n & \\ b_n & a_{(n+1)} & b_{(n+1)} \\ & b_{(n+1)} & a_{(n+2)} \end{pmatrix}. \tag{19}$$

Similarly, let $A_{(n-1)}$ denote the respective rows of $\tilde{A}_{(n-1)}$, i.e.

$$A_{(n-1)} = \begin{pmatrix} \alpha_{(n-1)} & \beta_{(n-1)} & & \\ & x_n & y_n & \\ & b_n & a_{(n+1)} & b_{(n+1)} \\ & & b_{(n+1)} & b_{(n+2)} \end{pmatrix}. \tag{20}$$

In addition note that $(\hat{x}_n, \hat{y}_n) = \pm(x_n, y_n)$ from Corollary 3.2. In case $(\hat{x}_n, \hat{y}_n) = (x_n, y_n)$, we proceed immediately with the respective rotations of $\hat{T}_2$, and we are done. We therefore assume henceforth that the signs have been changed.

Let G_n be the last Givens rotation applied to the first two rows of $T_{1,(n-1)}$, and let $G_{(n+1)}$ be the next. Then we proceed with $T'_{1,n} = G^t_{n+1}T_{1,(n-1)}$, i.e.

$$T'_{1,n} = \begin{pmatrix} 1 & & \\ & c_{(n+1)} & s_{(n+1)} \\ & -s_{(n+1)} & c_{(n+1)} \end{pmatrix} \begin{pmatrix} \alpha_{(n-1)} & \beta_{(n-1)} & \\ & x_n & y_n \\ & b_n & a_{(n+1)} \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_{(n-1)} & \beta_{(n-1)} & \\ & \alpha'_n & \beta_n \\ & & * \end{pmatrix}, \tag{21}$$

to be followed by $T_{1,n} = T'_{1,n}G_n$, i.e.

$$T_{1,n} = \begin{pmatrix} \alpha_{(n-1)} & \beta_{(n-1)} & \\ & \alpha'_n & \beta_n \\ & & * \end{pmatrix} \begin{pmatrix} c_n & -s_n & \\ s_n & c_n & \\ & & 1 \end{pmatrix}$$

$$= \begin{pmatrix} d_{(n-1)} & e_{(n-1)} & \\ e_{(n-1)} & \alpha_n & \beta_n \\ & & * \end{pmatrix}, \tag{22}$$

where $e_{(n-1)} = \alpha'_n s_n$ and $\alpha_n = \alpha'_n c_n$. Finally, we obtain $T_{1,(n+1)} = T_{1,n}G_{(n+1)}$,

$$T_{1,(n+1)} = \begin{pmatrix} d_{(n-1)} & e_{(n-1)} & \\ e_{(n-1)} & \alpha_n & \beta_n \\ & * & * \end{pmatrix} \begin{pmatrix} 1 & & \\ & c_{(n+1)} & -s_{(n+1)} \\ & s_{(n+1)} & c_{(n+1)} \end{pmatrix}$$

$$= \begin{pmatrix} d_{(n-1)} & e_{(n-1)} & \\ e_{(n-1)} & d_n & * \\ & & * \end{pmatrix}, \tag{23}$$

where $d_n = \alpha_n c_{(n+1)} + \beta_n s_{(n+1)}$.

Next, we consider the rotations of $\hat{T}_2$. Let $\hat{G}_{n+1}$, $G_{(n+2)}$ be the first two row rotations to be applied to $T_{2,0}$. We then obtain $T_{2,1} = \hat{G}^t \mathrm{r}_{(n+1)} T_{2,0}$, i.e.,

$$T_{2,1} = \begin{pmatrix} \hat{c}_{(n+1)} & \hat{s}_{(n+1)} & \\ -\hat{s}_{(n+1)} & \hat{c}_{(n+1)} & \\ & & 1 \end{pmatrix} \begin{pmatrix} -x_n & -y_n & \\ b_n & a_{(n+1)} & b_{(n+1)} \\ & b_{(n+1)} & a_{(n+2)} \end{pmatrix}$$

$$= \begin{pmatrix} * & * & \\ & x_{(n+1)} & y_{(n+1)} \\ & b_{(n+1)} & a_{(n+2)} \end{pmatrix}, \tag{24}$$

and from $T'_{2,2} = G^t_{(n+2)} T_{2,1}$ we get $T_{2,2} = T'_{2,2} \hat{G}_{(n+1)}$, i.e.

$$T_{2,2} = \begin{pmatrix} * & * & \\ & \alpha'_{(n+1)} & \beta_{(n+1)} \\ & & x_{(n+2)} \end{pmatrix} \begin{pmatrix} \hat{c}_{(n+1)} & -\hat{s}_{(n+1)} & \\ \hat{s}_{(n+1)} & \hat{c}_{(n+1)} & \\ & & 1 \end{pmatrix}$$

$$= \begin{pmatrix} * & * & \\ e_n & \alpha_{(n+1)} & \beta_{(n+1)} \\ & & x_{(n+2)} \end{pmatrix}, \tag{25}$$

where $e_n = \alpha'_{(n+1)} \hat{s}_{(n+1)}$, and $\alpha_{(n+1)} = \alpha'_{(n+1)} \hat{c}_{(n+1)}$. We now present an equivalent sequential transformation that has the same effect as these two parallel transformations. Let $A_{(n-1)}$ be as in (20), and let G be the orthogonal matrix that simulates the action of the two orthogonal matrices G_{n+1} and $\hat{G}_{n+1}$. Then we proceed with $A'_n = G^t A_{(n-1)}$, i.e.

$$A'_n = \begin{pmatrix} 1 & & & \\ & c_{(n+1)} & s_{(n+1)} & \\ & \hat{s}_{(n+1)} & \hat{c}_{(n+1)} & 1 \end{pmatrix} \begin{pmatrix} \alpha_{(n-1)} & \beta_{(n-1)} & & \\ & x_n & y_n & \\ & b_n & a_{(n+1)} & b_{(n+1)} \\ & & b_{(n+1)} & a_{(n+2)} \end{pmatrix} \tag{26}$$

$$= \begin{pmatrix} \alpha_{(n-1)} & \beta_{(n-1)} & & \\ & \alpha'_n & \beta_n & \\ & & x_{(n+1)} & y_{(n+1)} \\ & & b_{(n+1)} & a_{(n+2)} \end{pmatrix}, \tag{27}$$

to be followed by $A_n = A'_n G_n$, i.e.

$$A_n = \begin{pmatrix} \alpha_{(n-1)} & \beta_{(n-1)} & & \\ & \alpha'_n & \beta_n & \\ & & x_{(n+1)} & y_{(n+1)} \\ & & b_{(n+1)} & a_{(n+2)} \end{pmatrix} \begin{pmatrix} c_n & -s_n & & \\ s_n & c_n & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \tag{28}$$

$$= \begin{pmatrix} d_{(n-1)} & e_{(n-1)} & & \\ e_{(n-1)} & \alpha_n & \beta_n & \\ & & x_{(n+1)} & y_{(n+1)} \\ & & b_{(n+1)} & a_{(n+2)} \end{pmatrix}. \tag{29}$$

At this point we conclude the following:

1. The first $n - 1$ subdiagonal and diagonal elements, i.e.

$$e_{i-1}, d_i, \qquad i = 1, \dots, n-1, \tag{30}$$

are the same as in $\overline{T}_1$.

2. Proceeding with the respective rotations of $\hat{T}_2$, the last $n - 1$ subdiagonal and diagonal elements, i.e.

$$e_{i-1}, d_i, \qquad i = n+2, \dots, N, \tag{31}$$

will be the same as in $\overline{T}_2$.

Hence, the only difference between $\tilde{A}$ and the computed $\overline{A}$ can occur in the two middle rows of $A_{(n+2)}$. Proceeding with the next rotation of $\hat{T}_2$, we obtain $A'_{(n+1)} = G^t_{(n+2)} A_n$, and then $A_{(n+1)} = A'_{(n+1)} G$, i.e.

$$A_{(n+1)} = \begin{pmatrix} d_{(n-1)} & e_{(n-1)} & & \\ e_{(n-1)} & \alpha_n & \beta_n & \\ & & \alpha'_{(n+1)} & \beta_{(n+1)} \\ & & & x_{(n+2)} \end{pmatrix} \times \begin{pmatrix} 1 & & & \\ & c_{(n+1)} & \hat{s}_{(n+1)} & \\ & s_{(n+1)} & \hat{c}_{(n+1)} & \\ & & & 1 \end{pmatrix} \tag{32}$$

$$= \begin{pmatrix} d_{(n-1)} & e_{(n-1)} & & \\ e_{(n-1)} & d_n & \hat{e}_n & \\ & \hat{e}_n & \alpha_{(n+1)} & \beta_{(n+1)} \\ & & & x_{(n+2)} \end{pmatrix}. \tag{33}$$

Comparing with Equation (23) and with Equation (25), we observe that the only change that may occur is in $\hat{e}_n = \alpha'_{(n+1)} s_{(n+1)}$. However, $s_{(n+1)} = \pm\hat{s}_{(n+1)}$, from Lemma 3.4, and therefore $\hat{e}_n = \pm e_n$. Hence, from Proposition 3.3, the transformed matrix $\hat{A}$ is equivalent to $\bar{A}$. ■

COROLLARY 3.6. *Let A be an $N \times N$ unreduced symmetric tridiagonal matrix, and suppose we have computed the $(\hat{x}_{in}, \hat{y}_{in})$, $i = 1, \dots, p-1$, as in* (13), *by some orthogonalization method. Then we can apply the parallel QR stage to the corresponding blocks $\hat{T}_i$ of Theorem* 3.5, *and obtain an equivalent QR transformation of A.*

4. DIVIDE AND CONQUER

We present in this section the divide and conquer parallel algorithm for the precomputation of the x, y pairs. We assume as before that A is an $N \times N$ symmetric tridiagonal matrix where $N = np$ and $p = 2^k$ is the number of processors available. We further assume that the matrix is initially divided into blocks of rows between the processors, each having n consecutive rows. We denote these blocks $T_i \in \mathcal{M}(n, n+2)$, $i = 1, \dots, p$, by

$$T_i = \begin{pmatrix} b_{(i-1)n} & a_{(i-1)n+1} & b_{(i-1)n+1} & & \\ & \ddots & \ddots & \ddots & \\ & & b_{in-1} & a_{in} & b_{in} \end{pmatrix}. \tag{34}$$

We compute the x, y pairs in three stages as follows: (i) diagonalization, (ii) bottom-up sweep, (iii) top-down sweep. In the diagonalization stage the processors work completely in parallel with no communication overhead, doing approximately the same amount of work. In parallel complexity terms this is the most significant stage of the algorithm. In fact, its speedup is linear, so that the whole process is very efficient. The next two stages can be viewed as a bottom-up sweep followed by a top-down sweep of a complete binary

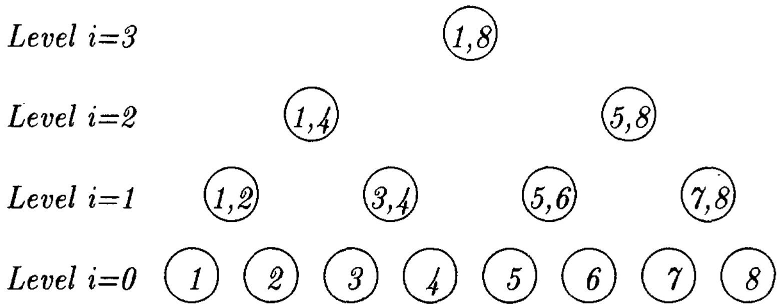

FIG. 1. The active processors for $p = 2^3$.

tree with p leaves; see Figure 1. In level i, $i = 1, \ldots, k - 1$, of the tree, processors

$$((j - 1)2^i + 1, j2^i), \qquad j = 1, \ldots, p/2^i, \tag{35}$$

are active in step $s = i$ of the bottom-up sweep, and in step $s = i - l$ of the top-down sweep. Each pair of processors in (35) exchanges constant information and performs a computation which can be carried out in constant time. Hence, the overall time complexity of these stages is of order $\log p$, and the communication overhead on any parallel architecture that allows an efficient embedding of a treelike network is low. As an example, Bar-On and Munk present an implementation of a related problem on the hypercube [6].

We may further distinguish between the extreme processors and the other ones. Processor 1 starts immediately with the QR transformation of its enlarged block as in (9). At the end of this computation, it delivers $(\hat{x}_n, \hat{y}_n)$ as a by-product. Later, it remains active in the bottom-up and top-down stages, and becomes inactive during the last parallel step, i.e. the application of the QR transformation. Processor p is inactive in the divide and conquer step, and becomes active only later in applying the QR transformation to its block. Hence, a single processor can simulate the computation of these two processors, thus providing us with an improved speedup. Throughout the rest of the paper, we will disregard this kind of local improvements for the sake of a simpler presentation. In what follows we denote the computed x, y pairs by $(\hat{x}_{in}, \hat{y}_{in})$, $i = 1, \ldots, p - 1$.

4.1. DIAGONALIZATION

Let us denote for simplicity the block of a given processor by

$$T = \begin{pmatrix} b_0 & a_1 & b_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & b_{n-1} & a_n & b_n \end{pmatrix}. \tag{36}$$

Then each processor applies a sequence of Givens rotations, i.e.

$$\tilde{T} = Q^t T = \left(G^t_{2(n-1)} \cdots G^t_2\right) T = \begin{pmatrix} e & u & & v & f \\ & & * & & \\ g & & & w & h \end{pmatrix}, \tag{37}$$

to its block, to obtain the above transformed two extreme rows. Note that we are not interested in the elements in the rows in between. In what follows, we present an exact implementation. Let

$$W_1 = \begin{pmatrix} b_0 & a_1 & b_1 & & \\ & b_1 & a_2 & b_2 \end{pmatrix}; \tag{38}$$

then we start by eliminating b_1 in the second row, i.e.

$$W_2 = G^t_2 W_1 = \begin{pmatrix} e_2 & u_2 & v_2 & f_2 \\ g_2 & & w_2 & h_2 \end{pmatrix}. \tag{39}$$

For $i = 2, \ldots, n-1$ let

$$W_{i,0} = \begin{pmatrix} e_i & u_i & v_i & f_i & \\ g_i & & w_i & h_i & \\ & & b_i & a_{i+1} & b_{i+1} \end{pmatrix}; \tag{40}$$

then we apply the following two Givens rotations:

1. A Givens rotation to eliminate b_i,

$$W_{i,1} = G^t_{2i-1} W_{i,0} = \begin{pmatrix} e_i & u_i & v_i & f_i & \\ * & & * & * & * \\ g_{i+1} & & & w_{i+1} & h_{i+1} \end{pmatrix}. \tag{41}$$

2. A Givens rotation to eliminate v_i,

$$W_{i,2} = G^t_{2i} W_{i,1} = \begin{pmatrix} e_{i+1} & u_{i+1} & & v_{i+1} & f_{i+1} \\ * & * & * & * & * \\ g_{i+1} & & & w_{i+1} & h_{i+1} \end{pmatrix}. \tag{42}$$

We denote the resulting two extreme rows by

$$W_{i+1} = \begin{pmatrix} e_{i+1} & u_{i+1} & v_{i+1} & f_{i+1} \\ g_{i+1} & & w_{i+1} & h_{i+1} \end{pmatrix}, \tag{43}$$

and proceed as above.

We denote the resulting two extreme rows of each block by

$$T^0_i = \begin{pmatrix} e^0_i & u^0_i & v^0_i & f^0_i \\ g^0_i & & w^0_i & h^0_i \end{pmatrix}, \qquad i = 1, \dots, p-1. \tag{44}$$

Note that processor 1 does not perform rotation 2 above, that $(w^0_1, h^0_1) = (\hat{x}_n, \hat{y}_n)$, and that the other elements in T^0_1 are null. Moreover, processor 1 completes the QR transformation of its block at the same time. We further note that the above process requires only $O(1)$ additional space, as we are interested in the two extreme rows only.

4.2. *A Bottom-Up Sweep*

For $s = 1, \dots, k-1$, let

$$T^s_{i,0} = \begin{pmatrix} T^{s-1}_{2i-1} \\ T^{s-1}_{2i} \end{pmatrix} = \begin{pmatrix} e^{s-1}_{2i-1} & u^{s-1}_{2i-1} & v^{s-1}_{2i-1} & f^{s-1}_{2i-1} & & \\ g^{s-1}_{2i-1} & & w^{s-1}_{2i-1} & h^{s-1}_{2i-1} & & \\ & & e^{s-1}_{2i} & u^{s-1}_{2i} & v^{s-1}_{2i} & f^{s-1}_{2i} \\ & & g^{s-1}_{2i} & & w^{s-1}_{2i} & h^{s-1}_{2i} \end{pmatrix} \tag{45}$$

for $i = 1, \dots, p/2^s$. Then apply the following sequence of Givens rotations:

1. A Givens rotation to eliminate h_{2i-1}^{s-1},

$$T_{i,1}^s = \begin{pmatrix} e_{2i-1}^{s-1} & u_{2i-1}^{s-1} & v_{2i-1}^{s-1} & f_{2i-1}^{s-1} & & \\ * & & * & & * & * \\ * & & * & * & * & * \\ & & g_{2i}^{s-1} & & w_{2i}^{s-1} & h_{2i}^{s-1} \end{pmatrix}. \tag{46}$$

2. A Givens rotation to eliminate g_{2i}^{s-1},

$$T_{i,2}^s = \begin{pmatrix} e_{2i-1}^{s-1} & u_{2i-1}^{s-1} & v_{2i-1}^{s-1} & f_{2i-1}^{s-1} & & \\ * & & * & & * & * \\ * & & * & * & * & * \\ g_i^s & & & & w_i^s & h_i^s \end{pmatrix}. \tag{47}$$

3. A Givens rotation to eliminate f_{2i-1}^{s-1},

$$T_{i,3}^s = \begin{pmatrix} * & * & \hat{v}_{2i-1}^{s-1} & & * & * \\ * & & * & & * & * \\ * & * & * & * & * & * \\ g_i^s & & & & w_i^s & h_i^s \end{pmatrix}. \tag{48}$$

4. A Givens rotations to eliminate $\hat{v}_{2i-1}^{s-1}$,

$$T_{i,4}^s = \begin{pmatrix} e_i^s & u_i^s & & & v_i^s & f_i^s \\ * & * & * & & * & * \\ * & * & * & * & * & * \\ g_i^s & & & & w_i^s & h_i^s \end{pmatrix}. \tag{49}$$

We finally denote the two extreme rows of $T_{i,4}^s$ by

$$T_i^s = \begin{pmatrix} e_i^s & u_i^s & v_i^s & f_i^s \\ g_i^s & & w_i^s & h_i^s \end{pmatrix}. \tag{50}$$

Note that for $i = 1$ we do not apply rotations 3 and 4 above, since the elements in the first row and column of $T_{1,0}^s$ are all null.

LEMMA 4.1. *By the end of step s, above, $s = 1, \dots, k-1$, we can choose*

$$\left(\hat{x}_{2^s n} = w_1^s,\ \hat{y}_{2^s n} = h_1^s\right). \tag{51}$$

Proof. Let $B_1^s = A_{[1:2^s n]}^{[1:2^s n+1]}$, and consider the rotations applied to the rows of B_1^s in the diagonalization stage and in the first s steps of the bottom-up sweep. Denote by C_1^s the corresponding transformed matrix. Then we observe that the last row of C_1^s is just the last row of T_1^s, which is similar to the last row of C_{2^s} in (13). The rest now follows from Corollary 3.2. ■

We will denote hereafter the last row of T_1^s by

$$R_1^s = \left(x_1^s, y_1^s\right) = \left(\hat{x}_{2^s n}, \hat{y}_{2^s n}\right). \tag{52}$$

4.3. *A Top-Down Sweep*

For $s = k-2, \dots, 0$ let

$$R_{i,0}^s = \begin{pmatrix} R_j^{s+1} \\ T_i^s \end{pmatrix} = \begin{pmatrix} x_j^{s+1} & y_j^{s+1} & & \\ e_i^s & u_i^s & v_i^s & f_i^s \\ g_i^s & & w_i^s & h_i^s \end{pmatrix}, \qquad i = 2j+1, \tag{53}$$

for $j = 1, \dots, p/2^{s+1} - 1$. Then we apply the following two Givens rotations:

1. A Givens rotation to eliminate y_j^{s+1},

$$R_{i,1}^s = \begin{pmatrix} * & & * & * \\ * & * & * & * \\ g_i^s & & w_i^s & h_i^s \end{pmatrix}. \tag{54}$$

2. A Givens rotation to eliminate g_i^s,

$$R_{i,2}^s = \begin{pmatrix} * & & * & * \\ * & * & * & * \\ & & x_i^s & y_i^s \end{pmatrix}. \tag{55}$$

Finally, we denote the last row of $R_{i,2}^s$ by $R_i^s = (x_i^s, y_i^s)$, and let $R_i^s = R_{i/2}^{s+1}$ for even i.

COROLLARY 4.2. *By the end of step s above, $s = k - 2, \dots, 0$, we can choose*

$$\left(\hat{x}_{i2^s n} = x_i^s,\ \hat{y}_{i2^s n} = y_i^s\right), \qquad i = 2j + 1, \tag{56}$$

for $j = 0, \dots, p/2^{s+1} - 1$.

Proof. The proof follows the same lines as the proof of Lemma 4.1. By the end of the bottom-up sweep, the formula (56) holds for $s = k - 1$. We will now prove by induction that being true for $s + 1$, it is also true for s. Let $B_i^s = A_{[(i-1)2^s n+1:i2^s n]}$, and consider the rotations applied to B_i^s in the diagonalization stage and in the first s steps of the bottom-up sweep. Denote by C_i^s the transformed matrix. Then the extreme two rows of C_i^s are just the extreme two rows of T_i^s in (53), and by induction, the formula (56) holds for R_j^{s+1}. Hence, the next two rotations in the top-down sweep transform the last row of C_i^s to a form similar to that of the last row of C_{i2^s} in (13). The rest now follows from Corollary 3.2 as before. ■

4.4. *Complexity Analysis*

We carry out a complexity analysis of our method, assuming a model of computation as described in the introduction to this paper. We say that two processors are adjacent if they are directly connected to each other, and we assume that the connection is bidirectional. We then assume that each step of the bottom-up sweep as well as of the top-down sweep can be implemented by exchanging $O(1)$ data between adjacent processors. Such an assumption amount in practice to having an interconnection network that allows embedding of a tree structure; see for example Bar-On and Munk [6] for an implementation of a similar problem on the hypercube. We then conclude that the computational cost of each iteration is given by $O(N/p + \log p) = O(N/p)$, since $\log p \ll N/p$. We further note that in practice, the running time behaves like

$$T_p(N) = c_1 n + c_2(p) \log_2 p, \qquad n = N/p, \tag{57}$$

where c_1 is a constant, and $c_2(p)$ depends on the communication network and the number of processors. A reasonable criterion for choosing the number of processors is then

$$c_1 n \geqslant 2c_2(p) \log_2 p, \qquad \log_2 p \leqslant n/c(p), \tag{58}$$

and practical consideration should dictate the value of $c(p)$. For example,

$$c(p) \leqslant 4\log_2(p) \Rightarrow p \leqslant 2^{\sqrt{n}/2}; \tag{59}$$

then taking $n = 4096$, we may solve a problem of order $N = 2^{44}$ using $p = 2^{32}$ processors in seconds.

5. STABILITY ANALYSIS

We will consider in this section the rounding error properties of the parallel QR algorithm presented in this paper. We assume that the matrix elements are given in finite precision, their order of magnitude does not vary widely, and they are normalized so that $\|A\|_2 \sim 1$. Balancing (see Parlett and Reinsch [14]) or the implicit QR method (see Francis [8, 9] and Dubrulle, Martin, and Wilkinson [7]) may be more appropriate for some special cases, but their parallel implementation is beyond the scope of the current paper. We further assume that the off diagonal elements are in absolute value above some threshold related to the accuracy of the computation. Otherwise the matrix is split into separate, independent blocks which can be dealt with in parallel. The criterion for such a threshold is not related to the parallel implementation of the algorithm and thus will not be considered here. Let A be an $N \times N$ tridiagonal symmetric matrix whose elements are represented in finite precision, say $\theta_1 = 2^{-t}$, and let the calculations be performed in a slightly higher precision than $\theta_2 \leqslant \theta_1$. Consider a single QR transformation; then

$$\tilde{A} = \mathrm{fl}(Q^tAQ) = Q^tAQ + E, \qquad \|E\|_2 \leqslant \theta_1, \tag{60}$$

where $\theta_2 \leqslant \theta_1/(2N)$; see Corollary A.1. The elements of $\tilde{A}$ are actually truncated to θ_1 precision, and the overall accuracy of the algorithm in the general case seems to be reasonable. However, in many cases the rounding errors do not accumulate, as can be seen from the discussion at the end of Section A.1, so that a much lower precision can be used in the computation.

5.1. *A Posteriori Error Bounds*

Let A denote an $N \times N$ unreduced symmetric tridiagonal matrix, and let $\tilde{A}$ be the transformed matrix as in Theorem 3.5. We will assume in what follows that computation is performed on <u>all</u> the elements of the matrix. Hence, the computed transformed matrix $\bar{A}$ is upper Hessenberg, but its lower subdiagonal and main diagonal are the same; see the related discussion in Section A.1.

We begin with a discussion of the case of two processors; the general one follows by induction. Let $\hat{T}_i$, $\bar{T}_i$, $i = 1, 2$, denote the corresponding blocks of the modified transformation, and let P, R be their respective QR transformations, i.e.

$$\begin{aligned} \bar{T}_1 &= \mathrm{fl}\left(P^t\hat{T}_1P\right) = P^t\hat{T}_1P + E_1, \qquad \|E_1\|_F \leqslant (n+1)\theta_2, \\ \bar{T}_2 &= \mathrm{fl}\left(R^t\hat{T}_2R\right) = R^t\hat{T}_2R + E_2, \qquad \|E_2\|_F \leqslant (n+1)\theta_2. \end{aligned} \tag{61}$$

Let F_i, $i = 2, \dots, n+1$, be the sequence of Givens rotations applied to $\hat{T}_1$, and let $P_{i,j} = F_i \cdots F_j$, denote any consecutive subsequence of these rotations. We then divide the transformations applied to $\hat{T}_1$ as follows:

$$\begin{aligned} &\bar{T}_{1,1} = \mathrm{fl}\left(P^t_{2,n}\hat{T}_1P_{2,(n-1)}\right) = P^t_{2,n}\hat{T}_1P_{2,(n-1)} + E^1_1, \\ &\|E^1_1\|_F \leqslant (n-1)\theta_2, \\ &\bar{T}_{1,2} = \mathrm{fl}\left(F^t_{(n+1)}\bar{T}_{1,1}F_nF_{(n+1)}\right) = F^t_{(n+1)}\bar{T}_{1,1}F_nF_{(n+1)} \\ &\qquad +E^2_1, \qquad \|E^2_1\|_F \leqslant 2\theta_2. \end{aligned} \tag{62}$$

Similarly, let H_i, $i = 1, \dots, n$, be the sequence of Givens rotations applied to $\hat{T}_2$, and let $R_{i,j} = H_i \cdots H_j$ denote any consecutive subsequence of these rotations. We then divide the transformations applied to $\hat{T}_2$ as follows:

$$\begin{aligned} \bar{T}_{2,1} &= \mathrm{fl}\left(H^t_2H^t_1\hat{T}_2H_1\right) = H^t_2H^t_1\hat{T}_2H_1 + E^1_2, \qquad \|E^1_2\|_F \leqslant 2\theta_2, \\ \bar{T}_{2,2} &= \mathrm{fl}\left(R^t_{3,n}\bar{T}_{2,1}R_{2,n}\right) = R^t_{3,n}\bar{T}_{2,1}R_{2,n} + E^2_2, \qquad \|E^2_2\|_F \leqslant (n-1)\theta_2. \end{aligned} \tag{63}$$

THEOREM 5.1. *Consider the last three rows of* $\bar{T}_{1,1}$, *and the first three rows of* $\hat{T}_2$ *as in* (19). *Let*

$$(\hat{x}_n, \hat{y}_n) = \pm(x_n(1+\varepsilon), y_n + \eta), \qquad |\varepsilon, \eta| \leqslant \theta. \tag{64}$$

Then there exists an orthogonal transformation Q *that simulates these two parallel transformations and satisfies*

$$\tilde{A}' - Q^tAQ = E, \qquad \|E\|_2 = O(N\theta_2 + \theta), \tag{65}$$

where $\tilde{A}'$ *is an equivalent transformation to* $\tilde{A}$.

Proof. Let $\tilde{T}_{1,1}$ denote the same matrix as $\overline{T}_{1,1}$, but with (x_n, y_n) replaced by

$$(\tilde{x}_n, \tilde{y}_n) \equiv (x_n(1+\varepsilon), y_n + \eta). \tag{66}$$

Let $\tilde{F}_{(n+1)}$ be the Givens rotations that should now be applied to $\tilde{T}_{1,1}$ instead of $F_{(n+1)}$, and let G be the corresponding rotation that simulates the action of $\tilde{F}_{(n+1)}$, H_1 as in Lemma 3.4, i.e. $G = \tilde{F}_{(n+1)}$, or

$$G^t = \begin{pmatrix} \tilde{c}_{(n+1)} & \tilde{s}_{(n+1)} \\ s_1 & c_1 \end{pmatrix}, \qquad \tilde{F}^t_{(n+1)} = \begin{pmatrix} \tilde{c}_{(n+1)} & \tilde{s}_{(n+1)} \\ -\tilde{s}_{(n+1)} & \tilde{c}_{(n+1)} \end{pmatrix},$$

$$H_1^t = \begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix}. \tag{67}$$

We will prove in what follows that $Q = P_{2,(n-1)} G R_{2,n}$. Let $B = P^t_{2,n} A P_{2,(n-1)}$, and let $\overline{B}$ be the matrix B modified so that

$$\overline{B}^{[1:(n+1)]}_{[1:n]} = \left(\tilde{T}_{1,1}\right)^{[1:(n+1)]}_{[1:n]}. \tag{68}$$

Then

$$\overline{B} = B + X_1 + X_2, \qquad \|X_1\|_F \leqslant \|E^1_1\|_F \leqslant (n-1)\theta_2, \quad \|X_2\|_F \leqslant \sqrt{2}\,\theta. \tag{69}$$

We next consider the matrix $C = H^t_2 G^t \overline{B} F_n G$, and denote by $\overline{C}$ the matrix C modified so that

$$\overline{C}^{[(n-1):n]}_{[(n-1):n]} \equiv \left(\overline{T}_{1,2}\right)^{[(n-1):n]}_{[(n-1):n]}, \tag{70}$$

$$\overline{C}^{[n:(n+3)]}_{[(n+1):(n+2)]} \equiv \left(\overline{T}_{2,1}\right)^{[1:4]}_{[2:3]}. \tag{71}$$

Here, by $\equiv$ we mean an equivalence relation, that is, we allow the subdiagonal elements in positions $(n, n-1)$, $(n+1, n)$ to change sign. We then conclude from Theorem 3.5, and from Lemma A.2, that we can set

$$\overline{C} = C + X_3 + X_4 + X_5, \tag{72}$$

where

$$\|X_3\|_F \leqslant \|E_1^2\|_F \leqslant 2\theta_2, \qquad \|X_4\|_F \leqslant \sqrt{26}\,\theta, \qquad \|X_5\|_F \leqslant \|E_2^1\|_F \leqslant 2\theta_2. \tag{73}$$

Finally, let $D = R_{3,n}^t \bar{C} R_{2,n}$, and denote by $\bar{D}$ the matrix D modified so that

$$\bar{D}^{[(n+1):N]}_{[(n+1):N]} = \left(\bar{T}_{2,2}\right)^{[2:(n+1)]}_{[2:(n+1)]}. \tag{74}$$

Then

$$\bar{D} = D + X_6, \qquad \|X_6\|_F \leqslant (n-1)\theta_2. \tag{75}$$

Collecting the different terms above, we conclude that

$$\bar{D} = Q^t A Q + X, \qquad \|X\|_F \leqslant (N+2)\theta_2 + 8\theta. \tag{76}$$

Finally, $\bar{D}$ is upper Hessenberg, with the same lower subdiagonal and main diagonal as in $\tilde{A}$. The conclusion of the theorem now follows. ■

COROLLARY 5.2. *Consider the case of p processors, and let*

$$(\hat{x}_{in}, \hat{y}_{in}) = \pm\left(x_{in}(1+\varepsilon_i), y_{in} + \eta_i\right), \qquad |\varepsilon_i, \eta_i| \leqslant \phi_i, \tag{77}$$

for $i = 1, \dots, p-1$, and let $\sum_{i=1}^{p} \phi_i \leqslant \phi$. Then there exists an orthogonal transformation Q that simulates the corresponding p parallel transformations and satisfies

$$\tilde{A}' - Q^t A Q = E, \qquad \|E\|_2 = O(N\theta_2 + \phi), \tag{78}$$

where $\tilde{A}'$ is an equivalent transformation to $\tilde{A}$.

However, as observed before for the sequential QR transformation, the term $N\theta_2$ is most pessimistic, and the second term, which can be easily computed in the course of the algorithm, may give a better estimate of the accuracy of the algorithm.

5.2. *A Priori Error Analysis*

The analysis in the previous section shows that the stability of the algorithm is closely related to what we shall term the forward stability of the shifted matrix. The pair (x_n, y_n) corresponds to the exact pair of one small perturbation of the matrix, and the pair $(\hat{x}_n, \hat{y}_n)$ to another one. The shifted matrix is forward stable if these pairs do not differ by much, and in this case the parallel algorithm is stable as a result of Corollary 5.2. We now proceed to elaborate on this point.

We consider the errors made in computing the corresponding pairs

$$(x_{in}, y_{in}), \quad (\hat{x}_{in}, \hat{y}_{in}), \qquad i = 1, \dots, p-1, \tag{79}$$

of Theorem 3.5. We consider, for simplicity, any given such pairs. Referring to Proposition A.4, we conclude that they are derived from the same matrix, say $\overline{B} \in \mathscr{M}((n_B + 1) \times (n_B + 2))$, where $n_B \leqslant n + 2\log(p/2)$. The computation of the two different pairs then proceeds as follows:

$$\overline{C}_i = \mathrm{fl}(P_i^t \overline{B}) = P_i^t(\overline{B} + \overline{E}_i) = \begin{pmatrix} X_i & v_i & w_i \\ 0 & x_i & y_i \end{pmatrix},$$

$$X \in \mathscr{M}(n_B), \quad v, w \in R^{n_B}, \quad i = 1, 2, \tag{80}$$

where we may assume that $\|\overline{E}_i\|_2 = O(n\theta_2)$ as in Proposition A.4. Let B and C_i, E_i, $i = 1, 2$, denote the respective square submatrices corresponding to the first $n_B + 1$ columns, and let us assume that B is nonsingular; then

$$C_i = P_i^t(B + E_i)(B^{-1}B) = P_i^t(I + E_i B^{-1})B = Q_i^t B, \qquad i = 1, 2, \tag{81}$$

where Q_i is not necessarily orthogonal. Let $B = QR$ be the unique QR factorization of B; then

$$Q_i^t = Z_i^t Q^t, \qquad Z_i^t = \begin{pmatrix} R_i & z_i \\ 0 & \alpha_i \end{pmatrix}, \quad Q^t B = \begin{pmatrix} \tilde{R} & z \\ 0 & x \end{pmatrix}. \tag{82}$$

Hence, $x_i = \alpha_i x$, and α_i^{-2} can be obtained from

$$Z_i^{-t} Z_i^{-1} = Q^t Q_i^{-t} Q_i^{-1} Q = Q^t\left[(I + B^{-t}E_i^t)(I + E_i B^{-1})\right]^{-1} Q. \tag{83}$$

Let $\|B^{-1}\|_2 \leqslant \beta$; then

$$Z_i^{-t} Z_i^{-1} = Q^t(I + F_i)^{-1} Q, \qquad \|F_i\|_2 \leqslant \varepsilon^2 + 2\varepsilon \tag{84}$$

$$= I + G_i, \qquad \|G_i\|_2 \leqslant \frac{\varepsilon^2 + 2\varepsilon}{1 - \epsilon^2 - 2\varepsilon}, \tag{85}$$

where $\varepsilon = O(\beta n \theta_2)$. Hence,

$$\alpha_i^{-2} = (1 + \eta_i)^{-2}, \qquad |\eta_i| \leqslant 2\varepsilon, \tag{86}$$

and therefore,

$$\frac{\alpha_1}{\alpha_2} = \frac{1 + \eta_1}{1 + \eta_2} = 1 + \eta, \qquad |\eta| \leqslant 5\varepsilon. \tag{87}$$

We further conclude from (81) that

$$P_i^t = Q_i^t\left(I + E_i B^{-1}\right)^{-1}, \tag{88}$$

and ignoring terms of $O(\theta_2^2)$, we have

$$(x_1, y_1) = \pm(x_2(1 + \eta), y_2 + \eta), \qquad |\eta| = O(n\beta\theta_2). \tag{89}$$

Let

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_N| \tag{90}$$

denote the sequence of eigenvalues of the shifted matrix, and let β_i, $i = 1, \dots, p - 1$, denote the norm of the inverse of the inth leading submatrix. Then, with each iteration of the QR algorithm,

$$\beta_i \to \left(|\lambda_{in}|\right)^{-1} \leqslant \left(|\lambda_{N-n}|\right)^{-1}; \tag{91}$$

see Watkins [21]. Hence, β should be small, unless the sought-for eigenvalue belongs to a very large cluster of at least n eigenvalues, and for $\beta \leqslant \theta_1^{-1}$, using double precision, we bound the rounding errors of the computed x, y pairs by $O(n\theta_1)$. In many cases this will suffice, as $n \ll N$ and the errors usually do not accumulate.

We finally note that in the rare cases where instability does occur, we may simply restart the iteration with a slightly modified shift, or change slightly the redistribute of rows between the processors, or deflate the matrix by the methods described in Bar-On [5] and in Parlett and Le [13].

5.3. *Numerical Examples*

We have made several numerical tests of the parallel algorithm by simulation on a sequential computer. We have tested matrices up to the order of $N = 2^{19}$ using single and double precision, on a variety of randomly created matrices, with up to 512 processors, and our results have agreed with those of the sequential method. We give here two examples related to the tridiagonal matrix $A = (-1, 2, -1)$, whose eigenvalues are known exactly. The first is a matrix of order 2^{19} in single precision, where we have looked for the eigenvalues near 0.3. The nearest 10 eigenvalues are depicted in the first column of Table 1. All eigenvalues were found with an error of order 10^{-7}, and the error in the computed x, y pairs was of a similar size. The second is a matrix of order 2^{18} in double precision, where we have looked for the eigenvalues near 0.0, which are rather small; see the second column of Table 1. All eigenvalues were found with an error of order 10^{-15}, although the error in the computed x, y pairs was of order 10^{-13}.

We further include an example of Wilkinson test matrix of order $n = 128$, using $p = 4$ processors. The first column in Table 2 depicts the largest eigenvalues computed using Matlab, and the second column depicts the corresponding eigenvalues computed with the sequential QR algorithm and with the parallel QR algorithm using $p = 4$ processors. The results for the sequential and the parallel algorithm were the same, and they agree to 14 digits with the results of Matlab, as the precision warrants.

6. CONCLUSION

In this paper we have tried a first approach towards a fast and stable parallel implementation of the QR method for tridiagonal systems. We have developed an efficient algorithm whose numerical stability is similar to that

TABLE 1

EXACT EIGENVAUES FOR $A = (-1, 2, -1)$

$n = 2^{19}$, $r = 0.3$	$n = 2^{18}$, $r = 0.0$
0.29999747229076	0.00000000014362
0.30000378536701	0.00000000057448
0.29999115927556	0.00000000129258
0.30001009850429	0.00000000229793
0.29998484632139	0.00000000359051
0.30001641170261	0.00000000517034
0.29997853342827	0.00000000703741
0.30002272496197	0.00000000919172
0.29997222059618	0.00000001163326
0.30002903828237	0.00000001436206

TABLE 2

WILKINSON TEST MATRIX W_{129}^{+}

Matlab	$p = 1,4$
6.100395200266527E + 01	6.100395200266531E + 01
6.100395200266536E + 01	6.100395200266536E + 01
6.203894111930644E + 01	6.203894111930640E + 01
6.203894111930657E + 01	6.203894111930642E + 01
6.321067864733298E + 01	6.321067864733303E + 01
6.321067864733305E + 01	6.321067864733303E + 01
6.474619418290330E + 01	6.474619418290333E + 01
6.474619418290337E + 01	6.474619418290335E + 01

achieved by the sequential QR method, and we believe that a similar approach can be used for systems of larger bandwidth. We note that the parallel QR algorithm can be used in conjunction with the parallel LR [3] and bisection [2] algorithms to improve convergence rates and execution time. Furthermore, the method is most useful when implemented with a divide and conquer approach for computing the eigenvalues of symmetric tridiagonal matrices; see Bar-On [4, 5].

APPENDIX

A.1. *Sequential Error Analysis*

Let A be an $N \times N$ unreduced symmetric tridiagonal matrix, normalized so that $\|A\|_2 \sim 1$, and consider a single QR transformation,

$$A_n = \mathrm{fl}\left(G_{(n+1)}^t A_{(n-1)} G_n\right) = G_{(n+1)}^t A_{(n-1)} G_n + E_n, \qquad n = 1, \dots, N, \tag{92}$$

where $A_0 = A$, and $G_1 = G_{N+1} = I$ is the identity matrix. Here, the analysis of rounding errors is nontrivial, since we compute the elements only on the lower and main diagonals. We then denote a "true" sequence by

$$B_n = \mathrm{fl}\left(G_{(n+1)}^t B_{(n-1)} G_n\right) = G_{(n+1)}^t B_{(n-1)} G_n + F_n, \qquad n = 1, \dots, N, \tag{93}$$

where $B_0 = A$. Here, by "true" we mean that we proceed by computing all the elements of the matrix as usual. However, we assume w.l.o.g. that any additional computation not actually performed in (92) is done in exact precision. We then employ the standard error analysis properties of Givens rotations; see Wilkinson [22, 23]. Let the matrix elements be given in θ_1 precision, and let the computations proceed in a slightly higher precision than θ_2. Then the matrix elements that contribute to the error in (93) are bounded by ~ 1, and we may assume that $\|F_n\|_F \leqslant \theta_2$, where $\|\cdot\|_F$ is the so-called Frobenius norm. Hence, an overall bound for the entire process is

$$\tilde{B} = \mathrm{fl}(Q^tAQ) = Q^tAQ + F, \qquad \|F\|_F \leqslant N\theta_2, \tag{94}$$

where $Q = G_2 \cdots G_N$. Now, $\tilde{B}$ is upper Hessenberg, and its lower and main diagonals are the same as in $\tilde{A}$. Hence,

$$\tilde{A} - Q^tAQ = E = E_l + E_d + E_l^t, \qquad \|E_l + E_d\|_F \leqslant \|F\|_F, \tag{95}$$

where E_l is strictly lower triangular, and E_d is diagonal.

COROLLARY A.1. *Consider the sequential QR transformation as in* (92). *Then the computed transformed matrix satisfies*

$$\tilde{A} = Q^tAQ + E, \qquad \|E\|_2 \leqslant (2N)\theta_2. \tag{96}$$

However, this bound is overpessimistic, as the following discussion will show. We first recall the structure of a Givens rotation. Let

$$G^tv = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} x \\ b \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \end{pmatrix} \tag{97}$$

be the Givens rotation that annihilates b. Then we choose c, s as follows:

$$t = \begin{cases} b/x, & |b| \leqslant |x|, \quad c = 1/d, \quad s = t/d, \\ x/b, & |x| < |b|, \quad c = t/d, \quad s = 1/d, \end{cases} \tag{98}$$

where $d = \sqrt{1 + t^2}$; see Golub and Van Loan [10]. A close observation of the QR transformation reveals that in case all the Givens rotations satisfy $|s| \leqslant 1 - q$ for some $q < 1$, then $\|E\|_2 = O(\theta_2/q)$. For example, for $|s| \leqslant 1 - \theta_1$, that is, $|x| \geqslant 2\theta_1^{1/2}|b|$ in (98), using double precision will suffice to achieve $\|E\|_2 = O(\theta_1)$ in (96) irrespective of the matrix order. Moreover, this assumption is quite reasonable, since with each iteration the b's tend to zero and the x's tend to the respective eigenvalues in decreasing order.

A.2. A Posteriori Error Analysis

We assume in what follows that $\|A\|_2 \leqslant 1$.

LEMMA A.2. *Consider the following sequence of Givens rotations*:

$$A_1 = G^t A = \begin{pmatrix} 1 & & \\ & c & s \\ & -s & c \end{pmatrix} \begin{pmatrix} * & * & \\ & x & y \\ & b & a \end{pmatrix} = \begin{pmatrix} * & * & \\ & \alpha' & \beta \\ & & * \end{pmatrix}, \qquad (99)$$

where $\alpha' = cx + sb$ *and* $\beta = cy + sa$. *Next, let*

$$A_2 = A_1 G' = \begin{pmatrix} * & * & \\ & \alpha' & \beta \\ & & * \end{pmatrix} \begin{pmatrix} c' & -s' & \\ s' & c' & \\ & & 1 \end{pmatrix} = \begin{pmatrix} * & * & \\ e & \alpha & \beta \\ & & * \end{pmatrix}, \qquad (100)$$

where $e = \alpha' s'$ *and* $\alpha = \alpha' c'$. *Finally, let*

$$A_3 = A_2 G = \begin{pmatrix} * & * & \\ e & \alpha & \beta \\ & & * \end{pmatrix} \begin{pmatrix} 1 & & \\ & c & -s \\ & s & c \end{pmatrix} = \begin{pmatrix} * & * & \\ e & d & * \\ & & * \end{pmatrix}, \qquad (101)$$

where $d = \alpha c + \beta s$. *We now consider the related sequence applied to* $\tilde{A}$ *which is the same as A but with*

$$(\hat{x}, \hat{y}) = (x(1+\varepsilon), y + \eta), \qquad |\varepsilon, \eta| \leqslant \theta. \qquad (102)$$

Then,

$$||\hat{e}| - |e|| \leqslant \theta, \qquad |\hat{d} - d| \leqslant 5\theta. \qquad (103)$$

Proof. Consider the first bound in (103); then

$$|\hat{\alpha}'| = |\alpha'| \frac{\sqrt{x^2(1+\varepsilon)^2 + b^2}}{\sqrt{x^2 + b^2}} = |\alpha'| \sqrt{1 + \frac{x^2}{x^2 + b^2}(\varepsilon^2 + 2\varepsilon)} \qquad (104)$$

$$= |\alpha'|(1 + \varepsilon_1), \qquad 0 \leqslant \varepsilon_1 \leqslant \varepsilon \text{ or } 0 \geqslant \varepsilon_1 \geqslant \varepsilon. \qquad (105)$$

We can therefore conclude that

$$||\hat{e}| - |e|| = |s'||\hat{\alpha}'| - |\alpha'|| \leqslant |\alpha'|\theta \leqslant \theta. \tag{106}$$

For the second bound in (103), note that

$$|\hat{d} - d| \leqslant |\alpha c - \hat{\alpha}\hat{c}| + |s^2 - \hat{s}^2|\,|a| + |scy - \hat{s}\hat{c}\hat{y}|. \tag{107}$$

For the first term, we observe from (98) that sign$(\alpha' c)$ = sign(x), so that

$$|\alpha c - \hat{\alpha}\hat{c}| = |c'|\,|\alpha' c - \hat{\alpha}'\hat{c}| \leqslant |x - \hat{x}| \leqslant \theta. \tag{108}$$

For the second term in (107), we obtain

$$|s^2 - \hat{s}^2| = |c^2 - \hat{c}^2| = c^2\left|1 - \left(\frac{1+\varepsilon}{1+\varepsilon_1}\right)^2\right| \leqslant |1 - (1+\varepsilon)^2| \leqslant 2\theta + \theta^2. \tag{109}$$

Finally, for the last term in (107), we get

$$|scy - \hat{s}\hat{c}\hat{y}| \leqslant \theta + |sc - \hat{s}\hat{c}|, \tag{110}$$

and therefore, assuming w.l.o.g. that $\varepsilon > 0$, we get

$$\begin{aligned} |sc - \hat{s}\hat{c}| &= \left|\frac{xb}{x^2+b^2} - \frac{xb(1+\varepsilon)}{[x(1+\varepsilon)]^2 + b^2}\right| \\ &\leqslant \frac{1}{2}\left|1 - (1+\varepsilon)\frac{x^2+b^2}{[x(1+\varepsilon)]^2+b^2}\right| \end{aligned} \tag{111}$$

Now,

$$1 - 2\varepsilon \leqslant \frac{x^2+b^2}{[x(1+\varepsilon)]^2+b^2} \leqslant 1, \tag{112}$$

and we conclude that

$$|sc - \hat{s}\hat{c}| \leqslant \tfrac{1}{2}|1 - (1 + \varepsilon)(1 - 2\varepsilon)| \leqslant \tfrac{1}{2}\theta + \theta^2. \tag{113}$$

Collecting the different terms, we conclude that

$$|\hat{d} - d| \leqslant \theta + (2\theta + \theta^2) + \theta + \left(\tfrac{1}{2}\theta + \theta^2\right) \leqslant 5\theta, \tag{114}$$

from which the second bound in (103) now follows. ■

A.3. *A Priori Error Analysis*

LEMMA A.3. *For $s = k - 1, \dots, 0$ the corresponding pairs*

$$(x_{in}, y_{in}), \quad (\hat{x}_{in}, \hat{y}_{in}), \qquad i = (2j + 1)2^s, \quad j = 1, \dots, p/2^{s+1} - 1, \tag{115}$$

are derived from the same set of $n_s + 1$ rows, where $n_s = 2s + n$, as follows:

1. *The row R_j^{s+1}.*
2. *The two extreme rows of*

$$T_l^r, \qquad r = 0, \dots, s - 1, \quad l = i/2^r - 1. \tag{116}$$

3. *The n rows of T_i.*

Proof. The proof is straightforward, and is left for the reader. ■

We will denote the matrix corresponding to the above ith pairs by B_i^s.

PROPOSITION A.4. *For $s = 0, \dots, k - 1$ let*

$$A_i^s = A_{[1:in]}^{[1:in+1]}, \qquad i = (2j + 1)2^s, \quad j = 0, \dots, p/2^{s+1} - 1. \tag{117}$$

Then B_i^s is obtained from A_i^s by a sequence of Givens rotations such that

$$B_i^s = \mathrm{fl}\big((Q_i^s)^t A_i^s\big) = (Q_i^s)^t A_i^s + E_i^s, \qquad \|E_i^s\|_2 = O(n\theta_2). \tag{118}$$

Proof. We give only an informal proof, leaving the details for the reader. The corresponding rows in Lemma A.3 can be seen to be the result of applying only part of the Givens rotations. This partial sequence, applied to A_i^s, is denoted by Q_i^s. Moreover, the corresponding rotations are applied in $O(\log\ p)$ steps, where in each step rotations are applied to independent (or almost independent) blocks of the matrix. In conclusion, the rounding errors due to the rotations in the diagonalization stage are of order $n\theta_2$, and those due to the bottom-up and top-down sweeps are of order $\log\ p$. Finally, $\log\ p \ll n$ for all practical purposes. ∎

REFERENCES

1 I. Bar-On, A new divide and conquer parallel algorithm for the Cholesky decomposition of band matrices, in 13th IMACS World Cong. on Comp. and Appl. Math., July 1991.

2 ——, A fast parallel bisection algorithm for symmetric band matrices, Tech. Report 726, Technion, Computer Science Department, 1992.

3 ——, Fast parallel *LR* and *QR* algorithms for symmetric band matrices, Tech. Report 749, Technion, Computer Science Department, 1992.

4 ——, Interlacing properties for tridiagonal symmetric matrices with applications to parallel computing, *SIAM J. on Matrix Analysis Applications,* (1993). Submitted.

5 ——, A new divide and conquer parallel algorithm for computing the eigenvalues of a symmetric tridiagonal matrix, Journal of Computational and Applied Mathematics, (1994). Submitted.

6 I. Bar-On and O. Munk, *A New DD^{t}BC Parallel Factorization of Band Matrices,* in Joint International Conference on Vector and Parallel Processing, Lyon, France, September 1–4, 1992, no. 634 in Lecture Notes in Computer Science, Springer-Verlag, 1992, pp. 251–262.

7 A. Dubrulle, R. Martin, and J. Wilkinson, The implicit *QL* algorithm, in *Handbook for Automatic Computation,* Vol. II (J. Wilkinson and C. Reinsch, Eds.), Springer-Verlag, 1971, pp. 241–248.

8 J. Francis, The *QR* transformation, part I, *Comput. J.* 4:265–271 (1961).

9 ——, The *QR* transformation, part II, *Comput. J.* 4:332–345 (1962).

10 G. H. Golub and C. F. V. Loan, *Matrix Computations,* Johns Hopkins U.P., 1989.

11 C. Ipsen and E. Jessup, Solving the symmetric tridiagonal eigenvalue problem on the hypercube, *J. Comput. System Sci.,* 1990, pp. 203–229.

12 S. Lo, B. Philippe, and A. Sameh, A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problem, *SIAM J. Sci. Statist. Comput.,* 8:s155–s165 (1987).

13 B. Parlett and J. Le, Forward instability of tridiagonal *QR*, *SIAM J. Matrix Anal. Appl.,* 1993, pp. 279–316.

14 B. Parlett and C. Reinsch, Balancing a matrix for calculation of eigenvalues and eigenvectors, in *Handbook for Automatic Computation,* Vol. II (J. Wilkinson and C. Reinsch, Eds.), Springer-Verlag, 1971, pp. 315–326.

15 B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall International, 1980.

16 G. Peters and J. Wilkinson, The calculation of specified eigenvectors by inverse iteration, in *Handbook for Automatic Computation*, Vol. II (J. Wilkinson and C. Reinsch, Eds.), Springer-Verlag, 1971, pp. 418–439.

17 C. Reinsch, A stable rational *QR* algorithm for the computation of the eigenvalues of an hermitian, tridiagonal matrix, *Math. Comp.* 25:591–597 (1971).

18 H. Rutishauser, Solution of eigenvalue problems with the *LR* transformation, *Nat Bur. Standards AMS* 49:47–81 (1958).

19 A. Sameh and D. Kuck, A parallel *QR* algorithm for symmetric tridiagonal matrices, *IEEE Trans. Comput.* C-26:147–152 (1977).

20 B. Smith, J. Boyle, et al., *EISPACK Guide*, Springer-Verlag, 1988.

21 D. Watkins, *Fundamentals of Matrix Computations*, Wiley, 1991.

22 J. H. Wilkinson, *Rounding Errors in Algebraic Process*, Prentice-Hall International, 1963.

23 ——, *The Algebraic Eigenvalue Problem*, Oxford Science Publications, 1988.

Received 24 February 1993; final manuscript accepted 9 December 1993