Matrix Condition Numbers

Higham, Nicholas J.

1983

MIMS EPrint: **2016.45**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

MATRIX CONDITION NUMBERS

N.J. HIGHAM

OCTOBER 1983

A thesis submitted to the
University of Manchester
for the degree of Master in
the Faculty of Science.

ACKNOWLEDGEMENTS

It is a pleasure to acknowledge the advice and encouragement of my supervisor Dr G. Hall, as well as stimulating discussions with other members of the Numerical Analysis Department. I am grateful to Professor G.H. Golub and Dr J.R. Cash for suggesting the work contained in Chapter 3.

The support of a SERC Research Studentship is acknowledged.

I thank Mrs Vivien Gell for her adept typing of this thesis.

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

ABSTRACT

Several properties of matrix norms and condition numbers are described. The sharpness of the norm bounds in the standard perturbation results for $A\underline{x} = \underline{b}$ is investigated. For perturbations in A the bounds are sharp and quite likely to be realistic. For perturbations in $\underline{b}$ the usual bound is not sharp and can be unduly pessimistic; a more suitable measure of the conditioning than cond(A) is suggested.

Some important concepts relating to the problem of condition estimation are discussed, careful consideration being given to the reliability and computational cost of condition estimators. The LINPACK condition estimation algorithm is described, its weaknesses, including two counter-examples, pointed out, and some observations given.

Let A be an nxn tridiagonal matrix. We show that it is possible to compute $||A^{-1}||_\infty$, and hence $\text{cond}_\infty(A)$, in O(n) operations. Several algorithms which perform this task are given. All but one of the algorithms apply to irreducible tridiagonal matrices: those having no zero elements on the subdiagonal and superdiagonal. It is shown how these algorithms may be employed in the computation of $||A^{-1}||_\infty$ when A is reducible.

If A is, in addition, positive definite then it is possible to compute $||A^{-1}||_\infty$ as the ℓ_∞ norm of the solution to a linear system involving A's comparison matrix, M(A), which is also positive definite and tridiagonal. Utilising a relation between the LDL^T factors of A and M(A) we show how the LINPACK routine SPTSL, which solves $A\underline{x} = \underline{b}$ for positive definite tridiagonal matrices A, can be modified so that it also computes $\text{cond}_\infty(A)$, the increase in computational cost being approximately 60 percent.

CONTENTS

Chapter 1. Matrix Norms and Condition Numbers

1.1 Introduction

The condition number (with respect to the problem of matrix inversion) of a nonsingular matrix A is

$$\text{cond}_p(A) := ||A||_p ||A^{-1}||_p, \tag{1.1.1}$$

where $||.||_p$ denotes a given matrix norm. The importance of the condition number stems from the prominent role played by cond(A) in results which measure the sensitivity of the solution $\underline{x}$ of the linear system

$$A\underline{x} = \underline{b} \tag{1.1.2}$$

to changes in the data A and $\underline{b}$. Perturbations in A and $\underline{b}$ in (1.1.2), measured relative to A and $\underline{b}$ respectively, may give rise to relative changes in $\underline{x}$ which are cond(A) times as large. For example, if A is perturbed to A + E and if

$$(A + E)(\underline{x} + \underline{h}) = \underline{b},$$

then one can show that

$$\frac{||\underline{h}||}{||\underline{x} + \underline{h}||} \leqslant \text{cond}(A) \frac{||E||}{||A||}.$$

In Chapter 1 we explore various characterisations of the condition number. We derive several perturbation results for (1.1.2) and we investigate the likelihood that the bounds which are obtained are realistic.

Computation of cond(A) for a general n x n matrix A is rather expensive, costing $O(n^3)$ floating-point operations ("flops" - defined more precisely in section 2.1). Hence researchers have devised methods which compute a relatively cheap estimate of the condition number, given some factorisation of the matrix A. We survey some of this work in Chapter 2 and examine the reliability of these condition estimators.

We show in Chapter 3 that if A is a tridiagonal matrix then it is possible to compute $||A^{-1}||$, and hence cond(A), exactly in O(n) operations (here and for the rest of this section the norm is the infinity-norm or the one-norm). This is an appreciable saving on the $O(n^2)$ operations which would be required to compute $||A^{-1}||$ by first forming A^{-1} and then taking the norm.

In section 3.2 we describe some special properties of the inverse of a bidiagonal matrix B. In particular, we show how cond(B) can be computed in 2n-1 flops. In sections 3.3, 3.4 and 3.5 we derive three algorithms for the efficient computation of cond(A) when A is an irreducible tridiagonal matrix (a tridiagonal matrix without any zeros on the subdiagonal and superdiagonal). The first of these algorithms is derived using the special form of the inverses of the bidiagonal LU factors of A. If A is irreducible but has some small superdiagonal or subdiagonal elements, these algorithms may suffer from numerical stability problems. An algorithm is given in section 3.8 which, while being more expensive, has excellent numerical stability properties.

In section 3.7 we describe an efficient method for computing cond(A) when A is a general tridiagonal matrix; this method utilises the algorithms for the irreducible case.

If A is tridiagonal and positive definite then there is a more efficient way of computing cond(A). This method, which only requires the solution of one tridiagonal system of equations, is described in section 3.6. We show how the LINPACK routine SPTSL, which solves $A\underline{x} = \underline{b}$ for positive definite tridiagonal A, can be modified so that it also computes cond(A); only minor modifications are necessary, the computational cost of the routine being increased by roughly 60 percent.

1.2 Matrix Norms

Let $\mathbb{C}^{nxn}$ denote the set of all nxn matrices with complex elements. Throughout this section, $A := (a_{ij}) \in \mathbb{C}^{nxn}$.

A matrix norm on $\mathbb{C}^{nxn}$ is a function $||.|| : \mathbb{C}^{nxn} \to \mathbb{R}$ satisfying, for any $A, B \in \mathbb{C}^{nxn}$, the four conditions

$$||A|| \geq 0, \quad ||A|| = 0 \text{ iff } a_{ij} = 0 \text{ for all } i, j, \tag{1.2.1}$$

$$||\alpha A|| = |\alpha| \; ||A|| \text{ for all } \alpha \in \mathbb{C}, \tag{1.2.2}$$

$$||A + B|| \leq ||A|| + ||B|| \tag{1.2.3}$$

and

$$||AB|| \leq ||A|| \; ||B||. \tag{1.2.4}$$

The norm ||.|| thus collapses the information contained in the n-squared elements of $A \in \mathbb{C}^{nxn}$ into a single number, ||A||, which measures the size of the matrix A in an intuitively natural way, as indicated by (1.2.1) - (1.2.4).

If the function $||.||$ satisfies conditions (1.2.1), (1.2.2) and (1.2.3) (but not necessarily (1.2.4)) then, Lancaster [21, p. 198], $||.||$ is called a generalised matrix norm. Any vector norm ν on $\mathbb{C}^n$ gives rise to a generalised matrix norm on $\mathbb{C}^{nxn}$ (see [21, p. 203] for the definition of vector norm). To see this, define for $X := (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n) \in \mathbb{C}^{nxn}$

$$str(X) := (\underline{x}_1^T, \underline{x}_2^T, \ldots, \underline{x}_n^T)^T \in \mathbb{C}^{n^2}.$$

Thus str denotes the operation of "stretching out" a matrix into one long vector, formed from the columns of the matrix. Then $||.||$, defined by

$$||X|| := \nu(str(X)),$$

is easily seen to be a generalised matrix norm. This approach yields the following three generalised matrix norms:

$$||A||_S := \sum_{i,j} |a_{ij}|, \tag{1.2.5}$$

$$||A||_F := \left(\sum_{i,j} |a_{ij}|^2\right)^{\frac{1}{2}} \tag{1.2.6}$$

$$||A||_{M} := \max_{i,j} |a_{ij}|, \tag{1.2.7}$$

corresponding to ν being in turn the ℓ_1, ℓ_2 and ℓ_∞ vector norms. The norm in (1.2.6) is called the Frobenius norm, also referred to as the Euclidean or Schur norm. The first two of the generalised matrix norms above satisfy (1.2.4) and so are matrix norms. $||.||_M$ is not a matrix norm since (1.2.4) is not satisfied for $A = B = \underline{e}\underline{e}^T$, where $\underline{e} := (1,1,\ldots,1)^T$, but

$$||A||_M := n \max_{i,j} |a_{ij}| \tag{1.2.8}$$

is a matrix norm. More generally, if $||.||$ is a generalised matrix norm then there exists a scalar σ such that $\sigma||.||$ is a matrix norm [18, p. 61].

Each of the three matrix norms in most common use in numerical analysis is a subordinate matrix norm. For any vector norm ν the subordinate matrix norm $||.||$ is defined by

$$||A|| := \sup_{\underline{x} \neq \underline{0}, \underline{x} \in \mathbb{C}^n} \frac{\nu(A\underline{x})}{\nu(\underline{x})}. \tag{1.2.9}$$

Conditions (1.2.1) - (1.2.3) are easily seen to be satisfied. From the definition (1.2.9),

$$\nu(A\underline{x}) \leq ||A||\, \nu(\underline{x}), \tag{1.2.10}$$

which leads to (1.2.4), so $||.||$ is indeed a matrix norm.

A compactness argument can be used to show that there is a vector $\underline{x}$, depending on A and ν, for which equality is attained in (1.2.9) [21, p. 208]. This fact is important for the condition estimation procedures described in Chapter 2.

Taking for ν in (1.2.9) the ℓ_1, ℓ_2 and ℓ_∞ vector norms we obtain the three aforementioned popular matrix norms:

$$||A||_1 = \max_j \sum_i |a_{ij}| \quad - \text{"max column-sum"}, \tag{1.2.11}$$

$$||A||_\infty = \max_i \sum_j |a_{ij}| \quad - \text{"max row-sum"} \tag{1.2.12}$$

and

$$||A||_2 = \rho(A^*A)^{\frac{1}{2}} \quad - \text{"spectral norm"}, \tag{1.2.13}$$

where ρ denotes the spectral radius and * the conjugate transpose. Proofs of the equalities (1.2.11) - (1.2.13) may be found in Froberg [12, pp. 69-72].

From (1.2.9), a subordinate matrix norm must satisfy $||I|| = 1$. $||I||_p > 1$ for p = S,M,F so these three matrix norms are clearly not subordinate to any vector norm.

If a matrix norm and vector norm are related in such a way that (1.2.10) is satisfied for any A and $\underline{x}$, then the two norms are said to be consistent. The following consistency relations hold:

matrix norm p	- consistent with -	vector norm ν
S		$\ell_1, \ell_2, \ell_\infty$
M		$\ell_1, \ell_2, \ell_\infty$
F		ℓ_2.

These are obtained by using some of the Lemma 1.2 inequalities (see below) in (1.2.10). For example,

$$||A\underline{x}||_2 \leq ||A||_2 \, ||\underline{x}||_2 \leq ||A||_F \, ||\underline{x}||_2.$$

However, in contrast to the situation for a vector norm and its subordinate matrix norm, these consistency relations are not sharp, that is for a given matrix A, there may not exist a vector $\underline{x}$ for which $\nu(A\underline{x}) = ||A||_p \nu(\underline{x})$.

There is a wide variety of different matrix norms. For example, if T is a nonsingular matrix and $||.||_p$ is any matrix norm, then

$$||A||_{T,p} := ||T^{-1}AT||_p$$

defines a new norm. Despite this profusion of matrix norms there is a

sense in which any two matrix norms are equivalent.

Theorem 1.1 (Generalised Matrix Norm Equivalence Theorem).

If $||.||_p$ and $||.||_q$ are any two generalised matrix norms on $\mathbb{C}^{nxn}$ then there is a positive constant γ_{pq} such that for all $A \in \mathbb{C}^{nxn}$

$$||A||_p \leq \gamma_{pq} ||A||_q.$$

Proof

See [21, p. 200]. □

The constant γ_{pq} in Theorem 1.1 will in general depend on n and could differ significantly from 1. However, for any pair of the norms introduced so far, $1 \leq \gamma_{pq} \leq n$, so these norms really do not differ very much, at least for small n. The precise relations between the matrix norms $||.||_p$, p = 1,2,∞,S,F,M are specified by the next lemma.

Lemma 1.2

Let the constants γ_{pq} be defined by

p \ q	1	2	∞	S	F	M
1	1	√n	n	1	√n	1
2	√n	1	√n	1	1	1
∞	n	√n	1	1	√n	1
S	n	n	n	1	n	1
F	√n	√n	√n	1	1	1
M	n	n	n	n	n	1.

Then for p,q = 1,2,∞,S,F,M

$$||A||_p \leq \gamma_{pq} ||A||_q \text{ for all } A \in \mathbb{C}^{nxn}$$

and there exists $A_{pq} \in \mathbb{C}^{nxn}$, $A_{pq} \neq 0$, such that

$$||A_{pq}||_p = \gamma_{pq} ||A_{pq}||_q .$$

Proof

The proofs are omitted - they are straightforward but tedious. We remark that each A_{pq} can be taken as the identity matrix, or in the form $\underline{x}\,\underline{y}^T$ where $\underline{x}, \underline{y} \in \{\underline{e}_i, \underline{e}\}$. □

1.3 Characterisation of the Condition Number

For the rest of this chapter we concentrate our attention on subordinate matrix norms and denote both the vector norm and the associated matrix norm by $||.||$.

An alternative formula for cond(A) is obtained from the observation

$$||A^{-1}|| := \max_{\underline{x}\neq\underline{0}} \frac{||A^{-1}\underline{x}||}{||\underline{x}||} = \max_{\underline{y}\neq\underline{0}} \frac{||\underline{y}||}{||A\underline{y}||} = \left(\min_{\underline{y}\neq\underline{0}} \frac{||A\underline{y}||}{||\underline{y}||} \right)^{-1} . \quad (1.3.1)$$

This yields

$$\text{cond}(A) = \frac{\max_{\underline{x}\neq\underline{0}} \frac{||A\underline{x}||}{||\underline{x}||}}{\min_{\underline{x}\neq\underline{0}} \frac{||A\underline{x}||}{||\underline{x}||}}$$

$$= \frac{\max_{||\underline{x}|| = 1} ||A\underline{x}||}{\min_{||\underline{x}|| = 1} ||A\underline{x}||} . \quad (1.3.2)$$

Hence cond(A) can be interpreted as the ratio of maximum stretch to minimum stretch when A operates on all vectors of unit length in the $||.||$ norm. It follows that $\text{cond}(A) \geq 1$, a result which is true for any matrix norm that satisfies $||I|| = 1$, since $||I|| = ||A^{-1}A|| \leq ||A^{-1}||\ ||A||$. For the ℓ_2 norm, $\text{cond}_2(A)$ determines the eccentricity of the ellipsoid into which A transforms the unit sphere of $\mathbb{C}^n$. Perfect conditioning, $\text{cond}_2(A) = 1$, corresponds to the ellipsoid being a sphere.

If A is a singular matrix we can loosely regard cond(A) as being infinite. The next theorem shows that if cond(A) is large, A must be near to a singular matrix in the sense that a relatively small perturbation to A, measured in the given norm, can produce singularity. In this theorem, the matrix norm is that subordinate to the vector p-norm $||\underline{x}||_p := (\sum_i |x_i|^p)^{1/p}$, $1 \leq p \leq \infty$.

Theorem 1.3

Let $A \in \mathbb{C}^{nxn}$ be a nonsingular matrix. Then

$$\frac{1}{\text{cond}_p(A)} = \min_{\substack{A+E \\ \text{singular}}} \frac{||E||_p}{||A||_p}, \quad 1 \leq p \leq \infty.$$

Proof

First note that if A+E is singular, then for some $\underline{x} \neq \underline{0}$, $(A + E)\underline{x} = \underline{0}$. Hence $\underline{x} = -A^{-1}E\underline{x}$, which implies that $||\underline{x}||_p \leq ||A^{-1}||_p\ ||E||_p\ ||\underline{x}||_p$, or $\frac{1}{||A^{-1}||_p} \leq ||E||_p$. We need only show that equality can occur for some E. We prove this for $p = 1,2,\infty$; the proof for general p may be found in [30, p. 169]. Professor C.F. Van Loan pointed out to us that this result is also contained in [20].

(a) ℓ_2 norm

Let $A = U\Sigma V^*$ be the singular value decomposition of A, where $\Sigma = \mathrm{diag}(\sigma_i)$, $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_n > 0$, $U := (\underline{u}_1,\ldots,\underline{u}_n)$, $V := (\underline{v}_1,\ldots,\underline{v}_n)$ and $U^*U = V^*V = I$. With $E := -\sigma_n \underline{u}_n \underline{v}_n^*$,

$$A + E = U\,\mathrm{diag}(\sigma_1,\sigma_2,\ldots,\sigma_{n-1},0)V^*,$$

so $A + E$ is singular and $||E||_2 = \sigma_n = ||A^{-1}||_2^{-1}$.

(b) ℓ_∞ norm

By (1.3.1),

$$||A^{-1}||_\infty^{-1} = \min_{\underline{y}\neq\underline{0}} \frac{||A\underline{y}||_\infty}{||\underline{y}||_\infty} =: \frac{||\underline{w}||_\infty}{||\underline{z}||_\infty},$$

where $\underline{w} = A\underline{z}$. Comparison with part (a) suggests the rank-one choice $E := \underline{w}\,\underline{u}^*$, where $\underline{u}$ is to be determined. The requirement $||E||_\infty = ||A^{-1}||_\infty^{-1}$ is equivalent to $||\underline{u}||_1 = ||\underline{z}||_\infty^{-1}$, since $||E||_\infty = ||\underline{w}||_\infty\,||\underline{u}||_1$. Also,

$$(A + E)\underline{z} = \underline{w} + (\underline{u}^*\underline{z})\underline{w}$$

$$= \underline{o} \quad \text{iff} \quad \underline{u}^*\underline{z} = -1.$$

The vector $\underline{u} := -\dfrac{\mathrm{sgn}(z_i)}{|z_i|}\,\underline{e}_i$, where $|z_i| = ||\underline{z}||_\infty$, satisfies both the required conditions and this completes the definition of E.

(c) ℓ_1 norm

The result of the theorem for the ℓ_1 norm is obtained from the result for the ℓ_∞ norm by replacing A by A^T and using $||A^T||_\infty = ||A||_1$. □

In words, Theorem 1.3 says that the reciprocal of $\text{cond}_p(A)$ is the relative distance from A to the nearest singular matrix, distance being measured in the ℓ_p matrix norm.

Example

The upper triangular matrix

$$T := \begin{bmatrix} 1 & -1 & \cdots & -1 \\ & 1 & -1 & \cdots & -1 \\ & & 1 & \ddots & \vdots \\ & & & \ddots & -1 \\ & & & & 1 \end{bmatrix} \in \mathbb{R}^{n\times n} \qquad (1.3.3)$$

has an inverse with ℓ_∞ norm 2^{n-1} [17, p. 14]. Wilkinson [34, p. 113] notes that if the (n,1) element of T is perturbed by -2^{2-n} then the perturbed matrix is singular, having rank n-1. This perturbation corresponds to a matrix F with norm $||F||_\infty = 2^{2-n}$. However, writing Theorem 1.3 in the form

$$\frac{1}{||A^{-1}||_p} = \min_{\substack{A+E \\ \text{singular}}} ||E||_p$$

we see that there must be a matrix G with $||G||_\infty = 2^{1-n}$, such that T + G is singular. By working through the proof of part (b) of the theorem for the matrix T one finds that $G = -\frac{1}{2^{n-1}} e\, e_1^T$, which does indeed have ℓ_∞ norm 2^{1-n}.

A lower bound for cond(A) may be obtained in terms of the eigenvalues of A. For any matrix norm $||B|| \geq \rho(B)$ [21, p. 201], so

$$\text{cond}(A) \geq \rho(A)\ \rho(A^{-1}) = \frac{\max\limits_i |\lambda_i|}{\min\limits_i |\lambda_i|}, \tag{1.3.4}$$

where $\{\lambda_i\}$ are the eigenvalues of A. This lower bound can be very poor. For a triangular matrix R (1.3.4) becomes

$$\text{cond}(R) \geq \frac{\max\limits_i |r_{ii}|}{\min\limits_i |r_{ii}|};$$

for R = T defined in (1.3.3) we have $\text{cond}_\infty(R) = n\ 2^{n-1}$, yet the lower bound is 1.

The ℓ_2 condition number of $A \in \mathbb{C}^{n\times n}$ can be expressed in terms of the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$ of A:

$$\text{cond}_2(A) = \frac{\sigma_1}{\sigma_n}.$$

For the special class of <u>normal</u> matrices, that is matrices satisfying $AA^* = A^*A$, the sets $\{\sigma_i\}_{i=1}^n$ and $\{|\lambda_i|\}_{i=1}^n$ are the same; therefore the lower bound of (1.3.4) is exact for the ℓ_2 norm when A is a normal matrix. Any matrix A which is Hermitian ($A^* = A$), skew-Hermitian ($A^* = -A$) or unitary ($A^*A = I$) is normal.

Since cond(A) is a product of two matrix norms it is clear that any two condition numbers are equivalent in the sense of Theorem 1.1. Indeed Lemma 1.2 implies that any two of the values $\{\text{cond}_p(A),\ p = 1,2,\infty,S,F,M\}$ can differ by at most a factor n^2.

1.4 Perturbation Results

In this section we derive several classical perturbation results for the linear system

$$A\underline{x} = \underline{b} \tag{1.4.1}$$

and investigate the sharpness of the bounds which are obtained.

We first note that $A\underline{x} = \underline{b}$ implies

$$||\underline{b}|| \leq ||A||\ ||\underline{x}|| \tag{1.4.2}$$

and

$$||\underline{x}|| \leq ||A^{-1}||\ ||\underline{b}||, \tag{1.4.3}$$

with equality possible in each case. Hence the range of values which the ratio $||\underline{x}||/||\underline{b}||$ can take is specified by

$$\frac{1}{||A||} \leq \frac{||\underline{x}||}{||\underline{b}||} \leq ||A^{-1}||. \tag{1.4.4}$$

It is clear from (1.4.4) that if cond(A) is large and either of the inequalities (1.4.2), (1.4.3) is approximately an equality, then the other inequality must be very slack. Let us write $\alpha \simeq \beta$ if the numbers α and β are of the same order of magnitude. We shall refer to a vector $\underline{x}$ for which $||\underline{x}|| \simeq ||A^{-1}||\ ||\underline{b}||$ as a large-normed solution to $A\underline{x} = \underline{b}$ and a vector $\underline{x}$ for which $||\underline{x}|| \simeq ||A||^{-1}\ ||\underline{b}||$ as a small-normed solution to $A\underline{x} = \underline{b}$. When A is well-conditioned these two concepts coincide.

There are three types of perturbation to consider for the problem (1.4.1): perturbations in $\underline{b}$, perturbations in A and simultaneous perturbations in A and $\underline{b}$.

Theorem 1.4

Let $A\underline{x} = \underline{b}$, where A is a nonsingular matrix.

(i) If $A(\underline{x} + \underline{h}) = \underline{b} + \underline{d}$ then

$$\frac{||\underline{h}||}{||\underline{x}||} \leq \text{cond}(A) \frac{||\underline{d}||}{||\underline{b}||}. \tag{1.4.5}$$

(ii) If $(A + E)(\underline{x} + \underline{h}) = \underline{b}$ and $||A^{-1}E|| < 1$ then

$$\frac{||\underline{h}||}{||\underline{x}||} \leq \frac{\text{cond}(A)}{1 - ||A^{-1}E||} \frac{||E||}{||A||}. \tag{1.4.6}$$

(iii) If $(A + E)(\underline{x} + \underline{h}) = \underline{b} + \underline{d}$ and $||A^{-1}E|| < 1$ then

$$\frac{||\underline{h}||}{||\underline{x}||} \leq \frac{\text{cond}(A)}{1 - ||A^{-1}E||} \left(\frac{||E||}{||A||} + \frac{||\underline{d}||}{||\underline{b}||} \right). \tag{1.4.7}$$

Proof

(i): $A\underline{h} = \underline{d}$ so $||\underline{h}|| \leq ||A^{-1}||\ ||\underline{d}||$. Multiplication of this inequality by (1.4.2) yields (1.4.5).

(ii): $(A + E)\underline{h} = -E\underline{x}$. The condition $||A^{-1}E|| < 1$ ensures that

$$||(A + E)^{-1}|| \leq \frac{||A^{-1}||}{1 - ||A^{-1}E||}, \tag{1.4.8}$$

by a standard result [27, p. 188]. Taking norms in $\underline{h} = -(A + E)^{-1}E\underline{x}$ leads to (1.4.6).

(iii): $(A + E)\underline{h} = \underline{d} - E\underline{x}$. This gives, using (1.4.8), $||\underline{h}|| \leq (1 - ||A^{-1}E||)^{-1}\ ||A^{-1}||\ (||\underline{d}|| + ||E||\ ||\underline{x}||)$. (1.4.7) is established by dividing throughout by $||\underline{x}||$ and then using (1.4.2). □

Remarks

(1) One can regard $\underline{x}_c := \underline{x} + \underline{h}$ as the computed solution to $A\underline{x} = \underline{b}$. Identifying the residual $\underline{r} := \underline{b} - A\,\underline{x}_c$ with $-\underline{d}$, (1.4.5) can be rewritten

$$\frac{||\underline{x} - \underline{x}_c||}{||\underline{x}||} \leq \text{cond}(A)\,\frac{||\underline{r}||}{||\underline{b}||}$$

- a bound for the relative error in the computed solution in terms of its residual.

(2) It is possible to coalesce the three parts of Theorem 1.4 by writing the perturbed system as $A(\underline{x} + \underline{h}) = \underline{b} + \underline{t}$ where $\underline{t}$ contains all the terms involving E and $\underline{d}$. Rice [25] adopts this cruder, but nonetheless informative approach.

Theorem 1.4 says, essentially, that perturbations of relative size ε in A or $\underline{b}$ can give rise to relative perturbations in $\underline{x}$ of size at most $\text{cond}(A)\varepsilon$. The proofs, which simply use norm inequalities, offer no immediate indication as to whether the error magnification factor cond(A) is likely to be achieved, or even can be achieved, for a particular A and $\underline{b}$. We now look at these questions in some detail for (1.4.5) and (1.4.6).

Perturbations in $\underline{b}$

The proof of part (i) of Theorem 1.4 begins by applying a norm inequality to the identity $A\underline{h} = \underline{d}$ to obtain

$$||\underline{h}|| \leq ||A^{-1}||\;||\underline{d}||. \tag{1.4.9}$$

This is equivalent to

$$\frac{||\underline{h}||}{||\underline{x}||} \leq \frac{||A^{-1}||\ ||\underline{b}||}{||\underline{x}||} \frac{||\underline{d}||}{||\underline{b}||},$$

that is

$$\frac{||\underline{h}||}{||\underline{x}||} \leq C_x \frac{||\underline{d}||}{||\underline{b}||}, \tag{1.4.10}$$

where $C_x := \dfrac{||A^{-1}||\ ||\underline{b}||}{||\underline{x}||}$ is the quantity which Rice [25] calls the natural condition number. The bounds

$$1 \leq C_x \leq \text{cond}(A) \tag{1.4.11}$$

follow from (1.4.4).

There is always some vector $\underline{d}$ for which (1.4.9) is an equality and for this $\underline{d}$, $(||\underline{h}||/||\underline{x}||)/(||\underline{d}||/||\underline{b}||) = C_x$. Hence C_x is the condition number for the problem $A\underline{x} = \underline{b}$ in the sense defined by Dahlquist and Bjorck [6, p. 54], where here, $\underline{b}$ is the input data and A is fixed. Note that we are measuring relative errors using the given norm here, thus a relative error of at most ε in the input data corresponds to $||\underline{d}|| \leq \varepsilon\ ||\underline{b}||$. If, instead, the requirement $|d_i| \leq \varepsilon\ |b_i|$ for all i is imposed, a different condition number results; see Skeel [26].

If A is well-conditioned then inequalities (1.4.9) and (1.4.10) must be sharp, by comparison with (1.4.4). Now suppose that A is badly-conditioned. Then if $\underline{d}$ is a random vector, the theory of section 2.2 implies that there is a high probability of (1.4.9) and (1.4.10) being sharp.

To summarise, for any A and $\underline{b}$, equality is obtained in (1.4.10) for some $\underline{d}$ and C_x is a realistic estimate of the relative error magnification factor $(||\underline{h}||/||\underline{x}||)/(||\underline{d}||/||\underline{b}||)$.

The bound (1.4.10) is not widely seen in the literature. This is probably because C_x depends on $\underline{x}$ and so (1.4.10) is an a posteriori bound. It is more common to replace C_x by the upper bound cond(A) thus obtaining (1.4.5), which is an a priori bound. However, if A is ill-conditioned, this replacement can severely weaken the inequality (1.4.10), as is clear from (1.4.11). The weakening is more severe when $\underline{x}$ is a large-normed solution; indeed if $||\underline{x}|| = ||A^{-1}||\ ||\underline{b}||$, that is $\underline{b}$ is a maximising vector for A^{-1} in the given norm, then $C_x = 1$ and (1.4.5) is unduly pessimistic.

The inequality (1.4.10) gives more insight into the conditioning of (1.4.1) with respect to perturbations in $\underline{b}$ than does (1.4.5) since it shows that for a given $\underline{b}$ the conditioning is better or worse according as $\underline{x}$ is a large-normed or small-normed solution. Hammarling and Wilkinson [16] show the interesting result that for the linear equations arising from the standard finite difference approximation applied to the differential equation $W''(x) = f(x)$, both small-normed solutions and large-normed solutions can be obtained by suitable choice of the forcing function f and the boundary conditions.

Perturbations in A

When the linear system (1.4.1) is solved on a computer by some form of Gaussian elimination, the computed solution $\underline{x}_c$ satisfies

$$(A + E)\underline{x}_c = \underline{b}, \tag{1.4.12}$$

where bounds are available for $||E||$ [27, p. 155]. Here, the second part of Theorem 1.4 is applicable. An alternative bound to (1.4.6) is obtained by subtracting (1.4.12) from (1.4.1) to give

$$A(\underline{x} - \underline{x}_c) = E\underline{x}_c$$

and hence

$$\underline{x} - \underline{x}_c = A^{-1}E\underline{x}_c. \tag{1.4.13}$$

Standard norm inequalities lead to

$$\frac{||\underline{x} - \underline{x}_c||}{||\underline{x}_c||} \lesssim \text{cond}(A)\,\frac{||E||}{||A||}, \tag{1.4.14}$$

which resembles (1.4.6) except that the error in the computed solution is measured relative to the computed solution rather than to the exact solution and no condition on $||A^{-1}E||$ is required. In view of our comments about (1.4.10), it is natural to ask whether the quality of (1.4.14) is dependent on $\underline{x}$ being a large-normed or small-normed solution. There does not appear to be any reason why this should be so. In the case of Gaussian elimination, the standard backward error analysis shows that E depends on $\underline{b}$ but the major contribution to E comes from the factorisation stage which is independent of $\underline{b}$.

Further insight can be gained by using the ideas of section 2.2. We are interested in the case where A is ill-conditioned. Defining $\underline{z} := E\underline{x}_c$ we have from (1.4.13),

$$\begin{aligned} ||\underline{x} - \underline{x}_c|| &= ||A^{-1}\underline{z}|| \\ &\simeq ||A^{-1}||\ ||\underline{z}|| \end{aligned} \tag{1.4.15}$$

with high probability, assuming that $\underline{z}$ is a random vector. Also, it seems reasonable to assume that

$$||\underline{z}|| \simeq ||E||\ ||\underline{x}_c||. \tag{1.4.16}$$

Combining (1.4.15) and (1.4.16) we obtain

$$||\underline{x} - \underline{x}_c|| = ||A^{-1}\underline{z}|| \simeq ||A^{-1}||\ ||E||\ ||\underline{x}_c||,$$

which says that the upper bound in (1.4.14) is of the correct order of magnitude.

There are special cases for which the upper bound in (1.4.14) is a severe overestimate if A is ill-conditioned, for example when E is a scalar multiple of A or $E\underline{x}_c = \underline{0}$. Nevertheless, the above arguments suggest that (1.4.14) is quite realistic in general, and in particular for the Gaussian elimination application. This view is confirmed by Cline, Moler, Stewart and Wilkinson [5], by Stewart [27, p. 195] and by Hammarling and Wilkinson [16] who state

> "the error analysis suggests that whatever the nature of the right-hand side $\underline{b}$, the accuracy of the computed solution will fully reflect the condition of the matrix of coefficients."

and present numerical evidence to support this statement.

We have not yet considered whether equality can be attained in (1.4.6) and (1.4.14). Consider the derivation of (1.4.6). We have

$$A(I + A^{-1}E)\underline{h} = (A + E)\underline{h} = -E\underline{x}.$$

Hence, recalling that $||A^{-1}E|| < 1$,

$$\underline{h} = -(I + A^{-1}E)^{-1}A^{-1}E\underline{x} = -A^{-1}E\underline{x} - \sum_{i=2}^{\infty} (-1)^{i+1}(A^{-1}E)^i\underline{x}.$$

Taking norms yields

$$||\underline{h}|| \lesssim ||A^{-1}||\ ||E||\ ||\underline{x}|| + O(||A^{-1}E||^2)\ ||\underline{x}||,$$

whence

$$\frac{||\underline{h}||}{||\underline{x}||} \lesssim \text{cond}(A)\ \frac{||E||}{||A||} + O(||A^{-1}E||^2). \qquad (1.4.17)$$

Our reason for deriving this inequality, which is very similar to (1.4.6) and to (1.4.14), is that there holds the following remarkable result of Van der Sluis [32]:

Theorem 1.5

For any vector norm and its subordinate matrix norm, for any A and $\underline{b}$, and for any γ such that $0 < \gamma < ||A^{-1}||^{-1}$, there exists E such that $||E|| = \gamma$ and equality holds in (1.4.17).

To prove the theorem we need

Lemma 1.6 see 1.19

For any $\underline{x}$, $\underline{y} \in \mathbb{C}^n$ and any vector norm, there exists $B \in \mathbb{C}^{n \times n}$ such that $\underline{y} = B\underline{x}$ and $\underline{x}$ is a maximising vector for B.

Proof

See Van der Sluis [31]. The proof uses the powerful Hahn-Banach theorem. □

Proof of Theorem 1.5

We need to find a matrix E such that $||A^{-1}E\underline{x}|| = ||A^{-1}||\ ||E||\ ||\underline{x}||$. Let $\underline{y}$ be a maximising vector for A^{-1}. Lemma 1.6 asserts the existence of a matrix B such that $\underline{y} = B\underline{x}$ and $\underline{x}$ is a maximising vector for B. The choice $E := \frac{\gamma}{||B||} B$ gives the sequence of equalities

$$||A^{-1}E\underline{x}|| = \frac{\gamma}{||B||}\,||A^{-1}B\underline{x}|| = \frac{\gamma}{||B||}\,||A^{-1}\underline{y}||$$

$$= \frac{\gamma}{||B||}\,||A^{-1}||\;||\underline{y}|| = \frac{\gamma}{||B||}\,||A^{-1}||\;||B||\;||\underline{x}||$$

$$= ||A^{-1}||\;||E||\;||\underline{x}||.$$

Finally, $||A^{-1}E|| \leq ||A^{-1}||\;||E|| = ||A^{-1}||\;\gamma < 1$, which is needed for (1.4.17). □

The inequalities (1.4.6), (1.4.14) and (1.4.17) are essentially the same, provided that $||A^{-1}E|| << 1$ (this implies $||\underline{x}||/||\underline{x}_c|| \simeq 1$). We conclude from Theorem 1.5 that if $||A^{-1}E|| << 1$ then the inequalities (1.4.6) and (1.4.14) are sharp, in the sense that for any A and $\underline{b}$ there is a perturbation matrix F, whose norm can be chosen arbitrarily as long as $||A^{-1}F|| << 1$, for which these inequalities are approximate equalities.

Summarising, the inequalities (1.4.6), (1.4.14) and (1.4.17) describing the worst possible effect on the solution $\underline{x}$ of a perturbation in A are quite realistic. We base this conclusion on the fact that these inequalities are sharp and on the heuristic, discussed above, that the bounds are quite likely to be of the correct order of magnitude. Of course, if A is well-conditioned, none of the inequalities under consideration can be very slack.

Chapter 2. Estimation of Matrix Condition Numbers

2.1 Computational Costs

When solving linear systems

$$A\underline{x} = \underline{b} \tag{2.1.1}$$

on a computer it is obviously desirable to know how accurate the computed solution is and how sensitive the true solution is to perturbations in the data. As we saw in Chapter 1, both these factors are to some extent governed by the condition number of the coefficient matrix. Therefore when solving (2.1.1) one would ideally like to not only have the computed solution, but also have the condition number of A available for inspection. The identification of a large condition number might in one case lead to the reformulation of the problem which gave rise to the linear system and in another case it might engender the realisation that the desired accuracy in the computed solution can only be obtained by recourse to higher precision arithmetic.

Unfortunately, computation of the condition number is rather expensive compared to computation of the solution of one linear system. As a measure of computational cost we use the term flop to denote both a combined floating-point multiplication/addition operation, corresponding to a FORTRAN statement of the form

```
S = S + A*B,
```

and a single floating-point multiplication, addition or division operation (cf. [22]). The nxn system (2.1.1) is usually solved in two stages. In the factorisation stage, Gaussian elimination with partial pivoting is

used to compute the factorisation

$$PA = LU. \tag{2.1.2}$$

In the substitution stage, the two triangular systems $L\underline{y} = P\underline{b}$ and $U\underline{x} = \underline{y}$ are solved, yielding the solution $\underline{x}$. (We are assuming here that all operations are exact). For a full matrix the factorisation stage costs $O(n^3/3)$ flops and the substitution stage costs $O(n^2)$ flops. The columns of A^{-1} are the solutions of the n equations $A\underline{x}_i = \underline{e}_i$, $i = 1,2,\ldots,n$, the computation of which, given (2.1.2), requires $O(2n^3/3)$ flops if the special nature of the right hand sides is taken into account and $O(n^3)$ flops otherwise. Therefore computation of cond(A) (by the obvious route of computing A^{-1} and evaluating $||A||\ ||A^{-1}||$) in addition to computation of the solution of (2.1.1) is roughly three or four times as expensive as computation of the solution of (2.1.1) alone.

However, once an LU factoriation has been computed for (2.1.1), extra right-hand sides can be processed at comparatively little extra cost. The condition estimation techniques in [4] and [5] take advantage of this. Given the factorisation (2.1.2) they compute an estimate of cond(A) in $O(n^2)$ flops.

One is usually only interested in the order of magnitude of the condition number. In this circumstance a good estimate which does not differ significantly from the true value should be acceptable. Cline, Conn and Van Loan [4] define a condition estimator γ to be reliable if there are constants c_1 and c_2, independent of A and of modest size, such that

$$c_1 \text{ cond}(A) \lesssim \gamma \lesssim c_2 \text{ cond}(A) \tag{2.1.3}$$

For the ℓ_1 and ℓ_∞ norms, $||A||$ can be computed cheaply using (1.2.11) and (1.2.12). For the ℓ_2 norm cheap, reliable estimates of $||A||_2$ are provided by either, or a combination, of the bounds (see [17])

$$\frac{1}{\sqrt{n}}\,\phi \leq ||A||_2 \leq \phi, \quad \phi := (||A||_1\ ||A||_\infty)^{\frac{1}{2}} \tag{2.1.4}$$

and

$$\alpha \leq ||A||_2 \leq ||A||_F \leq \sqrt{n}\ \alpha, \ \alpha := \max_i ||A\underline{e}_i||_2. \tag{2.1.5}$$

For these popular subordinate norms, the problem of condition estimation reduces to the problem of estimating

$$||A^{-1}|| = \max_{\underline{b}\neq\underline{0}} \frac{||A^{-1}\underline{b}||}{||\underline{b}||}. \tag{2.1.6}$$

The standard approach, [4], [5], is to look for a vector $\underline{c}$ such that $\underline{z}$ is a large-normed solution to $A\underline{z} = \underline{c}$. Then

$$||A^{-1}|| \simeq \frac{||\underline{z}||}{||\underline{c}||} \leq ||A^{-1}||. \tag{2.1.7}$$

Specifically, we would like to choose $\underline{c}$, normalised by $||\underline{c}|| = 1$, say, so that $||\underline{z}||$ is as large as possible, there being at least one vector $\underline{c}$ which gives the maximum value, $||A^{-1}||$.

If A is well-conditioned, the choice of $\underline{c}$ is not critical, for by (1.4.4) any $\underline{c}$ will yield a lower bound for $||A^{-1}||$ which is of the correct order of magnitude. Hence our main concern in the rest of this chapter is with the case where A is ill-conditioned.

2.2 Random Right-Hand Sides

A condition estimation strategy discussed by Rice [25] involves choosing $p << n$ random vectors $\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_p$ and taking the maximum of the corresponding lower bounds (2.1.7). The success of this approach rests on the validity of the heuristic that when A is ill-conditioned there is a high probability that one of the p solutions will be large-normed. In this section we investigate the properties of solutions which correspond to random right-hand sides.

Let $A \in \mathbb{R}^{n\times n}$ have the singular value decomposition

$$A = U\Sigma V^T, \tag{2.2.1}$$

where $\Sigma = \text{diag}(\sigma_i)$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n > 0$, $U := (\underline{u}_1, \ldots, \underline{u}_n)$, $V := (v_1, \ldots, v_n)$ and $U^TU = V^TV = I$. Any vector $\underline{b}$ can be expressed as

$$\underline{b} = \sum_{i=1}^{n} \alpha_i \underline{u}_i = U\underline{\alpha} \tag{2.2.2}$$

because the columns $\{\underline{u}_i\}$ of U span $\mathbb{R}^n$. Thus $A\underline{x} = \underline{b}$ implies

$$\underline{x} = A^{-1}\underline{b} = V\Sigma^{-1}U^T.U\underline{\alpha} = V\Sigma^{-1}\underline{\alpha},$$

that is

$$\underline{x} = \sum_{i=1}^{n} \frac{\alpha_i}{\sigma_i} \underline{v}_i.$$

Suppose $||\underline{b}||_2 = 1$, so that $||\underline{\alpha}||_2 = 1$. We have,

$$||\underline{x}||_2 = \left(\sum_{i=1}^{n} \frac{\alpha_i^2}{\sigma_i^2} \right)^{\frac{1}{2}}. \tag{2.2.3}$$

Thus $\underline{x}$ will be a large-normed solution, with $||\underline{x}||_2 \simeq \sigma_n^{-1}$, unless $|\alpha_n| << 1$ and $|\alpha_i| << 1$ for all i such that $\sigma_i \simeq \sigma_n$ - this is unlikely

for random $\underline{b}$ since this condition requires that $\underline{b}$ have very small components in the direction $\underline{u}_n$ and in the orthogonal directions corresponding to all the other "small" left singular vectors. To summarise,

Fact 2.1

For any nonsingular matrix A, if $\underline{b}$ is a random vector and $A\underline{x} = \underline{b}$ then there is a high probability that

$$||\underline{x}||_2 \simeq ||A^{-1}||_2 ||\underline{b}||_2. \quad \square$$

An interesting result is obtained by assuming that α_i in (2.2.2) is a random variable with $E(\alpha_i^2) = \theta^2$, $i = 1,2,\ldots n$. Then $||\underline{x}||_2^2$ is also a random variable and, from (2.2.3),

$$E(||\underline{x}||_2^2) = \theta^2 \sum_{i=1}^{n} \frac{1}{\sigma_i^2} = \theta^2 ||A^{-1}||_F^2. \tag{2.2.4}$$

The same result holds if

$$\underline{b} = \sum_{i=1}^{n} \alpha_i \underline{e}_i \tag{2.2.5}$$

(perhaps a more natural condition), though it is here necessary to further assume that the $\{\alpha_i\}$ are independent random variables with zero mean. For (2.2.2) and (2.2.5), $E(||\underline{b}||_2^2) = n\theta^2$. Thus we have

Fact 2.2

Let $A \in \mathbb{R}^{n\times n}$. Let the vector $\underline{b}$ be given in the form (2.2.2) in terms of the left singular vectors of A or in the form (2.2.5) in terms of the unit vectors, where the $\{\alpha_i\}$ are independent random variables with

$E(\alpha_i) = 0$ and $E(\alpha_i^2) = \theta^2$, $i = 1,2,\ldots,n$. Then

$$\frac{E(||\underline{x}||_2^2)}{E(||\underline{b}||_2^2)} = \frac{||A^{-1}||_F^2}{n},$$

or, to a rough approximation,

$$E\left(\frac{||\underline{x}||_2}{||\underline{b}||_2}\right) \cong \frac{||A^{-1}||_F}{\sqrt{n}}. \qquad \square$$

We remark that var $(||\underline{x}||_2^2)$ contains terms like σ_n^{-4}, which confirms the potential wide variability of $||\underline{x}||_2/||\underline{b}||_2$ expressed by (1.4.4).

A more precise result came to our attention at a late stage during the preparation of this thesis. Dixon [7] proves the following theorem:

Theorem 2.3

Let $A \in \mathbb{R}^{n\times n}$ be positive definite with eigenvalues $\lambda_1 \geqslant \ldots \geqslant \lambda_n$ and let $\theta > 1$ be a constant. If $\underline{x} \in \mathbb{R}^n$ is a random vector from the uniform distribution on the unit sphere $\{\underline{y} \in \mathbb{R}^n : \underline{y}^T\underline{y} = 1\}$ then the inequality

$$\underline{x}^T A \underline{x} \leqslant \lambda_1 \leqslant \theta\, \underline{x}^T A \underline{x}$$

holds with probability at least $1 - 0.8\, \theta^{-\frac{1}{2}} n^{\frac{1}{2}}$. $\square$

From this result we deduce

Theorem 2.4

Let $A \in \mathbb{R}^{nxn}$ be a nonsingular matrix. Let θ and $\underline{x}$ be as described in Theorem 2.3. Then the inequality

$$||A^{-1}\underline{x}||_2 \leq ||A^{-1}||_2 \leq \theta\, ||A^{-1}\underline{x}||_2 \qquad (2.2.6)$$

holds with probability at least $1 - 0.8\,\theta^{-1}n^{\frac{1}{2}}$.

Proof

Apply Theorem 2.3 to the positive definite matrix $(AA^T)^{-1}$. □

Vectors $\underline{x}$ from the above distribution can be generated from the formula

$$x_i := y_i/||\underline{y}||_2,$$

where $y_1,\ldots,y_n$ are independent N(0,1) distributed random variables [7]. For this particular distribution, Theorem 2.4 places Fact 2.1 on a sound theoretical footing. For example, if $n = 100$ and $\theta = 800$ then, under the assumptions of Theorem 2.4, the inequality (2.2.6) holds with probability at least .99.

2.3 The LINPACK Condition Estimator

LINPACK is a collection of FORTRAN subroutines which perform many of the tasks associated with linear equations, such as matrix factorisation and solution of a linear system. The LINPACK project took place at the Argonne National Laboratory and three American universities over a period of three years beginning in 1976. During this period the LINPACK condition estimation technique was developed. Most of the LINPACK routines incorporate a condition estimator.

The code DECOMP in [11] contains a preliminary version of the LINPACK condition estimator. The theory behind the LINPACK condition estimation algorithm is described in [5] and the practical details concerning the implementation of the algorithm are to be found in [8]. Here we summarise the main features of the algorithm.

The LINPACK condition estimation algorithm consists of three steps:

(1) choose a vector $\underline{e}$ such that $||\underline{x}||$ is large relative to $||\underline{e}||$, where $A^T\underline{x} = \underline{e}$;

(2) solve $A\underline{y} = \underline{x}$;

(3) estimate $\text{cond}(A) \geq ||A|| \dfrac{||\underline{y}||}{||\underline{x}||} \leq \text{cond}(A)$.

Step (1) corresponds to the idea mentioned at the end of section 2.1. Step (2) produces a vector $\underline{y}$ satisfying $\underline{y} = (A^TA)^{-1}\underline{e}$, so $\underline{y}$ is the result of one step of inverse iteration with matrix A^TA and starting vector $\underline{e}$. The reason for this two-stage process becomes apparent when the two equations are analysed in terms of the singular value decomposition (2.2.1) of A.

Let $\underline{e} =: V\underline{\alpha}$. Then

$$\underline{x} = U^T\Sigma^{-1}\underline{\alpha} \tag{2.3.1}$$

and

$$\underline{y} = V\Sigma^{-2}\underline{\alpha}. \tag{2.3.2}$$

Step (1) requires for its success that the component of $\underline{\alpha}$ corresponding to at least one "small" right singular vector $\underline{v}_i$ be of reasonable size (cf. section 2.2). If this condition is satisfied then $\underline{x}$ will have a substantial component in at least one "small" left singular $\underline{u}_i$ and, clearly, $\underline{x}$ will be richer in the desired "small" singular vectors than is $\underline{e}$. Therefore, one would expect to find $||\underline{y}||_2/||\underline{x}||_2 \geq$

$||\underline{x}||_2/||\underline{e}||_2$. The process could be continued by repeating steps (1) and (2) in typical inverse iteration fashion. While 'convergence' would be guaranteed, the extra steps could be rather expensive and two steps are usually sufficient to provide a good order of magnitude estimate of $||A^{-1}||$.

The above analysis is geared to the ℓ_2 norm and shows that under suitable assumptions on $\underline{e}$, $||\underline{y}||_2/||\underline{x}||_2 \cong ||A^{-1}||_2$. By Lemma 1.2 and the corresponding result for vector norms it follows that $||\underline{y}||_p/||\underline{x}||_p \cong ||A^{-1}||_p$ for $p = 1,\infty$.

From (2.3.1),

$$||\underline{x}||_2^2 = \sum_{i=1}^{n} \left(\frac{\alpha_i}{\sigma_i}\right)^2 = \frac{1}{\sigma_n^2} \sum_{i=1}^{n} \left(\frac{\sigma_n}{\sigma_i}\right)^2 \alpha_i^2 \leq \frac{1}{\sigma_n^2} ||\underline{e}||_2^2.$$

Also, by (2.3.2),

$$||\underline{y}||_2^2 = \sum_{i=1}^{n} \left(\frac{\alpha_i}{\sigma_i^2}\right)^2 \geqslant \frac{\alpha_n^2}{\sigma_n^4},$$

therefore

$$\frac{||\underline{y}||_2^2}{||\underline{x}||_2^2} \geqslant \frac{\alpha_n^2}{\sigma_n^2||\underline{e}||_2^2} = \frac{(\underline{v}_n^T\underline{e})^2}{\sigma_n^2||\underline{e}||_2^2},$$

which can be written

$$\frac{||\underline{y}||_2}{||\underline{x}||_2} \geqslant \cos(\underline{v}_n,\underline{e})||A^{-1}||_2$$

where $\cos(\underline{v}_n,\underline{e}) := \dfrac{\underline{v}_n^T\underline{e}}{||\underline{e}||_2}$ is the cosine of the angle between the vectors $\underline{v}_n$ and $\underline{e}$. This result, given by Cline, Conn and Van Loan [4], reinforces our observation above that the two-stage algorithm is bound to generate large growth in stage (2) if $\underline{e}$ is not too deficient in $\underline{v}_n$.

We now describe step (1) of the algorithm. The algorithm assumes that the factorisation

$$PA = LU \tag{2.3.3}$$

is available. Thus $A^T\underline{x} = \underline{e}$ is equivalent to

$$U^T\underline{z} = \underline{e}, \tag{2.3.4}$$

$$L^TP\underline{x} = \underline{z}. \tag{2.3.5}$$

Solution of (2.3.4) can be accomplished by

$z_1 := e_1/u_{11}$

For k := 2 to n

  $p_k := u_{1k}*z_1$

For i := 2 to n-1

  $z_i := (e_i - p_i)/u_{ii}$

  For k := i + 1 to n

    $p_k := p_k + u_{ik}*z_i$

$z_n := e_n/p_n$.

The strategy devised by Cline et al. [5] is to attempt to maximise $||\underline{z}||_1$ (the LINPACK condition estimator uses the ℓ_1 norm) by choosing

e_i : = ± 1 in the following way:

e_1 : = 1.

At the i th stage, $z_1, z_2, \ldots z_{i-1}$ have been determined and e_i is to be chosen. The two possible choices of e_i lead to, in obvious notation, z_i^+ : = $(1 - p_i)/u_{ii}$, z_i^- : = $(-1 - p_i)/u_{ii}$, p_k^+ : = $p_k + u_{ik} * z_i^+$ and p_k^- : = $p_k + u_{ik} * z_i^-$, k = i+1,...n.
e_i is chosen as + 1 if

$$|1 - p_i| + \sum_{k=i+1}^{n} |p_k^+| > |-1 - p_i| + \sum_{k=i+1}^{n} |p_k^-|$$

and - 1 otherwise.

Once $\underline{z}$ is determined, (2.3.5) is solved and the algorithm continued from step (2).

The original intention was that $||\underline{x}||_1/||\underline{e}||_1$ be large, but the choice of $\underline{e}$ ostensibly makes $||\underline{z}||_1/||\underline{e}||_1$ large. Indeed $\underline{e}$ is independent of L, so there is the possibility that any growth generated in (2.3.4) will be vitiated during the solution of (2.3.5). If partial pivoting is used in the computation of (2.3.3) then $|\ell_{ij}| \leq 1$ for all i > j. This implies [17, p. 14] that $||L||_1 \leq n$ and $||L^{-1}||_1 \leq 2^{n-1}$ so

$$\frac{1}{n} ||\underline{z}||_1 \leq ||\underline{x}||_1 \leq 2^{n-1} ||\underline{z}||_1$$

and thus the use of partial pivoting in the LU factorisation of A places a reasonable restriction on the amount by which $||\underline{x}||_1$ can be smaller than $||\underline{z}||_1$.

A more serious shortcoming of the above strategy is that it is possible for most of the information about A to be contained in L rather than U, for example if U = I and L has a non-trivial lower triangular

part. In this case there is no special reason for expecting $||\underline{x}||/||\underline{e}||$ to be large when $||A^{-1}||$ is large, since L is not taken into account when choosing $\underline{e}$. We give a counter-sample to the LINPACK condition estimator, based on this observation, below.

The quantity

$$\mu_1 := \frac{||\underline{y}||_1}{||\underline{x}||_1} \leq ||A^{-1}||_1$$

is not the only estimate of $||A^{-1}||_1$ available from the algorithm. As noted by O'Leary [23] another estimate of $||A^{-1}||_1$ is available from step (1), namely

$$\nu_1 := \frac{||\underline{x}||_\infty}{||\underline{e}||_\infty} = ||\underline{x}||_\infty \leq ||A^{-1}||_1.$$

Experimental tests in [23] show that ν_1 is often a sharper estimate of $||A^{-1}||_1$ than μ_1, the quantity used inthe LINPACK estimate of $\text{cond}_1(A)$, especially for small n.

For $A \in \mathbb{R}^{n\times n}$ equality in (2.1.6) is attained for the ℓ_∞ norm when the components of $\underline{b}$ are ± 1, the signs agreeing with the corresponding signs in the row of A^{-1} with the largest row-sum. Therefore the strategy of Cline et al. [5] can be interpreted as trying to choose the vector $\underline{e}$ so that $\frac{||\underline{x}||_\infty}{||\underline{e}||_\infty} = ||A^{-T}||_\infty = ||A^{-1}||_1$. This choice of $e_i := \pm 1$ is the natural one if it is required that ν_1 be a good estimate of $||A^{-1}||_1$. We note there is no evidence in [5] that the choice $e_i := \pm 1$ was made for this reason. Clearly, $\nu_1 = ||A^{-1}||_1$ is quite likely for small n - this behaviour is observed empirically by O'Leary [23].

Another observation is that the i th component of $\underline{x}$ is

$$x_i = \sum_{j=1}^{n} b_{ij}e_j = \sum_{j=1}^{n} \pm b_{ij},$$

where $B := A^{-T}$. Hence x_i is a sum of plus or minus the elements in the i th column of A^{-1}. If $\underline{e}$ is such that ν_1 is exact then the k th column of A^{-1} has the largest ℓ_1 norm, where $|x_k| = ||\underline{x}||_\infty$. It seems reasonable to expect that, in general, $|x_k| = ||\underline{x}||_\infty$ implies that the k th column of A^{-1} has a large ℓ_1 norm. This led us to consider the possibility of adding a fourth stage to the algorithm: (4) solve $A\underline{z} = \underline{e}_k$ where $|x_k| = ||\underline{x}||_\infty$ and compute $\gamma_1 := ||\underline{z}||_1$.

We performed a small number of numerical tests using random matrices with elements uniformly distributed on [- 1,1], and $n \leq 30$. The estimate γ_1 was frequently exact and was in every case better than ν_1 and μ_1. However, γ_1 fails on the first counter-example described in the next section. Thus in view of the generally satisfactory performance of the LINPACK condition estimator we do not propose that it be modified by the inclusion of γ_1.

Cline, Conn and Van Loan [4] describe a generalisation of the LINPACK algorithm which incorporates a "look-behind" technique. Whereas the LINPACK algorithm holds each e_i fixed once it has been assigned a value, the look-behind algorithm allows for the possibility of modifying previously chosen e_is. At the i th stage the look-behind algorithm maximises a function which includes a contribution from each equation, not only equations i to n as does the LINPACK algorithm. We refer the reader to [4] for further details of the ℓ_2 and ℓ_1 look-behind algorithms.

Cost

The LINPACK condition estimator requires $O(n^2)$ flops given the factorisation (2.3.3). Thus it satisfies the requirement of being "cheap" relative to the LU factorisation. However, this comparison is only valid when n is large. For small n, the constants in the order quantities play an important role. Cline [3] states that the LINPACK condition estimator requires up to $3n^2$ multiplications and $5n^2$ additions (the precise number depends on the $e_i = \pm 1$ decisions), evaluation of $2n^2$ absolute values and possibly an additional $4n^2$ multiplications for the scaling operations (to avoid overflow in the computation of what are potentially very large numbers). The total number of operations can thus be as high as $14n^2$, which is comparable to the $2n^3/3$ additions and multiplications required for the factorisation for $n \lesssim 21$. For $n \lesssim 20$, say, the overhead involved in computing the condition estimate is non-negligible - indeed it may be cheaper to compute A^{-1} than to compute the condition estimate!

The ℓ_2 look-behind algorithm, when applied to a triangular matrix, requires $O(5n^2)$ flops. Again, this is non-negligible compared to the cost of an LU factorisation, for small n.

2.4 Reliability

The principal requirement of a condition estimator is that it should be reliable, in the sense of (2.1.3). A reliable condition estimator will always detect the presence of ill-conditioning. To date, neither the LINPACK condition estimator nor the look-behind estimator has been proved reliable (in fact the former has been proved unreliable, as indicated below). Each of these estimators produces a lower bound for the condition number which, since it is not known to be reliable,

can only signal the presence of ill-conditioning; the lower bound cannot verify well-conditioning of the matrix.

To verify well-conditioning, an upper bound for the condition number is required. Some techniques for computing a cheap upper bound for cond(A) are described in [17]. As noted in [17], a pair of bounds for the condition number, consisting of a lower bound α and an upper bound β, carries a built-in reliability test: if $\beta/\alpha \cong 1$ then each bound must be a good approximation to the condition number.

We now describe two counter-examples to the LINPACK condition estimator. The first counter-example is the matrix $A := T^T \text{ diag}(-1, -1,\ldots,-1,1)$ where T is defined in (1.3.3). These factors of A are the LU factors produced by Gaussian elimination with partial pivoting. It is straightforward to show that $\nu_1 = 1$ and $\mu_1 = 1$ whilst $||A^{-1}||_1 = 2^{n-1}$. Surprisingly, this example is given in [5], yet it is not explicitly identified as a counter-example.

The second, less trivial, counter-example is the following matrix devised by Cline [3]:

$$A := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -2k & 0 \\ 0 & 1 & k & -k \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & k \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 & -2k & 0 \\ 0 & 1 & k & -k \\ 0 & 1 & k+1 & -(k+1) \\ 0 & 0 & 0 & k \end{bmatrix}$$

$\text{Cond}_1(A) = 8k^2 + 6k + 1$ and the estimate produced by the LINPACK estimator is $5.6\ k + 5.56 + 0\ (1/k)$. As $k \to \infty$ the underestimation factor becomes arbitrarily large.

In spite of these counter-examples, the LINPACK condition estimator performs very reliably in practice. Many thousands of tests have been performed using random matrices of order up to 50, with elements coming from various distributions; the condition number estimates have rarely been more than 10 times smaller than the true condition number [3], [5], [8], [23], [28]. It seems safe to say that the LINPACK condition estimates can be used with a high degree of confidence. If reliability is the overriding concern, one could always go to the trouble of inverting A and taking the norm, or one could use a decomposition such as the QR decomposition with column pivoting from which more reliable condition estimates may be computed [17].

Chapter 3. Efficient Algorithms for Computing the Condition Number of a Tridiagonal Matrix

3.1 Operation Counts

Tridiagonal matrices

$$A := \begin{bmatrix} a_1 & c_1 & & & 0 \\ b_2 & a_2 & c_2 & & \\ & b_3 & a_3 & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \\ & & & & c_{n-1} \\ 0 & & & b_n & a_n \end{bmatrix} \quad \varepsilon \, \mathbb{R}^{nxn} \qquad (3.1.1)$$

arise in many areas of numerical analysis such as spline analysis [6, p. 133], difference methods for boundary value problems [10] and the numerical solution of linear second order recurrence relations [2, p. 14 ff.]. Since the nonzero elements of A occur only within a band of width three, efficient methods are available for the solution of the system $A\underline{x} = \underline{b}$.

For a tridiagonal matrix, the standard linear algebra processes and their operation counts are as follows, where M, A and D denote one floating-point multiplication, addition or subtraction and division operation respectively:

(i) LU factorisation by Gaussian elimination without pivoting,

$$A = LU. \qquad (3.1.2)$$

Cost: $(n - 1)(M + A + D)$.

(ii) Solution of $A\underline{x} = \underline{b}$ using the factorisation computed in (i).

Cost: $2(n - 1)(M + A) + nD$.

(iii) Inversion of A using (3.1.2) to compute the columns of the inverse.

Cost: $\frac{3}{2}n^2 (M + A) + O(n)(M + A + D)$.

(iv) LU factorisation by Gaussian elimination with partial pivoting,

$$PA = LU. \tag{3.1.3}$$

Cost: $(2n - 3)M + (n - 1)A + (n - 1)D$.

(v) Solution of $A\underline{x} = \underline{b}$ using the factorisation computed in (iv).

Cost: $(3n - 4)(M + A) + nD$.

(vi) Inversion of A using (3.1.3) to compute the columns of the inverse.

Cost: $\frac{5}{2}n^2 (M + A) + O(n)(M + A + D)$.

Since A is tridiagonal, the factors L and U in (3.1.2) are unit lower bidiagonal and upper bidiagonal respectively.

The partial pivoting strategy in (3.1.3) has the effect of changing the form of the triangular factors: U is upper triangular with $u_{ij} = 0$ for $j > i + 2$ and L is lower triangular with at most two nonzero elements in each column. For $n = 5$, if the natural pivot is chosen at the first two stages and the partial pivot at each subsequent stage, L and U have the forms

$$L = \begin{bmatrix} 1 & & & \\ X & 1 & & \\ & & 1 & \\ & & & 1 \\ & X & X & X & 1 \end{bmatrix}, \quad U = \begin{bmatrix} X & X & & & \\ & X & X & & \\ & & X & X & X \\ & & & X & X \\ & & & & X \end{bmatrix},$$

where X denotes a potentially nonzero element and unmarked elements are zero. We note that the operation counts for (iv), (v) and (vi) above are dependent on the number of row interchanges performed during the elimination. The numbers quoted correspond to the worst case of an interchange at each stage.

The main feature of the operation counts for a tridiagonal matrix is that, in contrast to the situation for a full matrix, the processes of LU factorisation and solution of $A\underline{x} = \underline{b}$ using this factorisation are essentially equal in cost, requiring O(n) flops, whereas computation of the inverse is an order of magnitude more expensive. The inverse of a tridiagonal matrix is a full matrix in general and is rarely required; thus, for economy of both storage and floating-point operations, it is imperative that a condition estimation or computation algorithm for tridiagonal matrices does not require explicit computation of the inverse. In view of the operation counts an obvious requirement is that such an algorithm should use only O(n) flops.

To conclude this section we derive the recurrence relations which determine the factors L and U in (3.1.2). These are used in section 3.3. It is easily seen that the superdiagonal of A passes unchanged into U. Writing

$$L := \begin{bmatrix} 1 & & & 0 \\ \ell_2 & 1 & & \\ & \ell_3 & 1 & \\ & & \ddots & \ddots \\ 0 & & \ell_n & 1 \end{bmatrix}, \quad U := \begin{bmatrix} u_1 & c_1 & & 0 \\ & u_2 & c_2 & \\ & & \ddots & \ddots \\ & & & c_{n-1} \\ 0 & & & u_n \end{bmatrix} \tag{3.1.4}$$

and equating the $(i, i-1)$ and (i,i) elements in (3.1.2) gives

$$b_i = \ell_i u_{i-1} \quad , \quad a_i = \ell_i c_{i-1} + u_i .$$

Thus the unknown elements of L and U are given by

$$u_1 := a_1; \quad \ell_i := \frac{b_i}{u_{i-1}}, \; u_i := a_i - \ell_i c_{i-1}, \; i = 2,3,\ldots,n. \tag{3.1.5}$$

3.2 The Inverse of a Bidiagonal Matrix

In this section we find in explicit form the inverse of a general bidiagonal matrix. By utilising the form of the elements of the inverse we derive an efficient algorithm for computing the condition number of a bidiagonal matrix.

The i th column, $\underline{x}$, of the inverse of the unit lower bidiagonal matrix L in (3.1.4) is the solution of $L\underline{x} = \underline{e}_i$, where $\underline{e}_i$ is the i th unit vector. Hence the elements $x_1, x_2, \ldots, x_n$ of $\underline{x}$ are given by

$$\begin{aligned} x_r &= 0, \qquad r < i, \\ x_i &= 1, \\ x_r &= -\ell_r x_{r-1}, \quad r > i. \end{aligned}$$

Clearly,

$$x_r = \prod_{s=i+1}^{r} (-\ell_s) \quad , \quad r>i.$$

Writing

$$m_i := -\ell_i \tag{3.2.1}$$

it follows that

$$L^{-1} = \begin{bmatrix} 1 & & & & & 0 \\ m_2 & 1 & & & & \\ m_2m_3 & m_3 & \ddots & & & \\ m_2m_3m_4 & m_3m_4 & \ddots & \ddots & & \\ \vdots & \vdots & & & \ddots & \\ m_2m_3\ldots m_n & \cdots & \cdots & m_{n-1} & m_n & 1 \end{bmatrix} . \tag{3.2.2}$$

The nonsingular upper bidiagonal matrix

$$U := \begin{bmatrix} u_1 & c_1 & & & 0 \\ & u_2 & c_2 & & \\ & & u_3 & \ddots & \\ & & & \ddots & c_{n-1} \\ 0 & & & & u_n \end{bmatrix} \quad \varepsilon \ \mathbb{R}^{nxn} \tag{3.2.3}$$

can be written U = DT where D := diag (u_i) and T is unit upper bidiagonal. By comparison with (3.2.2),

$$T^{-1} = \begin{bmatrix} 1 & v_2 & v_3v_2 & \cdots & v_nv_{n-1}\cdots v_2 \\ & 1 & v_3 & & \cdot \\ & & 1 & & \cdot \\ & & \cdot & & \cdot \\ & & & \cdot & v_nv_{n-1} \\ & & & \cdot & v_n \\ 0 & & & & 1 \end{bmatrix},$$

where the $\{v_i\}$ are given by

$$v_i = - c_{i-1}/u_{i-1}. \tag{3.2.4}$$

Since $U^{-1} = T^{-1} D^{-1}$,

$$U^{-1} = \begin{bmatrix} w_1 & w_2v_2 & w_3v_3v_2 & \cdots & w_nv_nv_{n-1}\cdots v_2 \\ & w_2 & w_3v_3 & & \cdot \\ & & w_3 & \cdot & \cdot \\ & & & \cdot & w_nv_nv_{n-1} \\ & & & \cdot & w_nv_n \\ 0 & & & & w_n \end{bmatrix}, \tag{3.2.5}$$

where

$$w_i = \frac{1}{u_i}. \tag{3.2.6}$$

It is convenient at this point to introduce some notation. If $A \in \mathbb{C}^{n\times n}$ we define A's <u>comparison matrix</u>, M(A), by

$$M(A) := (m_{ij}), \text{ where } \quad m_{ij} = \begin{cases} |a_{ii}|, & i = j, \\ -|a_{ij}|, & i \neq j. \end{cases} \tag{3.2.7}$$

Also, if

$$A := (a_{ij}),\ B := (b_{ij}) \in \mathbb{C}^{n\times n} \text{ then}$$

$$|A| = |B| \quad \text{iff} \quad |a_{ij}| = |b_{ij}| \quad \text{for } 1 \leq i,j \leq n.$$

We observe that each element of U^{-1} in (3.2.5) is of the form $z = a/b$, where a and b are products of elements of U. It follows that $|z|$, and hence also $||U^{-1}||_\infty$, is independent of the signs of the elements of U. In other words,

$$|U| = |T| \text{ implies } ||U^{-1}||_\infty = ||T^{-1}||_\infty. \tag{3.2.8}$$

There is one particular distribution of the signs of the elements which yields a matrix for which the ℓ_∞ norm of the inverse is easily calculated. From (3.2.3) - (3.2.6) it is clear that $M(U)^{-1}$ has nonnegative elements, which we write as $M(U)^{-1} \geq 0$. Also, $|M(U)| = |U|$ so by (3.2.8), $||M(U)^{-1}||_\infty = ||U^{-1}||_\infty$. An observation which has appeared many times in the literature is that if $A^{-1} \geq 0$ then

$$||A^{-1}||_\infty = ||A^{-1}\underline{e}||_\infty, \tag{3.2.9}$$

where $\underline{e} := (1,1,\ldots,1)^T$. Thus for the matrix U in (3.2.3),

$$||U^{-1}||_\infty = ||M(U)^{-1}||_\infty = ||M(U)^{-1}\underline{e}||_\infty. \tag{3.2.10}$$

Hence, in order to calculate $||U^{-1}||_\infty$ for an upper bidiagonal matrix U it is only necessary to solve one bidiagonal linear system and then evaluate the ℓ_∞ norm of the solution vector. This process costs O(n) flops - a significant reduction on the $O(n^2)$ flops required to compute the inverse and then take the norm.

Since the above applies equally well to a lower bidiagonal matrix we have the following algorithm:

Algorithm B

Given a nonsingular nxn bidiagonal matrix B this algorithm computes $\gamma = ||B^{-1}||_\infty$.

(1) Solve $M(B)\underline{z} = \underline{e}$.

(2) Compute $\gamma := ||\underline{z}||_\infty$.

Cost: $(n - 1)(M + A) + nD + (3n - 1)$ ABS,
where ABS denotes an absolute value computation.

It is possible to compute a reliable estimate of the smallest singular value of B, σ_n, in O(4n) flops. For by applying Algorithm B to the matrices B and B^T, the quantity $\hat{\sigma} := (||B^{-1}||_1 ||B^{-1}||_\infty)^{-\frac{1}{2}}$ may be computed, and by (2.1.4)

$$\frac{1}{\sqrt{n}}\sigma_n \leq \hat{\sigma} \leq \sigma_n. \qquad (3.2.11)$$

The Golub-Reinsch SVD algorithm [13] reduces $A \in \mathbb{R}^{mxn}$ to upper bidiagonal form B before applying an iterative process to compute the singular values. The estimate $\hat{\sigma} \simeq \sigma_n(B) = \sigma_n(A)$ can be computed cheaply at the end of the reduction to bidiagonal form and may obviate the need for the second stage if only $\sigma_n(A)$ is of interest and $\sqrt{n}$ is an acceptable uncertainty factor.

3.3 An Algorithm Based on the LU Factorisation

Assume that for the nonsingular tridiagonal matrix A in (3.1.1) the LU factorisation (3.1.2) exists, that is to say each principal minor of A is nonzero. Then $A^{-1} = U^{-1}L^{-1}$ is the product of two matrices of the special forms (3.2.5) and (3.2.2). As the next theorem shows, A^{-1} itself has a special form. The following definition is required.

Definition 3.1

The tridiagonal matrix A in (3.1.1) is irreducible if $b_2, b_3, \ldots, b_n$ and $c_1, c_2, \ldots c_{n-1}$ are all nonzero; otherwise A is reducible. □

We remark that this definition is consistent with the more usual definition of irreducibility which states that a general square matrix A is irreducible if there does not exist a permutation matrix P such that

$$P^T A P = \begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix}$$

where A_1 and A_3 are square matrices. For a proof see [24, p. 104].

Theorem 3.2

Let the tridiagonal matrix A in (3.1.1) be nonsingular and irreducible and suppose that A has an LU factorisation. Then there exist nonzero vectors $\underline{x}$, $\underline{y}$, $\underline{p}$ and $\underline{q}$ such that $A^{-1} := (\alpha_{ij})$ is given by

$$\alpha_{ij} = \begin{cases} x_i\, y_j & , \quad j \geq i, \\ p_i\, q_j & , \quad i \geq j. \end{cases} \tag{3.3.1}$$

Proof

L and U are given by (3.1.4) and (3.1.5). The fact that none of the $\{b_i\}$ or $\{c_i\}$ vanishes implies that for all i, $\ell_i \neq 0$, $m_i \neq 0$, $v_i \neq 0$ and $w_i \neq 0$, by (3.1.5), (3.2.1), (3.2.4) and (3.2.6).

Consider the identity $A^{-1} = U^{-1}L^{-1}$, with $U^{-1} := (\beta_{ij})$ given by (3.2.5) and $L^{-1} := (\gamma_{ij})$ given by (3.2.2). Choosing $x_1 := 1$ and $y_j := \sum_{k=j}^{n} \beta_{1k}\,\gamma_{kj}$, $j = 1,2,\ldots,n$, then $\alpha_{1j} = x_1 y_j$, $j = 1,2,\ldots,n$. We claim that the $\{x_i\}$ defined by

$$x_i := x_{i-1}/v_i, \quad i = 2,\ldots,n \tag{3.3.2}$$

satisfy the remaining conditions in (3.3.1) for $j \geq i$. We prove this by induction.

Clearly, $x_i \neq 0$ for all i. Suppose

$$\alpha_{ij} = x_i y_j \quad \text{for } i \leq r \text{ and } j \geq i.$$

It suffices to show that

$$\alpha_{rj} = \frac{x_r}{x_{r+1}}\ \alpha_{r+1,j}$$

$$= v_{r+1}\ \alpha_{r+1,j}, \text{ for } j > r.$$

For $k > r$ it is clear from (3.2.5) that $\beta_{rk} = v_{r+1}\,\beta_{r+1,k}$, so for $j > r$

$$\alpha_{rj} = \sum_{k=j}^{n} \beta_{rk}\,\gamma_{kj} = v_{r+1} \sum_{k=j}^{n} \beta_{r+1,k}\,\gamma_{kj} = v_{r+1}\,\alpha_{r+1,j}$$

as required.

This establishes the first part of (3.3.1). The second part ($i \geq j$) follows immediately from the first part applied to A^T, which is again tridiagonal and satisfies all the required conditions. □

Theorem 3.2 shows that the upper half of A^{-1} is identical to the upper half of one rank-one matrix, $\underline{x}\,\underline{y}^T$, and the lower half of A^{-1} is identical to the lower half of a second rank-one matrix, $\underline{p}\,\underline{q}^T$. Suppose $\underline{x}$, $\underline{y}$, $\underline{p}$ and $\underline{q}$ have been computed. Formation of each element α_{ij} of A^{-1} explicitly, using (3.3.1), requires $O(n^2)$ multiplications. However, $||A^{-1}||_1$ or $||A^{-1}||_\infty$ can be evaluated without forming all the products in (3.3.1) at a cost of only $O(n)$ multiplications:

Algorithm Norm

Given $\underline{x}$, $\underline{y}$, $\underline{p}$ and $\underline{q}$ in (3.3.1) this algorithm computes $\gamma = ||A^{-1}||_\infty$.

$s_n := |y_n|$

For i := n - 1 to 1 step -1
  $s_i := s_{i+1} + |y_i|$

$t_1 := |q_1|$

For i := 2 to n
  $t_i := t_{i-1} + |q_i|$

$\gamma := \max\,(|x_1|*s_1,\ |p_n|*t_n)$

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

Cost: (2n - 2)M + (3n - 4)A + (4n - 2)ABS.

We now show that the vectors $\underline{x}$, $\underline{y}$, $\underline{p}$ and $\underline{q}$ can be computed in O(n) operations given the LU factorisation of A. Taking $x_1 := 1$, (3.3.2) shows that the remaining $\{x_i\}$ can be computed at a cost of n - 1 divisions. Because $x_1 = 1$, the first row of A^{-1} is $\underline{y}^T$, by (3.3.1). Hence $\underline{e}_1^T A^{-1} = \underline{y}^T$, that is

$$A^T \underline{y} = \underline{e}_1.$$

By solving this linear system, $\underline{y}$ is obtained in O(n) operations. The vectors $\underline{p}$ and $\underline{q}$ satisfy relations similar to those satisfied by $\underline{x}$ and $\underline{y}$. It is easily verified that if $q_1 := 1$,

$$q_i = - q_{i-1}/\ell_i \quad , \quad i = 2,\ldots,n.$$

Furthermore, $\underline{p}$ is the first column of A^{-1}, so

$$A\underline{p} = \underline{e}_1.$$

These relations yield the following algorithm:

Algorithm Vector

Given the LU factors of A in (3.1.4) this algorithm computes the vectors $\underline{x}$, $\underline{y}$, $\underline{p}$ and $\underline{q}$ in (3.3.1).

(1) $x_1 := 1$

For i := 2 to n
  $x_i := - x_{i-1} * u_{i-1}/c_{i-1}$.

(2) Solve $A^T\underline{y} = \underline{e}_1$.

(3) $q_1 := 1$

For i := 2 to n

  $q_i := -q_{i-1}/\ell_i$.

(4) Solve $A\underline{p} = \underline{e}_1$.

Cost: $(5n - 5)M + (2n - 2)A + (4n - 2)D$.

Combining Algorithms Norm and Vector we obtain our first algorithm for computing $||A^{-1}||_\infty$.

Algorithm 1

Given a nonsingular nxn irreducible tridiagonal matrix A of the form (3.1.1) and possessing an LU factorisation, this algorithm computes $||A^{-1}||_\infty$.

(1) Compute the LU factors according to (3.1.5).

(2) Apply Algorithm Vector.

(3) Apply Algorithm Norm.

Cost: $(8n - 8)M + (6n - 7)A + (5n - 3)D + (4n - 2)ABS$.

Algorithm 1 imposes two serious restrictions on the matrix A, that it be irreducible and have an LU factorisation. The latter condition is vital to the proof of Theorem 3.2 and hence to Algorithm 1. Nevertheless, as shown in the next section, Theorem 3.2 remains true if the LU assumption is removed. Algorithms 2 and 3 below do not require the existence of the LU factorisation. What to do if A is reducible is described in section 3.7.

3.4 The Inverse of a Tridiagonal Matrix

The next theorem generalises Theorem 3.2 and describes the form of the inverse of a general tridiagonal matrix. The first part of the theorem is the same as Theorem 2 of Ikebe [19] and, for the case A symmetric, is proved by Bukhberger and Emel'yanenko [1] and stated without proof by Graybill [14, p. 179].

Theorem 3.3

Let the tridiagonal matrix A in (3.1.1) be nonsingular.

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

$$\alpha_{ij} = \begin{cases} x_i y_j , & j \geq i, \\ p_i q_j , & i \geq j. \end{cases} \tag{3.4.1}$$

(2) If A is reducible then

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

$$A = \begin{bmatrix} A_1 & 0 \\ B_1 & A_2 \end{bmatrix}, \tag{3.4.2}$$

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

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

where $X \in \mathbb{R}^{(n-i)xi}$ is a rank-one matrix, and the theorem applies recursively to A_1 and A_2;

(b) if $b_i = 0$, part (a) applies to A^T.

Proof

(1): Let $A^{-1} := (\alpha_{ij})$,

$$P_j := \begin{bmatrix} c_1 & & & 0 \\ a_2 & c_2 & & \\ b_3 & a_3 & c_3 & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots & \ddots \\ 0 & & b_j & a_j & c_j \end{bmatrix} \in \mathbb{R}^{jxj},$$

and let $\underline{a}^{(j)}$ be the vector composed of the first j elements in the first column of A. Equating the j th columns in the identity $AA^{-1} = I$ gives

$$A \begin{bmatrix} \alpha_{1j} \\ \vdots \\ \alpha_{nj} \end{bmatrix} = \underline{e}_j,$$

the first j-1 equations of which can be written

$$\alpha_{1j}\, \underline{a}^{(j-1)} + P_{j-1} \begin{bmatrix} \alpha_{2j} \\ \vdots \\ \alpha_{jj} \end{bmatrix} = \underline{0}.$$

The irreducibility of A implies that the lower triangular matrix P_{j-1} is nonsingular, so

$$\begin{bmatrix} \alpha_{2j} \\ \vdots \\ \alpha_{jj} \end{bmatrix} = - \alpha_{ij} P_{j-1}^{-1} \underline{a}^{(j-1)}. \qquad (3.4.3)$$

Since P_{j-1} is lower triangular, P_{j-1}^{-1} is the leading principal submatrix of order j-1 of P_{n-1}^{-1}. Hence $P_{j-1}^{-1} \underline{a}^{(j-1)}$ is the same vector as that formed by the first j-1 elements of $P_{n-1}^{-1} \underline{a}^{(n-1)}$. It follows from (3.4.3) that we can take $y_j := \alpha_{1j}$ for all j and $x_1 := 1$, $(x_2,\ldots,x_n)^T := - P_{n-1}^{-1} \underline{a}^{(n-1)}$.

The vectors $\underline{p}$ and $\underline{q}$ are obtained by applying the above to A^T.

(2): If $c_i = 0$ then A is clearly of the form (3.4.2) and $0 \neq \det(A) = \det(A_1) \det(A_2)$, so A_1 and A_2 are nonsingular. It is easily verified that $X = - A_2^{-1} B_1 A_1^{-1}$. B_1 has only one nonzero element, b_{i+1} in its (1,i) position, so B_1 is of rank one. Hence X is of rank one. □

Remarks

(1) Our proof of the first part of Theorem 3.3 follows that of Ikebe's Theorem 1. The latter theorem applies to Hessenberg matrices. Note that we only need A to be lower Hessenberg with nonzero superdiagonal elements in order to produce $\underline{x}$ and $\underline{y}$ in (3.4.1).

(2) The presence of zeros on the subdiagonal or superdiagonal does not greatly complicate the form of the inverse. The inverse simply consists of diagonal blocks possessing the form (3.4.1) together with off-diagonal blocks which have either rank one or rank zero. The following example should make this clear.

$$A := \left[\begin{array}{cccccccc} 2 & -1 & & & & & & 0 \\ 0 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & -1 & 2 & -1 & & & \\ & & & -1 & 2 & -1 & & \\ & & & & -1 & 2 & 0 & \\ & & & & & -1 & 2 & -1 \\ 0 & & & & & & -1 & 2 \end{array}\right] \in \mathbb{R}^{8\times 8}.$$

$$A^{-1} = \left[\begin{array}{cccccccc} \frac{1}{2} & \frac{5}{12} & \frac{1}{3} & \frac{1}{4} & \frac{1}{6} & \frac{1}{12} & 0 & 0 \\ 0 & \frac{5}{6} & \frac{2}{3} & \frac{1}{2} & \frac{1}{3} & \frac{1}{6} & 0 & 0 \\ 0 & \frac{2}{3} & \frac{4}{3} & 1 & \frac{2}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{2} & 1 & \frac{3}{2} & 1 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} & 1 & \frac{4}{3} & \frac{2}{3} & 0 & 0 \\ 0 & \frac{1}{6} & \frac{1}{3} & \frac{1}{2} & \frac{2}{3} & \frac{5}{6} & 0 & 0 \\ 0 & \frac{1}{9} & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \frac{5}{9} & \frac{2}{3} & \frac{1}{3} \\ 0 & \frac{1}{18} & \frac{1}{9} & \frac{1}{6} & \frac{2}{9} & \frac{5}{18} & \frac{1}{3} & \frac{2}{3} \end{array}\right].$$

Ikebe [19] gives an algorithm, which we now describe, for computing the vectors $\underline{x}$ and $\underline{y}$ in (3.4.1). The algorithm requires that A be irreducible and is based on the observations that $A\, y_n\, \underline{x} = \underline{e}_n$ (from the last column in $AA^{-1} = I$) and $x_1\, \underline{y}^T A = \underline{e}_1^T$ (from the first row of $A^{-1}A = I$).

These relations are the same as

$$A\underline{x} = y_n^{-1}\,\underline{e}_n \tag{3.4.4}$$

and

$$A^T\underline{y} = x_1^{-1}\,\underline{e}_1. \tag{3.4.5}$$

$\underline{x}$ and $\underline{y}$ in (3.4.1) are unique up to a nonzero scale factor so any nonzero element of either vector can be assigned an arbitrary nonzero value. As in section 3.3 we choose $x_1 := 1$ (clearly $x_1 \neq 0$). Then the first n-1 equations in (3.4.4) become a nonsingular lower triangular system of equations for the unknowns $x_2,\ldots,x_n$. The n th equation in (3.4.4) determines y_n and the last n-1 equations in (3.4.5) form an upper triangular system of equations for the unknowns $y_{n-1},\ldots,y_1$. The first equation in (3.4.5) is redundant and could be used as a check. The vectors $\underline{p}$ and $\underline{q}$ in (3.4.1) are obtained in a similar way, using A^T in place of A.

We thus have a second algorithm for computing $||A^{-1}||_\infty$.

Algorithm 2

Given the nonsingular nxn irreducible tridiagonal matrix A in (3.1.1) this algorithm computes $||A^{-1}||_\infty$.

(1) $x_1 := 1$; $x_2 := -a_1/c_1$

For i := 3 to n
  $x_i := -(a_{i-1}*x_{i-1} + b_{i-1}*x_{i-2})/c_{i-1}$

$y_n := 1/(b_n*x_{n-1} + a_n*x_n)$

$y_{n-1} := -a_n*y_n/c_{n-1}$

```
For i := n-2 to 1 step -1
    y_i := - (a_{i+1}*y_{i+1} + b_{i+2}*y_{i+2})/c_i
```

$(a_1*y_1 + b_2*y_2 = 1$ - check).

(2) Repeat (1) with x_i, y_i, b_i and c_i replaced by q_i, p_i, c_{i-1} and b_{i+1} respectively.

(3) Apply Algorithm Norm.

Cost: $(10n - 12)M + (7n - 10)A + (4n - 2)D + (4n - 2)ABS$.

We note that equations (3.4.4) and (3.4.5) provide another derivation of Algorithm Vector. For if A = LU, (3.4.4) implies

$$U\underline{x} = y_n^{-1} L^{-1} \underline{e}_n = y_n^{-1} \underline{e}_n. \qquad (3.4.6)$$

On setting $x_1 := 1$, the first n-1 equations in (3.4.6) may be solved for $x_2,\ldots,x_n$, the required operations being precisely those of the first loop in Algorithm Vector.

Algorithms 1 and 2 are not equivalent however, for, as confirmed by the operation counts, they do not perform the same arithmetical operations.

3.5 Utilising Symmetry

Any irreducible tridiagonal matrix is a row scaling of a symmetric† tridiagonal matrix. Specifically, if A in (3.1.1) is irreducible and

$$D := \mathrm{diag}(d_i) = \mathrm{diag}\left(1, \frac{c_1}{b_2}, \frac{c_1}{b_2}\frac{c_2}{b_3}, \ldots, \frac{c_1}{b_2}\frac{c_2}{b_3}\cdots\frac{c_{n-1}}{b_n}\right)$$

then T := DA is symmetric† and tridiagonal. Assuming that A is nonsingular,

---

†irreducible

we have $A^{-1} = T^{-1}D$. The vectors $\underline{x}$ and $\underline{y}$ such that $T^{-1} := (\beta_{ij})$ with

$$\beta_{ij} = \begin{cases} x_i\, y_j & , \quad j \geq i, \\ y_i\, x_j & , \quad i \geq j, \end{cases} \tag{3.5.1}$$

can be computed by forming $T = DA$, at a cost of $O(3n)M + O(n)D$, and applying to the matrix T a version of Algorithm 2 adapted to take advantage of symmetry (see below). However, it is possible to calculate $\underline{x}$ and $\underline{y}$ in (3.5.1) without ever forming T explicitly.

Consider Algorithm 2 applied to $T = DA$. Equations (3.4.4) and (3.4.5) become

$$DA\underline{x} = y_n^{-1}\, \underline{e}_n \tag{3.5.2}$$

and

$$(DA)^T \underline{y} = x_1^{-1}\, \underline{e}_1. \tag{3.5.3}$$

We rewrite (3.5.2) as

$$A\underline{x} = (d_n y_n)^{-1}\, \underline{e}_n. \tag{3.5.4}$$

This rearrangement restricts D to an appearance in just one of the n equations. Choosing $x_1 := 1$ as usual, $x_2,\ldots,x_n$ are determined from the first n-1 equations in (3.5.4), as in Algorithm 2. The last equation in (3.5.4) gives

$$y_n := (b_n x_{n-1} + a_n x_n)^{-1}\, d_n^{-1}. \tag{3.5.5}$$

D is removed completely from (3.5.3) by utilising the symmetry of T to replace $(DA)^T$ by DA, yielding

$$A\underline{y} = x_1^{-1} d_1^{-1} \underline{e}_1 = \underline{e}_1. \tag{3.5.6}$$

There is a simple, direct way of solving (3.5.6) which takes advantage of the special nature of the right-hand side and the knowledge that $y_n \neq 0$ (from (3.5.1). A scalar multiple of the true solution vector is computed by setting $z_n := 1$ and solving the last n-1 equations in (3.5.6) for $z_{n-1},\ldots,z_1$. By construction $A\underline{z} = \theta\ \underline{e}_1$, where $\theta := a_1z_1 + c_1z_2$, so $\underline{y} := \frac{1}{\theta}\ \underline{z}$ is the solution of (3.5.6) ($\theta \neq 0$ since A is nonsingular). Note that we have not used (3.5.5) - this could be used as a check.

The advantage of the algorithm described here for the computation of $\underline{x}$ and $\underline{y}$ is that it has no explicit dependence on the matrix D.

Since $A^{-1} = T^{-1}D$ it is cheaper to evaluate $||A^{-1}||_1$ than to evaluate $||A^{-1}||_\infty$, given $\underline{x}$ and $\underline{y}$ in (3.5.1). The next algorithm computes $||A^{-1}||_1$ and incorporates a modified version of Algorithm Norm.

Algorithm 3

Given the nonsingular nxn irreducible tridiagonal matrix A in (3.1.1) this algorithm computes $\gamma = ||A^{-1}||_1$.

(1) $x_1 := 1$; $x_2 := -a_1/c_1$

For i := 3 to n
  $x_i := -(a_{i-1}*x_{i-1} + b_{i-1}*x_{i-2})/c_{i-1}$.

(2) $z_n := 1$; $z_{n-1} := -a_n/b_n$

For i := n-2 to 1 step -1
  $z_i := -(a_{i+1}*z_{i+1} + c_{i+1}*z_{i+2})/b_{i+1}$

$\theta := a_1*z_1 + c_1*z_2$.

(3) For i := 1 to n

$x_i := |x_i|$

$z_i := |z_i|$

$s_n := z_n$

For i := n-1 to 1 step -1

$s_i := s_{i+1} + z_i$

$t_1 := 1$

For i := 2 to n

$t_i := t_{i-1} + x_i$

$d_1 := 1$; $\gamma := s_1$

For j := 2 to n

$d_j := d_{j-1} * c_{j-1} / b_j$

$\gamma := \max(\gamma, (z_j * t_{j-1} + x_j * s_j) * |d_j|)$

$\gamma := \gamma / |\theta|$.

Cost: $(8n - 10)M + (5n - 6)A + (3n - 2)D + 3n$ ABS.

Algorithm 3 is based on a representation of A^{-1} which is the same as that specified by Theorem 2 of Yamamoto and Ikebe [35]. Indeed the recurrence relations in steps (1) and (2) of Algorithm 3 are identical to equations (5) and (6) in [35] provided that the arbitrary parameters

h_1, a_1 and h_2 therein are each chosen to be 1. (Note that our notation (3.1.1) differs from that in [35].)

If A is symmetric then $D = \text{diag}\left(\prod_{j=2}^{i} \frac{c_{j-1}}{b_j} \right) = I$. If this is accounted for in Algorithm 3, the operation count is reduced. Computational savings can also be made in Algorithms 1 and 2 if A is symmetric since $\underline{x} = \underline{q}$ and $\underline{p} = \underline{y}$ in (3.3.1) and (3.4.1).

In the following table we summarise the operation counts for Algorithms 1, 2 and 3. The O(1) terms have been omitted.

	Nonsymmetric A				Symmetric A			
Operations x n	M	A	D	ABS	M	A	D	ABS
Algorithm 1	8	6	5	4	5	5	3	2
Algorithm 2	10	7	4	4	6	5	2	2
Algorithm 3	8	5	3	3	6	5	2	2

(3.5.7)

For nonsymmetric tridiagonal matrices A Algorithm 3 is the most efficient and it is the easiest of the three algorithms to "code up". If $||A^{-1}||_\infty$ is required, Algorithm 3 should be applied to A^T.

There is a potential saving with Algorithm 1; cond(A) is most likely to be required in connection with solving a linear system and in this case the LU factorisation of A (without pivoting) may already be available. If this is so, the operation count for Algorithm 1 is reduced by O(n)(M + A + D). Note that it is the factorisation (3.1.2) which is required by Algorithm 1. An algorithm which uses the partial

pivoting factorisation (3.1.3) is described in section 3.8. The tridiagonal matrices which arise in spline analysis and difference methods for boundary value problems are frequently diagonally dominant [29, pp. 99-101], [24, p. 96 ff.]; for these matrices it is not necessary to use pivoting in Gaussian elimination [33, p 288].

When the matrix A is symmetric there is little to choose between Algorithms 1, 2 and 3. If A is positive definite, the method of the next section is to be preferred.

3.6 Positive Definiteness

Let

$$A := \begin{bmatrix} a_1 & b_2 & & & 0 \\ b_2 & a_2 & b_3 & & \\ & b_3 & a_3 & \ddots & \\ & & \ddots & \ddots & b_n \\ 0 & & & b_n & a_n \end{bmatrix} \in \mathbb{R}^{n \times n} \qquad (3.6.1)$$

be a positive definite tridiagonal matrix with, necessarily,

$$a_i > 0 \text{ for all } i. \qquad (3.6.2)$$

Such matrices occur quite frequently in, for example, spline analysis [6, p. 134] and difference methods for boundary value problems [29, p. 504]. In this section we derive a method for computing $||A^{-1}||_\infty$, for A given by (3.6.1), which is more efficient than the symmetric versions of Algorithms 1, 2 and 3.

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

$$M(A) = W := DAD, \tag{3.6.3}$$

where $M(A)$ is the comparison matrix defined in (3.2.7). From (3.6.3),

$$W_{ii} = a_i \quad , \quad i = 1,2,\ldots,n$$

and

$$W_{i,i+1} = d_i \, b_{i+1} \, d_{i+1} \quad , \quad i = 1,2,\ldots,n-1.$$

Let $d_1 := 1$ and $d_{i+1} := -\operatorname{sgn}(d_i b_{i+1})$, $i = 1,2,\ldots,n-1$, where

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

W is evidently tridiagonal and symmetric, $w_{ii} = a_i = |a_i|$ (using (3.6.2)), $|w_{i,i+1}| = |b_{i+1}|$ and $w_{i,i+1} \leqslant 0$. Hence $W = M(A)$.

A is positive definite and hence has a Choleski factorisation

$$A = LL^T, \tag{3.6.4}$$

where

$$L := \begin{bmatrix} \ell_1 & & & 0 \\ m_2 & \ell_2 & & \\ & m_3 & \ell_3 & \\ & & \ddots & \ddots \\ 0 & & & m_n \quad \ell_n \end{bmatrix}$$

with $\ell_i > 0$ for all i. We claim that

$$M(A) = M(L)M(L)^T. \tag{3.6.5}$$

This is proved by noting, first, that (3.6.4) implies $a_i = m_i^2 + \ell_i^2$ and $b_{i+1} = \ell_i\, m_{i+1}$. Then, writing $H := M(L)M(L)^T$,

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

and

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

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

Note that $M(A)$ is positive definite, by (3.6.5). In fact, the positive definiteness of A is independent of the signs of the off-diagonal elements $\{b_i\}$.

From section 3.2, $M(L)^{-1} \geq 0$, therefore

$$M(A)^{-1} = M(L)^{-T}\, M(L)^{-1} \geq 0. \tag{3.6.6}$$

The final result we require is

$$M(A)^{-1} = |A^{-1}|, \tag{3.6.7}$$

which is a consequence of (3.6.3).

It follows from (3.6.6) and (3.6.7) that if A is a positive definite tridiagonal matrix,

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

As in the bidiagonal matrix case, computation of the ℓ_∞ norm of the

inverse reduces to solution of a linear system involving the comparison matrix. Thus we have

Algorithm 4

Given the nonsingular nxn positive definite tridiagonal matrix A in (3.6.1) this algorithm computes $\gamma = ||A^{-1}||_\infty$.

(1) Solve $M(A)\underline{z} = \underline{e}$.

(2) Compute $\gamma := ||\underline{z}||_\infty$.

Cost: $(3n - 3)(M + A) + (2n - 1)D + (2n - 1)ABS$.

Step (1) of Algorithm 4 requires solution of a linear system whose coefficient matrix is tridiagonal and positive definite. This can be done by computing some factorisation of the coefficient matrix (since this is positive definite, pivoting is not necessary) and forward and back substituting. There is a choice of factorisations. Compared to the standard LU factorisation, the Choleski factorisation (3.6.4) offers no advantages in this context for symmetry confers no computational savings, the factorisation requires n square roots and the ensuing substitution phase requires an extra n divisions. Therefore it is preferable to use the LU factorisation, or the LDL^T factorisation, where

$$L := \begin{bmatrix} 1 & & & 0 \\ \ell_2 & 1 & & \\ & \ell_3 & 1 & \\ & & \ddots & \ddots \\ 0 & & \ell_n & 1 \end{bmatrix}, \quad D := \mathrm{diag}(d_i).$$

The operation count given for Algorithm 4 assumes the use of one of these factorisations in step (1). The LU and LDL^T factorisations are

related by $U = DL^T$, so U and D have the same diagonal elements. It follows from (3.1.5) that the LDL^T factors of A in (3.6.1) are generated by

$$d_1 := a_1 \quad ; \quad \ell_i := \frac{b_i}{d_{i-1}} \quad , \quad d_i := a_i - \ell_i b_i \quad , \quad i = 2,3,\ldots,n.$$

We recommend use of the LDL^T factorisation in step (1) of Algorithm 4.

LINPACK [8] has routines SGTSL and SPTSL which solve $A\underline{x} = \underline{b}$ for general tridiagonal and positive definite tridiagonal matrices A respectively. Both routines reduce A to some compact form, during the reduction carrying out the operations in parallel on the right-hand side $\underline{b}$. As we show below, it is possible to "nest" Algorithm 4 inside the routine SPTSL in such a way that the composite routine computes $\text{cond}_\infty(A)$ in addition to solving $A\underline{x} = \underline{b}$ and is computationally more efficient than separate applications of SPTSL and Algorithm 4. Because SPTSL uses a nonstandard factorisation method which is more complicated than the LDT^T factorisation, we first derive the nesting technique for the LDL^T method.

The Nesting Technique

A key feature which Algorithm 4 does not exploit is that the LDL^T factorisations of A and M(A) are related: by comparison with (3.6.4) and (3.6.5), if $A = LDL^T$ then $M(A) = M(L)\ D\ M(L)^T$. Therefore, solution of $M(A)\underline{z} = \underline{e}$ can be accomplished using the LDL^T factorisation of A, which has to be computed anyway in the course of solving $A\underline{x} = \underline{b}$; there is no need to explicitly factorise M(A). The next algorithm makes use of this observation, thereby saving 2(n-1) flops.

Algorithm 4/1

Given the nxn (n > 2) positive definite tridiagonal matrix A in (3.6.1) and the vector $\underline{f}$, this algorithm computes both the solution to the linear system $A\underline{x} = \underline{f}$ and $\gamma = \text{cond}_\infty(A)$. The solution overwrites $\underline{f}$.

(1) $\mu := \max(|a_1| + |b_2|, |b_n| + |a_n|)$

For i := 2 to n-1
  $\mu := \max(\mu, |b_i| + |a_i| + |b_{i+1}|)$.

(2) $d_1 := a_1$; $z_1 := 1$

For i := 2 to n
  $\ell_i := b_i/d_{i-1}$
  $d_i := a_i - \ell_i * b_i$
  $f_i := f_i - \ell_i * f_{i-1}$
  $z_i := 1 + |\ell_i| * z_{i-1}$.

(3) $f_n := f_n/d_n$; $z_n := z_n/d_n$; $\lambda := z_n$

For i := n-1 to 1 step -1
  $f_i := f_i/d_i - \ell_{i+1} * f_{i+1}$
  $z_i := z_i/d_i + |\ell_{i+1}| * z_{i+1}$
  $\lambda := \max(\lambda, z_i)$.

(4) $\gamma := \mu * \lambda$.

Cost: $(5n - 4)M + (7n - 7)A + (3n - 1)D + (5n - 5)ABS$.

We remark that the vectors $\underline{\ell}$ and $\underline{d}$ have been included for clarity and

are not necessary. 2n storage locations can be saved by overwriting b_i with ℓ_i and a_i with d_i.

Ignoring step (1) - which could be optimised ($|b_i|$ is calculated twice) and incorporated into the loop in step (2) - and ignoring the absolute value computations, we note that

(1) Algorithm 4/1 uses 8n-6 flops of which only 3n-2 are attributable to computation of the condition number.

(2) Computation of $\text{cond}_\infty(A)$ does not introduce any loops over and above those required for the solution of the linear system. This is an important feature, since the loop overheads could account for a significant portion of the execution time of Algorithm 4/1 on a typical computer [9].

We conclude that on a typical computer, the execution time for a carefully coded version of Algorithm 4/1 should be about 60 percent greater than that of an equivalent routine which only solves $A\underline{x} = \underline{b}$.

We now show how Algorithm 4 can be nested within the LINPACK routine SPTSL. First, we describe the nonstandard reduction technique which this routine uses. The reduction consists of a form of Gaussian elimination without pivoting in which subdiagonal elements are eliminated using row operations working from the top and, simultaneously, superdiagonal elements are eliminated using row operations working from the bottom. Thus zeros are introduced to the elements (2,1),(n-1,n), (3,2), (n-2,n-1),..., in this order, until the two eliminations meet in the middle. During the reduction the same row operations are applied to the right-hand side $\underline{b}$.

For n = 5 the process is illustrated by

$$\begin{bmatrix} X & X & & & \\ X & X & X & & \\ & X & X & X & \\ & & X & X & X \\ & & & X & X \end{bmatrix} \rightarrow \begin{bmatrix} X & X & & & \\ 0 & X & X & & \\ & X & X & X & \\ & & X & X & 0 \\ & & & X & X \end{bmatrix} \rightarrow \begin{bmatrix} X & X & & & \\ 0 & X & X & & \\ & 0 & X & 0 & \\ & & X & X & 0 \\ & & & X & X \end{bmatrix}.$$

When n is odd, x_{k+1}, where $n = 2k + 1$, is trivially obtained from the reduced system by one division, the remaining unknowns being determined in the order x_k, x_{k+2}, x_{k-1},...,x_1, x_n.

When n is even the above process can be carried out until there is a 2x2 block left in the middle of the reduced matrix. One extra row elimination is used to zero the subdiagonal element of this 2x2 block. For example, for n = 4,

$$\begin{bmatrix} X & X & & \\ X & X & X & \\ & X & X & X \\ & & X & X \end{bmatrix} \rightarrow \begin{bmatrix} X & X & & \\ 0 & X & X & \\ & X & X & 0 \\ & & X & X \end{bmatrix} \rightarrow \begin{bmatrix} X & X & & \\ 0 & X & X & \\ & 0 & X & 0 \\ & & X & X \end{bmatrix}.$$

The substitution phase is essentially the same as for the case where n is odd.

The reduction is guaranteed to succeed when A is positive definite, indeed all the pivots are positive (see Lemma 3.5).

The motivation for this "two-way" algorithm is that each of its two loops (one for the reduction phase and one for the substitution phase) is executed only half as many times as the corresponding loop in a standard algorithm because the two-way algorithm performs two eliminations or

substitutions on each run through a loop. The LINPACK manual claims that the two-way algorithm, which is in line with the philosophy of Dongarra and Hinds [9], can solve positive definite tridiagonal linear systems up to 25 percent faster than conventional algorithms.

Consider the two-way algorithm applied to the matrix A in (3.6.1). It is easy to see that during the elimination stage those off-diagonal elements which are not zeroed are left unchanged and each diagonal element is affected by only one elimination, except for a_{p+1} if n is odd and a_{p+2} if n is even, where

$$p := \left[\frac{n-1}{2}\right],$$

these elements being affected by two eliminations. Using a dash to denote the new transformed element after an elimination step, the equations describing the elimination are, with $a_1' := a_1$ and $a_n' := a_n$,

$$\left.\begin{aligned} a_{i+1}' &:= a_{i+1} - \frac{b_{i+1}}{a_i'} \cdot b_{i+1} \\ a_{n-i}' &:= a_{n-i} - \frac{b_{n-i+1}}{a_{n-i+1}'} \cdot b_{n-i+1} \end{aligned}\right\} \quad i = 1,2,\ldots,p$$

$$a_{p+2}'' := a_{p+2}' - \frac{b_{p+2}}{a_{p+1}'} \cdot b_{p+2}, \text{ if n is even.} \qquad (3.6.8)$$

The equations in (3.6.8) correspond to premultiplication of the i th reduced matrix $A^{(i)}$ ($A := A^{(1)}$) by an elementary lower bidiagonal matrix

$$L_i := I - \ell_i \, \underline{e}_{i+1} \, \underline{e}_i^T$$

and an elementary upper bidiagonal matrix

$$U_{n-i+1} := I - m_i \underline{e}_{n-i} \underline{e}_{n-i+1}^T,$$

for i = 1,2,...,p. Thus the reduction can be expressed

$$L_o(U_{n-p+1} L_p) \cdots (U_{n-1} L_2) (U_n L_1) A = B, \qquad (3.6.9)$$

where

$$L_o := \begin{cases} I & , \quad n \text{ odd}, \\ L_{p+1} & , \quad n \text{ even}, \end{cases}$$

and the nonzero elements of B lie on the diagonal and in positions (1,2), (2,3),..., (k,k+1), (k+2,k+1), (k+3,k+2),..., (n,n-1), where k = p if n is odd and k = p+1 if n is even.

Using the fact that L_i commutes with U_j for all i and j, one obtains from (3.6.9),

$$(U_{n-p+1} \cdots U_{n-1} U_n) (L_o L_p \cdots L_2 L_1) A = B,$$

from which it follows that

$$A = L U B, \qquad (3.6.10)$$

where L is unit lower bidiagonal and U is unit upper bidiagonal.

The nesting technique in Algorithm 4/2 below is based on a result, applying to the factorisation (3.6 10), which is analogous to the relation between the LDL^T factorisations of A and M(A). This result is proved in Lemma 3.5. To facilitate the statement of the next two lemmas we introduce some more notation. Let

$$M(A) =: \begin{bmatrix} c_1 & d_2 & & 0 \\ d_2 & c_2 & d_3 & \\ & d_3 & c_3 & \\ & & & d_n \\ 0 & & d_n & c_n \end{bmatrix}. \qquad (3.6.11)$$

Thus

$$c_i = a_i = |a_i| \quad , \quad d_i = - |b_i| \qquad (3.6.12)$$

for all i. Since A is positive definite, M(A) is positive definite, as noted earlier. Therefore the two-way algorithm will not fail when applied to M(A). Let the factorisation corresponding to (3.6 10) for M(A) be

$$M(A) = \tilde{L}\,\tilde{U}\,\tilde{B}. \qquad (3.6.13)$$

Lemma 3.4

When the two-way algorithm is applied to M(A) in (3.6.11),

$$\acute{c}_{i+1} = \acute{a}_{i+1} \quad , \quad \acute{c}_{n-i} = \acute{a}_{n-i} \qquad (3.6.14)$$

for i = 1,2,...,p, and if n is even,

$$\acute{\acute{c}}_{p+2} = \acute{\acute{a}}_{p+2}. \qquad (3.6.15)$$

Proof

The proof is by induction. From (3.6.8) and (3.6.12),

$$\acute{c}_2 = c_2 - \frac{d_2^2}{c_1} = a_2 - \frac{(- |b_1|)^2}{a_1}$$

$$= a_2 - \frac{b_1^2}{a_1} = a_2'$$

and similarly $c_{n-1}' = a_{n-1}'$. If (3.6.14) holds for $i \leq k$ then from (3.6.8),

$$c_{k+1}' = c_{k+1} - \frac{d_{k+1}^2}{c_k'} = a_{k+1} - \frac{(-|b_{k+1}|)^2}{a_k'}$$

$$= a_{k+1} - \frac{b_{k+1}^2}{a_k'} = a_{k+1}'$$

and similarly $c_{n-k-1}' = a_{n-k-1}'$. By induction (3.6.14) is true for $i = i,2,\ldots,p$. (3.6.15) follows similarly. □

Lemma 3.5

Let the positive definite matrix A in (3.6.1) and its comparison matrix M(A) be factorised according to (3.6.10) and (3.6.13) respectively. Then B has positive diagonal elements, $\tilde{L} = M(L)$, $\tilde{U} = M(U)$ and $\tilde{B} = M(B)$.

Proof

From Lemma 3.4

$$\tilde{b}_{ii} := c_i' = a_i' =: b_{ii},$$

for all i. We show that $a_i' > 0$ for all i.

By comparing the elimination stage of the two-way algorithm on A with standard Gaussian elimination on A, for which it is known that the pivots are positive, one can deduce that $a_i' > 0$ for all but one value of i (which depends on the parity of n). The positivity of this remaining a_i' is assured because $0 < \det(A) = \det(B) = \prod_{i=1}^{n} b_{ii} = \prod_{i=1}^{n} a_i'$.

The off-diagonal elements of B and $\tilde{B}$ are either zero or agree with the corresponding elements of the original matrix. It follows that $\tilde{B} = M(B)$.

From (3.6.8), each multiplier for the elimination on M(A) is minus the modulus of the corresponding multiplier for the elimination on A (since $a_i' > 0$), therefore $\tilde{L} = M(L)$ and $\tilde{U} = M(U)$. □

Lemma 3.5 shows that the equation $M(A)\underline{z} = \underline{e}$ can be solved using, solely, information gleaned during the solution of $A\underline{x} = \underline{b}$ by the two-way algorithm. Thus we have

Algorithm 4/2

Given the nxn (n > 2) positive definite tridiagonal matrix A in (3.6.1) and the vector $\underline{f}$, this algorithm computes both the solution to the linear system $A\underline{x} = \underline{f}$ (this overwrites $\underline{f}$) and $\gamma = \text{cond}_\infty(A)$, employing the two-way algorithm used in the LINPACK routine SPTSL.

(1) (*) The same as step (1) of Algorithm 4/1.

(2) (*) For K := 1 to N
  z_K := 1

P := $\frac{N-1}{2}$; KB := N-1

For K := 1 to P
  t := b_{K+1}/a_K

  a_{K+1} := a_{K+1} - t*b_{K+1} ; f_{K+1} := f_{K+1} - t*f_K

(*) z_{K+1} := z_{K+1} + |t|*z_K

```
                t := b_{KB+1}/a_{KB+1}

                a_{KB} := a_{KB} - t*b_{KB+1}  ;  f_{KB} := f_{KB} - t*f_{KB+1}
(*)             z_{KB} := z_{KB} + |t|*z_{KB+1}
                KB := KB - 1

KP1 := P + 1

If N mod 2 = 0 then
                M := KP1 + 1
                t := b_M/a_{KP1}
                a_M := a_M - t*b_M  ;  f_M := f_M - t*f_{KP1}
(*)             z_M := z_M + |t| * z_{KP1}
                KP1 := KP1 + 1

                f_{KP1} := f_{KP1}/a_{KP1}
(*)             z_{KP1} := z_{KP1}/a_{KP1}  ;  λ := z_{KP1}
                K := KP1 - 1  ;  KE := KP1 + P - 1
```

For KF := KP1 to KE

$f_K := (f_K - b_{K+1}*f_{K+1})/a_K$

$f_{KF+1} := (f_{KF+1} - b_{KF+1}*f_{KF})/a_{KF+1}$

(*) $z_K := (z_K + |b_{K+1}|*z_{K+1})/a_K$

(*) $z_{KF+1} := (z_{KF+1} + |b_{KF+1}|*z_{KF})/a_{KF+1}$

(*) $\lambda := \max(\lambda, z_K, z_{KF+1})$

$K := K - 1$

If N mod 2 = 0 then

$f_1 := (f_1 - b_2*f_2)/a_1$

(*) $z_1 := (z_1 + |b_2|*z_2)/a_1$

(*) $\lambda := \max(\lambda, z_1)$.

(3) (*) $\gamma := \mu*\lambda$.

Cost: the same as for Algorithm 4/1.

Algorithm 4/2 consists of the two-way algorithm given in the LINPACK manual [8, pp. 7.4, 7.5, C.97, C.98] together with extra statements, marked with an asterisk, which evaluate the condition number. Clearly, only minor modifications are required to the LINPACK routine

SPTSL in order for this routine to compute the condition number $\text{cond}_1(A)$ ($= \text{cond}_\infty(A)$ since $A = A^T$) in addition to solving $A\underline{x} = \underline{b}$. Two extra parameters are required by the modified routine, a work vector Z of length n and a REAL variable RCONDE in which to return the reciprocal of the condition number, E denoting "exact" in order to avoid confusion with the estimate RCOND $\approx \text{cond}_1(A)^{-1}$ returned by some of the other LINPACK routines. The strategy of scaling to avoid overflow, adopted by LINPACK in all the code which performs condition estimation, is applicable to the computation of $\underline{z}$ where $M(A)\underline{z} = \underline{e}$ and a vector $\underline{w}$ satisfying

$$||A\underline{w}||_1 = \text{RCONDE}\ ||A||_1\ ||\underline{w}||_1$$

is easily obtained from $\underline{z}$ by using (3.6.3). A modification to the LINPACK scaling algorithm, which would be appropriate here, is described in [15].

Eigenvalue Bounds

For a positive definite tridiagonal matrix A, reliable bounds for $\lambda_{\min}(A) = ||A^{-1}||_2^{-1}$ are readily obtained using Algorithm 4 since, from Lemma 1.2,

$$\frac{1}{\sqrt{n}}\ ||A^{-1}||_\infty \leq ||A^{-1}||_2 = \rho(A^{-1}) \leq ||A^{-1}||_\infty. \quad (3.6.16)$$

Also,

$$\alpha \leq ||A||_2 = \rho(A) \leq ||A||_\infty \leq \sqrt{3}\ \alpha, \quad (3.6.17)$$

where

$$\alpha := \max_i\ ||A\underline{e}_i||_2.$$

Reliable bounds for $\text{cond}_2(A)$ result from combining (3.6.16) and (3.6.17).

3.7 Dealing With Reducibility

Algorithms 1, 2 and 3 each involve division by the off-diagonal elements $b_2, \ldots, b_n$ and $c_1, \ldots, c_{n-1}$ of A in (3.1.1), either explicitly (Algorithms 2 and 3) or implicitly (the division by ℓ_i in Algorithm 1). Therefore these algorithms break down when A is reducible. For particular classes of matrix it is possible to verify irreducibility in advance. Difference methods for boundary value problems can lead to tridiagonal matrices with off-diagonal elements of the form $\alpha + O(h)$ where $\alpha > 0$ and h is the mesh length [24, p. 96 ff.]; here, irreducibility is assured for sufficiently small h. The tridiagonal matrices which arise in the numerical solution of linear second order recurrence relations are frequently irreducible [2, pp. 14, 21]. However, there are also important situations where it is impossible to guarantee irreducibility. For example, a tridiagonal iteration matrix $A = I - h\gamma J(x,\underline{y})$ in a code for stiff O.D.E.s may have zero subdiagonal or superdiagonal elements for certain values of x and $\underline{y}$. In this section we extend Algorithms 1, 2 and 3 to deal with general tridiagonal matrices A.

If A is reducible and symmetric then A has the form $\mathrm{diag}(A_1, A_2, \ldots, A_k)$ where each matrix A_i is either diagonal or irreducible and tridiagonal. Hence if A is nonsingular, $||A^{-1}||_\infty = \max_i ||A_i^{-1}||_\infty$, which can be computed in O(n) operations by applying Algorithm 1, 2 or 3 to each of the matrices A_i, as appropriate.

If A is reducible and nonsymmetric, the situation is more complicated, though again $||A^{-1}||_\infty$ can be computed in O(n) operations. Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular reducible tridiagonal matrix. Using the recursive process alluded to in Theorem 3.3 A can be expressed in the form

$$A = \begin{bmatrix} A_1 & C_1 & & & 0 \\ B_2 & A_2 & C_2 & & \\ & B_3 & A_3 & \ddots & \\ & & \ddots & \ddots & C_{k-1} \\ 0 & & & B_k & A_k \end{bmatrix},$$

where at least one of B_{i+1} and C_i is a zero matrix for $i = 1, \ldots, n-1$ and the diagonal blocks A_i are square, tridiagonal and irreducible. It is easily verified that $\det(A) = \prod_{i=1}^{k} \det(A_i)$, from which it follows that the diagonal blocks are nonsingular. The technique we describe entails computing the row sums of the inverses of the leading principal block submatrices of A in the natural order, beginning with $A^{[1]} := A_1$ and finishing with $A^{[k]} := A$. Consider the general r th stage. There are three cases.

(a) $$A^{[r]} = \begin{bmatrix} A^{[r-1]} & 0 \\ \tilde{B}_r & A_r \end{bmatrix}, \quad A^{[r-1]} \in \mathbb{R}^{pxp}, 0 \neq \tilde{B}_r \in \mathbb{R}^{qxp}.$$

$\tilde{B}_r := [0\ B_r]$ has the form

$$\tilde{B}_r = \tilde{b}_r \underline{e}_1 \underline{e}_p^T. \tag{3.7.1}$$

we have,

$$A^{[r]^{-1}} = \begin{bmatrix} A^{[r-1]^{-1}} & 0 \\ -A_r^{-1}\tilde{B}_r A^{[r-1]^{-1}} & A_r^{-1} \end{bmatrix}, \tag{3.7.2}$$

where, by (3.7.1),

$$A_r^{-1}\tilde{B}_r A^{[r-1]^{-1}} = \tilde{b}_r (A_r^{-1}\underline{e}_1)(A^{[r-1]^{-T}}\underline{e}_p)^T. \tag{3.7.3}$$

In order to calculate the row sums of $A^{[r]^{-1}}$, the row sums of the three nonzero blocks in (3.7.2) are required. Assume that the row sums of $A^{[r-1]^{-1}}$ are available from the previous stage. Since A_r is irreducible, the row sums of A_r^{-1} can be computed using Algorithm 1, 2 or 3. The i th row sum of the matrix $\underline{x}\,\underline{y}^T$ is $|x_i|\ ||\underline{y}||_1$. Therefore, for computation of the row sums of the matrix in (3.7.3), both the last row sum of $A^{[r-1]^{-1}}$ and the vector consisting of the first column of A_r^{-1} are required.

(b) $$A^{[r]} = \begin{bmatrix} A^{[r-1]} & \tilde{C}_{r-1} \\ 0 & A_r \end{bmatrix}, \quad A^{[r-1]} \in \mathbb{R}^{pxp}, 0 \neq \tilde{C}_{r-1} \in \mathbb{R}^{pxq}.$$

$\tilde{C}_{r-1} := \begin{bmatrix} 0 \\ C_{r-1} \end{bmatrix}$ has the form

$$\tilde{C}_{r-1} = \tilde{c}_{r-1}\, \underline{e}_p\, \underline{e}_1^T. \tag{3.7.4}$$

we have,

$$A^{[r]^{-1}} = \begin{bmatrix} A^{[r-1]^{-1}} & -A^{[r-1]^{-1}}\, \tilde{C}_{r-1}\, A_r^{-1} \\ 0 & A_r^{-1} \end{bmatrix},$$

where, by (3.7.4),

$$A^{[r-1]^{-1}}\, \tilde{C}_{r-1}\, A_r^{-1} = \tilde{c}_{r-1}(A^{[r-1]^{-1}}\, \underline{e}_p)(A_r^{-T}\, \underline{e}_1)^T. \tag{3.7.5}$$

Computation of the row sums of $A^{[r]^{-1}}$ is similar to case (a) except that here the first row sum of A_r^{-1} and the vector consisting of the last column of $A^{[r-1]^{-1}}$ are required in order to compute the row sums of the matrix in (3.7.5).

(c) $A^{[r]} = \begin{bmatrix} A^{[r-1]} & 0 \\ 0 & A_r \end{bmatrix}$.

This case is straightforward.

The key fact is that the information which needs to be carried over from the (r-1)st stage to the r th stage consists solely of (1) the row sums of $A^{[r-1]^{-1}}$ and, if case (b) pertains at the r th stage, (2) the last column of $A^{[r-1]^{-1}}$.

The computational cost of the algorithm outlined above and the storage required are dependent on the precise form of A. The maximum possible storage needed amounts to 6n locations: n for the row sums, n for the last column of $A^{[r-1]^{-1}}$ and 4n for the vectors $\underline{x}$, $\underline{y}$, $\underline{p}$ and $\underline{q}$ which specify A_r^{-1}. By linearity of the costs of Algorithms 1, 2 and 3, the cost of computing the row sums of the $\{A_r^{-1}\}$ is essentially the same as the cost of applying Algorithm 1, 2 or 3 to an nxn irreducible tridiagonal matrix. The extra costs in the above algorithm, associated with computation of the last column of $A^{[r-1]^{-1}}$ and the row sums in (3.7.3) and (3.7.5), do not exceed about 5n flops. One worthwhile refinement† if A has groups of consecutive zeros on the superdiagonal or subdiagonal is to allow the diagonal blocks $\{A_r\}$ to be bidiagonal and to use the fact that the row sums of B^{-1}, for bidiagonal B, are the components of $M(B)^{-1}\underline{e}$ (see section 3.2).

Thus, computation of $\text{cond}_\infty(A)$ for a general tridiagonal matrix is not substantially more expensive than computation of $\text{cond}_\infty(A)$ for an irreducible tridiagonal matrix.

---

†In fact, this refinement is essential in order to avoid the possibility of consecutive case (b)s, which could lead to an $O(n^2)$ operation count.

3.8 Computational Considerations

While the irreducibility of A guarantees that Algorithms 1, 2 and 3 will not fail, difficulties must be expected in practice when some or all of the superdiagonal and subdiagonal elements are small but nonzero. The major computational hazard is overflow due to division by a small number, but loss of accuracy is also possible. These problems can occur even when the matrix is well-conditioned, as illustrated by the example

$$A := \begin{bmatrix} 1 & \varepsilon \\ 1 & 1 \end{bmatrix}, \quad |\varepsilon| << 1, \quad \mathrm{cond}_\infty(A) \approx 4.$$

However, by use of the scalings described below, the difficulties caused by "near reducibility" can be overcome.

Consider equations (3.4.4) and (3.4.5) with $x_1 := 1$. The vector $\underline{y}$ is the first row of A^{-1}, therefore unless A is ill-conditioned the components of $\underline{y}$ will be of moderate size and overflow is very unlikely when (3.4.5) is solved using, say, the factorisation (3.1.3) of A. Also, $y_n \underline{x}$ is the last column of A^{-1}. Therefore it is natural to rewrite (3.4.4) as $A(y_n \underline{x}) = \underline{e}_n$ and solve for $y_n \underline{x}$; the same comments now apply as for (3.4.5). Similarly, it is natural to solve for $p_n \underline{q}$ rather than $\underline{q}$. Note that the presence of small superdiagonal or subdiagonal elements in the matrix A does not force any of the vectors $\underline{y}$, $y_n \underline{x}$, $\underline{p}$ and $p_n \underline{q}$ to have large components. The results of section 1.4 suggest that the accuracy of the computed approximations to these four vectors will reflect the condition of A. This is the best that can be expected since it can be shown that the condition number for the problem of computing $||A^{-1}||$ is cond(A).

These considerations suggest the following algorithm.

Algorithm 5

Given the nxn nonsingular irreducible tridiagonal matrix A in (3.1.1) this algorithm computes $\gamma = ||A^{-1}||_\infty$.

(1) Compute the factorisation PA = LU by Gaussian elimination with partial pivoting.

(2) Solve the equations

$$A^T\underline{y} = \underline{e}_1,$$

$$A\underline{z} = \underline{e}_n, \qquad (\underline{z} = y_n\underline{x})$$

$$A\underline{p} = \underline{e}_1,$$

$$A^T\underline{r} = \underline{e}_n, \qquad (\underline{r} = p_n\underline{q})$$

using the factorisation computed in step (1).

(3) $s_n := |y_n|$

For i := n-1 to 1 step -1
  $s_i := s_{i+1} + |y_i|$

$t_1 := |r_1|$

For i := 2 to n
  $t_i := t_{i-1} + |r_i|$

$$\gamma := \max((|z_1| * s_1)/|y_n|,\ t_n)$$

For i := 2 to n-1

    $\gamma := \max(\gamma,\ (|p_i| * t_{i-1})/|p_n| + (|z_i| * s_i)/|y_n|)$

Cost: $16n\ (M + A) + 7n\ D + 4n\ ABS + O(1)\ (M + A + D + ABS)$.

Remarks

(1) The operation count is obtained from (iv) and (v) in Section 3.1 and is a worst case. The precise operation count depends on the pivots used in step (1).

(2) Step (3) consists of a modified version of Algorithm Norm. The rescaling by y_n^{-1} and p_n^{-1} is performed at the last possible stage. If overflow occurs in step (3) then $||A^{-1}||_\infty$ itself would overflow.

(3) A particularly pleasing aspect of the algorithm is that Gaussian elimination with partial pivoting is stable when applied to a tridiagonal matrix, for the growth factor is bounded by two [27, p. 158].

(4) The scaling strategy utilised in Algorithm 5 could be incorporated into Algorithm 1, producing only a modest increase in cost. Alternatively, the LU factorisation without pivoting could be used in Algorithm 5. This would result in a much more favourable operation count.

REFERENCES

[1] BUKHBERGER, B. and EMEL'YANENKO, G.A. (1974) Methods of inverting tridiagonal matrices, U.S.S.R. Computational Math. and Math. Phys. 13, 10-20.

[2] CASH, J.R. (1979) Stable Recursions: with Applications to the Numerical Solution of Stiff Systems, Academic Press, London.

[3] CLINE, A.K. (1981) A set of counter-examples to the LINPACK condition number estimator, Manuscript, Comp. Sci. Dept., University of Texas at Austin.

[4] CLINE, A.K., CONN, A.R. and VAN LOAN, C.F. (1982) Generalizing the LINPACK condition estimator, in HENNART, J.P. [ed.] (1982) Numerical Analysis, Mexico 1981, Lecture Notes in Mathematics 909, Springer-Verlag, Berlin, 73-83.

[5] CLINE, A.K., MOLER, C.B., STEWART, G.W. and WILKINSON, J.H. (1979) An estimate for the condition number of a matrix, SIAM J. Numer. Anal. 16, 368-375.

[6] DAHLQUIST, G. and BJORCK, A. (1974) Numerical Methods, Prentice-Hall, Englewood Cliffs, N.J.

[7] DIXON, J.D. (1983) Estimating extremal eigenvalues and condition numbers of matrices, SIAM J. Numer. Anal. 20, 812-814.

[8] DONGARRA, J.J., BUNCH, J.R., MOLER, C.B. and STEWART, G.W. (1979) LINPACK Users' Guide, SIAM publications, Philadelphia.

[9] DONGARRA, J.J. and HINDS, A.R. (1979) Unrolling loops in FORTRAN, Software-Practice and Experience 9, 216-226.

[10] FISCHER, C.F. and USMANI, R.A. (1969) Properties of some tri-diagonal matrices and their applications to boundary value problems, SIAM J. Numer. Anal. 6, 127-142.

[11] FORSYTHE, G.E., MALCOLM, M.A. and MOLER, C.B. (1977) Computer Methods for Mathematical Computations, Prentice-Hall, Englewood Cliffs, N.J.

[12] FROBERG, C.-E. (1969) Introduction to Numerical Analysis (Second edition), Addison-Wesley, Reading, Massachusetts.

[13] GOLUB, G.H. and REINSCH, C. (1970) Singular value decomposition and least squares solutions, Numer. Math. 14, 403-420.

[14] GRAYBILL, F.A. (1969) Introduction to Matrices with Applications in Statistics, Wadsworth, Belmont, California.

[15] GRIMES, R.G. and LEWIS, J.G. (1981) Condition number estimation for sparse matrices, SIAM J. Sci. Stat. Comput. 2, 384-388.

[16] HAMMARLING, S. and WILKINSON, J.H. (1980) On linear systems arising from finite difference approximations to elliptic differential equations, Report DNACS 34/80, National Physical Laboratory, England.

[17] HIGHAM, N.J. (1983) Upper bounds for the condition number of a triangular matrix, Numerical Analysis Report No. 86, University of Manchester.

[18] HOUSEHOLDER, A.S. (1964) The Theory of Matrices in Numerical Analysis, Blaisdell, New York.

[19] IKEBE, Y. (1979) On inverses of Hessenberg matrices, Linear Algebra and Appl. 24, 93-97.

[20] KAHAN, W. (1966) Numerical linear algebra, Canadian Math. Bulletin 9, 757-801.

[21] LANCASTER, P. (1969) Theory of Matrices, Academic Press, New York.

[22] MOLER, C.B. (1982) MATLAB Users' guide, Technical Report CS81-1 (revised), Department of Computer Science, University of New Mexico.

[23] O'LEARY, D.P. (1980) Estimating matrix condition numbers, SIAM J. Sci. Stat. Comput. 1, 205-209.

[24] ORTEGA, J.M. (1972) Numerical Analysis: A Second Course, Academic Press, New York.

[25] RICE, J.R. (1981) Matrix Computations and Mathematical Software, McGraw-Hill, New York.

[26] SKEEL, R.D. (1979) Scaling for numerical stability in Gaussian elimination, J. Assoc. Comput. Mach. 26, 494-526.

[27] STEWART, G.W. (1973) Introduction to Matrix Computations, Academic Press, New York.

[28] STEWART, G.W. (1980) The efficient generation of random orthogonal matrices with an application to condition estimators, SIAM J. Numer. Anal. 17, 403-409.

[29] STOER, J. and BULIRSCH, R. (1980) Introduction to Numerical Analysis, Springer-Verlag, New York.

[30] TODD, J. (1978) Basic Numerical Mathematics Vol. 2: Numerical Algebra, Academic Press, New York.

[31] VAN DER SLUIS, A. (1969) Condition numbers and equilibration of matrices, Numer. Math. 14, 14-23.

[32] VAN DER SLUIS, A. (1970) Stability of solutions of linear algebraic systems, Numer. Math. 14, 246-251.

[33] WILKINSON, J.H. (1961) Error analysis of direct methods of matrix inversion, J. Assoc. Comput. Mach. 8, 281-330.

[34] WILKINSON, J.H. (1978) Singular-value decomposition - basic aspects, in JACOBS, D. [ed.] (1978) Numerical Software - Needs and Availability, Academic Press, New York, 109-135.

[35] YAMAMOTO, T. and IKEBE, Y. (1979) Inversion of band matrices, Linear Algebra and Appl. 24, 105-111.