SIAM J. MATRIX ANAL. APPL.
Vol. 26, No. 3, pp. 878–900

QRT: A QR-BASED TRIDIAGONALIZATION ALGORITHM FOR NONSYMMETRIC MATRICES*

ROGER B. SIDJE† AND K. BURRAGE†

Abstract. The stable similarity reduction of a nonsymmetric square matrix to tridiagonal form has been a long-standing problem in numerical linear algebra. The biorthogonal Lanczos process is in principle a candidate method for this task, but in practice it is confined to sparse matrices and is restarted periodically because roundoff errors affect its three-term recurrence scheme and degrade the biorthogonality after a few steps. This adds to its vulnerability to serious breakdowns or near-breakdowns, the handling of which involves recovery strategies such as the look-ahead technique, which needs a careful implementation to produce a block-tridiagonal form with unpredictable block sizes. Other candidate methods, geared generally towards full matrices, rely on elementary similarity transformations that are prone to numerical instabilities. Such concomitant difficulties have hampered finding a satisfactory solution to the problem for either sparse or full matrices. This study focuses primarily on full matrices. After outlining earlier tridiagonalization algorithms from within a general framework, we present a new elimination technique combining orthogonal similarity transformations that are stable. We also discuss heuristics to circumvent breakdowns. Applications of this study include eigenvalue calculation and the approximation of matrix functions.

Key words. matrix reduction, nonsymmetric tridiagonalization, QR

AMS subject classifications. 15A23, 65F15, 65F25

DOI. 10.1137/040612476

1. Introduction.

The tridiagonalization of a general square matrix represents the most compact similarity reduction that can be computed directly. Attempting to reduce further (diagonalization or bidiagonalization) implies the retrieval of eigenvalues, which can only be done iteratively in general, unless the order of the matrix is under five. As yet, however, no *stable* tridiagonalization algorithm has been found. In fact we know of only one *finite*, though unstable and impractical, tridiagonalization algorithm due to George et al. [7].

While the biorthogonal Lanczos process is in principle a candidate tridiagonalization algorithm, it is generally confined to sparse matrices because rounding errors build up in its three-term recurrence scheme and degrade the biorthogonality after a few steps, making it necessary to restart periodically (though the restart may also be motivated by memory considerations in large-scale problems). The process is also vulnerable to serious breakdowns or near-breakdowns, the handling of which involves recovery strategies such as the look-ahead technique. But the look-ahead needs a careful implementation, and furthermore it produces a block-tridiagonal form with unpredictable block sizes.

Other candidate methods, geared generally towards full matrices, are not immune to serious breakdowns or near-breakdowns either, relying on elementary similarity transformations that are prone to numerical instabilities. Such concomitant difficulties have hampered finding a satisfactory solution to the problem for either sparse or full matrices. This study focuses primarily on nonsymmetric full matrices.

*Received by the editors July 29, 2004; accepted for publication (in revised form) by L. Reichel September 20, 2004; published electronically April 29, 2005.
http://www.siam.org/journals/simax/26-3/61247.html

†Department of Mathematics, Advanced Computational Modelling Centre, University of Queensland, Brisbane QLD 4072, Australia (rbs@maths.uq.edu.au, kb@maths.uq.edu.au).

Historically, interest in tridiagonalization stemmed primarily from its usefulness in reducing the cost of the LR algorithm, which predated the QR algorithm for computing the eigenvalues of a general dense matrix. However, the quest for tridiagonalization algorithms has been marred by numerical instabilities (see Wilkinson [19]). With a tridiagonalization $A = STS^{-1}$, computing an eigenpair (λ, y) of T gives the corresponding eigenpair (λ, Sy) of A. Therefore an added drawback is that eigenvectors can be contaminated when they are later retrieved by reapplying the transformations at the source of the inaccuracies.

The discovery of the QR algorithm proved very popular because it works really well, especially in conjunction with other enhancements for quick convergence (e.g., double shift) and accuracy (e.g., balancing). Tridiagonalization is unnecessary because the tridiagonal form is not preserved. The QR algorithm uses instead a preliminary orthogonal reduction to Hessenberg form for improved efficiency.

The inherent difficulties associated with tridiagonalization, together with the fact that the QR algorithm already works so well, nearly halted interest in finding stable algorithms for the reduction of a general matrix to strict tridiagonal form. But interest was rekindled by Dax and Kaniel [3], who reported that theoretical predictions are much more pessimistic than observed in practice (especially considering today's 64-bit computer architecture). Further investigation was then carried out by Geist [6], who added pivoting strategies and reported that instabilities arise at larger matrix sizes. Unfortunately, the likelihood of instabilities means that practical implementations have to anticipate them in order to remain robust and competitive [5, 6, 10, 12]. Although consolidation techniques bring some benefits, their added complexity discourages users, causing them to prefer the standard elegant QR approach with its renowned stable foundation, albeit at higher cost.

Our own interest in the problem stemmed from the computation of matrix functions [2, 15, 16]. Given a matrix A and a function f for which A is admissible (i.e., $f(A)$ is defined), the matrix function may be computed more economically as $f(A) = Sf(T)S^{-1}$, provided $A = STS^{-1}$ is a preliminary reduction to condensed form. The generic nature of the problem makes it compelling to have reduction algorithms that are useful in applications other than eigenvalue estimation.

We shall first outline a general framework for tridiagonalization algorithms. We subsequently present a new elimination technique combining orthogonal similarity transformations that are stable. We then discuss recovery techniques when serious breakdowns are encountered. We provide a roundoff error analysis. Finally, we present some numerical results and give some concluding remarks.

2. General principles of tridiagonalization.

2.1. Elementary similarity transformations. We use $\mathbb{R}$ throughout our presentation to emphasize that complex arithmetic is avoided for real data, but with minor adjustments the discussion applies to $\mathbb{C}$ as well. Let $x = (x_1, \ldots, x_n)^T$, $y = (y_1, \ldots, y_n)^T$ be vectors of $\mathbb{R}^n$, and consider the problem of finding an invertible matrix $M \in \mathbb{R}^{n\times n}$ such that

$$\left\{ \begin{array}{lcl} Mx & = & \alpha e_1, \\ y^T M^{-1} & = & \beta e_1^T, \end{array} \right. \tag{2.1}$$

where α and β are scalars to be determined and e_j is the jth column of the identity matrix of appropriate size.

LEMMA 2.1. *Assume $x_1 \neq 0$, $y_1 \neq 0$, and $y^T x \neq 0$. Then the matrices*

$$M_1 \equiv I - \frac{1}{x_1} x e_1^T + \frac{1}{y_1} e_1 y^T, \qquad M_2 \equiv I - \frac{1}{x_1} x e_1^T - \frac{1}{y_1} e_1 y^T$$

are solutions of (2.1), *and the following hold:*

1. $M_1 x = \frac{y^T x}{y_1} e_1$; $\quad M_2 x = -\frac{y^T x}{y_1} e_1$.
2. $y^T M_1^{-1} = y_1 e_1^T$; $\quad y^T M_2^{-1} = -y_1 e_1^T$.
3. $M_1^{-1} = I - e_1 e_1^T - \frac{1}{y^T x} x (y - 2 y_1 e_1)^T$; $\quad M_2^{-1} = I - e_1 e_1^T - \frac{1}{y^T x} x y^T$.
4. $\det M_1 = \frac{y^T x}{y_1 x_1}$; $\quad \det M_2 = -\det M_1$.

Proof. (1) can be verified by a straightforward multiplication; (2) and (3) follow from the Sherman–Morrison formula; finally, (4) follows from expanding the determinant. □

Remarks.

1. In the same spirit as other elementary transformations such as Householder, Gauss, or Gauss–Jordan, the transformation M zeroes a part of a vector. However, in contrast to those transformations, it has the special feature that its inverse M^{-1} is also a transformation targeted at a different vector.
2. Multiplying these transformations by a scalar, or more generally a diagonal matrix, preserves their basic effect. The conditions $x_1 \neq 0$ and $y_1 \neq 0$ ensure that M is defined. The condition $y^T x \neq 0$ ensures that M is invertible.
3. The inverse M^{-1} is in general a full matrix. This is, however, of limited consequence because M^{-1} is not used in isolation. What matters is its action. Notice also that if $z \in \mathbb{R}^n$, then Mz and $z^T M^{-1}$ are computed using a dot product and a gaxpy.
4. There are other matrices that satisfy Lemma 2.1. M_1 is the matrix that is often used in the literature (Geist [6] and Dongarra, Geist, and Romine [4]). It can be written as $M_1 = N_r N_c^{-1}$, where $N_r = I - \frac{1}{x_1} x e_1^T$ is the usual Gauss transformation for the row and $N_c = I - \frac{1}{y_1} e_1 y^T$ is that for the column. As we shall show in section 3, our new algorithm relies on another different type of matrix that is constructed with improved stability.

For convenience in the rest of this section we shall simply consider one case, say $M \equiv M_2$. The next illustration is a motivation for what follows. Let a matrix $A \in \mathbb{R}^{n\times n}$ be partitioned in the form

$$A = \begin{pmatrix} \delta & y^T \\ x & Z \end{pmatrix},$$

where x, $y \in \mathbb{R}^{n-1}$ for compatibility. If the assumptions of Lemma 2.1 are satisfied, a transformation M of order $n-1$ can be constructed such that

$$\begin{pmatrix} 1 & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} \delta & y^T \\ x & Z \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & M \end{pmatrix}^{-1} = \left(\begin{array}{c|cccc} \delta & \beta & 0 & \cdots & 0 \\ \hline \alpha & & & & \\ 0 & & & & \\ \vdots & & MZM^{-1} & & \\ 0 & & & & \end{array}\right).$$

The same process may be carried out on MZM^{-1} and so on. Upon termination, we end up with a tridiagonal matrix that is *similar* to the original matrix A. The method can also be used just to reduce the bandwidth of a matrix [12]. An overview of the overall procedure is outlined in the following section.

2.2. Nonsymmetric tridiagonalization. Starting with $A_0 \equiv A$ and applying the process above we obtain an updated matrix $A_{k-1} \equiv (a_{i,j}^{k-1})$ at stage $k-1$ whose pattern is

$$A_{k-1} = \left(\begin{array}{cccc|cccc} \delta_1 & \beta_1 & & & & & & \\ \alpha_1 & \delta_2 & \ddots & & & & 0 & \\ & \ddots & \ddots & \beta_{k-2} & & & & \\ & & \alpha_{k-2} & \delta_{k-1} & \beta_{k-1} & & & \\ \hline & & & \alpha_{k-1} & a_{k,k}^{k-1} & a_{k,k+1}^{k-1} & \cdots & a_{k,n}^{k-1} \\ & & & & a_{k+1,k}^{k-1} & a_{k+1,k+1}^{k-1} & \cdots & a_{k+1,n}^{k-1} \\ & 0 & & & \vdots & \vdots & \ddots & \vdots \\ & & & & a_{n,k}^{k-1} & a_{n,k+1}^{k-1} & \cdots & a_{n,n}^{k-1} \end{array}\right)$$
$$= \begin{pmatrix} T_{k-1} & 0 \\ 0 & A_{22}^{k-1} \end{pmatrix} + \alpha_{k-1} e_k e_{k-1}^T + \beta_{k-1} e_{k-1} e_k^T.$$

The kth transformation is then constructed as

$$M_k = \left(\begin{array}{c|cccc} I_k & & & 0 & \\ \hline & 1 & -m_{k+1,k+2} & \cdots & -m_{k+1,n} \\ & -m_{k+2,k+1} & 1 & & \\ 0 & \vdots & & \ddots & \\ & -m_{n,k+1} & & & 1 \end{array}\right)$$
$$= I - m_k e_{k+1}^T - e_{k+1} \widetilde{m}_k^T$$

where the multipliers are

$$(2.2) \begin{cases} m_k = (\underbrace{0, \ldots, 0}_{k+1}, m_{k+2,k+1}, \ldots, m_{n,k+1})^T, \ m_{i,k+1} = \frac{a_{i,k}^{k-1}}{a_{k+1,k}^{k-1}}, \ i = k+2, \ldots, n \\ \widetilde{m}_k = (\overbrace{0, \ldots, 0}, m_{k+1,k+2}, \ldots, m_{k+2,n})^T, \ m_{k+1,j} = \frac{a_{k,j}^{k-1}}{a_{k,k+1}^{k-1}}, \ j = k+2, \ldots, n. \end{cases}$$

The move from stage $k-1$ to stage k is described by

$$A_k = M_k A_{k-1} M_k^{-1}.$$

The method will be of practical value if this update can be done cheaply, especially without having to manipulate M^{-1} explicitly. A handy result to that effect is summarized below, which shows how to update based on $Zx/y^T x$.

LEMMA 2.2. *For the case $M \equiv M_2$, the ij-entries of $\widetilde{Z} = MZM^{-1}$ satisfy the following identities.*

Case $i = 1, j = 1$: $\widetilde{z}_{1,1} = \frac{y^T Z x}{y^T x}$.

Case $i = 1, j \neq 1$: $\widetilde{z}_{1,j} = \frac{1}{y_1}(y_j \frac{y^T Z x}{y^T x} - y^T Z e_j)$.

Case $i \neq 1, j = 1$: $\widetilde{z}_{i,1} = x_i \frac{e_1^T Z x}{x_1} - y_1 \frac{e_i^T Z x}{y^T x}$.

Case $i \neq 1, j \neq 1$: $\widetilde{z}_{i,j} = z_{i,j} - \frac{x_i}{x_1} z_{1,j} - y_j (\frac{e_i^T Z x}{y^T x} - \frac{x_i}{x_1} \frac{e_1^T Z x}{y^T x})$.

Proof. The identities come after expanding $\widetilde{z}_{i,j} = e_i^T MZM^{-1}e_j$. □

The method does not necessarily preserve the symmetry when the matrix is symmetric. This is of no importance since there are other techniques best suited for the symmetric case. The major concerns are rather that the algorithm may break down due to a zero pivot $a_{k,k+1}^{k-1}$ or $a_{k+1,k}^{k-1}$ in the multipliers (2.2), or it may have near-breakdowns due to small pivots that amplify the multipliers and introduce severe roundoff errors.

There is an intimate connection with the nonsymmetric (also known as biorthogonal) Lanczos process, a full description of which can be found in Golub and Van Loan [8, section 9.4.3]. The algorithm given above is equivalent to applying the biorthogonal Lanczos process to A with the starting vectors $u_1 = e_1$ and $v_1 = e_1$. In fact, any similarity tridiagonalization $A = STS^{-1}$ can always be understood as representing the biorthogonal Lanczos process with the starting vectors $u_1^T = e_1^T S^{-1}$ and $v_1 = Se_1$. In *exact* arithmetic, therefore, all tridiagonalization algorithms seeded with the same vectors are essentially equivalent, producing a tridiagonal matrix and a transformation matrix that are identical to within diagonal scaling. This is called the implicit-Q theorem, and its implications are described in detail by Parlett [14]. In particular, the desirable pairs (u_1, v_1) immune to breakdowns can be characterized in terms of Hankel determinants. It is therefore hard to tell in advance whether a pair is good, and, as with the Lanczos process, all tridiagonalization algorithms are susceptible to breakdowns.

Numerically, however, algorithms implemented in finite arithmetic may behave differently due to different stability properties. In the Lanczos process, for example, rounding errors build up rapidly in the three-term recurrence scheme and degrade the biorthogonality. It becomes unclear whether a breakdown or near-breakdown is genuine or the consequence of inaccurate intermediate computations. It is also possible for transformations of type M_1 and M_2 above to break down just because $x_1 = 0$ and/or $y_1 = 0$, or more likely they may have near-breakdowns at the neighborhood of these critical points, corrupting the ongoing tridiagonalization. Clearly, although the tridiagonalization is unique to within diagonal scaling once the starting vectors are prescribed, there can be numerical differences between algorithms, as is the case in other contexts such as in the QR decomposition which is unique but much different if computed via Modified Gram–Schmidt (MGS) or via the Classical Gram–Schmidt (CGS). Consider also QR vs. normal equations in least-squares problems or the myriad ways to get to the unique solution of a linear system. The pivoting strategy in the tridiagonalization algorithm of Geist [6] was motivated by such concerns. As we shall see later, our primary contribution is that each step of our new algorithm is nearly optimal in terms of minimizing rounding errors.

2.3. Related tridiagonalization algorithms. We will present our new algorithm in section 3. The framework that we just outlined in section 2 builds on previous works (see below). This framework bears a striking resemblance to an earlier work of Bauer [1], who showed that a class of solutions to Lemma 2.1 can be represented in the form $M = I - \tau uv^T$ and can therefore be understood as generalizations of Householder reflectors. (M_1 and M_2 above are rank-two additions and do not belong to this class unless $x = 0$ or $y = 0$.) Bauer [1] used, however, a loose terminology, defining a solution as "stable" if it exists in the neighborhood of $x_1 = 0$ and $y_1 = 0$ even though it is unstable when $y^T x \approx 0$. His transformations are also set in $\mathbb{C}^n$ and involve $\sqrt{y^T x}$ if need be, thus inducing complex arithmetic when $y^T x < 0$. Intriguingly, his work is not well publicized in the community and has gone unreferenced in other works. We thank the anonymous referee who brought it to our attention. We believe that these general principles provide a unified and coherent approach of describing tridiagonalization algorithms. One such algorithm was described by La Budde [11], unwittingly using precisely the generalized Householder reflectors of Bauer [1]. La Budde seemed to think that his algorithm was breakdown-free. But this was promptly refuted by Parlett [13] and Wang and Gregory [18]. We cite some of the other tridiagonalization algorithms here.

ELR: This was introduced by Strachey and Francis [17]. It can be very unstable and was abandoned soon after the discovery of the QR algorithm. It was revived owing to the analysis and empirical results of Dax and Kaniel [3]. The algorithm first uses the standard Hessenberg reduction of a general matrix and then uses elementary similarity transformations to zero the terms in the upper part. One of the main weaknesses of this work is that it did not include recovery strategies.

ATOTRI: Geist [6] added a pivoting strategy to the elimination procedure in an attempt to stabilize the transformations. This proved successful in many practical problems. However, since the similarity must be preserved, there is no guarantee that the multipliers on both the column and the row will be bounded by unity. As we indicated in remark 4 above, this approach amounts to using M_1 but with pivoting. That is, the elementary transformation is $\widetilde{M} = \widetilde{N}_r(\widetilde{N}_c)^{-1}$, where $\widetilde{N}_r$ and $\widetilde{N}_c$ use Px and Py with P being a permutation matrix. Several multipliers can remain unbounded should there be no permutation P that is simultaneously suitable for $\widetilde{N}_r$ and $\widetilde{N}_c$.

BHESS: This is one of the many attempts to improve stability by reducing to a banded, as opposed to a tridiagonal, matrix. It is a variant of the elimination algorithm with pivoting in which the least stable Gauss transformations are omitted [10]. The drawback is that it produces a "trapezoidal" matrix as a compromise, i.e., an upper-Hessenberg matrix with an unfinished tridiagonalization. Thus the onus is on subsequent computations to exploit its special structure.

3. The QR-based tridiagonalization algorithm (QRT).

We now describe our new tridiagonalization algorithm. It involves the following core ingredients. At each stage, it first uses a stable orthogonal similarity transformation to reduce both the column and the row. This reduces the column fully but leaves one trailing element on the row. The algorithm then finds another similarity transformation to eliminate that element. In the event of a serious breakdown, the algorithm restarts in an attempt to bypass the critical point.

3.1. The algorithm.

To describe the technical aspects of the algorithm in detail, consider the partitioning

$$A = \begin{pmatrix} \delta & y^T \\ x & Z \end{pmatrix}, \tag{3.1}$$

and let

$$[x, y] = QR = Q \left(\begin{array}{cc} \alpha & \beta \\ 0 & \gamma \\ \hline 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{array}\right)$$

be the QR decomposition of the pair $[x, y]$. Now observe that

$$\begin{pmatrix} 1 & 0 \\ 0 & Q \end{pmatrix}^T \begin{pmatrix} \delta & y^T \\ x & Z \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & Q \end{pmatrix} = \left(\begin{array}{c|ccccc} \delta & \beta & \gamma & 0 & \cdots & 0 \\ \hline \alpha & & & & & \\ 0 & & & & & \\ \vdots & & & Q^TZQ & & \\ 0 & & & & & \end{array}\right).$$

We therefore have a configuration where the only element to be eliminated is γ. All the other unwanted elements have been eliminated in a stable manner by the Q factor. Although orthogonal transformations have been used by other tridiagonalization algorithms (e.g., for preliminary reduction to Hessenberg form), the two-sided approach that we have just illustrated above is new, and our algorithm is the first method based on this approach. This special feature forms the centerpiece of our contribution.

Now, provided γ is eliminated without significant loss of stability, we can anticipate that the overall algorithm will remain stable for many practical problems. If $|\gamma| \leq |\beta|$, it is sufficient to use an elementary similarity transformation, as we saw earlier. The following result gives a precise characterization of the elimination in advance. It also shows that it makes no difference whether we use $[x, y]$ or $[y, x]$ (i.e., the QL variant), but we shall see later that a particular choice can improve the stability of the next step.

LEMMA 3.1 (stability condition). *We have $\gamma/\beta = \tan_\theta(x, y)$, and so $|\gamma/\beta| \leq 1$ is equivalent to $|\theta| \leq \pi/4$, i.e.,*

$$|\cos_\theta(x, y)| = \frac{|x^T y|}{\|x\|_2 \|y\|_2} \geq \frac{\sqrt{2}}{2}. \tag{3.2}$$

Proof. The thin R factor in the QR decomposition of $[x, y]$ is

$$\begin{pmatrix} \alpha & \beta \\ 0 & \gamma \end{pmatrix} = \begin{pmatrix} \|x\|_2 & y^T x / \|x\|_2 \\ 0 & \|y - (y^T x / x^T x) x\|_2 \end{pmatrix}.$$

Evaluating the ratio γ/β gives the result. $\square$

While bounding the multiplier by unity assists safety, the process really depends on the condition number of the similarity transformation, as our roundoff error analysis will enlighten later. Hence for the case where $|\gamma| > |\beta| > 0$ it may still be possible to apply a Gauss transformation if $|\gamma/\beta|$ does not exceed some tolerance, as done by Dax and Kaniel [3] and Geist [6], who reported that doing so is not always as bad as it seems in practice. Ideally, we would like to only use an orthogonal similarity transformation, but this is not typically possible, and our transformation attempts to come as close as we can get to one. Since the elimination step of our algorithm has only a single element to deal with, it is worth looking at the impact of a small pivot in detail. Assume that $k - 1$ steps of the tridiagonalization process have been performed, and the kth orthogonal similarity transformation has just been applied to produce the following result:

$$
\begin{aligned}
A_k^Q &= Q_k^T A_{k-1} Q_k \\
&= \left(\begin{array}{cccc|ccc|ccc}
 & & & & 0 & 0 & 0 & & & \\
 & T_{k-1} & & & \vdots & \vdots & \vdots & & \mathbf{0} & \\
 & & & & 0 & 0 & 0 & & & \\
 & & & & \beta_{k-1} & 0 & 0 & & & \\
\hline
0 & \cdots & 0 & \alpha_{k-1} & \delta & \beta & \gamma & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & \alpha & p & s & g'_{k+3} & \cdots & g'_n \\
0 & \cdots & 0 & 0 & 0 & q & t & g_{k+3} & \cdots & g_n \\
\hline
 & & & & 0 & f'_{k+3} & f_{k+3} & & & \\
 & \mathbf{0} & & & \vdots & \vdots & \vdots & & H & \\
 & & & & 0 & f'_n & f_n & & &
\end{array}\right) \\
&= \left(\begin{array}{c|c|c}
T & X & 0 \\ \hline
Y & W & G \\ \hline
0 & F & H
\end{array}\right). \qquad (3.3)
\end{aligned}
$$

To zero γ, the Gauss elimination matrix is

$$
S_k = \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & I_{n-k-2} \end{pmatrix} \in \mathbb{R}^{n\times n}, \qquad S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \mu_k \\ 0 & 0 & 1 \end{pmatrix}, \quad \mu_k = \frac{\gamma}{\beta} \equiv \mu.
$$

Using (3.3) with the fact that $SY = Y$ and $XS^{-1} = X$, we get

$$
A_k^S = S_k A_k^Q S_k^{-1} = \left(\begin{array}{c|c|c}
T & X & 0 \\ \hline
Y & SWS^{-1} & SG \\ \hline
0 & FS^{-1} & H
\end{array}\right), \qquad (3.4)
$$

where

$$
\begin{aligned}
SWS^{-1} &= \begin{pmatrix} \delta & \beta & 0 \\ \alpha & \mu q + p & -\mu^2 q + \mu(t-p) + s \\ 0 & q & -\mu q + t \end{pmatrix}, \\
SG &= \begin{pmatrix} 0 & \cdots & 0 \\ \mu g_{k+3} + g'_{k+3} & \cdots & \mu g_n + g'_n \\ g_{k+3} & \cdots & g_n \end{pmatrix}, \\
FS^{-1} &= \begin{pmatrix} 0 & \cdots & 0 \\ f'_{k+3} & \cdots & f'_n \\ -\mu f'_{k+3} + f_{k+3} & \cdots & -\mu f'_n + f_n \end{pmatrix}^T.
\end{aligned}
$$

This leads to a stable elimination of γ when $|\mu| = |\gamma/\beta| \leq 1$. Unfortunately, it shows that the occurrence of a very large multiplier μ can affect a row in SG and a column in FS^{-1}. Therefore there is still a possibility of having roundoff errors that build up due to very small pivots.

In order to attempt to mitigate these difficulties, we now consider a general elimination procedure aimed at the case where $|\gamma| > |\beta| > 0$. Looking at (3.3), it can be seen that we have obtained a structure where the upcoming elimination step can be identified to the problem of reducing the smaller inner 3-by-3 block

$$
W = \begin{pmatrix} \delta & \beta & \gamma \\ \alpha & p & s \\ 0 & q & t \end{pmatrix}
$$

to tridiagonal form while using a transformation matrix that will preserve the existing zeros in the other surrounding blocks. It is not difficult to see that the corresponding transformation matrix for this must therefore have its first column and first row both equal to the first canonical basis vector (to within diagonal scaling). We can write the smaller tridiagonalization problem $\widetilde{S}W\widetilde{S}^{-1} = W'$ as

$$\left(\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & \xi_1 & \xi_2 \\ 0 & \xi_3 & \xi_4 \end{array}\right) \left(\begin{array}{c|cc} \delta & \beta & \gamma \\ \hline \alpha & p & s \\ 0 & q & t \end{array}\right) \left(\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & \xi_1 & \xi_2 \\ 0 & \xi_2 & \xi_3 \end{array}\right)^{-1} = \left(\begin{array}{c|cc} \delta & \beta' & 0 \\ \hline \alpha' & p' & s' \\ 0 & q' & t' \end{array}\right). \tag{3.5}$$

It is interesting to note that this problem does not have a solution if $\beta = 0$, just as the earlier Gauss elimination matrix was undefined in that case. This situation amounts precisely to the *serious breakdown* case in the nonsymmetric Lanczos algorithm. Indeed (3.5) is equivalent to applying the nonsymmetric Lanczos process to W with the starting vectors $u_1 = e_1$ and $v_1 = e_1$. As theory predicts [5, 7, 14], the first iteration can be taken only if we have nonzero Hankel determinants

$$\Delta_1 = u_1^T W^0 v_1 = 1 \neq 0 \quad \text{and} \quad \Delta_2 = \begin{vmatrix} u_1^T W^0 v_1 & u_1^T W^1 v_1 \\ u_1^T W^1 v_1 & u_1^T W^2 v_1 \end{vmatrix} = \alpha\beta \neq 0.$$

We will propose an heuristic for the breakdown later. Continuing for now with the premise that $|\gamma| > |\beta| > 0$, it is easily seen that a solution to (3.5) is given (to within diagonal scaling) by

$$\widetilde{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \tau & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad \tau = \frac{\beta}{\gamma} < 1.$$

All the quantities involved so far are computed in a stable manner. Letting

$$\widetilde{S}_k = \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & \widetilde{S} & 0 \\ 0 & 0 & I_{n-k-2} \end{pmatrix} \in \mathbb{R}^{n\times n} \tag{3.6}$$

and using (3.3) once more with the fact that $\widetilde{S}X = X$ and $Y\widetilde{S}^{-1} = Y$, we get

$$A_k^{\widetilde{S}} = \widetilde{S}_k A_k^Q \widetilde{S}_k^{-1} = \left(\begin{array}{c|c|c} T & Y & 0 \\ \hline X & \widetilde{S}W\widetilde{S}^{-1} & \widetilde{S}G \\ \hline 0 & F\widetilde{S}^{-1} & H \end{array}\right). \tag{3.7}$$

The appearance of $\widetilde{S}^{-1}$ involves a risky division by $\tau < 1$ in the computations. This may still introduce numerical difficulties, but the potential of growth is basically confined to $F\widetilde{S}^{-1}$, and we now show how to lessen its extent. Direct calculation gives

$$\widetilde{S}W\widetilde{S}^{-1} = \begin{pmatrix} \delta & \gamma & 0 \\ \tau\alpha & q/\tau + p & -q/\tau - p + \tau s + t \\ 0 & q/\tau & -q/\tau + t \end{pmatrix}, \tag{3.8}$$

$$\widetilde{S}G = \begin{pmatrix} 0 & \cdots & 0 \\ \tau g'_{k+3} + g_{k+3} & \cdots & \tau g'_n + g_n \\ g_{k+3} & \cdots & g_n \end{pmatrix}, \tag{3.9}$$

$$F\widetilde{S}^{-1} = \begin{pmatrix} 0 & \cdots & 0 \\ f'_{k+3}/\tau & \cdots & f'_n/\tau \\ -f'_{k+3}/\tau + f_{k+3} & \cdots & -f'_n/\tau + f_n \end{pmatrix}^T. \tag{3.10}$$

This suggests that the algorithm can remain reasonably robust if we can bound most of f'_i/τ, $i = k+3, \ldots, n$. To do so, consider the earlier partitioning (3.1), and let

$$[x, y, Zx] = \widetilde{Q} \begin{pmatrix} \alpha & \beta & \rho \\ 0 & \gamma & \sigma \\ 0 & 0 & \nu \\ \hline 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{pmatrix}$$

be the QR decomposition of the triplet $[x, y, Zx]$. Now observe that

$$\begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix}^T \begin{pmatrix} \delta & y^T \\ x & Z \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q} \end{pmatrix} = \begin{pmatrix} \delta & \beta & \gamma & 0 & \cdots & 0 \\ \alpha & p & \times & \times & \cdots & \times \\ 0 & q & \times & \times & \cdots & \times \\ 0 & r & \times & \times & \cdots & \times \\ 0 & 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \times & \times & \cdots & \times \end{pmatrix}$$

with $p = \rho/\alpha, q = \sigma/\alpha, r = \nu/\alpha$. Indeed this comes from the fact that $\widetilde{Q}^T Z \widetilde{Q} e_1 = \widetilde{Q}^T Zx/\|x\|_2$. Applying this process at step k, we will therefore obtain a structure of the form

$$(3.11) \quad A_k^{\widetilde{Q}} = \widetilde{Q}_k^T A_{k-1} \widetilde{Q}_k$$

$$= \left(\begin{array}{cccc|ccc|ccc} & & & & 0 & 0 & 0 & & & \\ & T_{k-1} & & & \vdots & \vdots & \vdots & & 0 & \\ & & & & 0 & 0 & 0 & & & \\ & & & & \beta_{k-1} & 0 & 0 & & & \\ \hline 0 & \cdots & 0 & \alpha_{k-1} & \delta & \beta & \gamma & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \alpha & p & s & g'_{k+3} & \cdots & g'_n \\ 0 & \cdots & 0 & 0 & 0 & q & t & g_{k+3} & \cdots & g_n \\ \hline & & & & 0 & r & f_{k+3} & & & \\ & 0 & & & 0 & 0 & . & & H & \\ & & & & \vdots & \vdots & \vdots & & & \\ & & & & 0 & 0 & f_n & & & \end{array}\right).$$

If $|r| \leq |q|$, the term r can further be annihilated in another safe preliminary step before zeroing γ. Overall, this reorganization shows that we can significantly restrict the side effects of a large multiplier when zeroing γ in the elimination stage (3.7). Regardless the size of the problem, there can be at most six vulnerable entries at any one time: p, q, r, s, t, f_{k+3}. These are the entries where q/τ or r/τ come into play. It becomes possible to monitor them before opting for a breakdown and a recovery method. The risky entries reduce to four if $|r| \leq |q|$ and the term r is annihilated. The downside of this approach is the extra cost of Zx, but this might be preferable to other alternatives such as restarting from scratch. Moreover, the zeros that have been introduced can be exploited at the next step in a production code. If large multipliers reappear, we can alternate between the row variant and the column variant by taking the QR decomposition of $[y, x, Z^T y]$ in an attempt to diffuse the side effects evenly.

However, the trade-off now is that it brings $\widetilde{S}^{-1}$ in the transformation matrix. When a precise distinction is necessary, we shall refer to the QR of $[x, y]$ (or $[y, x]$) as the xy-QR (or yx-QR) step and to the augmented QR of $[x, y, Zx]$ (or $[y, x, Z^T y]$) as the xyz-QR (or yxz-QR) step. A reference to the QR step means either of these cases. This provides a similar distinction as with the ijk forms of loop notation.

The term q/τ, which occurs in (3.8), and the term r/τ, which will now occur in (3.10), satisfy

$$\frac{q}{\tau} = \frac{\sigma}{\alpha}\frac{\gamma}{\beta}, \quad \frac{r}{\tau} = \frac{\nu}{\alpha}\frac{\gamma}{\beta},$$

and this shows that they depend on the common quantity $\omega = \gamma/\alpha\beta$. Different values arise if we iterate with the xy- or yx-QR step. Let ω_{xy} denote the value from using the xy-QR step and ω_{yx} that of the yx-QR step. A similar reasoning as in the proof of Lemma 3.1 shows that $\omega_{xy}^2/\omega_{yx}^2 = \|x\|_2^2/\|y\|_2^2$. Hence, of the pairs $[x, y]$ and $[y, x]$, the smallest ω comes from the pair where the first vector is of smaller norm. This is how we decide whether to take a xy- or yx-QR step in practice.

In general, the unified quantity $\omega = \gamma/\alpha\beta$ highlights the relative importance of the parameters of interest in a remarkable way. If $\gamma \approx 0$, the matrix obtained after the orthogonal similarity transformation (3.3) is already tridiagonal, and so the elimination step is unnecessary. If $\alpha \approx 0$, an invariant subspace has been found, and the process can still be continued (e.g., by pivoting to eliminate β safely, also known as *deflation*), but the user may actually prefer an early termination. If $\beta \approx 0$, and $\alpha\beta$ is still small compared to γ (i.e., $|\omega|^{-1} = |\alpha\beta/\gamma| \ll 1$), there is a serious breakdown needing a full recovery method, as we shall see later. It appears therefore that, when the algorithm does continue, it does so under favorable conditions. We can choose to avoid near-breakdowns to limit the risk of introducing severe roundoff errors.

Notice that the simpler Gauss elimination matrix is a particular case of this general procedure. We use the Gauss elimination directly when $|\gamma| \le |\beta|$, but it can be recovered here with the diagonal scaling $\mathrm{diag}(1, \frac{\gamma}{\beta}, 1)\widetilde{S}$. Also note that the transformation matrix of other solutions to (3.5) need not be necessarily triangular, though similar numerical issues arise.

For the sake of completeness, we mention another simpler but ad hoc measure which is reminiscent of diagonal scaling and thus comes with reservations. An implementation can scale (3.4) by μ^{-1} to avoid using μ, thereby preventing large numbers from being introduced as the tridiagonalization progresses. At the kth step, this scaling is summarized as

$$\mu_k^{-1} \cdots \mu_1^{-1} A_k = \mu_k^{-1} S_k Q_k^T (\cdots (\mu_1^{-1} S_1 Q_1^T A Q_1 S_1^{-1}) \cdots) Q_k S_k^{-1}.$$

Of course, the scaling factor is unity in those cases where the pivot is sufficiently large. After accumulating the scaling factors, applications can then scale back their end result when/if it is necessary to do so. A similar reasoning can be made with (3.7) using τ as scaling factors. In both cases, the danger is that not only entries of the working matrix can become quite small but that the cumulative effect of the large multipliers reappears again when unscaling the final result, suggesting that this way of doing so might not be trustworthy in general.

3.2. Breakdown and recovery. Focusing now on the more promising algorithm described earlier, it is worth noting that excessively small values of τ are often indicative of a serious breakdown requiring one to resort to recovery methods. Looking at τ alone can be too pessimistic, however. As our earlier analysis showed, the compound quantity $\omega = \gamma/\alpha\beta$ can provide valuable insight. There are cases where a so-called *happy breakdown* may arise as well. Such cases are detected if $\alpha \approx 0$ (case of invariant subspace) or $\gamma \approx 0$ (case when the elimination step is unnecessary). The case of a serious breakdown arises when $\beta \approx 0$ after the current xy-QR step, with additionally $|\omega|^{-1} = |\alpha\beta/\gamma| \ll 1$ so that it remains unsafe to use the augmented xyz-QR step, as discussed earlier. In the Lanczos algorithm, the look-ahead technique is a popular recovery strategy for such breakdowns. However, it introduces a block-tridiagonal structure with unpredictable block sizes.

To maintain the strict tridiagonal form, it is, unfortunately, necessary to restart. This is necessary because it is not generally possible to avoid breakdowns locally. Local attempts to avoid the division by zero destroy the existing tridiagonal form. (That is why the look-ahead is the other alternative.) Dealing with breakdowns remains one of the unsatisfactory aspects of tridiagonalization algorithms. Avenues are inhibited by the tight connection to Hankel determinants and the implicit-Q theorem, as we alluded to earlier. In [19], Wilkinson suggested restarting from scratch with NAN^{-1} for some N in the hope that failure will be avoided in the modified matrix. Similarly, one can use different starting vectors, as we now describe.

We presented the QRT algorithm using $u = e_1$ and $v = e_1$ as starting vectors. Other starting vectors can be used by just applying the algorithm to the augmented matrix

$$\begin{pmatrix} 0 & u^T \\ v & A \end{pmatrix}.$$

This is tridiagonalized by the QRT algorithm as

$$\begin{pmatrix} 0 & u^T \\ v & A \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & P \end{pmatrix}^{-1} \begin{pmatrix} 0 & (u^Tv/\|v\|_2)e_1^T \\ \|v\|_2 e_1 & T \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & P \end{pmatrix}$$

so that $A = P^{-1}TP$, with the first column of P^{-1} and the first row of P now

$$P^{-1}e_1 = \frac{v}{\|v\|_2}, \qquad e_1^T P = \frac{\|v\|_2}{u^Tv} u^T.$$

It is always possible to choose u and v that guarantee a termination of the algorithm [7]. In general, however, whether a choice is good is not known in advance. But an interesting aspect of the principle above is that it can also be used as a recovery method. Indeed, assuming a breakdown happens at the kth step and augmenting the unfinished tridiagonalization A_{k-1}, we get

$$\begin{pmatrix} 0 & u^T \\ v & A_{k-1} \end{pmatrix} = \left(\begin{array}{c|ccccc|cccc}
0 & u_1 & u_2 & \cdots & & u_{k-1} & u_k & u_{k+1} & \cdots & u_n \\ \hline
v_1 & \delta_1 & \beta_1 & & & & & & & \\
v_2 & \alpha_1 & \delta_2 & \ddots & & & & & 0 & \\
\vdots & & \ddots & & \ddots & \beta_{k-2} & & & & \\
v_{k-1} & & & & \alpha_{k-2} & \delta_{k-1} & \beta_{k-1} & & & \\ \hline
v_k & & & & & \alpha_{k-1} & a_{k,k}^{k-1} & a_{k,k+1}^{k-1} & \cdots & a_{k,n}^{k-1} \\
v_{k+1} & & & & & & a_{k+1,k}^{k-1} & a_{k+1,k+1}^{k-1} & \cdots & a_{k+1,n}^{k-1} \\
\vdots & & & 0 & & & \vdots & \vdots & \ddots & \vdots \\
v_n & & & & & & a_{n,k}^{k-1} & a_{n,k+1}^{k-1} & \cdots & a_{n,n}^{k-1}
\end{array}\right).$$

As we apply elimination steps to this augmented matrix, the existing tridiagonal form is, unfortunately, destroyed. But if we take $u = (1, r, 0, \ldots, 0)^T$ and $v = e_1$ or, alternatively, $u = e_1$ and $v = (1, r, 0, \ldots, 0)^T$, where r is a random number, we obtain what is sometimes termed "bulge chase" (see, e.g., Geist [6]). The advantage here comes from the fact that only one term has to be dealt with as the bulge is chased down to regain the tridiagonal form. This is a low-cost procedure taking $O(k)$ flops. However, in [14, section 13.3], Parlett warned that this simple recovery technique is not good enough. And indeed we observed in practice that it is not always effective in remedying breakdowns. We observed improvements when u and v had several nonzero terms, but this comes at extra cost. We could not draw from our extensive experiments a default number of nonzero terms suitable in all situations. We also noted that the bulge chase was excluded from the final Fortran code of Dongarra, Geist, and Romine [4], which was based on the work of Geist [6]. Consequently, our own implementation simply restarts by augmenting A with full vectors with components randomly chosen from a uniform distribution in the interval $(0, 1)$. We allowed only one restart in the experiments, but, as we noted before, repeatedly trying full vectors ultimately yields termination, although stability may suffer in the more difficult cases.

To detect breakdowns, we do not rely on the τ coefficients alone, as they are transient and can be too pessimistic. We instead rely on the condition number $\kappa_\infty(P) = \|P\|_\infty \|P^{-1}\|_\infty$ that can be updated incrementally from the computations. This allows us to account for the compound effects of near-breakdowns as well. In our experiments we took $\varepsilon_{\text{brk}} = 10^{-10}$ as the tolerance parameter for the breakdown; i.e., breakdown was assumed when the reciprocal of the condition number satisfies $1/\kappa_\infty(P) \le \varepsilon_{\text{brk}}$.

3.3. Roundoff error analysis. An error analysis of the elimination method in full was made by Dax and Kaniel [3]. Since the elimination step of our algorithm involves only a single element, we wish to carry out a comparative study. We leave aside the orthogonal similarity transformations. This is not much different from the approach in [3], which omitted the preliminary reduction to Hessenberg form since there are no numerical difficulties associated with orthogonal transformations. We assume the worst-case scenario of having used the augmented $\widetilde{Q}$ factor at every step. If we include roundoff errors in (3.7), the exact formulation of the kth elimination step becomes

$$A_k^{\widetilde{S}} = \widetilde{S}_k A_k^{\widetilde{Q}} \widetilde{S}_k^{-1} + \widetilde{E}_k \tag{3.12}$$

in which

$$\begin{aligned}
\widetilde{S}_k &= I + (\tau_k - 1) e_{k+1} e_{k+1}^T + e_{k+1} e_{k+2}^T, \\
\widetilde{S}_k^{-1} &= I + \left(\frac{1}{\tau_k} - 1 \right) e_{k+1} e_{k+1}^T - \frac{1}{\tau_k} e_{k+1} e_{k+2}^T, \\
\widetilde{E}_k &= \varepsilon_{pqr}^k e_{k+1}^T + \varepsilon_{stf}^k e_{k+2}^T + e_{k+1} (\varepsilon_{\alpha g}^k)^T.
\end{aligned}$$

$\widetilde{S}_k$ is the elimination matrix (3.6), and $\widetilde{E}_k$ denotes the error matrix with nonzero entries due to roundoff errors only on those positions affected by $\widetilde{S}_k$. We write $\widetilde{E}_k$ using three n-vectors for convenience. As the updating formulas (3.8)–(3.11) show, ε_{pqr}^k has only three nonzero components induced by the change of p, q, and r; ε_{stf}^k has only three nonzero components induced by the change of s, t, and f_{k+3}; and, finally,

$\varepsilon^k_{\alpha g}$ has $n-k-1$ nonzero components induced by the change of α, $g'_{k+3},\dots,g'_n$. Hence we have

$$e_i^T\varepsilon^k_{pqr} = 0 = e_i^T\varepsilon^k_{stf} \text{ if } i \notin \{k+1,k+2,k+3\}; \quad (\varepsilon^k_{\alpha g})^T e_j = 0 \text{ if } j \notin \{k,k+3,\dots,n\}.$$

Although all contributions are included in the analysis, $\varepsilon^k_{\alpha g}$ is in principle unessential since it is only induced by multiplicative terms with τ_k and by construction $\tau_k < 1$. From (3.12) the final computed result satisfies

$$A^{\widetilde{S}}_{n-2} = \widetilde{S}_{n-2}\cdots\widetilde{S}_1 A^{\widetilde{Q}}_1 \widetilde{S}_1^{-1}\cdots\widetilde{S}_{n-2}^{-1} - \widetilde{E},$$
$$\widetilde{E} = \widetilde{E}_{n-2} + \widetilde{S}_{n-2}\widetilde{E}_{n-3}\widetilde{S}_{n-2}^{-1} + \sum_{k=1}^{n-4}\widetilde{S}_{n-2}\cdots\widetilde{S}_{k+1}\widetilde{E}_k\widetilde{S}_{k+1}^{-1}\cdots\widetilde{S}_{n-2}^{-1}.$$

Owing to the special pattern of $\widetilde{E}_k$, we get

$$\widetilde{E} = \widetilde{E}_{n-2} + \widetilde{S}_{n-2}\widetilde{E}_{n-3}\widetilde{S}_{n-2}^{-1} + \sum_{k=1}^{n-4}(\widetilde{S}_{k+1} + (\tau_{k+2}-1)e_{k+3}e_{k+3}^T)\widetilde{E}_k\widetilde{S}_{k+1}^{-1}\cdots\widetilde{S}_{n-2}^{-1},$$

and using the fact that $\|\widetilde{S}_k\|_\infty = 1+|\tau_k| \le 2,\ \|\widetilde{S}_k^{-1}\|_\infty = 2/|\tau_k|$, we obtain

$$\begin{aligned}\|\widetilde{E}\|_\infty &\le \|\widetilde{E}_{n-2}\|_\infty + \sum_{k=1}^{n-3} 2\|\widetilde{E}_k\|_\infty\|\widetilde{S}_{k+1}^{-1}\|_\infty\cdots\|\widetilde{S}_{n-2}^{-1}\|_\infty \\ &\le 2(n-2)\max_{1\le k\le n-2}\left(\frac{2}{|\tau_k|}\right)^{n-k-2}\max_{1\le k\le n-2}\|\widetilde{E}_k\|_\infty. \end{aligned} \tag{3.13}$$

The theoretical upper bound is still pessimistic, however, and the usual trade-off between speed and accuracy appears. Placing a restrictive constraint on the multipliers (e.g., $\tau_k \approx 1$) implies a growth factor, as in Gaussian elimination with partial pivoting, but a potential risk here is that recovery techniques may be triggered more often than necessary. However, this can largely be offset by the payoff from using the tridiagonal representation depending of the application. For example, Geist [6] reported a 300-by-300 eigenvalue computation on a Sun 3/280 which took 2305.14 seconds for the Hessenberg HQR method and 23.84 seconds for the tridiagonal TLR method, i.e., a hundred-fold speedup.

Further analysis suggests that our method should in general be numerically preferable over other tridiagonalization methods. As stated in Golub and Van Loan [8, eq. (7.1.11)], any similarity transformation MZM^{-1} is susceptible to roundoff errors, roughly $\varepsilon\kappa_2(M)\|Z\|_2$, where ε is the machine precision and $\kappa_2(M) = \|M\|_2\|M^{-1}\|_2$ is the condition number. This heuristic bound implies that the safest transformation is that for which $\kappa_2(M)$ is minimum. Let M and N be transformation matrices that satisfy (2.1). There exists an invertible matrix X such that $N = XM$. It follows from (2.1) that $Xe_1 = e_1$ and $e_1^TX^{-1} = e_1^T$ (to within diagonal scaling). Take $M = M_{QRT}$ from our QRT algorithm. With the earlier notation it can be written as $M_{QRT} = SQ^T$, where $[x,y] = QR$ and $S = I + \mu e_1e_2^T$ or $S = I + (\tau-1)e_1e_1^T + e_1e_2^T = \mathrm{diag}(\mu^{-1},1,\dots,1)(I+\mu e_1e_2^T)$. Minimizing $\kappa_2(N)$ is equivalent to minimizing $\kappa_2(XSQ^T) = \kappa_2(XS)$ over all matrices X with $Xe_1 = e_1$ and $e_1^TX^{-1} = e_1^T$. Consider now the QR factorization given by $X = YU$, where Y is orthogonal and U is upper-triangular with $Ue_1 = e_1$ and $e_1^TU = e_1^T$. The problem becomes that of minimizing $\kappa_2(US)$ over all such U. The minimum is attained when $US = I$ or, more generally, when $US = \Pi$ with Π orthonormal. The case $US = I$ implies that $U = S^{-1}$, which is, however, inconsistent with the requirement that $e_1^T U = e_1^T$. Due to the particular structures of U and S, an effective choice has U close to the identity matrix. The case $US = \Pi$ implies that $\Pi^T US = I$, which amounts to the first case. We do not need an exact minimization of a heuristic bound. But this analysis hints at the near-optimality of our scheme with respect to minimizing roundoff errors.

The following example will illustrate the point. Let $x = (-1, 1, \dots, (-1)^n)^T$, $y = (1, 1, \dots, 1)^T$ of length n, with n odd to make $x^T y = 1$. Computing $M = M_2$ from Lemma 2.1 and S from the QRT scheme, we obtain

$$M = \begin{pmatrix} -1 & -1 & -1 & \cdots & -1 \\ 1 & 1 & 0 & \cdots & 0 \\ -1 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 & 0 \\ (-1)^n & 0 & \cdots & 0 & 1 \end{pmatrix}, \quad S = \begin{pmatrix} \frac{1}{\sqrt{n^2-1}} & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & \cdots & 0 & 1 & 0 \\ 0 & & \cdots & 0 & 0 & 1 \end{pmatrix}.$$

Hence M is stable in the Gauss sense since all its multipliers are bounded by unity. Lemma 3.1 stated the corresponding stability condition in the QRT context

$$|\cos_\theta(x, y)| = \frac{|x^T y|}{\|x\|_2 \|y\|_2} = \frac{1}{n} \not\geq \frac{\sqrt{2}}{2} = 0.707.$$

It would appear that the QRT step would not be stable in the Gauss sense, whereas methods from Lemma 2.1 would be. But for $n = 5$ we get

$$10 \approx \kappa_2(S) < \kappa_2(M) \approx 14,$$

and this shows that the QRT step is preferable, as far as the similarity transformation is concerned. Larger n gave similar observations with wider differences. For example $n = 101$ gave $\kappa_2(S) \approx 202$, whereas $\kappa_2(M) \approx 10^3$.

3.4. Pseudocode. We summarize the ideas discussed so far into a pseudocode that can be translated into a computer program. The tridiagonalization occurs in Algorithm 2 with Algorithm 1 being the driver.

ALGORITHM 1: Compute $[T, P, P_{\text{inv}}, rcond] = \boldsymbol{QRT}(A)$

$[T, P, P_{\text{inv}}, rcond] := \boldsymbol{QRTRI}(A)$;
{Attempt a recovery method if there is a breakdown}
if $rcond \leq \varepsilon_{\text{brk}}$ **then**
  Choose random u and v ;
  $[T, P, P_{\text{inv}}, rcond] := \boldsymbol{QRTRI}\left(\begin{pmatrix} 0 & u^T \\ v & A \end{pmatrix}\right)$;
  $T := T(2:n+1, 2:n+1)$;
  $P := P(2:n+1, 2:n+1)$;
  $P_{\text{inv}} := P_{\text{inv}}(2:n+1, 2:n+1)$;
endif

Note in the pseudocode that a quantity $\theta \approx 0$ if $|\theta| \leq \varepsilon_{\text{zero}}$. Our MATLAB implementation used the drop tolerance $\varepsilon_{\text{zero}} = 10^{-7}$. Note also that each iteration of the pseudocode begins by deciding whether to take a xy- or yx-QR step. Any subsequent action then uses the appropriate indices, depending on the step retained.

Details are omitted in the pseudocode for readability. When there is breakdown, the control is passed back to the driver routine to possibly initiate a recovery attempt. An implementation can choose to exit with the last good values before the breakdown in case the user wants them, albeit they represent a partial decomposition.

ALGORITHM 2: Compute $[T, P, P_{\text{inv}}, rcond] = \boldsymbol{QRTRI}(A)$

$P := I$; $P_{\text{inv}} := I$; $T := A$;
for $k := 1 : n-2$ **do**
  $x := T(k+1:n, k)$; $y := T(k, k+1:n)^T$;
  {Decide whether to use $[x, y]$ or $[y, x]$}
  if $\|x\|_2 \leq \|y\|_2$ **then**
    $[Q, R] := QR(x, y)$;
  else
    $[Q, R] := QR(y, x)$;
  endif
  $\alpha = R(1,1)$; $\beta = R(1,2)$; $\gamma = R(2,2)$;
  {Use the simple xy- or yx-QR step if no xyz- or yxz-QR step is needed}
  if $\alpha \approx 0$ **or** $\gamma \approx 0$ **or** $|\beta| \geq |\gamma|$ **then**
    $T := \begin{pmatrix} I_k & 0 \\ 0 & Q^T \end{pmatrix} T \begin{pmatrix} I_k & 0 \\ 0 & Q \end{pmatrix}$; $P := \begin{pmatrix} I_k & 0 \\ 0 & Q^T \end{pmatrix} P$; $P_{\text{inv}} := P_{\text{inv}} \begin{pmatrix} I_k & 0 \\ 0 & Q \end{pmatrix}$
  endif
  {Move on to the next step if no elimination is necessary}
  if $\gamma \approx 0$ **continue** ;
  {Deflation when we have an invariant subspace}
  if $\alpha \approx 0$ **then**
    • apply Gauss elimination with pivoting to eliminate γ or β in T
    • update the transformation matrix P and its inverse P_{inv}
    continue ;
  endif
  {Use the simple Gauss elimination if possible}
  if $|\beta| \geq |\gamma|$ **then**
    • apply Gauss elimination to eliminate γ in T
    • update the transformation matrix P and its inverse P_{inv}
    continue ;
  endif
  {Use the xyz- or yxz-QR elimination if possible}
  if $\beta \approx 0$ **then**
    {serious breakdown}
    set $rcond := 0$;
  else
    • apply the extended xyz- or yxz-QR step
    • eliminate the r term if possible in T — see the discussion following (3.11)
    • eliminate γ in T
    • update the transformation matrix P and its inverse P_{inv}
    • compute $rcond := 1/\|P\|_\infty \|P_{\text{inv}}\|_\infty$, the reciprocal of the condition number
  endif
  {Exit if there is a breakdown}
  if $rcond \leq \varepsilon_{\text{brk}}$ **return** ;
endfor

3.5. A breakdown-free variant. We outline here a modified variant useful in certain applications. This variant avoids serious breakdowns at the trade-off of not producing a strict tridiagonal form. Consequently, we call it the breakdown-free QRT (BFQRT). There are applications where a strict tridiagonal form (or a form with bandwidth fourth or more) is not essential. But having as many zeros as possible is key to efficiency because floating-point operations involving zero elements can be avoided. This can be seen, for example, in Nikolajsen [12], where skipping null elements in the Laguerre eigensolver resulted in a marked speedup over the QR algorithm.

The BFQRT variant consists of omitting the elimination steps that would normally trigger recovery attempts. A similar strategy is used in BHESS [10] and Nikolajsen [12]. However, their approach gives a "trapezoidal" matrix of increasing bandwidth, whereas our approach reduces the density further by retaining a tridiagonal matrix but with occasional rows on the upper part. These rows appear where the elimination steps have not been applied. The pseudocode for this looks similar to Algorithm 2, except that we use only the xy- or xyz-QR steps and do not alternate with the yx- or yxz-QR steps. Another difference is that if $\beta \approx 0$, we just move on to the next step. We also use the updated *rcond* merely to decide whether to revert to the last good values before proceeding with the next step. Below are examples of patterns that BFQRT may produce in a 7-by-7 case:

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & & & & \\ & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & & \\ & & & \times & \times & \times & \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{pmatrix}, \quad \begin{pmatrix} \times & \times & & & & & \\ \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & & & \\ & & \times & \times & \times & & \\ & & & \times & \times & \times & \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{pmatrix}.$$

Note that in practice it is not necessary to apply a xy-QR step on rows set to be filled again. We can use a simple Hessenberg step there and move on to the next iteration.

4. Numerical experiments. We report some numerical examples using an exploratory MATLAB implementation on a Sun4u Sparc Workstation. Given a matrix A, we apply the QRT algorithm to compute $T = PAP^{-1}$, where T is tridiagonal and P is the similarity transformation matrix.

We compare our method with the ATOTRI Fortran code of Dongarra, Geist, and Romine [4] and Geist [6]. To this end, we implemented a MEX interface to invoke the native Fortran code of ATOTRI from within MATLAB. The comparison is based therefore on their original Fortran implementation available in the TOMS directory at netlib.org.

In the first set of examples, we also use $[L, U] = \mathrm{lu}(A)$ in MATLAB to compute the LU decomposition with partial pivoting. We report $\|U\|_\infty$, which gives insight into the growth factor that would arise with the Gauss elimination procedure itself. The following statistics (as computed by MATLAB) are given to assist in the evaluation of the results:

n	order of the matrix A
X_{eig}	eigenvectors of A, as computed by $[X, D] = \mathrm{eig}(A)$ in MATLAB
$\|U\|_\infty$	L-∞ norm, indicator of the growth in the LU decomposition of A
$\kappa_2(P)$	condition number of the matrix P, $\kappa_2(P) = \|P\|_2 \|P^{-1}\|_2$

4.1. GFPP examples. Results are shown on Table 1. The matrices are generated using the function called `gfpp` in Higham's testsuite [9]. This function generates a matrix that has the effect of attaining the maximal growth factor in Gaussian elimination with partial pivoting. We use `gfpp(n, c)`, which sets all the multipliers to c and gives a growth factor $(1 + c)^{n-1}$.

TABLE 1
GFPP examples.

GFPP Problem	$\|A\|_2$	$\kappa_2(X_{\text{eig}})$	$\kappa_2(P)$	$\frac{\|A-P^{-1}TP\|_2}{\|A\|_2}$	$\|U\|_\infty$	$\frac{\|A-LU\|_2}{\|A\|_2}$
$n = 50$, $c = 0.3$	1.10E+01	3.42E+00	E+01	E−15	E+05	E−12
$n = 100$, $c = 0.3$	2.05E+01	4.74E+00	E+01	E−15	E+11	E−06
$n = 200$, $c = 0.3$	1.04E+07	1.06E+07	E+02	E−15	E+22	E−02

Although this problem clearly affects the LU algorithm as n increases, it is handled well by the tridiagonalization method. This supports the observation made by Dax and Kaniel [3] that tridiagonalization methods are not necessarily doomed to fail on practical problems. In the same spirit that LU can fail but is widely used nonetheless, cheap elimination methods can be tried first before resorting to other robust (but expensive) alternatives to compute eigenvalues.

4.2. EigTool examples. Results appear on Table 2 and Figure 1. Most of the matrices in the EigTool set [20] are notoriously pathological. They are specifically aimed at showcasing the importance of pseudospectra analysis, and so each eigensystem is very sensitive to small perturbations. We refer the reader to EigTool [20] for further details about these problems. The QRT tridiagonalization is successful for most cases but suffers from serious breakdowns in some cases. The column with label *err* gives an error exit status. A value of 0 means that the algorithm completed all the steps. A value in the form $(k_1)k_2$ means that the algorithm encountered a serious breakdown at the k_1th step and the conservative recovery technique discussed earlier in section 3.2 was applied. A null k_2 means that the recovery was successful. Otherwise it means that the recovery itself failed at the k_2th step. We allocated a similar column for ATOTRI, but as our analysis will show, its error exit status is not entirely reliable.

TABLE 2
EigTool examples.

			QRT			ATOTRI		
Problem	$\|A\|_2$	$\kappa_2(X_{\text{eig}})$	err	$\kappa_2(P)$	$\frac{\|A-P^{-1}TP\|_2}{\|A\|_2}$	err	$\kappa_2(P)$	$\frac{\|A-P^{-1}TP\|_2}{\|A\|_2}$
hatano 50x50	2.93E+00	E+07	0	1	0	0	1	0
demmel 50x50	3.19E+04	Inf	0	E+04	E−15	0	E+08	E−13
gallery3 3x3	8.18E+02	E+03	0	7.55	E−16	0	E+01	E−17
gallery5 5x5	1.01E+05	E+11	0	E+07	E−15	0	E+08	E−13
godunov 7x7	4.32E+03	E+14	0	E+03	E−14	0	E+04	E−15
convdiff 49x49	1.02E+04	E+12	0	E+04	E−15	0	E+04	E−14
chebspec 49x49	1.32E+03	E+13	0	E+02	E−13	0	E+03	E−13
kahan 50x50	5.76E+00	E+12	0	E+01	E−15	0	E+02	E−15
sparserandom 50x50	3.28E+00	E+01	0	E+04	E−11	0	E+04	E−11
random 50x50	1.95E+00	E+01	0	E+03	E−14	0	E+02	E−13
boeing 55x55	1.69E+07	E+06	(17)0	E+07	E−15	–	−−	−−
twisted 50x50	2.74E+00	E+05	(37)0	E+07	E−11	0	E+10	E−06
frank 50x50	6.73E+02	E+10	(7)0	E+07	E−12	0	E+11	E−02
grcar 50x50	3.23E+00	E+08	(15)0	E+08	E−09	0	E+14	E−02
companion 50x50	4.59E+64	E+63	(1)0	E+04	E−16	–	−−	−−
markov 55x55	1.18E+00	E+02	(26)0	E+04	E−10	0	E+10	E−03
randomtri 50x50	1.64E+00	E+18	(25)25	E+07	E−10	–	*E*+47	*E*+16
riffle 50x50	2.36E+00	E+44	(8)11	E+08	E−10	–	*E*+35	*E*+05

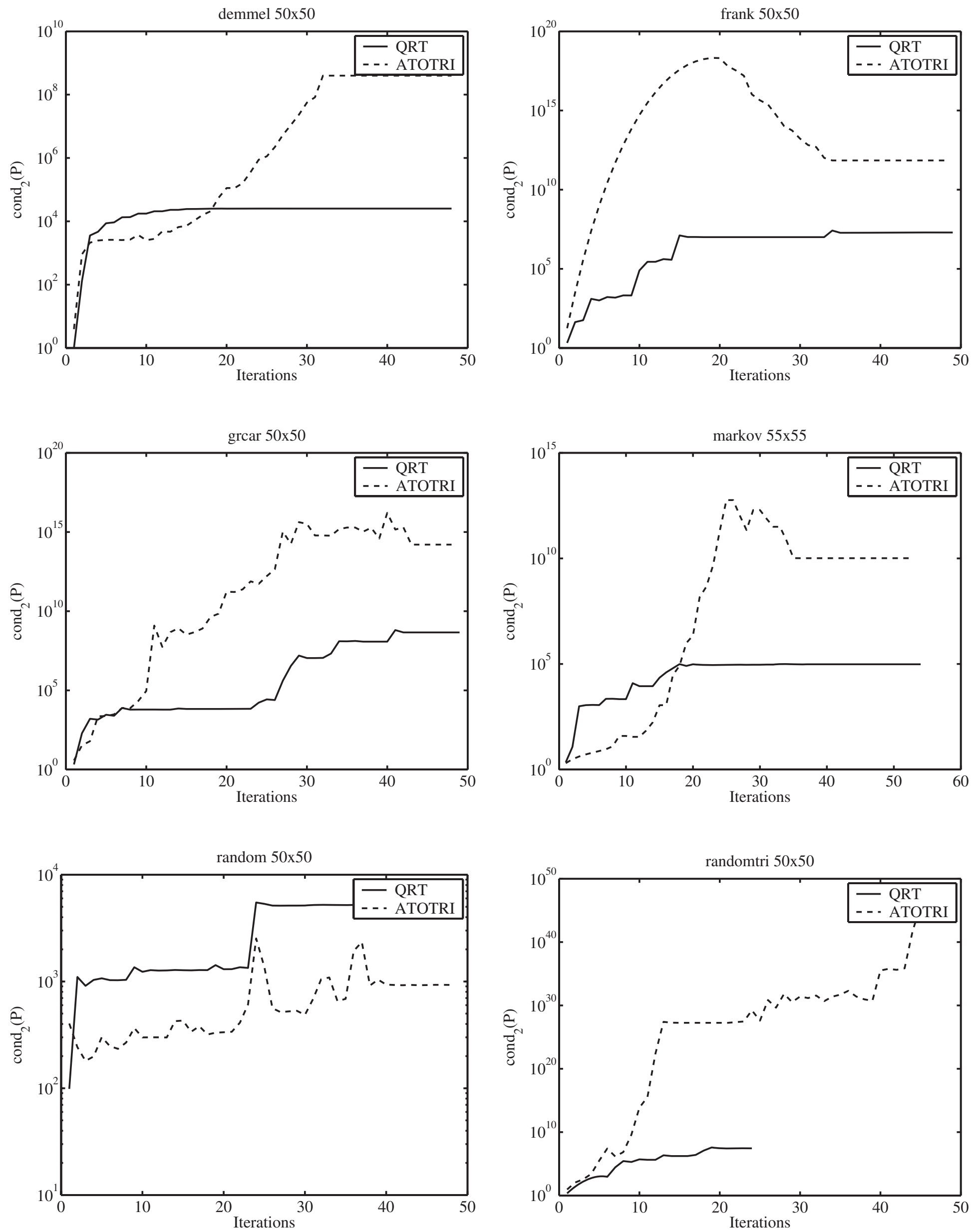

FIG. 1. *History of the condition number of the transformation matrix,* $\mathrm{cond}_2(P_k) = \|P_k\|_2\|P_k^{-1}\|_2$*, during the reduction of a few representative matrices from EigTool.*

Recall that $\frac{\|A-P^{-1}TP\|_2}{\|A\|_2}$ is bound to have rounding errors of order $\varepsilon\kappa_2(P)$, where ε is the machine precision, which is about 10^{-16} on the Sun4u Sparc Workstation where we conduct the experiments. We can make the following main observations:

- The *hatano* matrix is already tridiagonal and should be left untouched because the methods used $u = e_1$ and $v = e_1$ as default starting vectors. Thus this matrix served as an identity test for the codes.
- It is clear from the table that QRT is more accurate than ATOTRI in gen-

eral. The plots in Figure 1 depict the history of the condition number of the transformation matrix in various examples. There are occasional cases such as *random* where ATOTRI looks better. But even in those cases the relative error of QRT is as good as or better than ATOTRI, as seen on the main results on Table 2. In general, therefore, the transformation matrix produced by QRT tends to have the smallest condition number, leading to a smaller relative error. This agrees with the roundoff error analysis.

- The behavior of ATOTRI is disturbing in a number of pathological cases where its computed solution is seriously contaminated by roundoff errors, but the user is not given any warning. We use a dash ($-$) on Table 2 to draw the attention of the reader in those cases. The code actually returns an error exit status *err* of 0 that can mislead the user into thinking that the result is reliable when in fact there has been a total loss of accuracy. The *companion* example gave huge values. Another dramatic example was the *boeing* matrix that gave a transformation matrix for which the singular value decomposition to compute its condition number failed. Other examples are *riffle* and *randomtri*. As the history of *randomtri* in Figure 1 shows, QRT stopped at some point after reporting that its recovery attempt failed. But ATOTRI continued with meaningless data. Looking at the Fortran code of ATOTRI, we noted that it does not account for near-breakdowns. It detects the breakdown only if the inner product $x^T y = 0$. Other ramifications can be seen in the *frank* example: the condition number grows exaggeratedly before decaying, with the effect of corrupting the rest of the computations in a way not made apparent to the user. Such examples justify the careful attention for a more reliable breakdown criteria, as done by QRT.
- In the successful cases, a few problems (those for which *err* is in the form $(k_1)k_2$) needed recovery from breakdown. Recall that our recovery method consists of restarting with random vectors. Restarting was allowed only once.
- It can be seen that failure often arises because the conditioning of the eigensystem is simply too large compared with the norm of A. This is the case for the *randomtri* and *riffle* examples. A breakdown that happens very late in the tridiagonalization is suggestive of a critical choice of starting vectors. It is worth nothing that the results remain meaningful because they represent an unfinished tridiagonalization, which can still be useful, as the error bound shows. In those cases, it should be understood that $T = A_{k-1}$ for some k and is not really tridiagonal. See, for example, (3.11). Recall that the tridiagonalization is not an end in itself. When k is close to n, the remaining block can be reduced to Hessenberg form, and/or subsequent computations can take advantage of this nearly tridiagonal structure.

In other less pathological problems not reported here, QRT had a similar pattern of encouraging results. Overall, therefore, this algorithm was generally successful.

4.3. Eispack examples. We also applied our algorithm to matrices in the testsuite of Eispack. Results are displayed in Table 3 and Figure 2. This testsuite consists of 35 small matrices (none exceeding 20×20) that were thoughtfully designed to exercise the general purpose eigensolvers in Eispack. As in EigTool, the matrices are pathological with defective and/or derogatory cases. The examples do not appear as challenging as the EigTool examples, and we note that both algorithms were successful on all of the problems, and the accuracy remains very good. There are cases where recovery is needed at the very first step, suggesting that $u = e_1$ and $v = e_1$ are not

TABLE 3
Eispack examples.

			QRT			ATOTRI		
Problem	$\|A\|_2$	$\kappa_2(X_{\mathrm{eig}})$	err	$\kappa_2(P)$	$\frac{\|A-P^{-1}TP\|_2}{\|A\|_2}$	err	$\kappa_2(P)$	$\frac{\|A-P^{-1}TP\|_2}{\|A\|_2}$
1: 8x8	1.02E+03	1	0	1	E−15	0	2.05	E−16
2: 6x6	2.66E+09	5.73	0	1.69	E−16	0	3.28	E−16
3: 5x5	4.55E+01	E+08	0	E+02	E−16	0	E+02	E−15
4: 12x12	6.34E+01	1	0	1	E−16	0	1	E−16
5: 10x10	1.92E+08	E+02	0	E+02	E−16	0	E+02	E−15
6: 15x15	6.68E+06	E+01	0	E+01	E−15	0	E+01	E−15
7: 19x19	5.96E+05	E+01	0	E+03	E−14	0	E+03	E−13
8: 6x6	0	1	0	1	NaN	0	1	NaN
9: 6x6	5.58E+01	E+11	0	E+01	E−15	0	E+01	E−16
10: 6x6	1.67E+06	E+02	(2)0	E+01	E−15	(2)0	E+02	E−15
11: 5x5	2.41E+01	2.45	0	7.66	E−15	0	9.34	E−16
12: 5x5	1.93E+01	3.40	(1)0	6.32	E−16	(1)0	6.64	0
13: 5x5	1.93E+01	3.27	(1)0	E+01	E−14	(1)0	E+02	E−14
14: 5x5	2.07E+01	2.69	0	E+02	E−15	0	E+02	E−14
15: 5x5	2.07E+01	2.72	0	E+01	E−15	0	E+01	0
16: 3x3	1.80E+01	Inf	0	1	0	0	1	0
17: 3x3	1.07E+02	Inf	0	1	0	0	1	0
18: 3x3	1.06E+01	Inf	0	1	0	0	1	0
19: 4x4	1.26E+02	E+11	(1)0	2.49	E−17	(1)0	2.43	E−18
20: 3x3	1.00E+01	1	(1)0	5.17	E−15	(1)0	5.04	E−16
21: 4x4	1.00E+01	1	(1)0	2.75	E−15	(1)0	3.20	E−16
22: 5x5	1.00E+01	1	(1)0	E+01	E−14	(1)0	E+02	E−14
23: 6x6	1.00E+01	1	(1)0	E+01	E−14	(1)0	E+01	E−14
24: 8x8	1.00E+09	E+07	0	E+02	E−14	0	E+01	E−14
25: 4x4	7.01E+01	2.25	0	E+02	E−15	0	E+02	0
26: 3x3	7.12E+01	6.61	0	2.62	0	0	2.62	0
27: 4x4	4.35E+01	E+08	0	5.31	E−16	0	6.81	E−17
28: 4x4	1.23E+02	5.98	0	4.73	E−16	0	6.26	E−16
29: 6x6	7.28E+01	E+01	0	E+01	E−15	0	E+01	E−16
30: 6x6	1.93E+02	E+01	0	E+01	E−14	0	E+02	E−15
31: 8x8	2.37E+01	1.80	0	E+01	E−15	0	E+01	E−15
32: 4x4	1.78E+02	E+12	0	3.94	E−16	0	4.67	E−16
33: 6x6	1.46E+02	E+12	0	E+01	E−15	0	E+01	E−16
34: 8x8	3.12E+02	E+11	0	E+02	E−14	0	E+03	E−14
35: 10x10	1.08E+02	E+10	0	E+03	E−13	(5)0	E+02	E−15

suitable as default starting vectors there. Notice that the matrix in problem 8 is zero and that this is why the relative error is *NaN* (Not-a-Number).

5. Conclusion. We have described a promising algorithm for the tridiagonalization of nonsymmetric matrices. The algorithm primarily involves two stable Householder transformations per step and is twice as expensive as the symmetric tridiagonalization. The robust QR step provides a solid foundation to the proposed algorithm. There is still the possibility of suffering from the effect of a large multiplier, but we showed how to restrict risky roundoff errors to at most six entries irrespective of the size of the matrix. This suggests that the algorithm may be of assistance in a wide class of practical problems where a preliminary tridiagonalization is useful. Recovery techniques were discussed in the case where a serious breakdown happens or when a small pivot is rejected. A breakdown-free variant was described with the trade-off of not producing a strict tridiagonal form. Largely successful numerical experiments were conducted using a conservative restarting criteria to ascertain the robustness of the method. A comparison was made with a previous tridiagonalization algorithm of

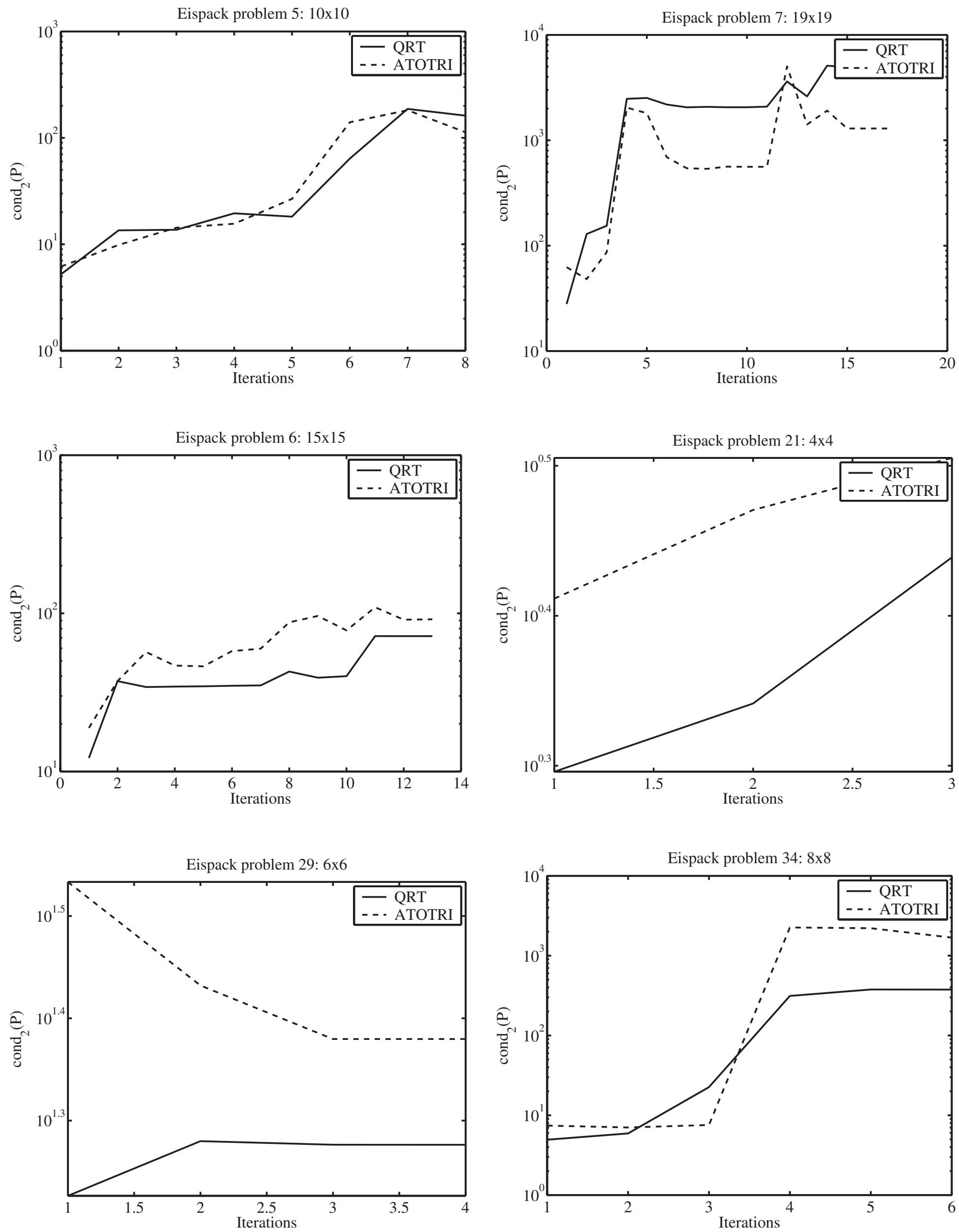

FIG. 2. *History of the condition number of the transformation matrix,* $\mathrm{cond}_2(P_k) = \|P_k\|_2\|P_k^{-1}\|_2$*, during the reduction of a few representative matrices from Eispack.*

Dongarra, Geist, and Romine [4] and Geist [6], and it shows that our algorithm is generally more robust and reliable. A roundoff error analysis suggests that our method should in general be numerically preferable over other tridiagonalization methods because it is nearly optimal in minimizing roundoff errors.

Acknowledgment. We would like to thank Prof. Nick Trefethen for his comments on drafts of this paper.

REFERENCES

[1] F. L. Bauer, *Sequential reduction to tridiagonal form*, J. Soc. Indust. Appl. Math., 7 (1959), pp. 107–113.

[2] P. I. Davies and N. J. Higham, *A Schur–Parlett algorithm for computing matrix functions*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 464–485.

[3] 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.

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

[5] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.

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

[7] A. George, K. Ikramov, A. N. Krivoshapova, and W.-P. Tang, *A finite procedure for the tridiagonalization of a general matrix*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 377–387.

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

[9] N. J. Higham, *The Test Matrix Toolbox for MATLAB (Version* 3.0*)*, Numerical Analysis Report 276, Department of Mathematics, University of Manchester, Manchester, UK, 1995.

[10] G. W. Howell, *Efficient computation of eigenvalues of randomly generated matrices*, Appl. Math. Comput., 66 (1994), pp. 9–24.

[11] C. D. La Budde, *The reduction of an arbitrary real square matrix to tridiagonal form using similarity transformations*, Math. Comp., 17 (1963), pp. 433–437.

[12] J. L. Nikolajsen, *An improved Laguerre eigensolver for unsymmetric matrices*, SIAM J. Sci. Comp., 22 (2000), pp. 822–834.

[13] B. N. Parlett, *A note on La Budde's algorithm*, Math. Comp., 18 (1964), pp. 505–506.

[14] B. N. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 567–593.

[15] R. B. Sidje, *EXPOKIT. A software package for computing matrix exponentials*, ACM Trans. Math. Software, 24 (1998), pp. 130–156.

[16] R. B. Sidje, K. Burrage, and B. Philippe, *An augmented Lanczos algorithm for the efficient computation of a dot-product of a function of a large sparse symmetric matrix*, in Proceedings of the International Conference on Computational Science, Lecture Notes in Comput. Sci. 2659, P. M. A. Sloot et al. eds., Springer-Verlag, Berlin, 2003, pp. 693–704.

[17] C. Strachey and J. G. F. Francis, *The reduction of a matrix to codiagonal form by eliminations*, Comput. J., 4 (1961), pp. 168–176.

[18] H. H. Wang and R. T. Gregory, *On the reduction of an arbitrary real square matrix to tridiagonal form*, Math. Comp., 18 (1964), pp. 501–505.

[19] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.

[20] T. G. Wright, *EigTool Software Package*; http://web.comlab.ox.ac.uk/pseudospectra/eigtool/.