Abstract

We present a stable and efficient divide-and-conquer algorithm for computing the eigendecomposition of an $N \times N$ symmetric tridiagonal matrix. It is based on a new, backward stable scheme for finding the eigendecomposition of a symmetric arrowhead matrix. Numerical results show that this algorithm is competitive with the QR algorithm, bisection with inverse iteration and Cuppen's divide-and-conquer algorithm for solving the symmetric tridiagonal eigenproblem. We also show how to use the fast multipole method of Carrier, Greengard and Rokhlin to speed up this algorithm from $O(N^2)$ time to $O(N \log_2 N)$ time for computing all the eigenvalues, and from $O(N^3)$ time to $O(N^2)$ time for computing all the eigenvalues and eigenvectors.

A Divide-And-Conquer Algorithm For The Symmetric Tridiagonal Eigenproblem[†]

Ming Gu and Stanley C. Eisenstat

Research Report YALEU/DCS/RR-932
27 November 1992

[†] This research was supported in part by U. S. Army Research Office contract DAAL03-91-G-0032.

1. Introduction

Given an $N \times N$ symmetric tridiagonal matrix

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{N-1} & \alpha_{N-1} & \beta_N \\ & & & \beta_N & \alpha_N \end{pmatrix} ,$$

the *symmetric tridiagonal eigenproblem* is to find the eigendecomposition:

$$T = X \Lambda X^T \quad ,$$

where Λ is diagonal and X is orthonormal. The diagonal elements of Λ are the eigenvalues of T, and the columns of X are the corresponding eigenvectors. This is a basic problem in numerical linear algebra [12, 24, 28, 29]. In this paper, we propose a divide-and-conquer algorithm, *arrowhead divide-and-conquer* (*ADC*), for solving this problem.

ADC divides[1] T into two smaller symmetric tridiagonal matrices T_1 and T_2, each of which is a principle submatrix of T. It then recursively computes the eigendecompositions of T_1 and T_2, and computes an orthonormal matrix Q such that $T = QHQ^T$, where H is a symmetric arrowhead matrix

$$H = \begin{pmatrix} \alpha & z^T \\ z & D \end{pmatrix} \quad ,$$

with D a diagonal matrix and z a vector. It computes the eigenvalues of T by computing the eigendecomposition

$$H = U \Lambda U^T \quad ,$$

where U is an orthonormal matrix[2]. It then computes the eigenvector matrix of T as QU (see Section 2).

Since error is associated with computation, a *numerical eigendecomposition* of T or H is usually defined as a decomposition of the form

$$T = \hat{X}\hat{\Lambda}\hat{X}^T + O(\epsilon \, \|T\|_2) \quad \text{or} \quad H = \hat{U}\hat{\Lambda}\hat{U}^T + O(\epsilon \, \|H\|_2) \quad , \tag{1.1}$$

where ϵ is the machine precision; $\hat{\Lambda}$ is diagonal; and $\hat{X}$ or $\hat{U}$ is *numerically orthonormal*. An algorithm that produces such a decomposition is said to be *backward stable* [28].

While the eigenvalues of T and H are always well-conditioned with respect to a symmetric perturbation, the eigenvectors can be extremely sensitive to such perturbations [12, 24, 28, 29]. That is, $\hat{\Lambda}$ can be guaranteed to be close to Λ, but $\hat{X}$ and $\hat{U}$ can be very different from X and U, respectively. Thus one is usually content with backward stable algorithms for computing the eigendecompositions of T and H.

[1] This dividing strategy has previously appeared in [1, 3, 13, 15].

[2] Since T and H are similar, they have the same eigenvalues.

The problem of computing the eigendecomposition of a symmetric arrowhead matrix is an interesting problem in its own right (see [3, 4, 23, 25, 26] and the references therein). Several schemes for solving this problem have been proposed [3, 23, 26]. While they can compute the eigenvalues to high absolute accuracy, in the presence of close eigenvalues they can have difficulties in computing numerically orthonormal eigenvectors, unless extended precision arithmetic is used [21, 27]. In this paper we present a new scheme for computing the eigendecomposition of a symmetric arrowhead matrix. It is similar to previous schemes in finding the eigenvalues, but it uses a completely different method in finding the eigenvectors, one that is backward stable. The amount of work is roughly the same as for previous schemes, yet it does not require the use or simulation of extended precision arithmetic. Since it uses this scheme, *ADC* is backward stable as well.

ADC computes all the eigenvalues of T in $O(N^2)$ time, and both the eigenvalues and eigenvectors of T in $O(N^3)$ time. We show that by using the fast multipole method of Carrier, Greengard and Rokhlin [9, 14], *ADC* can be accelerated to compute all the eigenvalues of a symmetric tridiagonal matrix in $O(N \log_2 N)$ time, and both the eigenvalues and eigenvectors in $O(N^2)$ time. These asymptotic time requirements are better than the corresponding worst-case time requirements ($O(N^2)$ and $O(N^3)$) for the QR algorithm [7, 12] and bisection with inverse iteration [12, 19, 20]. This is an important advantage of *ADC* for large matrices. Our scheme for finding all the eigenvalues of H takes $O(N^2)$ time, as do previous schemes [3, 23, 26]. By using the fast multipole method, it can be accelerated to compute all the eigenvalues of H in $O(N)$ time.

Cuppen's divide-and-conquer algorithm (*CDC*) [10, 11] uses a dividing strategy similar to that of *ADC* to compute the eigendecomposition of T, but it reduces T to a symmetric rank-one modification to a diagonal matrix, rather than to an arrowhead matrix. *CDC* is considered one of the fastest algorithms for solving the symmetric tridiagonal eigenproblem in both sequential and parallel settings [10, 11, 12, 18, 19], but it can be unstable [5, 10, 11], unless extended precision arithmetic is used [21, 27]. In contrast, *ADC* is backward stable, and our implementation is roughly twice as fast as existing implementations of *CDC* (TREEQL [11]) for relatively large matrices ($N \geq 256$), due to differences in how deflation is implemented (see Sections 4 and 6).

The occasional instability of *CDC* can be overcome without extended precision arithmetic by using our techniques in [17]; the deflation procedure for *ADC* can also be used in *CDC*, as can the fast multipole method. We have implemented only *ADC*. Our implementation shows that *ADC* is also very competitive with the QR algorithm [7, 12] and bisection with inverse iteration [12, 19, 20].

We take the usual model of arithmetic[3]

$$fl(x \circ y) = (x \circ y)\,(1 + \xi) \quad ,$$

[3] This model excludes machines like CRAYs and CDC Cybers that do not have a guard digit. *ADC* can easily be modified for such machines.

where x and y are floating point numbers; $\circ$ is one of $+, -, \times$, and $\div$; $fl(x \circ y)$ is the floating point result of the operation $\circ$; and $|\xi| \leq \epsilon$. We also require that

$$fl(\sqrt{x}) = \sqrt{x}\,(1+\xi)$$

for any positive floating point number x. For simplicity, we ignore the possibility of overflow and underflow.

Section 2 presents the dividing strategy used in *ADC*; Section 3 develops an efficient scheme for the eigendecomposition of a symmetric arrowhead matrix and shows that it is backward stable; Section 4 discusses the deflation procedure used in *ADC*; Section 5 discusses the application of the fast multipole method to speed up *ADC*; and Section 6 presents some numerical results.

2. "Dividing" the Matrix

Given a symmetric tridiagonal matrix T of size $N \times N$, *ADC* recursively divides T into two subproblems as follows:

$$T = \begin{pmatrix} T_1 & \beta_{k+1} e_k & 0 \\ \beta_{k+1} e_k^T & \alpha_{k+1} & \beta_{k+2} e_1^T \\ 0 & \beta_{k+2} e_1 & T_2 \end{pmatrix}, \qquad (2.1)$$

where T_1 and T_2 are $k \times k$ and $(N-k-1) \times (N-k-1)$ principle submatrices of T, respectively; and e_j is the j-th unit vector of appropriate dimension. Usually $k < N$ is taken to be $\lfloor N/2 \rfloor$.

Let $Q_i D_i Q_i{}^T$ be an eigendecomposition of T_i. Substituting these into (2.1), we get

$$\begin{aligned} T &= \begin{pmatrix} Q_1 D_1 Q_1{}^T & \beta_{k+1} e_k & 0 \\ \beta_{k+1} e_k^T & \alpha_{k+1} & \beta_{k+2} e_1^T \\ 0 & \beta_{k+2} e_1 & Q_2 D_2 Q_2{}^T \end{pmatrix} \\ &= \begin{pmatrix} 0 & Q_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & Q_2 \end{pmatrix} \begin{pmatrix} \alpha_{k+1} & \beta_{k+1} l_1^T & \beta_{k+2} f_2^T \\ \beta_{k+1} l_1 & D_1 & 0 \\ \beta_{k+2} f_2 & 0 & D_2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ Q_1{}^T & 0 & 0 \\ 0 & 0 & Q_2{}^T \end{pmatrix} \qquad (2.2) \\ &= Q U \Lambda U^T Q^T \\ &= X \Lambda X^T \end{aligned}$$

where l_1^T is the last row of Q_1; f_2^T is the first row of Q_2; Q is the first matrix in (2.2); and $U \Lambda U^T$ is an eigendecomposition of the middle arrowhead matrix. Thus T is reduced to an arrowhead matrix by the orthonormal similarity transformation Q.

ADC computes an eigendecomposition for the middle arrowhead matrix using the scheme described in Section 3. The eigenvalues of T are the diagonal elements of Λ, and the eigenvector matrix of T is obtained by computing the matrix-matrix product $X = QU$. To compute the eigendecompositions of T_1 and T_2, this process (equations (2.1) and (2.2)) can be recursively applied until the subproblems are sufficiently small. These small subproblems are then solved using the QR algorithm. There can be at most $O(\log_2 N)$ levels of recursion.

Equations (2.1) and (2.2) also suggest a recursion for computing only the eigenvalues. Let f_1^T be the first row of Q_1, and let l_2^T be the last row of Q_2. Suppose that D_i, f_i and l_i are given for $i = 1, 2$. Then after finding the eigendecomposition of the arrowhead matrix, the first row of X can be computed as $(0, f_1^T, 0)\ U$, and the last row of X can be computed as $(0, 0, l_2^T)\ U$. There is a similar recursion for CDC [10].

3. Computing the Eigendecomposition of a Symmetric Arrowhead Matrix

In this section we develop a stable and efficient scheme for finding the eigendecomposition of an $n \times n$ symmetric arrowhead matrix

$$H = \begin{pmatrix} \alpha & z^T \\ z & D \end{pmatrix} \quad ,$$

where $D = diag(d_2, \ldots, d_n)$ is a matrix of order $(n-1) \times (n-1)$, with $d_2 \leq d_3 \leq \ldots \leq d_n$; $z = (z_2, \ldots, z_n)^T$ is a vector of order $n-1$; and α is a scalar. The development closely parallels that in [17] for finding the eigendecomposition of a symmetric rank-one modification to a diagonal matrix.

We further assume that

$$d_{j+1} - d_j \geq \tau \|H\|_2 \quad \text{and} \quad |z_i| \geq \tau \|H\|_2 \quad , \tag{3.1}$$

where τ is a small multiple of ϵ to be specified later. Any symmetric arrowhead matrix can be reduced to one that satisfies these conditions by using the deflation procedure described in Section 4 and a simple permutation.

The following lemma characterizes the eigenvalues and eigenvectors of arrowhead matrices.

LEMMA 1 (WILKINSON [29]). *The eigenvalues* $\{\lambda_i\}_{i=1}^n$ *of* H *satisfy the* interlacing property

$$\lambda_1 < d_2 < \lambda_2 < \ldots < d_n < \lambda_n$$

and the secular equation

$$f(\lambda) = \lambda - \alpha + \sum_{j=2}^{n} \frac{z_j^2}{d_j - \lambda} = 0 \quad .$$

For each eigenvalue λ_i *of* H*, the corresponding eigenvector is given by*

$$u_i = \left(-1, \frac{z_1}{d_2 - \lambda_i}, \ldots, \frac{z_n}{d_n - \lambda_i}\right)^T \Bigg/ \sqrt{1 + \sum_{j=2}^{n} \frac{z_j^2}{(d_j - \lambda_i)^2}} \quad . \tag{3.2}$$

The following lemma allows us to construct an arrowhead matrix from its eigenvalues and its shaft.

LEMMA 2 (BOLEY AND GOLUB [6]). *Given a diagonal matrix* $D = \mathrm{diag}(d_2, \ldots, d_n)$ *and a set of numbers* $\{\hat{\lambda}_i\}_{i=1}^n$ *satisfying the interlacing property*

$$\hat{\lambda}_1 < d_2 < \hat{\lambda}_2 < \ldots < d_n < \hat{\lambda}_n \quad , \tag{3.3}$$

there exists an arrowhead matrix

$$\hat{H} = \begin{pmatrix} \hat{\alpha} & \hat{z}^T \\ \hat{z} & D \end{pmatrix}$$

whose eigenvalues are $\{\hat{\lambda}_i\}_{i=1}^n$. *The vector* $\hat{z} = (\hat{z}_2, \ldots, \hat{z}_n)^T$ *and the scalar* $\hat{\alpha}$ *are determined by*

$$|\hat{z}_i| = \sqrt{(d_i - \hat{\lambda}_1)\,(\hat{\lambda}_n - d_i) \prod_{j=2}^{i-1} \frac{(\hat{\lambda}_j - d_i)}{(d_j - d_i)} \prod_{j=i}^{n-1} \frac{(\hat{\lambda}_j - d_i)}{(d_{j+1} - d_i)}} \tag{3.4}$$

$$\hat{\alpha} = \hat{\lambda}_1 + \sum_{j=2}^{n} \left(\hat{\lambda}_j - d_j\right) \quad , \tag{3.5}$$

where the sign of $\hat{z}_i$ *can be chosen arbitrarily.*

3.1. Computing the Eigenvectors of H

For each *exact* eigenvalue λ_i, equation (3.2) gives the corresponding *exact* eigenvector. Observe that if λ_i was given *exactly*, then we could compute each difference, each ratio, each product and each sum in (3.2) to high relative accuracy, and thus compute u_i to componentwise high relative accuracy.

In practice we can only hope to compute an approximation $\hat{\lambda}_i$ to λ_i. But problems can arise if we approximate u_i by

$$\hat{u}_i = \left(-1, \frac{z_1}{d_2 - \hat{\lambda}_i}, \ldots, \frac{z_n}{d_n - \hat{\lambda}_i}\right)^T \Big/ \sqrt{1 + \sum_{j=2}^{n} \frac{z_j^2}{(d_j - \hat{\lambda}_i)^2}}$$

(i.e., replace λ_i by $\hat{\lambda}_i$ in (3.2), as in [3, 23]). For even if $\hat{\lambda}_i$ is close to λ_i, the approximate ratio $z_j/(d_j - \hat{\lambda}_i)$ can still be very different from the exact ratio $z_j/(d_j - \lambda_i)$, resulting in a unit eigenvector very different from u_i. And when all the approximate eigenvalues $\{\hat{\lambda}_i\}_{i=1}^n$ are computed and all the corresponding eigenvectors are approximated in this manner, the resulting eigenvector matrix may not be orthonormal.

Lemma 2 allows us to overcome this problem. After we have computed all the approximate eigenvalues $\{\hat{\lambda}_i\}_{i=1}^n$ of H, we can find a *new* matrix $\hat{H}$ whose *exact* eigenvalues are $\{\hat{\lambda}_i\}_{i=1}^n$, and then compute the eigenvectors of $\hat{H}$ using Lemma 1. Note that each difference, each product and each ratio in (3.4) can be computed to high relative accuracy. Thus $|\hat{z}_i|$ can be computed to high relative accuracy. The sign of $\hat{z}_i$ can be taken to be the sign of z_i. Substituting the *exact* eigenvalues $\{\hat{\lambda}_i\}_{i=1}^n$ and the computed $\hat{z}$ into equation (3.2), each eigenvector of $\hat{H}$ can be computed to component-wise high relative accuracy. Consequently, after all the eigenvectors are computed, the computed eigenvector matrix of $\hat{H}$ will be numerically orthonormal.

To ensure the existence of $\hat{H}$, the approximations $\{\hat{\lambda}_i\}_{i=1}^n$ must satisfy the interlacing property in Lemma 2. But since the exact eigenvalues of H satisfy the same interlacing property (see Lemma 1), this is only an accuracy requirement on the computed eigenvalues, and is not an additional restriction on H.

We can use the eigendecomposition of $\hat{H}$ as an approximation to that of H. Since

$$H = \begin{pmatrix} \alpha & z^T \\ z & D \end{pmatrix} = \hat{H} + \begin{pmatrix} \alpha - \hat{\alpha} & (z-\hat{z})^T \\ z - \hat{z} & 0 \end{pmatrix} \quad ,$$

we have

$$|\hat{\lambda}_i - \lambda_i| \le \|\hat{H} - H\|_2 \le |\alpha - \hat{\alpha}| + \|z - \hat{z}\|_2 \quad .$$

Thus such a substitution is backward stable (see (1.1)) as long as $\hat{\alpha}$ and $\hat{z}$ are close to α and z, respectively (cf. [17]).

3.2. Computing the Eigenvalues of H

In order to guarantee that $\hat{z}$ is close to z and $\hat{\alpha}$ is close to α, we must ensure that the approximations $\{\hat{\lambda}_i\}_{i=1}^n$ to the eigenvalues are sufficiently accurate. The key is the stopping criterion for the root-finder, which requires a slight reformulation of the secular equation (cf. [8, 17]).

Consider the eigenvalue $\lambda_i \in (d_i, d_{i+1})$, where $2 \le i \le n-1$; the cases $i = 1$ and $i = n$ are considered later. λ_i is a root of the secular equation

$$f(\lambda) = \lambda - \alpha + \sum_{j=2}^{n} \frac{z_j^2}{d_j - \lambda} = 0 \quad .$$

We first assume that[4] $\lambda_i \in (d_i, \frac{d_i+d_{i+1}}{2})$. Let $\alpha_i = d_i - \alpha$ and $\delta_j = d_j - d_i$, and let

$$\psi_i(\mu) \equiv \sum_{j=2}^{i} \frac{z_j^2}{\delta_j - \mu} \quad \text{and} \quad \phi_i(\mu) \equiv \sum_{j=i+1}^{n} \frac{z_j^2}{\delta_j - \mu} \quad .$$

Since

$$f(\mu + d_i) = \mu + \alpha_i + \psi_i(\mu) + \phi_i(\mu) \equiv g_i(\mu) \quad ,$$

we seek the root $\mu_i = \lambda_i - d_i \in (0, \delta_{i+1}/2)$ of $g_i(\mu) = 0$. Let $\hat{\mu}_i$ be the computed root, so that $\hat{\lambda}_i = d_i + \hat{\mu}_i$ is the computed eigenvalue.

[4] This can easily be checked by computing $f(\frac{d_i+d_{i+1}}{2})$. If $f(\frac{d_i+d_{i+1}}{2}) > 0$, then $\lambda_i \in (d_i, \frac{d_i+d_{i+1}}{2})$, otherwise $\lambda_i \in [\frac{d_i+d_{i+1}}{2}, d_{i+1})$.

An important property of $g_i(\mu)$ is that each difference $\delta_j - \mu$ can be evaluated to high relative accuracy for any $\mu \in (0, \delta_{i+1}/2)$. Indeed, since $\delta_i = 0$, we have $fl(\delta_i - \mu) = -fl(\mu)$; since $fl(\delta_{i+1}) = fl(d_{i+1} - d_i)$ and $0 < \mu < (d_{i+1} - d_i)/2$, we can compute $fl(\delta_{i+1} - \mu)$ as $fl(fl(d_{i+1} - d_i) - fl(\mu))$; and in a similar fashion, we can compute $\delta_j - \mu$ to high relative accuracy for any $j \neq i, i+1$.

Because of this property, each ratio $z_j^2/(\delta_j - \mu)$ in $g_i(\mu)$ can be evaluated to high relative accuracy for any $\mu \in (0, \delta_{i+1}/2)$. Moreover, α_i can be computed to high relative accuracy. Thus, since both $\psi_i(\mu)$ and $\phi_i(\mu)$ are sums of terms of the same sign, we can bound the error in computing $g_i(\mu)$ by

$$\eta n(|\mu| + |\alpha_i| + |\psi_i(\mu)| + |\phi_i(\mu)|) \quad ,$$

where η is a small multiple of ϵ that is independent of n and μ.

We now assume that $\lambda_i \in [\frac{d_i + d_{i+1}}{2}, d_{i+1})$. Let $\alpha_i = d_{i+1} - \alpha$ and $\delta_j = d_j - d_{i+1}$, and let

$$\psi_i(\mu) \equiv \sum_{j=2}^{i} \frac{z_j^2}{\delta_j - \mu} \quad \text{and} \quad \phi_i(\mu) \equiv \sum_{j=i+1}^{n} \frac{z_j^2}{\delta_j - \mu} \quad .$$

We seek the root $\mu_i = \lambda_i - d_{i+1} \in [\delta_i/2, 0)$ of the equation

$$g_i(\mu) \equiv f(\mu + d_{i+1}) = \mu + \alpha_i + \psi_i(\mu) + \phi_i(\mu) = 0 \quad .$$

Let $\hat{\mu}_i$ be the computed root, so that $\hat{\lambda}_i = d_{i+1} + \hat{\mu}_i$. For any $\mu \in [\delta_i/2, 0)$, each difference $\delta_j - \mu$ can again be computed to high relative accuracy, as can each ratio $z_j^2/(\delta_j - \mu)$ and the scalar α_i; and we can bound the error in computing $g_i(\mu)$ as before.

Next we consider the case $i = 1$. Let $\alpha_1 = d_2 - \alpha$ and $\delta_j = d_j - d_2$, and let

$$\psi_1(\mu) \equiv 0 \quad \text{and} \quad \phi_1(\mu) \equiv \sum_{j=2}^{n} \frac{z_j^2}{\delta_j - \mu} \quad .$$

We seek the root $\mu_2 = \lambda_1 - d_2 \in (-\|z\|_2 - |\alpha|, 0)$ of the equation

$$g_1(\mu) \equiv f(\mu + d_2) = \mu + \alpha_1 + \psi_1(\mu) + \phi_1(\mu) = 0 \quad .$$

Let $\hat{\mu}_1$ be the computed root, so that $\hat{\lambda}_1 = d_2 + \hat{\mu}_1$. For any $\mu \in (-\|z\|_2 - |\alpha|, 0)$, each ratio $z_j^2/(\delta_j - \mu)$ can be computed to high relative accuracy, as can α_1; and we can bound the error in computing $g_1(\mu)$ as before.

Finally we consider the case $i = n$. Let $\alpha_n = d_n - \alpha$ and $\delta_j = d_j - d_n$, and let

$$\psi_n(\mu) \equiv \sum_{j=2}^{n} \frac{z_j^2}{\delta_j - \mu} \quad \text{and} \quad \phi_n(\mu) \equiv 0 \quad .$$

We seek the root $\mu_n = \lambda_n - d_n \in (0, \|z\|_2 + |\alpha|)$ of the equation

$$g_n(\mu) \equiv f(\mu + d_n) = \mu + \alpha_n + \psi_n(\mu) + \phi_n(\mu) = 0 \quad .$$

Again, let $\hat{\mu}_n$ be the computed root, so that $\hat{\lambda}_n = d_n + \hat{\mu}_n$. For any $\mu \in (0, \|z\|_2 + |\alpha|)$, each ratio $z_j^2/(\delta_j - \mu)$ can be computed to high relative accuracy, as can α_n; and we can bound the error in computing $g_n(\mu)$ as before.

In practice the root-finder cannot make any progress at a point μ where it is impossible to determine the sign of $g_i(\mu)$ numerically. Thus we propose the stopping criterion

$$|g_i(\mu)| \leq \eta n \left(|\mu| + |\alpha_i| + |\psi_i(\mu)| + |\phi_i(\mu)|\right) \quad , \tag{3.6}$$

where, as before, $\eta n(|\mu| + |\alpha_i| + |\psi_i(\mu)| + |\phi_i(\mu)|)$ is an upper bound on the round-off error in computing $g_i(\mu)$. Note that for each i, there is at least one floating point number that satisfies this stopping criterion numerically, namely $fl(\mu_i)$.

We have not specified the scheme used to find the root of $g_i(\mu)$. We used a modified version of the rational interpolation strategy in [8] for the numerical experiments, but bisection and its variations [23, 26] or the improved rational interpolation strategies given in [13, 22] would also work. What is most important is the stopping criterion and the fact that, with the reformulation of the secular equation given above, we can find a μ that satisfies it.

3.3. Numerical Stability

In this subsection we show that $\hat{\alpha}$ and $\hat{z}$ are indeed close to α and z, respectively, as long as the root-finder guarantees that each $\hat{\mu}_i$ satisfies the stopping criterion (3.6).

Since $f(\lambda_i) = 0$, we have

$$|\alpha_i| = \left| -\mu_i - \sum_{j=2}^{n} \frac{z_j^2}{d_j - \lambda_i} \right| \leq |\mu_i| + \sum_{j=2}^{n} \frac{z_j^2}{|d_j - \lambda_i|} \quad ,$$

and (3.6) implies that the computed eigenvalue $\hat{\lambda}_i$ satisfies

$$|f(\hat{\lambda}_i)| \leq \eta n \left(|\mu_i| + |\hat{\mu}_i| + \sum_{j=2}^{n} \frac{z_j^2}{|d_j - \lambda_i|} + \sum_{j=2}^{n} \frac{z_j^2}{|d_j - \hat{\lambda}_i|} \right) \quad .$$

Since

$$f(\hat{\lambda}_i) = f(\hat{\lambda}_i) - f(\lambda_i) = (\hat{\lambda}_i - \lambda_i) \left(1 + \sum_{j=2}^{n} \frac{z_j^2}{(d_j - \hat{\lambda}_i)(d_j - \lambda_i)} \right) \quad ,$$

it follows that

$$\begin{aligned} |\hat{\lambda}_i - \lambda_i| & \left(1 + \sum_{j=2}^{n} \frac{z_j^2}{|(d_j - \hat{\lambda}_i)(d_j - \lambda_i)|} \right) \\ & \leq \eta n \left(|\mu_i| + |\hat{\mu}_i| + \sum_{j=2}^{n} \frac{z_j^2}{|d_j - \hat{\lambda}_i|} + \sum_{j=2}^{n} \frac{z_j^2}{|d_j - \lambda_i|} \right) \quad . \end{aligned} \tag{3.7}$$

Note that for any i and j,

$$|\mu_i| + |\hat{\mu}_i| \leq 4\|H\|_2 + |\hat{\lambda}_i - \lambda_i| \quad \text{and} \quad |d_j - \hat{\lambda}_i| + |d_j - \lambda_i| \leq 4\|H\|_2 + |\hat{\lambda}_i - \lambda_i| \quad .$$

Substituting these relations into (3.7), we get

$$|\hat{\lambda}_i - \lambda_i| \left(1 + \sum_{j=2}^{n} \frac{z_j^2}{|(d_j - \hat{\lambda}_i)(d_j - \lambda_i)|}\right)$$
$$\leq \eta n \left(4\|H\|_2 + |\hat{\lambda}_i - \lambda_i| + \sum_{j=2}^{n} \frac{(4\|H\|_2 + |\hat{\lambda}_i - \lambda_i|)z_j^2}{|d_j - \hat{\lambda}_i||d_j - \lambda_i|}\right) \quad ,$$

or

$$|\hat{\lambda}_i - \lambda_i| \leq \frac{4\eta n\|H\|_2}{1 - \eta n} \quad .$$

i.e., all the eigenvalues are computed to high absolute accuracy. Applying (3.5) of Lemma 2 to both H and $\hat{H}$, we have

$$\alpha = \lambda_1 + \sum_{j=2}^{n} (\lambda_j - d_j) \quad \text{and} \quad \hat{\alpha} = \hat{\lambda}_1 + \sum_{j=2}^{n} (\hat{\lambda}_j - d_j) \quad ,$$

and therefore

$$|\alpha - \hat{\alpha}| = \left|\sum_{j=1}^{n} (\lambda_j - \hat{\lambda}_j)\right| \leq \sum_{j=1}^{n} \left|\lambda_j - \hat{\lambda}_j\right| \leq \frac{4\eta n^2\|H\|_2}{1 - \eta n} \quad . \tag{3.8}$$

To show that $\hat{z}$ is close to z, we further note that for any i and j, we have

$$|\hat{\mu}_i| \leq |\mu_i| + |\hat{\lambda}_i - \lambda_i| \quad ,$$

and

$$\frac{1}{|d_j - \hat{\lambda}_i|} + \frac{1}{|d_j - \lambda_i|} \leq \frac{2}{|(d_j - \hat{\lambda}_i)(d_j - \lambda_i)|^{\frac{1}{2}}} + \frac{|\hat{\lambda}_i - \lambda_i|}{|(d_j - \hat{\lambda}_i)(d_j - \lambda_i)|} \quad .$$

Substituting these relations into (3.7) and using the Cauchy-Schwartz inequality, we get

$$|\hat{\lambda}_i - \lambda_i| \left(1 + \sum_{j=2}^{n} \frac{z_j^2}{|(d_j - \hat{\lambda}_i)(d_j - \lambda_i)|}\right)$$
$$\leq \frac{2\eta n}{1 - \eta n} \left(|\mu_i| + \sum_{j=2}^{n} \frac{z_j^2}{|(d_j - \hat{\lambda}_i)(d_j - \lambda_i)|^{\frac{1}{2}}}\right)$$
$$\leq \frac{2\eta n}{1 - \eta n} \sqrt{|\mu_i|^2 + \|z\|_2^2} \sqrt{1 + \sum_{j=2}^{n} \frac{z_j^2}{|(d_j - \hat{\lambda}_i)(d_j - \lambda_i)|}} \quad .$$

Since $|\mu_i|^2 + \|z\|_2^2 \leq 5\|H\|_2^2$, we have

$$|\hat{\lambda}_i - \lambda_i| \;\leq\; \frac{2\eta n}{1 - \eta n} \sqrt{|\mu_i|^2 + \|z\|_2^2} \Bigg/ \sqrt{1 + \sum_{j=2}^{n} \frac{z_j^2}{|(d_j - \hat{\lambda}_i)(d_j - \lambda_i)|}}$$

$$\leq \frac{2\sqrt{5}\eta n\|H\|_2}{(1-\eta n)|z_j|}\sqrt{|(d_j-\hat{\lambda}_i)(d_j-\lambda_i)|}$$
$$\leq \frac{2\sqrt{5}\eta n\|H\|_2}{(1-\eta n)|z_j|}\left(|d_j-\lambda_i|+\frac{1}{2}|\hat{\lambda}_i-\lambda_i|\right) \quad .$$

Letting $\beta_j = 2\sqrt{5}\eta n\|H\|_2/((1-\eta n)|z_j|)$, this implies that

$$|\hat{\lambda}_i-\lambda_i| \leq \frac{\beta_j}{1-\frac{1}{2}\beta_j}\,|d_j-\lambda_i| \tag{3.9}$$

for every $2 \leq j \leq n$, provided that $\beta_j < 2$.

Let $\hat{\lambda}_i - \lambda_i = \alpha_{ij}(d_j-\lambda_i)/z_j$ for all i and j. Suppose that we pick $\tau = 6\eta n^2$ in equation (3.1) of Section 3. Then $|z_j| \geq 6\eta n^2\|H\|_2$. Assume further that $\eta n < 1/100$. Then $\beta_j \leq 2/5$, and (3.9) implies that $|\alpha_{ij}| \leq \alpha \equiv 6\eta n\|H\|_2$ for all i and j. Thus

$$|\hat{z}_i| = \sqrt{\frac{\prod_{j=1}^n(\hat{\lambda}_j-d_i)}{\prod_{j=2,j\neq i}^n(d_j-d_i)}} = \sqrt{\frac{\prod_{j=1}^n(\lambda_j-d_i)\left(1+\frac{\alpha_{ji}}{z_i}\right)}{\prod_{j=2,j\neq i}^n(d_j-d_i)}} = |z_i|\sqrt{\prod_{j=1}^n\left(1+\frac{\alpha_{ji}}{z_i}\right)}$$

and, since $\hat{z}_i$ and z_i have the same sign,

$$\begin{aligned}|\hat{z}_i - z_i| &= |z_i|\left|\sqrt{\prod_{j=1}^n\left(1+\frac{\alpha_{ji}}{z_i}\right)}-1\right| \leq |z_i|\left(\left(1+\frac{\alpha}{|z_i|}\right)^{\frac{n}{2}}-1\right)\\ &\leq |z_i|\left(\exp\left(\frac{\alpha n}{2|z_i|}\right)-1\right) \leq (e-1)\,\alpha n/2\\ &\leq 6\eta n^2\|H\|_2 \quad ,\end{aligned} \tag{3.10}$$

where we have used the fact that $\alpha n/(2|z_i|) \leq 1$ and that $(e^x-1)/x \leq e-1$ for $0 < x \leq 1$.

One factor of n in τ and equations (3.8) and (3.10) comes from the stopping criterion (3.6). This is quite conservative and could be reduced to $\log_2 n$ by using a binary tree structure for summing up the terms in $\psi_i(\mu)$ and $\phi_i(\mu)$. The other factor of n comes from the upper bound for $\sum_{j=1}^n(\lambda_j-\hat{\lambda}_j)$ in (3.8) and $\prod_j(1+\alpha_{ji}/z_i)$ in (3.10). This also seems quite conservative. Thus we might expect the factor of n^2 in τ and equations (3.8) and (3.10) to be more like $O(n)$ in practice.

4. Deflation for *ADC*

4.1. Deflation for *H*

Consider the arrowhead matrix

$$H = \begin{pmatrix} \alpha & z^T \\ z & D \end{pmatrix} \quad ,$$

where $D = diag(d_2, \ldots, d_n)$ and $z = (z_2, \ldots, z_n)^T$. We now show that we can reduce H to an arrowhead matrix which further satisfies (cf. (3.1))

$$|d_i - d_j| \geq \tau \|H\|_2 \quad \text{for} \quad i \neq j, \qquad \text{and} \qquad |z_i| \geq \tau \|H\|_2 \quad ,$$

where τ is specified in Section 3.3. We illustrate the reductions for $n = 3$, $i = 3$ and $j = 2$. Similar reductions for a symmetric rank-one modification to a diagonal matrix appear in [8, 10, 11].

Assume that $|z_i| \leq \tau \|H\|_2$. Then

$$H = \begin{pmatrix} \alpha & z_2 & z_3 \\ z_2 & d_2 & \\ z_3 & & d_3 \end{pmatrix} = \begin{pmatrix} \alpha & z_2 & 0 \\ z_2 & d_2 & \\ 0 & & d_3 \end{pmatrix} + O(\tau \|H\|_2) \quad . \tag{4.1}$$

We perturb z_i to zero. Then H is perturbed by $O(\tau\|H\|_2)$. d_i is an approximate eigenvalue of H and is deflated. The $(n-1) \times (n-1)$ leading principle submatrix of the perturbed matrix is another arrowhead matrix with smaller dimensions. This deflation procedure is backward stable (see (1.1)).

Now assume that $|d_i - d_j| \leq \tau \|H\|_2$. Apply a Givens rotation to H as follows:

$$\begin{aligned} &\begin{pmatrix} 1 & & \\ & c & s \\ & -s & c \end{pmatrix} \begin{pmatrix} \alpha & z_2 & z_3 \\ z_2 & d_2 & \\ z_3 & & d_3 \end{pmatrix} \begin{pmatrix} 1 & & \\ & c & -s \\ & s & c \end{pmatrix} \\ &= \begin{pmatrix} \alpha & r & 0 \\ r & d_2c^2 + d_3s^2 & cs(d_3 - d_2) \\ 0 & cs(d_3 - d_2) & d_2s^2 + d_3c^2 \end{pmatrix} \\ &= \begin{pmatrix} \alpha & r & 0 \\ r & d_2c^2 + d_3s^2 & \\ 0 & & d_2s^2 + d_3c^2 \end{pmatrix} + O(\tau\|H\|_2) \quad , \end{aligned} \tag{4.2}$$

where $r = \sqrt{z_i^2 + z_j^2}$, $c = z_j/r$ and $s = z_i/r$. Similarly we perturb $cs(d_i - d_j)$ to zero. Then H is perturbed by $O(\tau\|H\|_2)$. $d_js^2 + d_ic^2$ is an approximate eigenvalue of H and can be deflated. The $(n-1) \times (n-1)$ leading principle submatrix of the matrix in (4.2) is another arrowhead matrix with smaller dimensions. This deflation procedure is again backward stable (see (1.1)).

4.2. Local Deflation for *ADC*

In the dividing strategy for *ADC* (see (2.2)), we write

$$\begin{aligned} T &= \begin{pmatrix} 0 & Q_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & Q_2 \end{pmatrix} \begin{pmatrix} \alpha_{k+1} & \beta_{k+1} l_1^T & \beta_{k+2} f_2^T \\ \beta_{k+1} l_1 & D_1 & 0 \\ \beta_{k+2} f_2 & 0 & D_2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ Q_1{}^T & 0 & 0 \\ 0 & 0 & Q_2{}^T \end{pmatrix} \\ &= (QU)\Lambda(QU)^T \quad , \end{aligned} \tag{4.3}$$

where Q is the first matrix in (4.3); l_1^T is the last row of Q_1; f_2^T is the first row of Q_2; and $U\Lambda U^T$ is an eigendecomposition of the middle arrowhead matrix.

Note that Q is a block matrix with some zero blocks. When we compute the matrix-matrix product QU, we would like to take advantage of this structure. Since the main cost of ADC is in computing such products, we get a speedup close to 2 by doing so. In this subsection we design a deflation procedure for ADC that takes advantage of this structure. This is not done in any current implementation of CDC.

If the vector $(\beta_{k+1}l_1^T, \beta_{k+2}f_2^T)$ has components with small absolute value, then we can apply reduction (4.1). The block structure of Q is preserved. If D_1 has two close diagonal elements, then we can apply reduction (4.2). The block structure is again preserved. We can do the same when D_2 has two close diagonal elements.

However, when D_1 has a diagonal element that is close to a diagonal element in D_2 and we apply reduction (4.2), the block structure of Q is changed. To illustrate, assume that after applying a permutation the first diagonal element of D_1 is close to the last diagonal element of D_2. Let $Q_1 = (q_1 \;\; \tilde{Q}_1)$ and $Q_2 = (\tilde{Q}_2 \;\; q_2)$; let

$$\beta_{k+1}l_1 = \begin{pmatrix} z_2 \\ \tilde{z}_1 \end{pmatrix} \quad \text{and} \quad \beta_{k+2}f_2 = \begin{pmatrix} \tilde{z}_2 \\ z_N \end{pmatrix} \quad ;$$

and let $D_1 = diag(d_2, \tilde{D}_1)$ and $D_2 = diag(\tilde{D}_2, d_N)$. By assumption, d_2 and d_N are close. Set $r = \sqrt{z_2^2 + z_N^2}$, $c = z_2/r$ and $s = z_N/r$. Similar to (4.2), we apply the Givens rotation[5]

$$G = \begin{pmatrix} 1 & & & \\ & c & & s \\ & & I_{N-3} & \\ & -s & & c \end{pmatrix}$$

to the middle matrix in (4.3) to rotate z_N to zero. This creates some non-zero elements in the second and N-th columns of Q:

$$T = \begin{pmatrix} 0 & q_1 & \tilde{Q}_1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \tilde{Q}_2 & q_2 \end{pmatrix} G^T G \begin{pmatrix} \alpha_{k+1} & z_2 & \tilde{z}_1^T & \tilde{z}_2^T & z_N \\ z_2 & d_2 & & & \\ \tilde{z}_1 & & \tilde{D}_1 & & \\ \tilde{z}_2 & & & \tilde{D}_2 & \\ z_N & & & & d_N \end{pmatrix} G^T G \begin{pmatrix} 0 & q_1 & \tilde{Q}_1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \tilde{Q}_2 & q_2 \end{pmatrix}^T$$

$$= \begin{pmatrix} 0 & cq_1 & \tilde{Q}_1 & 0 & -sq_1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & sq_2 & 0 & \tilde{Q}_2 & cq_2 \end{pmatrix} \begin{pmatrix} \alpha_{k+1} & r & \tilde{z}_1^T & \tilde{z}_2^T & 0 \\ r & \tilde{d}_2 & & & \\ \tilde{z}_1 & & \tilde{D}_1 & & \\ \tilde{z}_2 & & & \tilde{D}_2 & \\ 0 & & & & \tilde{d}_N \end{pmatrix} \begin{pmatrix} 0 & cq_1 & \tilde{Q}_1 & 0 & -sq_1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & sq_2 & 0 & \tilde{Q}_2 & cq_2 \end{pmatrix}^T$$

$$+O(\tau\|T\|_2) \quad , \qquad (4.4)$$

[5] I_i is the $i \times i$ identity matrix.

where $\tilde{d}_2 = d_2c^2 + d_Ns^2$ and $\tilde{d}_N = d_2s^2 + d_Nc^2$ are as in (4.2).

$\tilde{d}_N$ is an approximate eigenvalue of T and can be deflated. The corresponding approximate eigenvector is the last column of the first matrix in (4.4). The leading $(N-1)\times(N-1)$ principle submatrix of the middle matrix is again an arrowhead matrix. We deflate this matrix in a similar fashion until $\tilde{D}_1$ does not have any diagonal element that is close to a diagonal element of $\tilde{D}_2$. Thus, after this procedure T can be written as

$$T = \begin{pmatrix} \tilde{X}_1 & \tilde{X}_2 \end{pmatrix} \begin{pmatrix} \tilde{H}_1 & \\ & \tilde{\Lambda}_2 \end{pmatrix} \begin{pmatrix} \tilde{X}_1 & \tilde{X}_2 \end{pmatrix}^T + O(\tau\|T\|_2) \quad . \tag{4.5}$$

$\tilde{\Lambda}_2$ is a diagonal matrix whose diagonal elements are the deflated eigenvalues; the columns of $\tilde{X}_2$ are the corresponding approximate eigenvectors. $\tilde{H}_1$ is the arrowhead matrix

$$\tilde{H}_1 = \begin{pmatrix} \alpha_{k+1} & \tilde{z}_0^T & \tilde{z}_1^T & \tilde{z}_2^T \\ \tilde{z}_0 & \tilde{D}_0 & & \\ \tilde{z}_1 & & \tilde{D}_1 & \\ \tilde{z}_2 & & & \tilde{D}_2 \end{pmatrix} \quad ,$$

where the dimension of $\tilde{D}_0$ is the number of deflations; $\tilde{D}_i$ contains the diagonal elements of D_i not affected by deflation; and $\tilde{z}_0$, $\tilde{z}_1$ and $\tilde{z}_2$ are defined accordingly. $\tilde{X}_1$ is of the form

$$\tilde{X}_1 = \begin{pmatrix} 0 & \tilde{Q}_{0,1} & \tilde{Q}_1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \tilde{Q}_{0,2} & 0 & \tilde{Q}_2 \end{pmatrix} \quad , \tag{4.6}$$

where the column dimension of both $\tilde{Q}_{0,1}$ and $\tilde{Q}_{0,2}$ is the number of deflations, and the columns of $\tilde{Q}_1$ and $\tilde{Q}_2$ are those of Q_1 and Q_2 not affected by deflation.

If $\tilde{D}_0$ has a diagonal element that is close to a diagonal element of either $\tilde{D}_1$ or $\tilde{D}_2$, then we can use reduction (4.2) to deflate without changing the structure of $\tilde{X}_1$. In the following we assume that no further deflation is possible in (4.5).

Let $\tilde{U}_1\tilde{\Lambda}_1\tilde{U}_1^T$ be an eigendecomposition of $\tilde{H}_1$. Then

$$\begin{aligned} T &= \begin{pmatrix} \tilde{X}_1 & \tilde{X}_2 \end{pmatrix} \begin{pmatrix} \tilde{H}_1 & \\ & \tilde{\Lambda}_2 \end{pmatrix} \begin{pmatrix} \tilde{X}_1 & \tilde{X}_2 \end{pmatrix}^T + O(\tau\|T\|_2) \\ &= \begin{pmatrix} \tilde{X}_1 & \tilde{X}_2 \end{pmatrix} \begin{pmatrix} \tilde{U}_1\tilde{\Lambda}_1\tilde{U}_1^T & \\ & \tilde{\Lambda}_2 \end{pmatrix} \begin{pmatrix} \tilde{X}_1 & \tilde{X}_2 \end{pmatrix}^T + O(\tau\|T\|_2) \\ &= \begin{pmatrix} \tilde{X}_1\tilde{U}_1 & \tilde{X}_2 \end{pmatrix} \begin{pmatrix} \tilde{\Lambda}_1 & \\ & \tilde{\Lambda}_2 \end{pmatrix} \begin{pmatrix} \tilde{X}_1\tilde{U}_1 & \tilde{X}_2 \end{pmatrix}^T + O(\tau\|T\|_2) \quad . \end{aligned}$$

Thus $\begin{pmatrix} \tilde{X}_1\tilde{U}_1 & \tilde{X}_2 \end{pmatrix}$ is an approximate eigenvector matrix of T. The matrix $\tilde{X}_1\tilde{U}_1$ can be computed while taking advantage of the block structure of $\tilde{X}_1$ in (4.6).

We refer to these deflations as *local* deflations since they are only associated with individual subproblems of *ADC*.

4.3. Global Deflation of *ADC*

To illustrate *global* deflation, we look at 2 levels of the dividing strategy (cf. (2.2)). For simplicity, non-important entries of T are denoted by x.

$$T = \begin{pmatrix} T_1 & \beta_{i+j+2}e_{i+j+1} & \\ \beta_{i+j+2}e_{i+j+1}^T & x & xe_1^T \\ & xe_1 & T_2 \end{pmatrix}$$

$$= \begin{pmatrix} T_{1,1} & xe_i & & & \\ xe_i^T & x & \beta_{i+2}e_1^T & & \\ & \beta_{i+2}e_1 & T_{1,2} & \beta_{i+j+2}e_j & \\ & & \beta_{i+j+2}e_j^T & x & xe_1^T \\ & & & xe_1 & T_2 \end{pmatrix},$$

where T_1, T_2, $T_{1,1}$ and $T_{1,2}$ are principle submatrices of T of dimensions $(i+j+1)\times(i+j+1)$, $(N-i-j-2)\times(N-i-j-2)$, $i\times i$ and $j\times j$, respectively.

Let $Q_{1,2}D_{1,2}Q_{1,2}^T$ be the eigendecomposition of $T_{1,2}$, and let $f_{1,2}^T$ and $l_{1,2}^T$ be the first and last rows of $Q_{1,2}$, respectively. Then

$$T = \begin{pmatrix} T_{1,1} & xe_i & & & \\ xe_i^T & x & \beta_{i+2}e_1^T & & \\ & \beta_{i+2}e_1 & Q_{1,2}D_{1,2}Q_{1,2}^T & \beta_{i+j+2}e_j & \\ & & \beta_{i+j+2}e_j^T & x & xe_1^T \\ & & & xe_1 & T_2 \end{pmatrix}$$

$$= Y\begin{pmatrix} T_{1,1} & xe_i & & & \\ xe_i^T & x & \beta_{i+2}f_{1,2}^T & & \\ & \beta_{i+2}f_{1,2} & D_{1,2} & \beta_{i+j+2}l_{1,2} & \\ & & \beta_{i+j+2}l_{1,2}^T & x & xe_1^T \\ & & & xe_1 & T_2 \end{pmatrix}Y^T, \qquad (4.7)$$

where $Y = diag(I_i, 1, Q_{1,2}, 1, I_{N-i-j-2})$.

Let $\bar{d}_s$ be the s-th diagonal element of $D_{1,2}$. Then $\bar{d}_s$ is also the $(i+s+1)$-st diagonal element of the middle matrix. Let $\bar{f}_s$ and $\bar{l}_s$ be the s-th components of $f_{1,2}$ and $l_{1,2}$, respectively. Then, by ignoring all zero components, the $(i+s+1)$-st row of the middle matrix in (4.7) is $(\beta_{i+2}\bar{f}_s, \bar{d}_s, \beta_{i+j+2}\bar{l}_s)$. Thus if both $|\beta_{i+2}\bar{f}_s|$ and $|\beta_{i+j+2}\bar{l}_s|$ are small, then we can perturb them both to zero. T has $\bar{d}_s$ as an approximate eigenvalue with the $(i+s+1)$-st column of Y being the corresponding approximate eigenvector. This eigenvalue and its eigenvector can be deflated from all subsequent subproblems. We call this *global* deflation.

Consider the deflation procedure for computing the eigendecomposition of T_1 in Section 4.2. If $|\beta_{i+2}\bar{f}_s|$ is small, then it can be perturbed to zero. This is a local deflation if only $|\beta_{i+2}\bar{f}_s|$ is small, and a global deflation if $|\beta_{i+j+2}\bar{l}_s|$ is also small.

5. Acceleration by the Fast Multipole Method

Suppose that we want to evaluate the complex function

$$\Phi(x) = \sum_{j=1}^{n} c_j \varphi(x - x_j) \tag{5.1}$$

at m points in the complex plane, where $\{c_j\}_{j=1}^{n}$ are constants and $\varphi(x)$ is one of $\log(x)$, $1/x$ and $1/x^2$. The direct computation takes $O(nm)$ time. But the *fast multipole method* (*FMM*) proposed by Carrier, Greengard and Rokhlin [9, 14] takes only $O(n+m)$ time to approximate $\Phi(x)$ at these points to a precision specified by the user[6]. In this section we briefly describe how *FMM* can be used to accelerate algorithm *ADC*. A more detailed description appears in [16] in the context of updating the singular value decomposition.

Let

$$H = \begin{pmatrix} \alpha & z^T \\ z & D \end{pmatrix} ,$$

where $D = diag(d_2, \ldots, d_n)$ is a matrix of order $(n-1) \times (n-1)$ with $d_2 < d_3 < \ldots < d_n$; $z = (z_2, \ldots, z_n)^T$ is a vector of order $n-1$ with $z_i \neq 0$; and α is a scalar. Let $U \Lambda U^T$ denote the eigendecomposition of H with $U = (u_1, \ldots, u_n)$ and $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$.

We consider the cost of computing $U^T q$ for a vector $q = (q_1, \ldots, q_n)^T$. By equation (3.2) in Lemma 1, the i-th component $u_i^T q$ of $U^T q$ can be written as

$$u_i^T q = \frac{-q_1 + \Phi_1(\lambda_i)}{\sqrt{1 + \Phi_2(\lambda_i)}} ,$$

where

$$\Phi_1(\lambda) = \sum_{k=2}^{n} \frac{z_k q_k}{d_k - \lambda} \quad \text{and} \quad \Phi_2(\lambda) = \sum_{k=2}^{n} \frac{z_k^2}{(d_k - \lambda)^2} .$$

Thus we can compute $U^T q$ by evaluating $\Phi_1(\lambda)$ and $\Phi_2(\lambda)$ at n points. Since these two functions are of the form (5.1), we can do this in $O(n)$ time using *FMM*. To achieve better efficiency, we modify *FMM* to take advantage of the fact that all the computations are real (see [15, 16]).

Let T be a symmetric tridiagonal matrix of size N. When *ADC* is used to compute all the eigenvalues and eigenvectors, the main cost for each subproblem is in forming $\tilde{X}_1 U$ (see (4.5)), where $\tilde{X}_1$ is a column orthonormal matrix[7]. Each row of $\tilde{X}_1 U$ is of the form $q^T U = (U^T q)^T$, and there are $O(n)$ rows. Thus the cost of computing $\tilde{X}_1 U$ is $O(n^2)$ using *FMM*. There are $\log_2 N$ levels of recursion and 2^{k-1} subproblems at the k-th level, each of size $O(N/2^k)$. Thus the cost at the k-th level is $O(N^2/2^k)$, and the total time is $O(N^2)$.

[6] The constant hidden in the O notation depends on the logarithm of the precision.

[7] $\tilde{X}_1$ is also a block-structured matrix (see (4.6)). But here we view it as a dense matrix to simplify the presentation, even though *FMM* is more efficient when it exploits this structure.

We may also have to apply an orthonormal matrix Y to the eigenvector matrix of T, e.g., when T is obtained by reducing a dense matrix to tridiagonal form [12, 24, 28, 29]. For each subproblem, we can apply the eigenvector matrix of the corresponding arrowhead matrix directly to Y. The cost for each subproblem is $O(Nn)$ using *FMM*, and there are $O(N/n)$ subproblems at each level. Thus the cost at each level is $O(N^2)$, and the total time is $O(N^2 \log_2 N)$.

When *ADC* is used to compute only the eigenvalues, the main cost for each subproblem is computing two vectors of the form $q^T U$, finding all the roots of the reformulated secular equation, and computing the vector $\hat{z}$. We now show how to find all the roots of H and all the components of $\hat{z}$ in $O(n)$ time.

A root-finder computes successive iterates for each eigenvalue λ_i. We assume that the number of iterations for each root is bounded. The main cost for each iteration is in evaluating the function

$$g_i(\mu) = \mu + \alpha_i + \psi_i(\mu) + \phi_i(\mu) = \mu + \alpha_i + \sum_{j=2}^{n} \frac{z_j^2}{\delta_j - \mu} \quad .$$

To compute all the eigenvalues simultaneously, we must evaluate $g_i(\mu)$ at $O(n)$ points. The function $g_i(\mu)$ is similar to the form (5.1), and thus we can evaluate $g_i(\mu)$ at these points in $O(n)$ time using *FMM*[8]. In other words, all the eigenvalues of H can be computed in $O(n)$ time.

To compute $\hat{z}$, we note that equation (3.4) can be rewritten as

$$|\hat{z}_i| = \sqrt{(d_i - \hat{\lambda}_1)\,(\hat{\lambda}_n - d_i)}\ \exp\left(\frac{1}{2}\sum_{j=2}^{i-1} \log\left(\frac{\hat{\lambda}_j - d_i}{d_j - d_i}\right) + \frac{1}{2}\sum_{j=i}^{n-1} \log\left(\frac{\hat{\lambda}_j - d_i}{d_{j+1} - d_i}\right)\right) \quad .$$

Thus we can compute all of the components of $\hat{z}$ in $O(n)$ time using *FMM*.

We have shown that when computing all the eigenvalues of T using *ADC*, we can solve each subproblem in $O(n)$ time. Since there are $O(N/n)$ subproblems at each level, the cost at each level is $O(N)$, and thus the total time is $O(N \log_2 N)$.

6. Numerical Results

In this section we present a numerical comparison among *ADC* and three other algorithms for solving the symmetric tridiagonal eigenproblem:

1. B/II: bisection with inverse iteration [19, 20] (subroutines DSTEBZ and DSTEIN from LAPACK [2]);
2. *CDC*: Cuppen's divide-and-conquer algorithm [10, 11] (subroutine TREEQL [11] from `netlib`);

[8] See [15, 16] for a version of *FMM* for computing $\psi_i(\mu)$ and $\phi_i(\mu)$ and their derivatives at $O(n)$ points in $O(n)$ time. This is needed for the root-finders in [8, 22] and to check the stopping criterion.

3. QR: the QR algorithm [12] (subroutine DSTEQR from LAPACK [2]).

Since none of our test matrices is particularly large, *FMM* is not used in this comparison.

All codes are written in FORTRAN and were compiled with optimization enabled. All computations were done on a SPARCstation 1 in double precision. The machine precision is $\epsilon = 1.1 \times 10^{-16}$.

Let $[\beta, \alpha_i, \beta]$ denote the $N \times N$ symmetric tridiagonal matrix with β on the off-diagonals and $\alpha_1, \ldots, \alpha_N$ on the diagonal. We use the following sets of test matrices.

1. a random matrix with both diagonal and off-diagonal elements being uniformly distributed random numbers in $[-1, 1]$;
2. the Wilkinson matrix $W_N^+ = [1, w_i, 1]$, where $w_i = |(N+1)/2 - i|$ and N is odd;
3. the glued Wilkinson matrix[9] W_g^+: a 25×25 block matrix, with each diagonal block being the Wilkinson matrix W_k^+; the off-diagonal elements $\beta_{i \times k+1} = g$, for $i = 1, \ldots, 24$;
4. the Toeplitz matrix $[1, 2, 1]$;
5. the matrix $[1, \mu_i, 1]$, where $\mu_i = i \times 10^{-6}$;
6. the matrix $[1/100, 1 + \mu_i, 1/100]$, where $\mu_i = i \times 10^{-6}$;
7. Type 8 to Type 21 test matrices[10] from the LAPACK test suite.

Tables 1-5 present the numerical results. An asterisk means that the algorithm did not converge. Since the numerical results in Tables 1-3 suggest that *CDC* and *QR* are not as competitive, we only compare *ADC* with B/II for the LAPACK test matrices (see Tables 4 and 5).

The residual and orthonormality measures for *ADC* are always comparable with those for QR and B/II, and *ADC* is roughly twice as fast as *CDC* for large test matrices, due to differences in how deflation is implemented (see Section 4.2). In most cases *ADC* is faster than the others by a considerable margin, and in many cases is more than 5-10 times faster. When *ADC* is slower than B/II (by at most 10%), the matrix size is large ($N = 512$) and there are few deflations. These are cases where *ADC* can be significantly speeded up by *FMM*.

References

[1] L. Adams and P. Arbenz, *Towards a divide and conquer algorithm for the real nonsymmetric eigenvalue problem*, Tech. Report No. 91-8, Department of Applied Mathematics, University of Washington, August 1991.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, 1992.

[9] In general, the block size and the glue g vary. See [19] for other glued Wilkinson matrices.

[10] Type 1 to Type 7 test matrices in LAPACK are all diagonal matrices and are thus ignored.

Matrix Set	Order N	Execution Time (seconds)			
		ADC	B/II	*CDC*	QR
Random	64	0.96	2.17	0.99	1.73
	128	3.12	8.50	3.90	11.63
	256	10.43	33.35	14.88	85.86
	512	20.89	133.61	34.31	654.52
W_N^+	65	0.58	1.78	0.60	1.37
	129	1.44	6.54	1.46	9.87
	257	3.43	25.00	3.74	66.86
	513	8.26	97.57	14.76	497.55
$W_{10^{-8}}^+$	125	4.42	7.05	*	5.70
	275	26.88	32.94	*	52.71
	525	52.15	123.20	*	330.82
$W_{10^{-14}}^+$	125	0.63	5.88	*	5.12
	275	2.22	28.83	*	47.35
	525	8.23	121.84	*	353.41
$[1, 2, 1]$	64	0.90	2.18	0.69	1.49
	128	3.91	8.49	3.72	10.21
	256	21.89	33.68	22.77	72.40
	512	138.79	144.43	213.01	545.05
$[1, \mu_i, 1]$	64	1.02	2.19	1.16	1.49
	128	4.48	8.54	6.66	10.17
	256	24.20	33.64	43.02	72.14
	512	148.95	135.48	302.06	544.65
$[1/100, 1 + \mu_i, 1/100]$	64	1.05	3.14	1.18	1.49
	128	4.57	16.93	6.86	9.83
	256	24.45	102.81	43.01	70.65
	512	149.50	692.64	301.58	539.48

Table 1: Execution Time

Matrix Set	Order N	$\frac{\max_i \|Tq_i - d_iq_i\|_2}{N\epsilon\|T\|_2}$ *ADC*	B/II	*CDC*	QR
Random	64	0.96×10^{-1}	0.12×10^{-1}	0.77×10^{-1}	0.19×10^{0}
	128	0.49×10^{-1}	0.11×10^{-1}	0.10×10^{1}	0.16×10^{0}
	256	0.43×10^{-1}	0.47×10^{-2}	0.74×10^{0}	0.82×10^{-1}
	512	0.23×10^{-1}	0.28×10^{-2}	0.13×10^{1}	0.69×10^{-1}
W_N^+	65	0.10×10^{0}	0.10×10^{-1}	0.40×10^{0}	0.12×10^{0}
	129	0.67×10^{-1}	0.86×10^{-2}	0.59×10^{0}	0.61×10^{-1}
	257	0.17×10^{-1}	0.39×10^{-2}	0.15×10^{0}	0.35×10^{-1}
	513	0.44×10^{-2}	0.21×10^{-2}	0.67×10^{0}	0.21×10^{-1}
$W_{10^{-8}}^+$	125	0.23×10^{-1}	0.85×10^{-2}	$*$	0.90×10^{-1}
	275	0.64×10^{-1}	0.85×10^{-1}	$*$	0.82×10^{-1}
	525	0.19×10^{-1}	0.54×10^{-1}	$*$	0.13×10^{0}
$W_{10^{-14}}^+$	125	0.11×10^{0}	0.16×10^{0}	$*$	0.22×10^{0}
	275	0.27×10^{-1}	0.36×10^{-1}	$*$	0.11×10^{0}
	525	0.15×10^{-1}	0.66×10^{-1}	$*$	0.14×10^{0}
$[1, 2, 1]$	64	0.78×10^{-1}	0.11×10^{-1}	0.35×10^{-1}	0.71×10^{-1}
	128	0.41×10^{-1}	0.70×10^{-2}	0.31×10^{-1}	0.52×10^{-1}
	256	0.22×10^{-1}	0.12×10^{-1}	0.25×10^{-1}	0.35×10^{-1}
	512	0.12×10^{-1}	0.35×10^{-2}	0.20×10^{-1}	0.25×10^{-1}
$[1, \mu_i, 1]$	64	0.88×10^{-1}	0.20×10^{-1}	0.80×10^{-1}	0.12×10^{0}
	128	0.46×10^{-1}	0.16×10^{-1}	0.67×10^{-1}	0.90×10^{-1}
	256	0.23×10^{-1}	0.11×10^{-1}	0.47×10^{-1}	0.64×10^{-1}
	512	0.12×10^{-1}	0.79×10^{-2}	0.36×10^{-1}	0.47×10^{-1}
$[1/100, 1 + \mu_i, 1/100]$	64	0.36×10^{-1}	0.17×10^{-1}	0.20×10^{-1}	0.85×10^{-1}
	128	0.22×10^{-1}	0.79×10^{-2}	0.11×10^{-1}	0.60×10^{-1}
	256	0.12×10^{-1}	0.42×10^{-2}	0.11×10^{-1}	0.38×10^{-1}
	512	0.59×10^{-2}	0.21×10^{-2}	0.64×10^{-2}	0.28×10^{-1}

Table 2: Residual

Matrix Set	Order N	$\frac{\max_i \|Q^T q_i - e_i\|_2}{N\epsilon}$ *ADC*	B/II	*CDC*	QR
Random	64	0.20×10^0	0.18×10^0	0.16×10^0	0.62×10^0
	128	0.94×10^{-1}	0.30×10^0	0.54×10^{-1}	0.59×10^0
	256	0.66×10^{-1}	0.86×10^{-1}	0.17×10^0	0.54×10^0
	512	0.35×10^{-1}	0.72×10^{-1}	0.30×10^0	0.47×10^0
W_N^+	65	0.15×10^0	0.69×10^{-1}	0.77×10^{-1}	0.28×10^0
	129	0.78×10^{-1}	0.35×10^{-1}	0.54×10^{-1}	0.80×10^0
	257	0.39×10^{-1}	0.19×10^{-1}	0.89×10^{-1}	0.13×10^1
	513	0.19×10^{-1}	0.12×10^{-1}	0.72×10^{-1}	0.13×10^1
$W_{10^{-8}}^+$	125	0.64×10^{-1}	0.36×10^{-1}	$*$	0.22×10^0
	275	0.33×10^{-1}	0.17×10^0	$*$	0.18×10^0
	525	0.23×10^{-1}	0.29×10^{-1}	$*$	0.28×10^0
$W_{10^{-14}}^+$	125	0.64×10^{-1}	0.56×10^{-1}	$*$	0.38×10^0
	275	0.33×10^{-1}	0.16×10^0	$*$	0.31×10^0
	525	0.20×10^{-1}	0.34×10^{-1}	$*$	0.32×10^0
$[1, 2, 1]$	64	0.14×10^0	0.50×10^0	0.11×10^0	0.17×10^0
	128	0.70×10^{-1}	0.78×10^0	0.36×10^0	0.13×10^0
	256	0.47×10^{-1}	0.35×10^0	0.18×10^0	0.70×10^{-1}
	512	0.39×10^{-1}	0.21×10^0	0.14×10^0	0.44×10^{-1}
$[1, \mu_i, 1]$	64	0.94×10^{-1}	0.10×10^1	0.16×10^0	0.23×10^0
	128	0.62×10^{-1}	0.92×10^0	0.17×10^0	0.12×10^0
	256	0.49×10^{-1}	0.12×10^1	0.21×10^0	0.62×10^{-1}
	512	0.35×10^{-1}	0.55×10^0	0.76×10^0	0.48×10^{-1}
$[1/100, 1 + \mu_i, 1/100]$	64	0.11×10^0	0.62×10^{-1}	0.71×10^{-1}	0.15×10^0
	128	0.78×10^{-1}	0.35×10^{-1}	0.91×10^{-1}	0.12×10^0
	256	0.62×10^{-1}	0.23×10^{-1}	0.11×10^0	0.78×10^{-1}
	512	0.61×10^{-1}	0.21×10^{-1}	0.93×10^{-1}	0.40×10^{-1}

Table 3: Orthonormality

Matrix	Order	Execution Time		$\frac{\max_i \|Tq_i - d_iq_i\|_2}{N\epsilon\|T\|_2}$		$\frac{\max_i \|Q^Tq_i - e_i\|_2}{N\epsilon}$	
Type	N	*ADC*	B/II	*ADC*	B/II	*ADC*	B/II
8	64	1.04	2.38	0.68×10^{-1}	0.16×10^{-1}	0.11×10^{0}	0.19×10^{0}
	128	4.64	8.68	0.47×10^{-1}	0.82×10^{-2}	0.86×10^{-1}	0.16×10^{0}
	256	24.27	33.63	0.27×10^{-1}	0.54×10^{-2}	0.66×10^{-1}	0.25×10^{0}
	512	140.04	133.58	0.17×10^{-1}	0.30×10^{-2}	0.47×10^{-1}	0.21×10^{0}
9	64	0.67	2.38	0.83×10^{-1}	0.88×10^{-2}	0.20×10^{0}	0.47×10^{-1}
	128	2.32	12.66	0.13×10^{0}	0.14×10^{-1}	0.12×10^{0}	0.20×10^{-1}
	256	9.37	76.99	0.28×10^{-1}	0.30×10^{-2}	0.53×10^{-1}	0.27×10^{-1}
	512	44.62	517.45	0.12×10^{-1}	0.92×10^{-1}	0.33×10^{-1}	0.84×10^{-1}
10	64	0.01	1.99	0.23×10^{-1}	0.34×10^{-1}	0.16×10^{-1}	0.83×10^{0}
	128	0.01	12.36	0.11×10^{-1}	0.10×10^{-1}	0.14×10^{-1}	0.70×10^{0}
	256	0.04	84.54	0.45×10^{-2}	0.70×10^{-2}	0.78×10^{-2}	0.52×10^{-1}
	512	0.17	613.86	0.38×10^{-2}	0.21×10^{-2}	0.78×10^{-2}	0.21×10^{-1}
11	64	1.29	2.21	0.81×10^{-1}	0.16×10^{-1}	0.11×10^{0}	0.41×10^{0}
	128	5.24	8.64	0.45×10^{-1}	0.11×10^{-1}	0.62×10^{-1}	0.22×10^{0}
	256	25.88	33.52	0.28×10^{-1}	0.53×10^{-2}	0.55×10^{-1}	0.17×10^{0}
	512	144.37	132.32	0.17×10^{-1}	0.29×10^{-2}	0.41×10^{-1}	0.20×10^{0}
12	64	1.03	2.23	0.95×10^{-1}	0.15×10^{-1}	0.11×10^{0}	0.17×10^{0}
	128	4.54	8.75	0.46×10^{-1}	0.85×10^{-2}	0.86×10^{-1}	0.19×10^{0}
	256	24.44	33.94	0.25×10^{-1}	0.54×10^{-2}	0.51×10^{-1}	0.30×10^{0}
	512	141.57	133.83	0.16×10^{-1}	0.31×10^{-2}	0.47×10^{-1}	0.20×10^{0}
13	64	1.09	2.20	0.70×10^{-1}	0.17×10^{-1}	0.12×10^{0}	0.20×10^{0}
	128	4.61	8.66	0.43×10^{-1}	0.69×10^{-2}	0.62×10^{-1}	0.33×10^{0}
	256	23.49	33.53	0.20×10^{-1}	0.53×10^{-2}	0.70×10^{-1}	0.15×10^{0}
	512	131.57	132.03	0.13×10^{-1}	0.24×10^{-2}	0.41×10^{-1}	0.25×10^{0}
14	64	1.30	2.21	0.13×10^{0}	0.13×10^{-1}	0.25×10^{0}	0.19×10^{0}
	128	5.16	8.61	0.46×10^{-1}	0.79×10^{-2}	0.12×10^{0}	0.28×10^{0}
	256	24.88	33.55	0.24×10^{-1}	0.50×10^{-2}	0.51×10^{-1}	0.39×10^{0}
	512	134.86	131.96	0.12×10^{-1}	0.24×10^{-2}	0.43×10^{-1}	0.29×10^{0}

Table 4: LAPACK Type 8-14 Matrices

Matrix Type	Order N	Execution Time		$\frac{\max_i \|Tq_i - d_i q_i\|_2}{N\epsilon\|T\|_2}$		$\frac{\max_i \|Q^T q_i - e_i\|_2}{N\epsilon}$	
		ADC	B/II	*ADC*	B/II	*ADC*	B/II
15	64	1.04	2.23	0.71×10^{-1}	0.18×10^{-1}	0.16×10^{0}	0.11×10^{0}
	128	4.50	8.71	0.49×10^{-1}	0.73×10^{-2}	0.78×10^{-1}	0.12×10^{0}
	256	23.44	33.99	0.24×10^{-1}	0.45×10^{-2}	0.41×10^{-1}	0.17×10^{0}
	512	131.71	133.53	0.13×10^{-1}	0.26×10^{-2}	0.41×10^{-1}	0.13×10^{0}
16	64	1.05	2.21	0.40×10^{-1}	0.16×10^{-1}	0.19×10^{0}	0.13×10^{0}
	128	4.54	8.68	0.22×10^{-1}	0.87×10^{-2}	0.11×10^{0}	0.11×10^{0}
	256	24.18	33.73	0.13×10^{-1}	0.41×10^{-2}	0.64×10^{-1}	0.93×10^{-1}
	512	139.29	132.47	0.14×10^{-1}	0.19×10^{-2}	0.35×10^{-1}	0.15×10^{0}
17	64	0.71	2.45	0.48×10^{-1}	0.11×10^{-1}	0.23×10^{0}	0.73×10^{-1}
	128	2.67	12.88	0.30×10^{-1}	0.30×10^{-2}	0.11×10^{0}	0.58×10^{-1}
	256	11.78	76.55	0.38×10^{-1}	0.31×10^{-2}	0.51×10^{-1}	0.82×10^{-1}
	512	63.23	521.70	0.16×10^{-1}	0.17×10^{-2}	0.33×10^{-1}	0.29×10^{-1}
18	64	0.01	1.99	0.23×10^{-1}	0.30×10^{-1}	0.23×10^{-1}	0.17×10^{0}
	128	0.01	12.34	0.13×10^{-1}	0.78×10^{-2}	0.16×10^{-1}	0.39×10^{-1}
	256	0.04	83.95	0.98×10^{-2}	0.39×10^{-2}	0.59×10^{-2}	0.29×10^{-1}
	512	0.17	614.26	0.44×10^{-2}	0.19×10^{-2}	0.20×10^{-2}	0.16×10^{0}
19	64	1.24	2.19	0.39×10^{-1}	0.17×10^{-1}	0.94×10^{-1}	0.62×10^{-1}
	128	5.08	8.58	0.25×10^{-1}	0.79×10^{-2}	0.90×10^{-1}	0.92×10^{-1}
	256	25.47	33.32	0.14×10^{-1}	0.40×10^{-2}	0.70×10^{-1}	0.16×10^{0}
	512	142.16	131.26	0.13×10^{-1}	0.20×10^{-2}	0.31×10^{-1}	0.97×10^{-1}
20	64	1.01	2.22	0.38×10^{-1}	0.18×10^{-1}	0.13×10^{0}	0.96×10^{-1}
	128	4.46	8.68	0.20×10^{-1}	0.75×10^{-2}	0.62×10^{-1}	0.10×10^{0}
	256	24.12	33.72	0.13×10^{-1}	0.47×10^{-2}	0.51×10^{-1}	0.17×10^{0}
	512	139.29	132.75	0.13×10^{-1}	0.21×10^{-2}	0.33×10^{-1}	0.12×10^{0}
21	64	0.54	2.43	0.12×10^{0}	0.81×10^{-2}	0.22×10^{0}	0.31×10^{-1}
	128	1.85	12.86	0.45×10^{-1}	0.34×10^{-2}	0.70×10^{-1}	0.45×10^{-1}
	256	6.26	75.81	0.38×10^{-1}	0.27×10^{-2}	0.35×10^{-1}	0.16×10^{-1}
	512	21.07	517.17	0.15×10^{-1}	0.12×10^{-2}	0.31×10^{-1}	0.20×10^{-1}

Table 5: LAPACK Type 15-21 Matrices

[3] P. ARBENZ, *Divide-and-conquer algorithms for the bandsymmetric eigenvalue problem*, Tech. Report 170, Department Informatik, ETH, Zürich, Switzerland, 1991.

[4] P. ARBENZ AND G. H. GOLUB, *QR-like algorithms for symmetric arrow matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 655–658.

[5] J. L. BARLOW, *Error analysis of update methods for the symmetric eigenvalue problem*, Tech. Report CS-91-23, Department of Computer Science, The Pennsylvania State University, August 1991.

[6] D. BOLEY AND G. H. GOLUB, *Inverse eigenvalue problems for band matrices*, in Numerical Analysis, Proceedings, Biennial Conference, Dundee 1977, G. A. Watson, ed., vol. 630 of Lecture Notes in Mathematics, Springer-Verlag, 1977, pp. 23–31.

[7] H. BOWDLER, R. S. MARTIN, C. REINSCH, AND J. WILKINSON, *The QR and QL algorithms for symmetric matrices*, Numer. Math., 11 (1968), pp. 293–306.

[8] J. R. BUNCH, C. P. NIELSEN, AND D. C. SORENSEN, *Rank-one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

[9] J. CARRIER, L. GREENGARD, AND V. ROKHLIN, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 669–686.

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

[11] J. J. DONGARRA AND D. C. SORENSEN, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. s139–s154.

[12] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, second ed., 1989.

[13] W. B. GRAGG, J. R. THORNTON, AND D. D. WARNER, *Parallel divide and conquer algorithms for the symmetric tridiagonal eigenproblem and bidiagonal singular value problem*, in Proceedings of 23rd Annual Pittsburgh Conference, University of Pittsburgh School of Engineering, vol. 23 of Modelling and Simulation, 1992.

[14] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comp. Phys., 73 (1987), pp. 325–348.

[15] M. GU AND S. C. EISENSTAT, *A fast divide-and-conquer method for the symmetric tridiagonal eigenproblem.* Presented at the Fourth SIAM Conference on Applied Linear Algebra, Minneapolis, Minnesota, September 1991.

[16] ——, *A fast algorithm for updating the singular value decomposition.* To appear, 1992.

[17] ——, *A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem*, Tech. Report YALEU/DCS/RR-916, Department of Computer Science, Yale University, 1992. Submitted to SIAM J. Matrix Anal. Appl..

[18] I. C. F. IPSEN AND E. R. JESSUP, *Solving the symmetric tridiagonal eigenvalue problem on the hypercube*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 203–229.

[19] E. R. JESSUP, *Parallel Solution of the Symmetric Tridiagonal Eigenproblem*, Ph.D. thesis, Department of Computer Science, Yale University, 1989.

[20] E. R. JESSUP AND I. C. F. IPSEN, *Improving the accuracy of inverse iteration*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 550–572.

[21] W. KAHAN, *Rank-1 perturbed diagonal's eigensystem.* Unpublished manuscript, July 1989.

[22] R.-C. LI, *Solving secular equations stably and efficiently.* Unpublished manuscript, October 1992.

[23] D. P. O'LEARY AND G. W. STEWART, *Computing the eigenvalues and eigenvectors of symmetric arrowhead matrices*, J. Comp. Phys., 90 (1990), pp. 497–505.

[24] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, 1980.

[25] B. N. PARLETT AND B. NOUR-OMID, *The use of a refined error bound when updating eigenvalues of tridiagonals*, Lin. Alg. Applics., 68 (1985), pp. 179–219.

[26] W. E. SHREVE AND M. R. STABNOW, *An eigenvalue algorithm for symmetric bordered diagonal matrices*, in Current Trends in Matrix Theory, F. Uhlig and R. Grone, eds., Elsevier Science Publishing Co., Inc., 1987, pp. 339–346.

[27] D. C. SORENSEN AND P. T. P. TANG, *On the orthogonality of eigenvectors computed by divide-and-conquer techniques*, SIAM J. Numer. Anal., 28 (1991), pp. 1752–1775.

[28] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[29] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.