SIAM J. MATRIX ANAL. APPL.
Vol. 19, No. 3, pp. 776–796, July 1998

012

RELIABLE COMPUTATION OF THE CONDITION NUMBER OF A TRIDIAGONAL MATRIX IN $O(n)$ TIME*

INDERJIT S. DHILLON†

Abstract. We present one more algorithm to compute the condition number (for inversion) of an $n \times n$ tridiagonal matrix J in $O(n)$ time. Previous $O(n)$ algorithms for this task given by Higham [*SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 150–165] are based on the tempting compact representation of the upper (lower) triangle of J^{-1} as the upper (lower) triangle of a rank-one matrix. However they suffer from severe overflow and underflow problems, especially on diagonally dominant matrices. Our new algorithm avoids these problems and is as efficient as the earlier algorithms.

Key words. tridiagonal matrix, inverse, condition number, norm, overflow, underflow

AMS subject classifications. 15A12, 15A60, 65F35

PII. S0895479897314747

1. Introduction.

When solving a linear system $Bx = r$ we are interested in knowing how accurate the solution is. This question is often answered by showing that the solution computed in finite precision is exact for a matrix "close" to B and then measuring how sensitive the solution is to a small perturbation. The condition number of B,

$$\kappa(B) = \|B\| \cdot \|B^{-1}\|,$$

where $\|.\|$ is a matrix norm, is one such measure. It has been conjectured that the cost of computing the condition number with guaranteed accuracy is nearly the same as solving the linear system itself [10, 9]. For a dense $n \times n$ matrix B the cost of solving $Bx = r$ is $O(n^3)$, and the extra cost of computing the condition number accurately may be unacceptable. In such cases, an estimate of the condition number may be obtained at a reduced cost [15, 18].

When the coefficient matrix J is tridiagonal, the linear system $Jx = r$ may be solved in $O(n)$ time. The matrix J^{-1} is dense in general, and computation of $\kappa(J)$ by explicitly forming it would require $O(n^2)$ time. However J is completely determined by $3n-2$ parameters, and one may suspect that its inverse can be explicitly expressed in terms of an equal number of parameters. This is indeed true and J^{-1} does admit a more compact representation, namely that the upper (lower) triangle of J^{-1} is also the upper (lower) triangle of a rank-one matrix, which in turn is simply represented by the outer product of two vectors (see [3, 4, 6, 17, 21] and Theorem 2.1 below). This property of the inverse may be exploited to compute $\|J^{-1}\|_1$ and hence $\kappa_1(J)$, in $O(n)$ time; see the beginning of section 3 for details. Note that the 1-norm of a

*Received by the editors January 2, 1997; accepted for publication (in revised form) by N. J. Higham September 19, 1997; published electronically April 2, 1998. This research was supported in part, while the author was at the University of California, Berkeley, CA, by DARPA contract DAAL03-91-C-0047 through a subcontract with the University of Tennessee, DOE contract DOE-W-31-109-Eng-38 through a subcontract with Argonne National Laboratory, DOE grant DE-FG03-94ER25219, NSF grants ASC-9313958 and CDA-9401156, and DOE contract DE-AC06-76RLO 1830 through the Environmental Molecular Sciences construction project at Pacific Northwest National Laboratory (PNNL). The information presented here does not necessarily reflect the position or the policy of the U.S. Government and no official endorsement should be inferred.

http://www.siam.org/journals/simax/19-3/31474.html

†IBM Almaden Research Center, San Jose, CA 95120-6099 (dhillon@almaden.ibm.com).

matrix $B = (\beta_{ij})$ is given by

$$\|B\|_1 = \max_j \sum_i |\beta_{ij}|$$

and that $\|B\|_\infty = \|B^T\|_1$.

In [17], Higham gives three algorithms to compute $\|J^{-1}\|_\infty$ in $O(n)$ time for a general tridiagonal matrix J. However all these algorithms suffer from severe overflow and underflow problems, especially on diagonally dominant matrices. The reason for these seemingly unavoidable problems is that the intermediate quantities computed by these algorithms can vary widely in scale [20]. In this paper, we give a new algorithm that does not suffer from the above mentioned over/underflow problems. The new algorithm avoids such problems by computing sums of magnitudes of elements of the inverse itself.

For positive definite J, Higham gives another algorithm in [17] that does not suffer from over/underflow problems and is shown to be backward stable. However this algorithm is entirely different from the algorithms for a general tridiagonal. Our new algorithm works for any tridiagonal and includes positive definite J as a special case.

The paper is organized as follows. In section 2, we review the structure of the inverse of a tridiagonal matrix that enables computation of its norm in $O(n)$ time. In section 3, we present an outline of the algorithms given in [17] and show why they are unsuitable for general purpose use. We present the basic structure of our new algorithm in section 4. This algorithm works under the assumption that all principal leading and trailing submatrices are nonsingular. Section 5 sheds more light on the structure of the inverse when this assumption fails to hold. This leads to the improved algorithm of section 6, and in section 7 we give a roundoff error analysis that suggests its accuracy. This algorithm can overflow and underflow in rare cases, which is corrected by the algorithms of section 8. Accuracy of our new algorithms is confirmed by numerical results in section 10. Section 9 is a slight digression and presents an application of these algorithms for computing eigenvectors.

2. The inverse of a tridiagonal matrix. The results of this section are quite well known and are repeated here as we will frequently invoke them in later sections. A square matrix $B = (\beta_{ik})$ is called a *lower(upper) Hessenberg* matrix if $\beta_{ik} = 0$ for all pairs (i, k) such that $i + 1 < k$ $(k + 1 < i)$. Thus a lower Hessenberg matrix is nearly a lower triangular matrix but with a nonzero superdiagonal. The following theorem states that the upper half of the inverse of such a matrix admits a compact representation.

THEOREM 2.1. *Let $B = (\beta_{ik})$ be a nonsingular lower Hessenberg matrix of order n, and let $\beta_{i,i+1} \neq 0$, $i = 1, \ldots, n-1$. Then two column vectors x and y exist such that the upper half of B^{-1} equals the upper half of xy^T, i.e., $(B^{-1})_{ik} = x_i y_k$ for $i \leq k$.*

Proof. See [21]. □

Let

$$J = \begin{bmatrix} a_1 & c_1 & & & & 0 \\ b_1 & a_2 & c_2 & & & \\ & b_2 & a_3 & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & c_{n-1} \\ 0 & & & & b_{n-1} & a_n \end{bmatrix}. \tag{2.1}$$

The tridiagonal matrix given above is said to be *unreduced* or *irreducible* if $b_i \neq 0$ and $c_i \neq 0$ for all $i = 1, \ldots, n-1$. Since a tridiagonal matrix is both a lower and an upper Hessenberg matrix, we obtain the following theorem on the structure of the inverse of a tridiagonal matrix.

THEOREM 2.2. *Let J be a nonsingular unreduced tridiagonal matrix of order n. Then there exist vectors x, y, p, and q such that*

$$(J^{-1})_{ik} = \begin{cases} x_i y_k, & i \leq k, \\ p_i q_k, & i \geq k. \end{cases}$$

The vectors x and y (similarly p and q) are unique up to scaling by a nonzero factor. Note that $x_1 \neq 0$ and $y_n \neq 0$ since otherwise the entire first row or last column of J^{-1} would respectively be zero, contradicting our assumption that J is nonsingular. The above theorem seems to state that J^{-1} is determined by $4n-2$ parameters, but note that there is some redundancy in the representation of the diagonal elements since $x_i y_i = p_i q_i$ for $1 \leq i \leq n$. The following theorem makes it explicit that $3n-2$ parameters are sufficient to determine J^{-1} uniquely.

THEOREM 2.3. *Let J be a nonsingular unreduced tridiagonal matrix of order n. Then there exist vectors x and y such that*

$$(J^{-1})_{ik} = \begin{cases} x_i y_k d_k, & i \leq k, \\ y_i x_k d_k, & i \geq k, \end{cases}$$

where

$$d_1 = 1 \quad \text{and} \quad d_k = \prod_{j=1}^{k-1} \frac{c_j}{b_j}, \quad 2 \leq k \leq n.$$

Proof. The key observation is that the nonsymmetric matrix J may be written as $J = DT$, where $D = \operatorname{diag}(d_i)$ is as given above and T is symmetric. The result is then obtained by applying Theorem 2.2 to T^{-1}. See [17] for more details. □

When an off-diagonal entry is zero, it is easy to see that the "corresponding" block of the inverse is zero. For example, if $b_i = 0$ so that

$$J = \begin{bmatrix} J_1 & C_1 \\ 0 & J_2 \end{bmatrix},$$

then

$$J^{-1} = \begin{bmatrix} J_1^{-1} & X \\ 0 & J_2^{-1} \end{bmatrix},$$

where X is a rank-one matrix if $c_i \neq 0$ and zero otherwise. Note that the structure of X is consistent with Theorem 2.1.

3. Unreliability of earlier algorithms. In this section, we reproduce the three algorithms given in [17] and explain why they are unsatisfactory when implemented in finite precision. For more details on the algorithms see [16, 17].

From Theorem 2.2, the ith row sum of J^{-1} is

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

Algorithm HIGHAM_1
(1) $x_1 = 1$; $x_2 = -a_1/c_1$;
  for $i = 3$ to n
    $x_i = -(a_{i-1} * x_{i-1} + b_{i-2} * x_{i-2})/c_{i-1}$;
  $y_n = 1/(b_{n-1} * x_{n-1} + a_n * x_n)$;
  $y_{n-1} = -a_n * y_n/c_{n-1}$;
  for $i = n-2$ to 1 step -1
    $y_i = -(a_{i+1} * y_{i+1} + b_{i+1} * y_{i+2})/c_i$;
(2) Repeat step (1) with x_i, y_i, b_i, and c_i replaced by q_i, p_i, c_i, and b_i, 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 $n-1$
    $\gamma = \max(\gamma, |p_i| * t_{i-1} + |x_i| * s_i)$;
  $\|J^{-1}\|_\infty = \gamma$;

Algorithm HIGHAM_2
(1) $x_1 = 1$; $x_2 = -a_1/c_1$;
  for $i = 3$ to n
    $x_i = -(a_{i-1} * x_{i-1} + b_{i-2} * x_{i-2})/c_{i-1}$;
(2) $z_n = 1$; $z_{n-1} = -a_n/b_{n-1}$;
  for $i = n-2$ to 1 step -1
    $z_i = -(a_{i+1} * z_{i+1} + c_{i+1} * z_{i+2})/b_i$;
  $\theta = a_1 * z_1 + c_1 * z_2$;
(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 $i = 2$ to n
    $d_i = d_{i-1} * c_{i-1}/b_{i-1}$;
    $\gamma = \max(\gamma, (|z_i| * t_{i-1} + |x_i| * s_i) * |d_i|)$;
  $\gamma = \gamma/|\theta|$;
  $\|J^{-1}\|_1 = \gamma$;

FIG. 1. *Algorithms* HIGHAM_1 *and* HIGHAM_2 *compute* $\|J^{-1}\|_\infty$ *and* $\|J^{-1}\|_1$, *respectively.*

which can be simplified to

$$|p_i|(|q_1| + |q_2| + \cdots + |q_{i-1}|) + |x_i|(|y_i| + |y_{i+1}| + \cdots + |y_n|). \tag{3.1}$$

By forming the running sums

$$t_i = |q_1| + |q_2| + \cdots + |q_i|, \qquad s_i = |y_i| + |y_{i+1}| + \cdots + |y_n|,$$

all the row sums of J^{-1} may be computed in $O(n)$ time given the vectors x, y, p, and q. The vectors x and y (similarly p and q) may be computed by equating the last columns of $JJ^{-1} = I$ and the first rows of $J^{-1}J = I$ after setting x_1 to 1.

Algorithm HIGHAM_1 (see Figure 1) sets x_1 to 1 and solves $Jx = y_n^{-1}e_n$ for x and y_n. The last $n-1$ equations of $J^Ty = x_1^{-1}e_1$ are then used to solve for $y_1, \ldots, y_{n-1}$. $\|J^{-1}\|_\infty$ is then found by forming the running sums s_i, t_i and computing all the row sums using (3.1).

Algorithm HIGHAM_2 (see Figure 1) exploits Theorem 2.3 to compute $\|J^{-1}\|_1$. The vector x is computed as in the previous algorithm, and the last $n-1$ equations of $Jz = \theta e_1$ are then used to solve for $z = \theta y$. Finally the 1-norm of each column of J^{-1}, scaled by θ, is computed.

Algorithm HIGHAM_3 (see Figure 2) makes use of the LU factorization of J to solve for the first row and column of J^{-1}, which give the vectors y and p, respectively (x_1 and q_1 are set to 1). Similarly the last row and column of J^{-1} are also computed and then scaled by p_n^{-1} and y_n^{-1} to get the vectors q and x, respectively. These four vectors are then used to compute $\|J^{-1}\|_\infty$ as in Algorithm HIGHAM_1.

All of the above algorithms attempt to compute elements of the vectors x and y at some point. We show that these vectors are badly scaled especially when the matrix is diagonally dominant and, hence, well conditioned. Consider the $n \times n$ tridiagonal matrix with all diagonal elements equal to 4 and all off-diagonals equal to 1. The determinant of this matrix is asymptotical to θ^n with increasing n, where $\theta = 2 + \sqrt{3}$. By the Cauchy–Binet theorem that gives formulae for the elements of the inverse (see (4.6) below), $x_1y_1 = x_ny_n \approx \theta^{-1}$ while $|x_1y_n| \approx \theta^{-n}$. If we choose $x_1 = 1$, then $|y_n| \approx \theta^{-n}$ and $|x_n| \approx \theta^{n-1}$. The overflow threshold in double precision IEEE arithmetic is $2^{1023} \approx 10^{308}$ [2]. When $n = 540$, $\theta^{n-1} > 10^{308}$ and due

Algorithm HIGHAM_3
(1) Compute the LU factorization of J;
(2) Use the LU factorization to solve for the vectors y and z,
where $J^T y = e_1$ and $Jz = e_n$.
Similarly, solve for p and r, where $Jp = e_1$ and $J^T r = e_n$.
(3) Execute step (3) of Algorithm HIGHAM_1 with $q = p_n^{-1} r$ and $x = y_n^{-1} z$.

FIG. 2. *Algorithm* HIGHAM_3 *computes* $\|J^{-1}\|_\infty$.

to overflow all the above algorithms fail in double precision arithmetic. Note that since $|x_n/x_1| \approx \theta^{n-1}$ and $|y_n/y_1| \approx \theta^{-n+1}$, there is no choice of x_1 that can prevent overflow and underflow for all n. For the strongly diagonally dominant tridiagonal with $a_i = 1000$, $b_i = c_i = 1$, all three algorithms outlined above fail when n is only 105.

These over/underflow problems were recognized by Higham [17], [20, section 14.5], and consequently the existing LAPACK version 2.0 [1] has software only to estimate the condition number of a general tridiagonal matrix using Hager's condition estimator [15, 19]. For positive definite tridiagonals, LAPACK does contain software to accurately compute the condition number. This is based on an alternate algorithm given by Higham in [17] that is special to the positive definite case.

4. The new algorithm. As we illustrated above, the vectors x, y, p, and q that determine the inverse of a diagonally dominant matrix can be badly scaled. In this section, we present a new algorithm to compute $\|J^{-1}\|_1$ that computes sums of magnitudes of elements of J^{-1} without explicitly forming these vectors. Consequently our new algorithm does not suffer from over/underflow problems that are inevitable when x, y, p, and q are used.

Before giving all the details of our new algorithm, we illustrate the ideas on a 5×5 case. The structure of the inverse is

$$J^{-1} = \begin{bmatrix} \Delta_1 & x_1y_2 & x_1y_3 & x_1y_4 & x_1y_5 \\ p_2q_1 & \Delta_2 & x_2y_3 & x_2y_4 & x_2y_5 \\ p_3q_1 & p_3q_2 & \Delta_3 & x_3y_4 & x_3y_5 \\ p_4q_1 & p_4q_2 & p_4q_3 & \Delta_4 & x_4y_5 \\ p_5q_1 & p_5q_2 & p_5q_3 & p_5q_4 & \Delta_5 \end{bmatrix}, \quad \Delta_i \equiv x_iy_i = p_iq_i.$$

Let $s_u(i)$ denote the 1-norm of column i of the strict upper triangle of J^{-1}. Clearly

$$\begin{aligned} s_u(5) &= \left(\sum_{i=1}^{4} |x_i| \right) |y_5| \\ &= (s_u(4) + |\Delta_4|) \, \frac{|y_5|}{|y_4|}, \end{aligned}$$

and so there is a simple recurrence to build up $s_u(i)$ if Δ_i is known. Note that in the above we assumed that $y_4 \neq 0$, and, for now, we will assume that all x_i, y_i are nonzero. We can also build the following recurrence for Δ_i:

$$\Delta_{i+1} = x_{i+1}y_{i+1} \; = \; \Delta_i \, \frac{x_{i+1}}{x_i} \frac{y_{i+1}}{y_i}. \tag{4.1}$$

```
D+(1) = a1;
for i = 1 to n − 1
    L+(i) = bi/D+(i); U+(i) = ci/D+(i);
    D+(i + 1) = ai+1 − ci ∗ L+(i);
```

```
D−(n) = an;
for i = n − 1 to 1 step −1
    U−(i) = ci/D−(i + 1); L−(i) = bi/D−(i + 1);
    D−(i) = ai − bi ∗ U−(i);
```

FIG. 3. *Algorithms to compute the triangular decompositions of J.*

Having found Δ_i, $s_u(i+1)$ may be expressed as

$$s_u(i+1) = (s_u(i) + |\Delta_i|)\,\frac{|y_{i+1}|}{|y_i|}. \tag{4.2}$$

We will see later that the ratios x_{i+1}/x_i and y_{i+1}/y_i are easily evaluated. Similarly,

$$s_l(i-1) = (s_l(i) + |\Delta_i|)\,\frac{|q_{i-1}|}{|q_i|}, \tag{4.3}$$

where $s_l(i)$ denotes the 1-norm of the ith column of the strict lower triangle of J^{-1}.

It turns out that it is possible to express the above recurrences in terms of triangular factorizations of J; two of them, as it happens. For the moment assume that the following factorizations exist:

$$J = L_+ D_+ U_+, \tag{4.4}$$

$$J = U_- D_- L_-, \tag{4.5}$$

where L_+, L_- are unit lower bidiagonal, U_+ and U_- are unit upper bidiagonal, while D_+ and D_- are diagonal matrices. Note that in the above, we use "+" to indicate a process that takes rows in increasing order while "−" indicates a process that takes rows in decreasing order. Figure 3 details the algorithms for computing these factorizations. We denote the $(i+1, i)$ element of L_+ by $L_+(i)$ and the $(i, i+1)$ element of U_- by $U_-(i)$.

In our upcoming treatment we will extensively use the famous Cauchy–Binet formula

$$B \cdot \mathrm{adj}(B) = \det(B) \cdot I, \tag{4.6}$$

where $\mathrm{adj}(B)$ is the classical *adjugate* of B and is the transpose of the matrix of cofactors [24, p. 402], to get expressions for elements of B^{-1}.

Since J is tridiagonal, (4.6) implies that

$$\Delta_i \;=\; x_i y_i \;=\; \frac{\det(J^{1:i-1}) \cdot \det(J^{i+1:n})}{\det(J)},$$

where $J^{r:s}$ denotes the principal submatrix of J in rows and columns r through s. Hence the assumption that all x_i, y_i be nonzero is identical to the assumption that the triangular factorizations (4.4) and (4.5) exist. We will remove this assumption later.

Since $L_+ e_n = e_n$ and $e_1^T L_- = e_1^T$, the first row and last column of the inverse may be expressed as

$$w_1^T \;\equiv\; e_1^T J^{-1} = e_1^T L_-^{-1} D_-^{-1} U_-^{-1} \;=\; \frac{1}{D_-(1)} e_1^T U_-^{-1},$$

$$v_n \;\equiv\; J^{-1} e_n = U_+^{-1} D_+^{-1} L_+^{-1} e_n \;=\; \frac{1}{D_+(n)} U_+^{-1} e_n.$$

Algorithm NRMINV
Compute $J = L_+D_+U_+$ and $J = U_-D_-L_-$ (see Figure 3).
$\Delta_1 = 1/D_-(1)$;
for $i = 1$ to $n-1$
    $\Delta_{i+1} = \Delta_i * \frac{D_+(i)}{D_-(i+1)}$;
$s_u(1) = 0$;
for $i = 1$ to $n-1$
    $s_u(i+1) = (s_u(i) + |\Delta_i|) * |U_-(i)|$;
$s_l(n) = 0$;
for $i = n$ to 2 step -1
    $s_l(i-1) = (s_l(i) + |\Delta_i|) * |L_+(i-1)|$;
$\gamma = 0$;
for $i = 1$ to n
    $\gamma = \max(\gamma, s_u(i) + s_l(i) + |\Delta_i|)$;
$\|J^{-1}\|_1 = \gamma$;

FIG. 4. *Algorithm* NRMINV *computes* $\|J^{-1}\|_1$.

The crucial observation is that the ratios of successive entries in w_1 and v_n are given by entries in the triangular factorizations. More precisely, the above equations may be written as

$$U_-^T w_1 = \frac{1}{D_-(1)} e_1, \tag{4.7}$$

$$U_+ v_n = \frac{1}{D_+(n)} e_n. \tag{4.8}$$

By examining the $(i+1)$st equation of (4.7) and the ith equation of (4.8), $1 \le i \le n-1$, we get

$$-U_-(i) = \frac{w_1(i+1)}{w_1(i)} = \frac{x_1 y_{i+1}}{x_1 y_i}, \tag{4.9}$$

$$-U_+(i) = \frac{v_n(i)}{v_n(i+1)} = \frac{y_n x_i}{y_n x_{i+1}}. \tag{4.10}$$

Equations (4.9) and (4.10) may now be substituted in (4.1) and (4.2) to get

$$\Delta_{i+1} = \Delta_i \frac{U_-(i)}{U_+(i)} = \Delta_i \frac{D_+(i)}{D_-(i+1)}, \quad \Delta_1 = \frac{1}{D_-(1)}, \tag{4.11}$$

$$s_u(i+1) = (s_u(i) + |\Delta_i|) \cdot |U_-(i)|, \quad s_u(1) = 0. \tag{4.12}$$

Note that the first equation of (4.7) gives $w_1(1) = \Delta_1 = 1/D_-(1)$ while the last equation of (4.8) implies that $v_n(n) = \Delta_n = 1/D_+(n)$. Similarly, we get

$$s_l(i-1) = (s_l(i) + |\Delta_i|) \cdot |L_+(i-1)|, \quad s_l(n) = 0. \tag{4.13}$$

Equations (4.11), (4.12), and (4.13) lead to Algorithm NRMINV outlined in Figure 4. This new algorithm, when implemented in finite precision, delivers correct answers on the examples of the previous section. It is also more efficient than the

TABLE 1
Comparison of arithmetic operations.

Operations $\times$ n	Divisions	Multiplications	Additions
Algorithm HIGHAM_1	4	10	7
Algorithm HIGHAM_2	3	8	5
Algorithm HIGHAM_3	7	16	16
Algorithm NRMINV	**3**	**5**	**6**

algorithms of [17]. In Table 1, we list the approximate operation counts in Algorithm NRMINV and compare them to Higham's algorithms. Note that neither U_+ nor L_- is used in Algorithm NRMINV and hence the corresponding division operations to compute them (see Figure 3) are not counted in Table 1. For more details on the operation counts for Higham's algorithms, the reader is referred to discussions of Algorithms 2, 3, and 5 in his M.Sc. thesis [16].

Recall that for our new algorithm we assumed that the factorizations in (4.4) and (4.5) exist. In the next section, we shed more light on the structure of the inverse when triangular factorization breaks down, and in section 6, we present an algorithm that handles such a breakdown.

Formula (4.11) to compute the diagonal elements of the inverse is not new and has been known for some time to researchers, especially in boundary value problems. See Meurant's survey article [22] for such formulae and more on the behavior of the inverse of a tridiagonal matrix. More recently, the diagonal of the inverse has been used to compute eigenvectors of a symmetric tridiagonal matrix [12, 23, 13, 14]. Section 9 briefly explains the connection to eigenvectors.

5. More properties of the inverse. Consider the tridiagonal matrix J of even order with $a_i = 0$ and $b_i = c_i = 1$ for all i. The factorizations (4.4) and (4.5) do not exist and all the diagonal entries of its inverse equal zero; i.e., $x_i y_i = 0$. We now present a theory that enables us to handle such a case.

THEOREM 5.1. *Let J be a nonsingular tridiagonal matrix of order n. Then $\Delta_i \equiv (J^{-1})_{ii} = 0$ if and only if either $J^{1:i-1}$ or $J^{i+1:n}$ is singular.*

Proof. This follows from (4.6) which, due to J's tridiagonal structure, implies that

$$\Delta_i \equiv (J^{-1})_{ii} = \frac{\det(J^{1:i-1}) \cdot \det(J^{i+1:n})}{\det(J)}. \qquad \square \tag{5.1}$$

Since $\Delta_i = x_i y_i$, either $x_i = 0$ or $y_i = 0$ when $\Delta_i = 0$ (note that x_i and y_i cannot both be zero if J is nonsingular because otherwise by Theorem 2.3 J^{-1} would have a zero row and column). The following theorem states that $y_i = 0$ when $J^{i+1:n}$ is singular while $x_i = 0$ when the leading submatrix $J^{1:i-1}$ is singular.

THEOREM 5.2. *Let J be a nonsingular unreduced tridiagonal matrix of order n. Then*

$$s_u(i) \equiv \sum_{k=1}^{i-1} |(J^{-1})_{k,i}| = 0 \quad \textit{if and only if} \quad J^{i+1:n} \quad \textit{is singular.} \tag{5.2}$$

Similarly,

$$s_l(i) \equiv \sum_{k=i+1}^{n} |(J^{-1})_{k,i}| = 0 \quad \textit{if and only if} \quad J^{1:i-1} \quad \textit{is singular.}$$

Proof. By the Cauchy–Binet formula in (4.6), for $k < i$,

$$\det(J) \cdot (J^{-1})_{k,i} = (-1)^{k+i}(c_k c_{k+1} \cdots c_{i-1}) \det(J^{1:k-1}) \det(J^{i+1:n}). \tag{5.3}$$

Letting $k = 1$ (take $\det(J^{1:0}) = 1$), we see that for an unreduced J, $(J^{-1})_{1,i} = 0$ if and only if $J^{i+1:n}$ is singular. The result (5.2) now follows from (5.3). ∎

Note that if $J^{1:i-1}$ and $J^{i+1:n}$ are both singular, the above theorems imply that Δ_i, $s_u(i)$ and $s_l(i)$ are zero, i.e., J^{-1} has a zero column! This leads to the following corollary.

COROLLARY 5.3. *Let J be a tridiagonal matrix of order n. If J is nonsingular, then $J^{1:i-1}$ and $J^{i+1:n}$ cannot both be singular for any $i = 2, 3, \ldots, n-1$.*

The tridiagonal structure of J is essential to the above result. To emphasize this, consider

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

where $A^{1:1}$ and $A^{3:3}$ are singular but A is not.

Now we show that for a nonsingular J no two consecutive entries in x or y can be zero. In particular this implies that both $s_u(i)$ and $s_u(i+1)$ cannot be zero.

THEOREM 5.4. *Let J be a nonsingular unreduced tridiagonal matrix of order n. Then the last (first) column or row of J^{-1} cannot have two consecutive zero entries.*

Proof. Suppose that $v_{i-1} = v_i = 0$, where $Jv = e_n$. Then the ith equation $b_{i-1}v_{i-1} + a_i v_i + c_i v_{i+1} = 0$, where $i < n$ implies that $v_{i+1} = 0$. The $(i+1)$st equation further implies that $v_{i+2} = 0$ and so on. Thus $v_{n-1} = v_n = 0$ but then the last equation $b_{n-1}v_{n-1} + a_n v_n = 1$ cannot be satisfied. ∎

The following lemma is similarly proved using the three-term recurrence for tridiagonal matrices.

LEMMA 5.5. *Let J be an unreduced (or nonsingular) tridiagonal matrix of order n. Then no two consecutive leading (or trailing) principal submatrices of J are singular.*

Proof. Suppose that $J^{1:i-1}$ and $J^{1:i}$ are singular. Then, since

$$\det(J^{1:i+1}) = a_{i+1} \det(J^{1:i}) - b_i c_i \det(J^{1:i-1})$$

and

$$-\det(J^{1:i}) + a_i \det(J^{1:i-1}) = b_{i-1}c_{i-1} \det(J^{1:i-2}),$$

$J^{1:k}$ is singular for all $k = 1, 2, \ldots, n$. But if $J^{1:1}$ is zero, then $\det(J^{1:2}) = -b_1 c_1 \neq 0$ which leads to a contradiction. ∎

We make extensive use of the following theorem in the next section.

THEOREM 5.6. *Let J be an unreduced (or nonsingular) tridiagonal matrix of order n.*

$$\textit{If } J^{1:i} \textit{ is singular, then } \det(J) = \det(J^{1:i+1}) \det(J^{i+2:n}). \tag{5.4}$$

Similarly,

$$\textit{if } J^{i:n} \textit{ is singular, then } \det(J) = \det(J^{1:i-2}) \det(J^{i-1:n}). \tag{5.5}$$

Proof. Suppose that $J^{1:i}$ is singular. Then by Lemma 5.5, $J^{1:i+1}$ is nonsingular. The Schur complement of $J^{1:i+1}$ in J is

$$\mathcal{S}(J^{1:i+1}) \;=\; J^{i+2:n} - b_{i+1}c_{i+1}e_1e_{i+1}^T(J^{1:i+1})^{-1}e_{i+1}e_1^T.$$

By Theorem 5.1, the $(i+1, i+1)$ entry of $(J^{1:i+1})^{-1}$ must be 0. Hence $\mathcal{S}(J^{1:i+1}) = J^{i+2:n}$ and

$$\det(J) \;=\; \det(J^{1:i+1})\det(\mathcal{S}(J^{1:i+1})) \;=\; \det(J^{1:i+1})\det(J^{i+2:n}). \qquad \square$$

Next we see how to detect the singularity of a leading or trailing principal submatrix. When such a submatrix is singular, triangular factorization is said to break down. However even in such a case, we can allow the computation in Figure 3 to proceed by including $\pm\infty$ in the arithmetic. We elaborate on this in the next section.

THEOREM 5.7. *Let J be an unreduced (or nonsingular) tridiagonal matrix of order n, and let D_+ and D_- be the diagonal matrices as computed by the algorithms of Figure* 3. *Then $J^{1:i}$ is singular if and only if $D_+(i) = 0$ while $J^{i:n}$ is singular if and only if $D_-(i) = 0$.*

Proof. This follows from Lemma 5.5 and the fact that

$$\det(J^{1:i-1}) \cdot D_+(i) \;=\; \det(J^{1:i}) \quad \text{and} \quad \det(J^{i+1:n}) \cdot D_-(i) \;=\; \det(J^{i:n}).$$

Note that due to Theorem 5.6, the above formulae hold even when triangular factorization "breaks down" before the computation of $D_+(i)$ or $D_-(i)$. $\square$

Finally we give an alternate formula for computing the diagonal elements of J^{-1}. Other formulae that are computationally better than (5.6) may be found in Corollary 4 of [23].

THEOREM 5.8. *Let J be a nonsingular tridiagonal matrix of order n that permits the factorizations in* (4.4) *and* (4.5). *Then $\Delta_i \equiv (J^{-1})_{ii}$ may be computed as*

$$\frac{1}{\Delta_i} = D_+(i) + D_-(i) - J_{ii}. \tag{5.6}$$

Proof. See Theorem 2 and Corollary 3 of [23]. $\square$

6. Eliminating the assumptions. In this section, we extend the algorithm outlined in section 4 to handle breakdown of triangular factorization. The theory developed in the previous section leads to these extensions.

Triangular factorizations are said to fail, or not exist, if a zero "pivot," $D_+(i)$ or $D_-(i)$, is encountered prematurely. However one of the attractions of an unreduced tridiagonal matrix is that the damage done by a zero pivot is localized. Indeed if $\pm\infty$ is added to the number system, triangular factorization cannot break down and the algorithms in Figure 3 always map J into unique L_+, D_+, U_+ and U_-, D_-, L_-. There is no need to spoil the inner loop with tests. It may no longer be true that $J = L_+D_+U_+$ or $J = U_-D_-L_-$, but equality does hold for all entries except for those at or adjacent to any infinite pivot. The IEEE arithmetic standard [2] allows such computation to proceed without breakdown, and thus we do not have to worry about zero pivots. Expressions with $\pm\infty$ are not expensive to handle if done by the hardware; see [11] for a discussion.

If $\Delta_i = 0$, i.e., $x_i = 0$ or $y_i = 0$, then equation (4.1) or (4.11) cannot be used to compute Δ_{i+1} even in exact arithmetic. Similarly $s_u(i+1)$ cannot be computed

by (4.2) or (4.12) if $s_u(i) = 0$. We now derive alternate formulae to compute Δ_{i+1} and $s_u(i+1)$ in such cases.

If $J^{1:i-1}$ is singular, i.e., $D_+(i-1) = 0$, then by (5.1) and (5.4),

$$\Delta_{i+1} \;=\; \frac{\det(J^{1:i})\det(J^{i+2:n})}{\det(J)} \;=\; \frac{\det(J^{1:i})\det(J^{i+2:n})}{\det(J^{1:i})\det(J^{i+1:n})} \;=\; \frac{1}{D_-(i+1)}, \tag{6.1}$$

and this gives a formula to compute Δ_{i+1} when the leading submatrix $J^{1:i-1}$ is singular.

Similarly if $J^{i+1:n}$ is singular, i.e., $D_-(i+1) = 0$, then by (5.1) and (5.5),

$$\Delta_{i+1} \;=\; \frac{\det(J^{1:i})\det(J^{i+2:n})}{\det(J^{1:i-1})\det(J^{i:n})} \;=\; \frac{-D_+(i)}{b_i c_i}. \tag{6.2}$$

If $J^{i+1:n}$ is singular, then y_i and $s_u(i)$ equal zero. In this case, since y_i and y_{i-1} cannot both be zero by Theorem 5.4, $s_u(i+1)$ may be computed from $s_u(i-1)$ as follows:

$$s_u(i+1) \;=\; (s_u(i-1) + |\Delta_{i-1}|)\,\frac{|y_{i+1}|}{|y_{i-1}|} + |(J^{-1})_{i,i+1}|.$$

We now simplify the above recurrence. Consider the ith equation of $J^T(x_1 y) = e_1$ when $y_i = 0$, $i \neq 1$,

$$c_{i-1}y_{i-1} + a_i y_i + b_i y_{i+1} = 0$$

$$\Rightarrow \quad \frac{y_{i+1}}{y_{i-1}} \;=\; \frac{-c_{i-1}}{b_i}.$$

Since we are considering the case when $J^{i+1:n}$ is singular, (5.3) and (5.5) imply that

$$(J^{-1})_{i,i+1} \;=\; \frac{-c_i\det(J^{1:i-1})\det(J^{i+2:n})}{\det(J)} \;=\; \frac{-c_i\det(J^{1:i-1})\det(J^{i+2:n})}{\det(J^{1:i-1})\det(J^{i:n})} \;=\; \frac{1}{b_i}.$$

Thus when $J^{i+1:n}$ is singular, $s_u(i+1)$ may be computed as

$$s_u(i+1) \;=\; (s_u(i-1) + |\Delta_{i-1}|)\,\frac{|c_{i-1}|}{|b_i|} + \frac{1}{|b_i|}. \tag{6.3}$$

$s_l(i-1)$ may similarly be computed as follows from $s_l(i+1)$ when $J^{1:i-1}$ is singular:

$$s_l(i-1) \;=\; (s_l(i+1) + |\Delta_{i+1}|)\,\frac{|b_i|}{|c_{i-1}|} + \frac{1}{|c_{i-1}|}. \tag{6.4}$$

Equations (6.1), (6.2), (6.3), and (6.4) give formulae for computing Δ_i, $s_u(i)$, and $s_l(i)$ when leading or trailing principal submatrices are exactly singular. By combining these formulae with Algorithm NRMINV of Figure 4, we get Algorithm NRMINV_NOASSUMP that is given in Figure 5. In exact arithmetic, this algorithm correctly computes the condition number of the matrix mentioned at the beginning of section 5 with $a_i = 0$, $b_i = 1$, and n even. In finite precision arithmetic, we might suspect that this algorithm breaks down when a pivot, $D_+(i)$ or $D_-(i)$, is tiny but not exactly zero. We address such issues in section 8. We now do a roundoff error analysis of our new algorithms assuming no over/underflow and indicate why they are accurate.

Algorithm NRMINV_NOASSUMP
Compute $J = L_+D_+U_+$ and $J = U_-D_-L_-$ (see Figure 3).
Set $D_+(0) = D_-(n+1) = 1$.
if $(D_+(n) = 0$ or $D_-(1) = 0)$ then $\|J^{-1}\|_1 = \infty$; return;
for $i = 2$ to $n-1$
    if $(D_+(i-1) = 0$ and $D_-(i+1) = 0)$ then $\|J^{-1}\|_1 = \infty$; return;
$\Delta_1 = 1/D_-(1)$;
for $i = 1$ to $n-1$
    if $(D_+(i) = 0$ or $D_-(i+2) = 0)$ then $\Delta_{i+1} = 0$;
    elseif $(D_-(i+1) = 0)$ then $\Delta_{i+1} = -D_+(i)/b_ic_i$;
    elseif $(D_+(i-1) = 0)$ then $\Delta_{i+1} = 1/D_-(i+1)$;
    else $\Delta_{i+1} = \Delta_i * \frac{D_+(i)}{D_-(i+1)}$;
$s_u(1) = 0$;
for $i = 1$ to $n-1$
    if $(D_-(i+2) = 0)$ then $s_u(i+1) = 0$;
    elseif $(D_-(i+1) = 0)$ then $s_u(i+1) = (s_u(i-1) + |\Delta_{i-1}|) * |\frac{c_{i-1}}{b_i}| + |\frac{1}{b_i}|$;
    else $s_u(i+1) = (s_u(i) + |\Delta_i|) * |U_-(i)|$;
$s_l(n) = 0$;
for $i = n$ to 2 step -1
    if $(D_+(i-2) = 0)$ then $s_l(i-1) = 0$;
    elseif $(D_+(i-1) = 0)$ then $s_l(i-1) = (s_l(i+1) + |\Delta_{i+1}|) * |\frac{b_i}{c_{i-1}}| + |\frac{1}{c_{i-1}}|$;
    else $s_l(i-1) = (s_l(i) + |\Delta_i|) * |L_+(i-1)|$;
$\gamma = 0$;
for $i = 1$ to n
    $\gamma = \max(\gamma, s_u(i) + s_l(i) + |\Delta_i|)$;
$\|J^{-1}\|_1 = \gamma$;

FIG. 5. *Algorithm* NRMINV_NOASSUMP *computes* $\|J^{-1}\|_1$.

7. Roundoff error analysis. We consider Algorithm NRMINV under the assumption that triangular factorization does not break down. Our model of arithmetic is that the floating point result of a basic arithmetic operation $\circ$ satisfies

$$fl(x \circ y) = (x \circ y)(1+\eta) = (x \circ y)/(1+\delta),$$

where η and δ depend on x, y, $\circ$, and the arithmetic unit but satisfy

$$|\eta| \le \varepsilon, \quad |\delta| \le \varepsilon$$

for a given ε, the latter depending only on the arithmetic unit. We shall choose freely the form (η or δ) that suits the analysis. We also adopt the convention of denoting the computed value of x by $\hat{x}$.

We now show that the computed triangular factorizations (4.4) and (4.5) are almost exact for a slightly perturbed matrix $J+\delta J$. In particular, we show that the pivots computed by the algorithms in Figure 3, $\hat{D}_+(i)$, are small relative perturbations of quantities $\widehat{D}_+(i)$ that are exact pivots for $J+\delta J_+$, where δJ_+ represents a small componentwise perturbation in the off-diagonal elements of J. Since $U_+(i) = c_i/D_+(i)$ and $L_+(i) = b_i/D_+(i)$, $\hat{U}_+(i)$ and $\hat{L}_+(i)$ can similarly be related to quantities $\widehat{U}_+(i)$

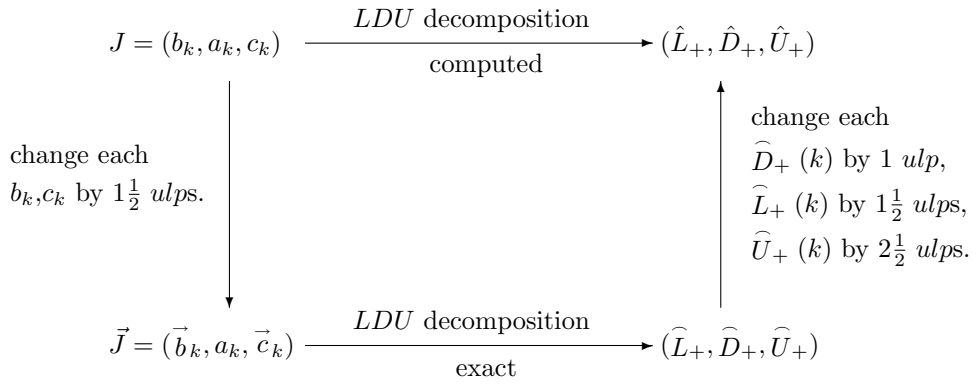

FIG. 6. *Effects of roundoff.*

and $\widehat{L}_+(i)$ that are exact for $J + \delta J_+$. An analogous result holds for the factorization $J = U_- D_- L_-$. The exact result we prove is summarized in Figure 6, where the acronym *ulp* stands for *u*nits in the *l* ast *p*lace held. It is the natural way to refer to *relative* differences between numbers. When a result is correctly rounded the error is not more than half an *ulp*.

THEOREM 7.1. *Let $J = (b_k, a_k, c_k)$ denote the tridiagonal matrix in* (2.1). *Let its LDU and UDL decompositions be computed as in Figure* 3. *In the absence of overflow and underflow, the diagram in Figure* 6 *commutes, and, for each k, $\hat{D}_+(k)$ differs from $\widehat{D}_+(k)$ by* 1 *ulp, $\hat{L}_+(k)$, $\hat{U}_+(k)$ differ from $\widehat{L}_+(k)$, $\widehat{U}_+(k)$ by $1\frac{1}{2}$ and $2\frac{1}{2}$ ulps, respectively, while $\vec{b}_k$, $\vec{c}_k$ differ from b_k, c_k by $1\frac{1}{2}$ ulps each. A similar result holds for the UDL factorization.*

Proof. We write down the exact equations satisfied by the computed quantities:

$$\hat{L}_+(k-1) = \frac{b_{k-1}}{\hat{D}_+(k-1)}(1+\varepsilon_/),$$

$$\hat{U}_+(k-1) = \frac{c_{k-1}}{\hat{D}_+(k-1)}(1+\varepsilon_{//}),$$

$$\hat{D}_+(k) = \left(a_k - c_{k-1}\hat{L}_+(k-1)\cdot(1+\varepsilon_*)\right)/(1+\varepsilon_k),$$

$$\Rightarrow \quad (1+\varepsilon_k)\hat{D}_+(k) = a_k - \frac{b_{k-1}c_{k-1}}{\hat{D}_+(k-1)}(1+\varepsilon_*)(1+\varepsilon_/). \tag{7.1}$$

In the above, all the ε depend on k but we have chosen to single out the one that accounts for the subtraction as it is the only one where the dependence on k must be made explicit. We now introduce the quantities

$$\widehat{D}_+(k) = \hat{D}_+(k)(1+\varepsilon_k), \tag{7.2}$$

$$\vec{b}_{k-1} = b_{k-1}\sqrt{(1+\varepsilon_*)(1+\varepsilon_/)(1+\varepsilon_{k-1})}, \tag{7.3}$$

$$\vec{c}_{k-1} = c_{k-1}\sqrt{(1+\varepsilon_*)(1+\varepsilon_/)(1+\varepsilon_{k-1})}. \tag{7.4}$$

Substituting (7.2), (7.3), and (7.4) in (7.1), we see that $\widehat{D}_+(k)$ is exact for $\vec{J} = [\vec{b}_k, a_k, \vec{c}_k]$, i.e.,

$$\widehat{D}_+(k) = a_k - \frac{\vec{b}_{k-1}\vec{c}_{k-1}}{\widehat{D}_+(k-1)}.$$

To satisfy the exact mathematical relations

$$\widehat{L}_+(k) = \frac{\vec{b}_k}{\widehat{D}_+(k)}, \quad \widehat{U}_+(k) = \frac{\vec{c}_k}{\widehat{D}_+(k)},$$

we set

$$\widehat{L}_+(k) = \hat{L}_+(k)\sqrt{\frac{1+\varepsilon_*}{(1+\varepsilon_k)(1+\varepsilon_/)}},$$

$$\widehat{U}_+(k) = \hat{U}_+(k)\frac{1}{1+\varepsilon_{//}}\sqrt{\frac{(1+\varepsilon_*)(1+\varepsilon_/)}{1+\varepsilon_k}},$$

and the result holds. The result for the factorization $J = U_- D_- L_-$ is similarly proved. □

The observant reader would have noted that the above is not a pure backward error analysis. We have put small perturbations not only on the input but also on the output. This property is called mixed stability in [7], but note that our perturbations are relative ones.

It is important to note that the backward perturbations for the LDU factorization differ from the ones for the UDL factorization. By (4.11), Δ_i is formed by a ratio of $D_+(i)$ and $D_-(i+1)$. Since this mixes the LDU and UDL decompositions, the roundoff error analysis given above *does not* enable us to relate the computed value of all the Δ_i to a *single* perturbed tridiagonal matrix. However if small relative changes to the off-diagonal entries of J lead to "small" changes in its LDU and UDL factorizations, then Theorem 7.1 implies that Algorithm NRMINV "accurately" computes the condition number of J. The latter implication is easily seen to be true by observing that the quantities Δ_i, $s_u(i)$, $s_l(i)$ are computed from the LDU and UDL factorizations by multiplications, divisions, and additions of nonnegative numbers. The case of Algorithm NRMINV_NOASSUMP is similar.

Often the triangular factorizations (4.4) and (4.5) can be very sensitive to small changes in the entries of the tridiagonal matrix. These are precisely the situations when a submatrix of J is close to being singular and there is element growth in the factorizations. Thus we may suspect that our algorithm delivers inaccurate answers in such cases. However numerical experience, given in section 10, indicates that the condition number is computed accurately despite element growth. It is an open problem to explain this phenomenon. We feel the situation is somewhat similar to Algorithms HIGHAM_1 and HIGHAM_2 that were outlined in section 3. In [17], Higham observes that when the latter algorithms do not over/underflow their answers are very accurate, but no error analysis has been able to explain this accuracy. One approach to proving accuracy of our algorithm may be to relate both sets of pivots, D_+ and D_-, to a *single perturbed matrix.*

8. Handling overflow and underflow. Algorithm NRMINV_NOASSUMP also suffers from the limited range of numbers that can be represented in a digital computer. Consider the matrix

$$J \;=\; \begin{bmatrix} 1000 & 100 \\ 100 & 10^{-306} \end{bmatrix}.$$

In its UDL decomposition, $D_-(2) = 10^{-306}$ while $D_-(1)$ is computed as

$$D_-(1) = 1000 - \frac{10^4}{10^{-306}} = 1000 - 10^{310}.$$

In IEEE double precision arithmetic, the above value overflows and $D_-(1)$ is set to $-\infty$ [2]. Since $\Delta_1 = 1/D_-(1)$, it is computed to be 0 by Algorithm NRMINV_NOASSUMP. Δ_2 is then computed as

$$\Delta_2 \;=\; \Delta_1 \cdot \left(\frac{D_+(1)}{D_-(2)}\right) \;=\; 0 \cdot \left(\frac{1000}{10^{-306}}\right).$$

Again the value $1000/10^{-306}$ overflows and Δ_2 is set to $0 \cdot \infty =$ Not a Number (NaN). Note that J is perfectly well conditioned with $\Delta_2 = -0.1$, and

$$J^{-1} \;\approx\; \begin{bmatrix} -10^{-310} & 0.01 \\ 0.01 & -0.1 \end{bmatrix}.$$

Thus Algorithm NRMINV_NOASSUMP malfunctions due to overflow problems. Underflow in computing Δ_i by (4.11) can cause similar problems.

We now show how to overcome such overflow and underflow. Before doing so we emphasize that the above over/underflow problems are not as severe as those in the algorithms of [17]. The discerning reader would have noticed that problems in the earlier algorithms are inevitable due to the explicit computation of the vectors x, y, p, and q; see section 3 for more details.

There are two problems that we must address. The first is to avoid NaNs in the computation. A NaN results when evaluating expressions such as $0 \cdot \infty$, $\frac{0}{0}$, and $\frac{\infty}{\infty}$. Algorithm NRMINV_FINAL1 given in Figure 7 prevents the formation of NaNs by explicitly avoiding such expressions and handling separately the special cases when $D_+(i)$ or $D_-(i)$ equals 0 or ∞.

The second difficulty occurs if Δ_i overflows or underflows to 0 when computed as

$$\Delta_i \;=\; \Delta_{i-1}\frac{D_+(i-1)}{D_-(i)}.$$

It is incorrect to use such a Δ_i to compute Δ_{i+1} by the above recurrence. We solve this problem by computing Δ_{i+1} as

$$\Delta_{i+1} \;=\; 1/\left(D_-(i+1) - \frac{b_i c_i}{D_+(i)}\right) \tag{8.1}$$

in such a case. The above formula is a consequence of Theorem 5.8. Note that (8.1) leads to the correct value of Δ_{i+1} when $D_+(i-1) = 0$ or $D_-(i+1) = 0$; see (6.1) and (6.2).

Thus Algorithm NRMINV_FINAL1 tries to cure the over/underflow problems, and we have found its computer implementation to be accurate on all tridiagonal matrices

Algorithm NRMINV_FINAL1
Compute $J = L_+D_+U_+$ and $J = U_-D_-L_-$ as follows:
$D_+(1) = a_1$;
for $i = 1$ to $n-1$
    if ($D_+(i) = 0$ and $b_ic_i = 0$) then $\|J^{-1}\|_1 = \infty$; return;
    else $L_+(i) = b_i/D_+(i)$; $D_+(i+1) = a_{i+1} - c_i * L_+(i)$;
$D_-(n) = a_n$;
for $i = n-1$ to 1 step -1
    if ($D_-(i+1) = 0$ and $b_ic_i = 0$) then $\|J^{-1}\|_1 = \infty$; return;
    else $U_-(i) = c_i/D_-(i+1)$; $D_-(i) = a_i - b_i * U_-(i)$;
if ($D_+(n) = 0$ or $D_-(1) = 0$) then $\|J^{-1}\|_1 = \infty$; return;
for $i = 2$ to $n-1$
    if ($D_+(i-1) = 0$ and $D_-(i+1) = 0$) then $\|J^{-1}\|_1 = \infty$; return;
$\Delta_1 = 1/D_-(1)$;
for $i = 1$ to $n-1$
    if ($D_+(i) = 0$ or $1/D_-(i+1) = 0$) then $\Delta_{i+1} = 0$;
    elseif ($D_-(i+1) = 0$) then $\Delta_{i+1} = -D_+(i)/b_ic_i$;
    elseif ($1/D_+(i) = 0$) then $\Delta_{i+1} = 1/D_-(i+1)$;
    elseif ($\Delta_i = 0$) then $\Delta_{i+1} = 1/\left(D_-(i+1) - \frac{b_ic_i}{D_+(i)}\right)$;
    else $\Delta_{i+1} = \frac{\Delta_i}{D_-(i+1)} * D_+(i)$;
    if ($1/\Delta_{i+1} = 0$) then $\|J^{-1}\|_1 = \infty$; return;
$s_u(1) = 0$;
for $i = 1$ to $n-1$
    if ($s_u(i) + |\Delta_i| = 0$) then
        $s_u(i+1) = (s_u(i-1) + |\Delta_{i-1}|) * |\frac{c_{i-1}}{b_i}| + |\frac{1}{b_i}|$;
    else $s_u(i+1) = (s_u(i) + |\Delta_i|) * |U_-(i)|$;
    if ($1/s_u(i+1) = 0$) then $\|J^{-1}\|_1 = \infty$; return;
$s_l(n) = 0$;
for $i = n$ to 2 step -1
    if ($s_l(i+1) + |\Delta_{i+1}| = 0$) then
        $s_l(i-1) = (s_l(i+1) + |\Delta_{i+1}|) * |\frac{b_i}{c_{i-1}}| + |\frac{1}{c_{i-1}}|$;
    else $s_l(i-1) = (s_l(i) + |\Delta_i|) * |L_+(i-1)|$;
    if ($1/s_l(i-1) = 0$) then $\|J^{-1}\|_1 = \infty$; return;
$\gamma = 0$;
for $i = 1$ to n
    if ($s_u(i) + s_l(i) + |\Delta_i| > \gamma$) then
        $\gamma = s_u(i) + s_l(i) + |\Delta_i|$;
$\|J^{-1}\|_1 = \gamma$;

FIG. 7. *Algorithm* NRMINV_FINAL1 *computes* $\|J^{-1}\|_1$.

in our test-bed. Numerical results to show this are presented in the next section. In addition, this algorithm also works for tridiagonal matrices that are not unreduced, i.e., where some of the off-diagonal entries may be zero. None of the elaborate techniques used in [16, 17] are needed to handle this special case. As written, the algorithm requires IEEE arithmetic but it is easily modified to prevent overflow.

In spite of the above precautions, Algorithm NRMINV_FINAL1 can march danger-

ously close to the overflow and underflow thresholds. When a pivot element $D_+(i)$ or $D_-(i)$ is tiny, intermediate quantities can vary widely in magnitude while computing $s_u(i)$ and $s_l(i)$ by (4.12) and (4.13). We now present an alternate algorithm that tries to avoid large intermediate numbers. To avoid division by the tiny pivot $D_-(i+1)$ in (4.12), we may write $s_u(i+1)$ in terms of $s_u(i-1)$ as follows:

$$\begin{aligned} s_u(i+1) &= (s_u(i) + |\Delta_i|) \left| \frac{c_i}{D_-(i+1)} \right| \\ &= (s_u(i-1) + |\Delta_{i-1}|) \left| \frac{c_{i-1}c_i}{D_-(i)D_-(i+1)} \right| + \left| \frac{\Delta_i c_i}{D_-(i+1)} \right|. \end{aligned} \tag{8.2}$$

Now, the formula for computing $D_-(i)$ (see Figure 3) implies that

$$D_-(i+1)D_-(i) = D_-(i+1)(a_i - b_i c_i / D_-(i+1)) = D_-(i+1)a_i - b_i c_i, \tag{8.3}$$

and using (5.6),

$$\frac{\Delta_i c_i}{D_-(i+1)} \;=\; \frac{c_i}{D_-(i+1)D_+(i) - b_i c_i}. \tag{8.4}$$

Substitution of (8.3) and (8.4) in (8.2) leads to the desired formula

$$s_u(i+1) = (s_u(i-1) + |\Delta_{i-1}|) \left| \frac{c_{i-1}c_i}{D_-(i+1)a_i - b_i c_i} \right| + \left| \frac{c_i}{D_-(i+1)D_+(i) - b_i c_i} \right|. \tag{8.5}$$

Unlike (4.12), the above formula does not involve division by the tiny pivot element $D_-(i+1)$. Thus no large intermediate quantities are formed. Similarly, $s_l(i-1)$ may be expressed in terms of $s_l(i+1)$ to avoid division by a small $D_+(i-1)$. Note that in the extreme case when $D_-(i+1) = 0$, (8.5) simplifies to (6.3). Equation (8.5) can alternatively be obtained by taking the 2×2 matrix

$$\begin{bmatrix} a_i & c_i \\ b_i & D_-(i+1) \end{bmatrix}$$

as a pivot in block Gaussian Elimination (instead of $D_-(i+1)$) and using the corresponding block $U_- D_- L_-$ factorization to compute $s_u(i+1)$. When $D_-(i+1)$ is tiny, it can be shown that using this 2×2 pivot prevents element growth unless J is nearly singular. Algorithm NRMINV_FINAL2 given in Figure 8 uses such a pivot strategy to compute $s_u(i)$ and $s_l(i)$. Also note that in Algorithm NRMINV_FINAL2 we use (5.6) instead of (4.11) to compute Δ_i.

Although Algorithm NRMINV_FINAL2 tends to have less element growth in its computation, it is not clear whether it is more accurate than Algorithm NRMINV_FINAL1. Numerical experience, given in section 10, indicates that both these algorithms are accurate. Our personal preference is for Algorithm NRMINV_FINAL2 since the intermediate quantities computed by it do not vary widely in scale.

9. Another application. In computing $\|J^{-1}\|$, we need to find the column of J^{-1} with the largest 1-norm. We now briefly mention another application where we may need to identify such a column.

Given a real, symmetric tridiagonal matrix T and an accurate approximation to an eigenvalue $\hat{\lambda}$, we can attempt to find the corresponding eigenvector by solving

$$(T - \hat{\lambda} I) z_k = e_k,$$

where e_k is the kth column of the identity matrix (the above may also be thought

Algorithm NRMINV_FINAL2
Compute $J = L_+D_+U_+$ and $J = U_-D_-L_-$ as follows:
$D_+(1) = a_1$;
for $i = 1$ to $n-1$
    if ($D_+(i) = 0$ and $b_ic_i = 0$) then $\|J^{-1}\|_1 = \infty$; return;
    else $L_+(i) = b_i/D_+(i)$; $D_+(i+1) = a_{i+1} - c_i * L_+(i)$;
$D_-(n) = a_n$;
for $i = n-1$ to 1 step -1
    if ($D_-(i+1) = 0$ and $b_ic_i = 0$) then $\|J^{-1}\|_1 = \infty$; return;
    else $U_-(i) = c_i/D_-(i+1)$; $D_-(i) = a_i - b_i * U_-(i)$;
if ($D_+(n) = 0$ or $D_-(1) = 0$) then $\|J^{-1}\|_1 = \infty$; return;
for $i = 2$ to $n-1$
    if ($D_+(i-1) = 0$ and $D_-(i+1) = 0$) then $\|J^{-1}\|_1 = \infty$; return;
$\Delta_1 = 1/D_-(1)$;
for $i = 1$ to $n-1$
    $\Delta_{i+1} = 1/(D_-(i+1) - \frac{b_ic_i}{D_+(i)})$;
    if ($1/\Delta_{i+1} = 0$) then $\|J^{-1}\|_1 = \infty$; return;
$s_u(1) = 0$;
for $i = 1$ to $n-1$
    $\mathrm{DET} = D_-(i+1)a_i - b_ic_i$;
    if ($1/D_-(i+1) = 0$ or $|D_-(i+1) \cdot a_i| \geq |\mathrm{DET}|$) then
        $s_u(i+1) = (s_u(i) + |\Delta_i|) * |U_-(i)|$;
    else
        $s_u(i+1) = (s_u(i-1) + |\Delta_{i-1}|) * |\frac{c_{i-1}c_i}{\mathrm{DET}}| + |\frac{c_i}{|D_-(i+1)D_+(i) - b_ic_i}|$;
    if ($1/s_u(i+1) = 0$) then $\|J^{-1}\|_1 = \infty$; return;
$s_l(n) = 0$;
for $i = n$ to 2 step -1
    $\mathrm{DET} = D_+(i-1)a_i - b_{i-1}c_{i-1}$;
    if ($1/D_+(i-1) = 0$ or $|D_+(i-1) \cdot a_i| \geq |\mathrm{DET}|$) then
        $s_l(i-1) = (s_l(i) + |\Delta_i|) * |L_+(i-1)|$;
    else
        $s_l(i-1) = (s_l(i+1) + |\Delta_{i+1}|) * |\frac{b_{i-1}b_i}{\mathrm{DET}}| + |\frac{b_{i-1}}{D_+(i-1)D_-(i) - b_{i-1}c_{i-1}}|$;
    if ($1/s_l(i-1) = 0$) then $\|J^{-1}\|_1 = \infty$; return;
$\gamma = 0$;
for $i = 1$ to n
    if ($s_u(i) + s_l(i) + |\Delta_i| > \gamma$) then
        $\gamma = s_u(i) + s_l(i) + |\Delta_i|$;
$\|J^{-1}\|_1 = \gamma$;

FIG. 8. *Algorithm* NRMINV_FINAL2 *computes* $\|J^{-1}\|_1$.

of as the first step of inverse iteration with e_k as the starting vector). However an arbitrary choice of k does not always work, as observed by Wilkinson in [25, 26]. Note that the pair $(\hat{\lambda}, z_k)$ has the residual norm

$$\frac{\|(T - \hat{\lambda}I)z_k\|}{\|z_k\|} = \frac{1}{\|(T - \hat{\lambda}I)^{-1}e_k\|}, \tag{9.1}$$

TABLE 2
Test matrices.

Matrix Type	Description
1	Nonsymmetric random tridiagonal — each element is uniformly distributed in the interval $[-1, 1]$.
2	Symmetric tridiagonal $J = Q^T DQ$ with Q random orthogonal and D diagonal with one element equal to 1 and all others equal to ϵ.
3	Symmetric tridiagonal $J = Q^T DQ$ with Q random orthogonal and D diagonal with one element equal to ϵ and all others equal to 1.
4	Symmetric tridiagonal $J = Q^T DQ$ with Q random orthogonal and D diagonal with elements geometrically distributed from ϵ to 1.
5	Symmetric tridiagonal $J = Q^T DQ$ with Q random orthogonal and D diagonal with elements uniformly distributed from ϵ to 1.
6	Symmetric Toeplitz tridiagonal with $a_i = 64$, $b_i = c_i = 1$.
7	Symmetric Toeplitz tridiagonal with $a_i = 10^8$, $b_i = c_i = 1$.
8	Symmetric Toeplitz tridiagonal with $a_i = 0$, $b_i = c_i = 1$.
9	Nonsymmetric random tridiagonal as in Type 1 but with some off-diagonals set to zero.

TABLE 3
Computation of $\kappa(J) = \|J\|_1 \cdot \|J^{-1}\|_1$ *on matrices of order* 41.

Matrix Type	$\kappa(J)$ computed by				
	Algorithm NRMINV_FINAL1	Algorithm NRMINV_FINAL2	Algorithm HIGHAM1	Algorithm HIGHAM2	LAPACK's condition estimator (DGTCON)
1	111.9	111.9	111.9	111.9	109.7
2	$7.5 \cdot 10^{15}$	$7.6 \cdot 10^{15}$	NaN	NaN	$7.6 \cdot 10^{15}$
3	$5.4 \cdot 10^{15}$	$5.4 \cdot 10^{15}$	NaN	NaN	$5.4 \cdot 10^{15}$
4	$6.3 \cdot 10^{15}$	$6.3 \cdot 10^{15}$	$6.3 \cdot 10^{15}$	$6.3 \cdot 10^{15}$	$6.3 \cdot 10^{15}$
5	$7.6 \cdot 10^{15}$	$7.7 \cdot 10^{15}$	$7.7 \cdot 10^{15}$	$7.6 \cdot 10^{15}$	$7.7 \cdot 10^{15}$
6	1.06	1.06	1.06	1.06	1.06
7	1.0	1.0	NaN	NaN	1.0
8	∞	∞	NaN	∞	∞
9	$1.3 \cdot 10^3$	$1.3 \cdot 10^3$	NaN	NaN	$1.3 \cdot 10^3$

where we assume that $\hat{\lambda}$ is not an exact eigenvalue of T. The goal is to obtain a small residual norm, but an arbitrary choice of k fails because not every column of $(T - \hat{\lambda}I)^{-1}$ is large in magnitude. However when $\hat{\lambda}$ is close to an eigenvalue, there must exist a column k of $(T - \hat{\lambda}I)^{-1}$ that has a large norm. The corresponding pair $(\hat{\lambda}, z_k)$ has a small residual norm, and it can be shown that z_k is close to an eigenvector. The optimal choice of k minimizes the residual norm (9.1), i.e., it maximizes $\|(T - \hat{\lambda}I)^{-1}e_k\|$. Thus the algorithms discussed earlier in the paper provide a solution to this problem in $O(n)$ time. Algorithms NRMINV_FINAL1 and NRMINV_FINAL2 are easily modified to give the solution when the 2-norm is considered.

Often when the corresponding eigenvalue is sufficiently isolated, it suffices to choose k such that the (k, k) entry of $(T - \hat{\lambda}I)^{-1}$ has the largest absolute value among all diagonal elements of the inverse. For more on this problem, the interested reader is referred to [14, 23] and [12, Chapter 3]. As a way to find an optimal k, Jesse Barlow [5] also independently discovered recurrences similar to (4.11), (4.12), and (4.13).

10. Numerical results. In this section, we present numerical results of our new algorithms and compare them with existing algorithms. A variety of tridiagonal matrices listed in Table 2 forms our test-bed. The matrices of type 2–5 were obtained by Householder reduction of a random dense symmetric matrix that had the desired spectrum. See [8] for more on the generation of such matrices.

TABLE 4
Computation of $\kappa(J) = \|J\|_1 \cdot \|J^{-1}\|_1$ on matrices of order 200.

Matrix Type	$\kappa(J)$ computed by				
	Algorithm NRMINV_FINAL1	Algorithm NRMINV_FINAL2	Algorithm HIGHAM1	Algorithm HIGHAM2	LAPACK's condition estimator (DGTCON)
1	$1.9 \cdot 10^3$	$1.9 \cdot 10^3$	$1.9 \cdot 10^3$	$1.9 \cdot 10^3$	$1.2 \cdot 10^3$
2	$7.4 \cdot 10^{15}$	$7.5 \cdot 10^{15}$	NaN	NaN	$7.4 \cdot 10^{15}$
3	$4.5 \cdot 10^{15}$	$4.5 \cdot 10^{15}$	NaN	NaN	$4.5 \cdot 10^{15}$
4	$9.9 \cdot 10^{15}$	$9.9 \cdot 10^{15}$	$9.9 \cdot 10^{15}$	$9.9 \cdot 10^{15}$	$9.9 \cdot 10^{15}$
5	$1.2 \cdot 10^{16}$	$1.2 \cdot 10^{16}$	$1.2 \cdot 10^{16}$	$1.2 \cdot 10^{16}$	$1.2 \cdot 10^{16}$
6	1.06	1.06	1.06	1.06	1.06
7	1.0	1.0	NaN	NaN	1.0
8	200.0	200.0	200.0	200.0	2.0
9	$2.0 \cdot 10^3$	$2.0 \cdot 10^3$	NaN	NaN	$2.0 \cdot 10^3$

TABLE 5
Timing results.

Matrix Type	Time taken by LAPACK's DGTCON (in ms.)			Time(DGTCON) / Time(NRMINV_FINAL1)			Time(DGTCON) / Time(NRMINV_FINAL2)		
	$n = 41$	$n = 200$	$n = 1000$	$n = 41$	$n = 200$	$n = 1000$	$n = 41$	$n = 200$	$n = 1000$
1	0.2	1.1	5.4	2.0	1.6	1.6	1.0	1.6	1.6
2	0.3	1.1	5.4	3.0	1.6	1.7	3.0	1.6	1.6
3	0.2	1.1	5.4	2.0	1.8	1.6	1.0	1.6	1.5
4	0.2	1.1	5.5	2.0	1.8	1.7	1.0	1.6	1.6
5	0.3	1.1	5.5	3.0	1.6	1.7	1.5	1.6	1.6
6	0.3	1.6	7.2	3.0	2.3	2.2	1.5	2.0	1.8
7	0.3	1.5	6.8	3.0	2.5	2.1	3.0	1.9	1.7
8	0.0	0.9	4.6	1.0	1.5	1.5	1.0	1.5	1.5
9	0.3	1.1	5.3	1.5	1.6	1.6	3.0	2.2	1.9

The results given in Tables 3 and 4 support our claim that the algorithms in [17] are susceptible to severe overflow and underflow problems. However they produce accurate answers when they do not suffer from such problems. The new algorithms outlined in the previous section, Algorithm NRMINV_FINAL1 and Algorithm NRMINV_FINAL2, give accurate answers on all our test matrices. Both the algorithms appear to be comparable in accuracy. In our numerical results, we have also included the current algorithm in LAPACK that estimates the condition number of a tridiagonal matrix [19]. This algorithm is guaranteed to give a lower bound on the condition number, and extensive testing done in [19] indicates that its estimates are good approximations to the exact condition number in most cases. For all our test matrices, except one, the condition numbers are estimated accurately. The only exception is the Toeplitz matrix with 0 on the diagonals and 1 on the off-diagonals; see Table 4. This example is similar to the one given in [19, p. 386], and LAPACK's condition estimator underestimates its condition number by a factor of $n/2$ for $n = 200$.

In Table 5, we compare the times taken by our new algorithms with LAPACK's condition estimator. The latter also appears to take $O(n)$ time but our new algorithms are up to three times faster. These timing experiments were conducted on an IBM RS/6000 processor.

11. Conclusions. In this paper, we have given stable algorithms to compute the condition number of a tridiagonal matrix in $O(n)$ time. Algorithm NRMINV (see Figure 4) contains the main new ideas and forms the basis of Algorithms NRMINV_FINAL1 and NRMINV_FINAL2 (see Figures 7 and 8). The latter algorithms may be directly implemented to give reliable numerical software and do not suffer from the inherent over/underflow problems of the earlier algorithms presented in [17].

Acknowledgments. I would like to thank Professors B. N. Parlett, J. W. Demmel, and N. J. Higham for a careful reading of the manuscript and for many useful suggestions.

REFERENCES

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

[2] ANSI/IEEE, New York, *IEEE Standard for Binary Floating Point Arithmetic*, Std 754-1985 ed., 1985.

[3] Edgar Asplund, *Inverses of matrices* $\{a_{ij}\}$ *which satisfy* $a_{ij} = 0$ *for* $j > i + p$, Math. Scand., 7 (1959), pp. 57–60.

[4] S. O. Asplund, *Finite boundary value problems solved by Green's matrix*, Math. Scand., 7 (1959), pp. 49–56.

[5] J. Barlow, *Private communication,* Pennsylvania State University, University Park, PA, 1996.

[6] B. Bukhberger and G. A. Emel'yanenko, *Methods of inverting tridiagonal matrices*, Comput. Math. Math. Phys., 13 (1973), pp. 10–20.

[7] L. S. DeJong, *Towards a formal definition of numerical stability*, Numer. Math., 28 (1977), pp. 211–220.

[8] J. Demmel and A. McKenney, *A Test Matrix Generation Suite*, LAPACK Working Note #9, Technical report, Courant Institute, Computer Science Department, New York, 1989.

[9] J. W. Demmel, *The Complexity of Condition Estimation*, manuscript, University of California, Berkeley, CA, 1997.

[10] James W. Demmel, *Open Problems in Numerical Linear Algebra*, LAPACK Working Note 47, IMA Preprint Series #961, Institute for Mathematics and Its Applications, University of Minnesota, Minneapolis, MN, April 1992.

[11] James W. Demmel and Xiaoye Li, *Faster numerical algorithms via exception handling*, IEEE Trans. Comput., 43 (1994), pp. 983–992.

[12] I. S. Dhillon, *A New* $O(n^2)$ *Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, Ph.D. thesis, Computer Science Division, EECS Dept., University of California, Berkeley, CA, May 1997; also available as Computer Science Division Technical report UCB//CSD-97-971.

[13] I. S. Dhillon and B. N. Parlett, *Orthogonal Eigenvectors Without Gram-Schmidt*, 1997, manuscript.

[14] K. V. Fernando, *On computing an eigenvector of a tridiagonal matrix. Part* I: *Basic results*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 1013-1034.

[15] W. W. Hager, *Condition estimators*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 311–316.

[16] N. J. Higham, *Matrix condition numbers*, M.Sc. thesis, Dept. of Mathematics, University of Manchester, Manchester, England, 1983.

[17] N. J. Higham, *Efficient algorithms for computing the condition number of a tridiagonal matrix*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 150–165.

[18] N. J. Higham, *A survey of condition number estimation for triangular matrices*, SIAM Rev., 29 (1987), pp. 575–596.

[19] N. J. Higham, *FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation*, ACM Trans. Math. Software, 14 (1988), pp. 381–396.

[20] N. J. Higham, *Accuracy and stability of numerical algorithms*, SIAM, Philadelphia, PA, 1996.

[21] I. Ikebe, *On inverses of Hessenberg matrices*, Linear Algebra Appl., 24 (1979), pp. 93–97.

[22] G. Meurant, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 707–728.

[23] B. N. Parlett and I. S. Dhillon, *Fernando's solution to Wilkinson's problem: an application of double factorization*, Linear Algebra Appl., 267 (1997), pp. 247–279.

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

[25] J. H. Wilkinson, *The calculation of the eigenvectors of codiagonal matrices*, Computer J., 1 (1958), pp. 90–96.

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