SIAM J. COMPUT.
Vol. 27, No. 4, pp. 1099–1115, August 1998

010

COMPUTING MATRIX EIGENVALUES AND POLYNOMIAL ZEROS WHERE THE OUTPUT IS REAL*

DARIO BINI† AND VICTOR Y. PAN‡

Abstract. Surprisingly simple corollaries from the Courant–Fischer minimax characterization theorem enable us to devise a very effective algorithm for the evaluation of a set S interleaving the set E of the eigenvalues of an $n \times n$ real symmetric tridiagonal (rst) matrix T_n (as well as a point that splits E into two subsets of comparable cardinalities). Furthermore, we extend this algorithm so as to approximate all the n eigenvalues of T_n at *nearly optimal* sequential and parallel cost, that is, at the cost of staying within polylogarithmic factors from the straightforward lower bounds. The resulting improvement of the known processor bound in NC algorithms for the rst-eigenproblem is roughly by factor n. Our approach extends the previous works [M. Ben-Or and P. Tiwari, *J. Complexity*, 6(1990), pp. 417–442] and [M. Ben-Or et al., *SIAM J. Comput.*, 17(1988), pp. 1081–1092] for the approximation of the zeros of a polynomial having only real zeros, and our algorithm leads to an alternative and simplified derivation of the known record parallel and sequential complexity estimates for the latter problem.

Key words. symmetric tridiagonal eigenvalues, approximation algorithms, computational complexity, real polynomial zeros

AMS subject classifications. 68Q05, 68Q40, 68H05

PII. S0097539790182482

1. Introduction.

1.1. The problem. The problem of approximating the eigenvalues of an $n \times n$ Hermitian or real symmetric matrix A is one of the central problems of practical matrix computations [GL], [Par]. The first step of its solution in all the customary algorithms is the reduction of the input matrix A to the real symmetric tridiagonal (rst) form; this step can be effectively parallelized [Pan87], [BP94, Proposition 5.4, p. 325].

In the present paper, we consider the remaining eigenvalue problem for an $n \times n$ rst-matrix T_n. From technical and theoretical points of view, this problem is closely related to approximating the zeros of polynomials having only real zeros. In computing practice, such polynomials usually appear as the characteristic polynomials of Hermitian (or real symmetric) matrices, which also cover the class of orthogonal polynomials. Given the coefficients of an nth-degree polynomial $p(\lambda)$ having only real zeros, $\lambda_1, \ldots, \lambda_n$, we may compute the entries of an $n \times n$ rst-matrix T_n that has the characteristic polynomial $p(\lambda)$ and the eigenvalues $\lambda_1, \ldots, \lambda_n$. In Appendix A and also in [BP94, pp. 117–120], we achieve this by applying the extended Euclidean scheme to $p(\lambda)$ and $p'(\lambda)$, whereas, for a given T_n, we may compute the coefficients of $p(\lambda)$ by applying a simpler algorithm, which we present in section 5. In the practice of matrix computations, the reduction of the rst-eigenvalue problem to computing

* Received by the editors June 1, 1990; accepted for publication June 11, 1996; published electronically May 19, 1998. The results of this paper were presented at the Second Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1991.
http://www.siam.org/journals/sicomp/27-4/18248.html
† Dipartimento di Matematica, Università di Pisa, 56100 Pisa, Italy (bini@dm.unipi.it). This research was supported by NSF grant 8805782 and by the Italian M.P.I. 40% funds.
‡ Department of Mathematics and Computer Science, Lehman College, City University of New York, Bronx, NY 10468 (vpan@lcvax.lehman.cuny.edu). This research was supported by NSF grant 8805782 and by PSC-CUNY Awards 668541 and 669210.

the coefficients of the characteristic polynomial $p(\lambda)$ is avoided because of the numerical stability problems (in particular, the value $|p(0)| = |\det T_n|$ can be very large); this does not apply, however, to computing the *values* of $p(\lambda)$, which is a customary auxiliary step of the rst-eigenvalue computation [Par], [GL, pp. 437–440].

1.2. Our results (outline). Our main result is a new parallel NC algorithm for approximating the eigenvalues of an rst-matrix T_n given its entries. We assume the customary arithmetic and Boolean PRAM models of parallel computation (where in each parallel step each nonidle processor performs one arithmetic or, respectively, Boolean operation), and we deduce *nearly optimum* upper bounds on parallel time and the number of processors required by our algorithm. (By nearly optimum we mean upper bounds that are within polylogarithmic factors from the known, and in our case straightforward, lower bounds.) This is a dramatic improvement of the previous best parallel solution, based on the reduction of the problem to approximating the zeros of $p(\lambda)$ and on the solution of the latter problem by means of the algorithm of [BOT]. Our approach avoids using extended Euclidean computations involved in the algorithm of [BOT], and this gives us the edge over [BOT].

The sequential version of our algorithm, as well as its extension to the sequential and parallel approximation of the zeros of $p(\lambda)$ given the coefficients, supports the same complexity estimates as the algorithm of [BOT] does, and in the sequential case, these upper estimates are nearly optimal too. Our approach, however, may be considered conceptually simpler and easier to comprehend, as the reader may observe from the comparison of our techniques with ones of [BOT], made in the beginning of section 2.

1.3. Our complexity estimates. To estimate the sequential and parallel cost, we will write $O_A(t,p)$ and $O_B(t,p)$ for the algorithms that require $O(t)$ parallel steps using p arithmetic processors and $O(t)$ parallel steps using p bit-serial processors, respectively. The bounds $O_A(t,sp)$ and $O_B(t,sp)$ imply the bounds $O_A(st,p)$ and $O_B(st,p)$, respectively, for $s \geq 1$, according to Brent's scheduling principle of parallel computing [Br]. (In particular, $O_A(t,p)$ and $O_B(t,p)$ imply the sequential arithmetic cost bound $O_A(tp,1)$ and the sequential Boolean cost bound $O_B(tp,1)$, respectively.) Under this notation, our algorithm supports approximating the eigenvalues of T_n (whose entries have magnitudes at most 2^m), within the absolute error bound 2^{-h}, at the arithmetic and Boolean parallel cost bounded by $O_A(\log^2 n(\log^2 b + \log n)\log\log n, n/\log\log n)$ and $O_B(\log^2 n \log(nb)(\log^2 b + \log n)\log\log n \log\log(nb), n^2 b/\log\log n)$, respectively, and at the sequential cost bounded by $O_A(n\log^2 n(\log^2 b + \log n), 1)$ and $O_B(n^2 b \log^2 n \log(nb)(\log^2 b + \log n)\log\log(nb), 1)$, respectively, where $b = m + h$. For approximating all the n zeros of a polynomial $p(\lambda)$ having only real zeros, the same sequential cost bounds hold, but the parallel cost bounds change (in particular, to $O_A(\log^2 n(\log^2 b + \log n), (n/\log b)^2)$ under the arithmetic PRAM model) since we need to add the cost of performing the extended Euclidean algorithm that supports the transition from $p(\lambda)$ to T_n. In section 8, we comment on some ways to further minor improvements.

1.4. A parallel modification. Our major concern in this paper is about the computational complexity estimates. On the other hand, caring more about numerical stability of lower precision computations than about decreasing their asymptotic complexity, we have modified our main algorithm in [BP92] (cf. our Remark 5.1 in section 5 and comments in section 8). In the parallel NC algorithm of [BP92], we only need about $n/\log n$ times more processors but achieve substantially improved

numerical stability, which makes the algorithm competitive with the known alternative practical algorithms for the symmetric eigenvalue problem. In [B] and in [BG1], further progress in this direction has been obtained.

1.5. Organization of the paper. We will organize our paper as follows: in section 2, we deduce some properties of interlacing sets by using the Cauchy interlace theorem for the matrix eigenvalues. In section 3, for each eigenvalue of T_n, we compute either its approximation within a fixed error bound or an interval containing this eigenvalue but no other eigenvalues of T_n. In section 4, given an interval containing only one eigenvalue, we compute an approximation to this eigenvalue within the required precision by means of Newton's method and of the bisection of the exponents. In section 5, we describe an algorithm for the simultaneous computation of the values of the characteristic polynomial and its derivative at a set of points. In section 6, we summarize the construction of sections 2–5 and devise an algorithm for the approximation of all the eigenvalues. In section 7, we estimate the bit-complexity of the algorithm. In section 8, we briefly discuss some results that appeared more recently, after this paper had been submitted for publication—in particular, some results related to the complexity estimates and to practical performance of the algorithm of the present paper. In section A.1 of the appendix, we discuss the relations between the polynomial root-finding problem and the problem of the computation of the eigenvalues of a matrix, and in section A.2, the reduction of a Hermitian or real symmetric matrix to the tridiagonal form.

2. Interlacing sets and splitting points for the set of the eigenvalues. In the following, all logarithms are to the base 2.

Hereafter, T_n denotes an $n \times n$ rst-matrix having diagonal entries $a_1, \dots, a_n$, subdiagonal entries $b_1, \dots, b_{n-1}$, and a set Λ of n eigenvalues, $\Lambda = \{\lambda_1 \le \cdots \le \lambda_n\}$. Without loss of generality, suppose that $b_1 \dots b_{n-1} \neq 0$. (Indeed, if $b_j = 0$ for some j, the eigenvalue problem would be split into two eigenvalue problems of smaller dimensions.) We assume that $|a_i|$, $|b_i| \le 2^m$, m is a positive integer, and then, by virtue of Gerschgorin's theorem ([GL, p. 341]),

$$-3(2^m) \le \lambda_i \le 3(2^m). \tag{2.1}$$

We say that the set $R = \{r_0, \dots, r_k\}$ *interleaves* the set $Q = \{q_1, \dots, q_k\}$ and that R is an *interlacing set* for Q if $r_0 \le q_1 \le r_1 \le q_2 \le \cdots \le q_{k-1} \le r_{k-1} \le q_k \le r_k$, where, in particular, we will allow $r_0 = -\infty$ and/or $r_k = +\infty$, and then we will write $R = \{r_i, \dots, r_{k-j}\}$, where $i, j = 0, 1$. We say that s is a *splitting point of the level* (g, h) for the set Q if $q_g < s < q_h$.

In this section, we will prove some simple corollaries from Cauchy's interlace theorem [Par, p. 186] or from the Courant–Fischer minimax characterization [GL, p. 411]. Later on, they will lead us to simple algorithms for the evaluation of the interlacing sets and splitting points for the set of the eigenvalues of T_n. Using such sets and/or points will enable us to devise divide-and-conquer algorithms for approximating the eigenvalues of T_n, and we will specify such an algorithm in section 6.

It is instructive to compare these results with the techniques of [BOT] and [BFKT] dealing with a polynomial $p(\lambda)$ that has only real zeros and can be considered as the characteristic polynomial of T_n. Specifically, the algorithm of [BOT] relies on computing the Sturm and pseudoremainder sequences associated with $p(\lambda)$ and defining a set S of real points that interleaves the set of the zeros of $p(\lambda)$. Computing such a set S is the central (and also most innovative, most intricate, and most involved) part of

the algorithm of [BOT], and here we will introduce our main innovation too: we will replace the major techniques of [BOT] by some simple corollaries from Cauchy's interlace theorem or the Courant–Fischer minimax characterization theorem, and then, we will immediately arrive at a set interleaving the set of the eigenvalues Λ of T_n.

Technically, it may also be interesting that Cauchy's theorem or the Courant–Fischer minimax characterization may replace Sturm sequences in at least one more application. Namely, in [BFKT], the Sturm sequences have been used for computing a splitting point (rather than the interleaving set) for the set of the zeros of $p(\lambda)$; then again, some simple corollary from Cauchy's theorem or the Courant–Fischer minimax characterization will give us a simple algorithm for computing such a splitting point for the eigenvalues of T_n, and this computation is more effective than one of [BFKT]. We will give more comments later on, after the statement of our Corollary 2.1.

Hereafter, $\mathrm{Diag}(B_1, \dots, B_s)$ denotes the block diagonal matrix having diagonal blocks $B_1, \dots, B_s$.

We will rely on the following result, known as Cauchy's interlace theorem.

THEOREM 2.1. *If A_r denotes an $r \times r$ principal submatrix of an $n \times n$ real symmetric matrix A, then the eigenvalues $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ of A and the eigenvalues $\mu_1 \le \mu_2 \le \cdots \le \mu_r$ of A_r satisfy the following relations:*

$$\lambda_i \le \mu_i \le \lambda_{i+n-r}, \quad i = 1, \dots, r.$$

Proof. The latter relations hold for $r = n-1$ (see Corollary 8.1-4 to the Courant–Fischer theorem in [GL, p. 411]). In the general case, it is sufficient to apply these relations to a sequence $A = A_n, A_{n-1}, A_{n-2}, \dots, A_r$ of principal submatrices of A such that A_i is a principal submatrix of A_{i+1}. An alternative proof can be found on pp. 186–187 of [Par]. □

We are ready to prove the following basic result.

THEOREM 2.2. *The eigenvalues of T_n satisfy the following relations:*

(a) *If nk is a multiple of $2(k+1)$ and if $\{\mu_1 < \mu_2 < \cdots < \mu_{\frac{k}{k+1}n}\}$ is the set of all the eigenvalues of the $k \times k$ principal submatrices $\tilde{T}_i$ of T_n, $i = 1, \dots, \frac{n}{k+1}$, $\tilde{T}_i$ containing the (s,s) entries a_s of T_n for $s = (i-1)(k+1)+j$, $j = 1, \dots, k$, then*

$$\lambda_{\frac{nk}{2(k+1)}} \le \mu_{\frac{nk}{2(k+1)}} \le \lambda_{\frac{n(k+2)}{2(k+1)}}.$$

(b) *If $1 \le j \le n-2$ and if $\{\gamma_1 \le \gamma_2 \le \cdots \le \gamma_{n-1}\}$ is the set of all the eigenvalues of the two principal submatrices of T_n,*

$$T_j = \begin{pmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{j-1} \\ & & b_{j-1} & a_j \end{pmatrix}, \quad \hat{T}_{n-j-1} = \begin{pmatrix} a_{j+2} & b_{j+2} & & \\ b_{j+2} & a_{j+3} & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{pmatrix},$$

then

$$\lambda_i \le \gamma_i \le \lambda_{i+1}, \quad i = 1, \dots, n-1.$$

In other words, part (a) *defines a splitting point of the level $(\frac{nk}{2(k+1)}, \frac{n(k+2)}{2(k+1)})$ for the set of the eigenvalues of T_n, and part* (b) *defines an interlacing set for this set of the eigenvalues.*

Proof. Theorem 2.2 immediately follows from Theorem 2.1. In particular, to prove part (a), apply Theorem 2.1 to the $r \times r$ principal submatrix of T_n (where $r = n - \frac{n}{k+1}$) obtained by deleting the $(i(k+1))$th rows and columns of T_n for $i = 1, 2, \ldots, \frac{n}{k+1}$. This submatrix is the block diagonal matrix $\mathrm{Diag}(\tilde{T}_1, \ldots, \tilde{T}_{\frac{n}{n+1}})$. To prove part (b), apply Theorem 2.1 to the $(n-1) \times (n-1)$ principal submatrix of T_n obtained by deleting the ith row and column of T_n; this submatrix is the 2×2 block diagonal matrix $\mathrm{Diag}(T_j, \hat{T}_{n-j-1})$. □

Remark 2.1. To extend part (a) to the case of any n, apply Theorem 2.2 to the tridiagonal matrix

$$\begin{pmatrix} T_n & O \\ O & qI_s \end{pmatrix}$$

for an appropriate $s < 2k+2$ and any fixed real q, where I_s is the $s \times s$ identity matrix.

Apply part (a) of Theorem 2.2 for $k = 1$, then modify it by replacing $\tilde{T}_i$ by the 1-by-1 matrices (a_{2i}) for $i = 1, \ldots, \frac{n}{2}$, and thus arrive at Corollary 2.1.

COROLLARY 2.1. *If n is a multiple of 4, $\{\sigma_1 \le \sigma_2 \le \cdots \le \sigma_{\frac{n}{2}}\} = \{a_{2i-1}, i = 1, \ldots, \frac{n}{2}\}$, $\{\theta_1 \le \theta_2 \le \cdots \le \theta_{\frac{n}{2}}\} = \{a_{2i}, i = 1, \ldots, \frac{n}{2}\}$, then*

$$\lambda_{\frac{n}{4}} \le \sigma_{\frac{n}{4}} \le \lambda_{\frac{3}{4}n}, \quad \lambda_{\frac{n}{4}} \le \theta_{\frac{n}{4}} \le \lambda_{\frac{3}{4}n}.$$

Corollary 2.1 in a weaker form has been proven in [BFKT], where it states some properties of the zeros of the polynomials generated by the Euclidean scheme for the two polynomials $p(\lambda) = \det(\lambda I - T_n)$ and $p'(\lambda)$ and where it is used in order to solve the root-finding problem in NC for a polynomial having only real zeros. The result of part (b) of Theorem 2.2 has been proven in [BOT], still in terms of the zeros of the polynomials generated by the Euclidean scheme. The proofs in both papers [BFKT] and [BOT] are quite intricate, whereas the matrix formulation of these properties is a straightforward consequence of Theorem 2.1, and the results can be immediately extended to polynomial zeros (as we pointed out in the introduction).

Let us again apply the Courant–Fischer theorem in order to obtain still another interlacing set for the set Λ of the eigenvalues of T_n; this time, we will rely on a suitable rank-one modification of the matrix T_n (cf. Remark 2.1 at the end of this section).

THEOREM 2.3. *Let $\{\phi_1 \le \phi_2 \le \cdots \le \phi_n\}$ be the set of all the eigenvalues of the matrices $S_k = T_k - \mathrm{Diag}(0, \ldots, 0, b_k)$, $R_{n-k} = \hat{T}_{n-k} - \mathrm{Diag}(b_k, \ldots, 0)$, where T_k and $\hat{T}_{n-k}$ are defined in Theorem 2.2. Set $\phi_{n+1} = \phi_n + 2b_k$, $\phi_0 = \phi_1 + 2b_k$.*

If $b_k > 0$, then

$$\phi_i \le \lambda_i \le \phi_{i+1}, \quad i = 1, \ldots, n.$$

If $b_k < 0$, then

$$\phi_{i-1} \le \lambda_i \le \phi_i, \quad i = 1, \ldots, n.$$

Proof. Theorem 2.3 follows from Theorem 8.1-5 of [GL, p. 412], applied to the matrix equation

$$T_n = \mathrm{Diag}(S_k, R_{n-k}) + 2b_k \mathbf{e}\mathbf{e}^T,$$

where $\mathbf{e} = (e_i)$, $e_i = 1/\sqrt{2}$ for $i = k$, and $i = k+1$, $e_i = 0$ elsewhere. □

A different proof of Theorem 2.3 is given in [Cu] and [BNS], where this theorem is used for separating the eigenvalues of an rst-matrix as a basis for devising practically effective divide-and-conquer algorithms for approximating the eigenvalues and eigenvectors of rst-matrices. As in [BNS] and [Cu], we may extend our algorithm to computing the eigenvectors of T_n, although our approach does not require us to compute the eigenvectors if we only need to compute the eigenvalues.

Remark 2.2. The algorithm supporting our asymptotic complexity estimates can be based on Theorems 2.2 or 2.3 as well. Application of Theorem 2.3 leads to some advantages for practical computation, particularly due to the possibility of using the so-called secular function (cf. [Cu], [BP92]).

3. Computing the number of the eigenvalues in the intervals of nearly interlacing sets. With a set interleaving the zeros of $p(\lambda)$ (and the eigenvalues of T_n) available, the subsequent approximations to these zeros (and to these eigenvalues) are obtained by using more customary techniques of [BOT]; for the sake of completeness, we will elaborate a modification of these techniques in this section and in section 4. (Some additional care is required here since we actually start not with an interlacing set but with its approximation.) The resulting algorithm consists of three main stages. At the first stage, specified in this section, we approximate some eigenvalues of T_n within a required error bound and cover each remaining eigenvalue by a real line interval containing no other eigenvalues of T_n. Such an eigenvalue may lie arbitrarily close to the end of the interval and, consequently, to other eigenvalues of T_n. However, the second stage ensures sufficiently strong isolation of all such eigenvalues from each other (by means of the bisection and the double exponential sieve algorithms). In the third stage, we rapidly approximate the isolated eigenvalues by means of Newton's iteration. The second and the third stages are described in section 4.

Now, let the set $\{d_0, \dots, d_n\}$ interleave the set Λ of the eigenvalues of T_n,

$$d_0 < \lambda_1 \le d_1 \le \lambda_2 \le d_2 \le \cdots \le d_{n-1} \le \lambda_n < d_n. \tag{3.1}$$

Let the pairs, d_i^-, d_i^+, of approximations to the real points d_i be given for $i = 1, \dots, n-1$, such that for a fixed Δ, we have

$$d_i^+ - d_i^- = 2\Delta, \quad d_i^- \le d_i \le d_i^+, \quad i = 0, \dots, n; \tag{3.2}$$

that is, the values d_i lie in the intervals

$$\mathcal{I}_i = \{\lambda : \ d_i^- \le \lambda \le d_i^+\}, \ i = 0, \dots, n. \tag{3.3}$$

Let us be given Δ, d_i^-, $d_i^+ = d_i^- + 2\Delta$ for $i = 0, \dots, n$, and the black box subroutine for the exact evaluation of the value at point λ of the characteristic polynomial of T_n,

$$p(\lambda) = \det(T_n - \lambda I). \tag{3.4}$$

In this section, for every eigenvalue λ_j of T_n (for $j = 1, \dots, n$), either we will compute its approximation, within the absolute error bound 2Δ, or we will determine that the interval

$$\mathcal{K}_j = \{\lambda : \ d_{j-1}^+ \le \lambda \le d_{j-}^-\} \tag{3.5}$$

contains λ_j and no other eigenvalues of T_n. This problem was solved in [BOT]; we propose an alternative solution.

With no loss of generality, we assume that

$$p(d_i^-)\ p(d_i^+) \neq 0, \quad i = 0, \dots, n, \tag{3.6}$$

and we will use the next definition and simple auxiliary results implied by (3.1)–(3.6).

DEFINITION 3.1. *The number of the eigenvalues of T_n in a real interval $\mathcal{I}$ is called the index of $\mathcal{I}$ and is denoted $c(\mathcal{I})$.*

PROPOSITION 3.1. *Any interval $\mathcal{K}_j$ cannot contain more than one eigenvalue of T_n.*

PROPOSITION 3.2. *$d_i^- \le \lambda_{i+1} \le d_{i+1}^+$, for $i = 0, 1, \dots, n-1$.*

COROLLARY 3.1. *If $\mathcal{I}_i \cap \mathcal{I}_{i+1} \cap \cdots \cap \mathcal{I}_{i+h} \neq \oslash$, then the points $\tilde{\lambda}_{i+j} = \frac{1}{2}(d_{i+j}^- + d_{i+j-1}^+)$ approximate λ_{i+j} within the absolute error bound 2Δ for $j = 1, 2, \dots, h$. Moreover, $h \le c(\mathcal{I})$, where $\mathcal{I} = \{\lambda :\ d_i^- \le \lambda \le d_{i+h}^+\}$. Furthermore, $c(\mathcal{I}) \le h + 2$, if in addition, $\mathcal{I}_{i-1} \cap \mathcal{I}_i = \mathcal{I}_{i+h} \cap \mathcal{I}_{i+h+1} = \oslash$.*

We will also use the following simple fact.

PROPOSITION 3.3. *Let $p(a)p(b) \neq 0$, $\mathcal{I} = \{\lambda :\ a \le \lambda \le b\}$. Then $p(a)p(b) < 0$ if and only if $c(\mathcal{I})$ is odd.*

ALGORITHM 3.1.

Input: Positive rational Δ, an integer n, and a set of rational numbers $D = \{d_i^-, d_i^+,\ \ i = 1, \dots, n-1\} \cup \{d_0^+ = -3(2^{-m}),\ \ d_n^- = 3(2^m)\}$ such that (3.1), (3.2), and (3.6) hold, and $d_0^+ \le \lambda_1,\ \ \lambda_n \le d_n^-$ (compare (2.1)).

Output: For every j, $j = 1, \dots, n$, either the interval $\mathcal{K}_j$ of (3.5) containing a unique eigenvalue λ_j of T_n or an approximation $\tilde{\lambda}_j$ to λ_j such that $|\lambda_j - \tilde{\lambda}_j| < 2\Delta$.

Stage 1 (form the union of the overlapping intervals): For $i = 0$ and for every i such that $d_i^+ < d_i^-$, determine the maximum $h = h(i)$ such that $d_{i+j-1}^+ \ge d_{i+j}^-$, for $j = 1, \dots, h$, letting $h(i) = 0$ if $d_i^+ < d_{i+1}^-$. Output $\tilde{\lambda}_{i+j} = (d_{i+j}^- + d_{i+j-1}^+)/2$, $j = 1, \dots, h$. (By virtue of Corollary 3.1, $|\lambda_{i+j} - \tilde{\lambda}_{i+j}| \le 2\Delta$.) Save the values $\eta_i = (d_i^- + d_i^+)/2$, $\nu_{i+h+1} = (d_{i+h}^- + d_{i+h}^+)/2$ as candidates for being approximations to λ_i and λ_{i+h+1}. Write

$$\mathcal{J}_{i,h} = \{\lambda :\ d_i^- \le \lambda \le d_{i+h}^+\} = \bigcup_{j=0}^{h} \mathcal{I}_{i+j} \tag{3.7}$$

and observe that

$$\lambda_i \le d_i^+ = d_i^- + 2\Delta,$$

$$\lambda_{i+h+1} \ge d_{i+h}^- \ge d_{i+h}^+ - 2\Delta,$$

so that

$$|\lambda_i - \eta_i| \le \Delta \ \text{ if } \ \lambda_i \in \mathcal{J}_{i,h},$$

$$|\lambda_{i+h+1} - \nu_{i+h+1}| \le \Delta \ \text{ if } \ \lambda_{i+h} \in \mathcal{I}_{i,h}.$$

Stage 2 (define the indices of the intervals $\mathcal{K}_j$ of (3.5) and $\mathcal{J}_{i,h}$ of (3.7)): Compute $p(d_i^-)$ and $p(d_i^+)$ for all i. Recall the relations (3.5)–(3.7), Propositions 3.1 and 3.3, and Corollary 3.1. For all i and j, define

$$c(\mathcal{K}_j) = \begin{cases} 0 & \text{if } p(d_{j+1}^-)\ p(d_j^+) > 0, \\ 1 & \text{otherwise,} \end{cases}$$

$$c(\mathcal{J}_{i,h}) = h+1 \ \text{ if } \begin{cases} \text{either } h \text{ is even and } p(d_i^-)p(d_{i+h}^+) < 0, \\ \text{or } h \text{ is odd and } p(d_i^-)p(d_{i+h}^+) > 0; \end{cases}$$

$$\text{otherwise, either } \ c(\mathcal{J}_{i,h}) - h = 0 \ \text{ or } \ c(\mathcal{J}_{i,h}) - h = 2. \tag{3.8}$$

Output the set of indices j such that $c(\mathcal{K}_j) = 1$, in which case $\lambda_{j-1} \in \mathcal{K}_j$.

Stage 3 (choose approximations among the candidate values): By the beginning of this stage, for every j, $j = 1, \dots, n$, it has been determined whether

$$\lambda_j \in \mathcal{K}_{j+1} \ (\text{see Stage 2})$$

or

$$|\lambda_j - \tilde{\lambda}_j| \le 2\Delta \ \ (\text{for } \tilde{\lambda}_j \ \text{ defined in Stage 1}),$$

or, otherwise, at least one of the next two bounds hold:

$$|\lambda_j - \eta_j| \le \Delta,$$

$$|\lambda_j - \nu_j| \le \Delta.$$

It remains to distinguish between the two latter cases and to choose an approximation to λ_j by one of the two candidates ν_j and η_j. This process relies on the two following simple rules:

(a) For every interval $\mathcal{J}_{i,h}$, of the two candidate values η_i and ν_{i+h+1}, exactly $c(\mathcal{J}_{i,h}) - h$ values should be selected as approximations within Δ to λ_i and/or λ_{i+h} (due to the observations made at the end of the description of Stage 1).

As soon as we decide about selecting one of the two candidate values η_i and ν_{i+h+1}, we apply the latter rule in order to decide if we should or should not select another of the two candidates (recall that we know the parity of $c(\mathcal{K}_j) - h$).

(b) For every j, $j = 1, \dots, n$, of the two consecutive candidate values ν_j and η_j, separated by the single interval $\mathcal{K}_j$, exactly $1 - c(\mathcal{K}_j)$ values should be selected as an approximation to the eigenvalue λ_j of T_n. (Indeed, $d_{j-1}^- < \nu_j < d_{j-1}^+ < d_j^- < \eta_j < d_j^+$ and $d_{j-1}^- \le \lambda_j \le d_j^+$, so that either $d_{j-1}^+ \le \lambda_j \le d_j^-$, and then $c(\mathcal{K}_j) = 1$, or $d_{j-1}^- \le \lambda_j \le d_{j-1}^+$, and then $|\nu_j - \lambda_j| \le \Delta$, or $d_j^- \le \lambda_j \le d_j^+$, and then $|\eta_j - \lambda_j| \le \Delta$.)

Rule (b) implies that neither of the values ν_j and η_j should be selected if $c(\mathcal{K}_j) = 1$; otherwise, we choose exactly one of them as an approximation to λ_j.

Recursive application of rules (a) and (b) completely defines the selection of the approximations to λ_j among all the candidate values η_j and ν_j for all j provided that we are given the values $p(d_i^+)$ and $p(d_i^-)$ for all i and that we know whether the leftmost interval $\mathcal{J}_{j,h}$ contains λ_j. The latter inclusion can be easily checked by using (3.1), (3.2), and Propositions 3.1–3.3. We shall decrease the parallel time of this

selection process by using the following equivalent procedure (the equivalence can be easily verified by inspection).

To select an approximation to λ_j by η_j or by ν_j, first partition the set S of all the intervals $\mathcal{J}_{i,h}$ satisfying (3.8) into the maximal subsets whose consecutive intervals are only interleaved with intervals $\mathcal{K}_j$ having indices 0 and with intervals $\mathcal{J}_{i,h}$ having indices $h+1$. Number the intervals in each maximal subset from left to right by $1, 2, \ldots$. For each subset, let $N(i,h)$ denote the number assigned to the interval $\mathcal{J}_{i,h}$ in this enumeration. Let $\delta = 0$ if there is no interval $\mathcal{K}_j$ having index 1 to the left of the subset and let $\delta = 1$ otherwise. Then determine the indices of the intervals of the subset as follows:

$$c(\mathcal{J}_{i,h}) = h + 1 + (-1)^{\delta + N(i,h)}. \tag{3.9}$$

Select both η_i and ν_{i+h+1} as approximations within Δ to λ_i and λ_{i+h+1} if $c(\mathcal{J}_{i,h}) = h+2$ and select none of them if $c(\mathcal{J}_{i,h}) = h$ (according to rule (a)).

Finally, for every i such that $c(\mathcal{J}_{i,h}) = h+1$, apply rule (b) to select one of the η_i and ν_{i+h+1} as an approximation within Δ to λ_i or to λ_{i+h+1}, respectively.

4. Approximation to the eigenvalues by using Newton's iterations, double exponential sieve, and the bisection method. In this section we will complement Algorithm 3.1 with an algorithm that approximates (within a fixed absolute output error Δ) an eigenvalue λ_j of T_n given a pair of real c and d, such that the interval $\mathcal{K} = \{\lambda : c \le \lambda \le d\}$ contains only this eigenvalue of T_n. We will apply the techniques of [BOT] and [R] based on the following result of [R].

THEOREM 4.1. *Let $p(x) = a_n \prod_{i=1}^{n}(x - \xi_i)$ be a polynomial. Let $x^{(0)} \in \mathbf{C}$ be such that $|x^{(0)} - \xi_1| \le |x^{(0)} - \xi_2| \le \cdots \le |x^{(0)} - \xi_n|$. If*

$$|x^{(0)} - \xi_1| < \frac{1}{5n^2}|x^{(0)} - \xi_2|, \tag{4.1}$$

then the sequence $x^{(i+1)} = x^{(i)} - p(x^{(i)})/p'(x^{(i)})$, $i = 0, 1, \ldots$, generated by Newton's method, converges to ξ_1; moreover, $|x^{(i)} - \xi_1| \le 2^{3-2^i}|x^{(0)} - \xi_1|$.

If we have an initial approximation $x^{(0)}$ such that $c < x^{(0)} < d$ and

$$|x^{(0)} - \lambda_j| \le \frac{1}{5n^2}\min\{x^{(0)} - c,\ d - x^{(0)}\}, \tag{4.2}$$

then we may invoke Theorem 4.1 and use Newton's iteration, thus arriving at the desired approximation to λ_j in $\lceil \log\log(0.8\frac{d-c}{\Delta n^2})\rceil$ Newton's steps. This observation leads us to the next procedure, which consists of the double exponential sieve process (Stages 2 and 3), bisection (Stage 4), together ensuring (4.1), and Newton's iteration (Stage 5), which outputs a real point $\tilde{\lambda}$ such that

$$|\tilde{\lambda} - \lambda| \le \Delta, \quad \lambda = \lambda_j. \tag{4.3}$$

ALGORITHM 4.1.

Input: Natural numbers $c < d$ such that $c < \lambda < d$ and a positive rational Δ.

Output: A rational $\tilde{\lambda}$ such that $|\lambda - \tilde{\lambda}| < \Delta$.

Stage 1. If $d - c < 2\Delta$, then output $\tilde{\lambda} = \frac{c+d}{2}$ and stop. Otherwise, set $\beta = 0$, compute $p(c)$ and $p((c+d)/2)$, set $c_0 = c$, $d_0 = d$, and restrict further computations to the interval $[c, \frac{c+d}{2}]$ if $p(c)p((c+d)/2) < 0$ and to the interval $[\frac{c+d}{2}, d]$ otherwise. Suppose, with no loss of generality, that $p(c) < 0$, $p((c+d)/2) > 0$, and perform the following stages.

Stage 2. Set $\gamma_0 = (c_0 + d_0)/2$ and apply the *bisection of the exponents* (also called the double exponential sieve) procedure to the interval $[c_0, (c_0 + d_0)/2]$; that is, successively evaluate $p(\gamma_i)$ for $\gamma_i = c_0 + (d_0 - c_0)2^{-2^i}$, $i = 1, 2, \ldots$, until either $\gamma_i - c_0 < 2\Delta$ or $\gamma_i - c_0 < \beta$ or $p(\gamma_i) \le 0$. In the first case, output $(c_0 + \gamma_i)/2$ and stop; in the second case, set $d_0 = \gamma_i$, $x_0 = (c_0 + \gamma_i)/2$ and go to Stage 4; otherwise, set $\beta = \gamma_i - c_0$ and go to the next stage. (Note that the second case may only occur for $\beta \ge 2\Delta > 0$, that is, not at the first pass through Stage 2.)

Stage 3. Set $c_1 = \gamma_i$ and $d_1 = \gamma_{i-1}$. (Note that $d_1 - c_1 \ge c_1 - c_0 \ge 2\Delta$.) Compute $p((c_1 + d_1)/2)$. If $p((c_1 + d_1)/2) < 0$, then set $c_0 = (c_1 + d_1)/2$, $d_0 = d_1$ and go to the next stage (in this case, $\lambda \in [c_0, d_0]$ and $d_0 - c_0 < \min\{c_0 - c,\ d - d_0\}$). Otherwise, set $c_0 = c_1$, $d_0 = (c_1 + d_1)/2$, and go to Stage 2.

Stage 4. Apply $\lceil \log(5n^2) \rceil$ bisection steps to $[c_0, d_0]$ in order to find a starting point $x^{(0)}$ satisfying (4.1).

Stage 5. Apply $\lceil \log\log(0.8\frac{d_0 - c_0}{\Delta n^2}) \rceil$ Newton's steps (starting with $x^{(0)}$) in order to arrive at an approximation $\tilde{\lambda}$ to λ such that $|\tilde{\lambda} - \lambda| < \Delta$.

Since β only grows, whereas $d_0 - c_0$ decreases by at least two times, in each recursive call to Stages 2 and 3, there can only be $O(\log^2(m + \log(\Delta^{-1})))$ such calls, and there can only be $O(\log(m + \log(\Delta^{-1})))$ Newton's iteration steps at Stage 5 provided that $-3(2^m) < c < d < 3(2^m)$.

5. Computing the values of the characteristic polynomial and its derivative at a set of points. We will concurrently apply Algorithm 4.1 to all the selected intervals, each containing a single eigenvalue of T_n. At any of the $O(\log^2(m + \log(\Delta^{(-1)})))$ steps we will have to compute the values $p(\lambda)$ and $p'(\lambda)$ at a set of at most n points. In this section we will devise an algorithm for computing the values of the characteristic polynomial $p(\lambda) = p_n(\lambda) = \det(T_n - \lambda I)$ of T_n and of its first derivative at a given set of n points.

Due to the tridiagonal structure of the matrix T_n, we have the following three-term recurrence for the polynomials $p_i(\lambda) = \det(T_i - \lambda I)$:

$$p_0(\lambda) = 1, \quad p_1(\lambda) = a_1 - \lambda,$$

$$p_{i+1}(\lambda) = (a_{i+1} - \lambda)\, p_i(\lambda) - b_i^2 p_{i-1}(\lambda), \quad i = n-1, n-2, \ldots, 1, \tag{5.1}$$

or in the equivalent matrix form,

$$\begin{pmatrix} p_{i+1}(\lambda) \\ p_i(\lambda) \end{pmatrix} = \begin{pmatrix} a_{i+1} - \lambda & -b_i^2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} p_i(\lambda) \\ p_{i-1}(\lambda) \end{pmatrix},$$

so that

$$\begin{pmatrix} p_{i+1}(\lambda) \\ p_i(\lambda) \end{pmatrix} = F_i \ldots F_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{5.2}$$

$$F_j = \begin{pmatrix} a_{j+1} - \lambda & -b_j^2 \\ 1 & 0 \end{pmatrix}, \quad j = 0, \ldots, i, \quad b_0 = 0. \tag{5.3}$$

Now, we will compute the coefficients of $p(\lambda)$. At this stage, we do not have to restrict our computation to the real or complex case; we may allow the input entries of T_n from any field $\mathbf{F}$.

ALGORITHM 5.1.

Input: A positive integer $n = 2^h$ (compare Remark 2.1), and $a_1, \dots, a_n, b_1, \dots, b_{n-1}$ (the entries of T_n, being the elements of a field $\mathbf{F}$).

Output: $\alpha_0, \dots, \alpha_n \in \mathbf{F}$, such that $p(\lambda) = \det(T_n - \lambda I) = \sum_{i=0}^{n} \alpha_i \lambda^i$.

Computation: First set $H_j^{(0)} = F_j$, $j = 0, \dots, n-1$; then, for $i = 1, 2, \dots, \log n = h$, compute $H_j^{(i)} = H_{2j+1}^{(i-1)} H_{2j}^{(i-1)}$, $j = 0, \dots, 2^{-i}n - 1$, output the coefficients of the polynomial $p(\lambda) = (1 \;\; 0) H_0^{(h)} \binom{1}{0}$.

The ith level of the computation is essentially reduced to $8n/2^i$ multiplications of the entries of the matrices $H_j^{(i)}$, which are polynomials of degrees at most 2^i, and to four additions of the products. Over the complex or real fields $\mathbf{F}$, this means the $O_A(\log n, n)$ cost bound at each level i [AHU], [BM]; that is, the overall cost of Algorithm 5.1 is $O_A(\log^2 n, n)$.

Given the coefficients of $p(\lambda)$, and therefore, of $p'(\lambda)$, we may compute the values of both $p(\lambda)$ and $p'(\lambda)$ on a fixed set of n points, at the cost $O_A(\log^2 n \log\log n, n/\log\log n)$ (see [AHU], [BM], [RT]).

Now recall that we apply Newton's iteration to approximate the n eigenvalues of T_n. Each Newton's iteration step essentially amounts to computing the ratio $p(x)/p'(x)$ at n points, which now means the cost $O_A(\log^2 n \log\log n, n/\log\log n)$, and this is also an estimate for the cost of performing Algorithm 3.1.

Remark 5.1. Given the matrix T_n and a scalar x, $p(\lambda)$ can be evaluated at $\lambda = x$ at the cost $O_A(\log n, n)$. It is sufficient to apply Algorithm 5.1 replacing λ by $\lambda + x$ and performing the computation modulo λ. The same computation modulo λ^2 outputs the linear polynomial $p(x) + p'(x)\lambda$. Thus, given an $n \times n$ rst-matrix T_n, we may compute $p(\lambda)$ and $p'(\lambda)$ at $O(n)$ points at the cost $O_A(\log n, n^2)$, avoiding the evaluation of the coefficients of $p(\lambda)$. Even though the number of processors grows, the numerical stability of the computations is greatly improved in this way, which is the basis of the practical modification of our algorithm presented in [BP92]. The latter algorithm actually uses a slightly different modification of Algorithm 5.1, which yields the same output values of $p(\lambda)$ and $p'(\lambda)$ at the same computation cost but in, numerically, a more stable way. Besides improved numerical stability, we have also modified (in [BP92]) the isolation stage so as to bound the *relative* output errors, which is important for practical implementation.

6. The main algorithm. By using the tools and steps described in the preceding sections, we will now devise our main algorithm for the approximation of the eigenvalues of an $n \times n$ rst-matrix T_n with integer entries. This is a divide-and-conquer algorithm, which recursively reduces the original computational problem to two problems of half-size.

ALGORITHM 6.1.

Input: Two positive integers m and n, positive u; n integers $a_1, \dots, a_n$ and $n-1$ nonzero integers $b_1, \dots, b_{n-1}$, such that n is a power of 2 (compare Remark 2.1), $|a_i|$, $|b_i| \le 2^m$. (This input defines an rst-matrix T_n and the tolerance 2^{-u} to the output errors.)

Output: Reals $\gamma_1, \dots, \gamma_n$ such that $|\gamma_i - \lambda_i| < 2^{-u}$, where λ_i are the eigenvalues of T_n, $i = 1, \dots, n$.

Stage 1. Compute the coefficients of the characteristic polynomial of T_n by applying Algorithm 5.1.

Stage 2. Apply Algorithm 6.1 to the input set

$$u+1,\ m,\ \frac{n}{2},\ a_1,\dots,a_{\frac{n}{2}-1},a_{\frac{n}{2}}-b_{\frac{n}{2}},b_1,\dots,b_{\frac{n}{2}-1},$$

which defines an rst-matrix $S_{\frac{n}{2}}$, and to the input set

$$u+1,\ m,\ \frac{n}{2},\ a_{\frac{n}{2}+1}-b_{\frac{n}{2}},a_{\frac{n}{2}+2},\dots,a_n,b_{\frac{n}{2}},\dots,b_{n-1},$$

which defines an rst-matrix $R_{\frac{n}{2}}$, thus obtaining approximations $\delta_1 \le \delta_2 \le \cdots \le \delta_n$ to the eigenvalues of $S_{\frac{n}{2}}$ and $R_{\frac{n}{2}}$ within the absolute error bound $\Delta = 2^{-u-1}$. (The matrices $S_{\frac{n}{2}}$ and $R_{\frac{n}{2}}$ have been defined in Theorem 2.3.)

Stage 3. Recall that the set of all the eigenvalues of $S_{\frac{n}{2}}$ and $R_{\frac{n}{2}}$ interleaves the set $\{\lambda_i\}$ (see Theorem 2.3), set $d_i^+ = \delta_i + \Delta$, $d_i^- = \delta_i - \Delta$, and apply Algorithms 3.1 and 4.1, to obtain $\gamma_1,\dots,\gamma_n$ such that $|\gamma_i - \lambda_i| < 2^{-u}$.

The overall cost of performing the algorithm is $O_A(\log^3 n \log\log n(\log^2 b + \log n), n/\log\log n)$, $b = u+m$, but this bound can be decreased to $O_A(\log^2 n \log\log n(\log^2 b + \log n), n/\log\log n)$, since we need the output of Algorithm 4.1 with a precision lower than b bits until we arrive at the stage of approximating the eigenvalues of the original matix T_n (see the details in [BOT]).

Remark 6.1. An equivalent version of Algorithm 6.1 can be obtained by replacing the matrices $S_{\frac{n}{2}}$ and $R_{\frac{n}{2}}$ at Stage 3 by the matrices $T_{\frac{n}{2}}$ and $\hat{T}_{\frac{n}{2}-1}$ of part (b) of Theorem 2.2.

7. Bit-complexity estimates. We will prove the following result.

THEOREM 7.1. *Algorithm* 6.1 *can be implemented at the Boolean cost bounded by* $O_B(\log(nb)\log^2 n(\log^2 b + \log n)\log\log n \log\log(bn), n^2\frac{b+\log n}{\log\log n})$, *where* $|a_i|$, $|b_i| \le 2^m$, $b = m+u$.

The most expensive computations in Algorithm 6.1 are the evaluation of the coefficients of the characteristic polynomial $p(\lambda)$ of T_n at Stage 1, performed by means of Algorithm 5.1, and the evaluation at Stage 3 of the values of $p(\lambda)$ and $p'(\lambda)$ on a set of n points, performed by means of the multipoint polynomial evaluation algorithm of [AHU], [BM].

We will use the following auxiliary result.

THEOREM 7.2. *Let* n *be a power of* 2, k *a positive integer,* $p = 2^{nk/2}+1$, $u(x)$ *and* $v(x)$ *two polynomials with coefficients in* $\mathbf{Z}_p$ *(*$\mathbf{Z}_p$ *is the ring of integers with arithmetic modulo* p*),* $\deg v(x) = n_2 \le \deg u(x) = n_1 = O(n)$. *Then the coefficients of the polynomial*

$$w(x) = u(x)v(x)$$

can be computed modulo p *at the Boolean cost* $O_B(\log n \log(nk)\log\log(nk), n^2k)$. *Moreover, if the polynomial* $v(x)$ *is monic, the coefficients of the two polynomials* $q(x)$ *and* $r(x)$ *such that*

$$u(x) = v(x)q(x) + r(x)$$

and degree$(r(x)) < n_2$ *can be computed modulo* p *at the Boolean cost bounded by* $O_B(\log n \log\log n \log(nk)\log\log(nk), \frac{n^2k}{\log\log n})$.

Proof. We just need to combine the known estimates from [AHU] and [RT]. The Boolean cost of each arithmetic operation in $\mathbf{Z}_p$ is $O_B(\log d \log\log d, d)$, $d = \lceil \log p \rceil$

(see [AHU, p. 226], [RT]). Moreover, $2^{jk} \neq 1 \bmod p$, $j = 1, \ldots, n-1$, and $2^{nk} = 1 \bmod p$; that is, 2^k is a primitive nth root of 1 in $\mathbf{Z}_p$, and any integer power of 2, in particular, n, has its inverse in $\mathbf{Z}_p$. Therefore, in $\mathbf{Z}_p$, we may compute the product of a pair of nth-degree polynomials, by means of three FFTs, at the arithmetic cost $O_A(\log n, n)$ and, therefore, at the Boolean cost $O_B(\log n \log(nk) \log\log(nk), n^2 k)$. Finally, no divisions are needed for computing $q(x)$ and $r(x)$, since $v(x)$ is monic, and the computation can be performed in $\mathbf{Z}_p$ by using the polynomial division algorithm of [RT]. We arrive at the cost bounds $O_A(\log n \log\log n, n/\log\log n)$ and, therefore,

$$O_B(\log n \log(nk) \log\log n \log\log(nk), n^2 k/\log\log n). \qquad \square$$

Now, recall that the ith stage of Algorithm 5.1 (which evaluates the coefficients of the characteristic polynomial of T_n) consists in evaluating $8(n/2^i)$ products of pairs of polynomials of degrees at most 2^{i-1} and having integer coefficients whose absolute values are less than $2^{i+m2^{i+1}}$, for $i = 1, 2, \ldots, \log n$. Such coefficients are well defined by their values modulo $p = 2^{nk/2} + 1$, already for $k = 4m + 4$. Therefore, we may use integer arithmetic modulo p, apply Theorem 7.2, and deduce that the cost of performing Algorithm 5.1 is given by

$$O_B(\log^2 n \log(nm) \log\log(nm), n^2 m).$$

This bound is dominated by the cost bound of Theorem 7.1, since $b = m + u$, $u > 0$, and Brent's principle enables us to multiply the time bound by $s = \log\log n$ and simultaneously divide the processor bound by s.

Now, consider the evaluation of $p(x)$ and $p'(x)$ at a set of points $\gamma_1, \gamma_2, \ldots, \gamma_n$. We first observe that $|\gamma_i| \leq 2^{m+2}$, since γ_i, for $i = 1, \ldots, n$, are approximations, within absolute errors at most $2^{-u} \leq 1$, to the eigenvalues of rst-matrices having the 1-norms at most $3(2^m)$. Moreover, we may consider γ_i a binary value, $\gamma_i = y_i/2^{u+\log n}$, where y_i is an integer, $|y_i| \leq 2^{m+2+u+\log n}$, since we are looking for an approximation within the absolute error bound $2^{-u-\log n}$. Therefore, the evaluation of $p(\gamma_i)$ and $p'(\gamma_i)$ can be kept within the set of integers in the following way.

Consider $Q(y) = \sum_{i=-1}^{n} \alpha_i 2^{(u+\log n)(n-i)} y^i$, where $p(x) = \sum_{i=0}^{n} \alpha_i x^i$. The polynomial $Q(y)$ has integer coefficients, each represented with at most $n(m+u+2\log n)$ bits, and satisfies the following relation: $Q(y) = 2^{(u+\log n)n} p(x)$, $y = 2^{u+\log n} x$. Moreover, the size of the integers $Q(y_i)$ is bounded as follows:

$$|Q(y_i)| \leq 2^{n(2m+2u+3\log n+2)} = 2^{nk/2}, \quad k = O(m+u+\log n). \tag{7.1}$$

Now, we apply the known algorithm for multipoint polynomial evaluation [AHU], [BM], where we use integer arithmetic modulo $p = 2^{nk/2} + 1$ in the following way.

Recursively compute the coefficients of the monic polynomials $S_i^{(j)}$ of degrees 2^j,

$$S_k^{(0)} = y - y_k, \quad k = 1, \ldots, n,$$

$$S_i^{(j)} = S_{2i-1}^{(j-1)} S_{2i}^{(j-1)}, \quad i = 1, \ldots, \frac{n}{2^j}, \quad j = 1, \ldots, \log n - 1.$$

Recursively compute the coefficients of the polynomials $r_k^{(j)}$ of degrees at most $n/2^j$,

$$r_1^{(0)} = Q(y),$$

$$r_{2i-1}^{(j)} = r_i^{(j-l)} \bmod S_{2i-1}^{(\log n - j)},$$

$$r_{2i}^{(j)} = r_i^{(j-l)} \bmod S_{2i}^{(\log n - j)},$$

$i = 1, \dots, 2^{j-1}$, $j = 1, \dots, \log n$, such that $r_k^{(\log n)} = Q(y_k)$, $k = 1, \dots, n$. Combining the relations (7.1) and Theorem 7.1 and recalling that $b = m + \mu$, we deduce that such a multipoint polynomial evaluation can be performed at the overall cost

$$O_A(\log^2 n \log\log n, n/\log\log n)$$

and, consequently,

$$O_B(\log^2 n \log\log n \log(nb) \log\log(nb), n^2 b/\log\log n).$$

Therefore, Algorithm 6.1 can be carried out at the cost

$$O_B(\log^2 n \log(nb)(\log^2 b + \log n) \log\log n \log\log(nb), n^2 b/\log\log n),$$

performing a total of

$$O(n^2 b \log^2 n \log(nb)(\log^2 b + \log n) \log\log(nb))$$

bit-operations.

8. Discussion. We keep our results and proofs in their original form (cf. also [BP91]) assuming the PRAM models, though this assumption is not pertinent to the efficiency of our algorithms. Since the time of the submission of the present paper, some related results have appeared. The recent divide-and-conquer algorithms of [P95], [P96] approximate within 2^{-b} all the n complex zeros of any nth-degree polynomial, with its zeros in the unit disc, by using $O(n \log^2 b \log^2 n)$ arithmetic operations or $O((b+n)n^2 \log^2 n \log(bn) \log\log(bn))$ bit-operations; moreover, these algorithms have NC- and processor-efficient parallelization. The latter complexity bounds apply to any polynomial and are only slightly inferior to the current record bounds of the [BOT] and the present paper, which are restricted to the case where all the zeros of $p(x)$ are real. Combining the results of [P95], [P96] with the known algorithms for computing the coefficients of the characteristic polynomial of a general $n \times n$ matrix A gives an NC- and processor-efficient algorithm for the unsymmetric eigenvalue problem for A. On the other hand, the algorithms of [P95], [P96] do not extend directly to approximating the eigenvalues of unsymmetric matrices; extension of the divide-and-conquer techniques to the latter problem is a challenging open problem of practical importance.

Some minor improvements of the parallel complexity estimates of the present paper are possible, for instance, due to the recent improvement of parallel polynomial division achieved in [BP93] or via replacement of some integer divisions by multiplications.

The algorithm of this paper has several attractive features for its practical application; in particular, its computational cost is low, and its rapid convergence is guaranteed even where the input rst-matrix has clustered eigenvalues. The only major obstacle for the practical implementation is the stage of fast multipoint polynomial evaluation, which is known to be numerically unstable. There are two ways out of this difficulty, not counting recent progress in improving multipoint polynomial evaluation

(cf. [P95a], [PSLT], [PZHY]). One way, elaborated upon in [BP92], [B], and [BG1], proceeds by replacing the latter stage by slower but numerically stable computation. Another way, proposed and elaborated in [GE], relies on replacing the stage of multipoint polynomial evaluation by solving the associated secular equation by means of the multipole algorithm of [Ro85]. Unfortunately, [GE] deceptively claims its contribution to decreasing the known estimates for the computational complexity of the symmetric tridiagonal eigenproblem. In fact, the paper [GE] contains neither computational complexity estimates nor proper analysis of the case of clustered eigenvalues (effectively treated by the techniques of [BOT], [BP91], and [BP92]). Furthermore, [GE] fails to inform its readers about the existence of the *much earlier* papers [BOT], [BP91], and [BP92], with faster algorithms for the rst-eigenvalues, whereas comparison and, perhaps, combination of the techniques and the results of these papers with ones of [GE] could be informative and useful for the study of the symmetric tridiagonal eigenproblem. We also recall the papers [R93] (as the rediscovery of [BOT]) and [R97], whose main result (on multipoint polynomial evaluation) repeats one of [PSLT] and [PZHY].

Appendix A.

A.1. Reduction of approximating polynomial zeros to approximating matrix eigenvalues. Let $p(x)$ be a polynomial of a degree n having integer coefficients in the range from 2^m to 2^m and having only real zeros. There exist many $n \times n$ rst-matrices T_n whose characteristic polynomials are proportional to $p(x)$, that is, equal to $p(x) = p_n(x)$ after their appropriate normalization. We may specify T_n much better if, in addition to $p_n(x)$, we will fix the characteristic polynomials $p_{n-1}(x)$ of T_{n-1}, the $(n-1) \times (n-1)$ leading principal submatrix of T_n, and apply the extended Euclidean algorithm to $p_n(x)$ and $p_{n-1}(x)$. Suppose that we have chosen $p_{n-1}(x)$ such that this algorithm performs all its $n-1$ recursive steps, producing the $(n-1-i)$th-degree polynomial in the ith step, for $i = 1, \dots, n-1$. (This holds, in particular, if $p(x)$ has only real zeros and if $p_{n-1}(x) = -p'(x)$.) Then, due to (5.1), $p(x) = p_n(x)$ is the characteristic polynomial of an rst-matrix T_n whose entries a_i, b_i satisfy (5.1) for all i, and such a matrix T_n is defined uniquely, except that we may vary the signs of b_i as we like.

Let us analyze the complexity and errors of these computations assuming that $p_{n-1}(x) = -p'(x)$. Then the computation by the extended Euclidean algorithm can be performed at the cost bounded by $O_A(n \log^2 n, 1)$ [AHU] or $O_A(\log^3 n, n^2/\log n)$ [BP94]. Combining these bounds with the cost bounds of Algorithm 6.1 implies the estimates $O_A(n \log^2 n(\log^2 b + \log n), 1)$ and $O_A(\log^2 n(\log^2 b + \log n), (n/\log b)^2/\log n)$ for the sequential and parallel arithmetic complexity of approximating the zeros of a polynomial having only real zeros (compare [BOT]), and similarly, the sequential and parallel Boolean complexity estimates for approximating the zeros of $p(x)$ can be reduced to the estimates for Boolean complexity of the extended Euclidean computations and approximating the eigenvalues of T_n. The estimates for the sequential Boolean complexity and for the parallel Boolean time of the latter stage dominate the respective estimates for the overall complexity of approximating the zeros of $p(x)$, whereas the opposite is true for the bit-serial processor bound.

We will conclude this part of the appendix with two theorems that will enable us to bound the precision of the entries of T_n remaining within a prescribed tolerance to the errors of approximating the eigenvalues of T_n. As before, let a_i, $i = 1, \dots, n$, and b_j, $j = 1, \dots, n-1$, denote the diagonal and the subdiagonal entries of T_n, respectively. We do not assume any bounds on a_i and b_j but will deduce them.

THEOREM A.1. *The entries of T_n have absolute values at most $2^{m+1.5}$.*

Proof. Due to the Cauchy well-known bounds on the magnitudes of polynomial zeros, the zeros of $p(x)$ have absolute values at most $1+2^m < 2^{m+1}$. Due to Theorem 2.1, for $r = 1$, the diagonal entries of T_n, that is, $a_1, \dots, a_n$, have absolute values at most 2^{m+1}. By applying Theorem 2.1 with $r = 2$, we deduce that the eigenvalues of all the 2×2 submatrices

$$\begin{pmatrix} a_i & b_i \\ b_i & a_{i+1} \end{pmatrix}$$

have absolute values at most 2^{m+1}, that is, $|a_i a_{i+1} - b_i^2| \le 2^{2m+2}$; whence $|b_i| \le 2^{m+1.5}$. □

On the other hand, the eigenvalues of T_n are not very sensitive to the pertubation of the entries.

THEOREM A.2. *Let $\tilde{T}_n$ be an rst-matrix having diagonal and subdiagonal entries $\tilde{a}_i$, $i = 1, \dots, n$, and $\tilde{b}_j$, $j = 1, \dots, n-1$, respectively, such that $|\tilde{a}_i - a_i|$, $|\tilde{b}_j - b_j| \le 2^{-\nu}$, $i = 1, \dots, n$; $j = 1, \dots, n-1$. Then for any eigenvalue λ_i of T_n, there exists an eigenvalue $\tilde{\lambda}_i$ of $\tilde{T}_n$ such that $|\tilde{\lambda}_i - \lambda_i| \le 3(2^{-\nu})$.*

Proof. Theorem A.2 follows from the Bauer–Fike theorem (see [GL, p. 342]) applied for the Euclidean norm, since $\|\tilde{T} - T\|_2 \le 3(2^{-\nu})$. □

From the above theorems, we deduce that the rst-matrix $\tilde{T}_n$ obtained by setting $\tilde{a}_i = \lceil 2^\nu a_i \rceil$, $\tilde{b}_j = \lceil 2^\nu b_j \rceil$ has integer entries with absolute values at most $2^{\nu+m+1.5}$; furthermore, its eigenvalues, divided by $2^{-\nu}$, yield approximations $\tilde{\lambda}_i$ to the eigenvalues of T_n such that $|\tilde{\lambda}_i - \lambda_i| \le 3(2^{-\nu})$. Thus, to insure the latter bound, it suffices to compute the entries of T_n with the precision of $\lceil \nu + m + 1.5 \rceil$ bits.

A.2. Reduction of a Hermitian or real symmetric matrix to the tridiagonal form. Various randomized techniques are well known [GL] for the reduction of an $n \times n$ Hermitian or real symmetric matrix A to an rst-matrix T_n via similarity transformations (which leave invariant the eigenvalues and the characteristic polynomial of A). In [P87] and [BP94, Proposition 5.4, p. 325], a parallel implementation of such a tridiagonal reduction is shown and is analyzed. In particular, in the implementation of [BP94], tridiagonal reduction is essentially reduced

(a) to computation of the $2n + 1$ scalars $h_i = \vec{p}^T A^i \vec{q}$, $i = 0, 1, \dots, 2n$, for two random column vectors $\vec{p}$ and $\vec{q}$, and

(b) to the LDL^T (triangular) factorization of the associated Hankel matrix $H = (h_{i,j})$, $h_{i,j} = h_{i+j}$, $i, j = 0, 1, \dots, n-1$.

The overall computational cost of such a reduction can be bounded by $O_A(\log^2 n, n^3/\log n)$ or, alternatively, by $O_A(\log^3 n, P(n)/\log n)$, provided that a pair of $n \times n$ matrices can be multiplied at the cost bounded by $O_A(\log n, P(n)/\log n)$, $P(n) = O(n^{2.38})$.

Acknowledgments. The authors thank Prasoon Tiwari for kindly supplying a copy of [BOT] and the referee for helpful comments.

REFERENCES

[AHU] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, *The design and analysis of computer algorithms*, Addison-Wesley, Reading, MA, 1976.

[B] D. BINI, *Divide and conquer techniques for the polynomial root-finding problem*, in Proc. 1st World Congress of Nonlinear Analysts, Tampa, FL, 1992, V. Lakshmikantham, ed., Walter de Gruyter, Berlin, 1996, pp. 3885–3896.

[BFKT] M. Ben-Or, E. Feig, D. Kozen, and P. Tiwari, *A fast parallel algorithm for determining all roots of a polynomial with real roots*, SIAM J. Comput., 17 (1988), pp. 1081–1092.

[BG1] D. Bini and L. Gemigniani, *Iteration schemes for the divide-and-conquer eigenvalue solver*, Numer. Math., 67 (1994), pp. 403–425.

[BM] A. Borodin and I. Munro, *The Computational Complexity of Algebraic and Numeric Problems*, American Elsevier, New York, 1975.

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

[BOT] M. Ben-Or and P. Tiwari, *Simple algorithm for approximating all roots of a polynomial with real roots*, J. Complexity, 6 (1990), pp. 417–442.

[BP91] D. Bini and V. Y. Pan, *Parallel complexity of tridiagonal symmetric eigenvalue problem*, in Proc. 2nd Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, and SIAM, Philadelphia, 1991, pp. 384–393.

[BP92] D. Bini and V. Y. Pan, *Practical improvement of the divide-and-conquer eigenvalue algorithms*, Computing, 48 (1992), pp. 109–123.

[BP93] D. Bini and V. Y. Pan, *Improved parallel polynomial division*, SIAM J. Comput., 22 (1993), pp. 617–627.

[BP94] D. Bini and V. Y. Pan, *Matrix and Polynomial Computations, Volume 1: Fundamental Algorithms*, Birkhauser, Boston, 1994.

[Br] R. P. Brent, *The parallel evaluation of general arithmetic expressions*, J. Assoc. Comput. Mach., 21 (1974), pp. 201–208.

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

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

[GL] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins Univ. Press, Baltimore, MD, 1989.

[Pan 87] V. Y. Pan, *Complexity of parallel matrix computations*, Theoret. Comput. Sci., 54 (1987), pp. 65–85.

[P87] V. Y. Pan, *Sequential and parallel complexity of approximate evaluation of polynomial zeros*, Comput. Math. Appl., 14 (1987), pp. 591–622.

[P95] V. Y. Pan, *Optimal (up to polylog factors) sequential and parallel algorithms for approximating complex polynomial zeros*, in Proc. 27th Annual ACM Symposium on Theory of Computing, ACM, New York, 1995, pp. 741–750.

[P95a] V. Y. Pan, *An algebraic approach on approximate evaluation of a polynomial on a set of real points*, Adv. Comput. Math., 3 (1995), pp. 41–58.

[P96] V. Y. Pan, *Optimal and nearly optimal algorithm for approximating complex polynomial zeros*, Comput. Math. Appl., 31 (1996), pp. 97–138.

[Par] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[PSLT] V. Y. Pan, A. Sadikou, E. Landowne, and O. Tiga, *A new approach to fast polynomial interpolation and multipoint evaluation*, Comput. Math. Appl., 25 (1993), pp. 25–30.

[PZHY] V. Y. Pan, A. Zheng, X. Huang, and Y. Yu, *Fast multipoint polynomial evaluation and interpolation via computations with structured matrices*, Ann. Numer. Math., 4 (1997), pp. 483–510.

[Ro85] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.

[RT] J. H. Reif and S. H. Tate, *Optimal size integer division circuits*, in Proc. 21st ACM Symposium on Theory of Computing, ACM Press, New York, 1989, pp. 264–270.

[R] J. Renegar, *On the worst-case arithmetic complexity of approximating zeros of polynomials*, J. Complexity, 3 (1987). pp. 90–113.

[R93] J. H. Reif, *An $O(n \log^3 n)$ algorithm for the real root problem*, in Proc. 34th Annual IEEE Symposium on Foundations on Computer Science, IEEE, Piscataway, NJ, 1993, pp. 626–635.

[R97] J. H. Reif, *Approximate complex polynomial evaluation in near constant work per point*, in Proc. 29th Annual ACM Symposium on Theory of Computing, ACM, New York, 1997, pp. 30–39.