ELSEVIER

Journal of Computational and Applied Mathematics 123 (2000) 117–130

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS



# For tridiagonals $T$ replace $T$ with $LDL^t$

Beresford N. Parlett

*Mathematics Department and Computer Science Division, EECS Department, University of California, Berkeley, CA 94720, USA*

Received 21 July 1999; received in revised form 10 November 1999

**Abstract**

The same number of parameters determine a tridiagonal matrix $T$ and its triangular factors $L$, $D$ and $U$. The mapping $T \to LDU$ is not well defined for all tridiagonals but, in finite precision arithmetic, $L$, $D$ and $U$ determine the entries of $T$ to more than working precision. For the solution of linear equations $LDU\boldsymbol{x} = \boldsymbol{b}$ the advantages of factorization are clear. Recent work has shown that $LDU$ is also preferable for the eigenproblem, particularly in the symmetric case. This essay describes two of the ideas needed to compute eigenvectors that are orthogonal without recourse to the Gram–Schmidt procedure when some of the eigenvalues are tightly clustered. In the symmetric case we must replace $T$, or a translate of $T$, by its triangular factors $LDL^t$. 

## 1. Introduction and representations

This essay needs justification because it examines a problem that has been considered as solved for several decades: the symmetric eigenproblem for dense matrices. Section 2 describes the methods that have been used with satisfaction for a long time. However in 1995 there was a little trouble in Paradise. A team of computational chemists at Pacific Northwest Laboratories found that certain problems of order 1000–2000 were taking much longer than expected using the best available software. On investigation it turned out that in a three-stage process the middle part, which should have been negligible, was consuming 80% of the time. Further probing showed that 95% of the eigenvalues were judged by the program to be in a tight cluster (they all agreed to four or more decimals) and so the Gram–Schmidt orthonormalizing process was invoked to make sure that the computed eigenvectors were indeed orthonormal to working accuracy. Because the cluster was so large what is normally an $O(n^2)$ process, for $n \times n$ matrices, turned into an $O(n^3)$ marathon, see [11,8]. This incident provoked some interesting lines of thought. The conservative view would cite the inherent limitations of working in fixed precision arithmetic and would argue that very difficult

*E-mail address:* parlett@math.berkeley.edu (B.N. Parlett).

0377-0427/00/$ - see front matter 
PII: S0377-0427(00)00394-0

calculations should take more effort. This conservative view could be amplified and made quite persuasive. One central fact is that when the matrix elements are known to working precision, say 8 or 16 decimal places, then the eigenvalues and eigenvectors inherit a level of uncertainty that sets a limit on how accurately they can be computed. Indeed the closer some eigenvalues cluster together the less well determined are their eigenvectors. In the limit, for a multiple eigenvalue, it is only the eigenspace that is defined, there is no distinguished basis of eigenvectors. Consequently, extra measures, such as the Gram–Schmidt process, must be invoked to ensure that the program returns orthogonal eigenvectors for tight clusters of eigenvalues.

A different reaction to the 1995 revelation is to wonder whether there is a way to wriggle out of these difficult situations and to attain the following ambitious goal: given a $n \times n$ real symmetric tridiagonal matrix $T$ compute its eigenvalues and then send each eigenvalue, with a copy of $T$, to its own processor. Each processor computes its eigenvector, all at the same time, and the outputs turn out to be orthogonal to working precision without the need to check. That would be nice!

When the eigenvalues are nearly uniformly spaced in the spectrum then current methods can realize the goal. What might we do when several eigenvalues agree to 4 or 8 or 12 decimals?

There is a method, developed by Dhillon and me from 1996 to 1999, and software to implement it, but several new ideas are needed to justify the whole procedure and only one or two themes will be described in this essay. Section 4 shows the method in action on a $4 \times 4$ example. Before launching into more detail it is helpful to recall two key facts. First, eigenvectors are invariant under translation (or shifting) $T \rightarrow T - \sigma I$. Second, there is no loss in assuming that the next-to-diagonal entries $(i, i+1)$ and $(i+1, i)$ do not vanish, $i = 1, 2, \ldots, n-1$. In that case the true eigenvalues are distinct and the eigenvectors are well defined even though some eigenvalues may be equal to working precision. This is a subtle property of the tridiagonal form. Thus, there is a basis of eigenvectors even when some eigenvalues appear multiple to working precision. We can aim to compute extremely accurate eigenvectors and then orthogonality would follow automatically since the 'true' eigenvectors have this property.

We now describe the first of the new themes. The radical new goal is to compute an approximate eigenvector $\boldsymbol{x}$, $||\boldsymbol{x}|| = 1$, for a given approximate eigenvalue $\hat{\lambda}$ with the *relative residual* property

$$||T\boldsymbol{x} - \boldsymbol{x}\hat{\lambda}|| = \mathrm{O}(n\varepsilon)|\hat{\lambda}|, \quad \text{not just } \mathrm{O}(n\varepsilon||T||), \tag{1}$$

where $\varepsilon$ is the roundoff unit and we regard two normalized vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ as orthogonal to working precision if

$$|\boldsymbol{u}^{\mathrm{t}}\boldsymbol{v}| = \mathrm{O}(n\varepsilon). \tag{2}$$

We use big O notation to hide some modest constant between 1 and 100. Unfortunately (1) is not achievable for the simple reason that $\lambda$ is not always defined to high relative accuracy by $T$. Here $\lambda$ is the eigenvalue of $T$ closest to $\hat{\lambda}$. This means that small relative uncertainty in $T$'s entries may cause large relative uncertainties in tiny eigenvalues. A simple but mild example of this phenomenon is a Toeplitz matrix $a + b(N + N^{\mathrm{t}})$ where $N$ is the $n \times n$ Jordan block with eigenvalue 0. For $n = 4$,

$$N = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

When $b=1$, $a=-2$ we obtain the second difference matrix whose eigenvalues lie in $(-4,0)$. The one closest 0 is $-4\sin^2(\pi/4n) \approx -(\pi/2n)^2$. Take $n=10^3$ and change $a$ from $-2$ to $-2(1-10^{-8})$ to find that $\lambda_{\min}$ changes to $\lambda_{\min}(1-\frac{4}{5}10^{-2}+O(10^{-4}))$, a large relative change. The example is mild because the eigenvalues are not severely clustered. More complicated examples show this phenomenon of large relative changes in the smallest eigenvalues for small values of $n$, say $n=4$, see [7] and Section 4.

In order to achieve (1) it is not only necessary that each small eigenvalue, such as $\lambda$, be determined to high relative accuracy (by $T$) but we must find an algorithm that will approximate $\lambda$ by $\hat{\lambda}$ to high relative accuracy. If, for example, $|\lambda| = 10n\varepsilon$ then the residual norm in (1) must achieve the very small value $O(n^2\varepsilon^2)$.

Although there are special classes of tridiagonals that do define their small eigenvalues to high relative accuracy, see [2], the property fails most of the time.

In conclusion (1) seems to be an unattainable goal. At this low point we must emphasize, briefly, why (1) is so desirable. There is a well-known error bound that is, in addition, a fairly realistic estimate of the error (angle) between the $\boldsymbol{x}$ achieving (1) and the true eigenvector $\boldsymbol{s}$ for $\lambda$, the eigenvalue closest to $\hat{\lambda}$. This result is not restricted to tridiagonals, see [18] for a proof.

**Theorem 1.** *Let $T$ be real symmetric, $T\boldsymbol{s} = \boldsymbol{s}\lambda$, where $\lambda$ is the eigenvalue closest to $\hat{\lambda}$. For any $\boldsymbol{x}$, $||\boldsymbol{x}|| = 1$, and any $\hat{\lambda}$,*

$$\sin|\angle(\boldsymbol{x},\boldsymbol{s})| \leqslant \frac{||T\boldsymbol{x} - \boldsymbol{x}\hat{\lambda}||}{\text{gap}\ (\hat{\lambda})},$$

$$\text{gap}\ (\hat{\lambda}) = |\mu - \hat{\lambda}|,\ \mu \textit{ is the eigenvalue } (\neq \lambda) \textit{ closest to } \hat{\lambda}.$$

If (1) holds then the theorem assures us that

$$\sin|\angle(\boldsymbol{x},\boldsymbol{s})| \leqslant \frac{O(n\varepsilon)|\hat{\lambda}|}{|\mu - \hat{\lambda}|} \approx \frac{O(n\varepsilon)}{\text{relgap}(\hat{\lambda})},$$

where

$$\text{relgap}\ (\lambda) = \frac{|\lambda - \mu|}{|\lambda|}.$$

If (1) holds then

$$\frac{|\hat{\lambda} - \mu|}{|\hat{\lambda}|} \approx \frac{|\lambda - \mu|}{|\lambda|}.$$

For example, if $||T|| = 1$, $\lambda = 10^{-18}$ and $\mu = 10^{-19}$ then relgap $(\lambda) = 0.9$ and $\boldsymbol{x}$ is a very accurate approximation to $\boldsymbol{s}$. For more details see [16].

The message here is that if (1) can be achieved then very accurate eigenvectors can be produced for all eigenvalues with large *relative* gaps. The next link in the chain of ideas is the simple observation that *relative* gaps may be increased by a suitable shift of origin whereas absolute separation between eigenvalues is invariant since $|(\lambda - \sigma) - (\mu + \sigma)| = |\lambda - \mu|$.

Now we return to (1). The big new idea is to get rid of $T$! To convey the idea in a simple way consider the case when $T$ is positive definite and so $T$ permits a Cholesky decomposition

$$T = LL^{\mathrm{t}}, \tag{3}$$

where $L$ is lower bidiagonal. In 1967 Kahan proved that all the eigenvalues of $LL^{\mathrm{t}}$ are determined to high relative accuracy by the entries of $L$, not those of $T$. Today there are easy proofs of this result, see [9,10], and there is more than one way to compute the eigenvalues to this high accuracy. Of itself this result does not guarantee (1) but it is an essential element. That is the first theme. A method we shall not describe here, see [15], permits (1) to be achieved with $LL^{\mathrm{T}}$ replacing $T$.

Now we turn to a different theme. As suggested by the earlier remarks on relative gaps in the spectrum it may be necessary to shift the origin and use triangular factorization

$$T - \tau I = LL^{\mathrm{t}} - \tau I = L^{(1)} D_{+} L^{(1)^{\mathrm{t}}}, \tag{4}$$

where $L^{(1)}$ is a unit lower bidiagonal matrix containing the multipliers and $D_{+}$ is a diagonal matrix holding the 'pivots'. There is no general statement on how well the entries of $L^{(1)}$ and $D_{+}$ determine the eigenvalues of $L^{(1)} D_{+} L^{(1)^{\mathrm{t}}}$ but the results in [17] show that for most values of $\tau$ these factors $L^{(1)}$ and $D_{+}$ do give high relative accuracy for the small eigenvalues. There is nothing sacred in factoring from top to bottom. We can use as well a factorization from bottom to top:

$$T - \tau I = LL^{\mathrm{t}} - \tau I = U^{(1)} D_{-} U^{(1)^{\mathrm{t}}}, \tag{5}$$

where $U^{(1)}$ is a unit upper bidiagonal matrix and $D_{-}$ is diagonal. In fact, there is a whole family of $n$ twisted factorization of $T - \tau I$ and they all use the same number of parameters, namely $2n - 1$, see [17].

The implication of the preceding remarks is that we can compute very accurate eigenvectors if we can find representations, such as $L^{(1)} D_{+} L^{(1)^{\mathrm{t}}}$, that define their small eigenvalues to high relative accuracy. Recall that each shift changes the eigenvalues. However, one new representation will not (usually) suffice. We will need several representations, such as in (4), for different values of $\tau$. We will compute a subset of eigenvectors for each representation. This raises a new difficulty. When we change from one representation to another, say $\mathring{L}\mathring{D}\mathring{L}^{\mathrm{t}} = LDL^{\mathrm{t}} - \tau I$, we wonder whether the inevitable roundoff errors in computing $\mathring{L}$ and $\mathring{D}$ from $L$, $D$, and $\tau$ will break the link between the eigenvectors computed from $L$ and $D$ to those computed from $\mathring{L}$ and $\mathring{D}$. Fortunately, the recently discovered *differential* stationary qd algorithms, see [12], give a way to switch between representations and preserve high relative accuracy. That is the second theme and extends to nonsymmetric tridiagonals and is the topic of Section 3. Section 4 shows the new method in action on a difficult $4 \times 4$ example and Section 5 shows what extends to the nonsymmetric case and what still needs to be done.

## 2. The classical methods

As soon as digital computers became available to scientists around 1950 the search was begun for eigenvalue algorithms that were robust when executed in finite precision arithmetic. In 1954, very early in the game, Wallace Givens came up with a method for a symmetric matrix $A$ that has stood with little change for over 40 years. The defects of using the characteristic polynomial were quickly grasped. No one would like to meet a polynomial of degree 1000 on a dark night. It is extremely

volatile and prone to blow up under the slightest provocation. A promising alternative is to employ explicit similarity transformations until $A$ turns into $\Lambda$ diagonal. In principle, an infinite sequence of similarities is needed to reach $\Lambda$ and that brings on tricky questions of when to stop.

Givens proposed a compromise between the two approaches (explicit similarities and the characteristic polynomial) given above. The method has three distinct stages.

*Phase* 1: Reduce $A$ to tridiagonal $T$ by a *finite sequence* of plane rotations designed to eliminate one nontridiagonal entry at a time and preserve all previously created zero entries. Thus,

$$T = G_s^* \cdots G_1^* A G_1 \cdots G_s = F^* A F \tag{6}$$

where

$$s = \begin{pmatrix} n-2 \\ 2 \end{pmatrix}$$

and $G$ alters only two columns. The $G$'s are accumulated into $F$ and this phase costs $O(n^3)$ operations.

*Phase* 2: Apply 'bisection' to any given interval to find all, or some eigenvalues of $T$ to full precision (relative to $||T||$) or less. The tool is Sylvester's Inertia theorem applied to Gaussian elimination without interchanges. Let $T - \tau I = LDL^t$. Sylvester's Inertia theorem says the number of eigenvalues less than $\tau$ equals the number of negative entries on $D$'s diagonal. Once an interval contains a single eigenvalue bisection may be continued until a designated number of correct digits is obtained. The cost of each factorization is $2n$ operations, and so the cost of computing $k$ eigenvalues is $O(kn)$.

In his original technical report Givens did not invoke Sylvester's Inertia theorem nor triangular factorization. Instead he used a more complicated mechanism with a three-term recurrence and Sturm sequences but the two approaches are equivalent in exact arithmetic but Givens had to worry about over/underflow.

In order to compute the eigenvector belonging to a computed eigenvalue $\hat{\lambda}$ Givens solved $(T - \hat{\lambda} I)\boldsymbol{x} = \boldsymbol{e}_n$, where $\boldsymbol{e}_j$ is the $j$th column of the identity matrix $I$. This was the least successful feature of his method. Any fixed right-hand side will lead to trouble on some matrices. We now know that it is important to choose the right-hand side carefully, see [15] for more details. Again the cost for $\boldsymbol{x}$ is $O(n)$ so Phase 2 is an $O(kn)$ process for $k$ eigenpairs. As indicated in Section 1 numerical orthogonality depends on the separation of the eigenvalues and a Gram–Schmidt post-processing has to be available.

*Phase* 3: Let $T = S\Lambda S^t$. If the eigenvectors $Z$ of $A$ are wanted then $S$ is mapped into $Z$ via $Z = FS$. This is an $O(n^3)$ process. $F$ need not be found explicitly but can be represented by the sequence of $G$'s given in (6). If only k eigenvectors are wanted the cost reduces to $O(kn^2)$.

Finally, Givens produced an error analysis in fixed-point arithmetic showing that the computed eigenvalues were the exact eigenvalues of a matrix close to $T$ or $A$. This was one of the earliest instances of a 'backward' error analysis: the computed quantities solve exactly a nearby problem. It is worth emphasizing that a backward error analysis is not possible for all algorithms.

There is little to add for the task of computing a subset of eigenpairs. There is an alternative to Phase 2 when all eigenvalues are wanted. The QR algorithm, see [13,5], is applied to $T$ yielding $\Lambda = R_p^* \cdots R_1^* T R_1 \cdots R_p$ where each $R_i$ is a plane rotation $G$ and $p$ is the number of iterations used in the QR algorithm. Then $S = R_1 \cdots R_p$ and this accumulation of plane rotations produces an $S$ that is

orthogonal to working accuracy however close the eigenvalues may be. The price for this desirable property is an $O(n^3)$ algorithm for the spectral factorization of $T$. Since Phases 1 and 3 are $O(n^3)$ what is wrong with having Phase 2 also $O(n^3)$? Answer: the constant behind O is too big.

## 3. Changing representations

In this section we consider tridiagonals that are not necessarily symmetric. Instead of $T-\sigma I=LDL^{t}$ we will have $T-\tau I=LU$. We normalize our matrices in a way that would destroy symmetry. Any tridiagonal with nonzero off-diagonal entries is said to be *unreduced*. Any unreduced tridiagonal is diagonally similar to one with all super-diagonal entries equal to 1; $\Delta T\Delta^{-1}=J$. We designate such matrices by $J$ and note that the number of free parameters in $J$ is $2n-1$ for $n\times n$ cases. If $J$ permits triangular factorization, $J=LU$, then we write

$$L=\text{bidiag}\begin{pmatrix} 1 & 1 & 1 & . & 1 & 1 \\ & e_1 & e_2 & . & . & e_{n-1} & \end{pmatrix},$$

$$U=\text{bidiag}\begin{pmatrix} 1 & 1 & . & 1 & 1 \\ q_1 & q_2 & . & . & q_{n-1} & q_n \end{pmatrix}.$$

An attractive feature of this notation is that $UL$ is also a $J$-matrix and that feature is exploited later in the section.

Section 1 emphasized the advantages of exploiting the shift invariance of eigenvectors. Suppose that $L$ and $U$ determine well the eigenvalues of $J$. When we need a new representation, for $J-\sigma I$ say, we must compute $\overset{\circ}{L}$ and $\overset{\circ}{U}$ satisfying

$$\overset{\circ}{L}\overset{\circ}{U}=J-\sigma I=LU-\sigma I.$$

There are (at least) two ways to compute $\overset{\circ}{L}$ and $\overset{\circ}{U}$ from $L$, $U$, and $\sigma$. The first is called the *stationary* qd-algorithm *stqds* by Rutishauser, see [22].

The algorithm can be derived by equating entries on each side of $\overset{\circ}{L}\overset{\circ}{U}=LU-\sigma I$ in the appropriate order.

stqds($\sigma$):
$\overset{\circ}{q}_1=q_1-\sigma$
**for** $i=1,n-1$ **do**
$\overset{\circ}{e}_i=e_iq_i/\overset{\circ}{q}_i$
$\overset{\circ}{q}_{i+1}=e_i+q_{i+1}-\sigma-\overset{\circ}{e}_i$
**end for**

Unfortunately, when executed in finite precision arithmetic, this algorithm is not accurate enough to connect one representation to the other by making small *relative* changes to the parameters $q$ and $e$. There is more discussion of this point later in this section. Fortunately, there is an alternative implementation. It is easy to miss and Rutishauser never published it and seems to have discovered it only in the last two years of his life. The alternative was found independently of Rutishauser by Fernando and Parlett as recently as 1991 and in another context, see [12]. It is called the *differential*

stationary qd algorithm and so the old name is prefixed with a little d.

dstqds($\sigma$): $s_1 = -\sigma$
**for** $i = 1, n-1$ **do**
$\overset{\circ}{q}_i = s_i + q_i$
$\overset{\circ}{e}_i = e_i q_i / \overset{\circ}{q}_i$
$s_{i+1} = s_i q_i / \overset{\circ}{q}_i - \sigma$
**end for**
$\overset{\circ}{q}_n = s_n + q_n$

The auxiliary variable is called $s_i$ and the new value $s_{i+1}$ may be written over the old $s_i$. The essential property of the new algorithm is that it enjoys *mixed relative stability*. What does this mean?

Let $\overset{\circ}{L}$ and $\overset{\circ}{U}$ now denote the bidiagonal matrices actually computed by dstqds($\sigma$) in the computer. Then there exist special tiny end-figure changes to the entries of $L$, $U$, $\overset{\circ}{L}$, $\overset{\circ}{U}$ giving new matrices $\bar{L}$, $\bar{U}$, $\tilde{L}$, $\tilde{U}$, respectively, such that

$$\tilde{L}\tilde{U} = \bar{L}\bar{U} - \sigma I$$

*exactly*. The necessary change in most of the entries is two units (bits) in the last place held (i.e. in the last digit) and none exceeds four. Thus, the eigenvectors of $\tilde{L}\tilde{U}$ are identical to those of $\bar{L}\bar{U}$ and we only have to make sure that $\overset{\circ}{L}$, $\overset{\circ}{U}$ determine the (small) eigenvalues of $\overset{\circ}{L}\overset{\circ}{U}$ together with the associated eigenvectors to high relative accuracy. In addition $L$, $U$ must also determine those same eigenvectors to high relative accuracy. Symmetry is not essential.

It should be mentioned that when the original matrix is symmetric then there is a minor variation of dstqds that uses, not $L$ and $U$ but $L$ and $D$ where $LDL^t$ is the matrix in question. The same stability results hold with minor variations in the details, see [7,16]. That relative stability property of dstqds is the second ingredient in the method for computing accurate eigenvectors. It permits us to relate the eigenvectors computed from different representations to the eigenvectors of one single matrix $LL^t$ or $LDL^t$.

There is an essential component of the new method that has not been discussed so far. How can one calculate the eigenvalues of $LL^t$ or $LDL^t$ to high relative accuracy? In the symmetric case there is a variant of the well-known bisection algorithm, see [5], that may be used. This technique is effective for refining eigenvalues already known to good accuracy but is slow as a general tool. There is a much faster method, discovered in 1991/1992 by Fernando and Parlett, see [12], that computes the eigenvalues of $LL^t$ using the ideas in this section.

If $J = LU$ then the LR transform of $J$ is $\hat{J} = UL$. When shifts are incorporated we obtain

LR($\tau$): factor $J - \tau I = LU$,
form $\hat{J} = UL + \tau I$.

Thus $\hat{J} = L^{-1}(J - \tau I)L + \tau I = L^{-1}JL$.

The LR algorithm consists of iterating the LR transform with well chosen shifts. It was presented by Rutishauser in 1957, see [21], along with a proof of its surprising convergence property. For simplicity take all shifts to be zero. Then if $J$ has eigenvalues with distinct absolute values and if the factorization does not fail then, very slowly, the $q$-values tend to the eigenvalues in monotonic decreasing order, i.e., $q_n$ tends to the eigenvalue closest to 0. The $e$-values tend to zero.

Rutishauser implemented the LR transform so that $\hat{J}$ overwrote $J$ with no explicit reference to $L$ and $U$. Today the LR algorithm is remembered, if at all, as the algorithm that led to the celebrated QR algorithm which is a little slower than LR but never breaks down and is backward stable.

With our preference for $L$, $U$ over $J$ we let $\hat{J} = \hat{L}\hat{U}$ and want to compute $\hat{L}$ and $\hat{U}$, without reference to $J$ from $\hat{L}\hat{U} = UL - \tau I$. This may be accomplished by the qds algorithm that was discovered by Rutishauser in 1953/54, see [20], some years before he saw that qds was equivalent to LR in exact arithmetic.

qds($\tau$):
$\hat{q}_1 = q_1 + e_1 - \tau$
**for** $i = 1, n-1$ **do**
$\hat{e}_i = e_i * q_{i+1}/\hat{q}_i$
$\hat{q}_{i+1} = q_{i+1} + e_{i+1} - \tau - \hat{e}_i$
**end for**

This is not the most accurate implementation. There is a *differential* form of qds($\tau$) that was discovered by Fernando and Parlett as late as 1991, see [12].

dqds($\tau$):
$p_1 = q_1 - \tau$
**for** $i = 1, n-1$ **do**
$\hat{q}_i = p_i + e_i$
$\hat{e}_i = e_i * q_{i+1}/\hat{q}_i$
$p_{i+1} = p_i * q_{i+1}/\hat{q}_i - \tau$
**end for**
$\hat{q}_n = p_n$

An examination of both algorithms shows that the shift is not restored; $\hat{J} = \hat{L}\hat{U} = UL - \tau I$. Thus all eigenvalues have been reduced by $\tau$. This feature has advantages although it is troublesome to people familiar with the QR algorithm. It becomes necessary to keep a running sum $\sigma$ of all shifts used in order to recover the original eigenvalues. In practice, with dqds, the program checks when $e_{n-1}$ and $q_n$ are both negligible and then records $\sigma$ as an eigenvalue and reduces the order $n$ by 1.

The advantage of dqds over qds is that it enjoys high mixed relative stability even in the presence of element growth. Small end-figure changes to the input $L$, $U$ and to the output $\hat{L}$, $\hat{U}$ give an exact transformation and this feature permits all the eigenvalues to be computed to high relative accuracy *in the positive case*. When the original $J$ comes from a positive definite symmetric matrix $T$, via $J = \Delta T \Delta^{-1}$, then all $q$'s and $e$'s will be positive. The shifts must be chosen carefully and details can be found in [12,19]. The latest implementation is in LAPACK, see [1], and is almost as fast as the root free QR method that computes eigenvalues with errors $O(\varepsilon||T||)$, not $O(\varepsilon|\lambda|)$.

Apart from its use in computing eigenvalues the dqds($\sigma$) transform plays a role in computing 'twisted' factorizations of tridiagonals that are needed in the course of computing eigenvectors. We omit this material and refer the reader to [15,16].

## 4. A small example

The ideas sketched in Section 1 may be illustrated on a $4 \times 4$ matrix. This matrix is similar to one used by Dhillon in his dissertation [7]. It is contrived to produce an intermediate representation ($LDL^t$) that looks bad because it suffers severe element growth, $||L|| = 10^7||T||$. Nevertheless the representation is good enough for its special purpose.

Let $\varepsilon$ denote the roundoff unit for Matlab ($=2 \times 10^{-16}$) and let $\eta := \sqrt{\varepsilon}$. The tridiagonal $T$ is given by

$$\text{diagonal} = (1+\eta, 1-2\eta, 1+3\eta, 1+2\eta),$$

$$\text{off-diagonal} = (\sqrt{2}/2, \sqrt{2}/2, \eta).$$

The eigenvalues are, approximately,

$$-\varepsilon,\ 1+\tfrac{4}{3}\eta,\quad 1+\tfrac{8}{3}\eta,\quad 2+\varepsilon,$$

| Off-diag | Diag | Eigenvalues |
|---|---|---|
| 7.071067811865476e − 01 | 1.000000014901161e + 00 | 2.000000000000001e + 00 |
| 7.071067811865476e − 01 | 9.999999701976776e − 01 | 1.000000040339034e + 00 |
| 1.490116119384766e − 08 | 1.000000044703484e + 00 | 1.000000019265610e + 00 |
| 0 | 1.000000029802322e + 00 | −6.890205972143757e − 16 |

Matlab has no trouble computing an orthonormal set of eigenvectors because it uses the QR algorithm. We ignore the extreme eigenvalues and focus on the pair close to 1 whose separation is $\frac{4}{3}\eta = \mathrm{O}(\sqrt{\varepsilon})$.

First we performed standard inverse iteration using a good starting vector. Each computed vector $\boldsymbol{x}$ has an excellent residual norm; $||T\boldsymbol{x} - \boldsymbol{x}\lambda|| = \mathrm{O}(\varepsilon)$. The dot product between them is

$$\mathrm{O}(\sqrt{\varepsilon}) = \frac{\mathrm{O}(\varepsilon)}{\text{gap}} = \frac{\mathrm{O}(\varepsilon)}{4/3\eta}$$

as expected by standard theory, see Section 1. This is not good enough.

Next, we pursued a simple variation of inverse iteration. Since our two eigenvalues agree to eight decimals we may translate $T$ to $T - I$ and find that the shifted eigenvalues have no digits in common. We try inverse iteration again, using $T - I$, and find an improvement. The dot product is $10^{-10}$ instead of $10^{-8}$. This is not good enough. The calculations are shown in Fig 1.

Before proceeding to the central idea of new representations we discuss the starting vector for inverse iteration. The last entry in the two eigenvectors we seek is dominant and, in order to keep the example simple, we choose a special multiple of $e_4 := (0, 0, 0, 1)^t$ as the starting vector in all cases.

This special multiple simplifies the calculations. In each case we factor a nearly singular matrix

$$\mathring{L}\mathring{D}\mathring{L}^t - \tau I = LDL^t$$

and the approximate eigenvector $\boldsymbol{x}$ is computed from

$$LDL^t\boldsymbol{x} = \boldsymbol{e}_4\mu.$$

| Eigenvalues | | |
|---|---|---|
| $\lambda$ (for $T$) | | $\mu$ (for $T-I$) |
| -6.890205972143757e-16 | | -1.000000000000000e+00 |
| 1.000000019265610e+00 | | 1.926561034787264e-08 |
| 1.000000040339034e+00 | | 4.033903460463116e-08 |
| 2.000000000000001e+00 | | 1.000000000000000e+00 |
| Factor $T-\lambda_2 I$ | | |
| multipliers | pivots | Eigenvector |
| -1.620151311728313e+08 | -4.364449024407691e-09 | 7.071067783870038e-01 |
| 6.172263002603381e-09 | 1.145619979071370e+08 | 4.364449007128170e-09 |
| 7.071067783870034e-01 | 2.107342433887993e-08 | -7.071067783870034e-01 |
| 0 | 8.343291570375477e-17 | 1.000000000000000e+00 |
| $\|(T-\lambda_2 I)x\| = 8.343291570375483e-17$ | | |
| Factor $(T-I)-\mu_2 I$ | | |
| multipliers | pivots | Eigenvector |
| -1.620151263612359e+08 | -4.364449154024987e-09 | 7.071067870854733e-01 |
| 6.172263185909920e-09 | 1.145619945048253e+08 | 4.364449190434709e-09 |
| 7.071067870854730e-01 | 2.107342407964534e-08 | -7.071067870854730e-01 |
| 0 | -1.758016782764627e-16 | 1.000000000000000e+00 |
| $\|((T-I)-\mu_2 I)x\| = 1.375995657035863e-16$ | | |
| Factor $T-\lambda_3 I$ | | |
| multipliers | pivots | Eigenvector |
| -2.779740182130631e+07 | -2.543787314124302e-08 | -7.071067932881644e-01 |
| 3.597458519427228e-08 | 1.965573132721291e+07 | -2.543787357659366e-08 |
| -7.071067932881619e-01 | -2.107342389479081e-08 | 7.071067932881619e-01 |
| 0 | 3.606562560959997e-16 | 1.000000000000000e+00 |
| Factor $(T-I)-\mu_3 I$ | | |
| multipliers | pivots | Eigenvector |
| -2.779740152676420e+07 | -2.543787341078350e-08 | -7.071067751996087e-01 |
| 3.597458557546010e-08 | 1.965573111894018e+07 | -2.543787319540585e-08 |
| -7.071067751996061e-01 | -2.107342443387178e-08 | 7.071067751996061e-01 |
| 0 | -1.784247126666214e-16 | 1.000000000000000e+00 |
| dot prod from $\lambda$'s: -1.315511610755493e-08<br>dot prod from $\mu$'s: 1.244713221382199e-10 | | |

Fig. 1. Inverse iteration.

We chose $\mu = D_{4,4}$. Since $L$ is unit lower triangular $Le_4 = e_4$ and our choice of $\mu$ yields $L^{\mathrm{t}}x = e_4$. Backsolving yields

$$x = \begin{pmatrix} -\ell_1\ell_2\ell_3 \\ +\ell_2\ell_3 \\ -\ell_3 \\ 1 \end{pmatrix},$$

where $\ell_i = L(i+1,i)$, $i = 1,2,3$. Thus $||x||_2 > 1$ and the accuracy of $x$ is completely determined by the accuracy of $L$. In exact arithmetic

$$||(\mathring{L}\,\mathring{D}\,\mathring{L}^{\mathrm{t}} - \tau I)x|| = ||LDL^{\mathrm{t}}x|| = |D_{4,4}|$$

and, when $\tau$ is very accurate, then $D_{4,4}$ can be $O(\varepsilon|\tau|)$. The roundoff errors in computing $(\ell_2\ell_3)$ and $\ell_1(\ell_2\ell_3)$ make negligible difference. We ignore them here and refer to [16] for the way those roundoff errors may be dealt with in general.

In the figures that follow we exhibit the nontrivial entries in $L$, $D$, and $\boldsymbol{x}$ for various cases.

First, we compared inverse iteration on $T$ and $T-I$, see Fig. 1. The computed vectors are not orthogonal to working accuracy.

Next, we abandon $T$ and $T-I$ and take as our representation $L_1D_1L_1^t = T-I$. This looks bad because of element growth.

| Lower part of $L_1$ | Diagonal of $D_1$ |
|---|---|
| 4.745313281212578e + 07 | 1.490116119384766e − 08 |
| −2.107342425544699e − 08 | −3.355443200000004e + 07 |
| 2.500000000000001e − 01 | 5.960464477539061e − 08 |
| 0 | 2.607703208923340e − 08 |

$||(T-I)-L_1D_1L_1^t|| = 0.$

The computed product $L_1D_1L_1^t$ turned out to equal $T-I$ exactly.

We computed the eigenvalues $v$ of $L_1D_1L_1^t$ by bisection but never formed the product $L_1D_1L_1^t$. For each sample $\tau$ we computed $L_1D_1L_1^t - \tau I = LDL^t$ using the differential form of the stationary qd algorithm (Section 3) and counted the number of negative diagonal entries in $D$.

The eigenvalues $v_2$, $v_3$ differ from the $\mu_2$, $\mu_3$ computed by Matlab from $T-I$ in their last eight digits. This is because $L_1D_1L_1^t$ defines its two tiny eigenvalues to high relative accuracy despite the element growth. The large eigenvalues are not so well represented. There is a precise 'relative condition number', greater than or equal to one, that measures the relative change in an eigenvalue due to tiny relative changes in the parameters in $L_1$ and $D_1$. The condition numbers of our two eigenvalues $v_2$ and $v_3$ are less than 3 whereas the condition for the two extreme eigenvalues, near $-1$ and $+1$, are about $10^8$.

Fig. 2 shows the difference made by using $v_2$ and $v_3$ instead of $\mu_2$ and $\mu_3$ to obtain new factorizations. Notice how the last pivot changes from $10^{-16}$ to $10^{-23}$. The improved eigenvalues coupled with the high accuracy of the differential stationary qd algorithm combine to correct the lower halves of the entries of the $L$ factors and so give fully accurate eigenvectors of $L_1D_1L_1^t$. The computations are shown in Fig. 2.

We must mention that in this example $T-I$ also defines its two small eigenvalues to high relative accuracy. If we discard Matlab's eigenvalues $\mu_2$ and $\mu_3$ and use bisection we get $v_2$ and $v_3$ instead. If we then use these eigenvalues in inverse iteration starting from $e_4$ we get the same excellent eigenvectors as in the new method. In this case the diagonal entries of $T-I$ have the same exponent as the two small eigenvalues and the subtraction in the shift is done exactly. The point is that in general the standard representation does not define the small eigenvalues to this high relative accuracy. This example shows that element growth in the factorization does not stop the small eigenvalues being well determined by the triangular factors.

## 5. Unsymmetric case

This case needs more attention. Real matrices may have complex eigenvalues and, of more concern, some eigenvalues may be extremely sensitive to small changes in the matrix while others may be

| Eigenvalues | | |
|---|---|---|
| $\mu$ (for $T-I$) | | $\nu$ (for $L_1D_1L_1^t$) |
| -1.000000000000000e+00 | | -1.000000000000000e+00 |
| 1.926561034787264e-08 | | 1.926561025997181e-08 |
| 4.033903460463116e-08 | | 4.033903451541880e-08 |
| 1.000000000000000e+00 | | 1.000000000000000e+00 |
| Factor $L_1D_1L_1^t-\nu_2 I$ | | |
| multipliers | pivots | Eigenvector |
| -1.620151296242512e+08 | -4.364449066124159e-09 | 7.071067811865487e-01 |
| 6.172263061599378e-09 | 1.145619968121254e+08 | 4.364449066124166e-09 |
| 7.071067811865482e-01 | 2.107342425544699e-08 | -7.071067811865482e-01 |
| 0 | -1.985233470127266e-23 | 1.000000000000000e+00 |
| Factor $(T-I)-\mu_2 I$ | | |
| multipliers | pivots | Eigenvector |
| -1.620151263612359e+08 | -4.364449154024987e-09 | 7.071067870854733e-01 |
| 6.172263185909920e-09 | 1.145619945048253e+08 | 4.364449190434709e-09 |
| 7.071067870854730e-01 | 2.107342407964534e-08 | -7.071067870854730e-01 |
| 0 | -1.758016782764627e-16 | 1.000000000000000e+00 |
| Factor $L_1D_1L_1^t-\nu_3 I$ | | |
| multipliers | pivots | Eigenvector |
| -2.779740162425157e+07 | -2.543787332157115e-08 | -7.071067811865480e-01 |
| -7.071067811865458e-01 | -2.107342425544707e-08 | 7.071067811865458e-01 |
| 0 | -1.323488980084844e-23 | 1.000000000000000e+00 |
| Factor $(T-I)-\mu_3 I$ | | |
| multipliers | pivots | Eigenvector |
| -2.779740152676420e+07 | -2.543787341078350e-08 | -7.071067751996087e-01 |
| 3.597458557546010e-08 | 1.965573111894018e+07 | -2.543787319540585e-08 |
| -7.071067751996061e-01 | -2.107342443387178e-08 | 7.071067751996061e-01 |
| 0 | -1.784247126666214e-16 | 1.000000000000000e+00 |
| dot prod from $\mu$'s: 1.244713221382199e-10 | | |
| dot prod from $\nu$'s: -6.661338147750939e-16 | | |

Fig. 2. New method.

robust. Unsymmetric tridiagonal matrices arise as output from the (two-sided) Lanczos algorithm applied to a large sparse general matrix. The lack of a good tridiagonal eigensolver to complete the calculation has hindered the acceptance of the unsymmetric Lanczos algorithms for large sparse problems.

It is not easy to find an algorithm that preserves both the eigenvalues and the tridiagonal form. None of the current methods is, in addition, backward stable. A method is backward stable if the computed eigenvalues are exact for some matrix close to the original one.

One of the earliest methods was the LR algorithm described Section 3. In 1978 came the HR algorithm of Bunse–Gerstner, see [3,4]. In 1992 came XHR from Parlett and Liu, see [14]. In 1996 and 1998 came two related methods by Uhlig called DQR and IQR PWK, see [24,23]. All of these methods work with a tridiagonal matrix.

The ideas described in Sections 1 and 3 suggest that the triangular factors might be a preferable representation to their product even in the unsymmetric case. If this is so, at least in important special cases, then we are fortunate that we have an algorithm at hand, namely dqds, that avoids the loss of information inherent in explicitly forming the product of bidiagonals.

The nice high mixed relative stability property mentioned in Section 3 extends without change to this case. Two practical lessons have been learned in working with the LR algorithm. First, by doubling the storage (from $2n$ to $4n$ cells) a transformation may be computed and then either accepted or rejected. Thus, unsatisfactory transformations merely waste a little time. They may be discarded and a better shift invoked. Second, the standard simple shift strategies based on asymptotic properties need to be supplemented with sophisticated choices in the early stages of the process.

The motivation for the method described in Sections 1 and 3 was to compute orthogonal eigenvectors. Of course, this is out of the question in the unsymmetric case because the eigenvectors need not be orthogonal. Recall that in the symmetric case we achieved orthogonality *indirectly* by attempting to compute accurate eigenvectors. This goal we can retain. The reason for hope is that the high relative accuracy property of dstqds and dqds is independent of symmetry. When $LU$ defines its small eigenvalues to high relative accuracy then we should achieve (1), the small relative residual property.

A prototype implementation of dqds algorithm for eigenvalues entirely in complex arithmetic has been used with excellent results by David Day in building a nonsymmetric Lanczos algorithm for large sparse matrices. See [6].

There is room for more investigation on this topic.

## References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen, LAPACK Users' Guide, 2nd Edition, SIAM, Philadelphia, 1995, 324pp.

[2] J. Barlow, J.W. Demmel, Computing accurate eigensystems of scaled diagonally dominant matrices, SIAM J. Num. Anal. 27 (1990) 762–791.

[3] A. Bunse-Gerstner, Berechnung der Eigenwerte einer Matrix mit dem *HR*-Verfahren (German and English summary), Numerische Behandlung von Eigenwertaufgaben, Band 2 (Tagung, Tech. Univ. Clausthal, Clausthal, 1978), pp. 26–39, Internat. Schriftenreihe Numer. Math., 43, Birkhauser, Basel-Boston, MA, 1979.

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

[5] J. Demmel, Applied Numerical Algebra, SIAM, Philadelphia, 1997.

[6] D. Day, An efficient implementation of the nonsymmetric Lanczos algorithm, SIAM J. Matrix Anal. Appl. 18 (1997) 566–589.

[7] I.S. Dhillon, A new $O(n^2)$ algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem, Ph.D. Thesis, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, California, May 1997. Also available as Computer Science Division Technical Report No. UCB//CSD-97-971.

[8] I.S. Dhillon, G.I. Fann, B.N. Parlett, Application of a new algorithm for the symmetric eigenproblem to computational chemistry, in: Proceedings of the Eigth SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, SIAM, Philadelphia, March 1997.

[9] S. Eisenstat, I.C.F. Ipsen, Relative perturbation bounds for eigenspaces and singular vector subspaces, in: J.G. Lewis (Ed.), Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, SIAM, Philadelphia, 1994, pp. 62–65.

[10] S. Eisenstat, I.C.F. Ipsen, Relative perturbation techniques for singular value problems, SIAM J. Numer. Anal. 32 (1995) 1972–1988.

[11] G.I. Fann, R.J. Littlefield, Performance of a fully parallel dense real symmetric eigensolver in quantum chemistry applications, Proceedings of High Performance Computing'95, Simulation MultiConference, The Society for Computer Simulation, 1995.

[12] K.V. Fernando, B.N. Parlett, Accurate singular values and differential qd algorithms, Numer. Math. 67 (1994) 191–229.

[13] G.J.F. Francis, The QR Transformation, Parts I and II, Comput. J. 4 (1961-62), 265–271, 332–345.
[14] Z.S. Liu, On the extended HR algorithm, Ph. D. Thesis, Technical Report CPAM-564 (August 1992), Center for Pure and Applied Math., Univ. of California, Berkeley.
[15] B.N. Parlett, I.S. Dhillon, Fernando's solution to Wilkinson's problem: an application of double factorization, Linear Algebra Appl. 267 (1997) 247–279.
[16] I.S. Dhillon, B.N. Parlett, Orthogonal eigenvectors and relative gaps, submitted to SIAM J. Matrix Anal. Appl.
[17] B.N. Parlett, I.S. Dhillon, Relatively robust representations for symmetric tridiagonal matrices, Linear Algebra Appl. 309 (2000) 121–151.
[18] B.N. Parlett, The Symmetric Eigenvalue Problem, 2nd Edition, SIAM, Philadelphia, 1998, 398pp.
[19] B.N. Parlett, O.A. Marques, An implementation of the dqds algorithm (Positive case), Linear Algebra Appl. 309 (2000) 217–259.
[20] H. Rutishauser, Der quotienten-differenzen-algorithmus, Z. Angew. Math. Phys. 5 (1954) 233–251.
[21] H. Rutishauser, Solution of eigenvalue problems with the LR-transformation, Nat. Bur. Standards Appl. Math. Ser. 49 (1958) 47–81.
[22] H. Rutishauser, Lectures on Numerical Mathematics, Birkhäuser, Boston, 1990.
[23] F. Uhlig, A fast and simple method to compute the eigenvalues of general complex tridiagonal matrices, preprint, Olga Tausssky-Todd Memorial Meeting, June 1996.
[24] F. Uhlig, The *DQR* algorithm, basic theory, convergence, and conditional stability, Numer. Math. 76 (1997) 515–553.