# *Efficient Algorithms for Computing the Condition Number of a Tridiagonal Matrix*

Higham, Nicholas J.

1986

ISSN 1749-9097

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

# EFFICIENT ALGORITHMS FOR COMPUTING THE CONDITION NUMBER OF A TRIDIAGONAL MATRIX*

NICHOLAS J. HIGHAM†

**Abstract.** Let $A$ be a tridiagonal matrix of order $n$. We show that it is possible to compute $\|A^{-1}\|_\infty$, and hence $\text{cond}_\infty(A)$, in $O(n)$ operations. Several algorithms which perform this task are given and their numerical properties are investigated.

If $A$ is also positive definite then $\|A^{-1}\|_\infty$ can be computed as the norm of the solution to a positive definite tridiagonal linear system whose coefficient matrix is closely related to $A$. We show how this computation can be carried out in parallel with the solution of a linear system $Ax=b$. In particular we describe some simple modifications to the LINPACK routine SPTSL which enable this routine to compute $\text{cond}_1(A)$, efficiently, in addition to solving $Ax=b$.

**Key words.** matrix condition number, tridiagonal matrix, positive definite matrix, LINPACK

## 1. Introduction.

Tridiagonal matrices

$$A=\begin{bmatrix} a_1 & c_1 & & & 0 \\ b_2 & a_2 & c_2 & & \\ & b_3 & a_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & b_n & a_n \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{1.1}$$

arise in many areas of numerical analysis, such as spline analysis [5, p. 133], difference methods for boundary value problems [9] and the numerical solution of linear second order recurrence relations [2, pp. 14 ff.]. Since the nonzero elements of $A$ occur only within a band of width three, the cost of solving a tridiagonal system $Ax=b$ using Gaussian elimination with partial pivoting is $O(n)$ flops, as opposed to the $O(n^3)$ flops required when $A$ is a full matrix [5, p. 166]. (See [15] for the definition of "flop".)

Let $A$ be a square nonsingular matrix and $\|\cdot\|$ a matrix norm. When computing solutions of linear systems $Ax=b$ one would usually like to know the condition number of $A$,

$$\text{cond}(A)=\|A\|\,\|A^{-1}\|,$$

or at least an estimate, since this quantity measures the sensitivity of the true solution to perturbations in the data $A$ and $b$, and features in various bounds for the norm of the error in the computed solution [17, pp. 192 ff.]. For the $l_\infty$ matrix norm, given for $A=(a_{ij})$ by

$$\|A\|_\infty=\max_i \sum_j |a_{ij}|, \tag{1.2}$$

or the $l_1$ matrix norm, $\|A\|_1=\|A^T\|_\infty$, $\|A\|$ is readily obtained whereas computation of $\|A^{-1}\|$ is more difficult.

Let $A$ be a tridiagonal matrix of order $n$. One way of computing $\|A^{-1}\|_\infty$ is first to compute $A^{-1}$—using, say, Gaussian elimination with partial pivoting—and then to calculate the norm. However, this computation requires $O(n^2)$ flops, which, for large $n$, dominates the cost of solving $Ax=b$. We show that $\|A^{-1}\|_\infty$ may be computed in $O(n)$ flops; the methods used necessarily avoid explicit computation of the inverse.

* Received by the editors March 6, 1984. This work was carried out with the support of a SERC Research Studentship.

† Department of Mathematics, University of Manchester, Manchester M13 9PL, England.

The following definition is required.

DEFINITION. The tridiagonal matrix $A$ in (1.1) is *irreducible* if $b_2, b_3, \cdots, b_n$ and $c_1, c_2, \cdots, c_{n-1}$ are all nonzero; otherwise it is *reducible.*

We remark that this definition is consistent with the more usual definition of irreducibility which applies to a general square matrix [16, pp. 102, 104].

The algorithms to be derived in § 4 apply to irreducible tridiagonal matrices. In § 7 we suggest a simple way of dealing with a tridiagonal matrix which is reducible.

Sections 5 and 6 are concerned with the numerical properties of our algorithms when implemented in floating-point arithmetic. The numerical stability of one of the algorithms is demonstrated with the aid of a backward error analysis.

For the case where $A$ is positive definite we derive, in § 8, an alternative and more efficient way of computing $\|A^{-1}\|_\infty$. This method only requires the solution of one positive definite tridiagonal linear system. We show how the LINPACK routine SPTSL, which solves $Ax = b$ for positive definite tridiagonal matrices $A$, can be modified so that it also computes $\text{cond}_1(A)$, the latter computation proceeding in parallel with the solution of $Ax = b$.

The methods to be derived apply exclusively to the $l_1$ and $l_\infty$ norms; we comment briefly on the $l_2$ norm condition number. The quantity

$$\phi = (\text{cond}_\infty(A)\,\text{cond}_1(A))^{1/2}$$

provides an order of magnitude estimate of $\text{cond}_2(A)$ since [19, p. 82]

$$\text{cond}_2(A) \leqq \phi \leqq n\,\text{cond}_2(A),$$

and $\phi$ may be computed in $O(n)$ flops when $A$ is tridiagonal. A variety of alternative techniques are available for the estimation of $\text{cond}_2(A)$ when $A$ is symmetric tridiagonal; among these are the well-known methods of inverse iteration, Sturm sequences, and bisection [20].

**2. The inverse of a bidiagonal matrix.** We begin by developing some properties of the inverse of a bidiagonal matrix $B$. These lead to an efficient algorithm for the computation of $\|B^{-1}\|_\infty$, and are also of use in § 8 since the $LU$ factors of a tridiagonal matrix are bidiagonal, when they exist.

Consider the nonsingular upper bidiagonal matrix

$$(2.1)\qquad U = \begin{bmatrix} u_1 & c_1 & & & 0 \\ & u_2 & c_2 & & \\ & & u_3 & \ddots & \\ & & & \ddots & c_{n-1} \\ 0 & & & & u_n \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad u_i \neq 0, \quad 1 \leqq i \leqq n.$$

We find an explicit formula for the elements $\beta_{ij}$ of $U^{-1}$. The $j$th column, $x_j = (\beta_{1j}, \beta_{2j}, \cdots, \beta_{nj})^T$, of $U^{-1}$ satisfies $Ux_j = e_j$, where $e_j$ is the $j$th unit vector. It follows that

$$\beta_{ij} = 0, \qquad i > j,$$

$$\beta_{jj} = \frac{1}{u_j},$$

$$\beta_{ij} = -\frac{c_i \beta_{i+1,j}}{u_i}, \qquad i < j.$$

Hence

$$\beta_{ij}=\begin{cases}0, & i>j,\\ \dfrac{1}{u_j}\displaystyle\prod_{r=i}^{j-1}\left(\frac{-c_r}{u_r}\right), & i\leqq j,\end{cases} \tag{2.2}$$

where the empty product is defined to be 1. We observe from (2.2) that for all $i$ and $j$ $|\beta_{ij}|$ depends only on the moduli of the elements of $U$. In other words, the modulus of each element of $U^{-1}$ is independent of the signs of the elements of $U$. It follows, using (1.2), that

$$|T|=|U| \text{ implies } \|T^{-1}\|_\infty=\|U^{-1}\|_\infty, \tag{2.3}$$

where, for $A=(a_{ij})$, $|A|$ denotes the matrix $(|a_{ij}|)$.

There is one particular distribution of the signs of the elements which yields a matrix for which the $l_\infty$ norm of the inverse is easily calculated. To show this, we define for $A=(a_{ij})$ $A$'s *comparison matrix* $M(A)=(m_{ij})$ by

$$m_{ij}=\begin{cases}|a_{ij}|, & i=j,\\ -|a_{ij}|, & i\neq j.\end{cases} \tag{2.4}$$

From (2.2) it is clear that $M(U)^{-1}\geqq 0$, that is $M(U)^{-1}$ has nonnegative elements (cf. [12]). We now make use of the observation that if $A^{-1}\geqq 0$ then $\|A^{-1}\|_\infty=\|A^{-1}e\|_\infty$, where $e=(1,1,\cdots,1)^T$. Together with (2.3) and the fact that $|M(A)|=|A|$ this yields

$$\|U^{-1}\|_\infty=\|M(U)^{-1}\|_\infty=\|M(U)^{-1}e\|_\infty. \tag{2.5}$$

These relations are also valid if $U$ is lower bidiagonal.

Hence, in order to calculate $\|B^{-1}\|_\infty$ for a bidiagonal matrix $B$ it is only necessary to solve one bidiagonal linear system and then to evaluate the $l_\infty$ norm of the solution vector. We have the following algorithm, an analogue of which holds for a lower bidiagonal matrix.

ALGORITHM 2.1. Given the nonsingular upper bidiagonal matrix $U$ in (2.1) this algorithm computes $\gamma=\|U^{-1}\|_\infty$.

```
$z_n := 1/|u_n|$; $\gamma := z_n$
For $i := n-1$ to 1 step $-1$
    $z_i := (1+|c_i| * z_{i+1})/|u_i|$
    $\gamma := \max(\gamma, z_i)$.
```

*Cost.* $3n$ flops. (We count max $(\cdot)$ as a flop.)

Thus Algorithm 2.1 requires $O(n)$ flops, a significant reduction on the $O(n^2)$ flops which would be required to compute $U^{-1}$ and then calculate the norm.

**3. The inverse of a tridiagonal matrix.** Let $A$ be a nonsingular tridiagonal matrix of order $n$. $A$ can be represented in terms of $O(n)$ quantities: its nonzero elements. The next theorem shows that, as might be expected, $A^{-1}$ is also representable in terms of $O(n)$ quantities—even though $A^{-1}$ has in general $O(n^2)$ nonzero elements.

THEOREM 1. *Let the tridiagonal matrix $A$ in* (1.1) *be nonsingular.*

(1) *If $A$ is irreducible and $A^{-1}=(\alpha_{ij})$ then there are vectors $x$, $y$, $p$ and $q$ such that*

$$\alpha_{ij}=\begin{cases} x_i y_j, & i\leqq j,\\ p_i q_j, & i\geqq j.\end{cases} \tag{3.1}$$

(2) *If $A$ is reducible then*

(a) *if $c_i=0$, so that*

$$A=\begin{bmatrix} A_1 & 0\\ B_1 & A_2\end{bmatrix}, \tag{3.2}$$

*where $A_1\in\mathbb{R}^{i\times i}$ and $A_2\in\mathbb{R}^{(n-i)\times(n-i)}$ are tridiagonal, then*

$$A^{-1}=\begin{bmatrix} A_1^{-1} & 0\\ X & A_2^{-1}\end{bmatrix}, \tag{3.3}$$

*where $X\in\mathbb{R}^{(n-i)\times i}$ is a rank-one matrix if $b_{i+1}\neq 0$, or a zero matrix if $b_{i+1}=0$, and the theorem applies recursively to $A_1$ and $A_2$;*

(b) *if $b_{i+1}=0$, part* (a) *applies to $A^T$.*

*Proof.* (1). See [14, Thm. 2].

(2). If $c_i=0$ then $A$ is clearly of the form (3.2) and $0\neq \det(A)=\det(A_1)\det(A_2)$, so $A_1$ and $A_2$ are nonsingular. It is easily verified that $X=-A_2^{-1}B_1A_1^{-1}$. $B_1$ has at most one nonzero element, $b_{i+1}$, in its $(1,i)$ position, so if $b_{i+1}=0$, $X=0$; otherwise $B_1$ is of rank one and hence $X$ is of rank one. □

We remark that for the case $A=A^T$ part (1) of Theorem 1 is proved by Bukhberger and Emel'yanenko [1] and stated without proof by Graybill [10].

The vectors $x$ and $y$ (and similarly $p$ and $q$) in (3.1) are easily seen to be unique up to a nonzero scale factor; any nonzero element of $x$ or $y$ can be assigned an arbitrary nonzero value (clearly $x_1\neq 0$ and $y_n\neq 0$).

There is, in fact, some redundancy in the representation (3.1) of $A^{-1}$. For the four vectors $x$, $y$, $p$ and $q$ contain $4n-2$ "free" values, while $A$ depends on only $3n-2$ numbers. This redundancy arises from the way in which part (1) of Theorem 1 is proved, namely by considering the upper triangular and lower triangular parts of $A^{-1}$ separately. The following theorem provides a more concise representation of $A^{-1}$, in terms of $3n-2$ numbers.

THEOREM 2. *Let the tridiagonal matrix $A$ in* (1.1) *be nonsingular and irreducible. Then there exist vectors $x$ and $y$ such that $A^{-1}=(\alpha_{ij})$ is given by*

$$\alpha_{ij}=\begin{cases} x_i y_j d_j, & i\leqq j,\\ y_i x_j d_j, & i\geqq j,\end{cases} \tag{3.4}$$

*where*

$$d_j=\prod_{r=1}^{j-1}\left(\frac{c_r}{b_{r+1}}\right),\qquad 1\leqq j\leqq n.$$

*Proof.* Let $D=\operatorname{diag}(d_i)$. $D$ exists and is nonsingular since $A$ is irreducible. It is easily verified that $T=DA$ is tridiagonal, symmetric and irreducible. By applying Theorem 1 to $T$, and using the symmetry of $T$, we find that there are vectors $x$ and $y$ such that $T^{-1}=(\beta_{ij})$ is given by

$$\beta_{ij}=\begin{cases} x_i y_j, & i\leqq j,\\ y_i x_j, & i\geqq j.\end{cases}$$

From the relation $A^{-1}=T^{-1}D$ the result follows. □

*Remark.* Theorem 2, when combined with the method for computing $x$ and $y$ described in the next section, is essentially the same as [21, Thm. 2].

**4. Algorithms for the irreducible case.** Let the tridiagonal matrix $A$ in (1.1) be nonsingular and irreducible. We now show, using the results of the previous section, that $\|A^{-1}\|_\infty$ can be computed in $O(n)$ flops.

Theorem 1 asserts the existence of vectors $x$, $y$, $p$ and $q$ such that the elements $\alpha_{ij}$ of $A^{-1}$ are given by (3.1). Formation of each element $\alpha_{ij}$ of $A^{-1}$ explicitly, using (3.1), requires $O(n^2)$ multiplications. However, in order to evaluate $\|A^{-1}\|_\infty$ we need only the row sums of $A^{-1}$. The $i$th row sum of $A^{-1}$ is, from (3.1),

$$|p_iq_1|+|p_iq_2|+\cdots+|p_iq_{i-1}|+|x_iy_i|+|x_iy_{i+1}|+\cdots+|x_iy_n|,$$

which may be expediently rewritten as

$$|p_i|(|q_1|+|q_2|+\cdots+|q_{i-1}|)+|x_i|(|y_i|+|y_{i+1}|+\cdots+|y_n|).$$

Clearly, by accumulating the sums $t_i=|q_1|+\cdots+|q_i|$ and $s_i=|y_i|+\cdots+|y_n|$ the row sums of $A^{-1}$ can be evaluated in $O(n)$ flops, given the vectors $x$, $y$, $p$ and $q$. We now show how these vectors can be computed.

Following Ikebe [14] we equate the last columns in $AA^{-1}=I$ and the first rows in $A^{-1}A=I$, to obtain, using (3.1), $A(y_nx)=e_n$ and $(x_1y^T)A=e_1^T$, that is,

$$Ax=y_n^{-1}e_n, \tag{4.1}$$

$$A^Ty=x_1^{-1}e_1. \tag{4.2}$$

The one degree of freedom in the vectors $x$ and $y$ may be expended by setting $x_1=1$; then equations (4.1) and (4.2) may be solved for $x_2,\cdots,x_n,y_n,\cdots,y_1$ using the method of [14].

The vectors $p$ and $q$ are obtained in a similar way, using $A^T$ in place of $A$. Thus we obtain the following algorithm.

ALGORITHM 4.1. Given the nonsingular $n\times n$ irreducible tridiagonal matrix $A$ in (1.1) this algorithm computes $\gamma=\|A^{-1}\|_\infty$.

```
(1)  $x_1:=1$; $x_2:=-a_1/c_1$
     For $i:=3$ to $n$
         $x_i:=-(a_{i-1}*x_{i-1}+b_{i-1}*x_{i-2})/c_{i-1}$
     $y_n:=1/(b_n*x_{n-1}+a_n*x_n)$
     $y_{n-1}:=-a_n*y_n/c_{n-1}$
     For $i:=n-2$ to 1 step $-1$
         $y_i:=-(a_{i+1}*y_{i+1}+b_{i+2}*y_{i+2})/c_i$.
(2) Repeat step (1) with $x_i$, $y_i$, $b_i$ and $c_i$ replaced by $q_i$, $p_i$, $c_{i-1}$ and $b_{i+1}$ respectively.
(3)  $s_n:=|y_n|$
     For $i:=n-1$ to 1 step $-1$
         $s_i:=s_{i+1}+|y_i|$
     $t_1:=1$
```

```
For $i := 2$ to $n$
    $t_i := t_{i-1} + |q_i|$
$\gamma := \max(s_1, |p_n| * t_n)$
For $i := 2$ to $n-1$
    $\gamma := \max(\gamma, |p_i| * t_{i-1} + |x_i| * s_i)$.
```

*Cost.* $17n$ flops.

An algorithm for the computation of $\|A^{-1}\|_\infty$ which is in general more efficient than Algorithm 4.1 can be derived from Theorem 2. Equating the last columns, and the first columns, in $AA^{-1} = I$ we find, using (3.4), that $A(y_n d_n)x = e_n$ and $A(x_1 d_1)y = e_1$. These equations may be rewritten, using $d_1 = 1$, as

$$Ax = (y_n d_n)^{-1} e_n, \tag{4.3}$$

$$Ay = x_1^{-1} e_1. \tag{4.4}$$

Choosing $x_1 = 1$, we can solve for $x$ as in the previous algorithm and then determine $y$ from (4.4), making use of the knowledge that $y_n \neq 0$.

Because the factor $d_j$ is common to each element in the $j$th column of $A^{-1}$ (see (3.4)) it is more convenient to evaluate the $l_1$ norm of $A^{-1}$ than to evaluate the $l_\infty$ norm.

ALGORITHM 4.2. Given the nonsingular $n \times n$ irreducible tridiagonal matrix $A$ in (1.1) this algorithm computes $\gamma = \|A^{-1}\|_1$.

```
(1) $x_1 := 1$; $x_2 := -a_1/c_1$
    For $i := 3$ to $n$
        $x_i := -(a_{i-1} * x_{i-1} + b_{i-1} * x_{i-2})/c_{i-1}$.
(2) $z_n := 1$; $z_{n-1} := -a_n/b_n$
    For $i := n-2$ to $1$ step $-1$
        $z_i := -(a_{i+1} * z_{i+1} + c_{i+1} * z_{i+2})/b_{i+1}$
    $\theta := a_1 * z_1 + c_1 * z_2$.
(3) $s_n := |z_n|$
    For $i := n-1$ to $1$ step $-1$
        $s_i := s_{i+1} + |z_i|$
    $t_1 := 1$
    For $i := 2$ to $n-1$
        $t_i := t_{i-1} + |x_i|$
    $d_1 := 1$; $\gamma := s_1$
    For $j := 2$ to $n$
        $d_j := d_{j-1} * c_{j-1}/b_j$
        $\gamma := \max(\gamma, (|z_j| * t_{j-1} + |x_j| * s_j) * |d_j|)$
    $\gamma := \gamma/|\theta|$.
```

*Cost.* $14n$ flops.

We now derive an algorithm for computing $\|A^{-1}\|_\infty$ which makes use of the $LU$ factorisation of $A$, assuming this exists. The algorithm is based on the representation (3.1) and is obtained by choosing $x_1=1$ and rewriting (4.2) and (4.1) as

(4.5) $$A^Ty=e_1,$$

(4.6) $$Az=e_n \qquad (z=y_nx).$$

ALGORITHM 4.3. Given a nonsingular $n\times n$ irreducible tridiagonal matrix $A$ possessing a $LU$ factorisation $A=LU$, this algorithm computes $\|A^{-1}\|_\infty$.

(1) Compute the $LU$ factorisation of $A$.

(2) Use the $LU$ factorisation to solve for the vectors $y$ and $z$ in (4.5) and (4.6). Similarly, solve for $p$ and $r$, where $Ap=e_1$, $A^Tr=e_n$ $(r=p_nq)$.

(3) Execute step (3) of Algorithm 4.1 with $p$, $q$ and $x$ replaced by $p_n^{-1}p$, $r$ and $y_n^{-1}z$ respectively.

*Cost.* $18n$ flops.

We note that if Algorithms 4.1, 4.2 and 4.3 are adapted to take advantage of symmetry, then in each case the operation count is reduced to about $11n$ flops.

**5. Computational considerations.** Let the tridiagonal matrix $A$ in (1.1) be nonsingular and irreducible, and consider the representation (3.1) of $A^{-1}=(\alpha_{ij})$ (the following applies to $p$ and $q$ in place of $y$ and $x$ if $A$ is replaced by $A^T$).

Let $x_1=1$. Using the standard determinantal formula for the elements of the inverse [17, p. 402] one finds that

$$y_n=\alpha_{1n}=\frac{c_1c_2\cdots c_{n-1}}{\det(A)}.$$

From (3.1) the transpose of $y$ is the first row of $A^{-1}$ and $x$ is the last column of $A^{-1}$ scaled by $y_n^{-1}$. Clearly, the vectors $x$, $y$, $p$ and $q$ in (3.1) can be very badly scaled, in the sense that the nonzero elements of any particular vector can vary widely in order of magnitude. This is true whatever the choice of $x_1$, and is not related to the conditioning of $A$.

*Examples.* (1) For the $n\times n$ tridiagonal matrix $A$ defined by $a_i=4$, $b_i=c_{i-1}=1$, we have $(x_1=1)|x_n|\approx\theta^{n-1}$, $|y_1|\approx\theta^{-1}$ and $|y_n|\approx\theta^{-n}$, where $\theta=2+\sqrt{3}$; $\operatorname{cond}(A)\leqq 3$.

(2)
$$A=\begin{bmatrix}1 & \varepsilon\\ 1 & 1\end{bmatrix},\quad 0<\varepsilon\ll 1,\qquad \|A^{-1}\|_\infty\approx 2.$$

Here $(x_1=1)$ $x_2=-1/\varepsilon$, $y_1=1/(1-\varepsilon)$ and $y_2=-\varepsilon/(1-\varepsilon)$.

We make two observations. First, there is a strong possibility of overflow and harmful underflow when Algorithm 4.1 is implemented on a computer. Second, it is not clear that in the presence of rounding errors the norm computed by Algorithm 4.1 will be of the correct order of magnitude (one does not usually want the condition number to many significant digits). For as Examples (1) and (2) illustrate we may have $\alpha_{ij}=x_iy_j=O(1)$ with $|x_i|\gg 1$ and $|y_j|\ll 1$; this is an ill-conditioned representation of $\alpha_{ij}$ in the sense that the product is very sensitive to absolute perturbations in $y_j$.

Similar comments apply to Algorithm 4.2 and the representation (3.4).

Note that the elements of the vectors $y$, $z$, $p$ and $r$ in Algorithm 4.3 are elements of $A^{-1}$, so they will not overflow on a computer as long as $\|A^{-1}\|_\infty$ is not too large. Step (3) of Algorithm 4.3 results in the divisions by the potentially very small quantities $p_n$ and $y_n$ being carried out at the last possible stage.

**6. Error analysis.** The observations of the last section lead us to investigate the accuracy of the norms computed by Algorithms 4.1, 4.2 and 4.3 in floating-point arithmetic and to consider how overflows and underflows can be avoided when the algorithms are implemented.

Consider Algorithm 4.3 implemented on a computer with $t$-digit base $\beta$ floating-point arithmetic, and assume that the algorithm runs to completion. Make the usual assumption [20, p. 113] that

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\delta), \qquad \text{op} = *, /, +, -,$$

for some $|\delta| \leqq \varepsilon$, where $\varepsilon = \frac{1}{2}\beta^{1-t}$ is the relative machine precision.

A backward error analysis in the style of de Boor and Pinkus [6] reveals that the computed $LU$ factors from step (1) of Algorithm 4.3 satisfy (using a bar denote a computed quantity)

$$\bar{L}\bar{U} = A + E, \tag{6.1}$$

where $E$ satisfies

$$|E| \leqq \mu_1 \varepsilon |\bar{L}|\,|\bar{U}|; \tag{6.2}$$

$\mu_i$, $i = 1, 2, \cdots$, denotes a constant of order one. (The absence of a term involving $n$ in this and subsequent bounds is due to the fact that $A$ is tridiagonal.)

The standard backward error analysis for solution of a triangular system [17, p. 408] can be used to show that the computed vector $\bar{y}$ from step (2) of Algorithm 4.3 satisfies

$$(\bar{U} + \delta\bar{U})^T(\bar{L} + \delta\bar{L})^T \bar{y} = e_1,$$

where

$$|\delta\bar{U}| \leqq \mu_2 \varepsilon |\bar{U}| \quad \text{and} \quad |\delta\bar{L}| \leqq \mu_3 \varepsilon |\bar{L}|. \tag{6.3}$$

It follows that $\bar{y}$ is the true solution of

$$W^T \hat{y} = e_1, \tag{6.4}$$

where

$$W = A + F = (\bar{L} + \delta\bar{L})(\bar{U} + \delta\bar{U}).$$

Combining (6.1), (6.2) and (6.3) we have, writing $\mu = \max(\mu_1, \mu_2, \mu_3)$,

$$|F| \leqq (3\mu\varepsilon + \mu^2\varepsilon^2)|\bar{L}|\,|\bar{U}|,$$

from which, using (6.1) and (6.2), it can be deduced that $W$ is tridiagonal and irreducible.

Let $\hat{z}$ satisfy

$$W\hat{z} = e_n. \tag{6.5}$$

By comparing the back substitutions which determine $\hat{z}$ and the computed vector $\bar{z}$ from step (2) of Algorithm 4.3, one finds that

$$\bar{z} = (1+\rho)\hat{z}, \qquad |\rho| \leqq \mu_4 n \varepsilon. \tag{6.6}$$

Thus we have shown that the vectors $\bar{y}$ and $\bar{z}$ computed by Algorithm 4.3 in floating-point arithmetic are, respectively, $\hat{y}$, and an approximation to $\hat{z}$ with small componentwise relative error, where $\hat{y}$ and $\hat{z}$ are the true vectors in (4.5) and (4.6) corresponding to using the matrix $W = A + F$ in place of $A$.

Now assume that

$$\|F\|_\infty \approx \varepsilon \|A\|_\infty \tag{6.7}$$

and

$$\text{cond}_\infty(A)\varepsilon \ll 1. \tag{6.8}$$

Then, using the result [17, p. 189]

$$\frac{\|A^{-1}-(A+E)^{-1}\|}{\|A^{-1}\|} \leqq \frac{\phi}{1-\phi}, \qquad \phi = \text{cond}(A)\frac{\|E\|}{\|A\|} < 1,$$

we have, approximately,

$$\|A^{-1} - W^{-1}\|_\infty \leqq \text{cond}_\infty(A)\varepsilon\|A^{-1}\|_\infty. \tag{6.9}$$

From (6.4), (6.5) and (6.6) it follows that the "upper triangular parts" $\bar{R}_i$ of the row sums computed in step (3) of Algorithm 4.3 (that is, $\bar{R}_i$ approximates $|\alpha_{ii}| + \cdots + |\alpha_{in}|$, where $A^{-1} = (\alpha_{ij})$) satisfy

$$\bar{R}_i = \left(\sum_{j \geqq i} |\beta_{ij}|\right)(1 + O(n\varepsilon)),$$

where $W^{-1} = (\beta_{ij})$. Hence

$$\left|\sum_{j\geqq i} |\alpha_{ij}| - \bar{R}_i\right| \approx \left|\sum_{j\geqq i} (|\alpha_{ij}| - |\beta_{ij}|)\right|$$

$$\leqq \sum_{j\geqq i} |\alpha_{ij} - \beta_{ij}| \leqq \text{cond}_\infty(A)\varepsilon\|A^{-1}\|_\infty,$$

using (6.9). A similar result holds for the remaining parts of the computed row sums.

Thus we conclude that if (6.7) and (6.8) are satisfied, then the number $\bar{\gamma}$ computed by Algorithm 4.3 in floating-point arithmetic satisfies, approximately,

$$\frac{|\bar{\gamma} - \|A^{-1}\|_\infty|}{\|A^{-1}\|_\infty} \leqq 2\,\text{cond}_\infty(A)\varepsilon. \tag{6.10}$$

This is just about the best that can be expected, since it can be shown that the condition number for the problem of computing $\|A^{-1}\|_\infty$ is $\text{cond}_\infty(A)$.

Algorithm 4.3 breaks down in step (3) if the computed quantity $\bar{y}_n$ is zero. It is easily checked that this can happen only if $\bar{y}_n$, or some intermediate quantity, underflows to zero (cancellation cannot take place). Unfortunately, underflow can occur even for quite well-behaved matrices, as indicated by the first example in § 5. The possibility of underflow (and overflow) may be avoided by representing all the numbers which arise in Algorithm 4.3 by a pair $(d, e) \equiv d \times b^e$, where $1 \leqq d < b$ (say), $e$ is an integer, and $b$ is some base, preferably a power of the machine base $\beta$. (This idea can also be applied to Algorithms 4.1 and 4.2.) Then, for example, the computation $z = x * y$ becomes

$$z = (d_x * d_y) \times b^{e_x + e_y} = d_z \times b^{e_z},$$

where $d_z$ is kept in the desired range through a suitable scaling by a power of $b$ together with an adjustment of the exponent $e_z$. This technique, with $b = 10$, is used in LINPACK in all the code which evaluates determinants.

We have been unable to prove a result such as (6.10) for Algorithms 4.1 and 4.2. When Algorithm 4.2 for example is executed in floating-point arithmetic, steps (1) and

(2) can be interpreted as being exact for perturbed matrices $A+E_1$ and $A+E_2$ with $|E_i| \leqq \mu_i \varepsilon |A|$, $i=1,2$, but it is difficult to assess the effect of $E_1 \neq E_2$, which will be true in general, on the accuracy of the row sums computed in step (3). Note that a forward error analysis is not helpful because the coefficient matrices of the triangular systems which are solved in Algorithms 4.1 and 4.2 can be arbitrarily ill-conditioned—their diagonal elements are $b_i$'s or $c_i$'s.

However, our experience in using Algorithms 4.1 and 4.2 is that they both produce extremely accurate results, whatever the condition number of $A$. In fact, we have not come across an instance in which the norms computed by Algorithms 4.1, 4.2 and 4.3 differed from each other in more than the last two significant digits—even when the computed vectors had elements covering the whole spectrum of machine numbers: from underflow level to overflow level. The machine used here was a Commodore 4032 microcomputer with $\beta=2$, $t=32$ and exponent range $-128 \leqq e \leqq 127$.

We feel that this unexpectedly high accuracy of the computed norms is due to the phenomenon observed by Wilkinson [19, pp. 103 ff.], [20, pp. 249 ff.] (see also [17, p. 150]), whereby the accuracy of the computed solution to a triangular system is commonly independent of the condition number of the coefficient matrix.

Condition (6.7) relates to the stability of Gaussian elimination without pivoting and will certainly be satisfied if $A$ is diagonally dominant; this is often the case in recurrence relation applications [3]. To avoid both large element growth and the possibility of Gaussian elimination breaking down one could use partial pivoting in Algorithm 4.3, obtaining $PA=LU$, with a guaranteed bound of 2 for the "growth factor" [17, p. 158]. Note, however, that pivoting changes the form of the triangular factors ($L$ is now lower triangular with at most two nonzero elements per column and $U$ is upper triangular with $u_{ij}=0$ for $j>i+2$), resulting in a possible extra $n$ flops for both the factorisation stage and each substitution; and our proof of the result (6.10) does not apply in this case.

Note that everything we have said concerning the properties and computation of the vectors in (3.1) and (3.4) applies perforce to the methods for inverting irreducible tridiagonal matrices which have been proposed in [1], [14], [21].

## 7. Dealing with reducibility.

For particular classes of matrix it is possible to verify irreducibility in advance. Difference methods for boundary value problems can lead to tridiagonal matrices with off-diagonal elements of the form $\alpha + O(h)$, where $\alpha > 0$ and $h$ is the mesh length [16, pp. 96 ff.]; here, irreducibility is assured for sufficiently small $h$. The tridiagonal matrices which arise in the numerical solution of linear second order recurrence relations can be assumed, without loss of generality, to be irreducible [3]. In some situations, however, one will be unable to guarantee irreducibility. In this section we describe how to deal with the case where $A$ is reducible.

Let $A$ be an $n \times n$ nonsingular reducible tridiagonal matrix. If $A$ is symmetric then $A$ has the block form $\operatorname{diag}(A_1, A_2, \cdots, A_k)$, where each matrix $A_i$ is nonsingular and either diagonal, or irreducible and tridiagonal. Hence $\|A^{-1}\|_\infty = \max_i \|A_i^{-1}\|_\infty$, which can be computed in $O(n)$ flops by applying Algorithm 4.3 (say) to each of the matrices $A_i$, as appropriate.

If $A$ is nonsymmetric then we suggest the following approach. Let $E$ be the tridiagonal matrix whose diagonal elements are zero, and whose off-diagonal elements are zero or $\delta/2$ according as the corresponding elements of $A$ are nonzero or zero; here, $\delta$ is a small nonzero number. The matrix $G=A+E$ is irreducible and tridiagonal. If

$$\delta \|A^{-1}\|_\infty \leqq 0.1 \tag{7.1}$$

then [18, p. 293]

$$\tfrac{8}{9}\|A^{-1}\|_\infty \leqq \|G^{-1}\|_\infty \leqq \tfrac{10}{9}\|A^{-1}\|_\infty.$$

Thus, by choosing $\delta$ small enough a satisfactory approximation to $\|A^{-1}\|_\infty$ can be computed in $O(n)$ flops, by applying one of the algorithms for the irreducible case to $G$. A suitable choice for $\delta$ is a small multiple of the relative machine precision $\varepsilon$. For such a $\delta$, (7.1) will be satisfied unless $A$ is very nearly singular to working precision (assuming $\|A\|_\infty \approx 1$). We note that the poor scaling discussed in § 5 is quite likely to be associated with $G$ if $A$ has many zero elements on its subdiagonal and superdiagonal.

An alternative method for dealing with the case where $A$ is nonsymmetric and reducible, leading to computation of $\|A^{-1}\|_\infty$ rather than an approximation, is described in [13]. The method is motivated by part (2) of Theorem 1 and regards $A$ as a block tridiagonal matrix, where the diagonal blocks are tridiagonal and irreducible. The computational cost is again $O(n)$ flops but the method is rather more difficult to implement than the above technique.

## 8. Positive definiteness.

**8.1.** Let

$$A=\begin{bmatrix} a_1 & b_2 & & & 0 \\ b_2 & a_2 & b_3 & & \\ & b_3 & a_3 & \ddots & \\ & & \ddots & \ddots & b_n \\ 0 & & & b_n & a_n \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{8.1}$$

be a positive definite tridiagonal matrix with, necessarily,

$$a_i > 0, \qquad 1 \leqq i \leqq n. \tag{8.2}$$

In this section we derive a method for computing $\|A^{-1}\|_\infty$ which is more efficient than the "symmetric" versions of Algorithms 4.1, 4.2 and 4.3 and which does not require $A$ to be irreducible.

We begin by showing that there is a matrix $D = \operatorname{diag}(d_i)$, $d_i = \pm 1$, such that

$$M(A) = DAD, \tag{8.3}$$

where $M(A)$ is the comparison matrix defined in (2.4). Writing $W = DAD$, we have

$$w_{ii} = a_i, \qquad 1 \leqq i \leqq n$$

and

$$w_{i,i+1} = d_i b_{i+1} d_{i+1}, \qquad 1 \leqq i \leqq n-1.$$

Let $d_1 = 1$ and $d_{i+1} = -\operatorname{sgn}(d_i b_{i+1})$, $1 \leqq i \leqq n-1$, where

$$\operatorname{sgn}(x) = \begin{cases} 1, & x \geqq 0, \\ -1, & x < 0. \end{cases}$$

$W$ is evidently tridiagonal and symmetric, $w_{ii} = a_i = |a_i|$ (using (8.2)), $|w_{i,i+1}| = |b_{i+1}|$ and $w_{i,i+1} \leqq 0$. Hence $W = M(A)$, which establishes (8.3).

$A$ is positive definite and hence has a Choleski factorisation

$$A = LL^T, \tag{8.4}$$

where

$$L=\begin{bmatrix} l_1 & & & & 0 \\ m_2 & l_2 & & & \\ & m_3 & l_3 & & \\ & & \ddots & \ddots & \\ 0 & & & m_n & l_n \end{bmatrix}$$

with $l_i > 0$ for all $i$. We claim that

(8.5) $$M(A)=M(L)M(L)^T.$$

This is proved by noting, first, that (8.4) implies $a_i=m_i^2+l_i^2$ and $b_{i+1}=l_i m_{i+1}$. Then, writing $H=M(L)M(L)^T$,

$$h_{ii}=(-|m_i|)(-|m_i|)+l_i^2=a_i=|a_i|$$

and

$$h_{i,i+1}=l_i(-|m_{i+1}|)=-|b_{i+1}|.$$

Hence $H=M(A)$ as required.

Note that $M(A)$ is positive definite, by (8.5). In fact, the positive definiteness of $A$ is independent of the signs of the off-diagonal elements $\{b_i\}$.

From § 2, $M(L)^{-1}\geqq 0$, therefore

(8.6) $$M(A)^{-1}=M(L)^{-T}M(L)^{-1}\geqq 0.$$

The final result we require is

(8.7) $$M(A)^{-1}=|A^{-1}|,$$

which is a consequence of (8.3).

It follows from (8.6) and (8.7) that if $A$ is a positive definite tridiagonal matrix,

$$\|A^{-1}\|_\infty=\|\,|A^{-1}|\,\|_\infty=\|M(A)^{-1}\|_\infty=\|M(A)^{-1}e\|_\infty.$$

As in the bidiagonal matrix case, computation of the $l_\infty$ norm of the inverse reduces to solution of a linear system involving the comparison matrix. Thus we have

ALGORITHM 8.1. Given an $n\times n$ positive definite tridiagonal matrix $A$ this algorithm computes $\gamma=\|A^{-1}\|_\infty$.

Solve $M(A)z=e$, evaluating $\gamma:=\max_{1\leqq i\leqq n} z_i=\|z\|_\infty$.

*Cost.* $6n$ flops.

The operation count given for Algorithm 8.1 assumes the use of the $LDL^T$ factorisation of $M(A)$. The closely related Choleski factorisation is unattractive in this context because it requires $n$ square roots, and an extra $n$ divisions in the substitution stage. The $LDL^T$ factors of the matrix $A$ in (8.1),

$$L=\begin{bmatrix} 1 & & & & 0 \\ l_2 & 1 & & & \\ & l_3 & 1 & & \\ & & \ddots & \ddots & \\ 0 & & & l_n & 1 \end{bmatrix}, \qquad D=\operatorname{diag}(d_i),$$

are generated by

$$d_1 = a_1,$$

$$l_i = \frac{b_i}{d_{i-1}}, \qquad d_i = a_i - l_i b_i, \qquad 2 \leqq i \leqq n.$$

**8.2. The nesting technique.** LINPACK [7] has a routine SPTSL which solves $Ax = b$ for positive definite tridiagonal matrices $A$. We shall show that it is possible to "nest" Algorithm 8.1 inside the routine SPTSL in such a way that the composite routine computes $\mathrm{cond}_\infty(A)$ in addition to solving $Ax = b$, and is computationally more efficient than separate applications of SPTSL and Algorithm 8.1. Because SPTSL uses a nonstandard factorisation method which is more complicated than the $LDL^T$ factorisation, we first derive the nesting technique for the $LDL^T$ method.

An important feature which Algorithm 8.1 does not exploit is that the $LDL^T$ factorisations of $A$ and $M(A)$ are related: by comparison with (8.4) and (8.5), if $A = LDL^T$ then $M(A) = M(L)DM(L)^T$. Therefore, solution of $M(A)z = e$ can be accomplished using the $LDL^T$ factorisation of $A$, which has to be computed anyway in the course of solving $Ax = b$; there is no need to explicitly factorise $M(A)$. The next algorithm makes use of this observation, thereby saving $2n$ flops.

ALGORITHM 8.2. Given the $n \times n$ positive definite tridiagonal matrix $A$ in (8.1) and the vector $f$, this algorithm computes both the solution to the linear system $Ax = f$ and $\gamma = \|A^{-1}\|_\infty$. The solution overwrites $f$.

The statements marked with an asterisk are those which have been added to the basic routine for $Ax = f$ in order to compute $\|A^{-1}\|_\infty$.

```
(1)       $d_1 := a_1$
    (*)   $z_1 := 1$
          For $i := 2$ to $n$
              $l_i := b_i / d_{i-1}$
              $d_i := a_i - l_i * b_i$
              $f_i := f_i - l_i * f_{i-1}$
    (*)       $z_i := 1 + |l_i| * z_{i-1}$.
(2)       $f_n := f_n / d_n$
    (*)   $z_n := z_n / d_n$; $\gamma := z_n$
          For $i := n-1$ to 1 step $-1$
              $f_i := f_i / d_i - l_{i+1} * f_{i+1}$
    (*)       $z_i := z_i / d_i + |l_{i+1}| * z_{i+1}$
    (*)       $\gamma := \max(\gamma, z_i)$.
```

*Cost.* $9n$ flops.

In Algorithm 8.2 computation of $\|A^{-1}\|_\infty$ adds only $4n$ flops to the cost of solving $Ax = f$. Furthermore, computation of $\|A^{-1}\|_\infty$ does not introduce any loops over and above those required for solution of the linear system. This is an important feature, since typically the loop overheads may account for a significant portion of the machine execution time of Algorithm 8.2 [8]. We conclude that on most computers the execution time for a routine based on Algorithm 8.2 should be less than 80% greater than that of an equivalent routine which only solves $Ax=f$.

We now show how Algorithm 8.1 can be nested within the LINPACK routine SPTSL. First, we describe briefly the nonstandard reduction technique which this routine uses; for full details see [7]. The reduction consists of a form of Gaussian elimination without pivoting in which subdiagonal elements are eliminated using row operations working from the top, and, simultaneously, superdiagonal elements are eliminated using row operations working from the bottom. Thus zeros are introduced to the elements in positions $(2,1)$, $(n-1,n)$, $(3,2)$, $(n-2,n-1),\cdots$, in this order, until the two eliminations meet in the middle. The reduced system is such that the unknowns can be determined by a simple substitution process, which works from the middle to the top and bottom of the matrix simultaneously.

The algorithm used in SPTSL is known as the "burn at both ends" (BABE) algorithm. The motivation for the BABE algorithm is that each of its two loops (one for the reduction phase and one for the substitution phase) is executed only half as many times as the corresponding loop in a standard algorithm (since two eliminations or two substitutions are performed on each run through a loop), so that the loop overhead is reduced by a factor of two (cf. [8]). The LINPACK manual claims that the BABE algorithm can solve positive definite tridiagonal linear systems up to 25% faster than conventional algorithms.

It can be shown (see [13] for example) that the BABE algorithm corresponds to the factorisation

$$A=LUB, \tag{8.8}$$

where $L$ is unit lower bidiagonal, $U$ is unit upper bidiagonal, and the nonzero elements of $B$ lie on the diagonal and in positions $(1,2)$, $(2,3),\cdots,(k,k+1)$, $(k+2,k+1)$, $(k+3,k+2),\cdots,(n,n-1)$, where $k=[n/2]$ ($[x]$ denotes the integer part of $x$).

The following lemma is the basis for the application of the nesting technique to the BABE algorithm.

LEMMA. *If the positive definite tridiagonal matrix A is factored according to* (8.8), *then*

$$M(A)=M(L)M(U)M(B).$$

*Proof.* The proof proceeds by comparing, elementwise, the reduction phases of the BABE algorithm applied to $A$ and to $M(A)$. For further details see [13]. □

The lemma shows that the equation

$$M(A)z=e \tag{8.9}$$

can be solved using information gleaned during the solution of $Ax=b$ by the BABE algorithm, namely, the elements of $L$, $U$ and $B$.

The modifications to the LINPACK routine SPTSL which are required in order for this routine to compute $\text{cond}_1(A)$ (the same as $\text{cond}_\infty(A)$ since $A$ is symmetric) in addition to solving $Ax=b$ consist simply of extra statements, some for the evaluation of $\|A\|_1$ and some analogous to those marked with an asterisk in Algorithm 8.2. Two extra parameters are required by the modified routine: a work vector $Z$ of length $n$ and a REAL variable CONINV in which to return the reciprocal of the condition number. A vector $w$ satisfying

$$\|Aw\|_1=\text{CONINV}\|A\|_1\|w\|_1$$

is easily obtained from $z$ in (8.9) by using (8.3).

All the LINPACK code which performs condition estimation incorporates a scaling technique, designed to prevent overflow during computation of the condition estimates [4], [7]. Similar precautions are clearly desirable in the computation of the solution of (8.9). However, the LINPACK scaling technique requires $O(n^2)$ flops, which is prohibitively expensive in the context of SPTSL; for this special case we suggest a simple modification which brings the cost down to $O(n)$ flops. The modified scaling technique takes advantage of the bidiagonal nature of the matrix factors in (8.8).

ALGORITHM SCALE. Let $L=(l_{ij})$ be a nonsingular lower bidiagonal matrix. This algorithm computes a vector $z$, with $\|z\|_\infty \leqq 1$, and a number $\theta$, such that $Lz=\theta b$.

```
(1)        ($l_{1,0}:=0$; $z_0:=0$)
           $\theta:=1$
           For $i:=1$ to $n$
(*)            $\alpha:=\theta*b_i-l_{i,i-1}*z_{i-1}$
               If $|\alpha|>|l_{ii}|$ then
                   $\beta:=l_{ii}/\alpha$
(*)                $\theta:=\theta*\beta$
                   $w_i:=\beta$
                   $z_i:=1$
                   else
                   $w_i:=1$
                   $z_i:=\alpha/l_{ii}$.
(2)        $\mu:=1$
           For $i:=n-1$ to 1 step $-1$
(*)            If $w_{i+1}\neq 1$ then $\mu:=\mu*w_{i+1}$
(*)            $z_i:=z_i*\mu$.
```

*Cost.* $2n$ flops, plus at most $4n$ flops for the scalings, in the statements marked with an asterisk.

The LINPACK scaling technique would, on introducing a scaling at the $i$th stage, immediately rescale all the previously computed values $z_1, \cdots, z_{i-1}$. Instead Algorithm Scale stores the scale factors in the vector $w$ (thus an extra $n$ storage locations are required), and is thereby able to rescale at the final stage in $O(n)$ flops.

A modification to the LINPACK scaling technique which attempts to reduce the frequency of the scalings is described in [11]; this modification could profitably be adopted in Algorithm Scale. In a small number of tests that we performed using a version of Algorithm Scale which incorporates the technique in [11], the cost of the scaling in Algorithm Scale was never greater than $2n$ flops.

Finally, we note that the basic modifications to SPTSL suggested above increase the computational cost of the routine from $5n$ flops to $11n$ flops ($2n$ of which arise from the evaluation of $\|A\|_1$). Incorporation of the scaling method used in Algorithm Scale could at worst add another $8n$ flops to the operation count.

**9. Concluding remarks.** We conclude by briefly discussing the choice of algorithm for computing $\|A^{-1}\|_\infty$ from among those presented. If the tridiagonal matrix $A$ is positive definite or bidiagonal, then Algorithms 8.1 and 2.1 respectively are recommended. Both these algorithms are very satisfactory from the point of view of numerical stability. We note that the triangular systems which are solved are of the form discussed by Wilkinson [20, p. 250].

For general irreducible tridiagonal matrices $A$ the choice is between Algorithms 4.1, 4.2 and 4.3, for which the differences in computational cost are relatively small. In the context of solving a linear system $Ax = b$, a factorisation $PA = LU$ is likely to be already available, in which case Algorithm 4.3 (minus the first step and using $PA = LU$ in the second step) is attractive. In general, Algorithm 4.2 is both more efficient and easier to "code-up" than Algorithms 4.1 and 4.3. As regards the very real dangers of overflow and underflow when these algorithms are implemented, and their numerical stability, see §§ 5 and 6.

**Acknowledgments.** I would like to thank Professor G. H. Golub and Dr. J. R. Cash for suggesting this work.

I also wish to thank Dr. I. Gladwell and Dr. G. Hall for their constructive criticism of the manuscript, and the referees and editor for their helpful suggestions.

## REFERENCES

[1] B. Bukhberger and G. A. Emel'yanenko, *Methods of inverting tridiagonal matrices*, USSR Computat. Math. and Math. Phys., 13 (1973), pp. 10-20.

[2] J. R. Cash, *Stable Recursions: With Applications to the Numerical Solution of Stiff Systems*, Academic Press, London, 1979.

[3] ———, private communication, 1983.

[4] A. K. Cline, C. B. Moler, G. W. Stewart and J. H. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368-375.

[5] G. Dahlquist and A. Bjorck, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[6] C. de Boor and A. Pinkus, *Backward error analysis for totally positive linear systems*, Numer. Math., 27 (1977), pp. 485-490.

[7] J. J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[8] J. J. Dongarra and A. R. Hinds, *Unrolling loops in* FORTRAN, Software-Practice and Experience, 9 (1979), pp. 216-226.

[9] C. F. Fischer and R. A. Usmani, *Properties of some tridiagonal matrices and their application to boundary value problems*, SIAM J. Numer. Anal., 6 (1969), pp. 127-142.

[10] F. A. Graybill, *Introduction to Matrices with Applications in Statistics*, Wadsworth, Belmont, CA, 1969.

[11] R. G. Grimes and J. G. Lewis, *Condition number estimation for sparse matrices*, this Journal, 2 (1981), pp. 384-388.

[12] N. J. Higham, *Upper bounds for the condition number of a triangular matrix*, Numerical Analysis Report No. 86, Univ. Manchester, England, 1983; submitted for publication.

[13] ———, *Matrix condition numbers*, M.Sc. Thesis, University of Manchester, England, 1983.

[14] Y. Ikebe, *On inverses of Hessenberg matrices*, Linear Algebra and Appl., 24 (1979), pp. 93-97.

[15] C. B. Moler, MATLAB *User's Guide*, Technical Report CS81-1 (revised), Dept. Computer Science, Univ. New Mexico, Albuquerque, 1982.

[16] J. M. Ortega, *Numerical Analysis: A Second Course*, Academic Press, New York, 1972.

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

[18] J. H. Wilkinson, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281-330.

[19] ———, *Rounding Errors in Algebraic Processes*, Notes on Applied Science No. 32, Her Majesty's Stationery Office, London, 1963.

[20] ———, *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, Oxford, 1965.

[21] T. Yamamoto and Y. Ikebe, *Inversion of band matrices*, Linear Algebra and Appl., 24 (1979), pp. 105-111.