Numer. Math. 62, 539–555 (1992)

Numerische Mathematik
© Springer-Verlag 1992

Estimating the matrix p-norm

Nicholas J. Higham
Nuffield Science Research Fellow, Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK

Received November 22, 1991

Summary. The Hölder p-norm of an $m \times n$ matrix has no explicit representation unless $p = 1, 2$ or ∞. It is shown here that the p-norm can be estimated reliably in $O(mn)$ operations. A generalization of the power method is used, with a starting vector determined by a technique with a condition estimation flavour. The algorithm nearly always computes a p-norm estimate correct to the specified accuracy, and the estimate is always within a factor $n^{1-1/p}$ of $\|A\|_p$. As a by-product, a new way is obtained to estimate the 2-norm of a rectangular matrix; this method is more general and produces better estimates in practice than a similar technique of Cline, Conn and Van Loan.

Mathematical Subject Classification (1991): 65F35

1 Introduction

Four matrix norms are commonly used in scientific computing: the 1, 2 and ∞-norms and the Frobenius norm. The first three are particular cases of the Hölder p-norm

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}, \quad A \in \mathbb{R}^{m \times n},$$

where $p \geqq 1$ and $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$. For $p = 1, \infty$ the norm is given explicitly in terms of the elements of A by

$$\|A\|_\infty = \max_{1 \leqq i \leqq m} \sum_{j=1}^n |a_{ij}|, \qquad \|A\|_1 = \|A^T\|_\infty,$$

and the 2-norm has the characterisation $\|A\|_2 = \rho(A^T A)^{1/2}$, where ρ denotes the spectral radius. No such formulas for $\|A\|_p$ are known for other values of p, and how best to estimate or compute $\|A\|_p$ is an open question that we address in this work.

The problem of computing $\|A\|_p$ is of interest for several reasons. First, it has connections with matrix condition number estimation [15]. As a by-product of this

work we obtain a new way to estimate $\|A\|_2$ as well as further insight into the condition number estimator used in LAPACK. Second, p-norms are well-studied in approximation theory and various algorithms have been developed for approximation in the p-norm [7, 22, 23, 32]. The ability to compute the p-norm of a matrix may be useful in this context; for example, one may wish to compute the relative residual $\|b - Ax\|_p/(\|A\|_p\|x\|_p)$ for an approximate l_p solution to an overdetermined system. Further motivation for this work is that we have occasionally found it inconvenient that MATLAB's built-in function norm [24] is unable to compute $\|A\|_p$ if $p \neq 1, 2$, or ∞, although it will compute $\|x\|_p$ for all $p \geqq 1$. Using the algorithms developed here we have written a MATLAB M-file *pnorm.m* that overcomes this limitation (see Appendix); this M-file has the desirable property that it works with a sparse matrix argument in MATLAB 4.0 [9].

We will assume throughout that vectors and matrices are real. All our algorithms are valid for complex matrices if transposes are changed to conjugate transposes, but the convergence results from [2] and [3] described in Sect. 2 have been proved only for real A.

Before considering the numerical computation of $\|A\|_p$ we summarise some useful theoretical results about the p-norm. A fundamental inequality for vectors is the Hölder inequality

$$x^T y \leqq \|x\|_p \|y\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1 . \tag{1.1}$$

This is an equality when $p, q > 1$ if the vectors $(|x_i|^p)$ and $(|y_i|^q)$ are linearly dependent and $\operatorname{sign}(x_i y_i)$ is constant for all i; equality is also possible when $p = 1$ and $p = \infty$, as is easily verified. For a general vector norm $\|\cdot\|$ the dual norm is defined by

$$\|x\|_D = \max_{z \neq 0} \frac{z^T x}{\|z\|} . \tag{1.2}$$

It follows from the Hölder inequality that the dual of the p-norm is the q-norm, where $p^{-1} + q^{-1} = 1$.

How much two Hölder norms of a vector can differ is shown by the attainable inequalities [8, p. 28], [11, Lemma 1.1]

$$\|x\|_{p_2} \leqq \|x\|_{p_1} \leqq n^{\left(\frac{1}{p_1} - \frac{1}{p_2}\right)} \|x\|_{p_2}, \quad p_1 < p_2 . \tag{1.3}$$

Using (1.3), and (1.13) below, one can derive the upper bounds in

$$\max_j \|A(:,j)\|_p \leqq \|A\|_p \leqq n^{1-1/p} \max_j \|A(:,j)\|_p , \tag{1.4}$$

$$\max_i \|A(i,:)\|_{p/(p-1)} \leqq \|A\|_p \leqq m^{1/p} \max_i \|A(i,:)\|_{p/(p-1)} , \tag{1.5}$$

where we have used MATLAB-style indexing notation, as in [12].

Matrix norms can be compared using the following elegant result of Schneider and Strang [28] (see also [20, p. 303]): if $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ denote two vector norms and the corresponding subordinate matrix norms, then for $A \in \mathbb{R}^{m \times n}$

$$\max_{A \neq 0} \frac{\|A\|_\alpha}{\|A\|_\beta} = \left(\max_{0 \neq x \in \mathbb{R}^n} \frac{\|x\|_\alpha}{\|x\|_\beta} \right) \left(\max_{0 \neq x \in \mathbb{R}^n} \frac{\|x\|_\beta}{\|x\|_\alpha} \right) . \tag{1.6}$$

From (1.3) and (1.6), we have, when $m = n$,

$$\max_{A \neq 0} \frac{\|A\|_{p_1}}{\|A\|_{p_2}} = n^{\left(\frac{1}{\min(p_1, p_2)} - \frac{1}{\max(p_1, p_2)}\right)} . \tag{1.7}$$

Note that, unlike for vectors, $p_1 < p_2$ does not imply $\|A\|_{p_1} \geqq \|A\|_{p_2}$. The result (1.7) implies, for example, that for all $p \geqq 1$

$$\frac{\|A\|_1}{n^{1-1/p}} \leqq \|A\|_p \leqq n^{1-1/p} \|A\|_1 , \tag{1.8}$$

$$\frac{\|A\|_2}{n^{|1/p-1/2|}} \leqq \|A\|_p \leqq n^{|1/p-1/2|} \|A\|_2 . \tag{1.9}$$

Upper bounds for $\|A\|_p$ that do not involve m or n can be obtained from the interesting property that $\log \|A\|_p$ is a convex function of $1/p$ for $p \geqq 1$ (see Fig. 1), which is a consequence of the Riesz-Thorin theorem [14, pp. 214, 219], [10]. The convexity implies that if $f(\alpha) = \|A\|_{1/\alpha}$, then for $0 \leqq \alpha, \beta \leqq 1$,

$$\log f(\theta\alpha + (1-\theta)\beta) \leqq \theta \log f(\alpha) + (1-\theta) \log f(\beta), \quad 0 \leqq \theta \leqq 1 .$$

Writing $p_1 = 1/\alpha$ and $p_2 = 1/\beta$, this inequality can be expressed as

$$\|A\|_p \leqq \|A\|_{p_1}^{\theta} \|A\|_{p_2}^{1-\theta}, \qquad p = \frac{p_1 p_2}{(1-\theta)p_1 + \theta p_2}, \tag{1.10}$$

$$1 \leqq p_1, p_2 \leqq \infty, \quad 0 \leqq \theta \leqq 1 .$$

Two interesting special cases are

$$\|A\|_p \leqq \|A\|_1^{1/p} \|A\|_\infty^{1-1/p} , \tag{1.11}$$

which is proved directly in [21, p. 29] and [30, pp. 25–26], and

$$\|A\|_p \leqq \|A\|_1^{2/p-1} \|A\|_2^{2-2/p}, \quad 1 \leqq p \leqq 2 . \tag{1.12}$$

Note that a special case of (1.11) is the well-known inequality $\|A\|_2 \leqq \sqrt{\|A\|_1 \|A\|_\infty}$.

Finally, two further results that are familiar for $p = 1, 2, \infty$ are

$$\|A^{\mathrm{T}}\|_p = \|A\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1 , \tag{1.13}$$

(see, for example, [20, p. 309]) and

$$\left\| \begin{bmatrix} 0 & A \\ A^{\mathrm{T}} & 0 \end{bmatrix} \right\|_p = \max(\|A\|_p, \|A\|_q) .$$

The bounds (1.8) and (1.9) imply that given the ability to compute $\|A\|_1$, $\|A\|_2$ and $\|A\|_\infty$ we can estimate $\|A\|_p$ correct to within a factor $n^{1/4}$. These a priori estimates are at their best when p is close to 1, 2 or ∞, but in general they will not provide even one correct significant digit. The bound in (1.10) can be much smaller than the other upper bounds given above, but how tight it is depends on how nearly $\log \|A\|_p$ is linear in p. To obtain better estimates numerical methods are needed.

It is clear from the definition that computing $\|A\|_p$ is a nonlinear optimization problem over $\mathbb{R}^n$. The objective function is non-convex, so there will usually be local maxima having function values less than the global maximum $\|A\|_p$. We do not know of any numerical method that can guarantee to compute the global maximum at reasonable cost. We present here three methods that estimate $\|A\|_p$ in $O(mn)$ flops and which often return an estimate that is exact or has several correct significant digits. (A flop is any floating point operation [12].) Thus these methods are in the same spirit as matrix condition number estimators.

The three methods are described in Sects. 2–4. The first method is a generalization of the well-known power method. The second method uses condition number estimation ideas and is not iterative. The third method iteratively computes a sequence of weighted 2-norms. Numerical experiments with the methods are presented in Sect. 5. In particular, we test a hybrid method that uses the method of Sect. 3 to provide a starting vector for the p-norm power method; this hybrid method is the one we recommend in Sect. 6.

It is worth noting a connection with the total approximation problem [26, 33]. This problem can be transformed to the problem

$$\text{minimize } \|Bv\|_\alpha \text{ subject to } \|v\|_\beta = 1 ,$$

where B is $m \times n$. If B is square and nonsingular, α and β represent a p-norm, and $A = B^{-1}$, then the minimum is $1/\|A\|_p$. Numerical methods for the total approximation problem [26, 27] can therefore be used to compute $\|A\|_p$ when A is square and nonsingular. However, numerical methods are only available for certain p, and the special-purpose methods described here are more efficient.

2 The power method

First, we consider an iterative "power method" for computing $\|A\|_p$. It reduces to the usual power method applied to A^TA when $p = 2$. We use the notation $\text{dual}_p(x)$ to denote any vector y of unit q-norm such that equality holds in the Hölder inequality (1.1). Throughout, q is defined by $p^{-1} + q^{-1} = 1$.

Algorithm PM. Given $A \in \mathbb{R}^{m \times n}$ and $x_0 \in \mathbb{R}^n$ this algorithm computes γ and x such that $\gamma \leqq \|A\|_p$ and $\|Ax\|_p = \gamma \|x\|_p$.

$x_0 = x_0/\|x_0\|_p$
repeat
    $y = Ax$
    $z = A^T \text{dual}_p(y)$
    if $\|z\|_q \leqq z^Tx$
        $\gamma = \|y\|_p$
        quit
    end
    $x = \text{dual}_q(z)$
end

Algorithm PM requires about $4rmn$ flops if there are r iterations for convergence. The convergence test can be written in several different ways, as we explain below;

the form chosen here is the one used in [13, 16, 17]. The power method was first derived and analysed by Boyd [3], and it was later investigated by Tao [29]. Tao applies the method to an arbitrary mixed subordinate norm

$$\|A\|_{\alpha,\beta} = \max_{x \neq 0} \frac{\|Ax\|_\alpha}{\|x\|_\beta}, \tag{2.1}$$

while Boyd takes the α and β-norms to be p-norms (possibly different). Algorithm PM can be converted to estimate $\|A\|_{\alpha,\beta}$ by making straightforward modifications to the norm-dependent terms.

There are several ways to derive Algorithm PM. Perhaps the most natural way is to examine the optimality conditions for

$$F(x) = \frac{\|Ax\|_p}{\|x\|_p}.$$

First, we note that the subdifferential (that is, the set of subgradients) of an arbitrary vector norm $\|\cdot\|$ is given by [6, p. 379]

$$\partial\|x\| = \{\lambda : \lambda^T x = \|x\|, \|\lambda\|_D \leqq 1\}.$$

If $x \neq 0$ then $\lambda^T x = \|x\| \Rightarrow \|\lambda\|_D \geqq 1$, from (1.2), and so if $x \neq 0$

$$\begin{aligned}\partial\|x\| &= \{\lambda : \lambda^T x = \|x\|, \|\lambda\|_D = 1\} \\ &\equiv \{\mathrm{dual}(x)\}.\end{aligned}$$

It can also be shown that if A has full rank,

$$\partial\|Ax\| = \{A^T \mathrm{dual}(Ax)\}.$$

We assume now that A has full rank, $1 < p < \infty$ and $x \neq 0$. Then it is easy to see that there is a unique vector $\mathrm{dual}_p(x)$, so $\partial\|x\|_p$ has just one element, that is, $\|x\|_p$ is differentiable. Hence we have

$$\nabla F(x) = \frac{A^T \mathrm{dual}_p(Ax)}{\|x\|_p} - \frac{\|Ax\|_p \mathrm{dual}_p(x)}{\|x\|_p^2}. \tag{2.2}$$

The first order Kuhn-Tucker condition for a local maximum of F is therefore

$$A^T \mathrm{dual}_p(Ax) = \frac{\|Ax\|_p}{\|x\|_p} \mathrm{dual}_p(x).$$

Since $\mathrm{dual}_q(\mathrm{dual}_p(x)) = x/\|x\|_p$ if $p \neq 1, \infty$, this equation can be written as

$$x = \frac{\|x\|_p^2}{\|Ax\|_p} \mathrm{dual}_q(A^T \mathrm{dual}_p(Ax)). \tag{2.3}$$

The power method is simply functional iteration applied to this transformed set of Kuhn-Tucker equations (the scale factor $\|x\|_p^2/\|Ax\|_p$ is irrelevant since $F(\alpha x) = F(x)$). A different derivation can be given for the 1 and ∞-norms, which are not everywhere differentiable: the subgradient inequality [6, p. 364] is used to try to maximize the convex function $\|Ax\|_p$ over the convex set $\{x : \|x\|_p \leqq 1\}$. For

all $1 \leqq p \leqq \infty$ the power method has the desirable property of generating an increasing sequence of norm approximations.

Lemma 2.1. *In Algorithm PM, the vectors from the kth iteration satisfy*
(i) $z^{k^T}x^k = \|y^k\|_p$, *and*
(ii) $\|y^k\|_p \leqq \|z^k\|_q \leqq \|y^{k+1}\|_p \leqq \|A\|_p$.
The first inequality in (ii) *is strict if convergence is not obtained on the kth iteration.*

Proof. $z^{k^T}x^k = \mathrm{dual}_p(y^k)^T A x^k = \mathrm{dual}_p(y^k)^T y^k = \|y^k\|_p$. Then $\|y^k\|_p = z^{k^T}x^k \leqq \|z^k\|_q \|x^k\|_p = \|z^k\|_q = z^{k^T}x^{k+1} = \mathrm{dual}_p(y^k)^T A x^{k+1} \leqq \|\mathrm{dual}_p(y^k)\|_q \|Ax^{k+1}\|_p = \|y^{k+1}\|_p \leqq \|A\|_p$. For the last part, note that in view of (i) the convergence test "$\|z^k\|_q \leqq z^{k^T}x^k$" can be written as "$\|z^k\|_q \leqq \|y^k\|_p$" □

It is clear from Lemma 2.1 that the convergence test "$\|z\|_q \leqq z^T x$" in Algorithm PM is equivalent to "$\|z\|_q = z^T x$", and since $\|x\|_p = 1$ this is equivalent to $x = \mathrm{dual}_q(z)$. Thus, although the convergence test compares two scalars, it is actually testing for equality in (2.3).

The convergence properties of Algorithm PM are as follows. First, in view of Lemma 2.1, the scalars $\gamma_k = \|y^k\|_p$ form an increasing and convergent sequence. This does not necessarily imply that Algorithm PM converges, since the algorithm tests for convergence of the x^k, and these vectors could fail to converge. However, a subsequence of the x^k must converge to a limit, $\bar{x}$ say. Boyd [3] shows that if $\bar{x}$ is a strong local maximum of F with no zero components, then $x^k \to \bar{x}$ linearly.

If Algorithm PM converges it converges to a stationary point of $F(x)$ when $1 < p < \infty$. Thus, instead of the desired global maximum $\|B\|_p$, we may obtain only a local maximum or even a saddle point. When $p = 1$ or ∞, if the algorithm converges to a point at which F is not differentiable, that point need not even be a stationary point. On the other hand, for $p = 1$ or ∞ Algorithm PM terminates in at most $n + 1$ iterations (assuming that when dual_p or dual_q is not unique an extreme point of the unit ball is taken), since the algorithm moves between the vertices e_i of the unit ball in the 1-norm, increasing F on each stage ($x = \pm e_i$ for $p = 1$, and $\mathrm{dual}_p(y) = \pm e_i$ for $p = \infty$). An example where n iterations are required for $p = 1$ is given in [17]. In fact, a more general result holds [2]. If the power method is applied to the norm (2.1) and one of the α and β norms is polyhedral (that is, its unit ball has a finite number of extreme points), then the iteration converges in a finite number of steps. Moreover, under a reasonable assumption, this number of steps can be bounded in terms of the number of extreme points of the unit balls in the α-norm and the dual of the β-norm. See [2] for a detailed analysis of the power method applied to (2.1) where α or β is polyhedral.

Other pleasing properties of Algorithm PM are as follows.

(1) If $A = xy^T$ (rank 1), the algorithm converges on the second step with $\gamma = \|A\|_p = \|x\|_p \|y\|_q$, whatever x_0.

(2) Boyd [3] shows that if A has nonnegative elements, $A^T A$ is irreducible, $1 < p < \infty$, and x_0 has positive elements, then the x^k converge and $\gamma_k \to \|A\|_p$.

In the case $p = 1$, Algorithm PM is a 1-norm estimation algorithm devised by Hager [13] (independently of [3] and [29]) and subsequently analysed and modified by the present author [16, 17]. The algorithm given in [16] is the basis of all condition number estimation in LAPACK [1]. Algorithm PM has two remarkable properties when $p = 1$: it almost always converges within four iterations (when

$x_0 = (1, 1, \ldots, 1)^T$, say) and it frequently yields $\|A\|_1$ exactly. This rapid, finite termination is also obtained for $p = \infty$, and is related to the fact that Algorithm PM moves amongst the finite set of extreme points of the unit ball. The question arises of whether Algorithm PM performs better for $p = \infty$ than for $p = 1$. Our numerical experiments suggest that the accuracy is similar for the two norms but slightly more iterations are required on average for $p = \infty$ (we have taken into account the fact that the convergence test in Algorithm PM is sensitive to rounding errors for $p = \infty$); a similar observation is made in [2].

For other values of p the convergence behaviour is typical of a linearly convergent method: exact convergence is not usually obtained in a finite number of steps and arbitrarily many steps can be required for convergence, as is well-known for the 2-norm power method. We can summarise, then, with the pleasing statement that the LAPACK condition estimator uses the "best of the p-norm power methods".

In view of the potentially slow convergence, it is desirable to supplement the convergence test of Algorithm PM with a test for convergence of the norm estimates $\gamma_k = \|y^k\|_p$. We use the convergence test

$$\text{(2.4)} \qquad \text{if}\left(\|z\|_q \leqq z^T x \text{ or } \frac{\gamma_k - \gamma_{k-1}}{\gamma_k} \leqq \text{tol} \right) \quad \text{and} \quad k > 1 ,$$

where tol is a tolerance. We force at least two iterations to be taken ($k > 1$) since this was found to be desirable when $p = 1$ [16].

Another important practical consideration is the choice of starting vector. In the absence of particular knowledge of A or p, $x_0 = (1, 1, \ldots, 1)^T$ is a natural choice. In the next two sections we derive two new algorithms for estimating $\|A\|_p$. These can be regarded as ways to generate a good starting vector for Algorithm PM.

3 A one step estimator

Our second method for estimating $\|A\|_p$ is not iterative. It chooses the components of x in the order $x_1, x_2, \ldots, x_n$ in an attempt to maximize $\|Ax\|_p / \|x\|_p$. Suppose $x_1, \ldots, x_{k-1}$ satisfying $\|x(1:k-1)\|_p = 1$ have been determined and let $\gamma_{k-1} = \|A(:, 1:k-1)x(1:k-1)\|_p$. We now try to choose x_k, and at the same time revise $x(1:k-1)$, to give the next partial product a larger norm. Defining

$$g(\lambda, \mu) = \lambda A(:, 1:k-1)x(1:k-1) + \mu A(:, k)$$

we set

$$x_k = \mu^*, \qquad x(1:k-1) \leftarrow \lambda^* x(1:k-1) ,$$

where

$$g(\lambda^*, \mu^*) = \max\left\{ g(\lambda, \mu): \left\| \begin{bmatrix} \lambda \\ \mu \end{bmatrix} \right\|_p = 1 \right\} .$$

Then $\|x(1:k)\|_p = 1$ and

$$\gamma_k = \|A(:, 1:k)x(1:k)\|_p \geqq \gamma_{k-1} .$$

This method for choosing x is similar to the look-behind/look-ahead method of [5, 31] for estimating $\|T^{-1}\|_2$ or $\|T\|_2$, where T is a triangular matrix. However,

that algorithm chooses the right-hand side d to make $\|x\|_2$ large (for $\|T^{-1}\|_2$) or small (for $\|T\|_2$), where $Tx = d$, and it relies on the triangularity of T. Our algorithm can be summarised as follows.

Algorithm OSE. Given $A \in \mathbb{R}^{m\times n}$ this algorithm computes γ and x such that $\gamma \leqq \|A\|_p$ and $\|Ax\|_p = \gamma\|x\|_p$.

$y = 0$
for $k = 1:n$
  if $k = 1$
    $\lambda^* = 0$, $\mu^* = 1$
  else
    Find $\begin{bmatrix}\lambda^*\\ \mu^*\end{bmatrix}$ that maximizes $\left\|[yA(:,k)]\begin{bmatrix}\lambda\\ \mu\end{bmatrix}\right\|_p$ subject to $\left\|\begin{bmatrix}\lambda\\ \mu\end{bmatrix}\right\|_p = 1$.
  end
  $x(1:k-1) = \lambda^* x(1:k-1)$
  $x(k) = \mu^*$
  $y = \lambda^* y + x(k)A(:,k)$
end
$\gamma = \|y\|_p$

Algorithm OSE requires a means for computing the maximizing vector in the p-norm of an $m\times 2$ matrix $W = [yA(:,k)]$. When $p = 2$ this vector can be obtained from the singular value decomposition (SVD), which can be computed explicitly in $O(m)$ flops (see the code SIGMAN of [31]). Since we can compute the gradient of $F(x) = \|Wx\|_p/\|x\|_p$ using (2.2), one possibility for $1 < p < \infty$ is to maximize f using an optimization method, such as a quasi-Newton method or the conjugate gradient method. Indeed, we could estimate $\|A\|_p$ directly in this way. However, our experience with optimization techniques is that while they work well for $p \in (1.1, 11)$ (say), they exhibit slow convergence when p is outside this range, because of the loss of smoothness of F as p approaches 1 or ∞ (cf. the comments in [22] concerning the l_p solution of an overdetermined linear system). We desire a method that works well for all values of p. A simple sampling procedure has been found to be suitable. We approximate the maximizing vector for $F(x) = \|Wx\|_p/\|x\|_p$ by the vector of the form $[\cos(\theta), \sin(\theta)]^T$ that maximizes $F(x)$ over the equally spaced sample points

$$\theta_i = \frac{i\pi}{r}, \quad i = 0:r\,.$$

To include the vectors $x = [1, 0]$, $[1, 1]/\sqrt{2}$, $[-1, 1]/\sqrt{2}$ and $[0, 1]$ we take $r = 4k$; this guarantees that our discrete approximation attains the value $\|W\|_p$ when $p = 1$ or ∞. The computational cost of Algorithm OSE with this approximate inner maximization is about $4rmn$ flops.

It is easy to see that Algorithm OSE produces an exact estimate for all p when A is diagonal (assuming that $r = 2k \geqq 2$ if sampling is used). Moreover, for any A the estimate γ satisfies

$$\|A\|_p \geqq \gamma \geqq \max_j \|A(:,j)\|_p \geqq \frac{\|A\|}{n^{1-1/p}}, \tag{3.1}$$

where the last inequality is (1.4).

Note that if we minimize instead of maximize in Algorithm OSE we obtain an estimate of $\min_{x \neq 0} \|Ax\|_p / \|x\|_p$, and hence of $\|A^{-1}\|_p$ when A is square. If the estimates obtained via minimizing were good we would have a remarkably useful condition estimation technique. Unfortunately, numerical tests reveal these estimates to be poor, so we do not pursue this idea.

4 Iteratively re-weighted 2-norms

The third method we consider is motivated by the observation that a p-norm is related to a weighted 2-norm:

$$\|x\|_p^p = \sum_{i=1}^{n} |x_i|^p = \sum_{i=1}^{n} |w_i x_i|^2, \qquad w_i = |x_i^{p/2}|/|x_i| .$$

Hence we can write

$$F(x) = \frac{\|Ax\|_p^p}{\|x\|_p^p} = \frac{\|D_1 Ax\|_2^2}{\|D_2 x\|_2^2} = \frac{\|D_1 A D_2^{-1} y\|_2^2}{\|y\|_2^2} = G(y) ,$$

where D_1 and D_2 are diagonal matrices and $y = D_2 x$. The idea is to find the vector y^* that maximizes $G(y)$, then transform to an estimate of $\max_x F(x)$ via $x = D_2^{-1} y$. This x determines new weights that define D_1 and D_2, and the process repeats iteratively.

Algorithm IRW. Given $A \in \mathbb{R}^{m \times n}$, $x_0 \in \mathbb{R}^n$ and a convergence tolerance tol, this algorithm computes γ and x such that $\gamma \leqq \|A\|_p$ and $\|Ax\|_p = \gamma \|x\|_p$.

$\gamma_0 = \|Ax_0\|_p / \|x_0\|_p$
for $k = 1, 2, \ldots$
  $D_1 = \operatorname{diag}(|Ax|)^{(p-2)/2}$
  $D_2 = \operatorname{diag}(|x|)^{(p-2)/2}$
  Compute v such that $\dfrac{\|D_1 A D_2^{-1} v\|_2}{\|v\|_2} = \|D_1 A D_2^{-1}\|_2$.
  $x = D_2^{-1} v$
  $\gamma_k = \|Ax\|_p / \|x\|_p$
  if $|\gamma_k - \gamma_{k-1}|/\gamma_k \leqq$ tol, $\gamma = \gamma_k$, quit, end
end

The inner 2-norm computation can be done via the power method (Algorithm PM); the power method requires $O(mn)$ flops and need not be iterated to high accuracy until convergence is approached. For the iteration to be defined it is necessary that x has no zero components, and that Ax has no zero components if $p < 2$; in practice, zero elements can be perturbed to a tolerance times the ∞-norm of the vector (the tolerance is the unit roundoff in our experiments).

Algorithm IRW is in the same spirit as iteratively re-weighted least squares methods. By analogy with the convergence theory for these methods [4, 25, Sect. 5.4] we expect, at best, linear convergence to a local maximum of $F(x)$.

5 Numerical experiments

In this section we investigate the performance of the three p-norm estimators. Our chief concern is the quality of the estimate γ, as measured by the underestimation ratio $\mu = \gamma / \|A\|_p \leq 1$. Unlike in condition estimation, where an estimate of the correct order of magnitude is usually acceptable, we ask for correct digits in our estimate of $\|A\|_p$. (Recall that a priori estimates correct to within a factor depending only on n are available from (1.4), (1.5), (1.8), (1.9), (1.11) and (1.12).) Thus an estimate with $\mu > 0.95$ is desired.

The tests we report are organised as follows. We choose a matrix A and estimate $\|A\|_p$ for the 21 equally spaced values $p = 1, 1.05, 1.10, \ldots, 2$ (we can restrict to the interval [1, 2] in view of the duality result (1.13)). The exact value of $\|A\|_p$ is computed directly for $p = 1, 2$, and for $1 < p < 2$ it is approximated by running Algorithm PM with x_0 equal to the maximizing vector for the previous value of p (and with tol in (2.4) equal to the unit roundoff); with these good starting vectors we expect convergence to $\|A\|_p$. For added safety we also estimate $\|A^T\|_q$ using Algorithm PM with $x_0 = (1, 1, \ldots, 1)^T$ (where $p^{-1} + q^{-1} = 1$) and use this value if it exceeds the previous $\|A\|_p$ approximation (it did in only one case in the tests reported here). For each value of p we estimate $\|A\|_p$ by the following algorithms.

(1) Algorithm PM with $x_0 = (1, 1, \ldots, 1)^T$, and with the convergence test (2.4) with tol $= 10^{-4}$.

(2) Algorithm OSE. The inner maximization is done using sampling with $r = 8$ if $p \neq 2$ and via the SVD if $p = 2$.

(3) Algorithm IRW with tol $= 10^{-4}$. Because the algorithm does not always converge we impose a limit of 50 iterations and take as the final estimate the largest estimate generated over all the iterations.

(4) Algorithm Pnorm: this is a hybrid algorithm that uses the approximate maximizing vector from Algorithm OSE as the starting vector for Algorithm PM; the details are otherwise as in (1) and (2).

We have run many examples, three of which we report here. For each estimator we give the minimum, maximum and average of both the underestimation ratio and the number of iterations for convergence. For Algorithm IRW if the maximum number of iterations is 50 the number of occurrences is indicated in parentheses. All the tests were performed in MATLAB [24], which has unit roundoff $u \approx 1.1 \times 10^{-16}$. Plots of $\|A\|_p$ for $1 \leq p \leq 2$ are given for the three matrices in Fig. 1. The fourth plot in Fig. 1 is based on the data from the first three plots and shows $1/p$ versus $\log \|A\|_p$; this displays clearly the fact that $\log \|A\|_p$ is a convex function of $1/p$.

The first matrix is a 12×12 Hadamard matrix H_{12}. $H \in \mathbb{R}^{n \times n}$ is a Hadamard matrix if $|h_{ij}| \equiv 1$ and $HH^T = nI$. The following lemma evaluates $\|H\|_p$.

Lemma 5.1. *If $H \in \mathbb{R}^{n \times n}$ is a Hadamard matrix then $\|H\|_p = \max(n^{1/p}, n^{1-1/p})$.*

Proof. H^T is also a Hadamard matrix, so by the duality result (1.13) it suffices to show that $\|H\|_p = n^{1/p}$ for $1 \leq p \leq 2$. Since $|h_{ij}| \equiv 1$, (1.4) gives $\|H\|_p \geq n^{1/p}$. Since $\|H\|_1 = n$ and $\|H\|_2 = n^{1/2}$, (1.12) gives $\|H\|_p \leq n^{1/p}$, and so $\|H\|_p = n^{1/p}$ for $1 \leq p \leq 2$, as required □

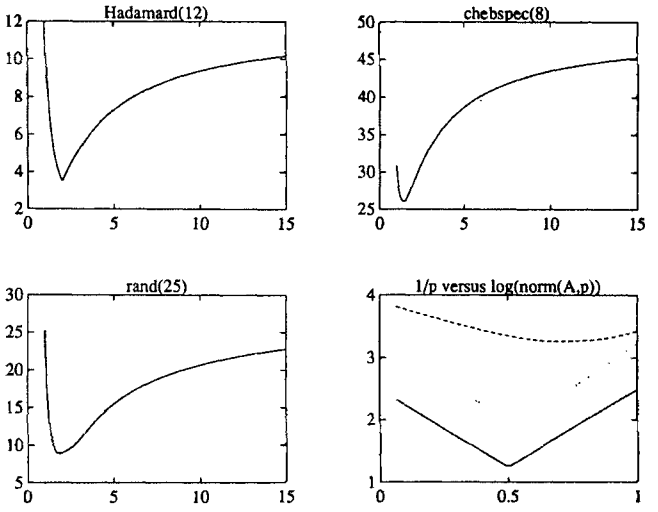

Fig. 1. Plots of $\| A \|_p$

Table 1. Hadamard(12)

	Underestimation ratio			Iterations		
	min	max	ave	min	max	ave
Alg. PM	0.1056	1.000	0.5190	2	2	2
Alg. OSE	1.000	1.000	1.000	—	—	—
Alg. IRW	0.4638	1.000	0.7919	1	50 (6)	23.286
Alg. Pnorm	1.000	1.000	1.000	2	2	2

Table 2. chebspec(8)

	Underestimation ratio			Iterations		
	min	max	ave	min	max	ave
Alg. PM	0.9597	1.000	0.9961	2	21	11.10
Alg. OSE	0.9431	1.000	0.9805	—	—	—
Alg. IRW	0.9340	1.000	0.9838	2	50 (5)	18.810
Alg. Pnorm	0.9972	0.9996	1.000	2	31	12.33

The results in Table 1 show that Algorithm PM and Algorithm IRW return many unacceptably poor estimates for H_{12}, although convergence is quick for Algorithm PM. The estimates from Algorithms OSE and Pnorm are exact, as (3.1) shows they must be.

The second matrix is an 8×8 nilpotent Chebyshev spectral differentiation matrix (*chebspec*(8) from [19]); the results are in Table 2. The number of iterations required by Algorithm PM is relatively large. This matrix produces some relatively

Table 3. rand(25)

	Underestimation ratio			Iterations		
	min	max	ave	min	max	ave
Alg. PM	0.7293	1.000	0.9328	2	28	14.81
Alg. OSE	0.8525	1.000	0.9508	—	—	—
Alg. IRW	0.6310	1.000	0.8969	2	50 (11)	31.19
Alg. Pnorm	0.9999	1.000	1.000	2	16	7.238

poor estimates from Algorithm OSE, but even the worst estimate provides a correct digit.

The third example, reported in Table 3, is a 25×25 random matrix with elements from the normal (0, 1) distribution. For this matrix all three of the basic algorithms produce some poor estimates, but Algorithm Pnorm yields excellent accuracy.

We repeated the same examples with tol $= u$. For the Hadamard matrix only the estimates from Algorithm IRW were improved. In the second example all but one of the estimates from Algorithm PM and Algorithm Pnorm became exact. In the third example the estimates from Algorithm PM were mostly unchanged, but those from Algorithm Pnorm all became exact. Many more iterations can be required by the algorithms for tol $= u$ (the largest for Algorithm Pnorm was 274 iterations in the second example).

We make several more observations based on our experience with the methods.

(1) In Algorithm PM the relative change $(\gamma_k - \gamma_{k-1})/\gamma_k$ in the estimate γ_k tends to decrease monotonically with k, although occasionally there are slight increases for a few iterations. The convergence test (2.4) is therefore a reliable way of terminating the iteration.

(2) In Algorithm IRW the estimates γ_k often increase monotonically. However, they can also vary erratically, showing no sign of convergence. Algorithm IRW is clearly not competitive with Algorithm OSE in accuracy or cost.

(3) The estimates produced by Algorithm Pnorm are excellent. In almost all cases we have tried with tol $= 10^{-4}$ they have at least two correct significant digits, increasing to full accuracy if the tolerance is the unit roundoff. One way to investigate worst-case behaviour of a condition estimator or norm estimator is to apply direct search minimization, with the elements of the matrix A as the variables. It is shown in [18] that this approach readily reveals poor estimates for the LINPACK estimator and the estimator of [16]. We have used the direct search routines from [18] with Algorithms OSE and Pnorm, with $r = 8$, $m = n \leqq 4$, and tol $= 10^{-4}$. For Algorithm OSE the worst underestimation ratio we have produced is $\mu_{\min} = 0.7456$ ($p = 1.9$), and for Algorithm Pnorm $\mu_{\min} = 0.8017$ ($p = 1.5$). Interestingly, increasing r or decreasing tol has little effect on the quality of the estimates in these two examples found by direct search (note from (3.1) that for $n = 4$ and $p \leqq 2$, $\mu \geqq 4^{-1/2} = 0.5$.) Concerning computational cost it is interesting to note from Table 2 that Algorithm Pnorm can require more iterations than Algorithm PM; it can also produce smaller estimates than Algorithm PM, but in our experience this happens only when both estimates are very accurate.

(4) The quality of the estimate from Algorithm OSE often improves when the number of samples r is increased, particularly when p is close to 2. However,

increasing r usually cannot make an inexact estimate exact. The value $r = 8$ seems entirely adequate for use in Algorithm Pnorm.

6 Conclusions

The p-norm of an $m \times n$ matrix can be reliably estimated in $O(mn)$ operations using Algorithm Pnorm, which uses Algorithm OSE to generate a good starting vector for the power method (Algorithm PM). The algorithm is guaranteed to produce $\|A\|_p$ exactly when A is diagonal or rank 1, or when A has nonnegative elements and A^TA is irreducible (Algorithm OSE itself is not always exact in latter case, or when A has rank 1). The estimate is always within a factor $n^{1-1/p}$ of $\|A\|_p$ (see (3.1)). The cost and accuracy of Algorithm Pnorm depend on the convergence tolerance tol in (2.4). If tol $= 10^{-4}$, as in our experiments, then correct significant digits are almost always obtained and the operation count can be estimated very roughly as 70 mn flops (the number of iterations in Algorithm PM varies greatly with the matrix A). Greater accuracy can be obtained by taking a smaller tolerance in Algorithm PM.

A less expensive estimate can be obtained from Algorithm OSE alone, but this estimate is more likely not to have any correct significant digits. In the case of the 2-norm Algorithm OSE is particularly efficient because each inner maximization can be done exactly in $O(m)$ flops via the SVD. Algorithm OSE then resembles the $\|T\|_2$ estimator of [5] for triangular matrices T, but it has the advantages of working with an arbitrary rectangular matrix instead of a square triangular matrix and in practice it gives sharper estimates than the estimator of [5]. (Note, however, that the method of [5] also yields very good estimates for the more elusive quantity $\|T^{-1}\|_2$, given only T.)

In summary, we recommend Algorithm Pnorm for estimating $\|A\|_p$. A MATLAB M-file *pnorm.m* implementing the algorithm is listed in the Appendix. It has the same functionality as MATLAB's built-in function *norm* but it works for any $1 \leqq p \leqq \infty$ for both matrices and vectors.

Appendix: Listings of MATLAB routines

These routines are available by anonymous FTP from the machine at Internet address 130.88.16.10, in directory pub/higham.

PNORM.M

```
function    [est, x, k] = pnorm(A, p, tol, noprint)
% PNORM     [EST, x, k] = PNORM(A, p, TOL) estimates the p-norm of a
%           matrix A, using the p-norm power method with a specially chosen
%           starting vector.
%           TOL is a relative convergence tolerance (default 1E-4).
%           Returned are the norm estimate EST, the corresponding
%           approximate maximizing vector x, and the number of power method
%           iterations k.
```

```
%           A nonzero fifth argument suppresses output to the screen.
%           If A is a vector, this routine simply returns NORM(A, p).
%
%           Note: The estimate is exact for p = 1, but is not always exact for
%           p = 2 or p = inf. Code could be added to treat p = 2 and p = inf
%           separately.
%
%           Calls DUAL, and SEQA from [19].
%
%           Reference
%           N.J. Higham, Estimating the matrix p-norm, Numerical Analysis;
%           Report No. 202, University of Manchester, October 1991;
%           to appear in Numer. Math.

[m, n] = size(A);
If min(m, n) = = 1, est = norm(A, p); return, end

if nargin < 4, noprint = 0; end
if nargin < 3, tol = 1e-4; end

% Stage I. Use Algorithm OSE to get starting vector x for power method.
% Form y = B * x, at each stage choosing x(k) = c and scaling previous
% x(k + 1:n) by s, where norm ([c s], p) = 1.

sm = 9;  % Number of samples.
y = zeros (m, 1); x = zeros (n, 1);

for k = 1:n
   if k = = 1
     c = 1; s = 0;
   else
     W = [A(:, k) y];
     if p = = 2  % Special case. Solve exactly for 2-norm.
       [U, S, V] = svd(W);
       % For Matlab 4.0 replace the previous line by the next one
       % so that this routine works for sparse input A.
       % [U, S, V] = svd (full (W));
       c = V(1, 1); s = V(2, 1);
   else

       fopt = 0;
       for th = seqa(0, pi, sm)
          c1 = cos(th); s1 = sin(th);
          nrm = norm ([c1 s1], p);
          c1 = c1/nrm; s1 = s1/nrm;   % [c1 s1] has unit p-norm.
          f = norm(W*[c1 s1]', p);
          if f > fopt
            fopt = f;
            c = c1; s = s1;
          end
       end

   end
end
```

```
x(k) = c;
y = x(k)*A(:,k) + s*y;
if k > 1, x(1 : k - 1) = s*x(1 : k - 1); end

end

est = norm(y, p);
if ~ noprint, fprintf('Alg OSE: % 9.4e\n', est), end

% Stage II.  Apply Algorithm PM (the power method).

q = dual(p);
k = 1;

while 1

      y = A*x;
      est_old = est;
      est = norm(y, p);

      z = A'*dual(y, p);

      if ~noprint
           fprintf('%2.0 f: norm(y) = %9.4e, norm(z) = %9.4e', ...
                     k, norm(y, p), norm(z, q))
           fprintf('   rel_incr(est) = %9.4e\n', (est-est_old)/est)
      end

      if (norm(z, q)<=z'*x| abs(est-est_old)/est<=tol) & k > 1
        return
      end

      x = dual(z, q);
      k = k + 1;

end
```

DUAL.M

```
function   y = dual(x,p)
%DUAL  Y = DUAL(X,p) is a vector of unit q-norm that is dual to X with
%      respect to the p-norm, that is, norm(Y,q) = 1 and there is
%      equality in the Holder inequality: X'*Y = norm(X,p) norm(Y,q).
%      Special case: DUAL(X), where X>=1 is a scalar, returns Y such
%                    that 1/X + 1/Y = 1.

if max(size(x))==1 & nargin==1
  p = x;
end

% The following test avoids a 'division by zero message' when p = 1.
if p==1
  q = inf;
else
  q = 1/(1 - 1/p);
end
```

```
if max(size(x))==1 & nargin==1
   y = q;
   return
end

if norm(x, inf)==0, y = x; return, end

if p==1

   y = sign(x) + (x==0);   %y(i) = +1 or -1 (if x(i) real).

elseif p==inf

   [xmax, k] = max(abs(x));
   f = find(abs(x)==xmax); k = f(1);
   y = zeros(x);
   y(k) = sign(x(k));      %y is a multiple of unit vector e_k.

else %1 < p < inf. Dual is unique in this case.
   x = x/norm(x, inf);     % This scaling helps to avoid under/over-flow.
   y = abs(x).^(p - 1).*(sign(x) + (x==0));
   y = y/norm(y, q);       % Normalize to unit q-norm.

end
```

Acknowledgements. I thank Des Higham for suggesting improvements to the manuscript.

References

1. Anderson, E., Bai, Z., Bischof, C.H., Demmel, J.W., Dongarra, J., Du Croz, J.J., Greenbaum, A., Hammarling, S.J., McKenney, A., Ostrouchov, S., Sorensen, D.C. (1992): LAPACK Users' Guide, Society for Industrial and Applied Mathematics, Philadelphia (to appear)
2. Bartels, S.G. (1991): Two topics in matrix analysis: Structured sensitivity for Vandermonde-like systems and a subgradient method for matrix norm estimation. M.Sc. Thesis, University of Dundee, Scotland
3. Boyd, D.W. (1974): The power method for l_p norms. Linear Algebra and Appl. **9**, 95–101
4. Cline, A.K. (1972): Rate of convergence of Lawson's algorithm. Math. Comput. **26**, 167–176
5. Cline, A.K., Conn, A.R., Van Loan, C.F. (1982): Generalizing the LINPACK condition estimator. In: J.P. Hennart, ed., Numerical Analysis, Mexico 1981. Lecture Notes in Mathematics 909. Springer, Berlin Heidelberg New York, pp.73–83
6. Fletcher, R. (1987): Practical Methods of Optimization, 2nd Ed. Wiley, Chichester
7. Fletcher, R., Grant, J.A., Hebden, M.D. (1971): The calculation of linear best L_p approximations, Comput. J. **14**, 276–279
8. Gastinel, N. (1970): Linear Numerical Analysis. Academic Press, London
9. Gilbert, J.R., Moler, C.B., Schreiber, R.S. (1992): Sparse matrices in MATLAB: Design and implementation. SIAM J. Matrix Anal. Appl. **13**, 333–356
10. Gillespie, T.A. (1991): Noncommutative variations on theorems of Marcel Riesz and others, In: J.H. Ewing, F.W. Gehring, eds., Paul Halmos: Celebrating 50 Years of Mathematics. Springer, Berlin Heidelberg New York
11. Goldberg, M., Straus, E.G. (1983): Multiplicativity of l_p norms for matrices. Linear Algebra Appl. **52/53**, 351–360
12. Golub G.H., Van Loan, C.F. (1989): Matrix Computations, 2nd Ed. Johns Hopkins University Press, Baltimore, Maryland
13. Hager, W.W. (1984): Condition estimates. SIAM J. Sci. Stat. Comput. **5**, 311–316
14. Hardy, G.H., Littlewood, J.E., Pólya, G. (1952): Inequalities, 2nd Ed. Cambridge University Press, Cambridge

15. Higham, N.J. (1987): A survey of condition number estimation for triangular matrices. SIAM Rev. **29**, 575–596
16. Higham, N.J. (1988): FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation (Algorithm 674). ACM Trans. Math. Soft. **14**, 381–396
17. Higham, N.J. (1990): Experience with a matrix norm estimator. SIAM J. Sci. Stat. Comput. **11**, 804–809
18. Higham, N.J. (1993): Optimization by direct search in matrix computations, Numerical Analysis Report No. 197, University of Manchester, England, January 1991; SIAM J. Matrix Anal. Appl. (to appear)
19. Higham, N.J. (1991): Algorithm 694: A collection of test matrices in Matlab, ACM Trans. Math. Soft. **17**, 289–305
20. Horn R.A., Johnson, C.R. (1985): Matrix Analysis. Cambridge University Press, Cambridge
21. Kato, T. (1976): Perturbation Theory for Linear Operators, 2nd Ed. Springer, Berlin Heidelberg New York
22. Li, Y. (1991): A globally convergent method for L_p problems. Technical Report 91-1212. Department of Computer Science, Cornell University, Ithaca, NY
23. Merle, G., Späth, H. (1974): Computational experiences with discrete L_p-approximation. Computing **12**, 315–321
24. Moler, C.B., Little, J.N., Bangert, S. (1989): 386-Matlab User's Guide. The Math Works, Inc., 24 Prime Parkway, Natick, MA
25. Osborne, M.R. (1985): Finite Algorithms in Optimization and Data Analysis. Wiley, Chichester
26. Osborne, M.R., Watson, G.A. (1985): An analysis of the total approximation problem in separable norms, and an algorithm for the total l_1 problem. SIAM J. Sci. Stat. Comput. **6**, 410–424
27. Späth H., Watson, G.A. (1987): On orthogonal linear l_1 approximation. Numer. Math. **51**, 531–543
28. Schneider, H., Strang, W.G. (1962): Comparison theorems for supremum norms. Numer. Math. **4**, 15–20
29. Pham Dinh Tao, (1984): Convergence of a subgradient method for computing the bound norm of matrices [in French]. Linear Algebra Appl. **62**, 163–182
30. Todd, J. (1977): Basic Numerical Mathematics, Vol. 2. Numerical Algebra. Birkhäuser, Basel; Academic Press, New York
31. Van Loan, C.F. (1987): On estimating the condition of eigenvalues and eigenvectors. Linear Algebra Appl. **88/89**, 715–732
32. Watson, G.A. (1980): Approximation Theory and Numerical Methods. Wiley, Chichester
33. Watson, G.A. (1983): The total approximation problem. In C.K. Chui, L.L. Schumaker, J.D. Ward, eds., Approximation Theory IV. Academic Press, New York, pp. 723–728