On Estimating the Condition of Eigenvalues and Eigenvectors

Charles Van Loan*
Department of Computer Science
Cornell University
Ithaca, New York 14853

In memory of James H. Wilkinson

Submitted by Jack Dongarra

ABSTRACT

A method is developed for estimating the accuracy of computed eigenvalues and eigenvectors that are obtained via certain EISPACK subroutines. It does this at a cost of $O(n^2)$ flops per eigenpair, assuming that the eigenpair is known and assuming that the original matrix has been reduced to Hessenberg form. The heart of the technique involves estimating the smallest singular value of a certain nearly triangular submatrix. This is accomplished by some standard "zero chasing" with Givens transformations and with a 2-norm version of the LINPACK condition estimator. An EISPACK-compatible code has been developed, and its performance is discussed.

1. INTRODUCTION

Condition numbers play a valuable role in matrix computations insofar as they enable us to estimate the accuracy of computed results. For example, suppose that the linear system $Ax = b$ is solved via Gaussian elimination with partial pivoting. If x is the computed result, then in general we have

$$\frac{\|x - x\|_p}{\|x\|_p} \approx \text{macheps} \cdot \kappa_p(A),$$

where macheps is the machine precision and $\kappa_p(A) = \|A\|_p \|A^{-1}\|_p$ is the

* The research presented in this paper was supported by ONR Contract N00014-83-K-0640.

52 Vanderbilt Ave., New York, NY 10017 0024-3795/87/$3.50

p-norm condition of A with respect to inversion. An efficient $O(n^2)$ method for estimating $\kappa_1(A)$ is included in LINPACK. The method assumes that the matrix A has already been factored [7].

In this paper we propose an eigenvalue-eigenvector condition estimator that can be used in conjunction with the package EISPACK. We first review the necessary mathematics. Suppose λ is a distinct eigenvalue of $A \in C^{n\times n}$ and that

$$Ax = \lambda x, \qquad x \in \mathbf{C}^n, \quad \|x\|_2 = 1. \tag{1.1}$$

If $Q = [x, Q_1]$ is unitary, $Q_1 \in \mathbf{C}^{n\times(n-1)}$, then it can be shown that

$$Q^H A Q = \begin{bmatrix} \lambda & w^H \\ 0 & B \end{bmatrix}, \tag{1.2}$$

where $w \in \mathbf{C}^{n-1}$ and $B \in \mathbf{C}^{(n-1)\times(n-1)}$. It follows that if $z \in \mathbf{C}^{n-1}$ satisfies

$$(B - \lambda I)^H z = -w,$$

and

$$y = \frac{1}{\sqrt{1 + z^H z}} Q \begin{bmatrix} 1 \\ z \end{bmatrix},$$

then

$$y^H A = \lambda y^H, \qquad y \in \mathbf{C}^n, \quad \|y\|_2 = 1. \tag{1.3}$$

It turns out that the sensitivity of the eigenvalue λ depends upon the angle between the left eigenvector y and the right eigenvector x. In the worst case if $E = \varepsilon y x^H$ and ε is "small enough," then λ will be perturbed to an eigenvalue $\hat{\lambda}$ of $A + E$ that satisfies

$$|\hat{\lambda} - \lambda| = \frac{\varepsilon}{s(\lambda)} + O(\varepsilon^2), \tag{1.4}$$

where

$$s(\lambda) = \frac{|y^H x|}{\|y\|_2 \|x\|_2} = \frac{1}{\sqrt{1 + z^H z}}. \tag{1.5}$$

Clearly, the reciprocal of $s(\lambda)$ can be regarded as the condition of the eigenvalue λ; it measures the sensitivity of λ to perturbation. See [36] or [15] for more details. It should be stressed that not all $O(\varepsilon)$ perturbations of A induce $O(\varepsilon/s(\lambda))$ changes in λ. However, "random" perturbations of A due to roundoff almost always induce the worst case change.

If λ is a multiple eigenvalue, then its sensitivity properties are more complicated. To begin with, if A is normal ($A^HA = AA^H$), then λ is perfectly conditioned in that (1.4) holds with $s(\lambda) = 1$ regardless of multiplicity. For this reason, there is no need for us to consider further eigenvalue sensitivity questions pertaining to normal matrices.

In the nonnormal case, it is possible for $O(\varepsilon)$ perturbations in A to induce $O(\varepsilon^{1/p})$ changes in an eigenvalue associated with a p-dimensional Jordan block. Unfortunately, deducing Jordan block structure is difficult in practice [19]. However, in principle the $s(\lambda)$ can be used to shed light on A's nearness to a matrix with multiple eigenvalues. For example, Wilkinson [37] shows that if $s(\lambda) < 1$, then there exists a matrix E satisfying

$$\frac{\|E\|_2}{\|A\|_2} \leqslant \frac{s(\lambda)}{\sqrt{1 - s(\lambda)^2}}$$

for which $A + E$ has a multiple eigenvalue. Additional results of this flavor may be found in [38], the definitive work on eigenvalue sensitivity.

Another way to quantify the isolation of the eigenvalue λ involves the matrix B in (1.2). We define the *separation* of the eigenvalue λ by

$$sep(\lambda) = \sigma_{\min}(B - \lambda I), \tag{1.6}$$

where $\sigma_{\min}(\cdot)$ denotes the minimum singular value. Although the unitary matrix Q in (1.2) is not unique, it is easy to show using orthogonality that $sep(\lambda)$ is well defined. Using widely known properties of singular values (see [15] for example), it follows that there is a matrix F such that $\|F\|_2 = sep(\lambda)$ and $B - \lambda I + F$ is singular. Thus,

$$A + Q\begin{bmatrix} 0 & 0 \\ 0 & F \end{bmatrix}Q^H = Q\begin{bmatrix} \lambda & w^H \\ 0 & B + F \end{bmatrix}Q^H$$

has λ as a repeated eigenvalue. The function *sep* and its properties are surveyed in [34].

To assess the sensitivity of the eigenvector x, or more correctly, the sensitivity of the invariant subspace $span\{x\}$, we need the concept of

distance between subspaces. If S_1 and S_2 are two subspaces in $\mathbf{C}^n$ of equal dimension, then

$$dist(S_1, S_2) = \|P_1 - P_2\|_2,$$

where P_i is the orthogonal projection onto S_i. Using this measure of distance, it can be shown that in the "worst case" there exists an $E \in C^{n\times n}$ that satisfies $\|E\|_2 = \varepsilon$ so that λ's eigenspace $span\{x\}$ is perturbed by an amount $O(\varepsilon/sep(\lambda))$. In other words, the eigenpair $\{x, \lambda\}$ is perturbed to $\{\hat{x}, \hat{\lambda}\}$ with

$$dist(span\{x\}, span\{\hat{x}\}) \approx \frac{\varepsilon}{sep(\lambda)}. \tag{1.7}$$

Perturbation results of this flavor are detailed in [26, 27].

Equations (1.4) and (1.7) suggest that to estimate the sensitivity of an eigenpair (λ, x) we need to estimate both $s(\lambda)$ and $sep(\lambda)$. The thrust of this paper is to show how this can be accomplished in $O(n^2)$ flops for each eigenpair of interest, assuming that A is in Hessenberg form, i.e., $a_{ij} = 0$ whenever $i > j+1$. Our approach begins by determining a Householder matrix Q such that

$$Q^T A Q = \begin{bmatrix} \lambda & w^H \\ 0 & B \end{bmatrix}.$$

The QR factorization $Q_\lambda R_\lambda = B - \lambda I$ is then computed and $sep(\lambda) = \sigma_{\min}(B - \lambda I) = \sigma_{\min}(R_\lambda)$ is estimated using a 2-norm condition estimator.

The paper is organized as follows. In Section 2 we show how the QR decomposition of $B - \lambda I$ can be computed in $O(n^2)$ flops. The key is to recognize that B is a rank two correction of an upper Hessenberg matrix. The smallest singular value of the resulting triangular form R_λ is then estimated in $O(n^2)$ flops using a 2-norm generalization of the LINPACK condition estimator. This procedure is outlined in Section 3. The implementation of our method is then discussed in Section 4 along with numerical test results.

We conclude the introduction by mentioning related work. Several authors have worked on eigenproblem sensitivity estimation. Symm and Wilkinson [31] and Dongarra, Moler, and Wilkinson [10] have formulated and analyzed an iterative improvement scheme for an approximate eigenpair $(\hat{\lambda}, \hat{x})$ when $\hat{\lambda}$ corresponds to a distinct eigenvalue λ. Their technique improves the accu-

racy of the computed eigenpair and also returns error estimates. It works by iterating with a matrix that is obtained by replacing one of A's columns with the current approximate eigenvector $\hat{x}$. Unfortunately, if this matrix is ill conditioned, then the convergence is impeded, and this can happen even if λ is well conditioned. A more serious drawback is that the double precision computation of residuals $A\hat{x} - \hat{\lambda}\hat{x}$ is required, something that complicates the portability of software.

A perspective on this last statement can be obtained by considering the corresponding situation in the linear equation problem $Ax = b$. Iterative improvement in this context also means the double precision computation of residuals. As Forsythe and Moler [12] show, it is possible to get a heuristic estimate of A's condition number by performing a single step of iterative improvement. However, because multiple precision arithmetic complicates the portability of a program, the designers of LINPACK opted for a method of condition estimation that is reasonably machine-independent. We argue that a similar philosophy should apply to an "EISPACK condition estimator."

A step in this direction is the eigenvalue condition estimator of Chan, Feldman, and Parlett [5]. They propose to compute $s(\lambda)$ from its definition (1.5). The required right and left eigenvectors are found by back substitution after the eigenvalues are computed via ORTHES and HQR. Their method is attractive because no additional storage is required. However, it does not provide any information about *sep*.

Ruhe [22] suggests using the Golub-Reinsch SVD algorithm [16] to calculate separations, but this requires $O(n^3)$ flops per eigenpair. Thus, if condition estimates of all eigenpairs are desired, then $O(n^4)$ flops may be required.

Early work by Varah [32, 33] is concerned with the rigorous bounding of errors in a computed eigensystem. Although the bounds are rigorous and computable, they are somewhat complicated, require some mixed precision computation, and do not directly supply information about the separations.

Another approach to estimating errors in eigenvalues is taken by Yakamoto [39] and is based on some well-known theorems about nonlinear equations. Results based on the condition of the eigenvector matrix are discussed in [24]. Golub and Wilkinson [17] and Bavely and Stewart [2] offer comments that pertain to the condition of higher dimensional invariant subspaces.

2. THE OVERALL ALGORITHM

With appropriate references to EISPACK, here is our technique for estimating $s(\lambda)$ and $sep(\lambda)$.

ALGORITHM 2.1.

Step 1. Overwrite A with its Hessenberg form $H = U^H AU$, where U is unitary. If A is real, then the EISPACK routine ORTHES could be used. It is easy to show that both $s(\lambda)$ and $\text{sep}(\lambda)$ are preserved with this unitary transformation. Note: Hereafter in the algorithm we assume that A is in Hessenberg form.

Step 2. Compute the eigenpairs (λ_i, x_i) of interest, where $Ax_i = \lambda_i x_i$, $i = 1, \ldots, p$. This can be done using EISPACK routines. For example, in the real case, HQR can be used to get the eigenvalues and INVIT the desired eigenvectors.

Step 3. For each eigenpair (λ, x) whose condition is desired:

(a) Compute a Householder matrix $Q = I - 2uu^H/u^H u$ such that

$$Q^H AQ = \begin{bmatrix} \lambda & w^H \\ 0 & B \end{bmatrix}.$$

(b) Compute the decomposition $B - \lambda I_{n-1} = Q_\lambda R_\lambda$ where Q_λ is unitary and R_λ is upper triangular.

(c) Estimate $sep(\lambda) = \sigma_{\min}(B - \lambda I_{n-1}) = \sigma_{\min}(R_\lambda)$ using the 2-norm condition estimator in [6].

(d) Solve

$$R_\lambda^H v = -w^H$$

for v, and compute

$$s(\lambda) = \frac{1}{\sqrt{1 + v^H v}}.$$

This expression for $s(\lambda)$ follows from the fact that if $z = Q_\lambda v$ then $y^H = (1, z^H)Q^H$ is a left eigenvector. Since $x = Qe_1$ is a right eigenvector, we have

$$s(\lambda) = \frac{|y^H x|}{\|x\|_2 \|y\|_2} = \frac{1}{\sqrt{1 + z^H z}} = \frac{1}{\sqrt{1 + v^H v}}.$$

Note that the if $sep(\lambda) \approx \text{machep} \|A\|_2$ then λ can be regarded as a multiple eigenvalue and care must be exercised when solving the ill-conditioned system $R_\lambda^H v = -w^H$.

We first turn our attention to the practical computation of the decomposition (1.2) and to the estimation of $\sigma_{\min}(B-\lambda I)$. Assume that $A \in \mathbf{C}^{n\times n}$ is in Hessenberg form and that we have $Ax = \lambda x$ with $x \in C^n$ nonzero. Our plan is to find a unitary Q such that

$$Q^H A Q = \begin{bmatrix} \lambda & w^H \\ 0 & B \end{bmatrix} \tag{2.1}$$

can be formed in $O(n^2)$ flops with the added proviso that only $O(n^2)$ flops are required to compute the QR factorization of $B-\lambda I$.

The desire for $O(n^2)$ speed rules out some obvious choices for Q. For example, Businger [3] shows how Q could be determined such that B is Hessenberg using $O(n^2)$ Givens rotations. Thus, the QR factorization of $B-\lambda I$ could be found in $O(n^2)$ flops, but B itself would require $O(n^3)$ flops.

A Hessenberg B could in principle be achieved in $O(n^2)$ flops by performing a shifted QL step on A if we assume A has a nonzero subdiagonal. To be specific, suppose Givens rotations $J_{n-1},\dots,J_1$ are determined such that

$$(A-\lambda I)J_{n-1}\cdots J_1 = R$$

is upper triangular. Here, each J_k has the form

$$J_k \equiv J(k,c,s) = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & c & \bar{s} & \cdots & 0 \\ 0 & \cdots & -s & c & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \\ \\ k \\ k+1 \\ \\ \\ \end{matrix} \qquad c \in \mathbf{R}, \quad s \in \mathbf{C}.$$

$$\qquad\qquad\qquad\qquad\qquad\qquad k \quad k+1$$

where c and s satisfy $c^2 + |s|^2 = 1$. If $Q = J_{n-1}\cdots J_1$, then in exact arithmetic it can be shown that the decomposition (3.1) holds. Unfortunately, the computed (2,1) element of $Q^H A Q$ may not be negligible relative to macheps.

Our approach is to compute Q as a Householder matrix. If $x \in \mathbf{C}^n$ is nonzero and

$$Q = I - 2\frac{uu^H}{u^H u},$$

where $u = x + \exp[\arg(x_1)]\|x\|_2 e_1$ and e_1 is the first column of I, then

$$Qx = -\exp[\arg(x_1)]\,\|x\|_2 e_1.$$

Since $Q = Q^{-1} = Q^H$, it follows from the equation $Ax = \lambda x$ that

$$(Q^H A Q)e_1 = \lambda e_1.$$

Thus, λe_1 is the first column of $Q^H A Q$, and so we have (2.1). (Numerically, this is legitimate to assume so long as $\|Ax - \lambda x\|_2 \approx \text{macheps} \cdot \|A\|_2 \|x\|_2$.)

Note that B will be a full matrix. This appears to be counterproductive, since our overall plan is to compute the QR factorization of $B - \lambda I$ in $O(n^2)$ flops. However, B is a rank-two departure from Hessenberg form because

$$Q^H A Q = \left(I - 2\frac{uu^H}{u^H u}\right)^H A \left(I - 2\frac{uu^H}{u^H u}\right) = A + q_0 r_0^H + t_0 v_0^H,$$

where

$$q_0 = y - \beta(u^H y)u, \tag{2.2}$$

$$r_0 = u, \tag{2.3}$$

$$t_0 = u, \tag{2.4}$$

$$v_0 = z - \beta(u^H z)u, \tag{2.5}$$

with

$$\beta = \frac{1}{u^H u}, \qquad y = -2\beta A u, \quad \text{and} \quad z = -2\beta A^H u.$$

Thus, $B = G + qr^H + tv^H$, where G is the trailing $(n-1)$st order principal submatrix of A and q, r, t, and v are composed of the bottom $n-1$ components of q_0, r_0, t_0, and v_0 respectively. By exploiting this fact is it possible to compute the QR decomposition of $B - \lambda I$ in $O(n^2)$ flops. We'll need the following two algorithms.

ALGORITHM 2.2. Given an upper Hessenberg matrix $G \in \mathbf{C}^{m\times m}$, the following algorithm overwrites G with $R = J_{m-1}\cdots J_1 G$ where R is upper

triangular and $J_k = J(k, c_k, s_k)$ for $k = 1,2,\ldots,m-1$:

For $k = 1,\ldots,m-1$
Determine c_k and s_k $(c_k^2 + |s_k|^2 = 1)$ such that

$$\begin{bmatrix} c_k & \bar{s}_k \\ -s_k & c_k \end{bmatrix}\begin{bmatrix} g_{kk} \\ g_{k+1,k} \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

$G := J(k, c_k, s_k)G$
end k

ALGORITHM 2.3. Given $d \in \mathbf{C}^m$, the following algorithm computes $J_k = J(k, c_k, s_k)$ for $k = m-1,\ldots,1$ such that $J_1 \ldots J_{m-1}d$ is a multiple of $e_1 = (1,0,\ldots,0)^T$:

For $k = m-1,\ldots,1$
Determine c_k and s_k $(c_k^2 + |s_k|^2 = 1)$ such that

$$\begin{bmatrix} c_k & \bar{s}_k \\ -s_k & c_k \end{bmatrix}\begin{bmatrix} d_k \\ d_{k+1} \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

$d := J(k, c_k, s_k)d$
end k

With these two standard Givens routines at our disposal we can specify our method for computing the QR factorization of $B - \lambda I$.

ALGORITHM 2.4. Let G to be the trailing $(n-1)$st order principal submatrix of the upper Hessenberg matrix A, and let λ be a scalar. Let q, r, t, and v be composed of the bottom $n-1$ components of the vectors q_0, r_0, t_0, and v_0 that are defined in (2.2)–(2.5). This algorithm overwrites G with the upper triangular matrix R_λ, where $Q_\lambda R_\lambda = G - \lambda I + qr^H + w^H$ is the QR factorization:

Step 1. $G := G - \lambda I$ (an upper Hessenberg matrix).
Step 2. Apply Algorithm 2.2 to G, and apply the Givens rotations to both q and t:

$$q := J_{n-2}\cdots J_1 q, \qquad t := J_{n-2}\cdots J_1 t.$$

Step 3. Apply Algorithm 2.3 to q, and apply the Givens rotations to both G and t:

$$G := J_1\cdots J_{n-2}G, \qquad t := J_1\cdots J_{n-2}t.$$

Step 4. $G := G + qr^H$ (an upper Hessenberg matrix).
Step 5. Apply Algorithm 2.2 to G, and apply the Givens rotations to t:

$$t := J_{n-2} \cdots J_1 t.$$

Step 6. Apply Algorithm 2.3 to t, and apply the Givens rotations to G:

$$G := J_1 \cdots J_{n-2} G.$$

Step 7. $G := G + tv^H$ (an upper Hessenberg matrix).
Step 8. Apply Algorithm 2.2 to G. (At this stage G is upper triangular.)

3. ESTIMATING THE MINIMUM SINGULAR VALUE

The smallest singular value of an upper triangular matrix $R \in \mathbf{C}^{m\times m}$ can be estimated by using a technique described in [6]. A simpler derivation of that method has since been given and is presented in this section.

Suppose $R \in \mathbf{C}^{m\times m}$ is a nonsingular upper triangular matrix and that we have chosen a unit vector $d \in \mathbf{C}^m$ such that the solution to $Ry = d$ is large in norm. It follows that

$$\hat{\sigma}_{\min} \equiv \frac{1}{\|y\|_2} \geqslant \frac{1}{\|R^{-1}\|_2 \|d\|_2} = \sigma_{\min},$$

where $\sigma_{\min}$ is the smallest singular value of R. Clearly, the larger the value of $\|y\|_2$, the better $\hat{\sigma}_{\min}$ approximates $\sigma_{\min}$. Note that if $u_{\min}$ and $v_{\min}$ are left and right singular vectors associated with $\sigma_{\min}$ and if $d = u_{\min}$, then $y = v_{\min}/\sigma_{\min}$ and $\hat{\sigma}_{\min} = \sigma_{\min}$. The idea behind our "$\sigma_{\min}$ estimator" is to make d look like $u_{\min}$ by exploiting the back-substitution process that relates the solution $y = R^{-1}d$ to the right hand side d:

$p_i := 0 \ (i = 1,\ldots,m)$
For $k = m,\ldots,1$

$$y_k := \frac{d_k - p_k}{r_{kk}}$$
$$p_i := p_i + r_{ik} y_k \qquad (i = 1,\ldots,k-1)$$

end k

Usually, d is known in advance when back-substitution is applied. In our setting, however, we determine $d_m,\ldots,d_1$ dynamically and in such a way

that the y_i tend to be large. This is the approach of the LINPACK condition estimator, and we refer the reader to [7] for details. See also [8] and [21].

The dynamic determination of the d_i proceeds as follows. Assume scalars $d_{k+1},\dots,d_m$ and $y_{k+1},\dots,y_m$ are known such that

$$\begin{bmatrix} r_{k+1,k+1} & r_{k+1,k+2} & \cdots & r_{k+1,m} \\ 0 & r_{k+2,k+2} & \cdots & r_{k+2,m} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & r_{mm} \end{bmatrix} \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} d_{k+1} \\ d_{k+2} \\ \vdots \\ d_m \end{bmatrix}$$

and

$$|d_m|^2 + \cdots + |d_{k+1}|^2 = 1.$$

Assume also that the running sums

$$p_i = \sum_{j=k+1}^{m} r_{ij} y_j \qquad i = 1,\dots,k$$

are known. We proceed to the next step by considering the expanded system

$$\begin{bmatrix} r_{kk} & r_{k,k+1} & \cdots & r_{k,m} \\ 0 & r_{k+1,k+1} & \cdots & r_{k+1,m} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & r_{mm} \end{bmatrix} \begin{bmatrix} y_k^+ \\ y_{k+1}^+ \\ \vdots \\ y_m^+ \end{bmatrix} = \begin{bmatrix} d_k^+ \\ d_{k+1}^+ \\ \vdots \\ d_m^+ \end{bmatrix} \equiv \begin{bmatrix} c \\ sd_{k+1} \\ \vdots \\ sd_m \end{bmatrix},$$

where c and s satisfy $|c|^2 + |s|^2 = 1$ and are to be determined. Note that

$$|d_m^+|^2 + \cdots + |d_k^+|^2 = 1,$$

$$y_i^+ = sy_i \qquad (i = k+1,\dots,m),$$

$$y_k^+ = \frac{c - sp_k}{r_{kk}},$$

and that once we settle on c and s, the running sums $p_1,\dots,p_{k-1}$ become

$$p_i^+ = sp_i + r_{ik} y_k^+ \qquad (i = 1,\dots,k-1).$$

Summarizing all this, we obtain

$p_i := 0 \ (i = 1, \ldots, m)$
For $k = m, \ldots, 1$
  Determine c and s such that $|c|^2 + |s|^2 = 1$.
  $d_k := c$
  $d_i := sd_i \ \ (i = k+1, \ldots, m)$
  $y_i := sy_i \ \ (i = k+1, \ldots, m)$
  $y_k := (c - sp_k)/r_{kk}$
  $p_i := sp_i + r_{ik}y_k \ (i = 1, \ldots, k-1)$
end k

There are three factors to keep in mind when choosing c and s:

(1) We'd like $y_k^+ = (c - sp_k)/r_{kk}$ to be large.
(2) Since $y_i^+ = sy_i \ (i = k+1, \ldots, m)$, c should not be too large.
(3) Growth should be encouraged in $p_i^+ = sp_i + r_{ik}y_k^+ \ \ (i = 1, \ldots, k-1)$, since the size of subsequent y_i will depend on these running sums.

This suggests that c and s be chosen to maximize

$$f(c, s) = \sum_{i=1}^{k-1} |sp_i + r_{ik}y_k^+|^2 + |y_k^+|^2 + \sum_{i=k+1}^{m} |sy_i|^2.$$

This functional, which depends on k, can be simplified by defining the vectors

$$r = \begin{bmatrix} r_{1k} \\ \vdots \\ r_{k-1,k} \end{bmatrix}, \qquad p = \begin{bmatrix} p_1 \\ \vdots \\ p_{k-1} \end{bmatrix}, \qquad y = \begin{bmatrix} y_{k+1} \\ \vdots \\ y_m \end{bmatrix} \tag{3.1}$$

and using the definition $y_k^+ = (c - sp_k)/r_{kk}$. Indeed, it can be shown that

$$f(c, s) = \frac{1}{|r_{kk}|^2} \left\| \begin{bmatrix} 0 & r_{kk}y \\ 1 & -p_k \\ r & r_{kk}p - p_k r \end{bmatrix} \begin{bmatrix} c \\ s \end{bmatrix} \right\|_2^2.$$

Thus, the maximizing c and s that we are trying to compute define the right

singular vector associated with the largest singular value of the m-by-2 matrix

$$W_k = \begin{bmatrix} 0 & r_{kk}y \\ 1 & -p_k \\ r & r_{kk}p - p_k r \end{bmatrix}. \tag{3.2}$$

This results in the following algorithm.

ALGORITHM 3.1. Given a nonsingular upper triangular $R \in \mathbf{C}^{m\times m}$, this algorithm computes $\hat{\sigma}_{\min}$, an approximation to the smallest singular value of R:

$p_i := 0\ (i = 1,\ldots,m)$
For $k = m,\ldots,1$
  If $k = m$
    then
      $c := 1;\ s := 0$
    else
      Compute the SVD of the matrix W_k defined by (3.1)–(3.2) and let $[\bar{c}, \bar{s}]^H$ be the right singular vector associated with its largest singular value.
    endif
  $d_k := c$
  $y_k := (c - sp_k)/r_{kk}$
  $d_i := sd_i\ (i = k+1,\ldots,m)$
  $y_i := sy_i\ (i = k+1,\ldots,m)$
  $p_i := sp_i + r_{ik}y_k\ (i = 1,\ldots,k-1)$
end k
$\hat{\sigma}_{\min} = 1/\|y\|_2$

The implementation of this procedure and its behavior in practice are described in the next section. Note that it involves $O(n^2)$ flops.

4. IMPLEMENTATION DETAILS AND EXAMPLES

Real versions of Algorithms 2.4 and 3.1 have been implemented in FORTRAN. Let R be an upper triangular matrix stored in an array having row dimension rdim. The subroutine

SIGMAN(R, *rdim*, *istart*, *istop*, *sigma*, u, v)

computes the triple $(\sigma_{\min}, u_{\min}, v_{\min})$ associated with the submatrix $R(\text{istart:istop}, \text{istart:istop})$. The numerical properties of this routine are documented in [6]. Of interest to us in the current application is the fact that SIGMAN always returns an estimate of $\sigma_{\min}$ that is correct to within an order of magnitude. Indeed, on all but extremely well-conditioned examples the computed $\sigma_{\min}$ is usually correct to several significant digits. One should bear in mind that a condition estimator such as SIGMAN is really just an algorithm for obtaining a good starting vector for inverse iteration as applied to A^TA. Depending upon how crucial it is to get an accurate $\sigma_{\min}$, one can always follow SIGMAN with a few inverse iteration steps.

The subroutine

$$\text{CONDEV}(H, hdim, n, lambda, x, s, sep, A, adim, work)$$

computes $s(\lambda)$ and $sep(\lambda)$, where H is a real upper Hessenberg matrix and (λ, x) is a real eigenpair. Two workspaces are involved. The array A is two-dimensional and must be large enough to store an n-by-n matrix, while *work* is a linear array having dimension at least $4n$. If one does not care about destroying H, then A may be set to H in the calling sequence.

We tested CONDEV (which calls SIGMAN) on numerous eigenpairs obtained via the EISPACK path ORTHES-HQR-INVIT. For example, we computed $s(\lambda_i)$ and $\text{sep}(\lambda_i)$ for all the eigenvalues of the 12-by-12 Frank matrix F_{12}. (F_{12} is an upper Hessenberg matrix whose upper triangular entries are given by $f_{ij} = 13 - j$ and whose subdiagonal entries are given by $f_{j+1,j} = 12 - j$.) The results are given in Table 1. The computed $s(\lambda_i)$ agree with those reported in [18, p. 94]. The calculations were done on a VAX 780 with macheps $\approx 10^{-17}$. Consider λ_{12}. The true value as reported in [35, p. 152] is given by

$$\lambda_{12} = 0.031028060644010.$$

If we apply (1.4) with $\varepsilon = \text{macheps} \cdot \|F_{12}\|$, we find

$$|\hat{\lambda}_{12} - \lambda_{12}| \approx 10^{-10} \approx \frac{\varepsilon}{s(\lambda_{12})},$$

as expected.

Concerning the computed eigenvector $\hat{x}_{12}$ we find that it agrees with the exact x_{12} to the extent predicted by (1.8): see Table 2. In particular,

$$dist(span\langle \hat{x}_{12} \rangle, span\langle x_{12} \rangle) \approx 10^{-10} \approx \frac{\varepsilon}{sep(\lambda_{12})}.$$

TABLE 1

$\hat{\lambda}$	$s(\hat{\lambda})$	$\text{sep}(\hat{\lambda})$
.322288915015722E+02	.304240831905392E+00	.949568465957217E+01
.201989886458771E+02	.200790337133467E+00	.374255793480659E+01
.123110774008685E+02	.318225993866148E+00	.286518367995158E+01
.696153308556712E+01	.584473553642124E+00	.348436016372236E+01
.351185594858076E+01	.144467040367517E+00	.226586002038572E+00
.155398870913211E+01	.462655936357393E−02	.515747433899713E−02
.643505319005355E+00	.691238637430018E−04	.568776334060446E−04
.284749720539282E+00	.178472584657993E−05	.147158317974797E−05
.143646519970700E+00	.149222013377351E−06	.127653526144051E−06
.812276585015030E−01	.375295290160827E−07	.328077935376545E−07
.495074302114390E−01	.257906338088545E−07	.228193201193118E−07
.310280601739661E−01	.546942496623371E−07	.487484292832814E−07

TABLE 2

$\hat{x}_{12}$	x_{12}
−0.00000000059765	−0.00000000059765
0.00000001866394	0.00000001866384
−0.00000037097679	−0.00000037097681
0.00000557001105	0.00000557001123
−0.00006633973347	−0.00006633973492
0.00063559408254	0.00063559409179
−0.00488250613549	−0.00488250618149
0.02956829800842	0.02956829817932
−0.13659636790797	−0.13659636834907
0.45393927999813	0.45393927930764
−0.96897193982603	−0.96897193935599
1.00000000000000	1.00000000000000

The Frank matrix example is interesting, but it does not dramatize the need for computing *both* $s(\lambda)$ and *sep*(λ), since these two quantities have the same order of magnitude for F_{12}. To this end we applied CONDEV to compute the largest eigenpair (λ_1, x_1) of the matrix

$$H(\mu) \equiv \begin{bmatrix} 2\mu & 1 & -2 \\ \mu & -2 & 4 \\ 0 & 1 & -2 \end{bmatrix}$$

For small μ, $H(\mu)$ has three well-conditioned real eigenvalues $\lambda_1 > \lambda_2 = 0 >$

λ_3. Indeed, $s(\lambda_1) \approx O(1)$ as μ approaches zero. However, $\text{sep}(\lambda_1) = O(\mu)$. Thus, we can explore the deterioration of the eigenvector x_1 as μ gets small. For example, if $\mu = 2^{-30}$ then

$$\lambda_1 = \mu - 2 + \sqrt{4 + 5\mu + \mu^2} = 0.20954757928848267 \times 10^{-8},$$

while its exact unit 2-norm eigenvector (to working precision) is prescribed by

$$x_1 = \begin{bmatrix} 0.97049495884276928 \\ 0.21566554640950429 \\ 0.10783277309177166 \end{bmatrix}$$

The absolute error in the computed $\hat{\lambda}_1$ is correctly predicted by (1.4). In this case $s(\lambda_1) = 0.847$. On the other hand $sep(\lambda_1) \approx 10^{-9}$, which explains why the computed unit 2-norm eigenvector

$$\hat{x}_1 = \begin{bmatrix} 0.97049496026962221 \\ 0.21566554127283362 \\ 0.10783277052343633 \end{bmatrix}$$

is correct to only seven places.

We mention that our software only handles real eigenpairs. Of course, one will want to be able to process complex conjugate eigenpairs as well if CONDEV is to be a full partner to the EISPACK codes HQR and INVIT. Our codes could certainly be extended to handle the complex conjugate case. Unfortunately, CONDEV would then require an additional n-by-n workspace. However, for the case of complex conjugate eigenvalues one may be more interested in estimating the accuracy of the two dimensional invariant subspace associated with the real and imaginary parts of the corresponding eigenvectors. Indeed, an interesting topic for further research would be to investigate how to estimate the accuracy of computed invariant subspaces.

I would like to thank Mr. Marc Cohen for producing early versions of the software described in Section 4. The referees helped me a great deal with both the exposition and the technical content of the paper. Finally, I am indebted to Jim Wilkinson for originally kindling my interest in eigenproblem sensitivity.

REFERENCES

1 R. S. Bartels and G. W. Stewart, A solution of the equation $AX + XB = C$, *Comm. ACM* 15:820–826 (1972).

2 C. Bavely and G. W. Stewart, An algorithm for computing reduced subspaces by block diagonalization, *SIAM J. Numer. Anal.* 16:359–367 (1979).

3 P. Businger, Numerically stable deflation of Hessenberg and symmetric tridiagonal matrices, *BIT* 11:262–270 (1971).

4 R. Byers, Hamiltonian and symplectic algorithms for the algebraic Riccati equation, Ph.D. Thesis, Center for Applied Mathematics, Cornell Univ., Ithaca, N.Y. 14853, 1983.

5 S. P. Chan, R. Feldman, and B. N. Parlett, Algorithm 517—a probram for computing the condition numbers of matrix eigenvalues without computing eigenvectors, *ACM Trans. Math. Software* 3:186–203 (1977).

6 A. K. Cline, A. R. Conn, and C. F. Van Loan, Generalizing the LINPACK condition estimator, Cornell Computer Science Tech. Report TR 81-462, Ithaca, N.Y., 1981.

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

8 A. K. Cline and R. K. Rew, A set of counterexamples to three condition estimators, *SIAM J. Sci. Statis. Comput.* 4:602–611 (1983).

9 J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, LINPACK *User's Guide*, SIAM, Philadelphia, 1979.

10 J. Dongarra, C. B. Moler, and J. H. Wilkinson, Improving the accuracy of computed eigenvalues and eigenvectors, *SIAM J. Numer. Anal* 20:23–45 (1983).

11 J. Dongarra, Improving the accuracy of computed singular values, *SIAM J. Sci. Statist. Comput.* 4:712–719 (1983).

12 G. E. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1967.

13 P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, Methods for modifying matrix factorizations, *Math. Comp.* 28:505–535 (1974).

14 D. Goldfarb, Factorized variable metric methods for unconstrained optimization, *Math. Comp.* 30:796–811 (1976).

15 G. H. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins U.P., Baltimore, 1983.

16 G. H. Golub and C. Reinsch, Singular value decomposition and least squares solutions, *Numer. Math.* 14:403–420 (1970).

17 G. H. Golub and J. H. Wilkinson, Ill-conditioned eigensystems and the computation of the Jordan canonical form, *SIAM Rev.* 18:578–619 (1976).

18 R. Gregory and D. Karney, *A Collection of Matrices for Testing Computational Algorithms*, Wiley-Interscience, New York, 1969.

19 B. Kagstrom and A. Ruhe, An algorithm for numerical computation of the Jordan normal form of a complex matrix, *ACM Trans. Math. Software* 6:398–419 (1980).

20 C. Moler and G. W. Stewart, An algorithm for generalized matrix eigenvalue problems, *SIAM J. Numer. Anal.* 10:241–256 (1973).
21 D. P. O'Leary, Estimating matrix condition numbers, *SIAM J. Sci. Statist. Comput.* 1:205–209 (1980).
22 A. Ruhe, An algorithm for numerical determination of the structure of a general matrix, *BIT* 10:196–216 (1970).
23 B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines—*EISPACK *Guide*, 2nd ed., Springer, New York, 1970.
24 R. A. Smith, The condition numbers of the matrix eigenvalue problem, *Numer. Math.* 10:232–240 (1967).
25 G. W. Stewart, On the sensitivity of the eigenvalue problem $Ax = \lambda Bx$, *SIAM J. Numer. Anal.* 9:669–686 (1972).
26 G. W. Stewart, Error bounds for approximate invariant subspaces of closed linear operators, *SIAM J. Numer. Anal.* 8:796–808 (1971).
27 G. W. Stewart, Error and perturbation bounds for subspaces associated with certain eigenvalue problems, *SIAM Rev.* 15:727–764 (1973).
28 G. W. Stewart, *Introduction to Matrix Computations*, Academic, New York, 1973.
29 G. W. Stewart, Algorithm 506: HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix, *ACM Trans. Math. Software* 2:275–280 (1976).
30 G. W. Stewart, On the perturbation of pseudo-inverses, projections, and linear least squares problems, *SIAM Rev.* 19:634–662 (1977).
31 H. J. Symm and J. H. Wilkinson, Realistic error bounds for a simple eigenvalue and its associated eigenvector, *Numer. Math.* 35:113–126 (1980).
32 J. M. Varah, Rigorous machine bounds for the eigensystem of a general complex matrix, *Math. Comp.* 22:793–801 (1968).
33 J. M. Varah, Computing invariant subspaces of a general matrix when the eigensystem is poorly determined, *Math. Comp.* 24:137–149 (1970).
34 J. M. Varah, On the separation of two matrices, *SIAM J. Numer. Anal.* 16:216–222 (1979).
35 J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, N.J., 1963.
36 J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford U.P., Oxford, 1965.
37 J. H. Wilkinson, Note on matrices with a very ill-conditioned eigenproblem, *Numer. Math.* 19:176–178 (1972).
38 J. H. Wilkinson, Sensitivity of Eigenvalues, *Utilitas Math.* 25:5–76 (1984).
39 T. Yakamoto, Error bounds for computed eigenvalues and eigenvectors, *Numer. Math.* 34:189–199 (1980).

Received 13 January 1986; revised 25 June 1986