SIAM J. MATRIX ANAL. APPL.
Vol. 27, No. 1, pp. 153–175

THE EHRLICH–ABERTH METHOD FOR THE NONSYMMETRIC TRIDIAGONAL EIGENVALUE PROBLEM*

DARIO A. BINI†, LUCA GEMIGNANI†, AND FRANÇOISE TISSEUR‡

Abstract. An algorithm based on the Ehrlich–Aberth iteration is presented for the computation of the zeros of $p(\lambda) = \det(T-\lambda I)$, where T is a real irreducible nonsymmetric tridiagonal matrix. The algorithm requires the evaluation of $p(\lambda)/p'(\lambda) = -1/\mathrm{trace}(T-\lambda I)^{-1}$, which is done by exploiting the QR factorization of $T-\lambda I$ and the semiseparable structure of $(T-\lambda I)^{-1}$. The choice of initial approximations relies on a divide-and-conquer strategy, and some results motivating this strategy are given. Guaranteed a posteriori error bounds based on a running error analysis are proved. A Fortran 95 module implementing the algorithm is provided and numerical experiments that confirm the effectiveness and the robustness of the approach are presented. In particular, comparisons with the LAPACK subroutine `dhseqr` show that our algorithm is faster for large dimensions.

Key words. nonsymmetric eigenvalue problem, symmetric indefinite generalized eigenvalue problem, tridiagonal matrix, root finder, QR decomposition, divide and conquer

AMS subject classification. 65F15

DOI. 10.1137/S0895479803429788

1. Introduction.

Real nonsymmetric tridiagonal eigenvalue problems arise as intermediate steps in a variety of eigenvalue problems. For example, the nonsymmetric eigenvalue problem can be reduced in a finite number of steps to nonsymmetric tridiagonal form [13], [18]. In the sparse case, the nonsymmetric Lanczos algorithm produces a nonsymmetric tridiagonal matrix. Other motivation for this work comes from the symmetric quadratic eigenvalue problem

$$(\lambda^2 M + \lambda C + K)x = 0,$$

with M, C, and K real symmetric matrices, which is frequently encountered in structural mechanics [40]. The standard way of dealing with this problem in practice is to reformulate it as a generalized eigenvalue problem (GEP) $Ax = \lambda Bx$ of twice the dimension, a process called linearization. Symmetry in the problem can be maintained with an appropriate choice of linearization [40], such as, for example,

$$A = \begin{bmatrix} 0 & K \\ K & C \end{bmatrix}, \quad B = \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix}, \quad x = \begin{bmatrix} u \\ \lambda u \end{bmatrix}.$$

The resulting A and B are symmetric but not definite, and in general the pair (A, B) is indefinite. When the pair (A, B) is of small to medium size, it can be reduced to a symmetric tridiagonal-diagonal pair (S, D) using one of the procedures described by Tisseur [39]. This is the most compact form that can be obtained in a finite number

*Received by the editors June 5, 2003; accepted for publication (in revised form) November 15, 2004; published electronically July 13, 2005.
http://www.siam.org/journals/simax/27-1/42978.html

†Dipartimento di Matematica, Università di Pisa, via Buonarroti 2, 56127 Pisa (bini@dm.unipi.it, gemignan@dm.unipi.it). The work of the first author was supported by MIUR grant 2002014121. The work of the second author was supported by GNCS of Istituto Nazionale Di Alta Matematica.

‡Department of Mathematics, University of Manchester, Manchester, M13 9PL, England (ftisseur@ma.man.ac.uk, http://www.ma.man.ac.uk/~ftisseur/). The work of this author was supported by Engineering and Physical Sciences Research Council grant GR/R45079 and Nuffield Foundation grant NAL/00216/G.

of steps. For large and sparse matrices the pseudo-Lanczos algorithm of Parlett and Chen [31] applied to $A - \lambda B$ yields a projected problem $S - \lambda D$ with S symmetric tridiagonal and D diagonal. In both cases, the eigenvalues of the symmetric pair (S, D) are the same as the eigenvalues of the nonsymmetric tridiagonal matrix $T = D^{-1}S$.

Our aim is to derive a robust algorithm that computes all the eigenvalues of a real $n \times n$ nonsymmetric tridiagonal matrix T in $O(n^2)$ operations. The QR algorithm [20] does not preserve tridiagonal structure: the matrix T is considered as a Hessenberg matrix and the upper part of T is filled in along the iterations. Therefore the QR algorithm requires some extra storage and the eigenvalues are computed in $O(n^3)$ operations. Two alternatives are the LR algorithm [38] for nonsymmetric tridiagonal matrices and the HR algorithm [7], [8]. Both algorithms preserve the tridiagonal form of T but may be unstable since they use nonorthogonal transformations. Attempts to solve the nonsymmetric tridiagonal eigenvalue by generalizing Cuppen's divide-and-conquer algorithm have been unsuccessful because of a lack of good root finders and because deflation is not as advantageous as it is in the symmetric case [2], [27].

In this paper we propose a root finder for the characteristic polynomial of T based on the Ehrlich–Aberth method [1], [15]. This method approximates simultaneously all the zeros of a polynomial $p(z)$: given a vector $z^{(0)} \in \mathbb{C}^n$ of initial approximations to the zeros of $p(z)$, the Ehrlich–Aberth iteration generates a sequence $z^{(j)} \in \mathbb{C}^n$ which locally converges to the n-tuple of the roots of $p(z)$, according to the equation

$$z_j^{(k+1)} = z_j^{(k)} - \frac{\dfrac{p(z_j^{(k)})}{p'(z_j^{(k)})}}{1 - \dfrac{p(z_j^{(k)})}{p'(z_j^{(k)})} \sum_{k=1, k\neq j}^{n} \dfrac{1}{z_j^{(k)} - z_k^{(k)}}}, \quad j = 1{:}\,n. \tag{1.1}$$

The convergence is superlinear (cubic or even higher if the implementation is in the Gauss–Seidel style) for simple roots and linear for multiple roots. In practice, the Ehrlich–Aberth iteration has good global convergence properties, though no theoretical results seem to be known about global convergence. In principle, we cannot exclude the possibility that the Ehrlich–Aberth simultaneous iteration fails to converge or fails to approximate certain roots. However, in practice this has never been encountered (see [5], [6]).

Other techniques for simultaneous iterations such as Laguerre, Durand–Kerner, Euler-like, and Halley-like methods [17], [34], [35], [36], [37] could be used as well. However, our computational experience in polynomial root-finding indicates that compared with these iterations, the Ehrlich–Aberth method has the advantages of requiring a small number of iterations for convergence and having a small cost per iteration. Note that the quasi-Laguerre iteration in [14] relies on the eigenvalues to be real.

The main requirements when using the Ehrlich–Aberth method for computing the roots of $p(z)$ are

1. a fast, robust, and stable computation of the Newton correction $p(z)/p'(z)$;
2. a criterion for choosing the initial approximations to the zeros, $z^{(0)}$, so that the number of iterations needed for convergence is not too large.

For the first issue, Bini [4] shows that Horner's rule is an effective tool when $p(z)$ is expressed in terms of its coefficients. In this case the cost of each simultaneous iteration is $O(n^2)$ operations. Moreover, Horner's rule is backward stable and its computation provides a cheap criterion to test whether the given approximation is in the root-neighborhood (pseudospectrum) of the polynomial [4]. This makes the Ehrlich–Aberth method an effective tool for approximating polynomial roots [6], and

it is now part of the MPSolve package (Multiprecision Polynomial Solver) [5].

In our context, where $p(\lambda) = \det(T - \lambda I)$ is not available explicitly, we need a tool having the same features as Horner's rule, that is, a tool that allows us to compute in a fast, stable, and robust way the Newton correction $p(\lambda)/p'(\lambda)$. This issue is discussed in section 2, where we use the QR factorization of $T - \lambda I$ and the semiseparable structure of $(T - \lambda I)^{-1}$ to compute the Newton correction by means of the equation

$$\frac{p(\lambda)}{p'(\lambda)} = -\frac{1}{\operatorname{trace}(T - \lambda I)^{-1}}.$$

The algorithm that we obtain in this way fulfills the desired requirements of robustness and stability; in particular, it does not suffer unduly from underflow or overflow.

For the second issue, concerning the choice of initial approximations, we rely on the specific features of our eigenvalue problem. More precisely, we apply a divide-and-conquer strategy where the initial approximations are obtained by computing the eigenvalues of two suitable tridiagonal matrices of sizes $m = \lceil n/2 \rceil$ and $n - m$. Even though there are no theoretical results guaranteeing convergence under this choice, we provide in section 3 some theoretical results that motivate this strategy.

The complete algorithm is described in section 4, where we also deal with the issues of computing eigenvectors and running error bounds. Numerical experiments in section 5 illustrate the robustness of our algorithm. In particular, our results show that in most cases our algorithm performs faster than the LAPACK subroutine `dhseqr` for $n \geq 800$ and the speed-up for $n = 1600$ ranges from 3 to 70. The implementation in Fortran 95 is available as a module at www.dm.unipi.it/∼bini/software, file eigen_v1.1.tgz.

2. Computing the Newton correction. Our aim in this section is to derive a fast, robust, and stable method for computing the Newton correction $p(\lambda)/p'(\lambda)$, where $p(\lambda) = \det(T - \lambda I)$.

The tridiagonal matrix

$$T = \begin{bmatrix} \alpha_1 & \gamma_1 & & & 0 \\ \beta_1 & \alpha_2 & \gamma_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \gamma_{n-1} \\ 0 & & & \beta_{n-1} & \alpha_n \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{2.1}$$

is said to be *unreduced* or *irreducible* if $\beta_j\gamma_j \neq 0$ for $j = 1\colon n-1$. We denote by T_k the leading principal submatrix of T in rows and columns 1 through k and let $p_k = \det(T_k - \lambda I)$.

A natural approach is to compute $p(\lambda) = p_n(\lambda)$ and its derivative by using the recurrence

$$\begin{aligned} &p_0(\lambda) = 1, \quad p_1(\lambda) = \alpha_1, \\ &p_k(\lambda) = (\alpha_k - \lambda)p_{k-1}(\lambda) - \beta_{k-1}\gamma_{k-1}p_{k-2}(\lambda), \quad k = 2\colon n, \end{aligned} \tag{2.2}$$

obtained by expanding $\det(T_k - \lambda I_k)$ by its last row. Since this recurrence is known to suffer from overflow and underflow problems [30], we adopt a different strategy.

Assume that λ is not a zero of p, that is, $p(\lambda) \neq 0$. Then

$$\frac{p'(\lambda)}{p(\lambda)} = -\sum_{j=1}^{n} \frac{1}{\lambda_j - \lambda} = -\operatorname{trace}\big((T - \lambda I)^{-1}\big) = -\sum_{j=1}^{n} \theta_j, \tag{2.3}$$

where θ_j is the jth diagonal element of $(T-\lambda I)^{-1}$.

In what follows, S denotes the shifted tridiagonal matrix

$$S := T - \lambda I.$$

If S is unreduced, S^{-1} can be characterized in terms of four vectors, u, v, y, and z [26], such that

$$(S^{-1})_{jk} = \begin{cases} u_j v_k \text{ if } j \le k, \\ y_j z_k \text{ if } j \ge k. \end{cases} \tag{2.4}$$

Note that $u_j v_j = y_j z_j$. These four vectors can be computed in $O(n)$ operations, and it is tempting to use them for the computation of Newton's correction via

$$p(\lambda)/p'(\lambda) = -1/\sum_{j=1}^{n} u_j v_j = -1/\sum_{j=1}^{n} y_j z_j.$$

However, as illustrated in [25], u, v, y, and z can be extremely badly scaled and their computation can break down because of overflow and underflow.

In the next two subsections, we describe two robust and efficient approaches for computing the trace of the inverse of a tridiagonal matrix and discuss our choice.

2.1. Dhillon's approach. Dhillon [11] proposes an algorithm to compute the 1-norm of the inverse of a tridiagonal matrix S in $O(n)$ operations that is more reliable than Higham's algorithm [24] based on the compact representation (2.4). As a by-product, Dhillon's approach provides trace(S^{-1}). His algorithm relies on the computation of the two triangular factorizations

$$S = L_+ D_+ U_+, \qquad S = U_- D_- L_-, \tag{2.5}$$

where L_+ and L_- are unit lower bidiagonal, U_+ and U_- are unit upper bidiagonal, while $D_+ = \mathrm{diag}(d_1^+, \ldots, d_n^+)$ and $D_- = \mathrm{diag}(d_1^-, \ldots, d_n^-)$. If these factorizations exist, the diagonal entries of S^{-1} denoted by θ_j, $j = 1\colon n$, can be expressed in terms of the diagonal factors D_+ and D_- through the recurrence

$$\theta_1 = 1/d_1^-, \qquad \theta_{j+1} = \theta_j \frac{d_j^+}{d_{j+1}^-}, \quad j = 1\colon n-1. \tag{2.6}$$

Note that the triangular factorizations (2.5) may suffer element growth. They can also break down prematurely if a zero pivot is encountered, that is, if $d_j^+ = 0$ or $d_{j+1}^- = 0$ for some j. To overcome this latter drawback, Dhillon [11] makes use of IEEE floating point arithmetic, which permits computations with $\pm\infty$. With this approach, Dhillon's algorithm always returns an approximation of trace(S^{-1}). It is worth pointing out that besides the multiplicative formulae (2.6) for computing θ_j, alternative formulae are introduced in [12] having an additive form which may make them more robust with respect to overflow and underflow.

In our implementation of the Ehrlich–Aberth method the computation of $\det(S)$ is needed at the last stage of the algorithm to provide an error bound for the computed eigenvalues (see section 4). Dhillon's algorithm can be modified to deal with the computation of $\det(S)$ by using a block formulation. Its stability properties are related to those of the (block) LU factorization for tridiagonal matrices.

2.2. A QR factorization approach. In this section we present an alternative algorithm for the computation of $\mathrm{trace}(S^{-1})$ in $O(n)$ operations that is based on the properties of QR factorizations of tridiagonal matrices. Our algorithm keeps the element growth under control and does not have any difficulty caused by overflow and underflow, so there is no need to augment the algorithm with tests for dealing with degenerate cases, as in [11]. In addition, it provides $\det(S)$. Its cost is about twice the cost of the algorithm based on the LU factorization described in section 2.1.

Recall that in our application S is the shifted tridiagonal matrix $T-\lambda I$, where T is given by (2.1). Since λ can be complex, S has real subdiagonal and superdiagonal elements and complex diagonal elements. We denote by G_j the $n\times n$ unitary Givens rotation equal to the identity matrix except in rows and columns j and $j+1$, where

$$G_j([j,j+1],[j,j+1]) = \begin{bmatrix} \phi_j & \psi_j \\ -\psi_j & \overline{\phi}_j \end{bmatrix}, \quad |\phi_j|^2+|\psi_j|^2 = 1.$$

In the following we choose ψ_j real; this choice enables us to reduce the arithmetic complexity of the computation. Let $S=QR$ be the QR factorization of S obtained by means of Givens rotations, so that

$$G_{n-1}\cdots G_2G_1S = R \quad \text{and} \quad Q^* = G_{n-1}\cdots G_2G_1. \tag{2.7}$$

Since S is tridiagonal, R is an upper triangular matrix of the form

$$R = \begin{bmatrix} r_1 & s_1 & t_1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & r_{n-2} & s_{n-2} & t_{n-2} \\ & & & r_{n-1} & s_{n-1} \\ 0 & & & & r_n \end{bmatrix} \in \mathbb{C}^{n\times n}.$$

If $\phi_1 = \overline{\alpha}_1\tau_1$ and $\psi_1 = \beta_1\tau_1$ with $\tau_1 = 1/\sqrt{|\alpha_1|^2+\beta_1^2}$, then

$$S_1 := G_1S = \left[\begin{array}{c|cccc} r_1 & s_1 & t_1 & 0 & \dots \\ \hline 0 & \widetilde{\alpha}_2 & \widetilde{\gamma}_2 & & 0 \\ \vdots & \beta_2 & \alpha_3 & \gamma_3 & \\ \vdots & 0 & \ddots & \ddots & \ddots \end{array}\right]$$

with

$$\begin{aligned} &r_1 = \phi_1\alpha_1+\psi_1\beta_1, \qquad s_1 = \phi_1\gamma_1+\psi_1\alpha_2, \quad t_1 = \psi_1\gamma_2, \\ &\widetilde{\alpha}_2 = -\psi_1\gamma_1+\overline{\phi}_1\alpha_2, \quad \widetilde{\gamma}_2 = \overline{\phi}_1\gamma_2. \end{aligned}$$

Recursively applying the same transformation to the $(n-1)\times(n-1)$ trailing principal submatrix of S_1 yields the factorization (2.7), where

$$\begin{aligned} &\tau_j = 1/\sqrt{|\widetilde{\alpha}_j|^2+\beta_j^2}, && \phi_j = \overline{\widetilde{\alpha}}_j\tau_j, && \psi_j = \beta_j\tau_j, \\ &r_j = \phi_j\widetilde{\alpha}_j+\psi_j\beta_j, && s_j = \phi_j\widetilde{\gamma}_j+\psi_j\alpha_{j+1}, && t_j = \psi_j\gamma_{j+1}, \\ &\widetilde{\alpha}_{j+1} = -\psi_j\widetilde{\gamma}_j+\overline{\phi}_j\alpha_{j+1}, && \widetilde{\gamma}_{j+1} = \overline{\phi}_j\gamma_{j+1} \end{aligned} \tag{2.8}$$

for $j = 1\colon n-1$, with $\widetilde{\alpha}_1 = \alpha_1$ and $\widetilde{\gamma}_1 = \gamma_1$. Note that all the ψ_j are real, and if S is unreduced, the ψ_j are nonzero.

The following result of [19] concerns the semiseparable structure of Q^* and is crucial in the computation of the diagonal entries of S^{-1} in $O(n)$ arithmetic operations.

THEOREM 2.1. *Let $S \in \mathbb{C}^{n\times n}$ be tridiagonal and unreduced and let $S = QR$ be its QR factorization computed according to* (2.8). *Define*

$$\begin{aligned} &D = \operatorname{diag}(1, -\psi_1, \psi_1\psi_2, \ldots, (-1)^{n-1}\psi_1\psi_2\cdots\psi_{n-1}),\\ &u = D^{-1}[1, \overline{\phi}_1, \overline{\phi}_2, \ldots, \overline{\phi}_{n-1}]^T,\\ &v = D[\phi_1, \phi_2, \phi_3, \ldots, \phi_{n-1}, 1]^T. \end{aligned} \tag{2.9}$$

Then

$$Q^* = \begin{bmatrix} v_1u_1 & \psi_1 & & & 0\\ v_2u_1 & v_2u_2 & \psi_2 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \vdots & & & v_{n-1}u_{n-1} & \psi_{n-1}\\ v_nu_1 & v_nu_2 & \cdots & v_nu_{n-1} & v_nu_n \end{bmatrix}.$$

For simplicity, we assume that S is nonsingular so that R^{-1} exists. Let w be the solution of the system $Rw = v$, where v is defined in (2.9). Then, using the structure of Q^* in Theorem 2.1 and the fact that R is triangular, the jth diagonal elements of S^{-1}, θ_j, is given by

$$\theta_j = e_j^*S^{-1}e_j = e_j^*R^{-1}Q^*e_j = u_je_j^*R^{-1}v = u_je_j^*w = u_jw_j,$$

and hence

$$\operatorname{trace}(S^{-1}) = \sum_{j=1}^{n} u_jw_j.$$

Observe that the computation of u and v by means of (2.9) generates underflow and overflow problems: since the diagonal entries of D are products of the ψ_j with $|\psi_j| \le 1$, then for large n, D may have diagonal entries that underflow to zero and inverting D would generate overflow. A way of avoiding this drawback is to scale the system $Rw = v$ with the diagonal matrix D of Theorem 2.1. This yields $\widehat{R}\widehat{w} = \widehat{v}$, where

$$\widehat{R} = D^{-1}RD, \quad \widehat{w} = D^{-1}w, \quad \widehat{v} = D^{-1}v = [\phi_1, \ldots, \phi_{n-1}, 1]^T. \tag{2.10}$$

With this scaling, no accumulation of products of ψ_j is needed. The entries of the matrix $\widehat{R}$ are given by

$$\widehat{r}_j = r_j, \quad \widehat{s}_j = -\psi_j s_j, \quad \widehat{t}_j = \psi_j\psi_{j+1}t_j \tag{2.11}$$

and their computation does not generate overflow since $|\psi_j| \le 1$. Underflow in the computation of $\widehat{s}_j$ and $\widehat{t}_j$ is not a problem since their inverses are not needed in the solution of $\widehat{R}\widehat{w} = \widehat{v}$. The only terms that must be inverted in the computation of $\widehat{w}$ are the diagonal elements of $\widehat{R}$.

We now show that if overflow is encountered in the evaluation of $\widehat{w}$, then R and therefore $S = T - \lambda I$ are numerically singular. In that case we have detected an eigenvalue. First observe that since R is upper triangular and $\widehat{R}^{-1} = D^{-1}R^{-1}D$, the elements of $\widehat{R}^{-1}$ are obtained by multiplying the corresponding elements of R^{-1} by

suitable products of ψ_j. Since $|\psi_j| \le 1$ for any j, we deduce that $|\widehat{R}^{-1}| \le |R^{-1}|$, where the inequality holds componentwise and $|R|$ denotes the matrix with elements $|r_{jk}|$. In this way we find that

$$\|\widehat{R}^{-1}\|_\infty \le \|R^{-1}\|_\infty. \tag{2.12}$$

Second, since $\|\widehat{v}\|_\infty = 1$, we have $\|\widehat{w}\|_\infty \le \|\widehat{R}^{-1}\|_\infty \|\widehat{v}\|_\infty = \|\widehat{R}^{-1}\|_\infty$, which complemented with (2.12) yields $\|\widehat{w}\|_\infty \le \|R^{-1}\|_\infty$. Therefore, overflow in $\widehat{w}$ implies overflow in R^{-1}; that is, $T - \lambda I$ is numerically singular.

Let

$$\widehat{u} = [1, \overline{\phi}_1, \overline{\phi}_2, \ldots, \overline{\phi}_{n-1}]^T. \tag{2.13}$$

Note that because $|\phi_j|^2 + |\psi_j|^2 = 1$, the components of $\widehat{u}$ are all bounded by 1 in modulus. Since $w = D\widehat{w}$ and $u = D^{-1}\widehat{u}$, we find that $u_j w_j = \widehat{u}_j \widehat{w}_j$, $j = 1\colon n$, so that

$$\operatorname{trace}(S^{-1}) = \sum_{j=1}^{n} \widehat{u}_j \widehat{w}_j \tag{2.14}$$

and

$$\det(S) = \prod_{j=1}^{n} r_j. \tag{2.15}$$

The relative error due to cancellation in computing (2.14) is bounded by a multiple of the unit roundoff times

$$|\widehat{u}|^T |\widehat{w}| / |\widehat{u}^T \widehat{w}| := \zeta, \tag{2.16}$$

where $|\widehat{u}|$ and $|\widehat{w}|$ denote the vectors whose components are the moduli of $\widehat{u}$ and $\widehat{w}$. By performing an asymptotic analysis of the order of $|\zeta|$ when $\lambda \to \mu$, where μ is a simple eigenvalue of T with right and left eigenvectors x and y, respectively, we find that

$$(T - \lambda I)^{-1} = (\mu - \lambda)^{-1} y x^T + O(1), \tag{2.17}$$

with $x^T y \neq 0$. Therefore, since the diagonal elements of $S^{-1} = (T - \lambda I)^{-1}$ are $\widehat{u}_j \widehat{w}_j$, $j = 1\colon n$, one has $\widehat{u}_j \widehat{w}_j = (\mu - \lambda)^{-1} x_j y_j + O(1)$. This implies that $|\widehat{u}|^T |\widehat{w}| = |\mu - \lambda|^{-1} |y|^T |x| + O(1)$ and $\widehat{u}^T \widehat{w} = (\mu - \lambda)^{-1} y^T x + O(1)$ so that

$$|\widehat{u}|^T |\widehat{w}| / |\widehat{u}^T \widehat{w}| = |y|^T |x| / |y^T x| + O(|\mu - \lambda|). \tag{2.18}$$

Recall that for ϵ in a neighborhood of 0, the matrix $T + \epsilon F$ has the eigenvalue $\mu + \epsilon\, y^T F x / (y^T x) + O(\epsilon^2)$ (see, for instance, [20]).

Moreover, if $F = \operatorname{diag}(f_i)$ with $|f_i| = 1$ such that $y_i f_i x_i$ is real nonnegative, then $\epsilon y^T F x / (y^T x) = \epsilon |y|^T |x| / (y^T x)$; that is, the ratio $|y|^T |x| / |y^T x|$ is the condition number of the eigenvalue μ with respect to a particular diagonal perturbation. Therefore, if the values $\widehat{u}_i$ and $\widehat{w}_i$ are computed in a numerically stable way, then the accuracy of the computed value of $\sum_{j=1}^{n} \widehat{u}_j \widehat{w}_j$ is consistent with the conditioning of the eigenvalue. We can perform a similar asymptotic analysis when the multiplicity ν of μ is greater than 1, since $(T - \lambda I)^{-1} = (\mu - \lambda)^{-\nu} x y^T + O((\mu - \lambda)^{-\nu+1})$. However, in this

case the condition $x^T y \neq 0$ is not generally satisfied. Concerning the accuracy of the computation of the diagonal elements of R, we refer the reader to section 4.1.

The analysis based on (2.18) is a "first order" analysis and keeps its validity only from the asymptotic point of view as $\lambda \to \mu$. Indeed, for a finite value of λ this analysis may lose its validity if, say, $|x^T y|$ is much smaller than $|\mu - \lambda|$ so that the $O(|\mu - \lambda|)$ term in (2.18) is no longer negligible. On the other hand, this analysis is useful for designing a heuristic stopping criterion.

Equations (2.8) and (2.10)–(2.14) constitute our algorithm for the computation of $p'(\lambda)/p(\lambda) = \mathrm{trace}\big((T - \lambda I)^{-1}\big) = \mathrm{trace}(S^{-1})$, which we summarize below in pseudocode. The function *Givens* constructs ϕ_j and ψ_j and guards against the risk of overflow. We refer to Bindel et al. [3] for a detailed explanation on how this function should be implemented.

function $\tau = \mathrm{trace\_Tinv}(\beta, \alpha, \gamma)$
% Compute $\tau = \mathrm{trace}(S^{-1})$, where $S = \mathrm{tridiag}(\beta, \alpha, \gamma)$ is $n \times n$ tridiagonal
% with real off-diagonals and complex diagonal.
$a = \alpha_1$, $g = \gamma_1$, $u_1 = 1$
% Computes vectors $\widehat{r}$, $\widehat{s}$, $\widehat{t}$ in (2.11), $\widehat{u}$ in (2.13) and $\widehat{v}$ in (2.10).
for $j = 1 : n-1$
    $(\phi, \psi) =$ Givens (a, β_j)
    $r_j = \phi a + \psi \beta_j$, $s_j = -\psi(\phi g + \psi \alpha_{j+1})$
    $a = -\psi g + \overline{\phi} \alpha_{j+1}$, $u_{j+1} = \overline{\phi}$, $v_j = \phi$
    if $j < n-1$, $t_j = \psi^2 \gamma_{j+1}$, $g = \overline{\phi} \gamma_{j+1}$, end
    if $j > 1$, $t_{j-1} = \psi t_{j-1}$, end
end
$r_n = a$, $v_n = 1$
% Solve the linear system $\widehat{R}\widehat{w} = \widehat{v}$.
$w_n = v_n / r_n$
$w_{n-1} = (v_{n-1} - w_n s_{n-1})/r_{n-1}$
for $j = n-2 : -1 : 1$
    $w_j = (v_j - w_{j+1} s_j - w_{j+2} t_j)/r_j$
    if $w_j =$ Inf, $\tau =$ Inf, return, end
end
$\tau = \sum_{j=1}^{n} u_j w_j$

The function trace_Tinv requires $O(n)$ operations.

3. Choosing initial approximations. The choice of the initial approximations $z_j^{(0)}$, $j = 1{:}\,n$, used to start the Ehrlich–Aberth iteration (1.1) crucially affects the number of steps needed by the method. For polynomials expressed in terms of their coefficients, Bini [4] derived a criterion for selecting initial guesses based on a combination of Rouché's theorem and the use of the Newton polygon. Here we follow a divide-and-conquer strategy which better exploits the tridiagonal nature of the problem and seems to perform well in practice. More precisely, the initial approximations are chosen from the eigenvalues of two suitable tridiagonal submatrices of order roughly $n/2$.

Rewrite T as

$$T = \widetilde{T} + uv^T, \tag{3.1}$$

where

$$\widetilde{T} = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} = T_1 \oplus T_2, \quad u = e_m + e_{m+1}, \quad v = \beta_m e_m + \gamma_m e_{m+1} \tag{3.2}$$

and

$$T_1 = \begin{bmatrix} \alpha_1 & \gamma_1 & & 0 \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \gamma_{m-1} \\ 0 & & \beta_{m-1} & \alpha_m - \beta_m \end{bmatrix}, \qquad T_2 = \begin{bmatrix} \alpha_{m+1} - \gamma_m & \gamma_{m+1} & & 0 \\ \beta_{m+1} & \alpha_{m+2} & \ddots & \\ & \ddots & \ddots & \gamma_{n-1} \\ 0 & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

There are no obvious connections between the eigenvalues of T and those of $\widetilde{T}$, unlike in the symmetric case, in which the eigenvalues of T interlace those of $\widetilde{T}$. However, in this section, we show that the eigenvalues of $\widetilde{T}$ generally provide good starting points for the Ehrlich–Aberth iteration for $p(\lambda) = \det(T - \lambda I)$.

First we consider the case where T_1 and T_2 have a common eigenvalue and then the case where either T_1 or T_2 has a multiple eigenvalue. In this regard, we observe that if a matrix $\widetilde{A}$ has an eigenvalue λ of geometric multiplicity at least 2, then λ is an eigenvalue of $A = \widetilde{A} + uv^T$ for any vectors u, v. Indeed, denote by x_1 and x_2 two linearly independent eigenvectors of $\widetilde{A}$ corresponding to λ and set $y = \nu_1 x_1 + \nu_2 x_2$; then it holds that $Ay = \lambda y + u(v^T y)$. Therefore, choosing ν_1 and ν_2 such that $\nu_1 v^T x_1 + \nu_2 v^T x_2 = 0$ with $\nu_1 \neq 0$ or $\nu_2 \neq 0$ yields $Ay = \lambda y$, $y \neq 0$. This implies the following result.

THEOREM 3.1. *If λ is a common eigenvalue of T_1 and T_2, or a multiple eigenvalue of T_j, $j = 1$ or 2, with geometric multiplicity at least 2, then λ is an eigenvalue of $T = (T_1 \oplus T_2) + uv^T$ for any vectors u and v.*

Proof. Observe that under these assumptions λ is an eigenvalue of $T_1 \oplus T_2$ with geometric multiplicity at least 2. Therefore λ is an eigenvalue of $T_1 \oplus T_2 + uv^T$ for any vector u, v. ∎

Our next theorem relates the eigenvalues of T_1 and T_2 with those of $T_1 \oplus T_2 + uv^T$ and relies on the following lemma from Henrici [22].

LEMMA 3.2. *Let $p(\lambda)$ be a polynomial of degree n in λ, and let z be any complex number. Then if $p'(z) \neq 0$, the disk of center z and radius $n|p(z)/p'(z)|$ contains at least one zero of $p(\lambda)$.*

THEOREM 3.3. *Assume that $T_1 \in \mathbb{R}^{m\times m}$ and $T_2 \in \mathbb{R}^{(n-m)\times(n-m)}$ are both diagonalizable, that is, there exist X_1, X_2 nonsingular such that $T_1 = X_1 D_1 X_1^{-1}$ and $T_2 = X_2 D_2 X_2^{-1}$, with $D_1 = \operatorname{diag}(d_1, d_2, \ldots, d_m)$ and $D_2 = \operatorname{diag}(d_{m+1}, d_{m+2}, \ldots, d_n)$. Let $\beta_m, \gamma_m \in \mathbb{R}$ and*

$$\eta_j = \begin{cases} \beta_m (e_j^T X_1^{-1} e_m)(e_m^T X_1 e_j) & \text{if } j \leq m, \\ \gamma_m (e_{j-m}^T X_2^{-1} e_1)(e_1^T X_2 e_{j-m}) & \text{if } j > m. \end{cases} \tag{3.3}$$

Then, in any disk of center d_j and radius

$$\rho_j = \begin{cases} n|\eta_j| \Big/ \left| 1 + \sum_{\substack{k=1 \\ k\neq j}}^{n} \frac{\eta_j + \eta_k}{d_k - d_j} \right| & \text{if } d_j \neq d_k \text{ for } k \neq j, \\ 0 & \text{if } d_j = d_k \text{ for some } k \neq j, \end{cases}$$

there exists an eigenvalue of $T = (T_1 \oplus T_2) + uv^T$, where $u = e_m + e_{m+1}$ and $v = \beta_m e_m + \gamma_m e_{m+1}$.

Proof. If $d_j = d_k$ for some $k \neq j$, then the result holds from Theorem 3.1. Consider the case $d_j \neq d_k$ for $k \neq j$. Let $\widetilde{T} = T_1 \oplus T_2$. We have

$$T - \lambda I = (\widetilde{T} - \lambda I)(I + (\widetilde{T} - \lambda I)^{-1} u v^T),$$

and taking determinants on both sides of the equation gives

$$p(\lambda) = \prod_{j=1}^{n} (d_j - \lambda)(1 + v^T (\widetilde{T} - \lambda I)^{-1} u).$$

Using the eigendecomposition of T_1 and T_2 and the definition of u and v, the expression for $p(\lambda)$ simplifies to

$$p(\lambda) = \prod_{j=1}^{n} (d_j - \lambda) + \sum_{j=1}^{n} \eta_j \prod_{k=1, k\neq j}^{n} (d_k - \lambda). \tag{3.4}$$

Thus

$$p(d_j) = \eta_j \prod_{\substack{k=1 \\ k\neq j}}^{n} (d_k - d_j),$$

and

$$p'(\lambda) = -\sum_{k=1}^{n} \prod_{\substack{h=1 \\ h\neq k}}^{n} (d_h - \lambda) - \sum_{\ell=1}^{n} \eta_\ell \left(\sum_{\substack{k=1 \\ k\neq \ell}}^{n} \prod_{\substack{h=1 \\ h\neq \ell, k}}^{n} (d_h - \lambda) \right)$$

so that

$$p'(d_j) = -\prod_{\substack{k=1 \\ k\neq j}}^{n} (d_k - d_j) \left(1 + \sum_{\substack{k=1 \\ k\neq j}}^{n} \frac{\eta_j + \eta_k}{d_k - d_j} \right).$$

Since $p'(d_j) \neq 0$, applying Lemma 3.2 completes the proof. ∎

According to Theorem 3.3, a small value for ρ_j indicates that there is an eigenvalue of $T_1 \oplus T_2$ close to an eigenvalue of T. An important question related to the effectiveness of using the eigenvalues of T_1 and T_2 as initial approximations for starting the Ehrlich–Aberth iteration is whether the number of "large" values for ρ_j is small or not. Note that if $j \leq m$, then η_j is a multiple of the product of the last components of the jth right and left eigenvectors of T_1 and, if $j > m$, then η_j is a multiple of the product of the first components of the $(j-m)$th left and right eigenvectors of T_2. If T is unreduced, then η_j is nonzero since the eigenvectors of unreduced tridiagonal matrices cannot have a 0 in the first and last components. We report in Table 3.1 ranges of values for ρ_j and $|\eta_j|$ obtained from 1000 randomly generated tridiagonal matrices T of size $n = 100$. The table shows that in more than 80% of the cases, $|\eta_j|$ and ρ_j are smaller than 10^{-4}. Note that the denominator $|1 + \sum_{k=1, k\neq j}^{n} \frac{\eta_j + \eta_k}{d_k - d_j}|$ in the definition of ρ_j does not seem to play an important role. The probability that for almost all the values of j this denominator is close to zero seems to be small. These experiments suggest that most eigenvalues of $T_1 \oplus T_2$ should be good initial values for the Ehrlich–Aberth iteration for $p(\lambda) = \det(T - \lambda I)$.

TABLE 3.1
Ranges of values for ρ_j and $|\eta_j|$ obtained from 1000 randomly generated tridiagonal matrices T of size $n = 100$.

%	$\le 10^{-16}$	$\le 10^{-12}$	$\le 10^{-8}$	$\le 10^{-4}$
ρ_j	48	60	71	82
$\lvert\eta_j\rvert$	54	65	77	88

From the results of this section we propose the following divide-and-conquer strategy to compute initial approximations for the Ehrlich–Aberth iterations. The matrix T is recursively split according to the rank-1 tearing in (3.1)–(3.2) until 2×2 or 1×1 subproblems are reached. The Ehrlich–Aberth iteration is then used to glue back the subproblems using the previously computed eigenvalues as starting guesses for the iterations.

Remark 3.4. A similar divide-and-conquer strategy can be obtained by choosing, as initial approximations, the eigenvalues of the leading principal $m \times m$ submatrix T_1 of T and of the trailing principal $(n-m) \times (n-m)$ submatrix T_2 of T for $m = \lceil n/2 \rceil$. These matrices are obtained by zeroing the entries in positions $(m, m+1)$ and $(m+1, m)$ of T which correspond to applying a rank-2 correction to the matrix T. An analysis similar to the one performed for the rank-1 tearing can be carried out. In all the numerical experiments we performed so far no substantial difference in the performance of the two strategies was observed.

4. The algorithm. In this section we describe an implementation of the Ehrlich–Aberth iteration where the choice of the initial approximations is performed by means of a divide-and-conquer strategy. Then we provide running error bounds needed for the validation of the computed approximations and discuss the computation of the eigenvectors.

The main algorithm for eigenvalue approximation is described below in pseudocode. This implementation follows section 3. A different implementation can be based on the rank-2 tearing of Remark 3.4, where the initial approximations are the eigenvalues of the principal submatrices obtained by zeroing the entries in positions $(m, m+1)$ and $(m+1, m)$, where $m = \lfloor n/2 \rfloor$.

First we report on the recursive part of the algorithm and then the main refinement engine, i.e., the Ehrlich–Aberth iteration.

function $z = \text{eigen}(\beta, \alpha, \gamma)$
% Computes the eigenvalues of the $n \times n$ tridiagonal matrix $T = \text{tridiag}(\beta, \alpha, \gamma)$
% *perturb* is a small scalar set to the maximum relative perturbation
% of intermediate eigenvalues
if $n = 1$, $z = \alpha_1$, return, end
if $n = 2$, set z to the eigenvalues of $\left[\begin{smallmatrix} \alpha_1 & \gamma_1 \\ \beta_1 & \alpha_2 \end{smallmatrix}\right]$, return, end
% Recursive stage
$i = \sqrt{-1}$
if $n > 2$
    $m = \lfloor n/2 \rfloor$
    $\alpha' = \alpha(1:m)$, $\alpha'_m = \alpha'_m - \beta_m$
    $z(1{:}\,m) = \text{eigen}\big(\beta(1:m-1), \alpha', \gamma(1:m-1)\big)$
    $\alpha'' = \alpha(m+1:n)$, $\alpha''_1 = \alpha''_1 - \gamma_m$
    $z(m+1:n) = \text{eigen}\big(\beta(m+1:n-1), \alpha'', \gamma(m+1:n-1)\big)$
    Choose random ρ such that $perturb/2 < \rho < perturb$

```
        z(1 : m) = (1 + iρ) ∗ z(1 : m)
        z(m + 1 : n) = (1 − iρ) ∗ z(m + 1 : n)
        % Refine the current approximations
        z = Aberth(β, α, γ, z)
    end
```

The role of *perturb* is to avoid different approximations collapsing to a single value. In fact, if $z_j = z_k$, with $j \neq k$, then the Ehrlich–Aberth iteration cannot be applied. Moreover, since in practice the Ehrlich–Aberth iteration performs better if the intermediate approximations are in the complex field, we have chosen a pure imaginary value as perturbation. In our code we set the parameter *perturb* to 10 times the machine precision.

The pseudocode for the Ehrlich–Aberth iteration is reported below.

```
function z = Aberth(β, α, γ, z)
% Refine the n approximations z = (z_j), j = 1 : n to the eigenvalues of
% T = tridiag(β, α, γ) by means of the Ehrlich–Aberth iteration.
% maxit is the maximum number of iterations,
% tol is a small quantity used for the convergence criteria.
it = 0
ζ = zeros(n, 1) % ζ_j = 1 if z_j has converged to an eigenvalue and 0 otherwise.
while (Σ_{j=1}^{n} ζ_j < n & it < maxit)
    it = it + 1
    for j = 1 : n
        if ζ_j = 0
            τ = trace_Tinv(β, α − z_j, γ)
            if τ =Inf
                nwtc = 0
            else
                nwtc = −1/τ
            end
            if |nwtc| < tol‖T − z_j I‖_∞, ζ_j = 1, end
            z_j = z_j − nwtc/(1 − nwtc Σ_{k=1,k≠j}^{n} 1/(z_j − z_k))
        end
    end
end
```

Observe that at the general itth sweep, the Ehrlich–Aberth correction is applied only to the approximations z_j which have not yet converged. This makes the cost of each sweep depend on the number of approximated eigenvalues. For this reason, in order to evaluate the complexity of the algorithm, it is convenient to introduce the *average number of iterations per eigenvalue* μ given by the overall number of Ehrlich–Aberth iterations applied to each eigenvalue divided by n. For example, if $n = 4$ and the number of iterations needed for computing $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ is 5, 10, 14, 21, respectively, then $\mu = 50/4 = 12.5$.

4.1. Running error bound. In this section we derive a running error bound for the error in the computed eigenvalues λ_ℓ, $\ell = 1{:}\, n$. Our bounds are based on the following result of Carstensen [9].

LEMMA 4.1. *Let $p(\lambda)$ be a monic polynomial of degree n in λ, and let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be pairwise distinct complex numbers. Denote by $\mathcal{D}(\lambda_\ell, \rho_\ell)$ the disk of center λ_ℓ and*

radius

$$\rho_\ell = \frac{n|p(\lambda_\ell)|}{\left|\prod_{\substack{j=1\\ j\neq\ell}}^{n}(\lambda_\ell - \lambda_j)\right|}.$$

Then $\mathcal{U} = \bigcup_\ell \mathcal{D}(\lambda_\ell, \rho_\ell)$ contains all the zeros of $p(\lambda)$. Moreover, any connected component of $\mathcal{U}$ made up of k disks contains k zeros. In particular, any isolated disk contains a single zero.

The set of disks $\{\mathcal{D}(\lambda_\ell, \rho_\ell),\ \ell = 1\colon n\}$ defined in the above lemma is called a *set of inclusion disks*. Note that because of rounding errors in the computation of $|p(\lambda_\ell)|$ and $\prod_{j=1,\ j\neq\ell}^{n}(\lambda_\ell - \lambda_j)$, the computed ρ_ℓ denoted by $\widehat{\rho}_\ell$ may be inaccurate. Thus, the disks $\mathcal{D}(\lambda_\ell, \widehat{\rho}_\ell)$ may not provide a set of inclusion disks.

Consider the standard model of floating point arithmetic [25, section 2.2],

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \qquad |\delta| \le u, \quad \operatorname{op} = +, -, *, /, \tag{4.1}$$

where u is the unit roundoff. Since λ can be complex, part of the computation is carried out in complex arithmetic. Under the standard model (4.1) (see [25, Lemma 3.5]),

$$\begin{aligned} fl(x \pm y) &= (x \pm y)(1+\delta), \quad |\delta| \le u, \\ fl(xy) &= xy(1+\delta), \quad |\delta| \le 2\sqrt{2}u, \\ fl(x/y) &= (x/y)(1+\delta), \quad |\delta| \le 4\sqrt{2}u, \end{aligned} \tag{4.2}$$

where we ignore second order terms in u.

Suppose we can compute an upper bound Δp_ℓ for the error $\big||\hat{p}_\ell| - |p(\lambda_\ell)|\big|$, where $|\hat{p}_\ell| = fl(|p(\lambda_\ell)|)$. Then from Lemma 4.1 we immediately deduce that the disk $\mathcal{D}(\lambda_\ell, \widetilde{\rho}_\ell)$, where

$$\widetilde{\rho}_\ell = (1+2nu)\frac{n(|\hat{p}_\ell| + \Delta p_\ell)}{\hat{\pi}_\ell}, \qquad \hat{\pi}_\ell = fl\left(\left|\prod_{\substack{j=1\\ j\neq\ell}}^{n}(\lambda_\ell - \lambda_j)\right|\right), \tag{4.3}$$

is such that $\mathcal{D}(\lambda_\ell, \rho_\ell) \subset \mathcal{D}_\ell(\lambda_\ell, \widetilde{\rho}_\ell)$, $\ell = 1\colon n$; therefore the set $\{\mathcal{D}(\lambda_\ell, \widetilde{\rho}_\ell),\ \ell = 1\colon n\}$ is a set of inclusion disks.

The main difficulty is in determining Δp_ℓ. Recall that $|p(\lambda_\ell)| = \prod_{j=1}^{n}|r_j|$, where r_j is the jth diagonal element of R in the QR factorization of $T - \lambda_\ell I$. Let $fl(r_j) = r_j + \delta r_j$. Then

$$fl\left(\prod_{j=1}^{n} r_j\right) = \prod_{j=1}^{n}(r_j + \delta r_j) \cdot \prod_{j=1}^{n-1}(1+\epsilon_j), \quad |\epsilon_j| \le 2\sqrt{2}u,$$

and, if we ignore the second order terms in u, we have $|\widehat{p}_\ell| = |p(\lambda_\ell)| + \delta p_\ell$, with

$$|\delta p_\ell| \le |\widehat{p}_\ell|\left(\sum_{j=1}^{n} \Delta r_j/|\widehat{r}_j| + (n-1)2\sqrt{2}u\right) =: \Delta p_\ell. \tag{4.4}$$

The Δr_j are computed along with the $\widehat{r}_j = fl(r_j)$ thanks to a systematic running error analysis of all the quantities involved in the calculation of r_j. For that we make use of the following lemma.

LEMMA 4.2. *Let $x = yz + \alpha v \in \mathbb{C}$, where α is given data, $y = \widehat{y} + \delta y$ with $|\delta y| \le \Delta y$, $z = \widehat{z} + \delta z$ with $|\delta z| \le \Delta z$, $v = \widehat{v} + \delta v$ with $|\delta v| \le \Delta v$, and $\widehat{y}, \widehat{z}, \widehat{v}, \Delta y, \Delta z, \Delta v$ are known computed quantities. Then $\widehat{x} = fl(x) = x + \delta x$ with*

$$|\delta x| \le \Delta x := |\widehat{y}|\Delta z + |\widehat{z}|\Delta y + 2\sqrt{2}u|\widehat{y}|\,|\widehat{z}| + |\alpha|\Delta v + 2\sqrt{2}u|\alpha|\,|\widehat{v}|.$$

Proof. The proof is a straightforward application of (4.1) and (4.2). ∎

We notice that in the function trace_Tinv (see section 2.2) all the quantities used to compute r_i can be rewritten in the form $x = yz + \alpha v$. Hence, assuming that the function Givens returns $\Delta\phi$ and $\Delta\psi$, we can add the lines

$$\begin{aligned}
\Delta r_j &= |\phi|\Delta a + |a|\Delta\phi + 2\sqrt{2}u|a|\,|\phi| + |\beta_j|\Delta\psi + 2\sqrt{2}u|\beta_j|\,|\psi|,\\
\Delta a &= |\psi|\Delta g + |g|\Delta\psi + 2\sqrt{2}u|\psi|\,|g| + |\alpha_{j+1}|\Delta\phi + 2\sqrt{2}u|\alpha_{j+1}|\,|\phi|,\\
\Delta g &= |\gamma_{j+1}|\Delta\phi + 2\sqrt{2}u|\gamma_{j+1}|\,|\phi|
\end{aligned}$$

to the function trace_Tinv after the computation of r_j, a, and g, respectively. The two quantities Δa and Δg are initially set to zero and $\Delta r_n = \Delta a$. The error bound Δp_ℓ is then obtained using (4.4).

4.2. Computing the eigenvectors. One of the most convenient methods for approximating an eigenvector of T once we are given an approximation λ of the corresponding eigenvalue is the inverse power iteration applied to the matrix $T - \lambda I$. A crucial computational issue is to determine a suitable initial guess $v^{(0)}$ for the eigenvector in order to start the iteration:

$$\begin{aligned}
&(T - \lambda I)w^{(j+1)} = v^{(j)},\\
&v^{(j+1)} = w^{(j+1)}/\|w^{(j+1)}\|,
\end{aligned} \qquad j = 0, 1, 2, \ldots.$$

This problem has been studied in several recent papers [11], [16], [32]. In particular, in [16] a strategy is described for the choice of $v^{(0)}$ which relies on the evaluation of the index k of the entry of maximum modulus in the main diagonal of $(T - \lambda I)^{-1}$. Our algorithm for the approximation of the eigenvalues provides, as a by-product, the diagonal entries of $(T - \lambda I)^{-1}$. Therefore the value of k is determined at no cost. Moreover, the QR factorization of the matrix $T - \lambda I$ that is computed by our algorithm can be used for performing each inverse power iteration without any significant additional cost.

5. Numerical experiments. We have implemented the algorithm for the approximation of the eigenvalues of the tridiagonal matrix T in Fortran 95. The code, organized as a Fortran 95 module, can be downloaded from http://www.dm.unipi.it/~bini/eigen_v1.1.tgz. The tests were performed on an Athlon 1800 with IEEE double precision arithmetic, where the code has been compiled with the Lahey/Fujitsu compiler (release L6.20a). In what follows, `eigen` refers to our subroutine implementing the Ehrlich–Aberth iteration.

We first consider the following test matrices, which we describe in the factored form

$$T = D^{-1}\text{tridiag}(1, \alpha, 1), \quad D = \text{diag}(\delta), \quad \alpha, \delta \in \mathbb{R}^n. \tag{5.1}$$

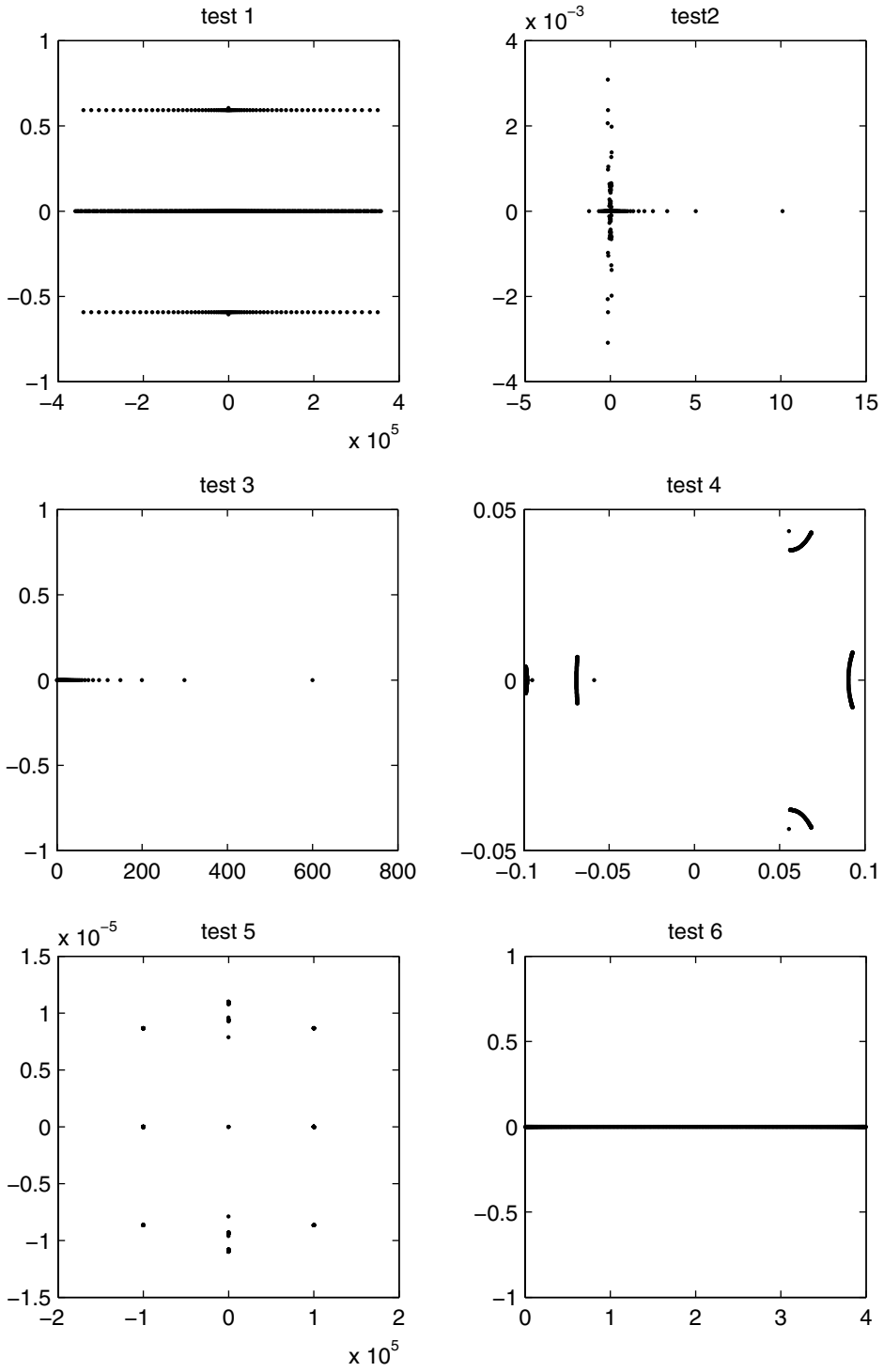

FIG. 5.1. *Eigenvalues of the test matrices* 1–6 *for* $n = 300$.

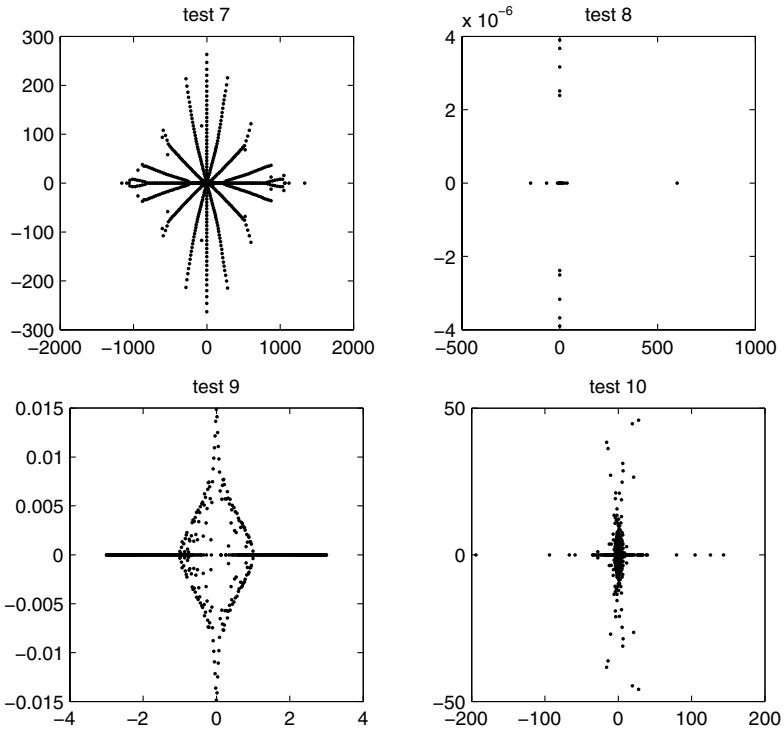

FIG. 5.2. *Eigenvalues of the test matrices* 7–10 *for* $n = 300$.

$$
\begin{aligned}
&\text{Test 1:} \quad \alpha_k = k(-1)^{\lfloor k/8 \rfloor}, \delta_k = (-1)^k/k, k = 1\colon n.\\
&\text{Test 2:} \quad \alpha_k = 10 \cdot (-1)^{\lfloor k/8 \rfloor}, \delta_k = k \cdot (-1)^{\lfloor k/9 \rfloor}, k = 1\colon n.\\
&\text{Test 3:} \quad \alpha_k = k, \delta_k = n - k + 1, k = 1\colon n.\\
&\text{Test 4:} \quad \alpha_k = (-1)^k, \delta_k = 20 \cdot (-1)^{\lfloor k/5 \rfloor}, k = 1\colon n.\\
&\text{Test 5:} \quad \alpha_k = 10^{5(-1)^k} \cdot (-1)^{\lfloor k/4 \rfloor}, \delta_k = (-1)^{\lfloor k/3 \rfloor}, k = 1\colon n.\\
&\text{Test 6:} \quad \alpha_k = 2, \delta_k = 1, k = 1\colon n.\\
&\text{Test 7:} \quad \alpha_k = \frac{1}{k} + \frac{1}{n-k+1}, \delta_k = \frac{1}{k}(-1)^{\lfloor k/9 \rfloor}, k = 1\colon n.\\
&\text{Test 8:} \quad \alpha_k = k \cdot (-1)^{\lfloor k/13 \rfloor + \lfloor k/5 \rfloor}, \delta_k = (n-k+1)^2 \cdot (-1)^{\lfloor k/11 \rfloor}, k = 1\colon n.\\
&\text{Test 9:} \quad \alpha_k = 1, k = 1\colon n;\ \delta_k = 1 \text{ if } k < n/2,\ \delta_k = -1 \text{ if } k \geq n/2.\\
&\text{Test 10:} \quad \alpha_k \text{ and } \delta_k \text{ take random values uniformly distributed in } [-0.5, 0.5].
\end{aligned}
\tag{5.2}
$$

The eigenvalues of these test matrices have a variety of distributions, as illustrated in Figures 5.1 and 5.2 for $n = 300$. In particular, the eigenvalues in Test 4 are distributed along curves and, in Test 7, along lines from the origin; in Test 5 the eigenvalues are distributed in tight clusters with very different moduli.

Let

$$
\kappa(\lambda) = \frac{\|x\|_2 \|y\|_2 \|T\|_2}{|\lambda|\, |y^T x|} \tag{5.3}
$$

TABLE 5.1
Largest eigenvalue condition number $\kappa_{\max}$ and largest relative error for the eigenvalues of the test matrices 1–9 computed with `dhseqr` *and* `eigen` *($n = 100$).*

Test matrices		1	2	3	4	5	6	7	8	9
$\kappa_{\max} = \max_i \kappa(\lambda_i)$		3e4	240	4e4	2.2	2e10	4e3	2e3	4e6	213
$\max_i \frac{\lvert\lambda_i - \hat{\lambda}_i\rvert}{\lvert\lambda_i\rvert}$	dhseqr	2e-13	6e-15	5e-15	4e-15	3e-6	3e-13	3e-14	7e-15	1e-14
	eigen	3e-16	2e-16	2e-16	2e-16	1e-10	2e-14	6e-16	5e-16	2e-15

TABLE 5.2
Condition numbers and relative errors for the three clusters of eigenvalues in Test 5, with $n = 20$.

		$\max_i \lvert\lambda_i - \hat{\lambda}_i\rvert/\lvert\lambda_i\rvert$	
	$\kappa(\lambda)$	dhseqr	eigen
$\lambda \approx -10^5$	1.2	3e-16	8e-18
$\lambda \approx 10^5$	2e2	1e-13	1e-14
$\lvert\lambda\rvert \approx 10^{-5}$	1e10	4e-7	1e-16

be the 2-norm condition number of λ with corresponding right eigenvector x and left eigenvector y. Note that $\kappa(\lambda) \geq 1$ for symmetric matrices. As shown in Table 5.1 the test matrices 1–9 have eigenvalue condition numbers varying from small to large.

To validate our code, we report in Table 5.1 the size of the largest relative error

$$\mathrm{err}(\lambda_i) = |\lambda_i - \hat{\lambda}_i|/|\lambda_i|. \tag{5.4}$$

Here λ_i is computed in quadruple precision and $\hat{\lambda}_i$ denotes the computed eigenvalue with either our subroutine `eigen` or with the LAPACK subroutine `dhseqr` that implements the QR algorithm for computing the eigenvalues of an upper Hessenberg matrix. For these test matrices the eigenvalues computed by `eigen` are, in general, more accurate than those computed by `dhseqr` and do not seem to be affected as much by large eigenvalue condition numbers. Also, `eigen` tends to compute eigenvalues with small moduli more accurately than `dhseqr`. This is better illustrated in Table 5.2 for the matrix in Test 5 with $n = 20$. This matrix has three groups of clustered eigenvalues: two around $\pm 10^5$ and one around 0. Both `dhseqr` and `eigen` compute the eigenvalues around -10^5 to high accuracy. For the eigenvalues with small modulus, the LAPACK routine yields relative errors as large as 10^{-7}, whereas `eigen` computes eigenvalues with relative errors of the order of the machine precision.

The Clement matrix $T = \mathrm{tridiag}(\beta, 0, \gamma)$ with $\beta_j = \gamma_{n-j}$ and $\gamma_j = j$, $j = 1\colon n-1$ [10], has eigenvalues plus and minus the numbers $n-1, n-3, \ldots, 1$ for n even and $n-1$, $n-3, \ldots, 0$ for n odd. We take $n = 50$. The condition number $\kappa(\lambda)$ in (5.3) is small at the ends of the spectrum and large in the middle, as illustrated in the left-hand plot in Figure 5.3. This explains the relative errors in the right-hand plot. On these plots, the dotted lines mark the machine precision. Note that all the eigenvalues computed by `eigen` have a relative error around 10^{-16} or below, whereas those computed by `dhseqr` all have a relative error significantly above 10^{-16}. In this case, our algorithm computes exactly the real part of the eigenvalues, which are integers. The imaginary part is approximated with an absolute error which is less than 10^{-25} for almost all the eigenvalues. Note that the Clement matrices are essentially symmetric; i.e., they can be transformed by a similarity to a symmetric form by means of diagonal scalings

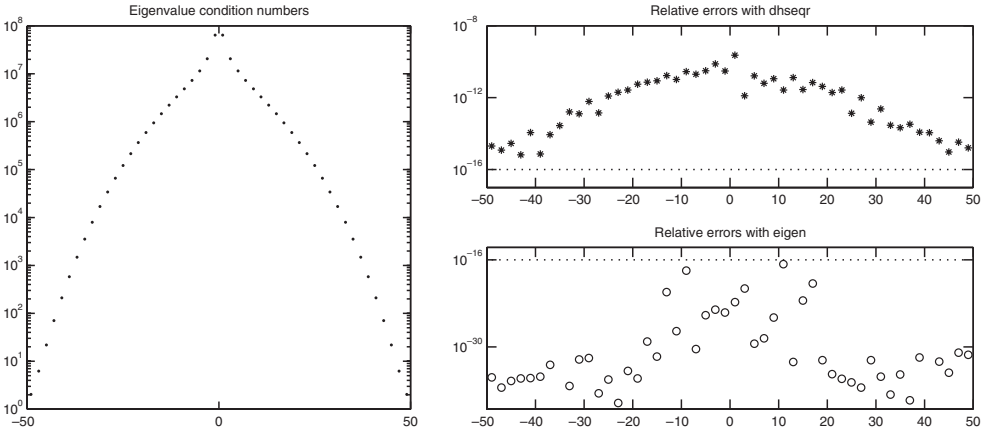

FIG. 5.3. *Clement matrix of size $n = 50$; on the x-abscissa are the n eigenvalues* $\{-49, -47, \ldots, -1, 1, \ldots, 47, 49\}$.

since $\beta_i\gamma_i \geq 0$. The eigenvalues of the symmetric matrices obtained in this way can be determined to high relative accuracy [28].

The Bessel matrices associated with generalized Bessel polynomials [33] are non-symmetric tridiagonal matrices defined by $T_{(a,b)} = \text{tridiag}(\beta, \alpha, \gamma)$ with

$$\alpha_1 = -\frac{b}{a}, \quad \beta_1 = \frac{\gamma_1}{a+1}, \quad \gamma_1 = -\beta_1,$$
$$\alpha_j = -b\frac{a-2}{(2j+a-2)(2j+a-4)}, \quad j = 2\colon n,$$
$$\beta_j = -b\frac{j}{(2j+a-1)(2j+a-2)}, \quad \gamma_j = b\frac{j+a-2}{(2j+a-2)(2j+a-3)}, \quad j = 2\colon n-1.$$

The spectrum is given in Figure 5.4 for $a = -8.5$, $b = 2$, and $n = 18$. These matrices are well known to have ill-conditioned eigenvalues. For this particular example, $\kappa(\lambda)$ ranges from 10^{10} to 10^{15}. The approximations provided by `eigen` are more accurate than those provided by `dhseqr` by a factor of 10 on average.

We tested Liu's tridiagonal pseudosymmetric matrices $T_n = \text{tridiag}(1, \alpha^n, \gamma^n)$ for $n = 14$ and $n = 28$ defined by [29]

$$\alpha^{14} = [0,0,0,0,0,0,-1,-1,0,0,0,0,0,0]^T,$$
$$\gamma^{14} = [-1,1,1,-1,1,-1,-1,-1,1,-1,1,1,-1]^T$$

and

$$\alpha^{28} = [0,0,0,0,0,0,-1,1,0,0,0,0,0,-1,1,0,0,0,0,0,1,-1,0,0,0,0,0,0]^T,$$
$$\gamma^{28} = [-1,1,1,-1,1,-1,-1,-1,1,-1,1,1,-1,-1,-1,$$
$$1,1,-1,1,-1,-1,-1,1,-1,1,1,-1]^T.$$

Both matrices have only one zero eigenvalue of multiplicity 14 and 28, respectively. As shown in Figure 5.5, the accuracy of the approximations delivered by `eigen` are as good as those provided by `dhseqr`.

Summarizing the results concerning the approximation errors, we may say that in all our tests `eigen` generally yields errors no larger, and sometimes much smaller,

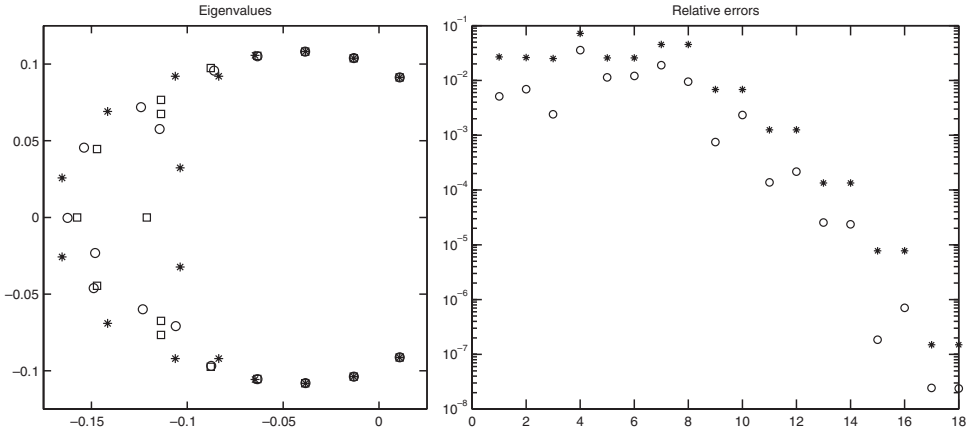

FIG. 5.4. *Bessel matrix, $n = 18$, $a = -8.5$, $b = 2$. Eigenvalues computed in quadruple precision* (□)*, and approximations to zero eigenvalues provided by* `eigen` (○) *and by the LAPACK subroutine* `dhseqr` (∗).

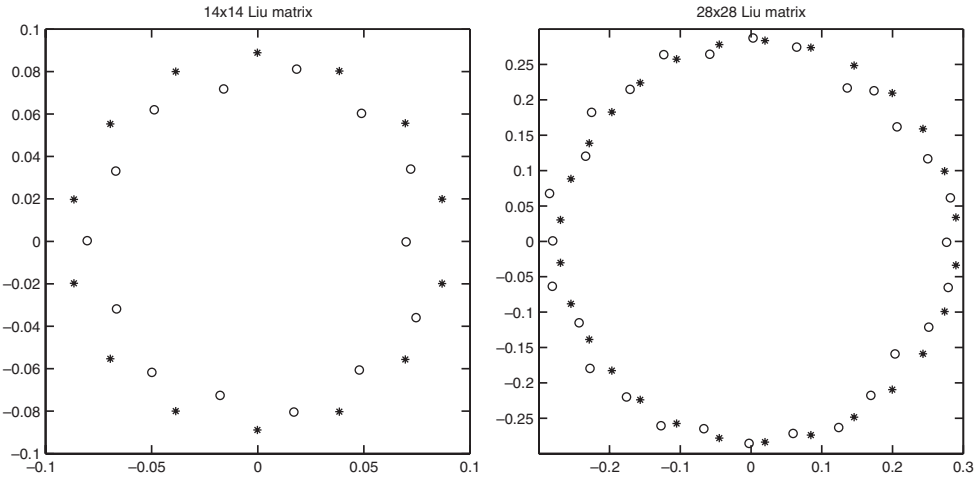

FIG. 5.5. *Liu's matrices. Approximations to zero eigenvalues provided by* `eigen` (○) *and by the LAPACK subroutine* `dhseqr` (∗).

than the LAPACK subroutine. The difference of the errors is more evident for the eigenvalues of small modulus.

To illustrate our running error bounds $\widetilde{\rho}$ in (4.3) we consider two matrices:

1. The tridiagonal matrix defined in factored form (5.1) by

$$\alpha_j = 10^{6(-1)^{j+1}}, \quad \delta_j = 1,\ j \le n/2, \quad \delta_j = -1,\ j > n/2$$

   and for which the eigenvalues are grouped into five clusters.
2. Liu's matrix T_{14} modified with $\alpha_6^{14} = 2^{-5}$ and $\alpha_{14}^{14} = 2^{-20}$ so that all its eigenvalues are distinct.

Note that these two matrices are exactly represented in base 2. In Tables 5.3 and 5.4 we report, for each eigenvalue λ, the relative error for its approximation by `eigen` and our running error bound $\widetilde{\rho}$ (4.3). We also report the structured condition number

TABLE 5.3
Bounds $\widetilde{\rho}$ on the relative error for the eigenvalues of T: $\alpha_j = 10^{6(-1)^{j+1}}$, $\delta_j = 1$, $j \le n/2$, $\delta_j = -1$, $j > n/2$, $n = 10$.

λ	$\frac{\lvert\lambda - \hat{\lambda}\rvert}{\lvert\lambda\rvert}$	$\widetilde{\rho}$	$\kappa_s(\lambda)$	$\eta_s(\lambda)$
-1.0e+6	0	5e-23	0.7	2e-22
-1.0e+6	0	2e-24	0.7	2e-22
-1.7e-6	7e-9	7e-8	5e11	2e-20
-7.4e-7	1e-4	1e-3	1e12	2e-16
2.2e-7 + i5.0e-7	4e-6	4e-5	1e12	5e-18
2.2e-7 - i5.0e-7	4e-10	4e-9	1e12	8e-22
2.0e-6	4e-6	4e-5	4e11	1e-17
1.0e+6	0	1e-16	0.7	3e-16
1.0e+6	0	1e-1	0.7	1e-16
1.0e+6	0	8e-17	0.7	2e-16

TABLE 5.4
Bounds $\widetilde{\rho}$ on the relative error: Liu's matrix T_{14}, modified with $\alpha_6 = 2^{-5}$ and $\alpha_{14} = 2^{-20}$.

λ	$\frac{\lvert\lambda - \hat{\lambda}\rvert}{\lvert\lambda\rvert}$	$\widetilde{\rho}$	$\kappa_s(\lambda)$	$\eta_s(\lambda)$
-7.2e-1	8e-16	5e-12	1e2	2e-15
-6.8e-1 ± i 3.5e-1	2e-15	4e-12	7e1	2e-15
-3.4e-1 ± i 6.9e-1	3e-15	4e-12	6e1	2e-14
-6.1e-3 ± i 7.1e-1	1e-15	5e-12	5e1	1e-14
-5.5e-3	2e-11	1e-8	4e6	1e-16
5.5e-3	9e-11	2e-8	4e6	1e-16
3.6e-1 ± i 6.8e-1	3e-15	4e-12	6e1	9e-15
6.9e-1 ± i 3.6e-1	1e-15	4e-12	6e1	2e-15
7.1e-1	3e-15	6e-12	1e2	4e-16

$\kappa_s(\lambda)$ and the structured backward error $\eta_s(\lambda)$ for which we impose the perturbations to be real tridiagonal [23]. The product $\kappa_s(\lambda)\eta_s(\lambda)$ provides a first order bound on the forward error $|\lambda - \hat{\lambda}|/|\lambda|$. The results in Tables 5.3 and 5.4 confirm that the running error bound is a strict upper bound and show that for these two matrices it exceeds the error by 1 to 3 orders of magnitude. It is important to notice that computing $\widetilde{\rho}$ is inexpensive, requiring just $O(n)$ flops per eigenvalue. On the other hand, the upper bound $\kappa_s(\lambda)\eta_s(\lambda)$ requires the knowledge of the right and left eigenvectors and is usually expensive to compute when structure in the perturbations is imposed. The small values for $\eta_s(\lambda)$ show that on these examples `eigen` is structured backward stable.

Some performance tests have been carried out with n ranging from 200 to 6400 on a PC with an Athlon 1800 CPU. Table 5.5 reports the CPU time in seconds required by `eigen` versus the time required by the LAPACK subroutine `dhseqr`. These timings show that in all the cases the growth of the CPU time required by our algorithm is a quadratic function of n, whereas the cost of `dhseqr` grows cubically with n. The threshold value for which our algorithm is faster than the LAPACK subroutine is about $n = 400$ for Tests 1, 2, 3, 8, 10, $n = 800$ for Tests 5, 7, 9, and $n = 1600$ for Tests 4 and 6. The speed-up reached for $n = 6400$ is in the range 6.5–189.9.

For all the test matrices with the exception of Tests 4 and 6 the algorithm converges quickly: the average number of iterations per eigenvalue required in the last

TABLE 5.5
CPU time in seconds for `eigen` *(in boldface) versus LAPACK's* `dhseqr`.

n	200	400	800	1600	3200	6400
Test 1	**0.1**/0.06	**0.4**/0.6	**1.4**/4.8	**5.6**/76	**24**/661	**103**/6209
Test 2	**0.1**/0.10	**0.4**/0.8	**1.2**/7.5	**4.5**/56	**19**/315	**82**/2378
Test 3	**0.1**/0.07	**0.3**/0.7	**1.1**/6.7	**4.3**/152	**18**/1510	**80**/15188
Test 4	**0.9**/0.07	**3.1**/0.6	**13.2**/4.3	**54**/75.6	**249**/896	**1100**/6092
Test 5	**0.2**/0.10	**1.1**/0.9	**4.9**/6.8	**21**/135	**84**/1239	**360**/11088
Test 6	**0.8**/0.08	**2.9**/0.6	**13.2**/5.4	**54**/93	**251**/744	**1128**/7351
Test 7	**0.3**/0.10	**0.9**/0.8	**3.1**/6.7	**12.0**/124	**49**/1460	**198**/9155
Test 8	**0.08**/0.08	**0.3**/0.6	**0.9**/4.0	**4.0**/49.9	**17**/382	**76**/2838
Test 9	**0.4**/0.11	**1.6**/0.9	**6.1**/8.2	**27**/175	**114**/1450	**543**/13250
Test 10	**0.2**/0.08	**0.6**/0.7	**2.1**/6.0	**7.7**/130	**32**/975	**137**/8990

TABLE 5.6
Average number of iterations per eigenvalue and maximum number of iterations (in parentheses) of the Ehrlich–Aberth method.

n	200	400	800	1600	3200	6400
Test 1	1.9 (3)	1.9 (26)	1.9 (21)	1.8 (20)	1.8 (21)	1.8 (19)
Test 2	1.9 (5)	1.8 (14)	1.5 (4)	1.5 (3)	1.5 (6)	1.5 (6)
Test 3	1.9 (4)	1.6 (4)	1.5 (4)	1.5 (3)	1.5 (6)	1.5 (6)
Test 4	21.7 (26)	16.8 (26)	19.5 (26)	18.0 (26)	19.0 (50)	18.1 (29)
Test 5	4.8 (17)	7.1 (27)	7.8 (28)	7.4 (25)	5.8 (27)	5.8 (36)
Test 6	22.6 (26)	16.2 (24)	21.5 (27)	18.7 (27)	19.7 (26)	18.9 (25)
Test 7	4.6 (10)	3.9 (10)	3.5 (11)	3.3 (14)	3.2 (17)	2.6 (19)
Test 8	1.4 (3)	1.4 (3)	1.4 (3)	1.4 (3)	1.4 (3)	1.4 (2)
Test 9	5.9 (14)	5.9 (13)	5.8 (15)	5.9 (22)	5.8 (17)	5.7 (21)
Test 10	2.7 (7)	2.4 (7)	2.3 (8)	2.1 (12)	2.1 (9)	2.0 (9)

recursive step is between 1.9 and 5.9 (see Table 5.6). In Test 6, despite the eigenvalues being real since the matrix is real symmetric, the average number of iterations per eigenvalue is larger than in the other tests for which the eigenvalues are complex. In general, the Ehrlich–Aberth iteration does not take any advantage of the reality of the eigenvalues; moreover the convergence speed is not improved by choosing initial real approximations, and the best choice remains to select initial approximations along a suitable circle in the complex plane [21]. The average number of Ehrlich–Aberth iterations per eigenvalue seems to be almost independent of n and varies according to the specific problem.

6. Conclusion. We have introduced an algorithm for the computation of the Newton quotient $p(\lambda)/p'(\lambda)$ for $p(\lambda) = \det(T - \lambda I)$, and T a tridiagonal matrix, based on the QR factorization of $T - \lambda I$ and on the semiseparable structure of $(T - \lambda I)^{-1}$. The algorithm, whose arithmetic cost is linear in the size n of the matrix T, is robust.

This algorithm was used for implementing the Ehrlich–Aberth iteration, which approximates all eigenvalues of T. Besides the straightforward way of choosing the initial approximations in the unit circle, a more elaborate strategy for the choice of the initial approximations was proposed. This strategy is based on a divide-and-conquer technique, which, even though heuristic, is motivated by some inclusion results that we have proved in section 3.2. Running error bounds for the errors in the computed eigenvalues were also provided.

The numerical experiments, performed with a large set of test matrices, confirmed the effectiveness of the algorithm. Comparisons with the LAPACK subroutine `dhseqr` showed that for moderately large values of n our algorithm is faster. In particular, the Ehrlich–Aberth iteration has a cost which is $O(n^2)$, whereas the LAPACK subroutine costs $O(n^3)$ operations. Moreover, the Ehrlich–Aberth iteration generally yields more accurate computed eigenvalues than the LAPACK subroutine `dhseqr`.

Acknowledgments. We thank Inderjit Dhillon and the referees for valuable suggestions that improved the paper, and Sven Hammarling for pointing out that Theorem 2.1, which we had obtained independently, appears in [19].

REFERENCES

[1] O. Aberth, *Iteration methods for finding all zeros of a polynomial simultaneously*, Math. Comp., 27 (1973), pp. 339–334.

[2] L. Adams and P. Arbenz, *Towards a divide and conquer algorithm for the real nonsymmetric eigenvalue problem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1333–1353.

[3] D. Bindel, J. Demmel, W. Kahan, and O. Marques, *On computing Givens rotations reliably and efficiently*, ACM Trans. Math. Software, 28 (2002), pp. 206–238.

[4] D. A. Bini, *Numerical computation of polynomial zeros by means of Aberth's method*, Numer. Algorithms, 13 (1996), pp. 179–200.

[5] D. A. Bini and G. Fiorentino, *MPSsolve: Numerical computation of polynomial roots*, v. 2.2, FRISCO report, 1999. Software and papers available online from ftp://ftp.dm.unipi.it/pub/mpsolve.

[6] D. A. Bini and G. Fiorentino, *Design, analysis, and implementation of a multiprecision polynomial rootfinder*, Numer. Algorithms, 23 (2000), pp. 127–173.

[7] M. A. Brebner and J. Grad, *Eigenvalues of $Ax = \lambda Bx$ for real symmetric matrices A and B computed by reduction to a pseudosymmetric form and the HR process*, Linear Algebra Appl., 43 (1982), pp. 99–118.

[8] A. Bunse-Gerstner, *An analysis of the HR algorithm for computing the eigenvalues of a matrix*, Linear Algebra Appl., 35 (1981), pp. 155–173.

[9] C. Carstensen, *Inclusion of the roots of a polynomial based on Gerschgorin's theorem*, Numer. Math., 59 (1991), pp. 349–360.

[10] P. A. Clement, *A class of triple-diagonal matrices for test purposes*, SIAM Rev., 1 (1959), pp. 50–52.

[11] I. S. Dhillon, *Reliable computation of the condition number of a tridiagonal matrix in $O(n)$ time*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 776–796.

[12] I. Dhillon, *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, Ph.D. thesis, UCB Tech. Report UCB//CSD-97-971, University of California, Berkeley, CA, 1997.

[13] J. J. Dongarra, G. A. Geist, and C. H. Romine, *Algorithm* 710: *FORTRAN subroutines for computing the eigenvalues and eigenvectors of a general matrix by reduction to general tridiagonal form*, ACM Trans. Math. Software, 18 (1992), pp. 392–400.

[14] Q. Du, M. Jin, T. Y. Li, and Z. Zeng, *The quasi-Laguerre iteration*, Math. Comp., 66 (1997), pp. 345–361.

[15] L. W. Ehrlich, *A modified Newton method for polynomials*, Comm. ACM, 10 (1967), pp. 107–108.

[16] K. V. Fernando, *On computing an eigenvector of a tridiagonal matrix. Part* I: *Basic results*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 1013–1034.

[17] I. Gargantini, *Parallel Laguerre iterations: The complex case*, Numer. Math., 26 (1976), pp. 317–323.

[18] G. A. Geist, *Reduction of a general matrix to tridiagonal form*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 362–373.

[19] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, *Methods for modifying matrix factorizations*, Math. Comp., 28 (1974), pp. 505–535.

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

[21] H. W. Guggenheimer, *Initial approximations in Durand-Kerner's root finding method*, BIT, 26 (1986), pp. 537–539.

[22] P. Henrici, *Applied and Computational Complex Analysis*, Vol. 1, Wiley, New York, 1974.

[23] D. J. Higham and N. J. Higham, *Structured backward error and condition of generalized eigenvalue problems*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 493–512.
[24] N. J. Higham, *Efficient algorithms for computing the condition number of a tridiagonal matrix*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 150–165.
[25] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.
[26] Y. Ikebe, *On inverses of Hessenberg matrices*, Linear Algebra Appl., 24 (1979), pp. 93–97.
[27] E. R. Jessup, *A case against a divide and conquer approach to the nonsymmetric eigenvalue problem*, Appl. Numer. Math., 12 (1993), pp. 403–420.
[28] W. Kahan, *Accurate Eigenvalues of a Symmetric Tri-diagonal Matrix*, Tech. Report CS-TR-66-41, Stanford University, Stanford, CA, 1966.
[29] Z. S. Liu, *On the Extended HR Algorithm*, Tech. Report PAM-564, Center for Pure and Applied Mathematics, University of California, Berkeley, CA, 1992.
[30] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.
[31] B. N. Parlett and H. C. Chen, *Use of indefinite pencils for computing damped natural modes*, Linear Algebra Appl., 140 (1990), pp. 53–88.
[32] B. N. Parlett and I. S. Dhillon, *Fernando's solution to Wilkinson's problem: An application of double factorization*, Linear Algebra Appl., 267 (1997), pp. 247–279.
[33] L. Pasquini, *Accurate computation of the zeros of the generalized Bessel polynomials*, Numer. Math., 86 (2000), pp. 507–538.
[34] M. S. Petković, *Iterative Methods for Simultaneous Inclusion of Polynomial Zeros*, Springer-Verlag, Berlin, Heidelberg, New York, 1989.
[35] M. S. Petković, L. D. Petković, and L. Rančić, *Higher-order simultaneous methods for the determination of polynomial multiple zeros*, Int. J. Comput. Math., 80 (2000), pp. 1407–1427.
[36] M. S. Petković, L. D. Petković, and D. Živković, *Laguerre-like methods for the simultaneous approximation of polynomial zeros*, in Topics in Numerical Analysis, Comput. Suppl. 15, Springer-Verlag, Vienna, 2001, pp. 189–209.
[37] M. S. Petković and D. V. Vranić, *The convergence of Euler-like method for the simultaneous inclusion of polynomial zeros*, Comput. Math. Appl., 39 (2000), pp. 95–105.
[38] H. Rutishauser, *Solution of eigenvalue problems with the LR-transformation*, Nat. Bur. Standards Appl. Math. Ser., 1958 (1958), pp. 47–81.
[39] F. Tisseur, *Tridiagonal-diagonal reduction of symmetric indefinite pairs*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 215–232.
[40] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–286.