LAGUERRE'S ITERATION IN SOLVING THE SYMMETRIC TRIDIAGONAL EIGENPROBLEM - REVISITED *

T. Y. LI† AND ZHONGGANG ZENG‡

Abstract. In this paper we present an algorithm for the eigenvalue problem of symmetric tridiagonal matrices. Our algorithm employs the determinant evaluation, split-and-merge strategy and Laguerre's iteration. The method directly evaluates eigenvalues and uses inverse iteration as an option when eigenvectors are needed. This algorithm combines the advantages of existing algorithms such as QR, bisection/multisection and Cuppen's divide-and-conquer method. It is fully parallel, and competitive in speed with the most efficient QR algorithm in serial mode. On the other hand, our algorithm is as accurate as any standard algorithm for the symmetric tridiagonal eigenproblem and enjoys the flexibility in evaluating partial spectrum.

Key words. eigenvalue, Laguerre's iteration, symmetric tridiagonal matrix

1. Introduction. For a symmetric tridiagonal matrix T with nonzero subdiagonal entries, the eigenvalues of T or the zeros of its characteristic polynomial

$$f(\lambda) = \det[T - \lambda I] \tag{1.1}$$

are all real and simple. Therefore, the globally convergent Laguerre iteration

$$L_{\pm}(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{(n-1)\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}}. \tag{1.2}$$

with cubic convergence rate for solving a polynomial equation with real and simple zeros seems perfectly suitable for finding eigenvalues of T. This method was mentioned many years ago in several places (e.g., see Wilkinson [28], pp 443-445 (1965)). There are also literatures about Laguerre's iteration for nonsymmetric eigenvalue problems [23, 27]. However, a serious investigation of the practicality of the method, such as efficiency and accuracy, and an attempt to efficiently implement the algorithm have not yet been done for the symmetric tridiagonal eigenproblem. The purpose of this paper is to revisit this method with an intensive examination of the theory and practicality and an effort to fully implement the algorithm efficiently. The proposed algorithm evaluates eigenvalues of a symmetric tridiagonal matrices without computing its eigenvectors. In case eigenvectors are also needed, the inverse iteration can be applied [14].

First of all, to use Laguerre's iteration (1.2) for finding zeros of the polynomial f in (1.1), one needs to evaluate f, f' and f''. Those values can be efficiently evaluated by certain three-term recurrence relations (see §2). It is well known that those three-term recurrences may suffer from a severe underflow-overflow problem. In §2, we propose an alternative scheme which can virtually avoid underflow-overflow problems. The accuracy of our scheme will also be analyzed.

To find some or all of the eigenvalues of T by Laguerre's iteration, a set of proper starting points is essential. For this purpose, our split-merge process proposed in

*The research was supported in part by NSF under Grant CCR-9024840

†Department of Mathematics, Michigan State University, East Lansing, MI 48824-1027 (li@math.msu.edu).

‡Department of Mathematics, Northeastern Illinois University, Chicago (zzeng@neiu.edu).

§3, similar to Cuppen's divide-and-conquer strategy [4, 7, 11, 25], provides a close matrix $\hat{T}$ of T, whose eigenvalues constitute an excellent set of starting points. These starting points can lead to desired eigenvalues of T with about 2–3 steps of Laguerre's iteration in general.

Laguerre's iteration converges only linearly when it is used to approximate zeros with multiplicity $r > 1$. Although all the eigenvalues of T are simple, some of them may be indistinguishable numerically. In the actual computation, the Laguerre iteration also converges linearly in this case as if those eigenvalues, say r of them, were a multiple eigenvalue with multiplicity r. Slow convergence also occurs when a group of r eigenvalues are relatively close to each other compared to the distance from the starting point of the iteration. In all those situations, the modified Laguerre iteration

$$L_{r\pm}(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{\frac{n-r}{r}\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}}$$

can be used to speed up the process and maintain the cubic convergence rate. The behavior of the modified Laguerre iteration will be studied in §2.2. The estimation of the number r, which plays an important role in practice, can be obtained from the information of the closeness of the eigenvalues of the matrix $\hat{T}$ in our split-merge process, and an algorithm is given in §3.2.

Homotopy continuation methods have recently been developed for algebraic eigenvalue problems, and very encouraging results have been obtained for both the symmetric and nonsymmetric cases [12, 16, 17, 19, 20, 21, 29]. While our algorithm does not employ the homotopy continuation method directly, the design of our algorithm is critically based on the consideration of the structure of the eigenvalue curves of the homotopy

$$h(t, \lambda) = \det[(1-t)\hat{T} + tT - \lambda I].$$

The symmetric tridiagonal eigenvalue problem has been intensively studied, and hence forms an excellent testing ground for newly developed algorithms. There are quite a few reliable algorithms for this problem with various features, such as QR (or QL) [8], D&C (Cuppen's divide-and-conquer) [4, 7] and B/M (bisection/multisection) [13, 22]. A detailed comparison of these algorithms can be found in [5]. In summary, the QR algorithm is believed to be the most efficient algorithm on serial machines for matrices of moderate sizes. D&C is an excellent parallel algorithm. B/M, which also parallelizable, leads in accuracy and possesses the flexibility in the evaluation of partial spectrum. When only eigenvalues are needed, both QR and B/M are able to evaluate eigenvalues without computing eigenvectors. The promise of our algorithm is that it contains most of the advantages of these algorithms:

(i) the same parallel structure as D&C but less memory contention;
(ii) the same accuracy as B/M;
(iii) the same flexibility as B/M in evaluating partial spectrum;
(iv) the same advantage as QR and B/M in separating the evaluation of eigenvalues and eigenvectors.

In §5, comprehensive numerical results comparing our algorithm with these methods on substantial variety of types of matrices are presented. It appears that, with our stopping criterion, our algorithm achieves the best accuracy in evaluating eigenvalues. The speed is competitive with the QR algorithm in serial mode, and leads

B/M as well as D&C by a considerable margin. When only a small fraction of the total number of eigenvalues are in demand, our algorithm can take full advantage of this situation by reducing the execution time to a similar fraction.

Modern scientific computing is marked by the advent of vector and parallel computers and the search for algorithms that are to a large extent parallel in nature. An important advantage of our algorithm is its natural parallelism, in the sense that each eigenvalue is computed totally independently of the others. Besides, our algorithm has a great capacity for vectorization. These considerations will be discussed in §6 and an intensive experiment will be reported in a separate article [26].

The code of our algorithm is electronically available from Zhonggang Zeng.

2. Laguerre's iteration.

2.1. Evaluation of the determinant and its accuracy. Let T be a symmetric tridiagonal matrix of the form

$$T = [\beta_{i-1}, \alpha_i, \beta_i] = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix}. \tag{2.1}$$

Without loss of generality, we may assume T is unreduced; that is, $\beta_i \neq 0$, $i = 1, \cdots, n-1$. For unreduced T, the characteristic

$$f(\lambda) \equiv \det(T - \lambda I) \tag{2.2}$$

is a polynomial with only *real* and *simple* zeros ([28], p300). Let

$$\lambda_1 < \lambda_2 < \cdots < \lambda_n$$

be the zeros of f. Set $\lambda_0 = -\infty$ and $\lambda_{n+1} = +\infty$. For $x \in (\lambda_0, \lambda_{n+1})$, Laguerre's iterates $L_+(x)$ and $L_-(x)$ are defined as follows:

$$L_\pm(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{(n-1)\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}}. \tag{2.3}$$

The following result is known ([28], pp443-445).

PROPOSITION 2.1. *For $x \in (\lambda_m, \lambda_{m+1})$, $m = 0, 1, \cdots, n$, the following results hold:*

1. $\lambda_m < L_-(x) < x < L_+(x) < \lambda_{m+1}$.
2. *There are constants c_+ and c_- such that*

$$|L_+(x) - \lambda_{m+1}| \leq c_+ |x - \lambda_{m+1}|^3 \quad \textit{if } x \textit{ is close to } \lambda_{m+1}$$

$$|L_-(x) - \lambda_m| \leq c_- |x - \lambda_m|^3 \quad \textit{if } x \textit{ is close to } \lambda_m.$$

From this proposition, if we let

$$x_+^{(k)} = L_+^k(x) \equiv \overbrace{L_+(L_+(\cdots L_+}^{k}(x)\cdots)) \quad \text{and}$$
$$x_-^{(k)} = L_-^k(x) \equiv \overbrace{L_-(L_-(\cdots L_-}^{k}(x)\cdots)).$$

then

$$\lambda_m \longleftarrow \cdots x_-^{(2)} < x_-^{(1)} < x < x_+^{(1)} < x_+^{(2)} \cdots \longrightarrow \lambda_{m+1}.$$

The two sequences $x_+^{(k)}$ and $x_-^{(k)}$ converge monotonically to λ_m and λ_{m+1} respectively in asymptotically cubic rate. That is, they eventually enter certain neighborhoods of their corresponding limits and then exhibit cubic convergence rate.

To use Laguerre's iteration (2.3) for finding zeros of $f(\lambda)$ in (2.2), it is necessary to evaluate f, f' and f'' efficiently with satisfactory accuracy. It is well known that the characteristic polynomial $f(\lambda) \equiv \det(T - \lambda I)$ of $T = [\beta_{i-1}, \alpha_i, \beta_i]$ and its derivatives with respect to λ can be evaluated by three-term recurrences ([28], p423):

$$\begin{cases} \rho_0 = 1, \quad \rho_1 = \alpha_1 - \lambda \\ \rho_i = (\alpha_i - \lambda)\rho_{i-1} - \beta_{i-1}^2 \rho_{i-2}, \quad i = 2, 3, \ldots, n \end{cases} \tag{2.4}$$

$$\begin{cases} \rho_0' = 0, \quad \rho_1' = -1 \\ \rho_i' = (\alpha_i - \lambda)\rho_{i-1}' - \rho_{i-1} - \beta_{i-1}^2 \rho_{i-2}', \quad i = 2, 3, \ldots, n \end{cases} \tag{2.5}$$

$$\begin{cases} \rho_0'' = 0, \quad \rho_1'' = 0 \\ \rho_i'' = (\alpha_i - \lambda)\rho_{i-1}'' - 2\rho_{i-1}' - \beta_{i-1}^2 \rho_{i-2}'', \quad i = 2, 3, \ldots, n \end{cases} \tag{2.6}$$

and

$$f(\lambda) = \rho_n, \qquad f'(\lambda) = \rho_n', \qquad f''(\lambda) = \rho_n''.$$

However, these recurrences may suffer from a severe underflow-overflow problem and require constant testing and scaling. We thus propose the following alternative scheme which scales ρ_i by ρ_{i-1}. Let

$$\xi_i = \frac{\rho_i}{\rho_{i-1}}, \quad i = 1, 2, \cdots, n.$$

Dividing both sides of (2.4) by ρ_{i-1} yields,

$$\begin{cases} \xi_1 &= \alpha_1 - \lambda, \\ \xi_i &= \alpha_i - \lambda - \dfrac{\beta_{i-1}^2}{\xi_{i-1}} \ , \quad i = 2, 3, \cdots, n. \end{cases} \tag{2.7}$$

We then scale ρ_i' and ρ_i'' by ρ_i by letting

$$\eta_i = -\frac{\rho_i'}{\rho_i} \quad \text{and} \quad \zeta_i = \frac{\rho_i''}{\rho_i} \qquad i = 0, 1, \cdots, n.$$

Divide (2.5) and (2.6) by ρ_i and consider the obvious relation

$$\rho_i = \prod_{j=1}^{i} \xi_j.$$

We obtain the following scaled recurrences

$$\text{(2.8)} \qquad \begin{cases} \eta_0 = 0, \quad \eta_1 = \dfrac{1}{\xi_1} \\ \eta_i = \dfrac{1}{\xi_i} \left[(\alpha_i - \lambda)\eta_{i-1} + 1 - \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}} \right) \eta_{i-2} \right], \quad i = 2, 3, \cdots, n \end{cases}$$

$$\text{(2.9)} \qquad \begin{cases} \zeta_0 = 0, \quad \zeta_1 = 0 \\ \zeta_i = \dfrac{1}{\xi_i} \left[(\alpha_i - \lambda)\zeta_{i-1} + 2\eta_{i-1} - \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}} \right) \zeta_{i-2} \right], \quad i = 2, 3, \cdots, n \end{cases}$$

and

$$-\frac{f'(\lambda)}{f(\lambda)} = \eta_n, \qquad \frac{f''(\lambda)}{f(\lambda)} = \zeta_n.$$

These are what one really needs in (2.3). Because of its self-scaling, this process can avoid underflow-overflow problems except in extraordinarily unusual cases. However, the algorithm may break down if $\xi_i = 0$ for some $1 \le i \le n$. To prevent this, the following guardian is designed:

- If $\xi_1 = 0$ (i.e., $\alpha_1 = \lambda$), set $\xi_1 = \beta_1^2 \varepsilon^2$
- If $\xi_i = 0$, $i > 1$, set $\xi_i = \dfrac{\beta_{i-1}^2 \varepsilon^2}{\xi_{i-1}}$

where ε is the machine precision. In case of $\xi_i = 0$ $(i > 1)$, the above remedy amounts to a small relative perturbation of β_{i-1}. More precisely, we simply replace β_{i-1} by $\beta_{i-1}(1 - \varepsilon^2/2)$. For $\lambda = \alpha_1$, a perturbation $\beta_1^2 \varepsilon^2$ is introduced upon α_1.

A determinant evaluation algorithm DETEVL (see Fig. 2.1) is formulated accordingly to the recurrences (2.7) (2.8) and (2.9). When ξ_i, $i = 1, \cdots, n$ are known, the Sturm sequence is available ([24], p47). Thus, as a by-product, DETEVL also evaluates the number of eigenvalues of T which are less than λ. This number is denoted by $\kappa(\lambda)$.

To analyze the accuracy of this algorithm, we first assume the usual model of arithmetic:

$$\text{(2.10)} \qquad fl(x \circ y) = (x \circ y) \cdot (1 + e)$$

where $\circ$ is one of the operations $+, -, \times$ and $/$, $fl(*)$ is the floating point result of $*$, and $|e| \le \varepsilon$, the machine precision. Then, from (2.7),

$$fl(\xi_1) = (\alpha_1 - \lambda)(1 + e_{11}), \quad \text{with} \quad |e_{11}| < \varepsilon$$

and for $i = 2, \cdots, n$,

$$\begin{aligned} fl(\xi_i) &= fl\left((\alpha_i - \lambda) - \frac{\beta_{i-1}^2}{fl(\xi_{i-1})} \right) \\ &= \left((\alpha_i - \lambda)(1 + e_{i1}) - \left(\frac{(\beta_{i-1}^2)(1 + e_{i3})}{fl(\xi_{i-1})} \right) (1 + e_{i4}) \right) (1 + e_{i2}), \end{aligned}$$

Algorithm DETEVL:

input: $T = [\beta_{i-1}, \alpha_i, \beta_i]$, λ

output: $-\frac{f'(\lambda)}{f(\lambda)} = \eta_n$, $\frac{f''(\lambda)}{f(\lambda)} = \zeta_n$ (where $f(\lambda) \equiv \det(T - \lambda I)$)
and $\kappa(\lambda) =$ *neg_count*
(=the number of eigenvalues of T less than λ)

begin DETEVL

$\xi_1 = \alpha_1 - \lambda$

if $\xi_1 = 0$ then $\xi_1 = \beta_1^2 \varepsilon^2$

$\eta_0 = 0$, $\eta_1 = \frac{1}{\xi_1}$, $\zeta_0 = 0$, $\zeta_1 = 0$, *neg_count* $= 0$

if $\xi_1 < 0$ then *neg_count* $= 1$

for $i = 2 : n$

$\xi_i = (\alpha_i - \lambda) - \left(\frac{\beta_{i-1}^2}{\xi_{i-1}}\right)$

if $\xi_i = 0$ then $\xi_i = \left(\frac{\beta_{i-1}^2}{\xi_{i-1}}\right)\varepsilon^2$

if $\xi_i < 0$ then *neg_count* = *neg_count* $+1$

$\eta_i = \frac{1}{\xi_i}\left[(\alpha_i - \lambda)\eta_{i-1} + 1 - \left(\frac{\beta_{i-1}^2}{\xi_{i-1}}\right)\eta_{i-2}\right]$

$\zeta_i = \frac{1}{\xi_i}\left[(\alpha_i - \lambda)\zeta_{i-1} + 2\eta_{i-1} - \left(\frac{\beta_{i-1}^2}{\xi_{i-1}}\right)\zeta_{i-2}\right]$

end for

end DETEVL

FIG. 2.1. **Algorithm** DETEVL

where, $|e_{im}| \le \varepsilon$, for $i = 2, \cdots, n$ and $m = 1, 2, 3, 4$. So,

$$\frac{fl(\xi_1)}{1 + e_{11}} = \alpha_1 - \lambda$$

and, for $i = 2, \cdots, n$,

$$\frac{fl(\xi_i)}{(1 + e_{i1})(1 + e_{i2})} = (\alpha_i - \lambda) - \frac{\beta_{i-1}^2 \frac{(1+e_{i3})(1+e_{i4})}{(1+e_{i1})(1+e_{i-1,1})(1+e_{i-1,2})}}{\frac{fl(\xi_{i-1})}{(1+e_{i-1,1})(1+e_{i-1,2})}}.$$

Let

$$\varsigma_1 = \frac{fl(\xi_1)}{1 + e_{11}} \quad and \quad \varsigma_i = \frac{fl(\xi_i)}{(1 + e_{i1})(1 + e_{i2})}, \quad i = 2, \cdots, n.$$

Then

$$\begin{cases} \varsigma_1 &= \alpha_1 - \lambda, \\ \varsigma_i &= \alpha_i - \lambda - \frac{[\beta_{i-1}(1+\varepsilon_{i-1})]^2}{\varsigma_{i-1}}, \quad i = 2, 3, \cdots, n. \end{cases} \tag{2.11}$$

where

$$1 + \varepsilon_{i-1} = \left[\frac{(1 + e_{i3})(1 + e_{i4})}{(1 + e_{i1})(1 + e_{i-1,1})(1 + e_{i-1,2})}\right]^{\frac{1}{2}}. \tag{2.12}$$

With (2.7) in mind, the relations in (2.11) provides the recurrence for evaluating the determinant of the matrix

$$\begin{pmatrix} \alpha_1-\lambda & \beta_1(1+\varepsilon_1) & & & & \\ \beta_1(1+\varepsilon_1) & \alpha_2-\lambda & \beta_2(1+\varepsilon_2) & & & \\ & \ddots & \ddots & \ddots & & \\ & & \beta_{n-2}(1+\varepsilon_{n-2}) & \alpha_{n-1}-\lambda & \beta_{n-1}(1+\varepsilon_{n-1}) & \\ & & & \beta_{n-1}(1+\varepsilon_{n-1}) & \alpha_n-\lambda & \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{pmatrix} + \begin{pmatrix} 0 & \beta_1\varepsilon_1 & & \\ \beta_1\varepsilon_1 & 0 & \ddots & \\ & \ddots & \ddots & \beta_{n-1}\varepsilon_{n-1} \\ & & \beta_{n-1}\varepsilon_{n-1} & 0 \end{pmatrix} - \lambda I$$

$$= (T+E) - \lambda I$$

with

$$E = \begin{pmatrix} 0 & \beta_1\varepsilon_1 & & \\ \beta_1\varepsilon_1 & 0 & \ddots & \\ & \ddots & \ddots & \beta_{n-1}\varepsilon_{n-1} \\ & & \beta_{n-1}\varepsilon_{n-1} & 0 \end{pmatrix}.$$

It follows from (2.12) that

$$-2.5\varepsilon + O(\varepsilon^2) \le \varepsilon_i \le 2.5\varepsilon + O(\varepsilon^2) \tag{2.13}$$

for $i = 1, \cdots, n$. If $\lambda = \alpha_1$, namely, the breakdown occurs at the beginning, then an additional backward error $\beta_1^2\varepsilon^2$ is added to the $(1,1)$ entry of E. A breakdown at the i-th step makes a further $O(\varepsilon^2)$ perturbation on β_{i-1}.

Now, since $\prod_{i=1}^n \varsigma_i = \det[(T+E)-\lambda I]$, so,

$$fl[f(\lambda)] = fl[\det(T-\lambda I)] = fl\left[\prod_{i=1}^n fl(\xi_i)\right] = fl\left[\prod_{i=1}^n \varsigma_i(1+e_{i1})(1+e_{i2})\right]$$

$$= \prod_{i=1}^n \varsigma_i(1+e_{i1})(1+e_{i2})(1+e_{i5}) = \det[(T+E)-\lambda I](1+\gamma)$$

where

$$1+\gamma = \prod_{i=1}^n (1+e_{i1})(1+e_{i2})(1+e_{i5}).$$

with $e_{12} = e_{15} = 0$ and $|e_{i5}| \le \varepsilon$, $i = 2, \cdots, n$. It can be easily seen that

$$-(3n-2)\varepsilon + O(\varepsilon^2) \le \gamma \le (3n-2)\varepsilon + O(\varepsilon^2).$$

By Weyl's Inequality ([3] p34, or Lemma 6.1 in §6.1), if $\breve{\lambda}_1 < \breve{\lambda}_2 < \cdots < \breve{\lambda}_n$ are zeros of $\breve{f}(\lambda) \equiv (1+\gamma)\det[(T+E)-\lambda]$, then

$$|\breve{\lambda}_i - \lambda_i| \le \|E\|_2 \le 2.5\varepsilon \left[\max_j(|\beta_j| + |\beta_{j+1}|)\right] + O(\varepsilon^2) \quad i = 1, 2, \cdots, n$$

where λ_i's are eigenvalues of T. Because the exact value of $\breve{\lambda}_i$ is generally not obtainable, the actual accuracy of the algorithm at λ_i, with $O(\varepsilon^2)$ omitted, can be

$$|fl(\breve{\lambda}_i) - \lambda_i| \le |\breve{\lambda}_i - \lambda_i| + |fl(\breve{\lambda}_i) - \breve{\lambda}_i| \le \frac{5\varepsilon}{2} \max_j(|\beta_j| + |\beta_{j+1}|) + |\lambda_i|\varepsilon. \tag{2.14}$$

The error analysis above, combined with the perturbation theory of Demmel and Kahan ([6], Theorem 2, p875), leads to an important extension of our algorithm in the evaluation of singular values of bidiagonal matrices. Based on this algorithm and a hybrid strategy, the singular values of bidiagonal matrices can be evaluated within a tiny relative error [18].

REMARK 2.2. Wilkinson [28, p303] analyzed the three term recurrence (2.4) and proved that $\rho_n = \hat{f}(\lambda)$ where $\hat{f}(\lambda)$ is the exact characteristic polynomial of $S + \delta S$ with

$$\delta S = \begin{bmatrix} \delta\alpha_1 & \delta\beta_1 & & \\ \delta\beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \delta\beta_{n-1} \\ & & \delta\beta_{n-1} & \delta\alpha_n \end{bmatrix}$$

such that $|\delta\alpha_i| \le 3.01\varepsilon(|\alpha_i| + |\lambda|)$ and $|\delta\beta_i| \le 1.51\varepsilon|\beta_i|$. Apparently, the modified recurrence (2.7) can have an improved error bound (2.13). The same argument can also be applied to the bisection method and a similar error analysis can be found in [15].

2.2. Pathologically close eigenvalues and modified Laguerre's iteration. Laguerre's iteration converges only linearly to multiple zeros of a polynomial (see [28], p445). Although an unreduced symmetric tridiagonal matrix can have only simple eigenvalues, some of them may be too close to be distinguishable numerically (e.g. Wilkinson matrices W_{21}^+, [28] pp308-309). In this situation, the Laguerre iteration converges with an apparently linear rate in actual computing as if the cluster of r eigenvalues were a multiple eigenvalue with multiplicity r.

For integer $0 < r \le n$, the modified Laguerre iterates L_{r+} and L_{r-} defined by

$$L_{r\pm}(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{\frac{n-r}{r}\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}} \tag{2.15}$$

produce two sequences

$$x_{r+}^{(k)} = L_{r+}^k(x) \equiv \overbrace{L_{r+}(L_{r+}(\cdots L_{r+}}^{k}(x)\cdots))$$
$$x_{r-}^{(k)} = L_{r-}^k(x) \equiv \overbrace{L_{r-}(L_{r-}(\cdots L_{r-}}^{k}(x)\cdots)).$$

When x^* is a r-fold zero of f, if $x < x^*$ and there is no zeros of f in (x, x^*), then the sequence $x_{r+}^{(k)}$, $k = 1, 2, \cdots$ converges increasingly to x^* in cubic rate ([28], p445). Similarly, if $x^* < x$ and there is no zeros of f in (x^*, x) then $x_{r-}^{(k)}$, $k = 1, 2, \cdots$ converges decreasingly to x^* in cubic rate. Numerical evidence shows that they also converge cubically when pathologically close zeros occurs. For instance,

the Wilkinson W_{21}^+ has two very close eigenvalues $\lambda_{20} = 10.74619418290332$ and $\lambda_{21} = 10.74619418290339$, with a difference of magnitude 10^{-14}. When starting from 10.0 targeting λ_{20}, the standard Laguerre's iteration L_+ and the modified Laguerre's iteration L_{2+} have a fundamental difference in performance, as shown below:

	L_+^k			L_{2+}^k	
k	$\lambda_{20} - x_+^{(k)} \approx$		k	$\lambda_{20} - x_{2+}^{(k)} \approx$	
1	0.7	linear	1	0.7	cubic
2	0.3	convergence	2	0.1	convergence
3	0.1		3	0.0004	
4	0.03		4	0.000000000008	
5	0.01		5	-0.00000000000004	overshoot
6	0.003				
7	0.0008				
8	0.0002				
9	0.00007				
10	0.00002				
11	0.000006				
12	0.000002				
13	0.0000005				
14	0.0000001				
16	0.00000004				
17	0.00000001				
18	0.000000003				
19	0.000000001				
20	0.0000000003				
21	0.00000000008				
22	0.00000000002				
23	0.000000000007				
24	0.000000000002				
25	0.0000000000005				
26	0.0000000000001				
27	0.00000000000002				

The behavior of the modified Laguerre iterates (2.15) can be seen in the following theorem.

THEOREM 2.3. *Let $\lambda_1 < \lambda_2 < \cdots < \lambda_n$ be zeros of $f(\lambda) = \det(T - \lambda I)$. Then, for $x \in (\lambda_m, \lambda_{m+1})$ and any positive integer r, we have*

$$\lambda_m < x < L_+(x) < L_{2+}(x) < \cdots < L_{r+}(x) < \lambda_{m+r} \text{ if } -\frac{f'(x)}{f(x)} > 0 \tag{2.16}$$

$$\lambda_{m-r+1} < L_{r-}(x) < \cdots < L_{2-}(x) < L_-(x) < x < \lambda_{m+1} \text{ if } -\frac{f'(x)}{f(x)} < 0 \tag{2.17}$$

with the convention $\lambda_i = -\infty$ for $i \le 0$ and $\lambda_i = +\infty$ for $i \ge n$.

(The proof is given in §7.1.)

REMARK 2.4. Let $x \in (\lambda_m, \lambda_{m+1})$ be the starting point of the Laguerre iteration for evaluating λ_{m+1}. In some occasions, there may exist a group of zeros to the right of λ_{m+1}, say $\lambda_{m+1} < \lambda_{m+2} < \cdots < \lambda_{m+r}$, which are relatively close compared to the distance from x to λ_{m+1}, as shown in Fig. 2.2.

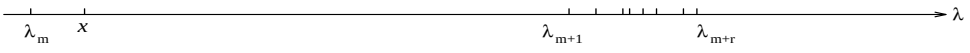

FIG. 2.2.

To reach λ_{m+1} from x, it may take many steps in the actual computation before reaching the cubic convergence neighborhood of λ_{m+1}, and the algorithm may be slowed down significantly. In this case, the modified Laguerre iteration can be used

to speed up the process, since, by the above theorem, $L_{r+}(x)$ is always bigger than $L_+(x)$. Specifically, if the number r of those relatively close zeros can be estimated, we may use modified Laguerre's iteration $L_{r+}(x)$. By Theorem 2.3,

$$x < L_+(x) < L_{r+}(x) < \lambda_{m+r},$$

and $L_{r+}(x)$ is relatively close to λ_{m+1}. It might happen that $\lambda_{m+1} < L_{r+}(x)$. In this case, we can reduce r according to $\kappa(L_{r+}(x))$, the number of eigenvalues of T smaller than $L_{r+}(x)$, which is a by-product of the algorithm DETEVL in Figure 2.1. An algorithm ESTMLT for estimating r will be given in §3.2.

2.3. Stopping criteria. Let x_i, $i = 1, 2, \cdots$ be a sequence generated by Laguerre's iteration and $x_i \longrightarrow x^*$. Then ultimately,

$$|x_{i+1} - x^*| = o(|x_{i+1} - x_i|).$$

From (2.14) an obvious criterion for stopping the iteration at x_{i+1} is

$$|x_{i+1} - x_i| \leq \quad 2.5\varepsilon \left[\max_j(|\beta_j| + |\beta_{j+1}|)\right] \quad + \quad |x_{i+1}|\varepsilon. \tag{2.18}$$

From our experience in numerical testing, this criterion seems too strict. In fact, the criterion

$$\frac{|x_{i+1} - x_i|^2}{|x_i - x_{i-1}|} \leq \quad 2.5\varepsilon \left[\max_j(|\beta_j| + |\beta_{j+1}|)\right] \quad + \quad |x_{i+1}|\varepsilon \tag{2.19}$$

has never failed. It comes from the following consideration: the Laguerre iteration converges cubically. Assuming $|x_{i+1} - x_i| \leq |x_i - x_{i-1}|$, we have

$$|x_{i+1} - x_i| \approx |x_i - x^*| \approx c|x_{i-1} - x^*|^3 \approx c|x_i - x_{i-1}|^3,$$
$$c \approx \frac{|x_{i+1} - x_i|}{|x_i - x_{i-1}|^3}.$$

Thus

$$|x_{i+1} - x^*| \approx c|x_i - x^*|^3 \approx c|x_{i+1} - x_i|^3 \approx |x_{i+1} - x_i| \left(\frac{|x_{i+1} - x_i|}{|x_i - x_{i-1}|}\right)^3 < \frac{|x_{i+1} - x_i|^2}{|x_i - x_{i-1}|}.$$

In our algorithm, we stop the iteration if either (2.18) or (2.19) satisfied.

The algorithm LAGIT in Fig. 2.3 summarizes the fundamental elements of Laguerre's iteration we described in this section.

3. Splitting and merging processes.

3.1. Splitting process. In order to find *some* or *all* the zeros of $f(\lambda) \equiv \det[T - \lambda I]$, where

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix}, \qquad \begin{array}{l} \beta_i \neq 0, \\ i = 1, \cdots, n-1, \end{array} \tag{3.1}$$

Algorithm LAGIT

```
    Input:  subscript i, initial point x ∈ (λ_{i-1}, λ_{i+1}),
            interval [a_i, b_i] ∋ λ_i, mlt evaluated by ESTMLT in Fig. 3.5.
    Output: the i-th eigenvalue λ_i of T.
    begin LAGIT
        x_1 = x
        for l = 1, 2, ···
            x_{l+1} = { L_{r+}(x_l),  if κ(x_l) < i
                      { L_{r-}(x_l),  if κ(x_l) ≥ i     with r = mlt
            δ_l = x_{l+1} − x_l
            tol = 2.5ε [max_j(|β_j| + |β_{j+1}|)] + |x_{i+1}|ε
            if (|δ_l| < tol) or (l > 1 and |δ_l^2 / δ_{l-1}| < tol) go to (#)
    (##)    evaluate −f'/f (x_{l+1}), f''/f (x_{l+1}) and κ(x_{l+1}) by DETEVL
            update [a_i, b_i] according to κ(x_{l+1})
            if mlt > 1 and |κ(x_{l+1}) − κ(x_l)| > 1 then
                mlt = |κ(x_{l+1}) − κ(x_l)|
                x_{l+1} = (x_l + x_{l+1})/2
                go to (##)
            end if
        end for
(#)     λ_i = x_{l+1}
    end LAGIT
```

FIG. 2.3. **Algorithm** LAGIT

by Laguerre's iteration, a set of proper starting values is desirable. For this purpose, we introduce the *split matrix* $\hat{T}$ by replacing some β_k in T with zero, i.e.,

$$\hat{T} = \begin{pmatrix} T_0 & 0 \\ 0 & T_1 \end{pmatrix} \tag{3.2}$$

where

$$T_0 = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{pmatrix}, \quad T_1 = \begin{pmatrix} \alpha_{k+1} & \beta_{k+1} & & \\ \beta_{k+1} & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{pmatrix}. \tag{3.3}$$

The eigenvalues of $\hat{T}$ consists of eigenvalues of T_0 and T_1. In this section, we shall study the relations between the eigenvalues of $\hat{T}$ and those of T. It turns out that eigenvalues of $\hat{T}$ form an excellent set of starting values of the Laguerre iteration.

Now, let us consider the homotopy $h : \boldsymbol{R} \times [0,1] \longrightarrow \boldsymbol{R}$ defined by

$$\begin{aligned} h(\lambda, t) &= \det\left(\left[(1-t)\hat{T} + tT\right] - \lambda I\right) \\ &= \det[T(t) - \lambda I] \end{aligned}$$

where $T(t) \equiv (1-t)\hat{T} + tT$. It is clear that for each $t \in (0,1]$, $T(t)$ is an unreduced symmetric tridiagonal matrix, and $h(\lambda_0, t_0) = 0$ if and only if λ_0 is an eigenvalue of the matrix $T(t_0)$. Let $\lambda_i(t)$ denote the i^{th} smallest eigenvalue of $T(t)$. Then every $\lambda_i(t)$ is a continuous function of t and for each $t \neq 0$, $\lambda_1(t) < \cdots < \lambda_n(t)$. Hence, the graphs of $\lambda_i(t)$'s are disjoint except possibly at $t = 0$. It was proved in [16] that any horizontal line $\lambda = \lambda^*$ can intersect at most one non-constant $\lambda_i(t)$ in [0,1] and consequently, for each $1 \leq i \leq n$, $\lambda_i(t)$ is either constant for all $t \in [0,1]$ or strictly monotonic, as shown in Fig. 3.1.

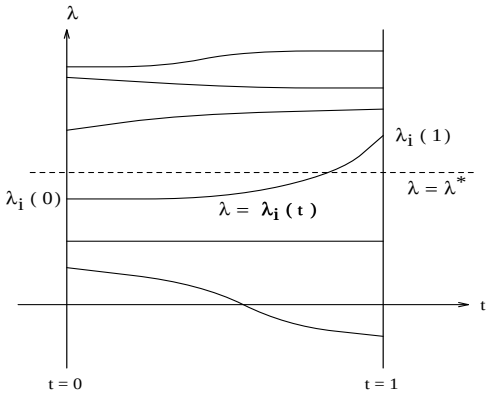

FIG. 3.1. *Eigenvalues of $T(t)$ when t increases from 0 to 1*

These important properties, along with certain interlacing and perturbation theorems, lead to the following theorem which is essential to our algorithm.

THEOREM 3.1. *Let*

$$\lambda_1 < \lambda_2 < \cdots \lambda_n \quad \text{and} \quad \hat{\lambda}_1 \leq \hat{\lambda}_2 \leq \cdots \leq \hat{\lambda}_n$$

be eigenvalues of T and $\hat{T}$ respectively. Then

$(i)\ \hat{\lambda}_1 \in (\lambda_1, \lambda_2),\ \ \hat{\lambda}_n \in (\lambda_{n-1}, \lambda_n),\ \ \hat{\lambda}_i \in (\lambda_{i-1}, \lambda_{i+1}),\ \ 2 \leq i \leq n-1;$

$(ii)\lambda_1 \in [\hat{\lambda}_1 - |\beta_k|, \hat{\lambda}_1),\ \ \lambda_n \in (\hat{\lambda}_n, \hat{\lambda}_n + |\beta_k|],\ \ \lambda_i \in (\hat{\lambda}_{i-1}, \hat{\lambda}_{i+1}),\ \ 2 \leq i \leq n-1.$

(The proof of this theorem is given in §7.2)

Notice that the multiplicity of any eigenvalue $\hat{\lambda}_i$ of $\hat{T}$ can be at most two, because both T_0 and T_1 are unreduced. Therefore, for any $2 \leq i \leq n-1$, $(\hat{\lambda}_{i-1}, \hat{\lambda}_{i+1})$ is always a non-empty interval.

It follows immediately from the above theorem the following

COROLLARY 3.2. *For each $i = 1, \cdots, n$, in the open interval $(\hat{\lambda}_i, \lambda_i)$ (or $(\lambda_i, \hat{\lambda}_i)$ if $\hat{\lambda}_i > \lambda_i$), there is no λ_j or $\hat{\lambda}_j$ for $j = 1, \cdots n$. In other words, for $i = 1, \cdots, n$, let $\mathcal{I}_i$ be the open interval with end points $\hat{\lambda}_i$ and λ_i, then $\{\mathcal{I}_i\}_{i=1}^n$ is a collection of disjoint intervals.*

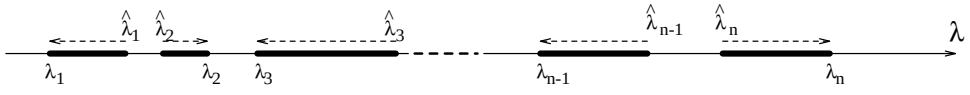

FIG. 3.2. *Corollary 3.2*

According to this corollary, for each $i = 1, \cdots, n$ the Laguerre iteration starting from $\hat{\lambda}_i$ can reach λ_i without any obstacles (See Fig. 3.2). Thus, to evaluate the i-th smallest eigenvalue λ_i of T, the corresponding eigenvalue $\hat{\lambda}_i$ of $\hat{T}$ is always used as a starting point in our algorithm.

3.2. Merging process. The eigenvalues of $\hat{T}$ in (3.1) consists of eigenvalues of T_0 and T_1 in (3.3). To find eigenvalues of T_0 and T_1, the *splitting process* may be applied recursively (See Fig. 3.3) until 2×2 or 1×1 matrices are reached.

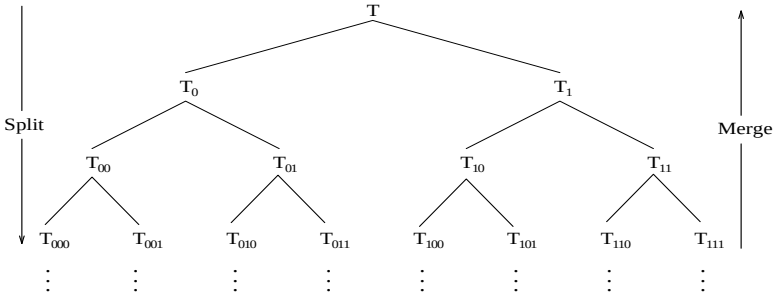

FIG. 3.3. *Splitting and merging*

After T is well split into a tree structure as in Fig. 3.3, a *merging process* in the reverse direction from 2×2 and/or 1×1 matrices can be started. More specifically, let T_σ be split into $T_{\sigma 0}$ and $T_{\sigma 1}$ in the splitting process. Let $\hat{\lambda}_1^{(\sigma)}, \cdots, \hat{\lambda}_m^{(\sigma)}$ be eigenvalues of $\hat{T}_\sigma = \begin{pmatrix} T_{\sigma 0} & 0 \\ 0 & T_{\sigma 1} \end{pmatrix}$ in ascending order. Then the Laguerre iteration is applied to the polynomial equation

$$f_\sigma(\lambda) \equiv \det[T_\sigma - \lambda I] = 0$$

from every $\hat{\lambda}_i^{(\sigma)}$ to obtain the corresponding eigenvalue $\lambda_i^{(\sigma)}$ of T_σ, $i = 1, \cdots, m$. This process is continued until T_0 and T_1 are merged into T. That is, in the final step all the eigenvalues of T are obtained by applying Laguerre's iteration to $f(\lambda) = \det(T - \lambda I)$ from eigenvalues of T_0 and T_1.

In the rest of this section, we discuss, without loss of generality, only the case of merging submatrices T_0 and T_1 in (3.3) into T in (3.1), or, finding eigenvalues

$$\lambda_1 < \lambda_2 < \cdots < \lambda_n$$

of T from eigenvalues of

$$\hat{\lambda}_1 \leq \hat{\lambda}_2 \leq \cdots \leq \hat{\lambda}_n$$

of $\hat{T}$ by Laguerre's iteration.

From property (i) of Theorem 3.1, for each $i = 1, \cdots, n$, $\hat{\lambda}_i \in (\lambda_{i-1}, \lambda_{i+1})$ with the convention $\lambda_0 = -\infty$ and $\lambda_{n+1} = +\infty$. Suppose $\hat{\lambda}_i$ is not a zero of

$$f(\lambda) = \det(T - \lambda I)$$

namely, $\hat{\lambda}_i \neq \lambda_i$. Then, either $\hat{\lambda}_i > \lambda_i$ or $\hat{\lambda}_i < \lambda_i$, this can be determined by the Sturm sequence at $\hat{\lambda}_i$. In the following discussion, we suppose $\hat{\lambda}_i > \lambda_i$. (If $\hat{\lambda}_i < \lambda_i$,

similar arguments along the same line given below follow.) It follows from Proposition 2.1, when Laguerre's iteration

$$L_{\pm}(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{(n-1)\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}}$$

is applied at $\hat{\lambda}_i$, we have

$$\lambda_i < L_-(\hat{\lambda}_i) < \hat{\lambda}_i < L_+(\hat{\lambda}_i) < \lambda_{i+1}$$

and two sequences

$$x_+^{(k)} = L_+^k(\hat{\lambda}_i) \equiv L_+(L_+^{k-1}(\hat{\lambda}_i)), \quad x_-^{(k)} = L_-^k(\hat{\lambda}_i) \equiv L_-(L_-^{k-1}(\hat{\lambda}_i))$$

$k = 1, 2, \cdots$, with $L_+^0(\hat{\lambda}_i) = L_-^0(\hat{\lambda}_i) = \hat{\lambda}_i$ can be generated with the property

$$\lambda_i \longleftarrow \cdots < x_-^{(2)} < x_-^{(1)} < \hat{\lambda}_i < x_+^{(1)} < x_+^{(2)} < \cdots \longrightarrow \lambda_{i+1}.$$

In order to reach λ_i from $\hat{\lambda}_i$, we want to stay with the sequence $\left\{x_-^{(k)}\right\}$. Recall that this sequence converges cubically to λ_i only when $x_-^{(k)}$ reach certain neighborhood N of λ_i. Before reaching N, the progress of Laguerre's iteration toward convergence may not be faster than that of the method of bisection. A simple observation (See Fig. 3.4) shows that for x near λ_i in $(\lambda_i, \lambda_{i+1})$, $-f'(x)/f(x)$ always assumes a negative value and thus the denominator in the definition of $L_{\pm}(x)$ has larger magnitude if "$-$" is chosen and the necessary condition for cubic convergence is satisfied ([28], p 445). To illustrate the role of the sign of $-f'/f$ in the convergence behavior, we use W_{21}^+ [28, p308] as an example. Starting from 0.27 and targeting eigenvalue $\lambda_3 = 0.9475343675292932$, the iteration L_+ is given below:

k	$x_+^{(k)} - \lambda_3 \approx$	sign of $-f'/f$	
1	0.8	+	linear convergence
2	0.7	+	
3	0.5	+	
4	0.2	−	$-f'/f$ changes sign
5	0.01	−	and cubic convergence begins
6	0.000004	−	
7	0.00000000000002	−	

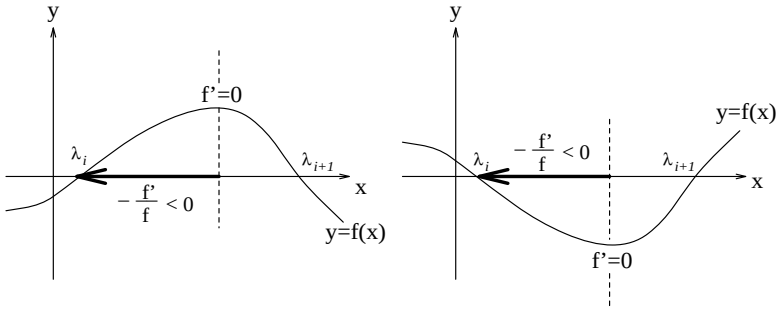

FIG. 3.4. *Natural direction of convergence*

Therefore, the sign of $-f'/f$ at $x_-^{(0)} \equiv \hat{\lambda}_i$ plays a critical role in deciding whether the Laguerre iteration at $x_-^{(0)}$ should be replaced by the method of bisection to accelerate the process:

Algorithm EstMlt

```
    input:  initial point x, sign(λ_i − x), subscript i,
                eigenvalues λ̂_1 < ··· < λ̂_n of T̂
    output: mlt, the estimated numerical multiplicity of
                the eigenvalue λ_i of T
    begin EstMlt
        mlt = 1
        for k = 1, 2, ···
            j = i + k · sign(λ_i − x)
            if |λ̂_i − λ̂_j| < 0.01|x − λ̂_i| then
                mlt = mlt + 1
            else
                go to (#)
            end if
        end for
(#) end EstMlt
```

FIG. 3.5. **Algorithm** EstMlt

(i) If $-f'/f < 0$ at $x_-^{(0)}$, then Laguerre's iteration should be used, i.e., $x_-^{(k)} = L_-^k(x_-^{(0)})$, $k = 1, 2, \cdots$.

(ii) If $-f'/f > 0$ at $x_-^{(0)}$, then $x_-^{(0)}$ is not in N. The method of bisection should be used at $x_-^{(0)}$. Namely, $x_-^{(0)}$ should be replaced by $(\hat{\lambda}_{i-1} + x_-^{(0)})/2$.

As we mentioned in Remark 2.4, if there are certain eigenvalues of T, $\lambda_{i-r+1} < \cdots < \lambda_{i-1} < \lambda_i$, which are relatively close to λ_i compared to the starting point $x_-^{(0)}$, then, it may take many steps of standard Laguerre's iteration L_- before showing cubic convergence. In this situation, the modified Laguerre iteration L_{r-} should be used at $x_-^{(0)}$. Specifically, after $x_-^{(0)}$ is altered in a bisection adjustment in (ii) above, we first estimate the number r of those eigenvalues of T from the information of the closeness of the eigenvalues $\hat{\lambda}_j$'s of $\hat{T}$: in our algorithm, we put λ_j in this group if

$$|\hat{\lambda}_j - \hat{\lambda}_i| < 0.01|x_-^{(0)} - \hat{\lambda}_i|$$

An overestimate of the number r causes essentially no harm because LagIt dynamically reduces r during the iteration, while an underestimate may result in slow convergence. An algorithm EstMlt to estimate r is designed in Fig. 3.5.

4. Cluster and partial spectrum.

4.1. Cluster evaluation. By a straightforward verification, one can obtain, from Theorem 3.1, the following

PROPOSITION 4.1. *If $[a, b]$ contains k eigenvalues of $\hat{T}$, then $[a, b]$ contains at least $k - 2$ and at most $k + 2$ eigenvalues of T. More precisely, let $\kappa(x)$ be the number of eigenvalues of T which are less than $x \in \boldsymbol{R}$ and $\hat{\lambda}_{s+1}, \cdots, \hat{\lambda}_{s+k}$ be all the eigenvalues of $\hat{T}$ in $[a, b]$. Then $s - 1 \le \kappa(a) \le s + 1$ and $s + k - 1 \le \kappa(b) \le s + k + 1$.*

This proposition can be illustrated by following eigenvalue curves $\lambda_i(t)$ of $T(t) \equiv (1 - t)\hat{T} + tT$, as two extreme cases are shown in Fig. 4.1.

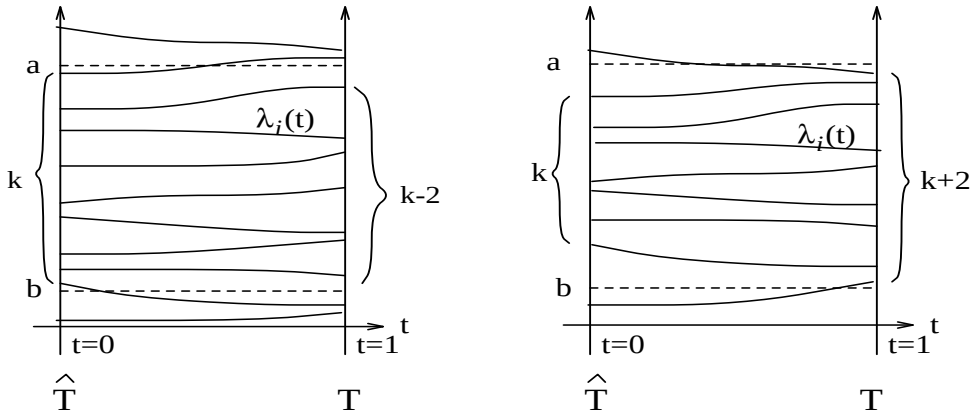

FIG. 4.1.

By this proposition, if $\hat{T}$ has a cluster of $m(\geq 3)$ eigenvalues, say $\hat{\lambda}_{i+1}, \cdots, \hat{\lambda}_{i+m}$, then $\hat{\lambda}_{i+2}, \cdots, \hat{\lambda}_{i+m-1}$ can be accepted as eigenvalues of T without further computations. In our implementation, we set $\lambda_i = \hat{\lambda}_i$ when

$$|\hat{\lambda}_{i+1} - \hat{\lambda}_{i-1}| < error\_tolerence.$$

4.2. Evaluating partial spectrum. In our algorithm, the i-th smallest eigenvalue $\hat{\lambda}_i$ of $\hat{T}$ is always used as the starting point of Laguerre's iteration to obtain the i-th smallest eigenvalue λ_i of T. Therefore, when eigenvalues of T in demand are identified in magnitude, say largest 20%, they can be evaluated by using the largest 20% eigenvalues of $\hat{T}$ as starting points without computing other eigenvalues of either $\hat{T}$ or T.

To find eigenvalues of T in a given interval $[a, b]$, the eigenvalues of $\hat{T}$ in [a,b] are known, say $\hat{\lambda}_{s+1}, \cdots, \hat{\lambda}_{s+k}$. By evaluating $\kappa(a)$ and $\kappa(b)$, the actual member of eigenvalues of T in $[a, b]$ is $\sigma = \kappa(b) - \kappa(a)$, so that $\lambda_{\kappa(a)+1}, \cdots, \lambda_{\kappa(a)+\sigma}$ are the eigenvalues of T in $[a, b]$. By Proposition 4.1, $s - 1 \leq \kappa(a) \leq s + 1$ and $s + k - 1 \leq \kappa(b) \leq s + k - 1$, so, σ can either be $k - 2$, $k - 1$, k, , $k + 1$ or $k + 2$. Thus, at most $k + 2$ starting points are necessary to evaluate these σ eigenvalues of T. Let

$$\hat{\hat{\lambda}}_s = a, \quad \hat{\hat{\lambda}}_{s+1} = \hat{\lambda}_{s+1}, \quad \cdots, \quad \hat{\hat{\lambda}}_{s+k} = \hat{\lambda}_{s+k}, \quad \hat{\hat{\lambda}}_{s+k+1} = b.$$

Among them, $\hat{\hat{\lambda}}_{\kappa(a)+1}, \cdots, \hat{\hat{\lambda}}_{\kappa(a)+\sigma}$ will lead to all σ eigenvalues of T in $[a, b]$.

The algorithm MERGE in Fig. 4.2 puts together all the elements discussed in §3 and §4.

5. Numerical testing. Our algorithm is implemented and tested on SPARC station 1 with IEEE floating point standard. The machine precision is $\varepsilon \approx 2.2 \times 10^{-16}$

5.1. Testing matrices. In the following description of matrix types, α_i, $i = 1, \cdots, n$ denote the diagonal entries and β_i, $i = 1, \cdots, n-1$ are the offdiagonal entries. The testing matrices are classified into 12 types.

Matrices with known eigenvalues

Type 1 Toeplitz matrices $[b, a, b]$. Exact eigenvalues: $\{a+2b\cos\frac{k\pi}{n+1}\}_{k=1}^{n}$ ([10], Example 7.4, p137).

Algorithm MERGE

```
input:  T = [β_{i-1}, α_i, β_i], interval [a, b]
        eigenvalues λ̂_{s+1} < ··· < λ̂_{s+k} of T̂ in [a, b].
Output: eigenvalues of T in [a, b].
begin MERGE
    set λ̂_s = a, λ̂_{s+k+1} = b
    i_1 = κ(a) + 1, i_2 = κ(b)
    for i = i_1 : i_2
        determine the interval (a_i, b_i) ∋ λ_i by Theorem 3.1
        if |b_i − a_i| > tol then
            x = λ̂_i
(#)         call DETEVL at x to obtain −f'(x)/f(x) , f''(x)/f(x)  and κ(x)
            (a_i, b_i) = { (x, b_i)   if x < λ_i (i.e.  κ(x) < i)
                         { (a_i, x)   if x > λ_i (i.e.  κ(x) ≥ i)
            if x ∉ (λ_{i-1}, λ_{i+1}) or sign(−f'(x)/f(x)) ≠ sign(λ_i − x) then
                x = (a_i + b_i)/2
                go to (#)
            end if
            call ESTMLT to estimate the multiplicity of λ_i
            call LAGIT to obtain λ_i
        else
            λ_i = (a_i + b_i)/2
        end if
    end for
end MERGE
```

FIG. 4.2. **Algorithm** MERGE

Type 2 $\alpha_1 = a - b$, $\alpha_i = a$ for $i = 2, \cdots, n-1$, $\alpha_n = a + b$. $\beta_j = b$, $j = 1, \cdots n-1$. Exact eigenvalues: $\{a + 2b\cos\frac{(2k-1)\pi}{2n}\}_{1\le k\le n}$ ([10], Example 7.6, p138).

Type 3 $\alpha_i = \begin{cases} a & \text{for odd } i \\ b & \text{for even } i \end{cases}$, $\beta_i = 1$. Exact eigenvalues :

$$\left\{ \frac{a+b \pm \left[(a-b)^2 + 16\cos^2\frac{k\pi}{n}\right]}{2} \right\}_{1\le k\le n/2} \quad \text{and } a \text{ if } n \text{ is odd}$$

([10], Example 7.8 and 7.9, p139).

Type 4 $\alpha_i = 0$, $\beta_i = \sqrt{i(n-i)}$. Exact eigenvalues: $\{-n + 2k + 1\}_{1\le k\le n}$ ([10], Example 7.10, p140).

Type 5 $\alpha_i = -[(2i-1)(n-1) - 2(i-1)^2]$, $\beta_i = i(n-i-2)$. Exact eigenvalues: $\{-k(k-1)\}_{1\le k\le n}$ ([10], Example 7.11, p141).

Wilkinson and random matrices

Type 6 Wilkinson matrices W_n^+. $\beta_i = 1$,

$$\alpha_i = \begin{cases} n/2 - i + 1 & \text{for even } n \text{ and } 1 \le i \le n/2 \\ i - n/2 & \text{for even } n \text{ and } n/2 < i \le n \\ (n-1)/2 - i + 1 & \text{for odd } n \text{ and } 1 \le i \le (n+1)/2 \\ i - (n+1)/2 & \text{for odd } n \text{ and } (n+1)/2 < i \le n \end{cases}$$

([28], pp308-309). Most of the eigenvalues are in pairs, consisting of two numerically indistinguishable eigenvalues.

Type 7 Random matrices. α_i's and β_i's are random numbers on $[0, 1]$.

LAPACK testing matrices

(Matrix types 8–12 were generated using the LAPACK test matrix generator [1].)

Type 8 Matrices with eigenvalues evenly distributed between its smallest and largest eigenvalues.

Type 9 Matrices with geometrically distributed eigenvalues. Namely, eigenvalues can be written as $\{q^k\}_{1 \le k \le n}$ for some $q \in (0, 1)$.

Type 10 Matrices with an eigenvalue 1 and the remaining in $(-\varepsilon, \varepsilon)$.

Type 11 Matrices with eigenvalues evenly distributed in the interval $(0, 1]$ except one eigenvalue with very small magnitude.

Type 12 Matrices with an eigenvalue 1 and the rest of the eigenvalue are evenly distributed in a small interval $[10^{-12} - \varepsilon, 10^{-12} + \varepsilon]$.

5.2. Testing codes. We compare the performance of the following five algorithms:

(1) S-M: our split-merge algorithm;
(2) B/M: bisection/multisection subroutine DSTEBZ in LAPACK;
(3) D&C: divide-and-conquer subroutine TREEQL developed by J. Dongarra and D. Sorensen, dated May 17, 1985 [7];
(4) RFQR: Root-free QR routine DSTERF in LAPACK, as recommended in LAPACK for evaluating eigenvalues only.
(5) QR: QR routine DSTEQR in LAPACK, as recommended in LAPACK for evaluating eigenvalues or eigenpairs.

5.3. Storage comparison. The storage requirements of these five codes for evaluating eigenvalues of an $n \times n$ matrices are:

RFQR: $2n$.
QR: $4n$. ($n^2 + O(n)$ if eigenvectors are also evaluated.)
S-M: $7n$.
B/M: $12n$.
D&C: $2n^2 + O(n)$.

The storage requirement of our algorithm, as well as those of QR and B/M, is low. Because eigenvectors are stored, the code D&C requires n^2 storage which makes it somewhat difficult to manipulate matrices with order higher than 500 on most of the workstations. The low storage of our algorithm also eases the memory contention in the parallel implementation.

5.4. Accuracy test. The accuracy is tested in two ways. First, we test all those algorithms on matrices chosen from Type 1 to Type 5 using their default error tolerances. These matrices have known eigenvalues. Thus the error can be directly

evaluated. Let $\tilde{\lambda}_i$ be the approximation of the exact eigenvalue λ_i. Table 5.1 gives the error

$$\max_i \left\{ |\tilde{\lambda}_i - \lambda_i| / \|T\| \right\}.$$

It appears that our algorithm achieves the best accuracy on these matrices. The accuracy of our algorithm, as well as B/M, is independent of the matrix size, whereas the QR, RFQR, and D&C seem less accurate when the matrix size becomes larger. On matrices of Type 4, with $n = 1999$, the error of RFQR is more than 1916 times larger than ours.

		n= 99	n=199	n=499	n= 999	n=1999
Type 1	S-M	0.67 ε	0.67 ε	0.67 ε	0.67 ε	0.67 ε
$a = 4$	B/M	1.00 ε	1.00 ε	1.00 ε	1.00 ε	1.00 ε
$b = 1$	D&C	1.00 ε	2.67 ε	4.67 ε	—	—
	QR	2.00 ε	4.00 ε	4.00 ε	4.67 ε	9.00 ε
Type 2	S-M	0.67 ε	0.67 ε	0.67 ε	0.67 ε	0.67 ε
$a = 4$	B/M	0.67 ε	0.67 ε	0.67 ε	0.67 ε	1.00 ε
$b = 1$	D&C	1.33 ε	2.67 ε	4.67 ε	—	—
	QR	2.67 ε	2.67 ε	4.00 ε	4.67 ε	6.67 ε
Type 3	S-M	0.80 ε	0.80 ε	0.80 ε	0.80 ε	0.80 ε
$a = 4$	B/M	0.80 ε	0.80 ε	0.80 ε	0.80 ε	0.80 ε
$b = 1$	D&C	3.68 ε	1.60 ε	2.40 ε	—	—
	QR	4.00 ε	4.00 ε	8.00 ε	8.00 ε	13.6ε
Type 4	S-M	0.16 ε	0.04 ε	0.13 ε	0.036 ε	0.032ε
	B/M	0.65 ε	1.29 ε	1.03 ε	1.23 ε	1.23 ε
	D&C	1.95 ε	7.76 ε	7.71 ε	—	—
	QR	7.18 ε	16.2 ε	15.4 ε	22.6 ε	24.1 ε
Type 5	S-M	0.53 ε	0.65 ε	0.65 ε	0.65 ε	0.65 ε
	B/M	0.85 ε	0.85 ε	1.06 ε	1.05 ε	1.05 ε
	D&C	1.27 ε	1.25 ε	1.85 ε	—	—
	QR	1.69 ε	4.99 ε	6.86 ε	3.26 ε	7.35 ε

TABLE 5.1

Accuracy on matrices with known eigenvalues. Because of the memory requirement, the D&C algorithm cannot be executed on our machine for $n = 999$ *or* 1999.

On the remaining types of matrices whose exact eigenvalues are unknown, the following test is made. From §2.1, the accuracy of DETEVL is

$$e(\lambda) = 2.5\varepsilon \left[\max_j (|\beta_j| + |\beta_{j+1}|) \right] + |\lambda|\varepsilon .$$

If an evaluated eigenvalue λ_i has this accuracy, then $\kappa(\lambda_i - 2e(\lambda_i)) < i$ and $\kappa(\lambda_i + 2e(\lambda_i)) \geq i$. We examine the percentage of failures of this test on all the methods. The results are in Table 5.2. As a reference, we also list the percentage of failures on matrices of Type 1 to Type 5. Notice that this test is apparently in favor of B/M and S-M because both algorithms involve the evaluation of $\kappa(\cdot)$, while the QR algorithm, which is considered quite accurate in general, may have a failure rate close to 70%.

		n= 99	n=199	n=499			n= 99	n=199	n=499
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 1	B/M	0%	0%	0%	Type 7	B/M	0%	0%	0%
	D&C	0%	10.6%	56.9%		D&C	error	error	error
	RFQR	5.1%	4.52%	6.81%		RFQR	11.1%	11.6%	26.9%
	QR	0%	7.53%	7.01%		QR	5.05%	12.1%	24.6%
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 2	B/M	0%	0%	0%	Type 8	B/M	0%	0%	0%
	D&C	0.%	9.04%	58.5%		D&C	0%	0%	4.01%
	RFQR	4.04%	4.02%	8.42%		RFQR	2.02%	1.50%	15.2%
	QR	3.03%	6.53%	8.42%		QR	0%	0.50%	16.8%
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 3	B/M	0%	0%	0%	Type 9	B/M	0%	0%	0%
	D&C	21.2%	0%	0%		D&C	error	error	error
	RFQR	3.03%	1.51%	3.61%		RFQR	1.01%	0%	0%
	QR	9.09%	7.54%	10.0%		QR	0%	0%	0%
	S-M	0%	0%	0%		S-M	0%	0%	0.20%
Type 4	B/M	0%	0%	0%	Type 10	B/M	0%	0%	0.40%
	D&C	0%	4.52%	0.40%		D&C	12.1%	2.51%	2.40%
	RFQR	6.06%	11.1%	14.6%		RFQR	0%	0%	0%
	QR	3.03%	8.54%	9.42%		QR	0%	0%	0.40%
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 5	B/M	0%	0%	0%	Type 11	B/M	0%	0%	0%
	D&C	0%	0%	0%		D&C	1.01%	1.01%	19.0%
	RFQR	1.01%	2.01%	2.00%		RFQR	0.81%	10.6%	18.2%
	QR	0%	1.00%	3.21%		QR	0.84%	6.53%	21.0%
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 6	B/M	4.04%	8.04%	8.22%	Type 12	B/M	10.1%	0%	3.21%
	D&C	8.08%	14.1%	36.9%		D&C	5.05%	1.51%	0.80%
	RFQR	51.5%	60.8%	69.3%		RFQR	0%	0%	0%
	QR	51.5%	56.3%	56.1%		QR	3.03%	0%	0.20%

TABLE 5.2

Accuracy on matrices with unknown eigenvalues (percentages are the rates of failure in the accuracy test). Because of the memory requirement, LAPACK matrix generator can not produce testing matrices for higher order.

5.5. Speed test. To compare the efficiency of these four algorithms, we perform the following experiments:

- *Evaluating all eigenvalues without computing eigenvectors.*
  The results are in Table 5.3. Our S-M algorithm shares this feature with B/M, QR and RFQR, while the D&C algorithm cannot separate the evaluation of eigenvalues and eigenvectors.
- *Evaluating all eigenvalues* and *eigenvectors.*
  Results are in Table 5.4. For this test, we use the standard QR routine DSTEQR instead of root-free QR DSTERF. Both S-M and B/M are basically designed to evaluate eigenvalues only. So, after running S-M and B/M, the inverse iteration code DSTEIN in LAPACK is used to evaluate the cor-

responding eigenvectors. In the table, the time for S-M is actually the time consumption of (S-M)+DSTEIN and the time for B/M is the time consumption of (B/M)+DSTEIN.

- *Evaluating 1/3 of largest eigenvalues without computing eigenvectors.* Both S-M and B/M have a great advantage in this case. The time consumption is reduced to about 1/3 of the time for evaluating entire spectrum. On the other hand, both QR and D&C must evaluate all the eigenvalues to obtain the largest 1/3. The numerical results are shown in Table 5.3.
- *Evaluation of 1/3 of largest eigenvalues* and *their corresponding eigenvectors.* Again, only S-M and B/M have this feature. The results are in Table 5.4. Both QR and D&C must evaluate all the eigenpairs even if 2/3 of them are unnecessary.

From those results, RFQR is apparently the fastest algorithm in computing all the eigenvalues. In this case, the speed of our algorithm S-M is competitive with RFQR with better accuracy. In evaluating all the eigenvalues and their corresponding eigenvectors, our algorithm leads DSTEQR by a considerable margin.

The results also show that our algorithm can be applied efficiently to evaluate partial spectrum and partial eigenpairs. The speed is several times faster than B/M in most of the cases. In comparison with D&C, our algorithm leads substantially except in the cases where a large number of clusters exist (Type 6, 10 and 12). However, the timing of D&C in those cases is somewhat misleading, because we obtained error messages. Sorensen-Tang [25] and Gu-Eisenstat [11] reported different approaches to achieve authogonality of eigenvectors for D&C algorithm. The version of D&C code TREEQL we used does not have this feature.

6. Discussions.

6.1. Rank two vs. rank one tearing.

For the matrix

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{pmatrix}, \qquad \begin{array}{l} \beta_i \neq 0, \\ i = 1, \cdots, n-1, \end{array} \tag{6.1}$$

Our split-merge process is similar to Cuppen's divide-and-conquer method [4], where T is split by a rank one tearing:

$$T^{(1)} \equiv \begin{pmatrix} T_0^{(1)} & 0 \\ 0 & T_1^{(1)} \end{pmatrix} = T - P^{(1)} \text{ for } P^{(1)} = \begin{pmatrix} & \vdots & \\ \cdots & \theta \quad \beta_k & \cdots \\ & \beta_k \quad \frac{\beta_k^2}{\theta} & \\ & \vdots & \end{pmatrix}. \tag{6.2}$$

In our algorithm, we split T by a rank two tearing:

$$T^{(2)} \equiv \begin{pmatrix} T_0^{(2)} & 0 \\ 0 & T_1^{(2)} \end{pmatrix} = T - P^{(2)} \text{ for } P^{(2)} = \begin{pmatrix} & \vdots & \\ \cdots & 0 \quad \beta_k & \cdots \\ & \beta_k \quad 0 & \\ & \vdots & \end{pmatrix}. \tag{6.3}$$

For $\theta \neq 0$, the matrix $P^{(1)}$ is of rank one, and the rank of $P^{(2)}$ is two. (Throughout this section, the superscripts $^{(1)}$ and $^{(2)}$ are used to indicate the rank one and rank two tearings respectively.) There are some interesting properties of Cuppen's rank one tearing such as a separation result stronger than Theorem 3.1 ([9], Theorem 8.6.2, p462). However, if we consider the distance from the eigenvalues of perturbed matrices $T^{(1)}$ or $T^{(2)}$ to the corresponding eigenvalues of T, our rank two tearing is better than the rank one tearing, because we can always apply Laguerre's iteration from closer starting points.

LEMMA 6.1 ([3] p34). *Let A and B be real symmetric matrices with eigenvalues $\lambda_1(A) \leq \cdots \leq \lambda_n(A)$ and $\lambda_1(B) \leq \cdots \leq \lambda_n(B)$ respectively. Then*

$$\max_j |\lambda_j(A) - \lambda_j(B)| \leq \|A - B\|.$$

LEMMA 6.2. *For real symmetric matrices A and B, let $\{\lambda_i(A)\}_{i=1}^n$, $\{\lambda_i(B)\}_{i=1}^n$ and $\{\lambda_i(A-B)\}_{i=1}^n$ be eigenvalues of A, B and $A-B$ (in ascending order) respectively. Then*

$$\sum_{i=1}^n |\lambda_i(A) - \lambda_i(B)| \leq \sum_{i=1}^n |\lambda_i(A - B)|.$$

Proof. Use Theorem 9.7 in [3] p45 with the Ky Fan n-norm ([3], p28). □

The following theorem compares the rank one and rank two perturbations as in (6.2) and (6.3). It shows that, the upper bound for rank two perturbation on each eigenvalue is no more than one half of the upper bound for rank one perturbation; and the total perturbation of rank two tearing is bounded above by the *lower* bound of total perturbation of rank one tearing.

THEOREM 6.3. *Let T, $T^{(1)}$ and $T^{(2)}$ be as in (6.1), (6.2) and (6.3) with spectrums $\{\lambda_i\}_{i=1}^n$, $\{\lambda_i^{(1)}\}_{i=1}^n$ and $\{\lambda_i^{(2)}\}_{i=1}^n$ respectively in ascending order. Then*

$$|\lambda_i - \lambda_i^{(1)}| \leq \frac{\theta^2 + \beta_k^2}{|\theta|} \quad i = 1, \cdots, n \tag{6.4}$$

$$|\lambda_i - \lambda_i^{(2)}| \leq |\beta_k| \leq \frac{1}{2}\frac{\theta^2 + \beta_k^2}{|\theta|} \quad i = 1, \cdots, n \tag{6.5}$$

$$\sum_{i=1}^n |\lambda_i - \lambda_i^{(2)}| \leq 2|\beta_k| \leq \sum_{i=1}^n |\lambda_i - \lambda_i^{(1)}|. \tag{6.6}$$

Proof. The spectrums of $P^{(1)}$ and $P^{(2)}$ are $\left\{0, \frac{\theta^2+\beta_k^2}{\theta}\right\}$ and $\{0, \pm\beta_k\}$ respectively. By Lemma 6.1, for each $1 \leq i \leq n$,

$$|\lambda_i - \lambda_i^{(1)}| \leq \|P^{(1)}\| = \frac{\theta^2 + \beta_k^2}{|\theta|}, \quad |\lambda_i - \lambda_i^{(2)}| \leq \|P^{(2)}\| = |\beta_k|.$$

It is easy to see that

$$\min_{\theta>0} \frac{\theta^2 + \beta_k^2}{\theta} = 2|\beta_k|.$$

Thus,

$$|\lambda_i - \lambda_i^{(2)}| \leq |\beta_k| \leq \frac{1}{2}\frac{\theta^2 + \beta_k^2}{|\theta|}.$$

It follows from Lemma 6.2,

$$\sum_{i=1}^{n} |\lambda_i - \lambda_i^{(2)}| \leq 2|\beta_k|.$$

On the other hand,

$$\begin{aligned}
&\sum_{i=1}^{n} |\lambda_i - \lambda_i^{(1)}| \geq |\sum_{i=1}^{n} (\lambda_i - \lambda_i^{(1)})| \\
&= |\sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \lambda_i^{(1)}| = |tr(T) - tr(T^{(1)})| = |tr(P^{(1)})| = |\theta| + \frac{\beta_k^2}{|\theta|} \geq 2|\beta_k|
\end{aligned}$$

where $tr(A)$ denotes the trace of a matrix A. □

From this theorem, one can easily see that to reach the eigenvalue λ_i of T by using $\lambda_i^{(1)}$ of $T^{(1)}$ as a starting point more work is expected than starting from $\lambda_i^{(2)}$ of $T^{(2)}$ instead. Indeed, numerical testing shows that for rank one perturbation, $\sum_{i=1}^{n} |\lambda_i - \lambda_i^{(1)}|$ can be four times as big as $\sum_{i=1}^{n} |\lambda_i - \lambda_i^{(2)}|$ in rank two perturbation. Table 6.1 lists the comparison of these two perturbations on 12 types of matrices given in §5 for $n = 99$.

In case of the singular value evaluation which is equivalent to the symmetric tridiagonal eigenvalue problem on matrices with zero diagonal entries, rank-two tearing has another advantage of keeping the structure of zero diagonal. Thus the eigenvalues of split matrices can be evaluated within tiny relative error [18].

6.2. On parallelization and vectorization. Our algorithm can be parallelized in a similar way as in D&C [7]. There are two levels of parallelism. First, when the matrix T is split in a tree as in Fig. 3.3, then solving the eigenproblem of each submatrix is independent of the others and thus can be done by one processor or a group of processors. Second, when solving the eigenproblem of a submatrix of order m, each one of the m eigenvalues can be evaluated separately by performing independent Laguerre's iteration. Thus if a group of processors are assigned to this submatrix, these independent Laguerre's iterations can be distributed to those processors. The combination of both levels makes our method fully parallel and an excellent candidate for advance architectures.

Our method is also suitable for vector computing. For instance, when m eigenvalues of a submatrix are being computed, the algorithm evaluates the determinants $\det(T - xI)$ and their derivatives at several starting points $x_1, \cdots, x_m$. This process can be performed with some do loops described in Fig. 6.1, where the i-loops are vector operations.

The implementation strategies and experimental results of the parallelization and vectorization of our method will be reported in a separate paper [26].

6.3. Other possible variations. Our algorithm presented in this paper mainly consists of a split-merge process and a carefully implemented Laguerre's iteration. The purpose of the split-merge process is to separate the eigenvalues of the target matrix T and obtain initial approximations to these eigenvalues. The Laguerre's iteration is used because of its superior global properties and rapid convergence. There are other possible substitutes for the elements of our method.

- *The use of rank one tearing in splitting.* It was shown in §6.1 that the rank one tearing makes larger perturbation to the spectrum and hence requires

```
for i = 1 : m
    initialize ξ_{i1}, η_{i0}, η_{i1}, ζ_{i0}, and ζ_{i1}
end for
for j = 2 : m
    for i = 1 : m
        ξ_{ij} = α_j − x_i − β²_{j−1} / ξ_{i,j−1}
        η_{ij} = 1/ξ_{ij} [ (α_j − x_i) η_{i,j−1} + 1 − ( β²_{j−1} / ξ_{i,j−1} ) η_{i,j−2} ]
        ζ_{ij} = 1/ξ_{ij} [ (α_j − x_i) ζ_{i,j−1} + η_{i,j−1} − ( β²_{i,j−1} / ξ_{i,j−1} ) ζ_{i,j−2} ]
    end for
end for
```

FIG. 6.1. *Vectorizable loops*

more iteration steps in merging. Nevertheless, its distinguished separation property and deflation potential may serve as a trade-off.

- *The use of Newton's iteration in merging.* Newton's iteration can certainly be used as a substitute for the Laguerre iteration in our method. Then the process will be slower in convergence but the cost-per-step become less. The major disadvantage of using Newton's iteration is the loss of the global monotonic convergence property. An efficient algorithm based on Newton's iteration may be developed if such difficulty can be resolved.
- *Acceleration of bisection.* Instead of splitting the matrix, one can split the spectrum by bisection/multisection using the Sturm sequence. After the eigenvalues are well separated, Laguerre's iteration can be used to achieve rapid convergence. This would be similar to the idea of [2]. The difficulties of this approach seem to be:
  (1) if the spectrum can not be well separated (e.g. when clusters exist), the algorithm would be reduced to bisection/multisection;
  (2) usually good initial points for the iteration are not available unless enough bisection steps have been taken.

7. Proofs of the theorems.

7.1. Proof of Theorem 2.3.

LEMMA 7.1. *Let $f(\lambda) \equiv \prod_{i=1}^{n}(\lambda - \lambda_i)$ be a polynomial with real zeros $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Then, for any real number x for which $f(x) \neq 0$, there is a parameter $\theta \neq x$ such that $L_{r+}(x)$ and $L_{r-}(x)$ defined in (2.15) are the two zeros of the quadratic function in y*

$$\phi(\theta, y) \equiv (x-y)^2 \sum_{i=1}^{n} \left(\frac{\theta - \lambda_i}{x - \lambda_i}\right)^2 - r(\theta - y)^2. \tag{7.1}$$

Proof. Let $\eta = x - y$ and $\mu = \theta - x$. Then

$$\sum_{i=1}^{n} \left(\frac{\theta - \lambda_i}{x - \lambda_i} \right)^2 = \mu^2 s_2 + 2\mu s_1 + n$$

where

$$s_1 = \frac{f'(x)}{f(x)}, \quad s_2 = \frac{f'(x)^2 - f(x)f''(x)}{f(x)^2}.$$

With these notations, $\phi(\theta, y) = 0$ can be rewritten as

$$\mu^2(\eta^2 s_2 - r) + 2\mu\eta(\eta s_1 - r) + \eta^2(n - r) = 0 \tag{7.2}$$

which can be considered a quadratic equation in $\mu = \theta - x$. By making the discriminant $\triangle$ of the equation (7.2) in μ zero, we have

$$[\eta(\eta s_1 - r)]^2 - (\eta^2 s_2 - r)\eta^2(n - r) = 0.$$

And consequently, for

$$\eta = \frac{n}{s_1 \pm \sqrt{\frac{n-r}{r}(ns_2 - s_1)}}$$

or

$$y = x - \eta = x - \frac{n}{s_1 \pm \sqrt{\frac{n-r}{r}(ns_2 - s_1)}} = L_{r\pm}(x)$$

equation (7.2) in μ has a double zero μ_*. It is clear that $\mu_* \neq 0$. Otherwise, $\eta = 0$ and thereby $L_{r\pm}(x) = x$, which leads to $f(x) = 0$. Thus $\phi(\theta, L_{r\pm}(x)) = 0$ for $\theta \neq x$. □

Proof of Theorem 2.3:

We shall only prove (2.16). Inequalities in (2.17) follows by a similar argument.

First of all, we show that there is always a unique zero $y(\theta)$ of equation (7.1) in $(x, \lambda_{m+r}]$ for every $\theta \neq x$. In fact, if $\theta < x$, then $\phi(\theta, x) = -r(\theta - y)^2 < 0$ and

$$\begin{aligned}
\phi(\theta, \lambda_{m+r}) &\geq \sum_{i=1}^{r} \left[\left(\frac{x - \lambda_{m+r}}{x - \lambda_{m+i}} \right)^2 (\theta - \lambda_{m+i})^2 - (\theta - \lambda_{m+r})^2 \right] \\
&> \sum_{i=1}^{r} \left[\left(\frac{\theta - \lambda_{m+r}}{\theta - \lambda_{m+i}} \right)^2 (\theta - \lambda_{m+i})^2 - (\theta - \lambda_{m+r})^2 \right] = 0.
\end{aligned}$$

Hence there is a zero of $\phi(\theta, \cdot)$ in (x, λ_{m+r}). This zero is unique because $\phi(\theta, \cdot)$ is quadratic. For $\theta > x$, equation (7.1) can be rewritten as

$$\left(\frac{\theta - y}{x - y} \right)^2 = \frac{\sum_{i=1}^{n} \left(\frac{\theta - \lambda_i}{x - \lambda_i} \right)^2}{r} \quad (> 0).$$

As a continuous function of y in (x,θ), $\left(\frac{\theta-y}{x-y}\right)^2$ is monotonically decreasing from $+\infty$ to 0. Thus there is a unique $y(\theta)\in(x,\theta)$ satisfying equation (7.1) for every $\theta > x$. If $\theta \le \lambda_{m+r}$, $y(\theta) < \theta \le \lambda_{m+r}$. If $\theta > \lambda_{m+r}$,

$$\begin{aligned}\phi(\theta,\lambda_{m+r}) &\ge \sum_{i=1}^{r}\left[\left(\frac{x-\lambda_{m+r}}{x-\lambda_{m+i}}\right)^2(\theta-\lambda_{m+i})^2-(\theta-\lambda_{m+r})^2\right]\\ &\ge \sum_{i=1}^{r}\left[(\theta-\lambda_{m+i})^2-(\theta-\lambda_{m+r})^2\right]\ge 0.\end{aligned}$$

Thus $y(\theta)\in(x,\lambda_{m+r}]$ since $\phi(\theta,x)<0$.

It is easy to verify that, $L_{r+}(x)$ is the closest zero of $\phi(\theta,y)$ to the right of x for some θ, that is, $y(\theta)$.

The inequality $L_+(x) < L_{r+}(x)$ for $r>1$ is trivial in case of $-\dfrac{f'(x)}{f(x)} > 0$. QED.

7.2. Proof of Theorem 3.1.

LEMMA 7.2 (Strict interlacing property). *Let A be an unreduced symmetric tridiagonal matrix. For $m = 1,2,\cdots,n$, let $A^{(m)}$ be the leading $m\times m$ principal minor, (i.e. $A^{(m)}$ consists of entries of A in the first m rows and m columns) with eigenvalues $\{\lambda_i^{(m)}\}_{i=1}^m$ in ascending order. Then*

$$\lambda_1^{(m+1)} < \lambda_1^{(m)} < \lambda_2^{(m+1)} < \lambda_2^{(m)} < \cdots < \lambda_m^{(m+1)} < \lambda_m^{(m)} < \lambda_{m+1}^{(m+1)}.$$

Proof. See [24], p131. □

LEMMA 7.3. *([24], p193) Let A and B be $n\times n$ symmetric matrices and $\lambda_i(A)$ and $\lambda_i(A+B)$ be the i-th smallest eigenvalues of A and $A+B$ respectively. Suppose B has π positive and ν negative eigenvalues. Then*

$$\lambda_{i-\nu}(A) \le \lambda_i(A+B) \le \lambda_{i+\pi}(A)$$

with the convention $\lambda_0(\cdot) = -\infty$ and $\lambda_{n+1}(\cdot) = +\infty$.

Proof of Theorem 3.1:

The matrix T_0 is the $k\times k$ leading principal minor of T. By the interlacing property given in Lemma 7.2, the spectrum of T_0 is contained in (λ_1,λ_n). We may permute the rows and columns of T in reverse order, so that the accordingly permuted T_1 is an $(n-k)\times(n-k)$ leading principal minor of permuted T. Hence the spectrum of T_1 is also contained in (λ_1,λ_n). It follows that $\lambda_1 < \hat\lambda_1$ and $\hat\lambda_n < \lambda_n$. Since $T = \hat T + P^{(2)}$ where $P^{(2)}$ is defined in (6.3) with $\|P^{(2)}\| = |\beta_k|$, by Weyl's Lemma, $|\lambda_j - \hat\lambda_j| \le |\beta_k|$. Hence $\lambda_1 - |\beta_k| \le \lambda_1 < \hat\lambda_1$ and $\hat\lambda_n + |\beta_k| \ge \lambda_n > \hat\lambda_n$. The eigenvalues of $P^{(2)}$ are 0 and $\pm\beta_k$, thus, by Lemma 7.3, $\hat\lambda_{i-1} \le \lambda_i \le \hat\lambda_{i+1}$ and $\lambda_{i-1} \le \hat\lambda_i \le \lambda_{i+1}$. It remains to show that $\hat\lambda_i \ne \lambda_{i-1}$, $\hat\lambda_i \ne \lambda_{i+1}$ and $\lambda_i \ne \hat\lambda_{i-1}$, $\lambda_i \ne \hat\lambda_{i+1}$. We shall prove $\hat\lambda_i \ne \lambda_{i-1}$. The remaining cases follow by similar arguments.

Consider the homotopy

$$\begin{aligned}h(\lambda,t) &= \det[(1-t)\hat T + tT - \lambda I]\\ &= \det[T(t) - \lambda I]\end{aligned}$$

where $T(\lambda) \equiv (1-t)\hat{T} + tT$. Let $\lambda_i(t)$ denote the i-th smallest eigenvalue of $T(t)$. Then every $\lambda_i(t)$ is a continuous function of t. For $t \neq 0$, $T(t)$ is unreduced, so $\lambda_1(t) < \cdots < \lambda_n(t)$, and the graphs of $\lambda_i(t)$'s are disjoint except possibly at $t = 0$.

By the three term recurrence (2.4), it is easy to see that $h(\lambda, t) = t^2\beta_k^2 f_1(\lambda) + f_2(\lambda)$ where f_1 and f_2 are polynomials in λ. Suppose $\hat{\lambda}_i = \lambda_{i-1} = \lambda_*$, then $h(\lambda_i(0), 0) = h(\lambda_{i-1}(1), 1) = 0$ which implies $f_1(\lambda_*) = f_2(\lambda_*) = 0$. Thus the constant function $\lambda(t) \equiv \lambda_*$ satisfies $h(\lambda(t), t) = 0$ for all t. This constant function $\lambda(t)$ must be one of $\{\lambda_j(t)\}_{j=1}^n$. Since $\lambda(1) = \lambda_* = \lambda_{i-1}(1)$, so $\lambda(t) \equiv \lambda_{i-1}(t)$ and hence $\lambda_{i-1}(t) \equiv \lambda_* = \hat{\lambda}_i$. Accordingly, $\lambda_{i-1}(0) = \hat{\lambda}_{i-1} = \hat{\lambda}_i$, or $\hat{\lambda}_i$ is a double eigenvalue of $\hat{T}$. Since both T_0 and T_1 are unreduced, they must have the same eigenvalue λ_* at which $f_1(\lambda_*) = 0$.

Now, by differentiating

$$h(\lambda, t) = t^2\beta_k^2 f_1(\lambda) + f_2(\lambda)$$

$$= \det \begin{pmatrix} & & & | & & \\ & T_0 - \lambda I & & | & & \\ & & & | & t\beta_k & \\ - \; - - - - - & - - & + & - - & - - - - - & - \\ & & t\beta_k & | & & \\ & & & | & & T_1 - \lambda I \\ & & & | & & \end{pmatrix}$$

twice with respect to t, one can verify that $f_1(\lambda)$ is the characteristic polynomial of the $(n-2) \times (n-2)$ matrix $\begin{pmatrix} \tilde{T}_0 & 0 \\ 0 & \tilde{T}_1 \end{pmatrix}$ where $\tilde{T}_0$ is the leading $(k-1) \times (k-1)$ minor of T_0, and $\tilde{T}_1$ equals to the submatrix T_1 with the first row and the first column deleted. By the interlacing property, $f_1(\lambda_*) \neq 0$ since λ_* is an eigenvalue of both T_0 and T_1 and thus cannot be an eigenvalue of either $\tilde{T}_0$ or $\tilde{T}_1$. This contradiction proves $\hat{\lambda}_i \neq \lambda_{i-1}$.

Acknowledgment. We would like to acknowledge the valuable suggestions from Zhaojun Bai, James Demmel, Stanley Eisenstat, Kuiyuan Li, Beresford Parlett and anonymous referees.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorenson, *LAPACK User's Guide*, SIAM, Philadelphia, 1992.

[2] H. Bernstein, *An accelerated bisection method for the calculation of eigenvalues of symmetric tridiagonal matrix.* Computer Science Dept., Technical Report 79, Courant Institute, New York, NY, 1983.

[3] R. Bhatia, *Perturbation Bounds for Matrix Eigenvalues*, John Wiley & and Sons, Inc, New York, 1987.

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

[5] J. Demmel, J. Du Croz, S. Hammarling, and D. Sorensen, *Guidelines for the design of symmetric eigenroutines, svd, and iterative refinement and condition estimation for linear systems*, ANL, MCS-TM-111, (LAPACK Working Note 4), (1988).

[6] J. Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Stat. Comput., 11 (1988), pp. 873–912.

[7] J. J. Dongarra and D. C. Sorensen, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 139–154.

[8] J. G. F. Francis, *The QR transformation, parts I and II*, Computer J., 4 (1961 and 1962), pp. 265–271 and 332–345.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations, 2nd Ed.*, The John Hopkins University Press, Baltimore, MD, 1989.
[10] R. T. Gregory and D. L. Karney, *A Collection of Matrices for Testing Computational Algorithms*, Robert E. Krieger Publishing Company, Huntington, New York, 1978.
[11] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem.* Research Report YALEU/DCS/RR-916, Dept of Comput. Sci., Yale University, 1992.
[12] L. J. Huang, *Parallel homotopy algorithm for large sparse symmetric eigenproblems.* Ph.D. Thesis, Michigan State University, 1992.
[13] E. R. Jessup, *Parallel solution of the symmetric tridiagonal eigenproblem.* Ph.D. Thesis, Yale University, 1989.
[14] E. R. Jessup and I. C. F. Ipsen, *Improving the accuracy of inverse iteration*, SIAM J. Sci. Stat. Comput., 13 (1992).
[15] W. Kahan, *Accurate eigenvalues of a symmetric tridiagonal matrix.* Tech. Report No. CS41, Computer Science Department, Stanford University, Stanford, CA (1966) (revised June 1968).
[16] K. Li and T. Y. Li, *An algorithm for symmetric tridiagonal eigenproblems —- divide and conquer with homotopy continuation.* to appear: SIAM J. Sci. Stat. Comput.
[17] T. Y. Li and N. H. Rhee, *Homotopy algorithm for symmetric eigenvalue problems*, Numer. Math., 55 (1989), pp. 265–280.
[18] T. Y. Li, N. H. Rhee, and Z. Zeng, *An efficient and accurate parallel algorithm for the singular value problem of bidiagonal matrices.* preprint, submitted to Numerische Mathematik.
[19] T. Y. Li and Z. Zeng, *Homotopy-determinant algorithm solving nonsymmetric eigenvalue problems*, Math. Comp., 59 (1992), pp. 483–502.
[20] T. Y. Li, Z. Zeng, and L. Cong, *Solving eigenvalue problems of real nonsymmetric matrices with real homotopies*, SIAM J. Numer. Anal., 29 (1992), pp. 229–248.
[21] T. Y. Li, H. Zhang, and X. H. Sun, *Parallel homotopy algorithm for symmetric tridiagonal eigenvalue problems*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 464–485.
[22] S.-S. Lo, B. Phillipe, and A. Sameh, *A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problems*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 155–165.
[23] B. N. Parlett, *Laguerre's method applied to the matrix eigenvalue problem*, Math. Comp., 18 (1964), pp. 464–485.
[24] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[25] D. C. Sorensen and P. P. T. Tang, *On the orthogonality of eigenvectors computed by divide-and-conquer techniques*, SIAM. J. Numer. Anal., 28 (1991).
[26] C. Trefftz, C. C. Huang, P. McKinley, T. Y. Li, and Z. Zeng, *A scalable eigenvalue solver for symmetric tridiagonal matrices.* priprint, Michigan State University (1993).
[27] R. C. Ward, *The QR algorithm and Hyman's method on vector computers*, Math. Comp., 30 (1976), pp. 132–142.
[28] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.
[29] Z. Zeng, *Homotopy-determinant method for solving matrix eigenvalue problems and its parallelizations.* Ph.D. Thesis, Michigan State University, 1991.

		n=99	n=199	n=499
	S-M	1.27 (0.52)	4.13 (1.55)	23.4 (8.18)
Type 1	B/M	3.40 (1.26)	12.4 (4.24)	73.5 (24.5)
$a = 4$	RFQR	0.76	2.21	12.5
$b = 1$	D&C	8.56	59.1	910.
	S-M	1.51 (0.66)	5.33 (1.94)	31.2 (10.5)
Type 2	B/M	3.30 (1.31)	12.3 (4.27)	73.5 (24.5)
$a = 4$	RFQR	0.69	2.28	12.6
$b = 1$	D&C	8.15	55.7	851.
	S-M	1.26 (0.61)	4.25 (1.61)	23.8 (8.49)
Type 3	B/M	3.26 (1.26)	12.2 (4.26)	72.4 (24.4)
$a = 4$	RFQR	0.69	2.05	11.5
$b = 1$	D&C	9.05	62.3	905.
	S-M	1.29 (0.56)	4.35 (1.64)	24.9 (8.55)
Type 4	B/M	3.43 (1.32)	12.8 (4.56)	77.4 (26.1)
	RFQR	0.71	2.30	13.0
	D&C	8.99	62.3	906.
	S-M	1.32 (0.56)	4.39 (1.67)	25.0 (8.85)
Type 5	B/M	3.38 (1.28)	12.5 (4.23)	74.8 (23.4)
	RFQR	0.70	2.31	12.7
	D&C	8.92	62.2	893.
	S-M	0.82 (0.34)	2.13 (0.79)	9.11 (3.14)
Type 6	B/M	2.21 (0.72)	7.13 (2.45)	40.7 (13.4)
	RFQR	0.68	2.05	10.0
	D&C	2.39	6.89	46.9
	S-M	1.07 (0.50)	3.71 (1.25)	15.9 (5.17)
Type 7	B/M	3.39 (1.26)	12.4 (4.39)	74.5 (25.4)
	RFQR	0.86	2.58	15.0
	D&C	7.27	23.7	141.
	S-M	1.47 (0.59)	5.12 (1.81)	29.1 (9.96)
Type 8	B/M	3.43 (1.32)	12.8 (4.43)	76.8 (26.0)
	RFQR	0.71	l2.4	12.0
	D&C	8.99	62.5	848.
	S-M	1.02 (0.46)	2.95 (1.21)	16.1 (5.82)
Type 9	B/M	1.98 (0.98)	7.00 (3.39)	40.6 (20.2)
	RFQR	0.45	1.30	6.61
	D&C	1.48	6.96	73.6
	S-M	0.51 (0.25)	0.57 (0.34)	2.53 (1.47)
Type 10	B/M	0.25 (0.27)	0.37 (0.39)	0.60 (0.79)
	RFQR	0.59	1.83	9.94
	D&C	0.94	4.06	41.9
	S-M	1.43 (0.63)	5.00 (1.81)	28.4 (9.39)
Type 11	B/M	3.38 (1.31)	12.7 (4.44)	75.4 (25.4)
	RFQR	0.72	2.17	12.2
	D&C	9.01	62.2	845.
	S-M	0.30 (0.28)	0.35 (0.33)	0.65 (0.69)
Type 12	B/M	0.26 (0.29)	0.32 (0.46)	0.60 (0.77)
	RFQR	0.53	1.47	7.59
	D&C	0.85	3.90	39.1

TABLE 5.3

Execution time (in seconds) for evaluating all eigenvalues without computing eigenvectors: figures outside parentheses are the number of seconds for evaluating all eigenvalues. The figures inside parentheses represent the times for the corresponding codes to evaluate 1/3 of the largest eigenvalues. Neither D&C nor QR can take advantage of this case and thus has no such time listed. The D&C code TREEQL must evaluate all eigenvectors in order to evaluate the eigenvalues.

		n=99	n=199	n=499
	S-M	3.45 (1.32)	13.6 (4.77)	116. (45.0)
Type 1	B/M	5.55 (2.02)	21.6 (7.38)	164. (60.9)
$a = 4$	D&C	8.56	59.1	910.
$b = 1$	QR	15.2	112.	1744
	S-M	3.73 (1.40)	14.8 (5.10)	124. (47.2)
Type 2	B/M	5.48 (2.00)	21.6 (7.45)	165. (60.8)
$a = 4$	D&C	8.15	55.7	851.
$b = 1$	QR	15.2	115.	1763
	S-M	3.50 (1.36)	14.0 (4.96)	202. (68.8)
Type 3	B/M	5.65 (2.05)	21.8 (7.48)	249. (84.1)
$a = 4$	D&C	9.05	62.3	905.
$b = 1$	QR	14.0	104.	1582
	S-M	3.55 (1.37)	13.4 (4.69)	81.7 (27.5)
Type 4	B/M	11.4 (2.75)	67.6 (12.6)	846. (124.)
	D&C	8.99	62.3	906.
	QR	14.9	115.	1815
	S-M	3.72 (1.39)	14.1 (5.34)	127. (72.7)
Type 5	B/M	11.7 (12.6)	67.7 (12.6)	843. (123.)
	D&C	8.92	62.2	892.
	QR	15.0	115.	1781
	S-M	3.05 (1.13)	11.3 (3.82)	65.6 (21.9)
Type 6	B/M	4.34 (1.50)	16.1 (5.45)	95.6 (31.9)
	D&C	2.39*	6.89*	46.9*
	QR	14.1	105.	1478
	S-M	3.38 (1.29)	13.1 (4.40)	75.1 (24.7)
Type 7	B/M	5.64 (2.10)	21.3 (7.36)	131. (44.0)
	D&C	error	error	error
	QR	17.4	135.	2108
	S-M	3.78 (1.40)	14.5 (4.97)	87.3 (28.9)
Type 8	B/M	5.61 (2.07)	21.8 (7.41)	132. (44.1)
	D&C	8.99	62.5	849.
	QR	15.3	121.	1818
	S-M	7.35 (1.75)	45.2 (7.30)	610. (76.4)
Type 9	B/M	7.95 (2.19)	45.2 (9.36)	601. (83.7)
	D&C	error	error	error
	QR	12.0	93.4	1265
	S-M	8.31 (1.79)	55.6 (8.30)	784. (99.4)
Type 10	B/M	8.16 (1.67)	55.1 (8.28)	771. (98.1)
	D&C	0.94*	4.06*	41.9*
	QR	13.1	100.	1906
	S-M	3.74 (1.41)	14.4 (5.01)	86.8 (28.9)
Type 11	B/M	5.61 (2.07)	21.4 (7.33)	130. (43.6)
	D&C	9.01	62.2	845.
	QR	15.2	121.	1906
	S-M	8.13 (1.69)	55.0 (8.30)	776. (99.1)
Type 12	B/M	8.56 (1.70)	56.2 (8.61)	769. (99.5)
	D&C	0.85*	3.90*	39.1*
	QR	11.3	80.7	1186

TABLE 5.4

Execution time (in seconds) for evaluating both eigenvalues and eigenvectors: the figures outside parentheses are the number of seconds for evaluating all eigenpairs. The figures in parentheses represent the times for the corresponding codes to evaluate eigenpairs of the largest 1/3 eigenvalues. Neither D&C nor QR can take advantage of this case and thus has no such time listed.
**: error messages output when executing the D&C code TREEQL*

matrix types	rank two $\sum_i \lvert\lambda_i - \lambda_i^{(2)}\rvert$	rank one $\sum_i \lvert\lambda_i - \lambda_i^{(1)}\rvert$	matrix types	rank two $\sum_i \lvert\lambda_i - \lambda_i^{(2)}\rvert$	rank one $\sum_i \lvert\lambda_i - \lambda_i^{(1)}\rvert$
Type 1	0.726965858	2.000000000	Type 7	0.151873922	0.455857515
Type 2	1.472573929	2.000000000	Type 8	0.451587126	0.805204125
Type 3	0.468625974	2.000000000	Type 9	2.3505×10^{-8}	3.9610×10^{-8}
Type 4	35.97841949	98.99494937	Type 10	$\approx 10^{-17}$	$\approx 10^{-17}$
Type 5	1780.795116	4900.000000	Type 11	0.229865425	0.397815459
Type 6	0.758494678	2.000000000	Type 12	$\approx 10^{-17}$	$\approx 10^{-17}$

TABLE 6.1
Numerical comparison between rank one and rank two perturbations