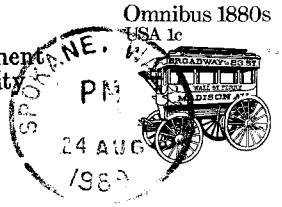
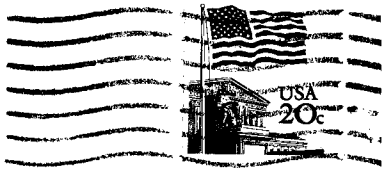
Dr. Alan Genz
Computer Science Department
Washington State University
Pullman, WA 99164-1210

Omnibus 1880s
USA 1c





AIR MAIL

Computer Science Dept.
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada

Attn: Technical Report Secretary

Dr. Alan Genz
Computer Science Department
Washington State University
Pullman, WA 99164-1210

Dear

I would greatly appreciate receiving a reprint of your article
"Toward a stable tridiagonalization algorithm for general matrices": RR CS-89-03

Thank you very much.
sent Aug. 30/89
Sincerely,

A. Genz

82101

Printing Requisition / Graphic Services

1. **Please complete unshaded areas** on form as applicable.
2. **Distribute copies** as follows: White and Yellow to Graphic Services. **Retain Pink Copies for your records.**
3. On completion of order the **Yellow copy will be returned** with the printed material.
4. **Please direct enquiries, quoting requisition number and account number,** to **extension 3451.**

TITLE OR DESCRIPTION: CS-89-03

DATE REQUISITIONED: Sept. 14 | DATE REQUIRED: ASAP | ACCOUNT NO.: 1/12/16/62841.1

REQUISITIONER - PRINT: M.P. Tang | PHONE: 4468 | SIGNING AUTHORITY: [signature]

MAILING INFO - NAME: S. DEANGELIS | DEPT.: C.S. | BLDG. & ROOM NO.: DC 2314 | ☑ DELIVER ☐ PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES: 24 | NUMBER OF COPIES: 20

TYPE OF PAPER STOCK: ☑ BOND ☐ NCR ____ PT. ☑ COVER ☐ BRISTOL ☑ SUPPLIED ☐ ____

PAPER SIZE: ☑ 8½ x 11 ☐ 8½ x 14 ☐ 11 x 17 ☐ ____

PAPER COLOUR: ☑ WHITE ☐ ____ | INK: ☑ BLACK ☐ ____

PRINTING: ☐ 1 SIDE ____ PGS. ☑ 2 SIDES ____ PGS. | NUMBERING: FROM ____ TO ____

BINDING/FINISHING: 3 down - left side ☑ COLLATING ☑ STAPLING ☐ ____ HOLE PUNCHED ☐ PLASTIC RING

FOLDING/PADDING | CUTTING SIZE

Special Instructions

Math fronts + backs enclosed.

COPY CENTRE — OPER. NO. | BLDG. | MACH. NO.

DESIGN & PASTE-UP — OPER. NO. | TIME | LABOUR CODE

			D01
			D01
			D01

TYPESETTING — QUANTITY

PAP000000					T01
PAP000000					T01
PAP000000					T01

PROOF

PRF					
PRF					
PRF					

NEGATIVES — QUANTITY | OPER. NO. | TIME | LABOUR CODE

FLM					C01
FLM					C01
FLM					C01
FLM					C01
FLM					C01

PMT

PMT					C01
PMT					C01
PMT					C01

PLATES

PLT					P01
PLT					P01
PLT					P01

STOCK

					001
					001
					001
					001

BINDERY

RNG					B01
RNG					B01
RNG					B01
MIS000000					B01

OUTSIDE SERVICES

$____ COST

TAXES - PROVINCIAL ☐ FEDERAL ☐

To Sue de Angelis From Linda. Stone.

Date 25 Jan '89.

memo

University of Waterloo

Re:

"Toward a Stable Tridiagonalization Algorithm for General Matrices"

by.

D.E. Han & W-P Tang.

Could you please leave.

4 copies in my mailbox.

Thanks. gave to Linda Jan 26/89 Linda →

Also:

6 copies of

"Relief from the Pain of Overlap."

Thanks.

Printing Requisition / Graphic Services

26012

1. **Please complete unshaded areas** on form as applicable.
2. **Distribute copies** as follows: White and Yellow to Graphic Services. **Retain Pink Copies for your records.**
3. On completion of order the **Yellow copy will be returned** with the printed material.
4. **Please direct enquiries, quoting requisition number and account number,** to extension 3451.

TITLE OR DESCRIPTION

Toward a Stable Tridiagonalization Algorithm for General Matrices CS-89-03

DATE REQUISITIONED	DATE REQUIRED	ACCOUNT NO.
Jan. 19/89	ASAP	1 1 2 6 6 2 8 4 4 1

REQUISITIONER – PRINT	PHONE	SIGNING AUTHORITY
W.P. Tang	4468	Sue DeAngelis / W.P. Tang

MAILING INFO – NAME	DEPT.	BLDG. & ROOM NO.	
Sue DeAngelis	C.S.	DC 2314	[X] DELIVER [ ] PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES 23 NUMBER OF COPIES 80

TYPE OF PAPER STOCK
[X] BOND [ ] NCR ____ PT. [X] COVER [ ] BRISTOL [X] SUPPLIED [ ] ____

PAPER SIZE
[X] 8½ x 11 [ ] 8½ x 14 [ ] 11 x 17 [ ] ____

PAPER COLOUR: [X] WHITE [ ] ____ INK: [X] BLACK [ ] ____

PRINTING: [ ] 1 SIDE ____ PGS. [X] 2 SIDES ____ PGS. NUMBERING FROM ____ TO ____

BINDING/FINISHING: 3 down left side
[X] COLLATING [X] STAPLING [ ] ____ PUNCHED [ ] PLASTIC RING

FOLDING/PADDING CUTTING SIZE

Special Instructions

Math fronts and backs enclosed

COPY CENTRE OPER. NO. | BLDG. | MACH. NO.

DESIGN & PASTE-UP OPER. NO. | TIME | LABOUR CODE
D 0 1
D 0 1
D 0 1

TYPESETTING QUANTITY
PAP 0 0 0 0 0 0 T 0 1
PAP 0 0 0 0 0 0 T 0 1
PAP 0 0 0 0 0 0 T 0 1

PROOF
PRF
PRF
PRF

NEGATIVES QUANTITY | OPER. NO. | TIME | LABOUR CODE
FLM C 0 1
FLM C 0 1
FLM C 0 1
FLM C 0 1
FLM C 0 1

PMT
PMT C 0 1
PMT C 0 1
PMT C 0 1

PLATES
PLT P 0 1
PLT P 0 1
PLT P 0 1

STOCK
0 0 1
0 0 1
0 0 1
0 0 1

BINDERY
RNG B 0 1
RNG B 0 1
RNG B 0 1
MIS 0 0 0 0 0 0 B 0 1

OUTSIDE SERVICES

$ ____ COST

TAXES – PROVINCIAL [ ] FEDERAL [ ]

GRAPHIC SERV. OCT. 85 482-2

January 17, 1989

Sue!

Here's the TR for Wei-Pai Tang. 80 copies
please using the Math covers front and back.

Thanks...

Toward a Stable Tridiagonalization Algorithm for General Matrices

by

D.E. Hare and W.-P. Tang

Faculty of Mathematics

University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Toward a Stable Tridiagonalization Algorithm for General Matrices

by

D.E. Hare and W.-P. Tang

Research Report CS-89-03
January 1989

Toward a Stable Tridiagonalization Algorithm for General Matrices

by

D. E. G. Hare[1] and W.-P. Tang[1]

Department of Computer Science
University of Waterloo
Waterloo, Canada

January 14, 1989

Abstract

The known tridiagonalization algorithms for general matrices suffer from serious breakdown and/or stability problems. In this paper, we present a new algorithm for reducing a general matrix to tridiagonal form by a sequence of similarity transformations. Stability is promoted by controlling the norms of the transformation matrices. Results of tests indicating the stability characteristics of the algorithm and comparison with the Lanczos algorithm are included.

---

[1] *Research partially supported by the Natural Sciences and Engineering Research Council of Canada and the University of Toronto/University of Waterloo Information Technology Research Centre*

AMS Subject Classification (1980): 65F30, 65F15

1. Introduction

For the problem of determining the eigenvalues of a general matrix, the QR algorithm remains the popular choice. To use the QR algorithm efficiently, the matrix is first reduced to Hessenberg form, an $O(n^3)$ operation. The QR algorithm then iteratively transforms the Hessenberg matrix to obtain the eigenvalues. As each iteration requires $O(n^2)$ work, in the general case, there is much interest in the possibility of further reducing the Hessenberg matrix to a condensed form which allows for an $O(n)$-work-per-step iterative method to be applied to find the eigenvalues.

The condensed form which has been attracting the most attention is the tridiagonal form. However, the known algorithms for tridiagonalizing a nonsymmetric matrix suffer from serious breakdown and/or stability problems: all of the algorithms require division by computed quantities, so the occurrence of a near-zero denominator effectively halts the algorithms (*breakdown*), while the occurrence of only relatively small denominators can significantly reduce the accuracy of the computed tridiagonal form (*stability problems*).

There are two main classes of algorithms for the tridiagonalization problem. The first is the class of Lanczos-type algorithms. These algorithms have the important feature of producing a tridiagonal matrix similar to the original matrix without modifying the original matrix. Thus, Lanczos algorithms are well suited to sparse problems. The stability problem for these algorithms is severe, however, making a practial algorithm difficult to achieve. Some pivoting techniques designed to reduce the chance of breakdown were considered in [5, 6 and 11] and some incomplete orthogonalization methods were tried in [10]. Variations of the Lanczos algorithm for certain special classes of matrices were studied in [2] and [7].

The second class of tridiagonalization algorithms consists of algorithms which use a sequence of similarity transformations to successively introduce zeros into the matrix being transformed (somewhat in the style of Gaussian elimination). As sparsity is quickly destroyed by such algorithms, these methods are not suited to sparse eigenvalue problems. These methods can also suffer from breakdown, with the appearance of a very small pivot, and from stability problems.

As the algorithm we propose in this paper belongs to the second class, we will restrict our discussion to that class of methods for the remainder of this section.

In [9], the last named author proposed an algorithm for recovering from breakdown. With that algorithm, breakdown occurring in the first half of the reduction

could be recovered from, while breakdown occurring in the remaining steps could be recovered from only at the cost of leaving non-zeros in the last column (thus, the reduced form of the matrix would be *bordered tridiagonal*). Numerical experiments indicate, however, that the algorithm still suffers stability problems when applied to large matrices. Thus, the breakdown problem has been (mathematically) solved, but the stability problem remains.

Breakdown itself, that is, the occurrence of a pivot which is near zero in absolute terms, is very rare. It is also not the major stumbling block in the way of a practical implementation of a tridiagonalization algorithm. Rather, as Wilkinson showed in [12], *the main source of numerical inaccuracy is the occurrence of large multipliers*, that is, of pivots which are small in relative terms.

In this paper, we present an algorithm which *guarantees* a small upper bound on the size of the multipliers (equivalently, on the norms of the transformation matrices), at the cost of some heuristic adjustments. These heuristics are not perfect, but numerical testing indicates a sufficiently high success rate to make this a practical algorithm. In addition to this bound on the larger multipliers, the steps of the reduction are so arranged that most of the multipliers are no more than 1 in absolute value.

Backward error analysis indicates a bound similar to that of Gaussian elimination with partial pivoting. However, this bound is, in our experience, overly pessimistic, as our initial numerical testing indicates that the algorithm has good stability behaviour, particularly in comparison with the Lanczos algorithm.

We begin the description of the algorithm in Section 2, with the case when the multipliers remain uniformly small. In this case, the algorithm is deterministic, and the analysis is straightforward. In Section 3 we describe how the algorithm deals with the occurrence of a large multiplier. In Section 4, we discuss some of the quantitative aspects of the algorithm's heuristics and present the results of our numerical experiments. The paper concludes with some possible alternative strategies for multiplier control, which we will explore further in a future paper.

2. The basic algorithm

In this section we describe the reduction of a general matrix to tridiagonal form under the assumption that no breakdown or large multiplier occurs. The description of how the algorithm deals with multipliers deemed unacceptably large is deferred to Section 3.

We first briefly describe the Lanczos algorithm, as the basic algorithm we are

presenting in this section is mathematically equivalent to it. Let A be an arbitrary $n \times n$ real matrix and let $p, q \in \mathbf{R}^n$ be arbitrary. Form the *Krylov vectors* $\{A^k q\}_{k \geq 0}$ and $\{p^t A^k\}_{k \geq 0}$. These vectors span subspaces of $\mathbf{R}^n$ (formally, the latter is a subspace of the dual of $\mathbf{R}^n$). Applying the two sided Gram-Schmidt process to these vectors and making some useful observations, we arrive at a three term recurrence relation, which, when $k = n$ represents a similarity transformation of the matrix A to tridiagonal form.

The three term recurrence relation produces a sequence of vectors which can be viewed as forming the rows and columns, respectively, of rectangular matrices, P_k and Q_k, such that after n steps, P_n and Q_n are $n \times n$, $Q_n = P_n^{-1}$ and $P_n A Q_n$ is tridiagonal. At each step, an orthogonalization is performed, which requires a division by the inner product of (multiples of) the vectors produced at the previous step. Breakdown thus occurs if any of these inner products is 0 (see [10]). Numerically, the algorithm yields highly suspect results if any of the inner products is small relative to the corresponding numerators.

It is known [4] that vectors p and q exist so that the Lanczos algorithm applied with these as starting vectors does not encounter breakdown. However, determining these vectors requires knowledge of the minimal polynomial of A, which essentially puts the cart before the horse. Further, there are no theoretical results showing that p and q can be chosen so as to avoid *small* inner products. Thus, no algorithm for successfully choosing p and q at the start of the computation yet exists.

The algorithm we propose here differs considerably from the Lanczos algorithm described above. However, as shown by Strachey and Francis [8], in the case when breakdown does not occur, our basic algorithm and the Lanczos algorithm, with $p = q = e_1$, the first standard basis vector of $\mathbf{R}^n$, produce the same tridiagonal matrix, if computed with infinite precision. We will make use of this fact throughout the paper. (Under certain circumstances (see Section 3), our algorithm makes adjustments which can be viewed as changing the starting vectors.)

In this algorithm, the reduction of A to tridiagonal form is accomplished by a sequence of similarity transformations. Two types of transformation matrices are employed: orthogonal transformations, denoted by H (for '*Householder*'), and elementary Gaussian transformations, denoted by G. A key feature of the algorithm is that the orthogonal and Gaussian transformations are applied alternately. This allows for a partial pivoting scheme to be incorporated, which, in turn, implies that, in the case being considered in this section, at most $n - 2$ of the $(n-1)(n-2)/2$ multipliers required for the reduction are larger than 1 in absolute value. A further benefit of the alternation of orthogonal and Gaussian steps is the empirically

$$A \equiv A_0 = \begin{pmatrix} * & * & * & * & \dots & * \\ * & * & * & * & \dots & * \\ \times & * & * & * & \dots & * \\ \times & * & * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \times & * & * & * & \dots & * \end{pmatrix} \xrightarrow{H_1} A_1 = \begin{pmatrix} * & * & \times & \times & \dots & \times \\ * & * & * & * & \dots & * \\ 0 & * & * & * & \dots & * \\ 0 & * & * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & * & * & * & \dots & * \end{pmatrix}$$

$$\xrightarrow{G_1} A_2 = \begin{pmatrix} * & * & 0 & 0 & \dots & 0 \\ * & * & * & * & \dots & * \\ 0 & * & * & * & \dots & * \\ 0 & \times & * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \times & * & * & \dots & * \end{pmatrix} \xrightarrow{H_2} A_3 = \begin{pmatrix} * & * & 0 & 0 & \dots & 0 \\ * & * & * & \times & \dots & \times \\ 0 & * & * & * & \dots & * \\ 0 & 0 & * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & * & * & \dots & * \end{pmatrix}$$

$$\xrightarrow{G_2} A_4 = \begin{pmatrix} * & * & 0 & 0 & \dots & 0 \\ * & * & * & 0 & \dots & 0 \\ 0 & * & * & * & \dots & * \\ 0 & 0 & * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \times & * & \dots & * \end{pmatrix} \xrightarrow{H_3} \dots$$

$$\dots \xrightarrow{G_{n-2}} A_{2(n-2)} = \begin{pmatrix} * & * & 0 & 0 & \dots & 0 \\ * & * & * & 0 & \dots & 0 \\ 0 & * & * & * & \dots & 0 \\ 0 & 0 & * & * & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & * \end{pmatrix}$$

Figure 1. *Representation of the basic algorithm*

observed tendency of orthogonal transformations to "smooth out" the elements of the matrix, thus reducing the potential for exponential growth of elements due to transformation by elementary Gaussian matrices (see also [9]).

We hasten to point out that as the tridiagonal form is uniquely determined by the starting vectors p and q, the alternation of orthogonal and Gaussian transformations and the inclusion of the partial pivoting scheme have no effect, mathematically, on the final matrix. However, in finite digit arithmetic, the final matrix *does* depend on the ordering of the transformations and on the norms of the transformation matrices (which the pivoting scheme is designed to minimize). The accuracy of the

computed eigenvalues of A therefore significantly depends on the algorithm's ability to preserve accuracy at all steps of the reduction. Our basic algorithm is designed to maximize the accuracy of the tridiagonal form with a given pair of starting vectors. In Section 3 we will consider modifying the starting vectors in order to obtain further improved results.

Graphically, the basic algorithm is shown in Figure 1 (the symbol "$\times$" indicates the elements to be reduced to 0 by the next transformation).

In detail, let $A_0 = A$ and define the following sequence of matrices A_k, similar to A.

If $k = 2(j-1)$ is even, determine an orthogonal matrix H_j so that the entries in positions (m, j), $m = j+2, \ldots, n$ of the matrix

$$A_{k+1} = H_j^t A_k H_j$$

are 0, and the zero profiles of the first $j-1$ columns of A_k are unaltered (e.g., let H_j be a Householder transformation or a product of Givens rotations). Note $H_j = I_j \oplus U_j$, where I_j is the $j \times j$ identity matrix and $U_j \in \mathbf{R}^{(n-j)\times(n-j)}$ is orthogonal.

Now suppose $k = 2j-1$ is odd. Observe that if $P_{m,l}$ is a permutation matrix (i.e., the identity matrix with columns m and l interchanged), then the matrix $P_{m,l}A_kP_{m,l}$ will have the same leading row and column zero profile as does A_k, as long as $m, l > j+1$. We thus perform a partial pivoting step before using Gaussian transformations to eliminate the zeroes beyond the super-diagonal in row j. Namely, choose l so that

$$|a_{jl}| = \max_{j+2 \le m \le n} |a_{jm}|,$$

(lower case a's denote the entries of A_k) and set

$$A_k^{(1)} = P_{j+2,l} A_k P_{j+2,l}.$$

Next, let

$$\Gamma_j^{(1)} = I - e_{j+2} \sum_{m=j+3}^{n} \alpha_m e_m^t$$

$$= \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & -\alpha_{j+3} & -\alpha_{j+4} & \dots & -\alpha_n & \leftarrow \textit{Row } j+2 \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \end{pmatrix}$$

where $\alpha_m = a_{jm}^{(1)} / a_{j,j+2}^{(1)}$ is a multiplier formed form the entries in row j of $A_k^{(1)}$ and $\{e_m\}_1^n$ is the standard basis of $\mathbf{R}^n$. Observe that transforming A_k by $P_{j+2,l}$ implies that $|\alpha_m| \leq 1$ for all $m = j+3, \dots, n$. Thus $\| \Gamma_j^{(1)} \|_1 \leq 2$.

The matrix $\Gamma_j^{(1)}$ is an elementary Gaussian transformation matrix, with inverse

$$\Gamma_j^{(1)^{-1}} = I + e_{j+2} \sum_{m=j+3}^{n} \alpha_m e_m^t .$$

Transforming $A_k^{(1)}$ by $\Gamma_j^{(1)}$ subtracts α_m times column $j+2$ from column m, and adds α_m times row m to row $j+2$, for $m = j+3, \dots, n$. Thus the matrix

$$A_k^{(2)} = \Gamma_j^{(1)^{-1}} A_k^{(1)} \Gamma_j^{(1)}$$

$$= \begin{pmatrix} & & \ddots & & & & & & & & \\ \dots & 0 & * & * & * & 0 & 0 & 0 & \dots & & \\ & \dots & 0 & * & * & * & \times & 0 & 0 & \dots & \leftarrow \textit{Row } j \\ & & \dots & 0 & * & * & * & * & * & * & \dots \\ & & & \dots & 0 & * & * & * & * & * & \dots \\ & & & \dots & 0 & * & * & * & * & * & \dots \\ & & & & \vdots & \vdots & & & \ddots & & \\ & & & & \uparrow & & & & & & \\ & & & & \textit{Column } j & & & & & & \end{pmatrix}$$

has, in addition to the zeroes introduced in A_k by previous steps, zeroes in posi-

tions (j, m), for $m = j+3, \ldots, n$.

To complete this Gaussian step, we must eliminate the entry in position $(j, j+2)$ (as indicated by the "$\times$" in the matrix above). Thus let

$$\Gamma_j^{(2)} = I - \alpha_{j+2} e_{j+1} e_{j+2}^t$$

$$= \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & -\alpha_{j+2} & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \end{pmatrix} \quad \leftarrow \textit{Row } j+1$$

where

$$\alpha_{j+2} = a_{j,j+2}^{(2)} / a_{j,j+1}^{(2)} = a_{j,j+2}^{(1)} / a_{j,j+1}^{(1)} = a_{jl} / a_{j,j+1} \,.$$

Then

$$A_{k+1} = \Gamma_j^{(2)^{-1}} A_k^{(2)} \Gamma_j^{(2)}$$

$$= \begin{pmatrix} & & \ddots & & & & & & & & \\ \ldots & 0 & * & * & * & 0 & 0 & 0 & \ldots & & \\ & \ldots & 0 & * & * & * & 0 & 0 & 0 & \ldots & \\ & & \ldots & 0 & * & * & * & * & * & * & \ldots \\ & & & \ldots & 0 & * & * & * & * & * & \ldots \\ & & & \ldots & 0 & * & * & * & * & * & \ldots \\ & & & & \vdots & \vdots & & & \ddots & & \end{pmatrix} \quad \leftarrow \textit{Row } j$$

$$\uparrow$$

$$\textit{Column } j$$

has zeroes in positions (j, m), for $m = j+2, \ldots, n$.

Set $G_j = P_{j+2,l} \Gamma_j^{(1)} \Gamma_j^{(2)}$, so that

$$A_{k+1} = G_j^{-1} A_k G_j.$$

To summarize, starting with the matrix $A_0 = A$, we perform the similarity

transformations

$$\begin{aligned} A_0 &\rightarrow A_1 \equiv H_1^t A_0 H_1 \\ &\rightarrow A_2 \equiv G_1^{-1} A_1 G_1 = G_1^{-1} H_1^t A_0 H_1 G_1 \\ &\rightarrow A_3 \equiv H_2^t A_2 H_2 \\ &\rightarrow \dots \\ &\rightarrow A_{2(n-2)} \end{aligned}$$

Each orthogonal transformation H_j introduces zeroes below the sub-diagonal in column j. Each elementary Gaussian transformation G_j introduces zeroes across row j beyond the super-diagonal. Neither type of transformation affects the zero profile of any of the earlier rows or columns.

Assuming breakdown does not occur (i.e., no super-diagonal element is 0), the matrix $A_{2(n-2)}$ is tridiagonal.

3. Toward stability

The Gaussian transformations, G_j, described in Section 2 are formed as products of three matrices: $G_j = P_{j+2,l} \Gamma_j^{(1)} \Gamma_j^{(2)}$. As mentioned in that section, the column interchange induced by the matrix $P_{j+2,l}$ yields $\| \Gamma_j^{(1)} \|_1 \leq 2$. Since we also have $\| P_{j+2,l} \|_1 = 1$, the critical term with respect to the norm of the transformation G_j is the matrix $\Gamma_j^{(2)}$, hence the multiplier $\alpha_{j+2} = a_{j,j+2}^{(2)} / a_{j,j+1}^{(2)}$.

To observe the effect of the transformation $\Gamma_j^{(2)}$, we focus on three rows and two columns of A_k, namely rows j, $j+1$ and $j+2$ and columns $j+1$ and $j+2$:

$$\begin{pmatrix} & & \ddots & & & & & & & & \\ \dots & 0 & * & * & * & 0 & 0 & 0 & \dots & & \\ & \dots & 0 & * & * & a & b & 0 & 0 & \dots & \\ & & \dots & 0 & * & c & d & * & * & * & \dots \\ & & & \dots & 0 & e & f & * & * & * & \dots \\ & & & \dots & 0 & * & * & * & * & * & \dots \\ & & & \dots & 0 & * & * & * & * & * & \dots \\ & & & & \vdots & \vdots & & & & \ddots & \end{pmatrix} \leftarrow \textit{Row } j$$

$$\uparrow$$

$$\textit{Column } j+1$$

For simplicity, we have denoted $a_{j,j+1}$ by a, $a_{j,j+2}$ by b, and so on. The multiplier at this stage is $\alpha = b/a$, and the element most affected by this multiplier (other

than b, which is eliminated by the transformation) is the element d:

$$d \leftarrow d - \alpha c + \alpha(f - \alpha e) = d + \alpha(f - c) - \alpha^2 e\,.$$

As observed by Wilkinson [12], it is the term $\alpha^2 e$ which, when $\alpha >> 1$, can wreak havoc with the numerical accuracy of the tridiagonalization algorithm. Thus the prime focus of any attempt to attain a stabilized tridiagonalization method must be control of these critical multipliers.

There are two possible causes of a large multiplier in the j^{th} Gaussian transformation matrix, G_j (specifically, in $\Gamma_j^{(2)}$):

(1) The ordering of the elimination steps, and

(2) The starting vectors for the reduction.

For the first case, observe that if the permutations, $P_{j+2,l}$, are omitted, then the sequence of orthogonal and Gaussian transformations can be reorganized in any manner so long as the ordering of the orthogonal steps is preserved, the ordering of the Gaussian steps is preserved and the j^{th} Gaussian transformation occurs after the j^{th} orthogonal transformation. As an extreme case, we could apply all the orthogonal transformations first, reducing A to upper Hessenberg form, and then apply all the Gaussian transformations, reducing the Hessenberg matrix to tridiagonal form (of course, the entries in the transformation matrices depend on the particular sequence). It is quite conceivable that some alternate arrangement of orthogonal and Gaussian steps would avoid an otherwise unacceptably large multiplier. Note that in infinite precision arithmetic, all these rearrangements would produce the same tridiagonal matrix, but in finite precision arithmetic this is not the case.

However, the organization of the elimination steps as given in Section 2, with the inclusion of the permutation transformations, has an important consequence, namely, that of the $(n-1)(n-2)/2$ element eliminations which must be accomplished by the Gaussian transformations, at most $n-2$ will involve multipliers larger than 1 in absolute value. As multipliers less than 1 in absolute value will not amplify round-off error, an arrangement of the elimination steps which maximizes the number of such multipliers is very desirable.

Nonetheless, there is one reorganization of the elimination steps which is simple to include in the basic algorithm, as described in Section 2. It involves little extra work, introduces at most one extra multiplier larger than 1 and is frequently successful in removing a problem multiplier from the reduction. It is simply to interchange G_j and H_{j+1} (again, this is formal: the entries in these matrices depend on the order in which they are applied).

The transformation sequence thus becomes (again, the symbol "$\times$" indicates an element to be eliminated by the next transformation):

$$\cdots \xrightarrow{G_{j-1}} A_{2j-2} = \begin{pmatrix} \ddots & \ddots & & & & & & \\ \ddots & * & * & 0 & 0 & \cdots & & \\ \ddots & * & * & * & * & * & * & \cdots \\ & 0 & * & * & * & * & * & \cdots \\ & 0 & \times & * & * & * & * & \cdots \\ & \vdots & \times & * & * & * & * & \cdots \\ & & \times & * & * & * & * & \cdots \\ & & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$\xrightarrow{H_j} A_{2j-1} = \begin{pmatrix} \ddots & \ddots & & & & & & \\ \ddots & * & * & 0 & 0 & \cdots & & \\ \ddots & * & * & * & * & * & * & \cdots \\ & 0 & * & * & * & * & * & \cdots \\ & 0 & 0 & * & * & * & * & \cdots \\ & \vdots & 0 & \times & * & * & * & \cdots \\ & & \vdots & \times & * & * & * & \cdots \\ & & & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$\xrightarrow{H_{j+1}} A_{2j} = \begin{pmatrix} \ddots & \ddots & & & & & & \\ \ddots & * & * & 0 & 0 & \cdots & & \\ \ddots & * & * & * & \times & \times & \times & \cdots \\ & 0 & * & * & * & * & * & \cdots \\ & 0 & 0 & * & * & * & * & \cdots \\ & \vdots & 0 & 0 & * & * & * & \cdots \\ & & \vdots & 0 & * & * & * & \cdots \\ & & & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$\xrightarrow{G_j} \quad A_{2j+1} = \begin{pmatrix} \ddots & \ddots & & & & & & \\ \ddots & * & * & 0 & 0 & \dots & & \\ \ddots & * & * & * & 0 & 0 & \dots & \\ & 0 & * & * & * & \times & \times & \dots \\ & 0 & 0 & * & * & * & * & \dots \\ & \vdots & 0 & 0 & * & * & * & \dots \\ & & \vdots & 0 & * & * & * & \dots \\ & & & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$\xrightarrow{G_{j+1}} \quad A_{2j+2} = \begin{pmatrix} \ddots & \ddots & & & & & & \\ \ddots & * & * & 0 & 0 & \dots & & \\ \ddots & * & * & * & 0 & 0 & \dots & \\ & 0 & * & * & * & 0 & 0 & \dots \\ & 0 & 0 & * & * & * & * & \dots \\ & \vdots & 0 & 0 & * & * & * & \dots \\ & & \vdots & 0 & \times & * & * & \dots \\ & & & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \xrightarrow{H_{j+2}} \dots$$

For definiteness, let $M > 0$ be chosen so that multipliers up to size M will be deemed acceptable, and suppose that in the j^{th} Gaussian step a multiplier $\alpha = a^{(2)}_{j,j+2}/a^{(2)}_{j,j+1}$ is encountered, with $|\alpha| > M$ (mathematically, α is the multiplier in $\Gamma^{(2)}_j$, but computationally we can check whether $|\alpha| > M$ before we apply $P_{j+2,l}$ and $\Gamma^{(1)}_j$, since, also, $\alpha = a_{j,l}/a_{j,j+1}$). Then compute the orthogonal transformation H_{j+1} which zeroes out the elements in A_k below the sub-diagonal in column $j+1$ (recall $k = 2j-1$). Observe that in the similarity transformation

$$A_{k+1} = H^t_{j+1} A_k H_{j+1}$$

the entries of the j^{th} row of A_k are affected only by the second of the two matrix multiplications (since $H_{j+1} = I_{j+1} \oplus U_{j+1}$), so that it is a simple (and inexpensive) matter to determine the j^{th} row of A_{k+1} and check it to see if the consequent multipliers are acceptable.

One ramification of this extra orthogonal step is that the partial pivoting scheme can move the largest element in row j only as far left as column $j+3$ (any further will

destroy the zero profile of column $j+1$, since $a_{j+1,j+2} \neq 0$). Thus, we determine l so that

$$|a_{jl}| = \max_{j+3 \leq m \leq n} |a_{jm}|$$

and then set

$$A_k^{(1)} = P_{j+3,l} A_{k+1} P_{j+3,l} .$$

The elementary Gaussian transformations $\Gamma_j^{(1)}$ and $\Gamma_j^{(2)}$ then have the form

$$\Gamma_j^{(1)} = I - e_{j+3} \sum_{m=j+4}^{n} \alpha_m e_m^t$$

$$= \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & -\alpha_{j+4} & -\alpha_{j+5} & \cdots & -\alpha_n & \leftarrow \textit{Row } j+3 \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \end{pmatrix}$$

$$\Gamma_j^{(2)} = I - \alpha_{j+2} e_{j+1} e_{j+2}^t - \alpha_{j+3} e_{j+1} e_{j+2}^t$$

$$= \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & -\alpha_{j+2} & -\alpha_{j+3} & & & \leftarrow \textit{Row } j+1 \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \end{pmatrix}$$

where

$$\alpha_m = a_{jm}^{(1)} / a_{j,j+3}^{(1)} , \qquad m = j+4, \ldots, n$$

$$\alpha_{j+2} = a_{j,j+2}^{(1)} / a_{j,j+1}^{(1)} = a_{j,j+2} / a_{j,j+1}$$

$$\alpha_{j+3} = a_{j,j+3}^{(1)} / a_{j,j+1}^{(1)} = a_{jl} / a_{j,j+1}$$

Thus, we have that for the subsequent Gaussian elimination step, all but two of the multipliers, namely, α_{j+3} and α_{j+2}, will be at most 1 in absolute value.

For exactly the same reasons as outlined above, we must still require $|\alpha_{j+2}| \leq M$. However, since the entry in position $(j+1, j+3)$ of A_{k+1} (the position directly below the element e of the first diagram of this section) is set to 0 by the transformation H_{j+1}, there is no term involving α_{j+3}^2 when the Gaussian transformation is applied, and hence we can afford a much larger multiplier in this position and still maintain a bound on the size of the terms which appear in the computations. For this multiplier, we need only require $|\alpha_{j+3}| \leq M^2$.

Note that only $O(n-j)$ work is required to determine H_{j+1}, apply it to row j of A_k and check the sizes of the resulting multipliers. Further, if the step is successful and the full transformation

$$A_{k+2} = G_j^{-1} H_{j+1}^t A_k H_{j+1} G_j$$

is computed, no special consideration need be given to the next Gaussian step, G_{j+1}, as the matrix has the same leading zero profile as it would have had had these orthogonal and Gaussian steps not been interchanged.

Thus this borrowing of an orthogonal transformation from one step ahead in the reduction sequence is easily accomplished, entails little extra work (which is 'extra' only in the event that it is unsuccessful) and, according to our numerical experiments, is very frequently successful in avoiding a large multiplier in a problem row (see Section 4 for the results of our numerical testing).

The second source of a large multiplier $\alpha = a_{j,j+2}/a_{j,j+1}$ is more fundamental. If the starting vectors for the Lanczos algorithm are chosen in such a way as breakdown, or near breakdown, will occur, then no rearrangement of the orthogonal and Gaussian steps during the reduction will prevent the appearance of large multipliers. (This is because the occurrence of a zero pivot is determined by the entries of the moment matrix, $[p^t A^{(i+j-2)} q]$, where p and q are the starting vectors; see [10]). Thus, somehow, the starting vectors must be modified and the reduction process recommenced, ideally without involving much work.

As mentioned before, no progress has been made on the problem of successfully choosing the starting vectors before the reduction starts. The approach taken by our algorithm when a large multiplier is encountered is to make a small adjustment to one of the starting vectors, changing only the first few components. Computationally, this results in only a small amount of extra work. Mathematically, the underlying assumption is that a small change in one of the starting vectors will not significantly reduce the size of any of the already computed pivots (which might cause a large multiplier to appear during this adjustment phase), and yet might

sufficiently reduce the problem multiplier which triggered this adjustment initially.

Specifically, if, in the matrix A_k, where $k = 2j - 1$, an unacceptably large and unavoidable multiplier occurs, a Gaussian transformation of the form

$$G = I + \beta_2 e_1 e_2^t + \beta_3 e_1 e_3^t ,$$

where β_2 and β_3 are small, is applied to A_k. This has the effect of adding multiples of column 1 to columns 2 and 3, and subtracting those multiples of rows 2 and 3 from row 1, effectively changing one of the starting vectors from e_1 to $e_1 + \gamma_2 e_2 + \gamma_3 e_3$, for some γ_2 and γ_3:

$$G^{-1} A_k G = \begin{pmatrix}
* & * & \times & \times & 0 & \cdots & & & & & \\
* & * & * & 0 & \cdots & & & & & & \\
0 & * & * & * & 0 & \cdots & & & & & \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & & & & & \\
 & & & & \ddots & \ddots & \ddots & & & & \\
 & & & & \ddots & * & * & 0 & \cdots & & \\
 & & & & \ddots & * & * & * & * & \cdots & \leftarrow \textit{Row } j \\
 & & & & & 0 & * & * & * & \cdots & \\
 & & & & & & 0 & * & * & \cdots & \\
 & & & & & & \vdots & & & \ddots &
\end{pmatrix}$$

Thus, non-zero entries are introduced into positions (1, 3) and (1, 4), which must be eliminated. The elementary Gaussian transformation used for this elimination in turn introduces non-zeros in positions (2, 4) and (2, 5), etc. Eventually, assuming no large multipliers are encountered during this adjustment phase, we arrive back at (a modified) row j, which can again be checked for the size of its multipliers. If these multipliers are acceptable, the adjustment has been successful and we continue. If not, or if a large multiplier occurred during the re-elimination phase, new values of β_2 and β_3 are generated and we try again.

If, after a few (e.g., 2) adjustment attempts of this type, the algorithm has not succeeded in avoiding a large multiplier in the problem row, it increases the scope of the adjustment to include the fourth component of the starting vector, namely, by trying a starting vector of the form $e_1 + \gamma_2 e_2 + \gamma_3 e_3 + \gamma_4 e_4$, and so on. Also, the other starting vector (also originally e_1) can be modified. In effect, this amounts to transposing the matrix A_k before beginning the adjustment.

Observe that if a large multiplier is encountered on row j, then each attempt at adjusting the starting vectors involves only $O(n)$ work, so the effect on the overall

Table 1
Stability characteristics of the algorithm
as a function of the multiplier bound

Dimensions of Matrices	*Bound on Multipliers*	*Number of Successes*[1,2]	*Relative Error in Computed Eigenvalues*	
			Average	*Maximum*
25	25.	98	.58e–12	.17e–10
	50.	100	.12e–11	.49e–10
	100.	100	.16e–11	.75e–10
	250.	100	.27e–11	.39e–10
	1000.	100	.36e–10	.31e–08
50	25.	99	.15e–11	.58e–10
	50.	100	.27e–11	.63e–10
	100.	100	.45e–11	.49e–10
	250.	100	.25e–10	.65e–09
	1000.	100	.38e–10	.11e–08
75	25.	98	.47e–11	.13e–09
	50.	99	.89e–11	.26e–09
	100.	100	.13e–09	.81e–08
	250.	100	.55e–10	.25e–08
	1000.	100	.19e–08	.16e–06
100	25.	91	.37e–10	.15e–08
	50.	99	.75e–10	.31e–08
	100.	100	.49e–10	.35e–08
	250.	100	.81e–10	.35e–08
	1000.	100	.36e–09	.20e–07

[1] *For each dimension, 100 matrices were reduced, each matrix against each value of M.*

[2] *A reduction was deemed to have failed if more than 100 adjustments were attempted.*

complexity of the algorithm, which is $O(n^3)$, clearly depends on how often such adjustment is necessary, and how many attempts at adjusting the starting vectors are made before one is successful. Empirical evidence suggests that this adjustment is done only a few times during the reduction, and that only a few attempts are required before an adjustment is successful.

The results of our testing are summarized in the next section.

4. Numerical Results

The algorithm was tested against a large number of matrices of dimensions between 20 and 400. Three types of tests were performed, one focussing on the sta-

Table 2
Performance of adjusting algorithm

Dimensions of Matrices	*Number of Matrices*	*Number of Successes*[1]	*Adjustment Attempts*[2] *Average*	*Maximum*	*Extra Orthogonal Transformations* *Average*	*Maximum*
25	500000	499765	.15	50	.17	3
50	50000	50000	.28	18	.56	3
100	5000	5000	.61	21	1.18	5
200	1000	997	1.77	71	3.31	10
400	100	99	4.73	37	8.94	17

[1] *A reduction was deemed to have failed if more than 100 adjustments were attempted.*

[2] *For all tests,* $M = 100$.

bility characteristics, one focussing on the heuristic adjustment part of the algorithm and one comparing the algorithm with the Lanczos algorithm. Test matrices in all cases were generated randomly with entries uniformly distributed in $[-1, 1]$. The algorithm was programmed in *FORTRAN 77* using double precision for all computations. The program was run on a *SEQUENT Symmetry* (the authors would like to thank Professors P. Larson and A. George for providing access to the *SEQUENT*).

For the stability test, given a matrix A, its eigenvalues were determined by the *QR* algorithm, as programmed in the *EISPACK* library. Then A was reduced to a tridiagonal matrix $\tilde{A}$ using our algorithm. Finally, the eigenvalues of $\tilde{A}$ were computed, again using the *EISPACK QR* algorithm, and the eigenvalues of A and $\tilde{A}$ compared. The results are summarized in Table 1. The correlation between M, the upper bound on the size of the multipliers, and the relative error in the eigenvalues of A and $\tilde{A}$ is clearly shown.

For the second test, only the tridiagonalization algorithm was applied to each matrix generated. The algorithm was considered to have failed if the number of adjustment attempts exceeded a preset threshold. By not including the eigenvalue computation, it was possible to test the algorithm against a large number of matrices. For these tests, the value of M was set at 100. The results are summarized in Table 2, and indicate that it might be better to have the value of M be a (slowly) increasing function of the dimension.

As mentioned in Section 3, any adjustment to the starting vectors should be fairly small. Our preliminary testing showed in addition that an adjustment was more likely to be successful if the ordering $|\beta_2| > |\beta_3| > \ldots$ was maintained. For all of the tests reported here, β_i was generated randomly from a uniform distribution

Table 3
Distribution of eigenvalues by number of correct digits (250 matrices)

	Number of Correct Digits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$n = 20$																
Our alg:	1937	1840	976	221	26	–	–	–	–	–	–	–	–	–	–	–
Lanczos:	736	1322	1236	785	374	277	161	71	26	5	2	–	–	–	–	–
$n = 40$																
Our alg:	1834	4293	3008	782	79	4	–	–	–	–	–	–	–	–	–	–
Lanczos:	380	1382	2049	1843	1252	902	593	484	289	254	159	152	91	78	57	35
$n = 60$																
Our alg:	1541	5775	5778	1691	200	15	–	–	–	–	–	–	–	–	–	–
Lanczos:	265	1112	2157	2167	1625	1075	755	762	722	649	581	627	599	604	944	356
$n = 80$																
Our alg:	1074	6743	8852	2948	324	56	3	–	–	–	–	–	–	–	–	–
Lanczos:	90	829	2276	2703	2284	1576	1192	884	825	800	810	801	827	921	2636	546

All matrices were successfully reduced by both algorithms in these tests.

in the interval $[-.1/2^i, .1/2^i]$, for $i = 2, 3, \ldots$.

The third test involved comparing our algorithm and the Lanczos algorithm, with eigenvalue errors being used as the measure of performance. Matrices of dimensions 20, 30, 40 and 50 were generated, with 250 matrices tested for each dimension. For each matrix, the eigenvalues were computed using the QR algorithm, then the matrix was reduced to tridiagonal form by each algorithm and the eigenvalues recomputed with the QR algorithm. Using the first set of eigenvalues as a benchmark, the relative errors in the before-and-after eigenvalues were then pigeon-holed, indexed by their exponents base 10, which corresponds to the number of correct digits. The results are shown in Table 3.

As is clear from the table, our algorithm was much more successful at preserving the accuracy of the eigenvalues through the reduction to tridiagonal form. (One of the characteric difficulties of the Lanczos algorithm is the introduction of spurious eigenvalues [3], resulting in relative eigenvalue errors on the order of unity or greater. As the table indicates, the prevalence of this problem increases with dimension.)

The tests just summarized show the high rate of success and the strong stability characteristics of our algorithm. The first test also clearly demonstrates Wilkinson's result [12] that the stability of a reduction algorithm depends critically on control-

ling the norms of the transformation matrices, as the results show that the larger the value of M, the poorer the results.

5. Conclusions and Future Directions

In this paper we have presented a new algorithm for reducing a general matrix to tridiagonal form. The priciple features of the algorithm are:

- Orthogonal and elementary Gaussian transformations are alternated, permitting a partial pivoting strategy, which, in turn, implies that most of the multipliers involved in the Gaussian transformations are no more than 1 in absolute value. The interleaving of the orthogonal transformations also appears to have a smoothing effect on the elements of the transformed matrices, providing evidence for a suggestion to this effect made in [9].
- A bound, M, is enforced on the size of the remaining multipliers. If the bound is violated, the algorithm first performs an extra orthogonal transformation (which is extra only in the event that it does not solve the problem), under the assumption that the large multiplier is a consequence of the organization of the reduction steps. One useful feature of this extra step is that it allows for much larger multipliers without inducing larger terms in the computations. Our preliminary testing showed that this extra step is very often successful.
- If the extra orthogonal transformation is not successful at reducing the problem multiplier, the algorithm assumes that the source of the problem is more fundamental, namely due to the choice of starting vectors. A heuristic algorithm is then used to adjust the starting vectors for the reduction. To enhance numerical stability, M should not be too large. To keep the total work done by the algorithm small, and to ensure a reasonable chance of the success of any necessary adjusting steps, M should not be too small. We have obtained good results with $M = 100$.
- The heuristic adjustment algorithm just mentioned requires very little extra work per application. Our numerical testing shows that adjustment to the starting vectors is required only a few times on average, hence, on average, the contribution of the adjustment algorithm to the total work done by the reduction algorithm is negligible.

One observation we have made is that when the heuristic adjustment algorithm fails, it is often in the very late stages of the reduction to tridiagonal form. It seems,

at such a point, that it might be more worthwhile to use the pivot adjustment strategy suggested in [9]. If that were done, the final form of the reduced matrix would not be strictly tridiagonal, but rather *bordered* tridiagonal, in that the last column of the reduced matrix would have some non-zero entries. However, as discussed by Wilkinson [11], it is actually the bordered tridiagonal form which is invariant with respect to eigenvalue deflation, and not the tridiagonal form itself. Thus, there is no loss in having non-zero entries in the last column of the reduced matrix.

In the case where the heuristic adjustment algorithm does not appear to be working, there is also the possibility of not strictly enforcing the bound, M, on the size of the multipliers, particularly if the problem multiplier is only slightly larger than M.

We will report on the results of including these and other variations on our tridiagonalization algorithm in a future paper.

References

[1] P. A. Businger, *Reducing a matrix to Hessenberg form*, Math. Comp., **23** (1969), 819–821.

[2] J. Collum and R. A. Willoughby, *A Lanczos procedure for the modal analysis of very large nonsymmetric matrices*, Proceedings of the 23rd *IEEE* Conference on Decision and Control (1984), 1758–1761.

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

[4] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Dover Publications, Inc., New York, 1974.

[5] B. N. Parlett and D. R. Taylor, *A look ahead Lanczos algorithm for unsymmetric matrices*, Center for Pure and Applied Mathematics, University of California, Berkeley, *PAM-43* (1981).

[6] B. N. Parlett, D. R. Taylor and Z. A. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., **44** (1985), 105–124.

[7] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Lin. Alg. Appl., **34** (1980), 269–295.

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

[9] W. P. Tang, *A stabilized algorithm for tridiagonalization of an unsymmetric matrix*, *Technical Report CS-88-14*, University of Waterloo, 1988.

[10] D. R. Taylor, *Analysis of the look ahead Lanczos algorithm*, Ph.D. Thesis, Center for Pure and Applied Mathematics, University of California, Berkeley, *PAM-108* (1982).

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

[12] ———, *Instability of the elimination method of reducing a matrix to tri-diagonal form*, The Computer Journal, **5** (1962), 61–70.