

COMPUTING THE CONDITION NUMBER OF TRIDIAGONAL AND DIAGONAL-PLUS-SEMISEPARABLE MATRICES IN LINEAR TIME *

GARETH I. HARGREAVES[†]

Abstract. For an $n \times n$ tridiagonal matrix we exploit the structure of its QR factorization to devise two new algorithms for computing the 1-norm condition number in $O(n)$ operations. The algorithms avoid underflow and overflow, and are simpler than existing algorithms since tests are not required for degenerate cases. An error analysis of the first algorithm is given, while the second algorithm is shown to be competitive in speed with existing algorithms. We then turn our attention to an $n \times n$ diagonal-plus-semiseparable matrix, A , for which several algorithms have recently been developed to solve $Ax = b$ in $O(n)$ operations. We again exploit the QR factorization of the matrix to present an algorithm that computes the 1-norm condition number in $O(n)$ operations.

Key words. condition number, tridiagonal matrix, diagonal-plus-semiseparable matrix, QR factorization

AMS subject classifications. 15A09, 65F35, 15A12

1. Introduction. Consider a nonsingular matrix $A \in \mathbb{R}^{n \times n}$ and the linear system $Ax = b$. The condition number of A ,

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

is often computed or estimated since it provides a measure of the sensitivity of the solution to perturbations in A and b . The condition number depends on the choice of matrix norm. We will consider the matrix 1-norm

$$\|A\|_1 = \max_j \sum_i |a_{ij}|.$$

Various techniques exist for estimating the condition number of a general matrix in $O(n^2)$ operations, given a suitable factorization of the matrix.

Some matrices with a special structure allow the linear system to be solved in $O(n)$ operations rather than the $O(n^3)$ operations required for a general matrix. Techniques for estimating the condition number of such matrices typically reduce to $O(n)$ operations. However, the structure of the inverse may make it possible to compute the condition number exactly in $O(n)$ operations. Two types of matrices for which this can be achieved are tridiagonal and diagonal-plus-semiseparable matrices.

Tridiagonal matrices occur in many areas of numerical analysis. The lower (upper) triangular part of the inverse of the tridiagonal can be represented by the lower (upper) triangular part of the outer product of two vectors [14]. It is this compact representation that Higham [11] exploits to give several algorithms for computing the 1-norm of the inverse in $O(n)$ operations. However, the algorithms for a general tridiagonal matrix are prone to underflow and overflow. The algorithm for the positive definite case, which is implemented in LAPACK [1], is shown not to have such problems.

Dhillon [6] describes four algorithms based on LDU factorizations. The first of these is less prone to underflow and overflow than Higham's algorithms but

*Numerical Analysis Report 447, Manchester Centre for Computational Mathematics, April 2004; revised November 2004. This work was supported by an Engineering and Physical Sciences Research Council Ph.D. Studentship.

[†]School of Mathematics, University of Manchester, Manchester, M13 9PL, England (hargreaves@ma.man.ac.uk, <http://www.ma.man.ac.uk/~hargreaves/>).

has restrictions on the tridiagonal matrix. These restrictions are eliminated using IEEE arithmetic [13] in the second algorithm, and various tests to guard against underflow and overflow problems are added to give the third and fourth algorithms. The resulting codes are complicated as they contain various tests to deal with degenerate cases.

A semiseparable matrix is the sum of the strictly upper triangular part of a rank-1 matrix and the lower triangular part of another rank-1 matrix. Therefore the inverse of a tridiagonal matrix is semiseparable. Another link between tridiagonal and semiseparable matrices is that a symmetric matrix can be reduced to either of these forms using orthogonal transformations. This has led to new algorithms for solving the symmetric eigenvalue problem by reduction to semiseparable form [20] instead of tridiagonal form.

A diagonal-plus-semiseparable (dpss) matrix is the sum of a diagonal matrix and a semiseparable matrix. Recently several algorithms have been developed to solve $Ax = b$, where A is a dpss matrix, in $O(n)$ operations [2], [5], [7], [17]. Applications of solving linear systems of this form include boundary value problems for ordinary differential equations [9], [16], [18], integral equations [15] and orthogonal rational functions [3]. Unlike tridiagonal matrices, we know of no existing methods for computing the condition number of a dpss matrix exactly in $O(n)$ operations.

In Section 2 we use the properties of the QR factorization of a tridiagonal matrix to present two new algorithms for computing the 1-norm of the inverse of a tridiagonal matrix in $O(n)$ operations. The algorithms are more elegant and simpler than Dhillon's algorithms since they do not require tests for degenerate cases. Numerical experiments show that the second algorithm is marginally slower than the quickest of Dhillon's algorithms, but is faster than his recommended algorithm. A rounding error analysis of the first algorithm is given.

In Section 3 we extend the techniques developed for the tridiagonal case to the dpss case, to present an algorithm that computes the 1-norm of a dpss matrix in $O(n)$ operations. The condition number of a dpss matrix can instead be estimated in $O(n)$ operations by adapting the LAPACK [1] condition number estimator to take advantage of the structure of the dpss matrix; we show, however, that our new algorithm is quicker than the possibly inaccurate estimate.

2. Tridiagonal matrices. Let the tridiagonal matrix

$$(2.1) \quad T = \begin{bmatrix} \alpha_1 & \gamma_1 & & & & \\ \beta_1 & \alpha_2 & \gamma_2 & & & \\ & \beta_2 & \ddots & \ddots & & \\ & & \ddots & \alpha_{n-1} & \gamma_{n-1} & \\ & & & \beta_{n-1} & \alpha_n & \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

Since we can solve a tridiagonal system in $O(n)$ operations, we would also like to compute the condition number at the same cost. Computing $\|T\|_1$ can trivially be done in $O(n)$ operations so the problem remains to compute $\|T^{-1}\|_1$ in $O(n)$ operations.

We present two new algorithms for computing $\|T^{-1}\|_1$ matrix in $O(n)$ operations. The algorithms use the properties of QR factorizations of tridiagonal matrices and extend some of the ideas of Bini, Gemignani and Tisseur [4] for computing the trace of the inverse of a tridiagonal matrix. The new algorithms attempt to avoid underflow and overflow without the need for tests to deal with degenerate cases as in [6].

2.1. A new algorithm to compute $\|T^{-1}\|_1$. The QR factorization of T in (2.1) can be obtained using $n - 1$ Givens rotations, G_i , so that

$$(2.2) \quad Q^T T = R \quad \text{and} \quad Q^T = G_{n-1} \dots G_2 G_1,$$

where R is upper triangular. The Givens rotation G_i is equal to the identity except for rows and columns i and $i + 1$, where

$$(2.3) \quad G_i([i, i + 1], [i, i + 1]) = \begin{bmatrix} \phi_i & \psi_i \\ -\psi_i & \phi_i \end{bmatrix}, \quad \phi_i^2 + \psi_i^2 = 1.$$

Since T is tridiagonal, the upper triangular matrix R is of the form

$$R = \begin{bmatrix} r_1 & s_1 & t_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & r_{n-2} & s_{n-2} & t_{n-2} & \\ & & & r_{n-1} & s_{n-1} & \\ & & & & & r_n \end{bmatrix}.$$

The following theorem of [8] describes the structure of Q^T , which allows us to compute the elements of T^{-1} in $O(n)$ operations.

THEOREM 2.1. *Let $T \in \mathbb{R}^{n \times n}$ be tridiagonal and unreduced and let $T = QR$ be its QR factorization computed according to (2.2). Define*

$$(2.4) \quad \begin{aligned} D &= \text{diag}(1, -\psi_1, \psi_1\psi_2, \dots, (-1)^{n-1}\psi_1\psi_2 \cdots \psi_{n-1}), \\ u &= D^{-1}[1, \phi_1, \phi_2, \dots, \phi_{n-1}]^T, \\ v &= D[\phi_1, \phi_2, \dots, \phi_{n-1}, 1]^T. \end{aligned}$$

Then

$$(2.5) \quad Q^T = \begin{bmatrix} v_1 u_1 & \psi_1 & & & & 0 \\ v_2 u_1 & v_2 u_2 & \psi_2 & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ \vdots & & & & v_{n-1} u_{n-1} & \psi_{n-1} \\ v_n u_1 & v_n u_2 & \cdots & v_n u_{n-1} & v_n u_n & \end{bmatrix}.$$

From Theorem 2.1, the (i, j) element of T^{-1} , η_{ij} , is given by

$$\eta_{ij} = e_i^T T^{-1} e_j = e_i^T R^{-1} Q^T e_j = u_j e_i^T R^{-1} v, \quad i \geq j.$$

Defining w as the solution of $Rw = v$, we have

$$(2.6) \quad \eta_{ij} = u_j e_i^T w = u_j w_i, \quad i \geq j.$$

We can therefore find the elements of T^{-1} in the lower triangle using (2.6).

We note that we can obtain the strictly upper triangular part of T^{-1} by applying the above procedure to

$$\tilde{T} = \begin{bmatrix} \alpha_n & \beta_{n-1} & & & \\ \gamma_{n-1} & \alpha_{n-1} & \beta_{n-2} & & \\ & \gamma_{n-2} & \ddots & \ddots & \\ & & \ddots & \alpha_2 & \beta_1 \\ & & & \gamma_1 & \alpha_1 \end{bmatrix},$$

so that

$$\tilde{T}^{-1} = \begin{bmatrix} \eta_{nn} & & \cdots & \eta_{n1} \\ & \ddots & & \vdots \\ \vdots & & \eta_{22} & \eta_{21} \\ \eta_{1n} & \cdots & \eta_{12} & \eta_{11} \end{bmatrix}.$$

The QR factorization of \tilde{T} (which is essentially the QL factorization of T) is given by $\tilde{T} = \tilde{Q}\tilde{R}$, where \tilde{Q} is made up of $n - 1$ Givens rotations and \tilde{R} is upper triangular. By Theorem 2.1 the lower triangular part of \tilde{Q}^T is determined by vectors \tilde{u} and \tilde{v} , where the components of \tilde{u} and \tilde{v} are defined by the $n - 1$ Givens rotations. Hence the (i, j) element of \tilde{T}^{-1} is given by

$$\tilde{\eta}_{ij} = e_i^T \tilde{T}^{-1} e_j = e_i^T \tilde{R}^{-1} \tilde{Q}^T e_j = \tilde{u}_j e_i^T \tilde{R}^{-1} \tilde{v} = \tilde{u}_j e_i^T \tilde{w} = \tilde{u}_j \tilde{w}_i, \quad i \geq j,$$

where \tilde{w} is the solution of $\tilde{R}\tilde{w} = \tilde{v}$.

The (i, j) element of T^{-1} , $i < j$, is therefore given by

$$\eta_{ij} = \tilde{u}_{n-j+1} \tilde{v}_{n-i+1}.$$

We now return to the elements of T^{-1} in the lower triangle. The computation of u and v using (2.4) may cause underflow and overflow since the diagonal entries of D are products of ψ_i with $|\psi_i| \leq 1$. In [4] it was noted that a way of avoiding such problems for computing the diagonal elements of T^{-1} is to scale the triangular system $Rw = v$ with the diagonal matrix D . This gives $R'w' = v'$, where

$$(2.7) \quad R' = D^{-1}RD, \quad w' = D^{-1}w, \quad v' = D^{-1}v = [\phi_1, \dots, \phi_{n-1}, 1]^T.$$

Using this scaling avoids products of ψ_i . The entries of R' are given by

$$(2.8) \quad r'_i = r_i, \quad s'_i = -\psi_i s_i, \quad t'_i = \psi_i \psi_{i+1} t_i,$$

and since $|\psi_i| \leq 1$, these values cannot overflow.

Let

$$(2.9) \quad u' = [1, \phi_1, \phi_2, \dots, \phi_{n-1}]^T.$$

As $\phi_i^2 + \psi_i^2 = 1$, the components of u' are all bounded by 1 in modulus. Since $w = Dw'$ and $u = D^{-1}u'$, the diagonal elements of T^{-1} are given by

$$\eta_{ii} = u_i w_i = u'_i w'_i, \quad i = 1:n.$$

The strictly lower triangular elements are given by

$$\eta_{ij} = u_j w_i = \frac{d_i}{d_j} u'_j w'_i, \quad i > j,$$

and hence

$$|\eta_{ij}| = |\psi_j \dots \psi_{i-1} u'_j w'_i|, \quad i > j,$$

for which we clearly have possible underflow problems if η_{ij} is computed in this way.

Fortunately, this problem can be overcome by considering how to compute the 1-norm of the j th column of the strictly lower triangular part of T^{-1} , which is given by

$$\sigma_j = |u'_j| (|w'_{j+1} \psi_j| + |w'_{j+2} \psi_j \psi_{j+1}| + \dots + |w'_n \psi_j \dots \psi_{n-1}|), \quad j = 1:n-1.$$

If σ_j is computed in this way $O(n^2)$ operations would be required to compute σ_j for $j = 1:n-1$. The underflow problem can be overcome and the operation count reduced by an order of magnitude using nested multiplication. We can rewrite σ_j as

$$\sigma_j = |u'_j| (((\dots ((|\psi_{n-1} w'_n| + |w'_{n-1}|) |\psi_{n-2}| + |w'_{n-2}|) \dots) |\psi_{j+1}| + |w'_{j+1}|) |\psi_j|), \\ j = 1:n-1.$$

Therefore σ_j can be computed using

```
(2.10) 1   $\sigma'_{n-1} = |\psi_{n-1}w'_n|$ 
        2  for  $j = n - 1 : -1 : 2$ 
        3       $\sigma'_{j-1} = (\sigma'_j + |w'_j|)|\psi_{j-1}|$ 
        4  end
        5  for  $j = 1 : n - 1$ ,  $\sigma_j = |u'_j|\sigma'_j$ , end
```

This method of computing σ_j , avoids forming explicit products of ψ_i and allows σ_j , $j = 1 : n - 1$, to be computed in $O(n)$ operations. For underflow to occur $\sigma_j/|u'_j|$ must underflow, and since $\sigma_j/|u'_j| \geq \sigma_j$, this can only occur if σ_j , the 1-norm of the j th column of the strictly lower triangular part of T^{-1} , underflows. We note that this is very unlikely and that underflow does not cause the algorithm to break down.

When considering \tilde{T} , we can use a similar approach that also adds the absolute values of the diagonal elements of T^{-1} . This enables us to compute the 1-norm of the j th column of the upper triangular part of T^{-1} , denoted by $\tilde{\sigma}_j$, in $O(n)$ operations without the risk of underflow. Defining $\sigma_n = 0$ the 1-norm of T^{-1} is then given by

$$\|T^{-1}\|_1 = \max_{j=1:n}(\sigma_j + \tilde{\sigma}_j).$$

The pseudocode in the following algorithm summarizes the method described above. The function $(\phi, \psi, r) = \text{Givens}(a, b)$ computes $r = \sqrt{a^2 + b^2}$, $\phi = a/r$ and $\psi = b/r$, and guards against overflow.

ALGORITHM 2.1. Computes $\tau = \|T^{-1}\|_1$, where T is given by (2.1).

```
1  for  $k = 1 : 2$ 
2       $a = \alpha_1$ ,  $g = \gamma_1$ ,  $u'_1 = 1$ 
3      for  $i = 1 : n - 1$ 
4           $(\phi_i, \psi_i, r'_i) = \text{Givens}(a, \beta_i)$ 
5          if  $r'_i = 0$ ,  $\tau = \infty$ , return, end
6           $s'_i = -\psi_i(\phi_i g + \psi_i \alpha_{i+1})$ 
7           $a = -\psi_i g + \phi_i \alpha_{i+1}$ ,  $u'_{i+1} = \phi_i$ ,  $v'_i = \phi_i$ 
8          if  $i < n - 1$ ,  $t_i = \psi_i^2 \gamma_{i+1}$ ,  $g = \phi_i \gamma_{i+1}$ , end
9          if  $i > 1$ ,  $t'_{i-1} = \psi_i t'_{i-1}$ , end
10     end
11      $r'_n = a$ ,  $v'_n = 1$ 
12     if  $r'_n = 0$ ,  $\tau = \infty$ , return, end
13      $w'_n = v'_n / r'_n$ 
14      $w'_{n-1} = (v'_{n-1} - w'_n s'_{n-1}) / r'_{n-1}$ 
15     for  $i = n - 2 : -1 : 1$ 
16          $w'_i = (v'_i - w'_{i+1} s'_i - w'_{i+2} t'_i) / r'_i$ 
17     end
18     if  $k = 1$ 
19         Compute  $\sigma_j$  using the code given in (2.10)
20          $\alpha = \alpha(n : -1 : 1)$ ,  $\delta = \beta$ ,  $\beta = \gamma(n - 1 : -1 : 1)$ ,  $\gamma = \delta(n - 1 : -1 : 1)$ 
21     else
22          $\tilde{\sigma}_n = |w'_n|$ 
23         for  $i = n - 1 : -1 : 1$ 
24              $\tilde{\sigma}_i = \tilde{\sigma}_{i+1} |\psi_j| + |w'_i|$ 
25         end
26         for  $i = 1 : n - 1$ ,  $\tilde{\sigma}_j = \tilde{\sigma}_j |u'_j|$ , end
27     end
28 end
29  $\tau = \max_{i=1:n}(\sigma_i + \tilde{\sigma}_{n-i+1})$ 
```

Cost: Assuming the function *Givens* requires 6 operations, the total cost is $51n$ operations.

Lines 2–12 compute the vectors r' , s' , t' in (2.8), u' in (2.9) and v' in (2.7) and lines 13–17 solve the linear system $R'w' = v'$. The 1-norms of the columns of the strictly lower and the upper triangular parts of T^{-1} are computed in lines 19 and 22–26 respectively. The only divisions in the algorithm are by r_i . Hence the test on line 5 prevents division by zero, which is possible if T is singular.

Algorithm 2.1 does not have to deal with the reduced case separately as in [11], or have numerous tests to deal with the reduced case as in [6]. It can also be easily adapted to compute $\|T^{-1}\|_1$ for $T \in \mathbb{C}^{n \times n}$ by using complex Givens rotations and the complex conjugate of v' in (2.7).

We note that there are certain similarities between Algorithm 2.1 and those in [6]. All the algorithms use factorizations: triangular factorizations in [6] and orthogonal factorizations in Algorithm 2.1. Also, Algorithm 2.1 and the algorithms in [6] both compute two factorizations, with the first proceeding from the top of T to the bottom and the second from bottom of T to the top. In Algorithm 2.1 this takes the form of a QR factorization followed by a QL factorization, whereas in [6] the factorizations $T = L_+D_+U_+$ and $T = L_-D_-U_-$ are computed, where L_+ and L_- are lower bidiagonal, U_+ and U_- upper bidiagonal, and D_+ and D_- diagonal.

2.2. Reducing the operation count. The matrix T in (2.1) is said to be unreduced if $\beta_i\gamma_i \neq 0$ for $i = 1:n-1$. If T is unreduced then it can be scaled using the diagonal matrix

$$D = \text{diag}(d_i), \quad d_1 = 1, \quad d_i = \frac{\beta_1 \dots \beta_{i-1}}{\gamma_1 \dots \gamma_{i-1}}, \quad i > 1,$$

so that $\tilde{T} = TD$ is symmetric. In this section we will use this property to obtain an algorithm for computing $\|T^{-1}\|_1$ that requires approximately half the number of operations of Algorithm 2.1.

We first note that it is also possible to use a similar scaling so that $\check{T} = \check{D}T$ is symmetric, where \check{D} is diagonal and is defined by products of γ_i/β_i . A naive method of computing $\|T^{-1}\|_1$ would be to use this scaling to form \check{T} , compute the absolute column sums of \check{T}^{-1} using the ideas of Section 2.1 and making use of its symmetry, and then recover the absolute column sums of T^{-1} using $T^{-1} = \check{T}^{-1}\check{D}$. Explicitly forming \check{T} in this approach can lead to underflow and overflow due to the computation of the elements of \check{D} . We will use the scaling $\tilde{T} = TD$ without forming \tilde{T} and use the structure of T^{-1} to deal with reduced T .

The QR factorization of \tilde{T} has the form

$$Q^T\tilde{T} = RD,$$

where Q and R satisfy $Q^T\tilde{T} = R$ and can be computed using (2.4). The matrix Q is defined by (2.5). The (i, j) element of \tilde{T}^{-1} , $\tilde{\eta}_{ij}$, is therefore given by

$$\tilde{\eta}_{ij} = d_i^{-1}u_j w_i,$$

where w is the solution of the triangular system $Rw = v$.

In Section 2.1 we showed how to scale the system $Rw = v$ to avoid underflow and overflow. Using the same procedure here we find that

$$|\tilde{\eta}_{ij}| = |d_i^{-1}\psi_j \dots \psi_{i-1}u'_j w'_i|, \quad i \geq j,$$

where u' is defined by (2.9) and w' is defined by (2.7).

Since $T^{-1} = D\tilde{T}^{-1}$ the (i, j) element of T^{-1} , η_{ij} , satisfies

$$|\eta_{ij}| = |\psi_j \dots \psi_{i-1}u'_j w'_i|, \quad i \geq j,$$

which is precisely what we found in Section 2.1. Therefore the 1-norm of the j th column of the strictly lower triangle of T^{-1} , σ_j , can be calculated as described in Section 2.1.

Using the symmetry of \tilde{T}^{-1} the upper triangular elements of \tilde{T}^{-1} satisfy

$$|\tilde{\eta}_{ij}| = |d_j^{-1}\psi_i \dots \psi_{j-1}u'_i w'_j|, \quad i \leq j,$$

and therefore using $T^{-1} = D\tilde{T}^{-1}$

$$(2.11) \quad \begin{aligned} |n_{ij}| &= \left| \frac{d_i}{d_j} \psi_i \dots \psi_{j-1} u'_i w'_j \right|, \quad i \leq j, \\ &= \left| \frac{\gamma_i \dots \gamma_{j-1}}{\beta_i \dots \beta_{j-1}} \psi_i \dots \psi_{j-1} u'_i w'_j \right|. \end{aligned}$$

Using nested multiplication, similar to that described for computing σ_j , we can compute the 1-norm of the j th column of the upper triangle of T^{-1} , $\tilde{\sigma}_j$, using

$$(2.12) \quad \begin{array}{l} 1 \quad \tilde{\sigma}_1 = |u'_1| \\ 2 \quad \text{for } j = 1:n-1 \\ 3 \quad \quad \tilde{\sigma}_{j+1} = \tilde{\sigma}_j |\gamma_j \psi_j / \beta_j| + |u'_{j+1}| \\ 4 \quad \text{end} \\ 5 \quad \text{for } j = 1:n, \tilde{\sigma}_j = |w'_j| \tilde{\sigma}_j, \text{ end} \end{array}$$

Computing $\tilde{\sigma}_j$ in this way allows us to make use of the lower triangular part of T^{-1} to find the upper triangular part and therefore approximately halves the number of operations required compared with the method of the previous section. We note that we encounter division by zero when computing $\tilde{\sigma}_j$ if $\beta_j = 0$ and hence the procedure described above is only valid for tridiagonal matrices with all $\beta_j \neq 0$. However, it is possible to remove this restriction by considering the structure of T when $b_k = 0$ for some k .

If the tridiagonal matrix T has $b_k = 0$ then we can write it as

$$T = \begin{bmatrix} T_1 & C \\ 0 & T_2 \end{bmatrix}, \quad T_1 \in \mathbb{R}^{k \times k}, T_2 \in \mathbb{R}^{(n-k) \times (n-k)}, C \in \mathbb{R}^{k \times (n-k)},$$

where $C = c_{k-1}e_k e_1^T$ and e_j denotes the j th unit vector. The inverse of the tridiagonal matrix is then given by

$$T^{-1} = \begin{bmatrix} T_1^{-1} & -T_1^{-1}CT_2^{-1} \\ 0 & T_2^{-1} \end{bmatrix},$$

and

$$(2.13) \quad T_1^{-1}CT_2^{-1} = c_{k-1}(T_1^{-1}e_k)(T_2^{-1}e_1)^T.$$

We note that the k th column sum of xy^T , where $x, y \in \mathbb{R}^n$, is $\|x\|_1|y_k|$ and therefore we can find the column sums of (2.13) using the last column sum of T_1^{-1} and the first row of T_2^{-1} .

Suppose we are using the procedure described in (2.12) to compute $\tilde{\sigma}_j$. Then if we encounter a β_j that is zero at line 3, the absolute last column sum of T_1^{-1} is given by $w'_j \tilde{\sigma}_j$. The absolute values of the entries of the upper triangular part of T_2^{-1} are given by (2.11). Since we know the last absolute column sum of T_1^{-1} and the absolute values of the first row of T_2^{-1} , we can combine these and the absolute values of the upper triangular part of T_2^{-1} to give the 1-norm of the j th column of the upper triangular part of T^{-1} for $j = k+1:n$. This is achieved by setting

$$\tilde{\sigma}_{j+1} = |\tilde{\sigma}_j w_j c_j u'_{j+1}| + |u'_{j+1}|,$$

instead of line 3. The $|u'_{j+1}|$ term corresponds to the 1-norm of the upper triangular part of T_2^{-1} and the other term corresponds to the 1-norm of the columns of (2.13). Clearly these ideas apply to T_1^{-1} and T_2^{-1} and therefore the procedure will work if there are several β_k equal to zero. We summarize this method of computing the absolute column sums of the upper triangular part of T^{-1} as follows:

```

1   $\tilde{\sigma}_1 = |u'_1|$ 
2  for  $j = 1:n-1$ 
3      if  $\beta_j = 0$ 
4           $\tilde{\sigma}_{j+1} = |\tilde{\sigma}_j w'_j \gamma_j u'_{j+1}| + |u'_{j+1}|$ 
(2.14) 5      else
6           $\tilde{\sigma}_{j+1} = \tilde{\sigma}_j |\gamma_j \psi_j / \beta_j| + |u'_{j+1}|$ 
7      end
8  end
9  for  $j = 1:n$ ,  $\tilde{\sigma}_j = |w'_j| \tilde{\sigma}_j$ , end

```

Defining $\sigma_n = 0$, we have $\|T^{-1}\|_1 = \max_{j=1:n}(\sigma_j + \tilde{\sigma}_j)$. The following algorithm computes $\|T^{-1}\|_1$ using the method described in this section.

ALGORITHM 2.2. *Computes $\tau = \|T^{-1}\|_1$, where T is given by (2.1).*

- 1 Use lines 2–12 and 19 in Algorithm 2.1 to compute σ_i
- 2 Compute $\tilde{\sigma}_i$ using the code in (2.14)
- 3 $\tau = \max_{i=1:n}(\sigma_i + \tilde{\sigma}_i)$

Cost: Assuming the function *Givens* requires 6 operations, the total cost is $31n$ operations.

2.3. Rounding error analysis. A rounding error analysis of Algorithm 2.1 is given. We have been unable to give a rounding error analysis of Algorithm 2.2 since when considering the upper triangular part of T^{-1} we obtain error results which are bounded by the norm of $\tilde{T} = TD$, which cannot be bounded a priori. However, extensive numerical experiments suggest that Algorithm 2.2 behaves in a stable way.

We will use the standard model for floating point arithmetic:

$$fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta_1) = \frac{x \text{ op } y}{1 + \delta_2}, \quad |\delta_1|, |\delta_2| \leq u, \quad \text{op} = +, -, /, *,$$

where u is the unit roundoff of the computer. We denote by $\|\cdot\|_F$ the unitarily invariant Frobenius norm and make use of the following lemmas [12].

LEMMA 2.2 ([12, Lem. 3.4]). *If $|\delta_i| \leq u$ and $\rho_i = \pm 1$ for $i = 1:n$, and $nu < 1$, then*

$$\prod_{i=1}^n (1 + \delta_i)^{\rho_i} = 1 + \theta_n,$$

where

$$|\theta_n| \leq \frac{nu}{1 - nu} =: \gamma_n.$$

We also make use of the constant

$$\tilde{\gamma}_n = \frac{cnu}{1 - cnu},$$

where c denotes a small integer constant whose exact value is unimportant.

LEMMA 2.3 ([12, Lem. 6.6]). *Let $A, B \in \mathbb{R}^{n \times n}$. If $|A| \leq |B|$ then $\|A\|_F \leq \|B\|_F$.*

LEMMA 2.4. *Let $A \in \mathbb{R}^{n \times n}$. Then $\|A\|_1 \leq \sqrt{n}\|A\|_F$ and $\|A\|_F \leq \sqrt{n}\|A\|_1$.*

Given a tridiagonal matrix $T \in \mathbb{R}^{n \times n}$, the first step of the algorithm is to compute the QR factorization of T . From [12, Thm. 19.10] we have that there exists an orthogonal Q such that the computed upper triangular matrix \widehat{R} satisfies

$$(2.15) \quad T + \Delta T = Q\widehat{R}, \quad \|\Delta T\|_F \leq \tilde{\gamma}_n \|T\|_F.$$

We make the simplifying assumption that the defining variables of the $n - 1$ Givens rotations, $\phi_1, \dots, \phi_{n-1}$ and $\psi_1, \dots, \psi_{n-1}$, are computed exactly. Using $D = \text{diag}(d_1, \dots, d_n)$, where D is given by (2.4), the components of the computed scaled upper triangular matrix \widehat{R}' (2.8) satisfy

$$\begin{aligned} \widehat{r}'_i &= \widehat{r}_i, \\ \widehat{s}'_i &= \frac{d_{i+1}}{d_i} \widehat{s}_i (1 + \Delta s_i), \quad |\Delta s_i| \leq \gamma_1, \\ \widehat{t}'_i &= \frac{d_{i+2}}{d_i} \widehat{t}_i (1 + \Delta t_i), \quad |\Delta t_i| \leq \gamma_2. \end{aligned}$$

We now consider solving the triangular system $\widehat{R}'w' = \widehat{v}'$ and the errors involved in solving the system using backward substitution. Since we have made the assumption that $\phi_1, \dots, \phi_{n-1}$ are computed exactly, the vectors $v' = D^{-1}v = [\phi_1, \dots, \phi_{n-1}, 1]^T$ and $u' = Du = [1, \phi_1, \dots, \phi_{n-1}]^T$ must also be computed exactly. The last component of the computed solution to the triangular system therefore satisfies

$$\widehat{r}_n \widehat{w}'_n (1 + \theta_1) = \frac{v_n}{d_n},$$

which simplifies to

$$d_n \widehat{r}_n (1 + \widetilde{\Delta r}_n) \widehat{w}'_n = v_n, \quad |\widetilde{\Delta r}_n| \leq \gamma_1.$$

The computed component \widehat{w}'_{n-1} satisfies

$$\widehat{r}_{n-1} \widehat{w}'_{n-1} (1 + \theta'_1) = \frac{v_{n-1}}{d_{n-1}} - \widehat{w}'_n \frac{d_n}{d_{n-1}} \widehat{s}_{n-1} (1 + \Delta s_{n-1}) (1 + \theta_3),$$

which simplifies to

$$d_{n-1} \widehat{r}_{n-1} (1 + \widetilde{\Delta r}_{n-1}) \widehat{w}'_{n-1} = v_{n-1} - \widehat{w}'_n d_n \widehat{s}_{n-1} (1 + \widetilde{\Delta s}_{n-1}),$$

where $|\widetilde{\Delta r}_{n-1}| \leq \gamma_1$, and $|\widetilde{\Delta s}_{n-1}| \leq \gamma_4$.

Similarly for $i = 1:n - 2$, \widehat{w}'_i satisfies

$$d_i \widehat{r}_i (1 + \widetilde{\Delta r}_i) \widehat{w}'_i = v_i - \widehat{w}'_{i+1} d_{i+1} \widehat{s}_i (1 + \widetilde{\Delta s}_i) - \widehat{w}'_{i+2} d_{i+2} \widehat{t}_i (1 + \widetilde{\Delta t}_i),$$

where

$$|\widetilde{\Delta r}_i| \leq \gamma_1, \quad |\widetilde{\Delta s}_i| \leq \gamma_4, \quad |\widetilde{\Delta t}_i| \leq \gamma_5.$$

Combining the above results for all the components of \widehat{w}' and using Lemma 2.3 gives

$$(2.16) \quad (\widehat{R} + \widetilde{\Delta R}) D \widehat{w}' = v,$$

where

$$\|\widetilde{\Delta R}\|_F \leq \gamma_5 \|\widehat{R}\|_F \leq \gamma_5 \|T\|_F + O(u^2).$$

By considering the errors in computing σ_j using nested multiplication as described in Section 2.1, the computed 1-norm of the j th column of the lower triangular part of T^{-1} is found to satisfy

$$(2.17) \quad \hat{\sigma}_j = \left(\sum_{i=j}^n |\hat{\eta}_{ij}| \right) (1 + \gamma_{2n-2}),$$

where

$$\begin{aligned} \hat{\eta}_{ij} &= (-1)^{i+j} \psi_j \dots \psi_{i-1} \hat{u}'_j \hat{w}'_i \\ &= d_i u_j e_i^T \hat{w}' \\ &= d_i u_j e_i^T D^{-1} (\hat{R} + \widetilde{\Delta R})^{-1} v \quad \text{using (2.16)} \\ &= e_i^T (\hat{R} + \widetilde{\Delta R})^{-1} Q^T e_j. \end{aligned}$$

Rearranging (2.15) gives $\hat{R} = Q^T (T + \Delta T)$ and hence

$$\begin{aligned} \hat{\eta}_{ij} &= e_i^T (Q^T (T + \Delta T) + \widetilde{\Delta R})^{-1} Q^T e_j \\ &= e_i^T (T + \Delta T + Q \widetilde{\Delta R})^{-1} e_j \\ &= e_i^T (T + \widetilde{\Delta T})^{-1} e_j, \end{aligned}$$

where

$$\|\widetilde{\Delta T}\|_F = \|\Delta T + Q \widetilde{\Delta R}\|_F \leq \tilde{\gamma}_n \|T\|_F.$$

Using Lemma 2.4 we have

$$\|\widetilde{\Delta T}\|_1 \leq n \tilde{\gamma}_n \|T\|_1.$$

Using the result [19]

$$\frac{\|A^{-1} - (A + E)^{-1}\|_1}{\|A^{-1}\|_1} \leq \frac{e}{1 - e}, \quad e = \kappa_1(A) \frac{\|E\|_1}{\|A\|_1} < 1,$$

and defining $(\hat{T}^{-1})_{ij} = \hat{\eta}_{ij}$ we have

$$\begin{aligned} \frac{\|\hat{T}^{-1} - T^{-1}\|_1}{\|T^{-1}\|_1} &= \frac{\|(T + \widetilde{\Delta T})^{-1} - T^{-1}\|_1}{\|T^{-1}\|_1} \\ &\leq \frac{e}{1 - e}, \quad e = \kappa_1(T) \frac{\|\widetilde{\Delta T}\|_1}{\|T\|_1} < 1. \end{aligned}$$

Therefore

$$(2.18) \quad \begin{aligned} \|\hat{T}^{-1} - T^{-1}\|_1 &\leq \kappa_1(T) \frac{\|\widetilde{\Delta T}\|_1 \|T^{-1}\|_1}{\|T\|_1} \\ &\leq n \tilde{\gamma}_n \kappa_1(T) \|T^{-1}\|_1. \end{aligned}$$

Using (2.17) and (2.18)

$$|\hat{\sigma}_j - \sigma_j| \approx \left| \sum_{i=j}^n (|\hat{\eta}_{ij}| - |\eta_{ij}|) \right| \leq \sum_{i=j}^n |\hat{\eta}_{ij} - \eta_{ij}| \leq n \tilde{\gamma}_n \kappa_1(T) \|T^{-1}\|_1.$$

A similar result holds for the 1-norm of the columns of the upper triangular part of T^{-1} . Hence if we denote by $\hat{\tau}$, our approximation to $\|T^{-1}\|_1$, computed in floating point arithmetic, we have

$$(2.19) \quad \frac{|\hat{\tau} - \|T^{-1}\|_1|}{\|T^{-1}\|_1} \leq 2n \tilde{\gamma}_n \kappa_1(T).$$

This error result is the best we can expect, since it can be shown that the condition number of computing the condition number is the condition number [10].

2.4. Numerical experiments. We compare the accuracy of our new algorithms against Dhillon's recommended algorithm *nrminv_final2*, Algorithm 4.2 from [11] and MATLAB's *cond*, which computes the condition number of a matrix in $O(n^3)$ operations. The speed of the new algorithms is tested with Dhillon's *nrminv_final2* but also the quicker algorithm *nrminv_final1*. The test matrices are described in Table 2.1.

TABLE 2.1
Test matrices.

Matrix Type	Description
1	Nonsymmetric random tridiagonal, elements uniformly distributed in $[-1, 1]$.
2	<code>gallery('randsvd', 100, 1e15, 2, 1, 1)</code> in MATLAB, which creates a random tridiagonal matrix with all singular values close to 1 except for one that is of order 10^{-15} , so that the 2-norm condition number is 10^{15} .
3	<code>gallery('randsvd', 100, 1e15, 3, 1, 1)</code> in MATLAB, which creates a random tridiagonal matrix with geometrically distributed singular values and 2-norm condition number 10^{15} .
4	Tridiagonal with $\alpha_i = 10^8$, $\beta_i = \gamma_i = 1$.
5	Tridiagonal with $\alpha_i = 10^{-8}$, $\beta_i = \gamma_i = 1$.
6	<code>gallery('lesp', 100)</code> in MATLAB.
7	<code>gallery('dorr', 100, 1e-4)</code> in MATLAB.
8	Nonsymmetric tridiagonal, elements uniformly distributed in $[-1, 1]$ except β_{50} given by 1×10^{-50} multiplied by a random number in $[-1, 1]$.
9	Nonsymmetric tridiagonal, elements of α and γ uniformly distributed in $[-1, 1]$ and β_i given by 1×10^{-50} multiplied by a random number in $[-1, 1]$, for $i = 1:n$.
10	Symmetric tridiagonal with $\alpha_i = 0$, $\beta_i = \gamma_i = 1$.

The results, given in Table 2.2, show that the new algorithms give the results up to four decimal place of accuracy for all but test matrix 7. Also, the new algorithms do not suffer from the underflow and overflow problems of Higham's algorithm, which breaks down for test matrices 2, 4, 6, and 9 due to overflow. All the algorithms correctly detect the singular test matrix 10. The difference in results for test matrix 7 are due to rounding errors, and in fact all the methods tested are inaccurate since the actual condition number is 8.885×10^{24} . As noted in Section 2.3, the best error bound we can expect to obtain for computing the condition number is of the form (2.19). In this case the bound (2.19) is approximately 10^{34} which suggests that the computed condition number may be inaccurate.

TABLE 2.2
Computation of $\kappa_1(T)$ on test matrices. Test matrices 1 – 9 are of order 100 and test matrix 10 is of order 99.

Matrix Type	Higham's algorithm	<i>nrminv_final2</i>	Algorithm 2.1	Algorithm 2.2	MATLAB's <i>cond</i>
1	2.9946e+03	2.9946e+03	2.9946e+03	2.9946e+03	2.9946e+03
2	NaN	1.4979e+15	1.4979e+15	1.4979e+15	1.4979e+15
3	4.5336e+15	4.5336e+15	4.5336e+15	4.5336e+15	4.5336e+15
4	NaN	1.0000	1.0000	1.0000	1.0000
5	100.0000	100.0000	100.0000	100.0000	100.0000
6	NaN	67.1164	67.1164	67.1164	67.1164
7	2.3712e+19	2.7617e+19	1.9403e+19	2.1895e+19	3.5211e+19
8	1.0568e+04	1.0568e+04	1.0568e+04	1.0568e+04	1.0568e+04
9	NaN	6.1603e+10	6.1603e+10	6.1603e+10	6.1603e+10
10	∞	∞	∞	∞	∞

Before we consider the times taken by the various algorithms we first consider the cost. For Algorithms 2.1 and 2.2, this depends on how the function *Givens* is implemented. In our experiments we use the LAPACK [1] routine *DLARTG* in place of *Givens*, which requires at most 10 operations but typically 6. The maximum

number of operations required by the algorithms for computing $\|T^{-1}\|_1$ are given in Table 2.3.

TABLE 2.3

The maximum number of operations required to compute the 1-norm of the inverse of an $n \times n$ tridiagonal matrix for Dhillon's algorithms and the algorithms presented here.

Algorithm	<i>nrminv_final1</i>	<i>nrminv_final2</i>	Algorithm 2.1	Algorithm 2.2
Total	$22n$	$27n$	$59n$	$35n$

Table 2.4 shows the times taken for the new algorithms and Dhillon's algorithms to compute $\kappa_1(T)$ for nonsymmetric random tridiagonal matrices. The algorithms were implemented in Fortran 77 and compiled using the NAGWare Fortran 95 compiler with the normal optimization level (-O2) and IEEE arithmetic. The machine used was a 2010MHz AMD Athlon machine running Linux.

The results show that *nrminv_final1* is the quickest as expected. However, Algorithm 2.2 runs much more quickly than the operation count would suggest, which could be due to the fewer if-statements required. As a result, Algorithm 2.2 is only marginally slower than *nrminv_final1*. However Algorithm 2.2 is quicker than *nrminv_final2*, which is the recommended algorithm in [6] due to less element growth in its computation. Interestingly, if compiled with no optimization (-O0) Algorithm 2.2 is as quick as *nrminv_final1*.

TABLE 2.4

Time taken in seconds to compute the 1-norm of the inverse of a random $n \times n$ tridiagonal matrix, for various n .

Dimension	10^6	2×10^6	4×10^6	6×10^6	8×10^6
<i>nrminv_final1</i>	0.31	0.60	1.22	1.78	2.54
<i>nrminv_final2</i>	0.37	0.73	1.45	2.17	3.08
Algorithm 2.1	0.60	1.20	2.37	3.54	4.87
Algorithm 2.2	0.33	0.65	1.26	1.93	2.72

3. Diagonal-plus-semiseparable matrices. A semiseparable matrix, S , is the sum of the strictly upper triangular part of a rank-1 matrix and the lower triangular part of another rank-1 matrix,

$$S = \text{tril}(qp^T, 0) + \text{triu}(xy^T, 1)$$

Here, $\text{tril}(A, i)$ denotes a matrix A with the entries above the i th diagonal set to zero, where $i = 0$ is the leading diagonal and $i > 0$, $i < 0$, is above and below the leading diagonal respectively. Similarly $\text{triu}(A, i)$ denotes A with the entries below the i th diagonal set to zero.

A diagonal-plus-semiseparable (dpss) matrix, A , is the sum of a diagonal matrix and a semiseparable matrix:

$$(3.1) \quad A = \text{diag}(z) + S = \begin{bmatrix} p_1q_1 + z_1 & x_1y_2 & \cdots & x_1y_n \\ p_1q_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & & x_{n-1}y_n \\ p_1q_n & \cdots & p_{n-1}q_n & p_nq_n + z_n \end{bmatrix}.$$

We assume that A is nonsingular.

Assuming that the defining vectors p , q , x , y and z are given, we show how to compute the condition number of A in $O(n)$ operations. This is achieved in a similar way to the tridiagonal case in Section 2.1. We will again make use of the special structure of Q and R in the QR factorization of A , solve a scaled linear

system and form absolute column sums using methods that avoid underflow and overflow.

We first note that it is possible to compute $\|A\|_1$ in $O(n)$ operations since the i th absolute column sum can be computed using

$$|y_i|(|x_1| + |x_2| + \cdots + |x_{i-1}|) + |p_i|(|q_{i+1}| + |q_{i+2}| + \cdots + |q_n|) + |p_i q_i + z_i|.$$

By accumulating the sums $|x_1| + |x_2| + \cdots + |x_{i-1}|$ and $|q_{i+1}| + |q_{i+2}| + \cdots + |q_n|$ we can compute the absolute column sums and hence $\|A\|_1$ in $O(n)$ operations. The problem remains to compute $\|A^{-1}\|_1$.

3.1. Computing the 1-norm of the inverse of a dpss matrix. The QR factorization of a diagonal-plus-semiseparable matrix can be considered in two stages. First, it is shown in [2] that by applying Givens rotations and scaling we can find an orthogonal matrix $Q_1 \in \mathbb{R}^{n \times n}$ so that $Q_1^T A = H$, where H is upper Hessenberg. We require the condition $q_n \neq 0$, however we later show that if $q_n = 0$ we need only consider a principal submatrix of A . The special structure of H and Q_1 is described in the following proposition which summarizes the result from [2].

PROPOSITION 3.1. *The upper Hessenberg matrix $H = Q_1^T A$ has an upper triangular part that is equal to the upper triangular part of the rank-2 matrix $ab^T + fg^T \in \mathbb{R}^{n \times n}$ and subdiagonal $h \in \mathbb{R}^{n-1}$, where*

$$(3.2) \quad a = \begin{bmatrix} 1 \\ q_1 \\ q_2 \\ \vdots \\ q_{n-1} \end{bmatrix}, \quad b = \begin{bmatrix} p_1 \mu_1 + z_1 q_1 \\ (q_1 x_1) y_2 + p_2 \mu_2 + z_2 q_2 \\ (\sum_{i=1}^2 q_i x_i) y_3 + p_3 \mu_3 + z_3 q_3 \\ \vdots \\ (\sum_{i=1}^{n-1} q_i x_i) y_n + p_n \mu_n + z_n q_n \end{bmatrix},$$

$$(3.3) \quad f = \begin{bmatrix} 0 \\ -\mu_1 x_1 \\ -(\sum_{i=1}^1 q_i x_i) q_2 - \mu_2 x_2 \\ \vdots \\ -(\sum_{i=1}^{n-2} q_i x_i) q_{n-1} - \mu_{n-1} x_{n-1} \end{bmatrix}, \quad g = y, \quad h = \begin{bmatrix} -z_1 \mu_2 \\ -z_2 \mu_3 \\ \vdots \\ -z_{n-1} \mu_n \end{bmatrix},$$

with $\mu_i = q_i^2 + \cdots + q_n^2$. The orthogonal matrix Q_1^T is upper Hessenberg with an upper triangular part that is equal to the upper triangular part of the rank-1 matrix $rs^T \in \mathbb{R}^{n \times n}$ and subdiagonal $t \in \mathbb{R}^{n-1}$, where

$$(3.4) \quad r = \begin{bmatrix} 1 \\ q_1 \\ q_2 \\ \vdots \\ q_{n-1} \end{bmatrix}, \quad s = q, \quad t = \begin{bmatrix} -\mu_2 \\ -\mu_3 \\ \vdots \\ -\mu_n \end{bmatrix}.$$

We note that the defining vectors of H and Q_1^T can all be computed in $O(n)$ operations.

The second stage is to reduce H to upper triangular form, which can be achieved by $n - 1$ Givens rotations, G_i , defined by (2.3).

The first Givens rotation G_1 is chosen to zero the first element of the subdiagonal of H , and is therefore defined by $\phi_1 = (a_1 b_1 + f_1 g_1) / \tau_1$ and $\psi_1 = h_1 / \tau_1$ with $\tau_1 = \sqrt{(a_1 b_1 + f_1 g_1)^2 + h_1^2}$. Since G_1 zeros h_1 and the upper triangular part of H is the upper triangular part of $ab^T + fg^T$, we apply G_1 to a and f to give $a^{(1)} = G_1 a$ and $f^{(1)} = G_1 f$. Except for the first diagonal element, the upper triangular part of $G_1 H$ is the upper triangular part of the rank-2 matrix $a^{(1)} b^T + f^{(1)} g^T$. The first diagonal element is given by τ_1 .

This can be repeated to zero the i th subdiagonal element of $G_{i-1} \dots G_1 H$ by choosing

$$\begin{aligned}\tau_i &= \sqrt{(a_i^{(i-1)} b_i + f_i^{(i-1)} g_i)^2 + h_i^2}, \\ \phi_i &= (a_i^{(i-1)} b_i + f_i^{(i-1)} g_i) / \tau_i, \quad \psi_i = h_i / \tau_i,\end{aligned}$$

and forming $a^{(i)} = G_i a^{(i-1)}$ and $f^{(i)} = G_i f^{(i-1)}$.

After the $n - 1$ Givens rotations have been applied

$$(3.5) \quad Q_2^T Q_1^T A = Q_2^T H = G_{n-1} \dots G_1 H = R,$$

where R is upper triangular. The strictly upper triangular part of R is the strictly upper triangular part of $a^{(n-1)} b^T + f^{(n-1)} g^T$, where

$$a^{(n-1)} = G_{n-1} \dots G_1 a \quad \text{and} \quad f^{(n-1)} = G_{n-1} \dots G_1 f,$$

and the i th diagonal of R is given by τ_i for $i = 1:n-1$ and the n th diagonal element is given by $\tau_n \equiv a_n^{(n-1)} b_n + f_n^{(n-1)} g_n$. Since Q_2^T is a product of $n-1$ Givens rotations applied from top to bottom the structure of Q_2^T is given by (2.5).

From (3.5), we have $A^{-1} = R^{-1} Q_2^T Q_1^T$. In order to compute $\|A^{-1}\|_1$ we first show how to find the lower triangular elements of $R^{-1} Q_2^T$ by using a similar approach to how the lower triangular elements of the inverse of a tridiagonal matrix are found in Section 2.1.

The (i, j) element of $R^{-1} Q_2^T$ is

$$(R^{-1} Q_2^T)_{ij} = e_i^T R^{-1} Q_2^T e_j = u_j e_i^T R^{-1} v, \quad i \geq j,$$

where e_i denotes the i th unit vector. If we let w be the solution of the triangular system $Rw = v$ then

$$(R^{-1} Q_2^T)_{ij} = u_j w_i, \quad i \geq j.$$

In [2] it is shown that if the strictly upper triangular part of R is the strictly upper triangular part of a rank-2 matrix then the triangular system $Rw = v$ can be solved in $O(n)$ operations. However this method would require products of ψ_i in u and v which may cause underflow and overflow problems. Instead, as in Section 2.1 we scale $Rw = v$ using the diagonal matrix D (2.4) in order to avoid products of ψ_i , to give $R'w' = v'$ where

$$R' = D^{-1} R D, \quad w' = D^{-1} w, \quad v' = D^{-1} v = [\phi_1, \dots, \phi_{n-1}, 1]^T.$$

The scaled system can also be solved in $O(n)$ operations due to the structure of D . Given R , D and v' , the following solves $R'w' = v'$ where $R' = D^{-1} R D$.

$$(3.6) \quad \begin{aligned} &1 \quad w'_n = v'_n / \tau_n \\ &2 \quad s' = (-1)^{n-1} \psi_{n-1} w'_n [b_n \ g_n]^T \\ &3 \quad w'_{n-1} = (v'_{n-1} - (-1)^{n-2} [a_{n-1} \ f_{n-1}] s') / \tau_{n-1} \\ &4 \quad \text{for } i = n-2: -1: 1 \\ &5 \quad \quad s' = (s' + (-1)^i w'_{i+1} [b_{i+1} \ g_{i+1}]^T) \psi_i \\ &6 \quad \quad w'_i = (v'_i - (-1)^{i-1} [a_i \ f_i] s') / \tau_i \\ &7 \quad \text{end} \end{aligned}$$

Let

$$u' = [1, \phi_1, \dots, \phi_{n-1}]^T.$$

Since $w = Dw'$ and $u = D^{-1}u'$, the diagonal elements of $R^{-1}Q_2^T$ are given by

$$(3.7) \quad u_i w_i = u'_i w'_i, \quad i = 1:n.$$

The strictly lower triangular elements are given by

$$(3.8) \quad u_j w_i = (-1)^{i+j-1} \psi_j \dots \psi_{i-1} u'_j w'_i, \quad i > j.$$

The lower triangular part of $R^{-1}Q_2^T$ is therefore defined by the three vectors, u' , w' and ψ .

The structure of $\text{tril}(R^{-1}Q_2^T, 0)$ and Q_1^T can now be exploited to find the j th absolute column sum of the strictly lower triangular part of $A^{-1} = R^{-1}Q_2^T Q_1^T$. Using (3.7), (3.8) and (3.4), the (i, j) , $i > j$, element of A^{-1} is given by

$$\begin{aligned} (A^{-1})_{ij} &= \sum_{k=1}^j \left((-1)^{k+i-1} \prod_{l=k}^{i-1} \psi_l \right) u'_k w'_i r_k s_j + \left((-1)^{j+i} \prod_{l=j+1}^{i-1} \psi_l \right) u'_{j+1} w'_i t_j \\ &= \left((-1)^{i+j} \prod_{l=j+1}^{i-1} \psi_l \right) w'_i \left(s_j \sum_{k=1}^j \left((-1)^{j+k} \prod_{l=k}^j \psi_l \right) u'_k r_k + u'_{j+1} t_j \right). \end{aligned}$$

The j th absolute column sum of the strictly lower triangular part of A^{-1} is therefore given by

$$\sigma_j = \left| \sum_{i=j+1}^n \left((-1)^{j+i} \prod_{l=j+1}^{i-1} \psi_l \right) w'_i \right| \cdot \left| s_j \sum_{k=1}^j \left((-1)^{k+j} \prod_{l=k}^j \psi_l \right) u'_k r_k + u'_{j+1} t_j \right|.$$

By considering

$$x'_j = \sum_{i=j+1}^n \left((-1)^{j+i} \prod_{l=j+1}^{i-1} \psi_l \right) w'_i$$

and

$$y'_j = s_j \sum_{k=1}^j \left((-1)^{k+j} \prod_{l=k}^j \psi_l \right) u'_k r_k + u'_{j+1} t_j,$$

separately, given w' , u' , ψ , r , s and t we can compute σ_j for $j = 1:n-1$ in $O(n)$ operations as follows:

$$(3.9) \quad \begin{array}{l} 1 \quad \psi = -\psi \\ 2 \quad y'_1 = \psi_1 u'_1 r_1 \\ 3 \quad x'_n = |w'_n| \\ 4 \quad \text{for } i = 2:n-1 \\ 5 \quad \quad y'_i = (y'_{i-1} + u'_i r_i) \psi_i \\ 6 \quad \quad x'_{n-i} = |x'_{n-i+1} \psi_{n-i+1} + w'_{n-i+1}| \\ 7 \quad \text{end} \\ 8 \quad \text{for } i = 1:n-1 \\ 9 \quad \quad \sigma_i = x'_i |y'_i s_i + u'_{i+1} t_i| \\ 10 \quad \text{end} \end{array}$$

In order to find the j th absolute column sum of the upper triangular part of A^{-1} , $\tilde{\sigma}_j$, we can consider $\tilde{A} = JAJ$, where J has only ones on the antidiagonal. By swapping p and q with y and x respectively, reversing the order of the vectors z , p , q , x and y , and setting $z_i = x_i y_i + z_i - p_i q_i$ for $i = 1:n$, we can repeat the above

process on \tilde{A} to find the absolute column sums of the strictly lower triangular part of \tilde{A}^{-1} and hence the absolute column sums of the strictly upper triangular part of A^{-1} .

We have shown how to find the strictly lower and strictly upper triangular parts of A^{-1} , and it remains to find the diagonal elements of A^{-1} . Given the strictly lower and strictly upper triangular parts of A^{-1} it is possible to use $AA^{-1} = I$ to find the diagonal elements of A^{-1} in $O(n)$ operations. However, this requires division by the diagonal elements of A , and the diagonal may contain zeros or elements close to zero that cause either breakdown of the algorithm or inaccurate results.

An alternative is to consider $RA^{-1} = Q_2^T Q_1^T$ and make use of

$$R(i, :)A^{-1}(:, i) = (Q_2^T Q_1^T)_{ii}.$$

The diagonal elements of A^{-1} therefore satisfy

$$(3.10) \quad (A^{-1})_{ii} = \begin{cases} \frac{(Q_2^T Q_1^T)_{ii} - R(i, i+1:n)A^{-1}(i+1:n, i)}{\tau_i}, & i < n, \\ \frac{(Q_2^T Q_1^T)_{ii}}{\tau_i}, & i = n. \end{cases}$$

It is possible to divide by $\tau_i \equiv R(i, i)$ since $\tau_i \neq 0$ as A is nonsingular. The structure of Q_1^T and Q_2^T is described in Proposition 3.1 and Theorem 2.1 respectively, and it is not difficult to check that given that we have used the code in (3.6) and (3.9), the following computes $c_i := (Q_2^T Q_1^T)_{ii}$ for $i < n$ in $O(n)$ operations.

$$(3.11) \quad \begin{array}{l} 1 \quad s' = w'_n [b_n \ g_n] \\ 2 \quad c_{n-1} = y'_{n-1} s' [a_{n-1} \ f_{n-1}]^T \\ 3 \quad \text{for } i = n-2: -1: 1 \\ 4 \quad \quad s' = \psi_{i+1} s' + w'_{i+1} [b_{i+1} \ g_{i+1}] \\ 5 \quad \quad c_i = y'_i s' [a_i \ f_i]^T \\ 6 \quad \text{end} \end{array}$$

Similarly, by considering the structure of R and the strictly lower triangular part of A^{-1} , we can compute $\tilde{c}_i := R(i, i+1:n)A^{-1}(i+1:n, i)$ for $i \leq n$ in $O(n)$ operations as follows.

$$(3.12) \quad \begin{array}{l} 1 \quad \tilde{c}_1 = u'_1 a_1 \\ 2 \quad \text{for } i = 2: n \\ 3 \quad \quad \tilde{c}_i = \tilde{c}_{i-1} \psi_{i-1} + u'_i a_i \\ 4 \quad \text{end} \\ 5 \quad \text{for } i = 1: n-1 \\ 6 \quad \quad \tilde{c}_i = \tilde{c}_i v_i s_i + \psi_i t_i \\ 7 \quad \text{end} \\ 8 \quad \tilde{c}_n = \tilde{c}_n v_n s_n \end{array}$$

The absolute column sums and the absolute value of the diagonal of A^{-1} can now be combined to give

$$\|A^{-1}\|_1 = \max_{i=1:n} (\sigma_i + \tilde{\sigma}_i + |(A^{-1})_{ii}|).$$

The following algorithm computes $\|A^{-1}\|_1$ in $O(n)$ operations.

ALGORITHM 3.1. *Computes $\xi = \|A^{-1}\|_1$, where A is an $n \times n$ dpss matrix given by (3.1) with $q_n \neq 0$ and $x_1 \neq 0$.*

- 1 Compute vectors in (3.2), (3.3) and (3.4) efficiently
- 2 for $i = 1: n-1$
- 3 $(\phi_i, \psi_i, \tau_i) = \text{Givens}(a_i b_i + f_i g_i, h_i)$ and let G_i be given by (2.3)

```

4      $a = G_i a, f = G_i f$ 
5      $u_{i+1} = \phi_i, v_i = \phi_i$ 
6 end
7  $\tau_n = a_n b_n + f_n g_n$ 
8 Solve  $R'w' = v'$  using the code in (3.6)
9 Compute  $\sigma_i$  for  $i < n$  using the code in (3.9)
10 Set  $\tilde{\sigma} = \sigma$  and repeat above on  $\tilde{A}$ 
11 Compute  $(Q_2^T Q_1^T)_{ii}$  using the code in (3.11)
12 Compute  $R(i, i+1 : )A^{-1}(i+1 : n, i)$  using the code in (3.12)
13 Compute  $(A^{-1})_{ii}$  for  $i \leq n$  using (3.10)
14  $\xi = \max_{i=1:n}(\tilde{\sigma}_i + \sigma_{n-i+1} + |(A^{-1})_{n-i+1, n-i+1}|)$ 

```

Cost: Assuming the function *Givens* requires 6 operations, the total cost is $133n$ operations.

The restrictions $q_n \neq 0$ and $x_1 \neq 0$ in Algorithm 3.1 can easily be removed by considering the structure of A^{-1} . For example, if q_i, \dots, q_n are all zero, then the last $n - i + 1$ columns of the strictly lower triangular part of A^{-1} are zero and $(A^{-1})_{jj} = 1/z_i$ for $j = i:n$. Therefore, the absolute column sums of the strictly lower triangular part of A^{-1} can be computed by applying lines 1–9 on $A(1:i-1, 1:i-1)$.

3.2. Numerical experiments. To test Algorithm 3.1 we gave the defining vectors of a dpss matrix random elements in $[-1, 1]$. We tested the resulting dpss matrix and the matrices obtained by raising the components of the defining vectors to the powers 2 to 6 elementwise. This was repeated twenty times to give 120 test matrices with condition numbers varying from approximately 10^4 to 5×10^{25} . The condition numbers of these test matrices were computed using Algorithm 3.1 and by forming the dpss matrices and using MATLAB's `cond`. The quantity $\beta = (\xi - \kappa_1(A))/\kappa_1(A)^2$ was computed, where ξ denotes the condition number computed using Algorithm 3.1 and $\kappa_1(A)$ denotes the condition number computed using `cond`, and they are shown in Figure 3.1. We divide by $\kappa_1(A)^2$ since the condition number of computing $\kappa_1(A)$ is $\kappa_1(A)$ [10]. Thus we expect $\beta = O(u)$ for a forward stable method, where $u \approx 10^{-16}$ is the unit roundoff. The results show that Algorithm 3.1 performs in a forward stable way.

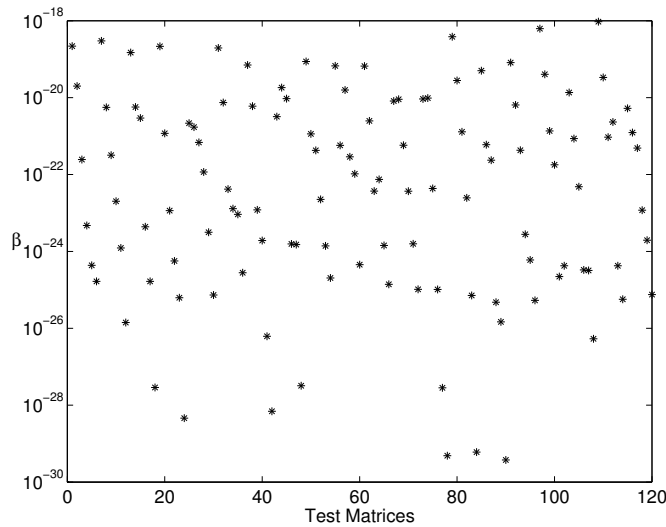


FIG. 3.1. $\beta = (\xi - \kappa_1(A))/\kappa_1(A)^2$ for 120 test matrices $A \in \mathbb{R}^{100 \times 100}$ with varying condition numbers.

The test matrices described above are all full matrices. Diagonal-plus-semiseparable matrices with various components of the defining vectors set to zero were also tested for which accurate results were obtained.

The condition number of a general real matrix can be estimated in $O(n^2)$ operations using the LAPACK [1] routine `DLACON`. The algorithm estimates $\|B\|_1$ iteratively, for an arbitrary B , by computing the matrix-vector products Bx and $B^T y$ at each iteration for carefully chosen x and y . No more than five iterations are usually required [12, Sec. 15.3]. Therefore if we have a factorization of A such as an LU factorization or a QR factorization then we can form the matrix-vector products for $B = A^{-1}$ in $O(n^2)$ operations and hence estimate $\|A^{-1}\|_1$ in $O(n^2)$ operations.

In [2], given the defining vectors of a dpss matrix, A , the QR factorization of A is considered to give an algorithm for solving the linear system $Ax = b$ in $O(n)$ operations. By storing the required vectors that define the QR factorizations of A and A^T , the LAPACK algorithm can be adapted to estimate $\|A^{-1}\|_1$ in $O(n)$ operations. We emphasize that this method only *estimates* $\|A^{-1}\|_1$ and that inaccurate estimates can easily occur.

We have found experimentally that estimating the 1-norm of the inverse of a dpss matrix in this way requires two iterations, but this still requires $177n$ operations which, is $44n$ more than that required for Algorithm 3.1. Table 3.1 shows the times taken in seconds to estimate $\|A^{-1}\|_1$ using an adapted version of the LAPACK algorithm for dpss matrices and the times taken to compute $\|A^{-1}\|_1$ using Algorithm 3.1. The algorithms were implemented using Fortran 95 and run on a 2010MHz AMD Athlon machine. The results show that Algorithm 3.1, which actually computes $\|A^{-1}\|_1$, is quickest.

TABLE 3.1

Time taken in seconds to compute the 1-norm of the inverse of a random $n \times n$ dpss matrix, for various n .

Algorithm	$n = 5 \times 10^5$	$n = 10^6$	$n = 2 \times 10^6$	$n = 5 \times 10^6$
Algorithm 3.1	0.78	1.55	3.04	7.44
LAPACK style	1.15	2.23	4.52	10.83

4. Conclusion. We have presented two algorithms that compute the condition number of a tridiagonal matrix in $O(n)$ operations. The algorithms avoid underflow and overflow and do not require tests for degenerate cases as in [6]. The second algorithm is marginally slower than the quickest algorithm in [6], but is faster than the recommended algorithm in [6]. An $O(n)$ algorithm to compute the condition number of a diagonal-plus-semiseparable matrix has also been given. Not only does this compute the condition number exactly, but it is also significantly quicker than a specialized implementation of the LAPACK condition number estimator.

5. Acknowledgements. I would like to thank Nick Higham for his invaluable comments on this work.

REFERENCES

- [1] E. ANDERSON, Z. BAI, C. BISCHOF, S. BLACKFORD, J. DEMMEL, J. DONGARRA, J. D. CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORESENSEN, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, third ed., 1999.
- [2] M. VAN BAREL, E. VAN CAMP, AND N. MASTRONARDI, *Two fast algorithms for solving diagonal-plus-semiseparable linear systems*, J. Comput. Appl. Math., 164–165 (2004), pp. 731–747.

- [3] M. VAN BAREL, L. G. D. FASINO AND, AND N. MASTRONARDI, *Orthogonal rational functions and structured matrices.*, Tech. Report TW350, Katholieke Universiteit Leuven, Belgium, 2002.
- [4] D. A. BINI, L. GEMIGNANI, AND F. TISSEUR, *The Ehrlich-Aberth method for the nonsymmetric tridiagonal eigenvalue problem*, Numerical Analysis Report No. 428, Manchester Centre for Computational Mathematics, Manchester, England, June 2003. To appear in SIAM J. Matrix Anal. Appl.
- [5] S. CHANDRASEKARAN AND M. GU, *Fast and stable algorithms for banded plus semiseparable matrices*, SIAM J. Matrix. Anal. Appl., 25 (2003), pp. 373–384.
- [6] I. DHILLON, *Reliable computation of the condition number of a tridiagonal matrix in $O(n)$ time*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 776–796.
- [7] Y. EIDELMAN AND I. GOHBERG, *A modification of the Dewilde-van der Veen method for inversion of finite structured matrices*, Linear Algebra and its Applications, 343–344 (2001), pp. 419–450.
- [8] P. E. GILL, G. H. GOLUB, W. MURRAY, AND M. A. SAUNDERS, *Methods for modifying matrix factorizations*, Mathematics of Computation, 28 (1974), pp. 505–535.
- [9] L. GREENGARD AND V. ROKHLIN, *On the solution of two-point boundary value problems*, Comm. Pure Appl. Math., 44 (1991), pp. 419–452.
- [10] D. J. HIGHAM, *Condition numbers and their condition numbers*, Linear Algebra Appl., 214 (1995), pp. 193–213.
- [11] N. J. HIGHAM, *Efficient algorithms for computing the condition number of a tridiagonal matrix*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 150–165.
- [12] ———, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second ed., 2002.
- [13] *IEEE standard for binary floating point arithmetic*, Tech. Report Std. 754–1985, IEEE/ANSI, New York, 1985.
- [14] Y. IKEBE, *On inverses of Hessenberg matrices*, Linear Algebra and its Applications, 24 (1979), pp. 93–97.
- [15] S.-Y. KANG, I. KOLTRACHT, AND R. RAWITSCHER, *Nyström–Clenshaw–Curtis quadrature for integral equations with discontinuous kernels*, Mathematics of Computation, 72 (2003), pp. 729–756.
- [16] J. LEE AND L. GREENGARD, *A fast adaptive numerical method for stiff two-point boundary value problems*, SIAM Journal on Scientific Computing, 18 (1997), pp. 403–429.
- [17] N. MASTRONARDI, S. CHANDRASEKARAN, AND S. VAN HUFFEL, *Fast and stable two-way algorithm for diagonal plus semi-separable systems of linear equations*, Numerical Linear Algebra with Applications, 8 (2001), pp. 7–12.
- [18] H. P. STARR, *On the numerical solution of one-dimensional integral and differential equations*, PhD thesis, Yale University, 1991.
- [19] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, NY, USA, 1973.
- [20] R. VANDEBRIL, M. VAN BAREL, AND N. MASTRONARDI, *An orthogonal similarity reduction of a matrix to semiseparable form.*, Tech. Report TW355, Katholieke Universiteit Leuven, Belgium, 2003. To appear in SIAM J. Matrix Anal. Appl.