REDUCTION OF A GENERAL MATRIX TO TRIDIAGONAL FORM *

GEORGE A. GEIST †

Abstract. An algorithm for reducing a nonsymmetric matrix to tridiagonal form as a first step toward finding its eigenvalues is described. The algorithm uses a variation of threshold pivoting, where at each step, the pivot is chosen to minimize the maximum entry in the transformation matrix that reduces the next column and row of the matrix. Situations are given where the tridiagonalization process breaks down, and two recovery methods are presented for these situations. Although no existing tridiagonalization algorithm is guaranteed to succeed, this algorithm is found to be very robust and fast in practice. A gradual loss of similarity is also observed as the order of the matrix increases.

Key words. tridiagonalization, nonsymmetric, eigenvalues

AMS(MOS) subject classifications. 15

1. Introduction. The standard method for computing all of the eigenvalues of a dense matrix is based on the QR iteration scheme [5]. In this scheme, orthogonal similarity transformations are successively applied to the matrix to reduce it to quasi-triangular form, so that the eigenvalues appear on the diagonal. Repeated application of these transformations to a general matrix is prohibitively expensive, however, so that in practice the original matrix is first reduced to a simpler form that can be preserved during the subsequent iterative phase. For a general matrix, the initial reduction is usually to upper Hessenberg form (upper triangular except for one additional subdiagonal) by elementary or orthogonal similarity transformations. The initial reduction to Hessenberg form requires $O(n^3)$ operations, where n is the order of the matrix. Computation of the eigenvalues of the reduced matrix usually requires only a few QR iterations per eigenvalue, totaling another $O(n^3)$ operations. Both the initial and iterative phases are costly, but less costly than iterating directly with the original matrix. This two-phase approach is implemented in the standard EISPACK software for the general eigenvalue problem [16].

If the original matrix is symmetric, then that symmetry can be preserved by using orthogonal transformations in the initial reduction, so that the result is in fact tridiagonal. Although the reduction to tridiagonal form costs $O(n^3)$ operations, the subsequent iterations preserve the tridiagonal form and are much less expensive, so that the total cost of the iterative phase is reduced to $O(n^2)$ operations. Again, standard software is available in EISPACK implementing this two-phase approach for the symmetric case [16].

The attractively low operation count of iterating with a tridiagonal matrix suggests that the tridiagonal form would be extremely beneficial in the nonsymmetric case as well. There are two difficulties with such an approach: First, QR iteration does not preserve the structure of a nonsymmetric tridiagonal matrix. This problem can be overcome by using LR iteration [15] instead, which preserves the tridiagonal form. Second, it is difficult to reduce a nonsymmetric matrix to tridiagonal form by similarity transformations in a numerically stable manner. This second problem is

* Received by the editors April 17, 1989; accepted for publication (in revised form) May 3, 1990.

† Mathematical Sciences Section, Oak Ridge National Laboratory, P.O. Box 2009, Oak Ridge, Tennessee 37831-8083. This research was supported by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

the primary focus of this paper.

The following notational conventions will be used throughout this paper. Lower case Greek letters will denote scalars; lower case Latin letters will denote vectors. Components of vectors are denoted by subscripts. Upper case Latin letters will denote square matrices and a single subscript, when present, denotes the matrix dimension. Throughout this paper, N is used to represent a matrix that applies a rank one change to another matrix. Special cases of N include N_c and N_r, which zero out the next column and row of a matrix, respectively.

In the early 1960's there was a great amount of interest and research devoted to finding a stable way to reduce a general matrix via similarity transformations to tridiagonal form [12], [14], [17]. The problem is addressed in some detail by Wilkinson [21], and several algorithms are given, but the overall conclusion was that no general purpose algorithm existed. Because of the success and numerical stability of the QR iteration scheme, little research was directed at the problem of reduction to tridiagonal form for nearly 15 years.

One reason for renewed interest in tridiagonalization is the relatively poor performance of the QR iteration on advanced computers. Algorithms for vector supercomputers [10] and parallel architectures [7] have been developed for reducing nonsymmetric matrices to tridiagonal form.

In 1981 Dax and Kaniel published a paper [2] that inspired most of the recent interest in the problem. They describe experiments with reduction from upper Hessenberg form to tridiagonal form using elementary similarity transformations. During the reduction, they monitor the size of the multipliers as follows. They define a *control parameter* for the reduction of row k as $m_k = \max_{i>k+1}\{|H_{k,i}/H_{k,k+1}|\}$. If m_k is greater than a specified value, μ, then *breakdown* is said to have occurred, and their algorithm aborts. They observe that for 100 random test matrices of order 50×50 the number of breakdowns as a function of the specified value μ is:

$\mu = 2^r$	r =	16	12	10	8	7
breakdowns		0	1	5	20	41

Dax and Kaniel refer to Wilkinson's detailed error analysis in [21] and conclude that with judicious use of double precision there is a low probability of having large errors in eigenvalues computed with the tridiagonal matrix, even when using control parameters as large as 2^{16}.

Wachspress [18] and Watkins [19] focus on the fact that in [2] Dax and Kaniel did not address possible ways to recover from breakdown during the reduction to tridiagonal form. Wilkinson states [21, p.404] that

> If breakdown occurs in the rth step of the reduction of a Hessenberg matrix to tridiagonal form we must return to the beginning and compute NAN^{-1} for some N in the hope that failure will be avoided in this matrix.

This recovery method is actually too restrictive. Wachspress and Watkins both describe efficient methods for finding matrices similar to A without returning to the beginning and wasting work already performed on the matrix. Hare and Tang [11] describe a combination of recovery methods and also investigate the effects of interleaving orthogonal and elementary similarity transformations during the tridiagonalization to reduce the number of multipliers that are greater than one.

In the next section we describe the inherent problems of tridiagonalizing nonsymmetric matrices. In §3 we present a reduction algorithm that incorporates a pivoting scheme designed to produce better conditioned transformation matrices than previous algorithms. We describe two recovery algorithms in §4 that significantly improve the robustness of the reduction algorithm. Section 5 presents empirical results showing the accuracy and performance of the new algorithm. While no finite stable tridiagonalization algorithm is known [13], the new algorithm significantly broadens the class of matrices that can be successfully reduced.

2. Tridiagonalization. The direct reduction of a general matrix to tridiagonal form is difficult because the elementary similarity transformations, which must be used at some point in the reduction, may have large multipliers. This phenomenon is illustrated by the following example. First note that computations of the form

$$\begin{pmatrix} I_{k-1} & & \\ & 1 & \\ & & G_{n-k} \end{pmatrix} \begin{pmatrix} F_{k-1} & & \\ & \alpha & w^T \\ & v & B_{n-k} \end{pmatrix} \begin{pmatrix} I_{k-1} & & \\ & 1 & \\ & & G^{-1} \end{pmatrix} = \begin{pmatrix} F_{k-1} & & \\ & \alpha & w^T G^{-1} \\ & Gv & GBG^{-1} \end{pmatrix}$$

preserve the inner product of the kth row and column, since $w^T G^{-1} G v = w^T v$. The tridiagonalization algorithms in [2], [6], [10], [11], [12], [17], [19] are all affected by this property.

Let the partially reduced matrix have the form shown in Fig. 1. Let $w^T v = 0$

$$\left(\begin{array}{cc|cc} T_{k-1} & & & \\ & & \times & \\ \hline & \times & \alpha & w^T \\ & & v & B_{n-k} \end{array}\right)$$

FIG. 1. *Partially reduced matrix.*

and $\bar{v} = Gv$, where G is designed to eliminate all but the first element of v. Let $\bar{w}^T = w^T G^{-1}$ and partition $\bar{w}^T = (\bar{w}_1 \tilde{w}^T)$. Since $w^T v = \bar{v}_1 \bar{w}_1$, $\bar{w}_1 = 0$. After all but the first entry of v have been eliminated, the matrix has the form

$$\left(\begin{array}{cc|ccc} T_{k-1} & & & & \\ & & \times & & \\ \hline & \times & \alpha & 0 & \tilde{w}^T \\ & & \bar{v}_1 & & \\ & & 0 & & \bar{B}_{n-k} \end{array}\right).$$

Any attempt to avoid the use of the zero as the pivot now destroys the existing tridiagonal form. This zero pivot will occur regardless of the pivot selection in v or whether orthogonal transformations are used to eliminate v.

Algorithms that include a stable reduction to upper Hessenberg form as an initial step to tridiagonal form will likely encounter small pivots during the reduction of the rows. Stable reduction of the columns tend to make $\bar{v}_1$ large. For example, stable elementary transformations choose $\bar{v}_1 = \max(v_i)$, and orthogonal transformations make $\bar{v}_1 = ||v||_2$. Let $\bar{w}_1$ be the first entry in $w^T G^{-1}$. Since the product of $\bar{v}_1$ and $\bar{w}_1$, the eventual pivot for the row, is fixed, $\bar{w}_1$ tends to be small, which can lead to breakdown when reducing the rows.

If $w^T v = 0$, then a breakdown condition will occur no matter what transformation is used. In this case, the algorithm must abort or apply some recovery method.

3. A tridiagonalization algorithm. In this section we present an algorithm that reduces the matrix directly to tridiagonal form by eliminating columns and rows using elementary similarity transformations so that the matrix always has the form shown in Fig. 1. This matrix structure allows us the freedom to pivot at each step to improve the overall stability of the algorithm. For example, the pivot could be chosen to minimize the maximum multiplier in the column and row reduction, or the pivot could be chosen to minimize the condition number of the transformation matrices. While these pivoting heuristics work well, the heuristic we found that works at least as well and sometimes better is to choose the pivot that minimizes the norm of the transformation matrix that reduces both the column and row. If C denotes this transformation matrix, then the norm used is $n\{\max|C_{ij}| : i,j = 1,2,\cdots,n\}$ because it can be computed in constant time for each possible permutation.

At step k of the algorithm shown in Fig. 2, the matrix has the form shown in Fig. 1. If v or $w = 0$, then the matrix has been deflated, and step k can be skipped. Otherwise the algorithm finds the permutation that minimizes the maximum element in $N_r^{-1}N_c$, where N_r and N_c are elementary matrices such that $N_cAN_c^{-1}$ reduces column k and $N_r^{-1}(N_cAN_c^{-1})N_r$ reduces row k.

This minimization can be done efficiently because of the special structure of $N_r^{-1}N_c$, which is

$$\left(\begin{array}{c|cc} I_k & & \\ \hline & \gamma & u^T \\ & x & I_{n-k-1} \end{array}\right).$$

The vector x contains the multipliers used in reducing column k, and u contains the negatives of the multipliers used in reducing row k. The pivoting algorithm shown in Fig. 3 finds the permutation at step k that minimizes the maximum multiplier used in the column and row reduction and γ. The term γ equals $1 - u^Tx$, which can be simplified to w_1v_1/w^Tv.

If $w^Tv = 0$, then the minimization problem has no solution. In this case $\max(|v_i|, |w_i|)$ is permuted into the pivot location before calling the recovery routine, FIXUP (see Fig. 4). The recovery routine is also called when the maximum element in $N_r^{-1}N_c$ exceeds a bound set by the user. If the maximum element is less than the bound, then the algorithm simply reduces column k followed by row k.

CLAIM. *The minimization problem can be solved in $O(n-k)$ time by observing that for a given permutation, the maximum multipliers in column k and row k, respectively, are:*

$$\begin{aligned} m_c &= \frac{\max_{i>1}|v_i|}{|v_1|} \\ m_r &= \frac{|v_1|\max_{i>1}|w_i|}{|w^Tv|}. \end{aligned}$$

Proof. Using Fig. 1 as a reference, given that column k is reduced first by an elementary similarity transformation $N_cAN_c^{-1}$, the expression for m_c is obvious. The form of N_c is

$$\left(\begin{array}{c|c} I_k & \\ \hline & G_{n-k} \end{array}\right)$$

```
A2TRI( a, n, tol )

    maxtol = tol
    cnt = 0
    m = 1
    for k = 1 to n - 2
label:
        Check number of recovery attempts
        if(cnt > 2) then
            maxtol = 10 * maxtol, print warning of increase.
            cnt = 0
            if(maxtol > 10 * tol) return and execute NEWSTART
        end if
        Find suitable pivot
        PIVOT(a, n, k, piv, maxmult, err)
        Check for deflation
        if( err = 1 ) m = k + 1, next k
        Interchange row(piv) and row(k)
        Interchange column(piv) and column(k)
        Check maximum multiplier against tolerance
        if( err = 2 or maxmult > maxtol ) then
            FIXUP( a, k, m, n )
            cnt = cnt + 1, print warning
            go to label:
        endif
        Zero out column k
        for i = k + 2 to n
            for j = k + 1 to n
                a_ij = a_ij - a_k+1j * a_ik / a_k+1k
            for j = k to n
                a_jk+1 = a_jk+1 + a_ij * a_ik / a_k+1k
        Zero out row k
        for i = k + 2 to n
            for j = k + 1 to n
                a_k+1j = a_k+1j - a_ij * a_ki / a_kk+1
            for j = k + 1 to n
                a_ji = a_ji + a_jk+1 * a_ki / a_kk+1
    end for
```

FIG. 2. *Algorithm for reducing an $n \times n$ matrix a to tridiagonal form while trying to bound all multipliers below tol.*

PIVOT(a, n, k, piv, $maxmult$, err)

```
err = 0
maxmult = ∞
Find maximum and next-to-maximum entries in row k and column k
maxcol = max( |a_ik| |i = k+1 to n)
pivc = index of maxcol
nmxcol = next-to-max( |a_ik| |i = k+1 to n)
maxrow = max( |a_ki| |i = k+1 to n)
pivr = index of maxrow
nmxrow = next-to-max( |a_ki| |i = k+1 to n)
inprod = sum_{i=k+1}^{n} a_ki * a_ik

Check if maximum element in row or column is zero
if( maxcol = 0 or maxrow = 0 ) err = 1, return
Check if inner product is zero
if( inprod = 0 ) then
    piv = index of max(maxcol, maxrow)
    err = 2
    return
endif
Calculate maximum entry of (N_r N_c)_i over all permutations i
for i = k+1 to n
    if( i = pivc ) maxnc = |nmxcol/a_ik|
    else maxnc = |maxcol/a_ik|
    if( i = pivr ) maxnr = |a_ik * nmxrow/inprod|
    else maxnr = |a_ik * maxrow/inprod|
    maxdiag = |a_ik * a_ki/inprod|
    temp = max(maxnr, maxnc, maxdiag)
    if( temp < maxmult ) then
        maxmult = temp
        piv = i
    endif
end for
```

FIG. 3. *Algorithm for finding the pivot that minimizes the maximum element in $N_r^{-1}N_c$ where $N_r^{-1}N_cAN_c^{-1}N_r$ reduces column k and then row k of the $n \times n$ matrix A.*

```
FIXUP( a, k, m, n )

    Apply a random shift
    r = random()
    a_mm = a_mm + r * a_m+1m
    a_mm+1 = a_mm+1 + r * (a_m+1m+1 - a_mm)
    a_mm+2 = r * a_m+1m+1

    Chase bulge down to row k - 1
    for i = m + 1 to k - 1
        m = a_i-1i+1 / a_i-1i
        a_i-1i+1 = 0
        a_ii = a_ii + m * a_i+1i
        a_ii+1 = a_ii+1 + m * (a_i+1i+1 - a_ii)
        a_ii+2 = m * a_i+1i+2
        a_i+1i+1 = a_i+1i+1 - m * a_i+1i
    end for
    Fill in row k - 1
    if (k = m + 1) m = r
    for i = k + 2 to n - 1
        a_k-1i = m * a_ki
    end for
    Eliminate row k - 1
    for i = k + 1 to n - 1
        m = a_k-1i / a_k-1k
        a_k-1i = 0
        for j = k to n - 1
            a_kj = a_kj + m * a_ij
        for j = k to n - 1
            a_ji = a_ji - m * a_jk
    end for
```

FIG. 4. *Recovery algorithm to apply an implicit single-shift* LR *iteration to rows m through k of the partially reduced matrix.*

and after the transformation is applied, $\bar{v} = Gv$ and $\bar{w}^T = w^T G^{-1}$. Thus, $\bar{w}^T \bar{v} = w^T G^{-1} G v = w^T v$. Since $\bar{v}_i = 0$ for $i > 1$, $\bar{w}_1 \bar{v}_1 = w^T v$. Since N_c is elementary, $\bar{v}_1 = v_1$ so $v_1 \bar{w}_1 = w^T v$ or $\bar{w}_1 = w^T v / v_1$. Therefore,

$$m_r = \frac{\max|w_i|}{|\bar{w}_1|} = \frac{|v_1| \max|w_i|}{|w^T v|} \quad ; \qquad i > 1.$$

For each possible choice of permutation only three terms must be evaluated: m_c, m_r, and γ. At step k there are only $n-k-1$ possible permutations. Thus the permutation that minimizes the maximum element can be found in $O(n-k)$ time, which totals $O(n^2)$ for the entire reduction.

The complexity of the overall tridiagonalization algorithm given in Fig. 2 is $(4/3)n^3 + O(n^2)$ flops, where a flop is defined as a floating point operation of the form $a + b * c$. This complexity is based on the assumption that the recovery routines, which we discuss in the next section, are called only a constant number of times.

An error analysis of tridiagonalization methods is given in [21]. Dax and Kaniel [2] also give an error analysis that shows that the bound on the eigenvalue errors depends on the spectral condition number of the tridiagonal matrix. The potential for $w^T v$ to vanish means that no finite tridiagonalization algorithm can be guaranteed to succeed. Even if the multipliers are all bounded below some modest value, say 10, there is the potential for catastrophic roundoff error.

On the other hand, this large growth has not been observed in practice using our algorithm. Instead, a gradual loss of similarity is observed as the matrix size increases. This degradation is conjectured to be caused by accumulated roundoff from using multipliers larger than one. Research continues into bounding the expected growth, and a future report will describe the results.

4. Recovery methods. In this section we describe the two recovery algorithms used in conjunction with the threshold pivoting algorithm. In most tridiagonalization schemes breakdown is defined as the situation where a multiplier (in our case an element in $N_r^{-1} N_c$) has exceeded some tolerance. When breakdown occurs, a number of options are available to circumvent the problem. Sometimes a local transformation can decrease the size of the multiplier so that the reduction can continue [11], but local methods cannot be robust because the tridiagonal form MAM^{-1} is unique once the first column and row of the transformation matrix M are fixed [13]. Thus, if this unique form has a small pivot, breakdown cannot be avoided without changing the first row or column. In [21], Wilkinson states that if a breakdown occurs, one can go back to the beginning and apply the transformation NAN^{-1} in the hope that breakdown will not occur again. No method of choosing N has been found that guarantees that the breakdown condition found in A will not exist in NAN^{-1}. For this reason, all proposed recovery methods choose another N and repeat the process if the previous choice of N fails to eliminate the breakdown condition.

The two recovery methods we propose differ in their choice of N, the amount of work they perform, and the matrix to which they are applied. In the first method, which is a variant of recovery methods proposed by Wachspress [8] and Watkins [19], a single random implicit single-shift LR iteration is applied to the matrix from the point of the last deflation down to row k. Since the partially reduced matrix is tridiagonal down to row k, one can start the iteration with either of the following forms of N:

$$\left(\begin{array}{cc|c} 1 & r & \\ 0 & 1 & O \\ \hline \multicolumn{2}{c|}{O} & I_{n-2} \end{array}\right) \quad \left(\begin{array}{cc|c} 1 & 0 & \\ r & 1 & O \\ \hline \multicolumn{2}{c|}{O} & I_{n-2} \end{array}\right).$$

Our first recovery method applies these two starting matrices alternately and uses a random value uniformly distributed on [0.1,1] for r. Figure 4 shows the FIXUP algorithm, which uses the left starting matrix above. Assuming no deflations have occurred, the first operations of the FIXUP algorithm introduce a nonzero in the a_{13} position. This "bulge" is then chased down the matrix with elementary similarity transformations to the point of the previous breakdown. Given that the breakdown occurs at row k, this chasing procedure fills in row $k-1$, which then must be annihilated to return the matrix to its prerecovery structure.

If breakdown occurs during the recovery or if the original breakdown condition persists after the recovery step, the recovery method is repeated with the alternate form of N. After three consecutive unsuccessful recovery attempts, the multiplier tolerance is temporarily increased by a factor of 10. After three additional unsuccessful attempts at this higher tolerance, the recovery attempts on the partially reduced matrix stop, and our second recovery method, NEWSTART, is initiated.

A small number of consecutive failures of the first recovery method is usually indicative of a matrix with a large number of small inner products. When this occurs, a random orthogonal matrix Q is applied to the original matrix, and the reduction is restarted with the modified matrix QAQ^T. The purpose of this operation is to reduce the probability of small inner products occurring in the modified matrix. The algorithm is simple and efficient to apply, requiring only $O(n^2)$ flops to execute, because Q is chosen to be a Householder transformation $Q = (I - 2ww^T)$. This routine, which we call NEWSTART, is initiated only as a last resort because it requires restarting the reduction.

5. Results. We report on empirical studies of three aspects of the new algorithm: its speed, robustness, and accuracy. All of the studies are based on finding the eigenvalues of nonsymmetric matrices, which is the primary use of the tridiagonalization algorithm. All computations were performed in double precision on a Sun 3/280.

To perform these studies we developed an algorithm, which we will refer to as TLR, for finding the eigenvalues of nonsymmetric tridiagonal matrices. TLR initially applies a diagonal similarity transformation to the tridiagonal matrix to scale the superdiagonal to contain all ones. Wilkinson [21] suggests this transformation because the superdiagonal is invariant under implicit LR iterations. Thus, the transformation saves space and floating point operations. In fact, the storage and flops per iteration are the same as for the symmetric tridiagonal case when using LR iteration. TLR applies implicit double shift LR iterations to the scaled tridiagonal matrix until all the eigenvalues are found. If the LR iteration breaks down due to encountering a small pivot element, (which can occur because pivoting is not performed) or it fails to converge to an eigenvalue after 30 iterations, then an arbitrary shift is applied to the matrix.

The potential dangers of using LR iteration are well documented [21], although some recent research [20] has attempted to put the algorithm on firmer theoretical ground. We chose LR iteration because it preserves nonsymmetric tridiagonal form. Our experience with TLR has been positive, as it has never failed to converge. On the other hand, we have seen tridiagonal matrices where the eigenvalues computed with TLR are not as accurate as the results from the standard EISPACK routines. For the interested reader, Dax and Kaniel [2] present more elaborate methods to improve the stability of the LR iteration and to refine the eigenvalues of the tridiagonal matrix iteratively.

A second alternative, used in [10], is to transform the real nonsymmetric tridiagonal matrix into a complex symmetric tridiagonal matrix. This is done by scaling the ith subdiagonal and superdiagonal entries to $\sqrt{b_i c_i}$, where b_i and c_i are opposing subdiagonal and superdiagonal entries, respectively. Then a complex arithmetic version of the QL iteration can be applied to finding all the eigenvalues. Details of the algorithm can be found in [1] along with a discussion of complex symmetric tridiagonal matrices and potential problems with finding their eigenvalues.

Table 1 compares the execution times in seconds of our algorithm with the EISPACK routines: ELMHES, ORTHES, and HQR for a series of test matrices ranging in size from 50 to 300. The matrices were random with entries distributed uniformly over the interval $[-1, 1]$. ELMHES reduces A to Hessenberg form using stabilized elementary similarity transformations while ORTHES reduces A to Hessenberg form H using Householder transformations. HQR finds the eigenvalues of H using an implicit double shift QR iteration. Our algorithms are presented in the table as A2TRI and TLR. A2TRI reduces A directly to tridiagonal form T as described in §3. TLR finds the eigenvalues of T. The time for either ELMHES or ORTHES should be added to the time for HQR and compared with the sum of the times for A2TRI and TLR.

TABLE 1
Execution times in seconds on a Sun 3/280 *for our new routines and the standard* EISPACK *routines.*

	EISPACK			NEW	
n	ELMHES	ORTHES	HQR	A2TRI	TLR
50	2.70	4.66	12.92	3.80	0.70
100	21.58	37.60	92.08	32.90	2.70
128	44.92	80.06	208.62	65.01	4.46
150	72.28	136.88	334.56	110.62	6.72
200	173.20	314.78	667.78	240.02	10.86
250	338.54	631.42	1388.36	509.94	16.94
300	582.64	1136.34	2305.14	893.48	23.84

It is clear from the table that our method can find the eigenvalues of a dense nonsymmetric matrix much faster than the EISPACK routines. A complexity analysis, where low order terms are ignored, shows that TLR requires $5n$ flops per iteration versus $4n^2$ flops for HQR. While the number of iterations varies between TLR and HQR, they both require only a few, usually fewer than 5, iterations per eigenvalue. Further, the arithmetic complexities of ELMHES and ORTHES are $(5/6)n^3$ and $(5/3)n^3$, respectively, while the complexity of A2TRI is $(4/3)n^3$ flops, assuming A2TRI needs to apply FIXUP only a constant number of times. The results in Table 1 reflect speedups greater than three for A2TRI/TLR over ELMHES/HQR, which are consistent with the relative complexities of the routines.

Random matrices are not necessarily good choices for testing the robustness of the tridiagonalization. Therefore, we input most of the nonsymmetric eigenvalue test matrices contained in the book by Gregory and Karney [9] as well as the EISPACK test suite of real general matrices into our algorithm. The test set included ill-conditioned, defective, and derogatory matrices in sizes up to 20×20. All were reduced successfully, although several required calls to the routine FIXUP. One matrix, Wilkinson's notoriously ill-conditioned matrix, required a call to the routine NEWSTART before it could be reduced. The eigenvalues calculated from the test matrices in Gregory and Karney were accurate to the expected number of digits given the condition numbers of the problems except for Wilkinson's matrix where only three digits of accuracy were obtained, instead of the expected eight digits. Table 2 gives the number of fixups executed by A2TRI and the largest relative error in any eigenvalue for each problem in the EISPACK test suite. Whenever the error is greater than 10^{-13}, the condition number of the corresponding eigenvalue is also given. In four cases, no error is made because A2TRI is able to permute the matrix into triangular form. In the two cases in which the relative error is greater than 1, the tridiagonal matrix returned by A2TRI is similar to the original matrix and all the error occurs in TLR. (This is not true in general.) There are also four cases where a significant amount of work is avoided because the problem deflates during the reduction to tridiagonal form. The apparent bad behavior in problem 2 is deceptive, because this problem has eigenvalues (λ_i) spread over six orders of magnitude. Let $\bar{\lambda_i}$ be the exact eigenvalues of A. Wilkinson [21] gives a bound on the absolute error of the eigenvalues as

$$|\lambda_i - \bar{\lambda_i}| \leq \frac{\| A \| \epsilon}{s_i},$$

where ϵ is machine precision and s_i is the inner product of the normalized left and right eigenvectors of λ_i. For problem 2 $\| A \| \approx 10^{10}$ and $s_i \approx 1$. Assuming $\epsilon = 10^{-16}$, the absolute error for the λ_i in problem 2 should be better than 10^{-6} and in fact we see an absolute error of 10^{-8}.

To determine the relative accuracy of the new algorithms, two comparisons were performed on a range of problem sizes. In the first comparison, the eigenvalues of the tridiagonal matrix were computed with HQR and compared with the corresponding eigenvalues of the original matrix as computed with ORTHES/HQR. This measures the loss of similarity caused by the reduction to tridiagonal form. The second comparison was between the eigenvalues computed by A2TRI/TLR and the eigenvalues computed by ORTHES/HQR. Figure 5 presents the accuracy seen during these comparisons.

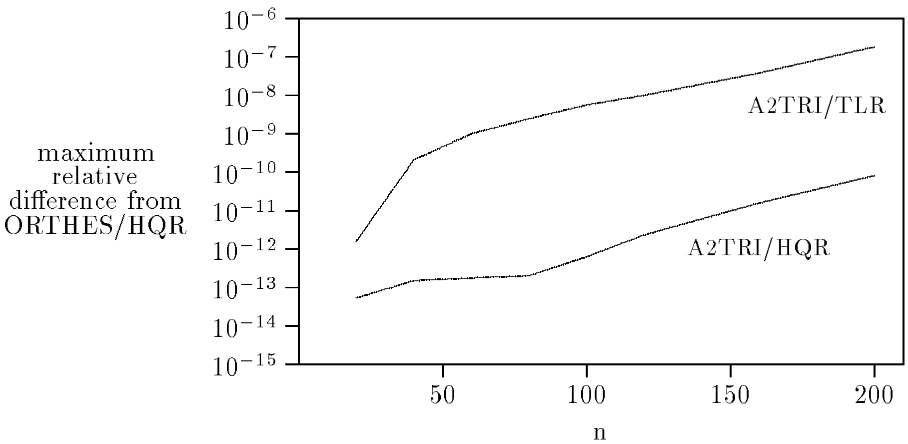

FIG. 5. *Typical degradation of eigenvalue accuracy seen with* A2TRI *and* TLR *for random orthogonal matrices.*

The graph clearly shows that the accuracy decreases as the matrix size increases. Similar results have been observed with other tridiagonalization methods [2], [11]. All calculations were performed in double precision. To avoid variations due to ill-conditioned eigenvalues, random orthogonal matrices were used in these tests. The

TABLE 2

Number of fixups, maximum relative error in any eigenvalue, and comments ($1/s_i$ *is the condition number of the corresponding eigenvalue*) *for* A2TRI/TLR *on the* EISPACK *test suite.*

EISPACK Test Suite of Real General Matrices			
Problem number	Number of fixups	Relative error in λ_i	Comments ($1/s_i$ =condition number of λ_i)
1	0	10^{-13}	
2	0	10^{-11}	$1/s_i = 1$
3	3	10^{-6}	$1/s_i = 10^7$
4	0	10^{-15}	
5	0	10^{-15}	
6	0	10^{-13}	
7	9	10^{-9}	$1/s_i = 10^{15}$
8	0	0	zero matrix
9	0	10^{-6}	$1/s_i = 10^{16}$
10	1	10^{-14}	
11	0	10^{-15}	
12	0	2.5	$1/s_i = 10^{15}$, Tridiagonal OK
13	2	10^{-14}	
14	2	10^{-15}	
15	0	10^{-15}	
16	0	0	permuted to triangular form
17	0	0	permuted to triangular form
18	0	0	permuted to triangular form
19	0	10^{-7}	$1/s_i = 10^{12}$
20	1	10^{-16}	
21	1	10^{-16}	
22	3	10^{-14}	
23	4	10^{-13}	
24	4	10^{-9}	$1/s_i = 10^{23}$
25	2	10^{-16}	
26	0	10^{-16}	deflated during reduction
27	0	10^{7}	$1/s_i = \infty$, Tridiagonal OK
28	0	10^{-16}	
29	1	10^{-16}	deflated during reduction
30	6	10^{-13}	
31	0	10^{-12}	$1/s_i = 10^{13}$
32	0	10^{-3}	$1/s_i = 10^{11}$
33	0	10^{-16}	deflated during reduction
34	0	10^{-16}	deflated during reduction
35	0	10^{-1}	$1/s_i = 10^{14}$

overall error of the new algorithms on random nonorthogonal matrices was comparable but showed a larger variation between problems.

The main advantage of a faster algorithm is the ability to solve larger problems, but the results of our study indicate that the accuracy of the larger problems may be poor. Methods exist for iteratively refining the accuracy of eigenvalues [4]. Presently, we are investigating an algorithm that improves the accuracy of the eigenvalues determined by TLR and avoids factorization of the original matrix by exploiting the already reduced tridiagonal form T. The algorithm differs from the iterative refinement in [2] in that the eigenvalues converge to the eigenvalues of the original matrix rather than those of T. Details of this work can be found in [3].

We have presented an algorithm for reducing a general matrix directly to tridiagonal form. Pivots are chosen that minimize the maximum element in the transformation matrices. We have described situations where the condition number of the transformation matrices can be large, and we have presented two recovery methods, which work well in practice when such situations arise. The new algorithm is fast and significantly broadens the class of matrices that can be successfully reduced.

Acknowledgments. The author would like to thank Gene Wachspress, Charles Romine, Beresford Parlett, and Gene Golub for helpful discussions during the course of this work. He would also like to thank the referees for their comments on earlier versions of this paper.

REFERENCES

[1] J. K. CULLUM AND R. A. WILLOUGHBY, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, Birkhäuser, Boston, 1985.

[2] A. DAX AND S. KANIEL, *The ELR method for computing the eigenvalues of a general matrix*, SIAM J. Numer. Anal., 18 (1981), pp. 597–605.

[3] J. J. DONGARRA, G. A. GEIST, AND C. H. ROMINE, *Computing the eigenvalues and eigenvectors of a general matrix by reduction to general tridiagonal form*, Tech. Report ORNL/TM-11669, Oak Ridge National Laboratory, Oak Ridge, TN, October 1990.

[4] J. J. DONGARRA, C. B. MOLER, AND J. H. WILKINSON, *Improving the accuracy of computed eigenvalues and eigenvectors*, SIAM J. Numer. Anal., 20 (1983), pp. 23–45.

[5] J. G. F. FRANCIS, *The QR transformation – Part* 2, Comput. J., 4 (1961), pp. 332–345.

[6] G. A. GEIST, *Reduction of a general matrix to tridiagonal form*, Tech. Report ORNL/TM-10991, Oak Ridge National Laboratory, Oak Ridge, TN, February 1989.

[7] ———, *Reduction of a general matrix to tridiagonal form using a hypercu be multiprocessor*, in Hypercube Concurrent Computers and Applications 1989, J. L. Gustafson, ed., Los Altos, CA, 1990, Golden Gate Enterprises, pp. 665–670.

[8] G. A. GEIST, A. LU, AND E. L. WACHSPRESS, *Stabilized Gaussian reduction of an arbitrary matrix to tridiagonal form*, Tech. Report ORNL/TM-11089, Oak Ridge National Laboratory, Oak Ridge, TN, February 1989.

[9] R. T. GREGORY AND D. L. KARNEY, *A Collection of Matrices for Testing Computational Algorithms*, Robert E. Krieger Publishing Company, Huntington, NY, 1978.

[10] R. G. GRIMES AND H. D. SIMON, *A new tridiagonalization algorithm for unsymmetric matrices*, Tech. Report SCA-TR-118, Boeing Computer Services, Seattle, WA, 1987.

[11] D. E. HARE AND W. P. TANG, *Toward a stable tridiagonalization algorithm for general matrices*, Tech. Report CS-89-03, Computer Science Dept., University of Waterloo, Waterloo, Ontario, Canada, January 1989.

[12] C. D. LABUDDE, *The reduction of an arbitrary real square matrix to tridiagona l form using similarity transformations*, Math. Comp., 17 (1963), pp. 443–447.

[13] B. N. PARLETT, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl. to appear.

[14] ———, *A note on LaBudde's algorithm*, Math. Comp., 19 (1964), pp. 505–506.

[15] H. RUTISHAUSER, *Solution of eigenvalue problems with the LR transformation*, Nat. Bur. Standards Appl. Math. Ser., 49 (1958), pp. 47–81.

[16] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garabow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines*-EISPACK *Guide*, Springer-Verlag, Heidelberg, 1974.

[17] C. Strachey and J. G. F. Francis, *The reduction of a matrix to codiagonal form by elimination*, Comput.J., 4 (1961), p. 168.

[18] E. L. Wachspress, *ADI solution of Lyapunov equations.* talk at Minnesota Supercomputer Institute Workshop on Practical Iterative Methods for Large-scale Computations, Minneapolis, MN, October 1988.

[19] D. Watkins, *Use of the LR algorithm to tridiagonalize a general matrix.* talk at Society for Industrial and Applied Mathematics Annual Meeting, Minneapolis, MN, July 1988.

[20] D. Watkins and L. Elsner, *Self-equivalent flows associated with the generalized eigenvalue problem*, Linear Algebra Appl., 118 (1989), pp. 107–127.

[21] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, U.K., 1965.