Accurate Singular Values of Bidiagonal Matrices

(Appeared in the SIAM J. Sci. Stat. Comput., v. 11, n. 5, pp. 873-912, 1990)

James Demmel
Courant Institute
251 Mercer Str.
New York, NY 10012

W. Kahan
Computer Science Division
University of California
Berkeley, CA 94720

Abstract

Computing the singular values of a bidiagonal matrix is the final phase of the standard algorithm for the singular value decomposition of a general matrix. We present a new algorithm which computes all the singular values of a bidiagonal matrix to high relative accuracy independent of their magnitudes. In contrast, the standard algorithm for bidiagonal matrices may compute small singular values with no relative accuracy at all. Numerical experiments show that the new algorithm is comparable in speed to the standard algorithm, and frequently faster.

Keywords: singular value decomposition, bidiagonal matrix, QR iteration

AMS(MOS) subject classifications: 65F20, 65G05, 65F35

1. Introduction

The standard algorithm for computing the singular value decomposition (SVD) of a general real matrix A has two phases [7]:

1) Compute orthogonal matrices P_1 and Q_1 such that $B = P_1^T A Q_1$ is in bidiagonal form, i.e. has nonzero entries only on its diagonal and first superdiagonal.

2) Compute orthogonal matrices P_2 and Q_2 such that $\Sigma = P_2^T B Q_2$ is diagonal and nonnegative. The diagonal entries σ_i of Σ are the *singular values* of A. We will take them to be sorted in decreasing order: $\sigma_i \geq \sigma_{i+1}$. The columns of $Q = Q_1 Q_2$ are the *right singular vectors*, and the columns of $P = P_1 P_2$ are the *left singular vectors*.

This process takes $O(n^3)$ operations, where n is the dimension of A. This is true even though Phase 2 is iterative, since it converges quickly in practice. The error analysis of this combined procedure has a widely accepted conclusion [8], and provided neither overflow nor underflow occurs may be summarized as follows:

> The computed singular values σ_i differ from the true singular values of A by no more than $p(n) \cdot \varepsilon \cdot A$, where $A = \sigma_1$ is the 2-norm of A, ε is the machine precision, and $p(n)$ is a slowly growing function of the dimension n of A.

This is a generally satisfactory conclusion, since it means the computed singular values have errors no larger than the uncertainty in the largest entries of A, if these are themselves the results of previous computations. In particular, singular values not much smaller than A are computable to high relative accuracy. However, small singular values may change completely, and so cannot generally be computed with high relative accuracy.

There are some situations where the smallest singular values are determined much more accurately by the data than a simple bound of the form $p(n)\varepsilon A$ would indicate. In this paper we will show that for bidiagonal matrices the singular values are determined to the same relative

precision as the individual matrix entries. In other words, if all the matrix entries are known to high relative accuracy, all the singular values are also known to high relative accuracy independent of their magnitudes. This will follow from an analogous theorem about the eigenvalues of symmetric tridiagonal matrices with zero diagonal.

In such situations it is desirable to have an algorithm to compute the singular values or eigenvalues to the accuracy to which they are determined by the data. In this paper we present an algorithm for computing all the singular values of a bidiagonal matrix to *guaranteed* high relative accuracy, independent of their magnitudes. Our algorithm is a variation of the usual QR iteration which is used in the standard SVD algorithm. Briefly, it is a hybrid algorithm of the usual QR iteration with a "zero-shifted" QR modified to guarantee forward stability. Numerical experience, which we report below, shows that it is generally faster than the standard algorithm, and ranges from 2.7 times faster to 1.6 times slower counting reduction to bidiagonal form (7.7 times faster to 3.4 times slower not counting reduction to bidiagonal form).

This perturbation theory and algorithm also apply to some classes of symmetric matrices. For example, they may be applied to symmetric tridiagonal matrices with zero diagonal; such matrices arise by reducing skew-symmetric matrices to tridiagonal form. Another class where the perturbation theory applies, so that small relative perturbations in the matrix entries only cause small relative perturbations in the eigenvalues, are scaled diagonally dominant symmetric matrices. A symmetric matrix H is scaled diagonally dominant if $H = DAD$ where D is an arbitrary diagonal matrix and A is symmetric and diagonally dominant in the usual sense. This class includes all symmetric positive definite matrices which may be consistently ordered [1], a class which arises in the numerical solution of elliptic partial differential equations. In particular, this class includes all symmetric positive definite tridiagonal matrices. As before, we can exhibit algorithms to compute the eigenvalues of H to their inherent accuracy. This work will be reported on elsewhere [1].

The rest of this paper is organized as follows. Section 2 presents perturbation theory for the singular values of a bidiagonal matrix, and shows that small relative perturbations in the nonzero entries of a bidiagonal matrix can only cause small relative perturbations in its singular values. We also present theorems which say when an offdiagonal entry can be set to zero without making large relative perturbations in any singular value; these theorems are the basis of the convergence criteria for the new algorithm. Section 3 presents the algorithm, which is QR iteration with a "zero shift," modified to be forward stable. This forward stability combined with the perturbation theorem of section 2 shows that QR can compute all the singular values with high relative accuracy. Section 4 discusses convergence criteria for the new algorithm, since the convergence criteria for the standard algorithm can cause unacceptably large perturbations in small singular values. It also discusses the practical algorithm, which is a hybrid of the standard algorithm and the algorithm of section 3. Details of the implementation, including high-level code for the entire algorithm, are presented in section 5. Sections 3, 4 and 5 may be read independently of section 2. Section 6 shows how to use bisection, Rayleigh quotient iteration, and various other schemes to compute the singular values of a bidiagonal matrix to high relative accuracy. Bisection will be used to verify the results in section 7, which discusses numerical experiments. Section 7 also addresses the implications of our results for the "perfect shift" strategy for computing singular vectors. Section 8 contains a detailed error analysis of the new algorithm. Section 9 discusses the accuracy of the computed singular vectors; a complete analysis of this remains an open question. Sections 6 through 9 may be read independently. Sections 7 and 8 depend only on sections 3 through 5. Section 10 contains suggestions for parallel versions of the algorithms presented, open questions, and conclusions.

2. Perturbation Theory for Singular Values of Bidiagonal Matrices

We say δa is a relative perturbation of a of size at most η if $|\delta a| \le \eta |a|$. If A and δA are matrices, we will let $|A|$ and $|\delta A|$ denote the matrices of absolute entries of A and δA. We will say that δA is a componentwise relative perturbation of A of size at most η if $|\delta A| \le \eta |A|$, where the inequality is understood componentwise.

In this section we will prove three perturbation theorems for singular values of bidiagonal matrices. The first theorem is needed to prove that our new QR iteration does not disturb any singular values, and the second two theorems justify our new convergence criteria (see section 4 below).

The first theorem shows that if δB is a componentwise relative perturbation of size η of the n by n bidiagonal matrix B, then the singular values σ_i' of $B+\delta B$ will be relative perturbations of the singular values σ_i of B of size less than about $(2n-1)\eta$, provided $(2n-1)\eta$ is small compared to 1. More precisely we will show that

$$(1-\eta)^{2n-1} \cdot \sigma_i \le \sigma_i' \le (1-\eta)^{1-2n} \cdot \sigma_i$$

(recall that σ_i' and σ_i are sorted in decreasing order). This will follow as a corollary of a more general result for symmetric tridiagonal matrices with zero diagonal.

The last two theorems say when we can set an offdiagonal entry of a bidiagonal matrix B to zero without making large relative perturbations in the singular values. They are based on a simple recurrence for estimating the smallest singular value of a bidiagonal matrix; if setting an offdiagonal entry of B to zero cannot change this recurrence significantly, we show that no singular value can be changed significantly either.

The proof of the first theorem depends on Sylvester's Law Of Inertia [6, p.297]:

Sylvester's Law Of Inertia: Let A be symmetric and U be nonsingular. Then A and UAU^T have the same number of positive, zero and negative eigenvalues.

In particular, suppose A is symmetric and tridiagonal, with diagonal entries $a_1, \ldots, a_n$ and offdiagonal entries $b_1, \ldots, b_{n-1}$. Then via Gaussian elimination without pivoting one can write $A - xI = LDL^T$, where L is unit lower triangular and bidiagonal, and D is diagonal with entries d_i given by the recurrence [15, p. 47]

$$\begin{aligned} d_1 &= a_1 - x \\ d_i &= a_i - x - b_{i-1}^2 / d_{i-1} \end{aligned} \qquad (2.1)$$

This recurrence will not break down ($d_i = 0$ for some $i < n$) as long as x is not one of the $n(n-1)/2$ eigenvalues of leading submatrices of A. Then by Sylvester's Law of Inertia, the numbers of eigenvalues of A less than x, equal to x, and greater than x are precisely the numbers of d_i which are negative, zero and positive, respectively.

We will also need the following classical eigenvalue perturbation theorem due to Weyl:

Theorem 1: [15, p. 191] Let $\lambda_1 \ge \cdots \ge \lambda_n$ be the eigenvalues of the symmetric matrix A, and $\lambda_1' \ge \cdots \ge \lambda_n'$ be the eigenvalues of the symmetric matrix $A+\delta A$. Then $-\delta A \le \lambda_{\min}(\delta A) \le \lambda_i' - \lambda_i \le \lambda_{\max}(\delta A) \le \delta A$. Here, $\lambda_{\min}$ and $\lambda_{\max}$ denote the smallest and largest eigenvalues, respectively.

Now we present our central result of this section (a slightly weaker version originally appeared in an unpublished report [12]):

Theorem 2: Let J be an n by n symmetric tridiagonal matrix with zero diagonal and offdiagonal entries $b_1, \ldots, b_{n-1}$. Suppose $J+\delta J$ is identical to J except for one offdiagonal entry, which changes to αb_i from b_i, $\alpha \ne 0$. Let $\bar{\alpha} = \max(|\alpha|, |\alpha^{-1}|)$. Let λ_i be the eigenvalues of J sorted into decreasing order, and let λ_j' be the eigenvalues of $J+\delta J$ similarly sorted. Then

$$\frac{\lambda_i}{\bar{\alpha}} \le \lambda_i' \le \bar{\alpha} \cdot \lambda_i \quad . \qquad (2.2)$$

In other words, changing any single entry of J by a factor α can change no eigenvalue by more

than a factor $|\alpha|$.

Proof: Assume without loss of generality that $\alpha > 0$, and no b_i is zero, since otherwise J is block diagonal, and each diagonal block may be analyzed separately. The recurrences corresponding to (2.1) for $J - xI$ and $J + \delta J - xI$ may be written

$$\begin{aligned} u_1 &= -x \\ u_{k+1} &= -x - b_k^2 / u_k \end{aligned} \qquad \text{and} \qquad \begin{aligned} v_1 &= -x \\ v_{k+1} &= -x - b_k^2 / v_k \quad , \quad k \neq i \\ v_{i+1} &= -x - \alpha^2 b_i^2 / v_i \end{aligned}$$

Since both J and $J + \delta J$ have nonzero offdiagonals, they must have simple eigenvalues [15, p. 124] λ_i and λ_i', respectively. As long as x is not one of the $n(n-1)$ eigenvalues of leading principal submatrices of J and $J + \delta J$, no division by zero will occur in these recurrences. Also, $u_n = 0$ if and only if x is an eigenvalue of J, and $v_n = 0$ if and only if x is an eigenvalue of $J + \delta J$.

Our goal is to show that each λ_i is the i-th eigenvalue of some symmetric matrix $J(\lambda_i)$ which differs from $J + \delta J$ by a matrix $X = J + \delta J - J(\lambda_i)$ satisfying

$$\begin{aligned} (\bar{\alpha}^{-1} - 1)\lambda_i \leq \lambda_{\min}(X) \leq \lambda_{\max}(X) \leq (\bar{\alpha} - 1)\lambda_i \quad \text{if } \lambda_i \geq 0 \\ (\bar{\alpha} - 1)\lambda_i \leq \lambda_{\min}(X) \leq \lambda_{\max}(X) \leq (\bar{\alpha}^{-1} - 1)\lambda_i \quad \text{if } \lambda_i < 0 \end{aligned} \quad ; \qquad (2.3)$$

together with Theorem 1 these inequalities will yield the desired result.

We construct $J(\lambda_i)$ as follows. Let

$$\begin{aligned} w_j &= u_j \cdot \alpha^{(-1)^{i-j}} \quad \text{if } j \leq i \\ w_j &= u_j \cdot \alpha^{(-1)^{i-j-1}} \quad \text{if } j > i \end{aligned}$$

Note that the w_j satisfy the recurrence

$$\begin{aligned} w_1 &= -x\alpha^{(-1)^{i-1}} \\ w_{j+1} &= -x\alpha^{(-1)^{i-j-1}} - b_j^2 / w_j \quad \text{if } j < i \\ w_{i+1} &= -x\alpha - \alpha^2 b_i^2 / w_i \\ w_{j+1} &= -x\alpha^{(-1)^{i-j}} - b_j^2 / w_j \quad \text{if } j > i \end{aligned} \quad ,$$

or

$$\begin{aligned} w_1 &= -x - x_1 \\ w_{j+1} &= -x - x_{j+1} - b_j^2 / w_j \quad \text{if } j < i \\ w_{i+1} &= -x - x_{i+1} - \alpha^2 b_i^2 / w_i \\ w_{j+1} &= -x - x_{j+1} - b_j^2 / w_j \quad \text{if } j > i \end{aligned} \quad ,$$

which is the recurrence for $J(x) = J + \delta J - X$ where $X = diag(x_i)$, $x_i = (\alpha^{\pm 1} - 1)x$. Now set $x = \lambda_i$. Since the w_j and u_j sequences have the same signs by construction (including $u_n = w_n = 0$), λ_i is the i-th eigenvalue of $J(\lambda_i)$. Further, $\lambda_{\max}(X)$ and $\lambda_{\min}(X)$ clearly satisfy (2.3) above. □

As an immediate corollary we get

Corollary 1: Let J be an n by n symmetric tridiagonal matrix with zero diagonal and offdiagonal entries $b_1, \ldots, b_{n-1}$. Let $J + \delta J$ have off diagonal entries $\alpha_1 b_1, \ldots, \alpha_{n-1} b_{n-1}$, $\alpha_i \neq 0$. Let $\bar{\alpha} = \prod_{i=1}^{n-1} \max(|\alpha_i|, |\alpha_i^{-1}|)$. Let λ_i be the eigenvalues of J sorted into decreasing order, and λ_i' be the eigenvalues of $J + \delta J$ similarly sorted. Then

$$\frac{\lambda_i}{\bar{\alpha}} \leq \lambda_i' \leq \bar{\alpha} \cdot \lambda_i \quad .$$

For example, if $1 - \eta \leq |\alpha| \leq 1 + \eta$, no eigenvalue can change by a factor exceeding $\bar{\alpha} = (1 - \eta)^{-n+1}$.

We can apply Theorem 2 to prove a similar theorem for singular values of bidiagonal matrices by noting that for any matrix B the eigenvalues of

$$B' \equiv \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$$

are the singular values of B, their negatives, and some zeros (if B is not square) [8, p. 286]. Suppose now that B is n by n and bidiagonal with diagonal entries $s_1, \ldots, s_n$ and superdiagonal entries $e_1, \ldots, e_{n-1}$. Then by permuting the rows and columns of B' to appear in the new order $1, n+1, 2, n+2, \cdots, n, 2n$, we see B' is orthogonally similar to the tridiagonal matrix B'' with zeros on the diagonal and offdiagonals $s_1, e_1, s_2, e_2, \cdots, e_{m-1}, s_m$ [7, p. 213]. Thus the singular values of B are the absolute values of the eigenvalues of the matrix B'' which is of the form required by Theorem 2. This proves

Corollary 2: Let B be an n by n bidiagonal matrix and suppose $\delta B_{ii} + B_{ii} = \alpha_{2i-1} B_{ii}$, $\delta B_{i,i+1} + B_{i,i+1} = \alpha_{2i} B_{i,i+1}$, $\alpha_j \neq 0$. Let $\bar{\alpha} = \prod_{i-1}^{2n-1} \max(|\alpha_i|, |\alpha_i^{-1}|)$. Let $\sigma_1 \geq \cdots \geq \sigma_n$ be the singular values of B, and let $\sigma_1' \geq \cdots \geq \sigma_n'$ be the singular values of $B + \delta B$. Then

$$\frac{\sigma_i}{\bar{\alpha}} \leq \sigma_i' \leq \bar{\alpha} \cdot \sigma_i \ .$$

For example, if $1-\eta \leq |\alpha_j| \leq 1+\eta$, then no singular value can change by more than a factor of $\bar{\alpha} = (1-\eta)^{1-2n}$.

That this result is essentially best possible may be seen by considering the n by n matrix

$$B(\eta) = \begin{bmatrix} 1-\eta & \beta(1+\eta) & & \\ & \cdot & \cdot & \\ & & \cdot & \beta(1+\eta) \\ & & & 1-\eta \end{bmatrix}$$

When $\beta >> 1$, the smallest singular value is approximately $\beta^{1-n}(1-(2n-1)\eta)$.

This theorem may be contrasted with the following classical perturbation bound for singular values, where it is only possible to bound the absolute perturbation in the singular values of a perturbed general matrix:

Theorem 3: [8, p. 286] Let $\sigma_1 \geq \cdots \geq \sigma_n$ be the singular values of A, and $\sigma_1' \geq \cdots \geq \sigma_n'$ be the singular values of $A + \delta A$. Then $|\sigma_i' - \sigma_i| \leq \delta A$.

One caveat about the use of Corollary 2 in practice is that phase 1 of the SVD algorithm, reduction to bidiagonal form, may produce completely inaccurate bidiagonal entries. Sometimes, however, the reduction to bidiagonal form is quite accurate, so that the singular values of the original matrix can be computed accurately (see [1] for discussion).

In section 6 we will show how to use recurrence (2.1) in practice to compute the singular values of a bidiagonal matrix with guaranteed high relative accuracy. This method, though not competitive in speed on a serial machine with the algorithm of the next section, can be used to efficiently verify the accuracy of the singular values computed by another method. The algorithm based on (2.1) may also be parallelized easily (see section 6).

The second result of this section tells us when we can set an offdiagonal of B to zero without making large relative changes in the singular values. This theorem will justify one of the convergence criteria we describe in section 4 below.

First we discuss a simple recurrence for approximating the smallest singular value of a bidiagonal matrix, which also appeared in [9]:

Lemma 1: Let B be a n by n bidiagonal matrix with nonzero diagonal entries $s_1, \ldots, s_n$ and nonzero offdiagonal entries $e_i, \ldots, e_{n-1}$. Consider the following recurrences:

$$\begin{array}{ll} \lambda_n = |s_n| & \mu_1 = |s_1| \\ \text{for } j=n-1 \text{ to } 1 \text{ step } -1 \text{ do} & \text{for } j=1 \text{ to } n-1 \text{ do} \\ \quad \lambda_j = |s_j| \cdot (\lambda_{j+1}/(\lambda_{j+1} + |e_j|)) & \quad \mu_{j+1} = |s_{j+1}| \cdot (\mu_j/(\mu_j + |e_j|)) \end{array} \tag{2.4}$$

Then $B^{-1}{}_{\infty}^{-1} = \min_j \lambda_j$ and $B^{-1}{}_1^{-1} = \min_j \mu_j$. Furthermore, letting $\underline{\sigma} \equiv \min(B^{-1}{}_1^{-1}, B^{-1}{}_{\infty}^{-1})$, we have

$$\begin{aligned} n^{-1/2} \cdot B^{-1}{}_{\infty}^{-1} \le \sigma_{\min}(B) \le n^{1/2} \cdot B^{-1}{}_{\infty}^{-1} \\ n^{-1/2} \cdot B^{-1}{}_1^{-1} \le \sigma_{\min}(B) \le n^{1/2} \cdot B^{-1}{}_1^{-1} \\ \underline{\sigma} \le \sigma_{\min}(B) \le n^{1/2} \cdot \underline{\sigma} \end{aligned} \quad . \tag{2.5}$$

Proof: By means of pre- and postmultiplication by unitary diagonal matrices with diagonal entries of unit modulus, we may assume that $s_i > 0$ and $e_i < 0$. Then B^{-1} is easily seen to have positive superdiagonal entries, so that $B^{-1}{}_{\infty} = B^{-1} u_{\infty}$ and $B^{-1}{}_1 = u^T B^{-1}{}_1$, where u is the vector of all ones. $v = B^{-1}u$ and $w^T = u^T B^{-1}$ are easily computed by back and forward substitution. Thus $B^{-1}{}_{\infty} = \max_i |v_i|$ and $B^{-1}{}_1 = \max_i |w_i|$. Modifying these back and forward substitution recurrences to compute $\lambda_i = 1/v_i$ and $\mu_i = 1/w_i$ yields the recurrences in (2.4). Since the eigenvalues of

$$H = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$$

are the positive and negative singular values of B,

$$B^{-1} = H^{-1} \le H^{-1}{}_{\infty} = \max(B^{-1}{}_{\infty}, B^{-T}{}_{\infty}) = \max(B^{-1}{}_{\infty}, B^{-1}{}_1) \ ,$$

proving the inequality $\underline{\sigma} \le \sigma_{\min}(B)$ in (2.5). The other inequalities are standard norm inequalities. □

From Lemma 1 it is clear that if $|e_j/\lambda_{j+1}| \le \eta < 1$, then changing e_j to 0 can make a relative change of at most η in λ_j and all subsequent λ_i, $i < j$. Thus the first upper and lower bounds on $\sigma_{\min}(B)$ in (2.5) can change only by a factor of η as well. Similar comments apply if $|e_j/\mu_j| \le \eta < 1$. This suggests the following criterion for setting e_j to 0:

Convergence Criterion 1:
Let $\eta < 1$ be the desired relative accuracy of computed singular values. Then if either $|e_j/\lambda_{j+1}| \le \eta$ or $|e_j/\mu_j| \le \eta$, set e_j to 0.

Now we will state and prove a theorem which justifies this criterion. We will only prove the theorem for the case $|e_j/\lambda_{j+1}| \le \eta$; the case $|e_j/\mu_j| \le \eta$ is analogous. First we need some notation. Let $\phi(\eta)$ be the unique positive solution of

$$\exp(2\phi) - 2\phi - 1 = \eta^2 \ ; \tag{2.6}$$

it is easy to see that $\phi(\eta)$ is asymptotically $\eta/2^{1/2}$ for small η and that for all η, $\phi(\eta) \le \eta/2^{1/2}$. Let B be a bidiagonal matrix as in Lemma 1 with singular values $\sigma_1 \ge \cdots \ge \sigma_n$, let $U = B$ except for entry e_j which is zero, and let $\sigma_1' \ge \cdots \ge \sigma_n'$ be the singular values of U. Let $_i(\eta)$ be the interval of σ's such that

$$-\phi(\eta) \le \ln(\sigma_i / \sigma_i') \le \phi(\eta) \ ; \tag{2.7}$$

$_i(\eta)$ is essentially the set of σ which differ from σ_i' by a relative perturbation of at most $\eta/2^{1/2}$. Some of these intervals may overlap; let (η) denote the collection of disjoint intervals made of connected components of $\bigcup_i {}_i(\eta)$. Now we may state

Theorem 4: Let B and U be bidiagonal matrices as described above, and suppose $|e_j/\lambda_{j+1}| \le \eta$. Then each singular value σ_i of B lies in the connected component of (η) containing $_i(\eta)$. In particular, if that connected component consists of m overlapping intervals $_j(\eta)$, then

$$-m\phi(\eta) \le \ln(\sigma_i' / \sigma_i) \le m\phi(\eta) \tag{2.8}$$

Therefore, the relative perturbation caused in σ_i by setting the offdiagonal entry in δB to zero is at most $n\eta/2^{1/2}$ if $n\eta 1$, and if σ_i is sufficiently separated from the other singular values, at most $\eta/2^{1/2}$.

For example, this theorem lets us conclude that setting η to 0 in

$$\begin{bmatrix} 1 & \eta & \\ & 1 & 1 \\ & & \alpha \end{bmatrix}$$

can change the singular values $2^{1/2}$, 1, and $2^{-1/2}\alpha$ by at most factors of $1\pm\eta$ if η is small, independent of α. Indeed, if D is *any* bidiagonal matrix this theorem guarantees that we can set η to 0 in

$$\begin{bmatrix} 1 & \eta \\ & D \end{bmatrix}$$

without making relative perturbations larger than η in *any* singular value.

The proof of this theorem will depend on a sequence of technical lemmas. The first is a trivial consequence of Taylor's Theorem:

Lemma 2: Let f and g be continuously differentiable functions on the nonnegative real axis, with $f(t) < g(t)$ for t positive and sufficiently small. Let $\xi \equiv \inf\{t > 0: f(t) \ge g(t)\}$, and $\xi = \infty$ if no such t exist. Then if ξ is finite, $f'(\xi) \ge g'(\xi)$.

We will use the contrapositive of this result to show when $f < g$ for all t; if $f(\xi) \ge g(\xi)$ would imply that $f'(\xi) < g'(\xi)$, then f must be less than g everywhere.

In our case, we define $f(t)$ and $g(t)$ as follows. Write

$$B = \begin{bmatrix} K & C \\ & R \end{bmatrix}$$

where K is j by j, R is $n-j$ by $n-j$, and $C = e_j l f^T$, where $l = (0, ..., 0, 1)^T$ and $f = (1, 0, ..., 0)^T$. Assume as in Lemma 1 that $e_j < 0$. Let

$$U(t) = \begin{bmatrix} K & C(t) \\ & R \end{bmatrix}$$

where $C(t) = -t\lambda_{j+1} l f^T$ so that $U(0) = U$ and $U(\eta) = B$. Let $\sigma_i'(t)$ be the i-th largest singular value of $U(t)$. Then we first let

$$f(t) = -\phi(t) \quad \text{and} \quad g(t) = \ln(\sigma_i'(t)/\sigma_i') \tag{2.9a}$$

and apply Lemma 2 to prove the first inequality in (2.8), and then let

$$f(t) = \ln(\sigma_i'(t)/\sigma_i') \quad \text{and} \quad g(t) = \phi(t) \tag{2.9b}$$

to prove the second inequality. In order to apply Lemma 2, we need to compute the derivatives of the functions in (2.9).

Lemma 3: Unless $\sigma_i'(t)$ is a singular value of R,

$$\min\left(\min_j \frac{t}{((\sigma_i'(t)/\sigma_j')^2 - 1)}, 0\right) < \frac{d}{dt}\ln(\sigma_i'(t)) < \max\left(\max_j \frac{t}{((\sigma_i'(t)/\sigma_j')^2 - 1)}, 0\right).$$

Proof: We begin with a simplifying assumption: We assume K and R have no common singular values. If this is not true, consider a sequence of problems with $K_n \to K$, $R_n \to R$ and where K_n and R_n have distinct singular values; the general result will follow from continuity.

We may define a singular value $\sigma(t)$ and its singular vectors $u(t)$ and $v(t)$ of $U(t)$ by the equations $Uv = \sigma u$ and $u^T U = \sigma v^T$ (where we have suppressed the argument t). Using the fact

that $u^T u = v^T v > 0$, we see from $\dot{U}v + U\dot{v} = \dot{\sigma}u + \sigma\dot{u}$ and from $\dot{u}^T U + u^T \dot{U} = \dot{\sigma} v^T + \sigma \dot{v}^T$ that

$$\dot{\sigma} = u^T \dot{U} v / v^T v = u^T \dot{U} v / u^T u \quad .$$

Now partition $u^T = (u_1^T, u_2^T)$ and $v^T = (v_1^T, v_2^T)$ conformally to B, whence

$$\begin{aligned} Kv_1 + Cv_2 &= \sigma u_1 \\ Rv_2 &= \sigma u_2 \end{aligned} \qquad \text{and} \qquad \begin{aligned} u_1^T K &= \sigma v_1^T \\ u_1^T C + u_2^T R &= \sigma v_2^T \end{aligned} \quad .$$

Now

$$\dot{U}(t) = \begin{bmatrix} 0 & \dot{C} \\ 0 & 0 \end{bmatrix} \quad \text{where} \quad \dot{C} = -\lambda_{j+1} \cdot l f^T \quad .$$

By rearranging the recurrence (2.4) for λ_{j+1} we see $\lambda_{j+1} = R^{-T} f_1^{-1}$. Thus

$$\dot{\sigma}(t) = \frac{u_1^T \dot{C} v_2}{u_1^T u_1 + u_2^T u_2} = \frac{-u_1^T l f^T v_2}{R^{-T} f_1 (u_1^T u_1 + u_2^T u_2)} \tag{2.11}$$

Now we derive another expression for v_2 in order to eliminate it from (2.11). Since

$$R^T R v_2 = \sigma R^T u_2 = \sigma^2 v_2 - \sigma C^T u_1 = \sigma^2 v_2 + \sigma t f l^T u_1 \; R^{-T} f_1^{-1}$$

we may solve for v_2 as follows provided σ is not a singular value of R:

$$v_2 = \sigma t l^T u_1 (R^T R - \sigma^2 I)^{-1} f \; R^{-T} f_1^{-1} = \sigma t l^T u_1 R^{-1} (I - \sigma^2 (R^T R)^{-1})^{-1} R^{-T} f \; R^{-T} f_1^{-1}$$

and so

$$\frac{d}{dt} \ln(\sigma(t)) = \dot{\sigma}(t)/\sigma(t) = t \cdot \frac{(l^T u_1)^2}{u_1^T u_1 + u_2^T u_2} \cdot \frac{R^{-T} f^2}{R^{-T} f_1^2} \cdot \frac{(R^{-T} f)^T (\sigma^2 (R^T R)^{-1} - I)^{-1} R^{-T} f}{R^{-T} f^2} \quad . \tag{2.12}$$

Since l is a unit vector, the second factor in this expression is between 0 and 1. It cannot be zero because otherwise $C^T u_1 = 0$, $Rv_2 = \sigma u_2$ and $u_2^T R = \sigma v_2^T$, and so σ would be a singular value of R contrary to assumption. The third factor is strictly between 0 and 1. The last factor is a Rayleigh quotient and so bounded by the extreme eigenvalues of the matrix in the middle, i.e. $\min_j ((\sigma/\sigma_{Rj})^2 - 1)^{-1}$, and $\max_j ((\sigma/\sigma_{Rj})^2 - 1)^{-1}$, where σ_{Rj} are the singular values of R. This is in turn bounded by the extreme values of $1/((\sigma/\sigma'_j)^2 - 1)$. This proves the lemma. □

Lemma 4: $\phi(0) = 0$ and $\dot{\phi}(0) = 2^{-1/2}$. $\phi(t)$ satisfies the differential equation

$$\dot{\phi}(t) = \frac{t}{\exp(2\phi(t)) - 1} \tag{2.13a}$$

and $\psi(t) = -\phi(t)$ satisfies the differential equation

$$\dot{\psi}(t) = \frac{t}{1 - \exp(-2\psi(t))} \tag{2.13b}$$

Proof: Simply differentiate the defining equation (2.6) for $\phi(t)$. □

Proof of Theorem 4: Now note that $\ln(\sigma_i'(0)/\sigma_i') = 0$ and its derivative $\dot{\sigma}_i'(0)/\sigma_i'(0) = 0$ as well since $\sigma_i'(t)$ is an even function of t. Since $\phi(0) = 0$ and $\dot{\phi}(0) = 2^{-1/2}$, we see that equation (2.8) is true (for $m = 1$) for sufficiently small η. To show it is true for all η, we assume to the contrary that there is some positive η for which it is false, and let ξ be the infimum of all these η. Then $\sigma_i'(\xi)$ will be on the boundary of (η), which means $|\ln(\sigma_i'(\xi)/\sigma_j'|$ will be at least $\phi(\xi)$ for all j. From Lemma 3 we see this implies

$$\frac{-t}{1 - \exp(-2\phi(\xi))} < \frac{d}{dt} \ln(\sigma_i'(\xi)/\sigma_i') < \frac{t}{\exp(2\phi(\xi) - 1} \quad .$$

But we also have from Lemma 4 that that

$$\dot{\psi}(\xi) = \frac{t}{1-\exp(-2\psi(\xi))} \quad \text{and} \quad \dot{\phi}(\xi) = \frac{t}{\exp(2\phi(\xi))-1} \ .$$

Therefore, the choice (2.9a) of f and g yields $\dot{f}(\xi) < \dot{g}(\xi)$, so $f(\xi)$ cannot equal $g(\xi)$. The choice (2.9b) yields the same conclusion. Therefore, $\sigma_i'(\xi)$ cannot lie on the boundary of (ξ) as supposed. This completes the proof of Theorem 4. □

The third result in this section supplies a convergence criterion which may occasionally succeed in setting an offdiagonal entry to zero before Convergence Criterion 1. However, it may only be applied when singular vectors are *not* computed, since it may cause rather large perturbations in them. Let

$$B = \begin{bmatrix} D & \tilde{e} \\ 0 & s \end{bmatrix} \quad \text{and} \quad B' = \begin{bmatrix} D & 0 \\ 0 & s \end{bmatrix} \tag{2.14}$$

where $\tilde{e}=[0, \cdots, 0, e]^T$, and D is bidiagonal. Let $\sigma_1 \geq \cdots \geq \sigma_n$ be the singular values of B and $\sigma'_1 \geq \cdots \geq \sigma'_n$ be the singular values of B'. Let $\tilde{}_i(\eta)$ be the interval of σ's such that $|\sigma - \sigma'_i| \leq \eta\sigma'_i$, and let $\tilde{}(\eta)$ be the collection of disjoint intervals which are the connected components of $\bigcup_i \tilde{}_i(\eta)$. Now we may state

Theorem 5: Let $0 < \eta < 1$ be a relative error tolerance, and suppose $gap \equiv \sigma_{\min}(D) - |s| > 0$ in (2.14). If

$$|e|^2 \leq .5 \cdot \eta \cdot gap \cdot (\sigma_{\min}(D) + |s|) = .5 \cdot \eta \cdot (\sigma^2_{\min}(D) - |s|^2) \tag{2.15}$$

then each singular value σ'_i of B' lies in the connected component of $\tilde{}(\eta)$ containing $\tilde{}_i(\eta)$. In particular, if that connected component consists of m overlapping intervals $\tilde{}_j(\eta)$, and $m\eta 1$, then $|\sigma'_i - \sigma_i|$ is at most about $m \cdot \eta \cdot \sigma'_i$.

Proof: We consider two cases: $se/[gap(\sigma_{\min}(D) + |s|)] \geq .5$, and $se/[gap(\sigma_{\min}(D) + |s|)] < .5$. In the first case (2.15) implies $e^2 < \eta se$ or $e < \eta s$. Then by Theorem 3 setting e to 0 in B can change no singular value by more than $|\eta s|$, proving the theorem in this case.

Now consider the second case. Instead of directly comparing the singular values of B and B', we compare the eigenvalues of BB^T and $B'B'^T$, which are the squares of the corresponding singular values. First we show that the smallest eigenvalues of BB^T and $B'B'^T$ must be close. The smallest eigenvalue of $B'B'^T$ is s^2. By Theorem 1, BB^T has one eigenvalue less than $s^2 + se \leq (\sigma^2_{\min}(D) - s^2)/2$, and the rest exceeding the same quantity. By the gap theorem [15, 11-7-1], the smallest eigenvalue σ_n^2 of BB^T satisfies

$$|\sigma_n^2 - s^2| \leq \frac{2s^2e^2}{\sigma^2_{\min}(D) - s^2} \leq \eta s^2$$

proving the theorem for the smallest eigenvalue.

Now we consider the larger eigenvalues of BB^T. Since $se/[gap(\sigma_{\min}(D) + |s|)] < .5$, Theorem 4.12 of [16] applies and we conclude that the larger eigenvalues of BB^T are the same as the eigenvalues of $DD^T + E_1 + E_2$, where

$$E_1 = \begin{bmatrix} 0 & \cdot & 0 \\ \cdot & \cdot & \cdot \\ 0 & \cdot & e^2 \end{bmatrix}$$

so that $E_{1F} = e^2$, and $E_{2F} \leq 2s^2e^2/[gap(\sigma_{\min}(D) - |s|)] \leq \eta s^2$; E_2 is in general nonsymmetric. By the Bauer-Fike Theorem [8] each eigenvalues of $DD^T + E_1 + E_2$ is within

$$E_{1F} + E_{2F} \leq e^2 + \eta s^2 \leq .5\eta(\sigma^2_{\min}(D) - s^2) + \eta s^2 \leq \eta\sigma^2_{\min}(D)$$

of an eigenvalue of DD^T. This completes the proof. □

This theorem justifies

Convergence Criterion 2:

Let $B = \begin{bmatrix} D & \tilde{e} \\ 0 & s \end{bmatrix}$ (or $B = \begin{bmatrix} s & \tilde{e}^T \\ 0 & D \end{bmatrix}$) where $\tilde{e} = [0, \cdots, 0, e]^T$ (or $\tilde{e} = [e, 0, \cdots, 0]^T$). Let $\eta < 1$ be the desired relative accuracy of the computed singular values. Then if $gap = \sigma_{\min}(D) - |s| > 0$ and $|e|^2 \leq .5 \cdot \eta \cdot gap \cdot (\sigma_{\min}(D) + |s|) = .5 \cdot \eta \cdot (\sigma^2_{\min}(D) - |s|^2)$, set e to zero. From (2.5), we may approximate $\sigma_{\min}(D)$ with the lower bound $\min_{j<n} \mu_j/(n-1)^{1/2}$ (or $\min_{j>1} \lambda_j/(n-1)^{1/2}$).

The following example shows that Convergence Criterion 2 may sometimes set e to zero before Convergence Criterion 1. Consider $B = \begin{bmatrix} 1 & e \\ 0 & .5 \end{bmatrix}$. Convergence Criterion 1 demands that $|e| \leq \eta$ to set it to zero, whereas Convergence Criterion 2 demands only that $|e| \leq (3\eta/8)^{1/2}$, which may be much larger.

This same example also shows why we do not want to use Convergence Criterion 2 when computing singular vectors. The right singular vectors of B and B_0 differ by $O(|e|)$, not $O(|e|^2)$.

In practice, we may estimate $\sigma_{\min}(D)$ using (2.4) and (2.5), and indeed we need only run the recurrences once to apply both Convergence Criteria 1 and 2.

3. QR iteration with a zero shift

The standard algorithm for finding singular values of a bidiagonal matrix B is the QR algorithm applied implicitly to B^TB [7]. The algorithm computes a sequence B_i of bidiagonal matrices starting from $B_0=B$ as follows. From B_i the algorithm computes a shift σ^2, which is usually taken to be the smallest eigenvalue of the bottom 2 by 2 block of $B_iB_i^T$. Then the algorithm does an implicit QR factorization of the shifted matrix $B_i^TB_i-\sigma^2 I = QR$, where Q is orthogonal and R upper triangular, from which it computes a bidiagonal B_{i+1} such that $B_{i+1}^TB_{i+1} = RQ+\sigma^2 I$. As i increases, B_i converges to a diagonal matrix with the singular values on the diagonal.

The roundoff errors in this algorithm are generally on the order of εB, where ε is the precision of the floating point arithmetic used. From Theorem 3 of the last section, this means we would expect absolute errors in the computed singular values of the same order. In particular, tiny singular values of B could be changed completely.

In this section we present a variation of this standard algorithm which computes all the singular values of a bidiagonal matrix, even the tiniest ones, with guaranteed high relative accuracy. We will call this the "implicit zero-shift QR" algorithm, since it corresponds to the above algorithm when $\sigma=0$. However, it is organized in such a way as to guarantee that each entry of B_{i+1} is computed from B_i to nearly full machine precision. Then Corollary 2 of the last section implies that the singular values of B_i and B_{i+1} all agree to high relative accuracy. When B_{i+1} has finally converged to a diagonal matrix, these diagonal entries must therefore also be accurate singular values for the initial $B=B_0$. Exactly how to detect this convergence is an interesting issue and discussed in the next section.

The rest of this section is organized as follows. First we review the standard algorithm for singular values of a bidiagonal matrix. Then we show how it simplifies when the shift is zero. Next we discuss an error analysis of the resulting implicit zero-shift QR algorithm which shows that it computes each entry of B_{i+1} with high relative accuracy (the details of the error analysis are in section 8). Finally, we discuss the asymptotic convergence rate.

The final algorithm is a hybrid of the standard QR and implicit zero-shift QR. Standard QR is used when the condition number of B (the ratio of the largest to smallest singular values) is modest. In this case the roundoff errors are guaranteed to make acceptably small perturbations in the smallest singular values of B. If the condition number is large, we use implicit zero-shift QR instead. The hybrid algorithm will be discussed more fully in the next section.

In order to summarize the standard QR algorithm, we need some notation. Let $J(i,j,\theta)$ denote the Given's rotation in entries i and j by angle θ. In other words, $J(i,j,\theta)$ is an n by n identity matrix except for rows and columns i and j whose intersections consist of the following 2 by 2 rotation matrix:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Given the vector x, choosing θ so that $x_j/x_i=\tan\theta$ means that the i-th and j-th entries of $J(i,j,\theta)x$ will contain $\pm\sqrt{(x_i^2+y_i^2)}$ and 0, respectively.

We will illustrate the algorithm on a 4 by 4 example, where we use x and + to indicate nonzero entries and 0 and blank to indicate zero entries. Initially B_i is in the form

$$B_i = \begin{bmatrix} x & x & & \\ & x & x & \\ & & x & x \\ & & & x \end{bmatrix} .$$

We begin by postmultiplying B_i by $J_1 \equiv J(1,2,\theta_1)$, where θ_1 will be discussed in a moment. This introduces a nonzero entry in the (2,1) position:

$$B_i J_1 = \begin{bmatrix} x & x & & \\ + & x & x & \\ & & x & x \\ & & & x \end{bmatrix} \quad . \tag{3.1}$$

$B_i J_1$ will now be pre- and postmultiplied by a sequence of Given's rotations whose purpose is to "chase the bulge" indicated by "+" off the end of the matrix. Choose θ_2 so that $J_2 \equiv J(1,2,\theta_2)$ introduces a zero in the (2,1) entry of $J_2 B_i J_1$:

$$J_2 B_i J_1 = \begin{bmatrix} x & x & + & \\ 0 & x & x & \\ & & x & x \\ & & & x \end{bmatrix} \quad . \tag{3.2}$$

Next choose θ_3 in $J_3 \equiv J(2,3,\theta_3)$, θ_4 in $J_4 \equiv J(2,3,\theta_4)$, θ_5 in $J_5 \equiv J(3,4,\theta_5)$, and θ_6 in $J_6 \equiv J(3,4,\theta_6)$, to give the following sequence of transformations:

$$J_2 B_i J_1 J_3 = \begin{bmatrix} x & x & 0 & \\ & x & x & \\ & + & x & x \\ & & & x \end{bmatrix} \qquad J_4 J_2 B_i J_1 J_3 = \begin{bmatrix} x & x & & \\ & x & x & + \\ & 0 & x & x \\ & & & x \end{bmatrix}$$

$$J_4 J_2 B_i J_1 J_3 J_5 = \begin{bmatrix} x & x & & \\ & x & x & 0 \\ & & x & x \\ & & + & x \end{bmatrix} \qquad B_{i+1} \equiv J_6 J_4 J_2 B_i J_1 J_3 J_5 = \begin{bmatrix} x & x & & \\ & x & x & \\ & & x & x \\ & & 0 & x \end{bmatrix}$$

The usual error analysis of Given's rotations [18, p. 131-139] shows that the computed B_{i+1} is the exact transformation of a matrix $B_i + E$ where E is on the order of $p(n)\varepsilon B_i$, $p(n)$ a modest function of n.

To choose θ_1 we compute a shift σ^2 which is generally the smallest eigenvalue of the bottom right 2 by 2 submatrix of $B_i B_i^T$. θ_1 is then chosen so J_1 introduces a zero into the (2,1) entry of $J_1^T(B_i^T B_i - \sigma^2 I)$. It is easy to see that this means that

$$\tan\theta_1 = \frac{(B_i^T B_i)_{12}}{\sigma^2 - (B_i^T B_i)_{11}} \quad . \tag{3.3}$$

This choice of shift, called Wilkinson's shift, guarantees at least linear convergence and generally yields asymptotic cubic convergence of the offdiagonal entries of B_i to zero [15, p. 151]. This is assuming arithmetic is done exactly.

Now let us take $\sigma = 0$. Let us also drop the subscript i on B_i for simplicity of notation. From (3.3) we see that $\tan\theta_1 = -b_{12}/b_{11}$ so that the result of the first rotation (for a 4 by 4 matrix) is

$$B^{(1)} \equiv B J_1 = \begin{bmatrix} b_{11}^{(1)} & 0 & & \\ b_{21}^{(1)} & b_{22}^{(1)} & b_{23} & \\ & & b_{33} & b_{34} \\ & & & b_{44} \end{bmatrix} \quad .$$

We let the superscript on the matrix and its entries indicate that J_1 has been applied. Comparing to (3.1) we see that the (1,2) entry is zero instead of nonzero. This zero will propagate through the rest of the algorithm and is the key to its effectiveness. After the rotation by J_2 we have

$$B^{(2)} \equiv J_2 B J_1 = \begin{bmatrix} b_{11}^{(2)} & b_{12}^{(2)} & b_{13}^{(2)} & \\ 0 & b_{22}^{(2)} & b_{23}^{(2)} & \\ & & b_{33} & b_{34} \\ & & & b_{44} \end{bmatrix} .$$

Note that

$$\begin{bmatrix} b_{12}^{(2)} & b_{13}^{(2)} \\ b_{22}^{(2)} & b_{23}^{(2)} \end{bmatrix} = \begin{bmatrix} \sin\theta_2 b_{22}^{(1)} & \sin\theta_2 b_{23} \\ \cos\theta_2 b_{22}^{(1)} & \cos\theta_2 b_{23} \end{bmatrix}$$

i.e. it is a rank one matrix. Therefore, postmultiplication by J_3 to zero out the (1,3) entry will also zero out the (2,3) entry:

$$B^{(3)} \equiv J_2 B J_1 J_3 = \begin{bmatrix} b_{11}^{(2)} & b_{12}^{(3)} & 0 & \\ 0 & b_{22}^{(3)} & 0 & \\ & b_{32}^{(3)} & b_{33}^{(3)} & b_{34} \\ & & & b_{44} \end{bmatrix} .$$

Comparing to (3.2) we see that there is an extra zero on the superdiagonal. Rotation by J_4 just repeats the situation: the submatrix of $J_4 J_2 B J_1 J_3$ consisting of rows 2 and 3 and columns 3 and 4 is rank one, and rotation by J_5 zeros out the (3,4) entry as well as the (2,4) entry. This regime repeats itself for the length of the matrix.

The following algorithm incorporates this observation. It uses a subroutine $ROT(f, g, cs, sn, r)$ which takes f and g as inputs and returns r, $cs = \cos\theta$ and $sn = \sin\theta$ such that

$$\begin{bmatrix} cs & sn \\ -sn & cs \end{bmatrix} \cdot \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} . \tag{3.4}$$

$ROT(f, g, cs, sn, r)$: takes f and g as input and returns cs, sn and r satisfying (3.4).

```
if (f = 0) then
    cs = 0; sn = 1; r = g
elseif (|f| > |g|) then
    t = g/f; tt = sqrt(1+t^2)
    cs = 1/tt; sn = t*cs; r = f*tt
else
    t = f/g; tt = sqrt(1+t^2)
    sn = 1/tt; cs = t*sn; r = g*tt
endif
```

Barring underflow and overflow (which can only occur if the true value of r itself would overflow), ROT computes cs, sn and r to nearly full machine accuracy (see section 8 below for details). It also uses fewer operations than the analogous routine "rotg" in LINPACK [5].

Implicit Zero-Shift QR Algorithm: Let B be an n by n bidiagonal matrix with diagonal entries $s_1, \ldots, s_n$ and superdiagonal entries $e_1, \ldots, e_{n-1}$. The following algorithm replaces s_i and e_i by new values corresponding to one step of the QR iteration with zero shift:

```
oldcs = 1
f = s_1
g = e_1
for i = 1, n-1
    call ROT( f, g, cs, sn, r )
    if (i≠ 1) e_{i-1} = oldsn*r
    f = oldcs*r
    g = s_{i+1}*sn
    h = s_{i+1}*cs
    call ROT( f, g, cs, sn, r )
    s_i = r
    f = h
    g = e_{i+1}
    oldcs = cs
    oldsn = sn
endfor
e_{n-1} = h*sn
s_n = h*cs
```

It is straightforward to verify that this algorithm "chases the bulge" in the manner described above. It is remarkable that outside the two calls to *ROT*, there are only 4 multiplications in the inner loop. This is to be contrasted with the usual QR algorithm, which in addition to two calls to *ROT* has 12 multiplications and 4 additions. Thus the inner loop is much more efficient than the standard algorithm. Note also that it is parallelizable, because $n/2$ rotations can be done at once. Since data need only be passed serially along the diagonal, it can also be implemented in a systolic array. However, the algorithms in section 6 seem much better suited to parallel processing.

This algorithm may be expressed in the following terser but equivalent form:

```
oldcs = 1
cs = 1
for i = 1, n-1
    call ROT( s_i*cs, e_i, cs, sn, r )
    if (i≠ 1) e_{i-1} = oldsn*r
    call ROT( oldcs*r, s_{i+1}*sn, oldcs, oldsn, s_i )
endfor
h = s_n*cs
e_{n-1} = h*oldsn
s_n = h*oldcs
```

The initial form will be more convenient for the error analysis in section 8.

This algorithm is also much more accurate than the standard algorithm. The source of the extra accuracy is the absence of possible cancellation, which means all roundoff errors appear multiplicatively (there is an addition in *ROT*, but it is harmless). Our model of arithmetic is the usual one:

$$fl(x \circ y) = (x \circ y)\cdot(1+e) \tag{3.5}$$

where $\circ$ is one of $+$, $-$, $*$ and $/$, $fl(x \circ y)$ is the floating point result of the operation $\circ$, and $|e| \leq \varepsilon$, where ε is the machine precision. This would appear to eliminate machines like the

Cray and Cyber from consideration, since those machines do not conform to (3.5) for addition and subtraction when cancellation is involved, but since we only need to use (3.5) for multiplication (as well as square root, division, and addition of positive quantities in *ROT*), this analysis covers those machines as well. We also assume overflow and underflow do not occur (we return to these issues in section 8).

We present two theorems about the accumulation of error in the algorithm. The proofs are given in section 8. The first theorem develops a bound for the relative error in the computed s_i and e_i of the form $cn\varepsilon$ (c is a modest constant) and uses it with Theorem 2 of section 2 to show that the relative difference between the singular values of the bidiagonal matrix B and the output matrix B' of the implicit zero-shift QR algorithm is $cn^2\varepsilon/(1-cn^2\varepsilon)$. In other words, the relative error in the computed singular values can only grow with the square of the dimension.

Theorem 6: Let B be an n by n bidiagonal matrix and B' the matrix obtained by running the implicit zero-shift QR algorithm on B. Let the singular values of B be $\sigma_1 \geq \cdots \geq \sigma_n$, and the singular values of B' be $\sigma_1' \geq \cdots \geq \sigma_n'$. Then if

$$\omega \equiv 69n^2\varepsilon < 1 \quad , \tag{3.6}$$

the relative differences between the singular values of B and the singular values of B' are bounded as follows:

$$|\sigma_i - \sigma_i'| \leq \frac{\omega}{1-\omega}\sigma_i \quad .$$

Let B_k be the matrix obtained after k repetitions of the implicit zero-shift QR algorithm, and let $\sigma_{k1} \geq \cdots \geq \sigma_{kn}$ be its singular values. Then if condition (3.6) holds we have

$$|\sigma_i - \sigma_{ki}| \leq \left(\frac{1}{(1-\omega)^k} - 1\right) \cdot \sigma_i \approx 69kn^2\varepsilon \cdot \sigma_i \quad ,$$

where the approximation to the last upper bound holds if $k\omega \ll 1$.

This result is actually rather pessimistic, as our second result shows: when we approach convergence in the sense that all rotations are through angles bounded away from $\pi/2$, errors do not accumulate at all and the error in the computed e_i and s_i is bounded by $c' \cdot \varepsilon$, c' another modest constant. With Theorem 2 this yields an error bound on the computed singular values of the form $c'n\varepsilon/(1-c'n\varepsilon)$.

Theorem 7: Let B be an n by n bidiagonal matrix and B' the matrix obtained by running the implicit zero-shift QR algorithm on B. Assume that all the rotation angles θ during the course of the algorithm satisfy $\sin^2\theta \leq \tau < 1$. Let the singular values of B be $\sigma_1 \geq \cdots \geq \sigma_n$, and the singular values of B' be $\sigma_1' \geq \cdots \geq \sigma_n'$. Then if

$$\omega \equiv \frac{88n\varepsilon}{(1-\tau)^2} < 1 \quad , \tag{3.7}$$

the relative differences between the singular values of B and the singular values of B' are bounded as follows:

$$|\sigma_i - \sigma_i'| \leq \frac{\omega}{1-\omega}\sigma_i \quad .$$

Let B_k be the matrix obtained after k repetitions of the implicit zero-shift QR algorithm, where we assume all rotation angles θ satisfy $\sin^2\theta \leq \tau < 1$. Let $\sigma_{k1} \geq \cdots \geq \sigma_{kn}$ be the singular values of B_k. Then if condition (3.7) holds we have

$$|\sigma_i - \sigma_{ki}| \leq \left(\frac{1}{(1-\omega)^k} - 1\right) \cdot \sigma_i \approx \frac{88kn\varepsilon}{(1-\tau)^2} \cdot \sigma_i \quad ,$$

where the approximation to the last upper bound holds if $k\omega \ll 1$.

Note that τ can easily be monitored by the algorithm as it proceeds.

The standard algorithm does not always achieve this accuracy for three reasons. First, the convergence criteria in the standard algorithm can change small singular values completely (this is discussed in detail in the next section). Second, rounding errors committed while "chasing the bulge" with a large shift can obscure small matrix entries and small singular values. Third, roundoff errors when the shift is zero result in nonzero entries appearing and propagating in those offdiagonal entries of intermediate results which should be zero, and which are kept zero by the new algorithm. This third effect seems mild, however, and as a result the standard algorithm sometimes computes small singular values with higher relative accuracy than the usual bound $p(n)\varepsilon A$ would lead us to expect (see, for example, the numerical examples of Class 1 in section 7).

The pattern of zeros above the diagonal during the QR sweep also appears when applying QR to a symmetric tridiagonal matrix. This pattern can be exploited to give fast, square root free versions of the algorithm (see [15, p. 164] for a discussion). Unfortunately, this does not yield forward stability and high accuracy as it does for the bidiagonal case.

Finally, we discuss the asymptotic convergence rate of the algorithm. It is well known that unshifted QR on a symmetric matrix is essentially the same as inverse iteration [15, p. 144]. Therefore we can conclude that the last offdiagonal element e_{n-1} should converge to zero linearly with constant factor $\sigma_{n-1}^2/\sigma_n^2$. If there is a cluster of m small singular values isolated from the remaining ones, e_{n-m} will converge to zero linearly with constant factor $\sigma_{n-m+1}^2/\sigma_{n-m}^2$.

4. Convergence Criteria

In this section we discuss convergence criteria for the new algorithm, and describe the practical version of the algorithm, which is a hybrid of the usual shifted QR and the implicit zero-shift QR. After showing that the LINPACK convergence criteria [5] are unsatisfactory, we restate the convergence criteria of section 2. The same analysis leading to the convergence criteria will lead to a criterion for switching from zero-shift QR to shifted QR without damaging any tiny singular values. The switching criterion depends on a user specifiable tolerance *tol* which is the desired relative accuracy in the singular values (*tol* should be less than 1 and greater than the machine precision ε). The resulting hybrid algorithm will therefore run about as fast as the standard algorithm on matrices without any tiny singular values. We will also discuss convergence criteria in the case where one is only interested in absolute precision in the singular values. Finally, we discuss the impact of underflow on the convergence criteria.

We begin by discussing the convergence criteria used in the current version of the algorithm [5], and explain why they are unsuitable for our algorithm. The code in LINPACK has two tests for setting entries of the bidiagonal matrix B to zero. Recall that $s_1, \ldots, s_n$ are the diagonal entries of B and $e_1, \ldots, e_{n-1}$ are the offdiagonal entries. The first test is

$$\text{if } (|e_i| + |e_{i-1}| + |s_i| = |e_i| + |e_{i-1}|), \quad \text{set } s_i \text{ to } 0 \tag{4.1}$$

This rather enigmatic looking test works as follows. If $|s_i| < .5 \cdot \varepsilon \cdot (|e_i| + |e_{i-1}|)$, the test will be satisfied and s_i set to zero. In other words, (4.1) is a way of asking whether one number is less than roundoff error in another number without needing to know the machine precision ε explicitly. The other convergence test is

$$\text{if } (|s_i| + |s_{i-1}| + |e_{i-1}| = |s_i| + |s_{i-1}|), \quad \text{set } e_{i-1} \text{ to } 0 \tag{4.2}$$

Both tests compare an entry x of B with its two nearest neighbors on the other diagonal, and set x to zero if it is negligible compared to those neighbors. One justification for these tests is that roundoff error during the rotations could make the matrix indistinguishable from one with a zero in x's position. Also, they clearly introduce errors no worse than $p(n)\varepsilon A$. (Both these tests may be unnecessarily slow for these purposes on machines where the quantities $|e_i| + |e_{i-1}| + |s_i|$, $|e_i| + |e_{i-1}|$, $|s_i| + |s_{i-1}| + |e_{i-1}|$ and $|s_i| + |s_{i-1}|$ are computed and compared in extended precision registers, where the effective ε is much tinier than in working precision.)

Both tests are unsatisfactory for our algorithm. Test (4.1) introduces a zero singular value where there was none before, so it is clearly unsatisfactory. The following example shows why (4.2) is also unsatisfactory. Suppose η is sufficiently small that in floating point arithmetic $1+\eta=1$. Consider the matrix

$$B(x) = \begin{bmatrix} \eta^2 & 1 & & \\ & 1 & x & \\ & & 1 & 1 \\ & & & \eta^2 \end{bmatrix}$$

When $x=\eta$ it is easy to verify that the smallest singular value of $B(\eta)$ is about η^3. Test (4.2) would set x to 0, but $B(0)$ has a smallest singular value of about $\eta^2/\sqrt{2}$, which is utterly different.

Our convergence criteria must guarantee that by setting some e_i to 0 (clearly no nonzero s_i can ever be set to zero), no singular value is perturbed too much. Let $\bar{\sigma}$ denote a reliable estimate or underestimate of the smallest singular value. Such a $\bar{\sigma}$ is provided by the recurrences for $B^{-1}{}_{\infty}^{-1}$ and $B^{-1}{}_{1}^{-1}$ in (2.4). Then the simplest acceptable convergence criterion would only set e_i to zero if it were less than $tol*\bar{\sigma}$. However, this method is overly conservative, and generally waits much too long to set e_i to 0. Much better estimates are given in section 2 and justified by Theorems 4 and 5. We repeat them here:

Convergence Criterion 1a:

Let μ_j be computed by the following recurrence ((2.4) from section 2):

$$\begin{aligned} &\mu_1 = |s_1| \\ &\text{for } j=1 \text{ to } n-1 \text{ do} \\ &\quad \mu_{j+1} = |s_{j+1}|\cdot(\mu_j/(\mu_j + |e_j|)) \end{aligned} \tag{4.3}$$

If $|e_j/\mu_j| \leq tol$, set e_j to 0.

Convergence Criterion 1b:

Let λ_j be computed by the following recurrence ((2.4) from section 2):

$$\begin{aligned} &\lambda_n = |s_n| \\ &\text{for } j=n-1 \text{ to } 1 \text{ step } -1 \text{ do} \\ &\quad \lambda_j = |s_j|\cdot(\lambda_{j+1}/(\lambda_{j+1} + |e_j|)) \end{aligned} \tag{4.4}$$

If $|e_j/\lambda_{j+1}| \leq tol$, set e_j to 0.

Convergence Criterion 2a:

Let μ_j be computed from (4.3). If singular vectors are not desired, and $e_{n-1}^2 \leq .5\cdot tol\cdot[(\min_{j<n} \mu_j/(n-1)^{1/2})^2 - |s_n|^2]$, set e_{n-1} to zero.

Convergence Criterion 2b:

Let λ_j be computed from (4.4). If singular vectors are not desired, and $e_1^2 \leq .5\cdot tol\cdot[(\min_{j>1} \lambda_j/(n-1)^{1/2})^2 - |s_1|^2)$, set e_1 to zero.

We have divided the criteria of section 2 into separate parts, because we will apply them in separate situations; see the subsection on "Applying the convergence criteria" in section 5.

These criteria are more expensive than the standard LINPACK criteria, but avoid situations like setting x to 0 in $B(x)$ in the last paragraph, and recognizes that setting x to zero in

$$\begin{bmatrix} 1 & x \\ & D \end{bmatrix}$$

is harmless if $|x| \leq tol$, independent of D.

Now we consider how to decide whether to use implicit zero-shift QR or standard shifted QR. In order to estimate the rounding errors which would occur during shifted QR, we need an estimate of B. We will use $\overline{\sigma} \equiv \max_i(|s_i|, |e_i|)$, which is easily seen to underestimate B by no more than a factor of 2. In terms of $\underline{\sigma}$, $\overline{\sigma}$, and *tol*, our decision algorithm is

```
if (fudge * tol * (σ/σ̄) ≤ ε)
   use the implicit zero-shift QR
else
   use shifted QR
endif
```

The test asks whether the rounding errors $\varepsilon\cdot\overline{\sigma}$ which shifted QR could introduce are greater than the largest tolerable perturbation $tol\cdot\underline{\sigma}$. The factor $fudge \geq 1$ is a fudge factor which makes zero-shifting less likely on tight clusters of singular values; we currently use $fudge = \min(n, m)$ if the original matrix was n by m.

In practice there is one further test for using the implicit zero-shift QR. If the above test chooses shifted QR, we must still compute the shift σ^2, which is the smallest eigenvalue of the

bottom 2 by 2 matrix of BB^T. From (3.3), we see the tangent of the first rotation angle is given by

$$\frac{s_1 * e_1}{\sigma^2 - s_1^2} = \frac{e_1}{s_1} \left(\frac{\sigma^2}{s_1^2} - 1\right)^{-1}$$

so if $\sigma^2/s_1^2 - 1$ rounds to -1, the first rotation is the same as in implicit zero-shift QR and we might as well use it, since it is faster and more accurate.

The choice of *tol* may be made by the user, or chosen automatically by the program. If *tol* is chosen close to 1, one almost always picks shifted QR, which still computes the singular values with good absolute accuracy, so only the smallest singular values will be inaccurate. If one chooses *tol* near ε, one will almost always use implicit zero-shift QR unless all the singular values are very close together, and therefore sacrifice the cubic convergence of shifted QR. See section 7 for descriptions of numerical experiments on the effect of varying ε. Choosing *tol* near 1 is useful for quickly obtaining estimates of singular values for rank determination. Note that as singular values converge and are deflated off, $\underline{\sigma}$ may be reestimated so that if *tol* is not too small, by the time all the small singular values have converged, the algorithm is doing shifted QR.

Note also that if one is only interested in computing the smallest singular value or values, $\underline{\sigma}$ provides a test for stopping the iteration early. If one or several small singular values have been deflated out, and the $\underline{\sigma}$ for the remaining matrix exceeds them sufficiently, one is guaranteed to have found the smallest singular value. A similar idea is expressed in [17].

Finally, we consider computing the singular values to guaranteed absolute accuracy instead of guaranteed relative accuracy. As stated in the introduction, this is what standard shifted QR guarantees. However, the convergence criteria (4.1) and (4.2) in the current standard implementation are much more stringent than necessary to meet this goal. Instead of comparing $|e_i|$ or $|s_i|$ to its neighbors to see if it is negligible, it is only necessary to compare to $\overline{\sigma} \approx \|B\|$. In other words substituting

$$\text{if } (|s_i| \leq tol * \overline{\sigma}) \text{ set } s_i \text{ to } 0 \tag{4.5}$$

for (4.1) and

$$\text{if } (|e_i| \leq tol * \overline{\sigma}) \text{ set } e_i \text{ to } 0 \tag{4.6}$$

for (4.2) will also guarantee absolute accuracy but possibly speed convergence considerably. In practice, our code uses the input parameter *tol* to choose between absolute and relative accuracy: if *tol* is positive, it indicates that relative accuracy *tol* is desired, and if *tol* is negative, it indicates that absolute accuracy $|tol| \cdot \overline{\sigma}$ is desired.

Underflow must also be accounted for in the convergence criteria to ensure convergence. For it may happen that the quantity to be subtracted from e_{n-1} in the course of driving it to zero may underflow, so that e_{n-1} never decreases. On machines with IEEE arithmetic, this may occur if all entries of B are denormalized. To prevent this, we make sure the convergence threshold to which we compare $|e_j|$ is at least $maxit * \lambda$, where *maxit* is the maximum number of QR inner loops the code will perform, and λ is the underflow threshold (the smallest positive normalized number). If the matrix has singular values near λ or smaller, this technique could destroy their accuracy; in this case the matrix should be scaled up by multiplying it by $maxit/\varepsilon$ before applying the algorithm, and multiplying the computed singular values by $\varepsilon/maxit$ afterwards.

5. Implementation Details

In this section we discuss a number of details of the implementation of the code:

Chasing the bulge up or down
Applying the convergence criteria
SVD of 2 by 2 triangular matrices and robust shift calculation
Deflation when $s_i = 0$

Finally, we present high-level code for the entire algorithm.

Chasing the bulge up or down. A bidiagonal matrix may be graded in many ways, but most commonly it will be large at one end and small at the other. The implicit zero-shift QR algorithm tries to converge the singular values in order from smallest to largest. If the matrix is graded from large at the upper left to small at the lower right, and the "bulge" is chased from upper left to lower right as in section 3, then convergence will be fast because the singular values are "ordered" correctly, i.e. the diagonal matrix entries are fairly close to their final values. If, however, the matrix is graded the opposite way (from small at the left to large at the right) then the algorithm will have to invert the order of the matrix entries as it converges. This may require many more QR steps. To avoid this, the implementation tests for the direction of grading (simply comparing $|s_1|$ and $|s_n|$), and chases the bulge in the direction from large to small. If a matrix breaks up into diagonal blocks which are graded in different ways, the bulge is chased in the appropriate direction on each block. The algorithm in [17] does this as well.

In order to avoid the possibility that the code might frequently change bulge chasing directions, and so converge very slowly, we only choose the direction of bulge chasing when beginning work on a submatrix disjoint from the previous one. Whether this is the optimal strategy is a question of future research.

This means the singular values may be quite disordered in the final converged matrix, and so must be sorted at the end (along with the singular vectors if desired). The LINPACK SVD uses bubble sort at the end, which could require $O(n^2)$ swaps of singular vectors. Since the LINPACK SVD always chases the bulge down, the singular values tend to converge in nearly sorted order, so bubble sort is relatively efficient. The new algorithm, in which the singular values could converge in any order, uses insertion sort instead, which does at most $2n$ moves of singular vectors.

Applying the convergence criteria. In section 4 we presented four convergence criteria. Since applying the convergence criteria costs approximately as many floating point operations ($O(n)$) as performing a QR sweep, it is important to test criteria only when they are likely to be satisfied. Our decision is based on the following empirical observation: When chasing the bulge down (up), the bottommost (topmost) entry s_n (s_1) often tends to converge to the smallest singular value, with e_{n-1} (e_1) tending to zero fastest of all offdiagonal entries. Therefore, when chasing the bulge down, we expect convergence criteria *1a* and *2a* to be successful, and possibly *1b* but only for the bottommost entry e_{n-1}. Criteria *2b* and *1b* for the other off diagonal entries are not as likely to succeed. Conversely, when chasing the bulge up, we only apply convergence criteria *1b*, *2b* and *1a* for e_1. One advantage of this scheme is that testing *2a* (for e_{n-1}, and if the test succeeds, for e_{n-2} too) costs only a few more operations after testing *1a*, since they share the same recurrence from (4.3). Similarly, *2b* (for e_1, and if the test succeeds, for e_2 too) is very cheap after applying *1b*.

SVD of 2 by 2 triangular matrices and robust shift calculation. The need for the singular value decomposition of 2 by 2 triangular matrices, or at least the smallest singular value of such a matrix, arises in two places in the code. The first time is when calculating the shift. As stated in section 3, the standard choice of shift, called Wilkinson's shift, is the smallest eigenvalue of the bottom 2 by 2 block of BB^T. It is easy to see that this is the square of the smallest singular value of the bottom 2 by 2 block of B. The second need for the SVD of a 2 by 2 triangular

matrix arises when the code has isolated a 2 by 2 block on the diagonal of B. Even though this appears to be an easy case for the algorithm in section 4, it turns out that roundoff can prevent convergence when the singular values are close. This is the case in

$$B = \begin{bmatrix} a & b \\ 0 & c \end{bmatrix}$$

when $|a|$ and $|c|$ are close and b is much smaller, just larger than $\varepsilon \cdot |a|$. It happens that on machines with sloppy arithmetic, roundoff can cause b to be no smaller after one step of QR than before, so that the algorithm never converges. It is also difficult in this situation to compute the singular vectors accurately, just as eigenvectors corresponding to multiple eigenvalues are difficult to compute.

To get around these difficulties, we have written a subroutine which takes the entries a, b and c of B and returns the two singular values as well as the left and right singular vectors. Barring overflow and underflow, the returned values are accurate to nearly full machine precision, even for nearly coincident singular values. The algorithm is comparable in speed to a straightforward implementation that does not attain similar accuracy. This property is based on the fact that the algorithm uses formulas for the answer which contain only

products, quotients, and square roots,
sums of terms of like sign,
differences of computed quantities only when cancellation is impossible, and
the difference $|a| - |c|$, which, if cancellation occurs, is exact (except possibly on a Cray or Cyber).

It is straightforward to use these properties to show that the final result is correct to nearly full precision.

The code is also robust in the face of over/underflow. Overflow is avoided where possible by using formulas in terms of ratios of matrix entries, and choosing the formulas so that the ratios are always bounded by 1 in magnitude. As a result of these precautions, overflow is impossible unless the exact largest singular value itself overflows (or is within a few units in the last place of overflowing). Underflow (of the conventional "store zero" variety) can damage the results only if the data and/or results are themselves close to the underflow threshold, specifically less than the underflow threshold divided by ε. Gradual underflow [2] makes the calculation of the singular values impervious to underflow (unless the final results themselves underflow) and the singular vectors much less susceptible to underflow problems.

Deflation when $s_i = 0$. The standard SVD algorithm [5] has special code to handle the case when $s_i = 0$. This code does a simplified sequence of rotations (similar to implicit zero-shift QR) to introduce a zero on the superdiagonal of the bidiagonal matrix (adjacent to the zero on the diagonal) and so break it into two smaller problems. It is easy to see that the implicit zero-shift QR algorithm does this deflation automatically, yielding one zero on the superdiagonal for each zero on the diagonal, but at the bottom (or top) of the matrix, rather than where the original zero occurred. This occurs after one pass of the algorithm, at which point both s_n and e_{n-1} will be zero if chasing the bulge down (s_1 and e_1 will be zero if chasing the bulge up) meaning that the zero singular value has been deflated exactly.

We can see this as follows. Assume we are chasing the bulge down. Whenever $s_{i+1} = 0$, both g and h will be set to 0, causing the sn returned by the second call to ROT to be 0. At the end of the loop, both $f = h$ and $oldsn = sn$ will also be zero. In fact, it is easy to see that from now on both h and the f at the bottom of the loop will be zero: at the top of the next loop iteration, the zero value of f causes the first call of ROT to compute $cs = 0$; this causes $h = s_{i+1} * cs$ to be zero and the pattern repeats. Also, when $oldsn = 0$ (which happens when $s_{i+1} = 0$), e_{i-1} is set to zero on the next iteration, i.e. $s_{i+1} = 0$ implies e_i becomes zero. Finally, at the end of all the loop iterations, h is still zero implying both e_{n-1} and s_n are set to zero. Note that when f is zero, as it frequently is in this case, the first call to ROT need only set $cs = 0$, $sn = 1$ and $r = g$; this is what the first "if" branch in ROT does.

Finally, we present a high-level description of the entire algorithm. In the interest of brevity we omit the code for updating the singular vectors or for the absolute error convergence criterion.

ε = machine precision
λ = underflow threshold (smallest positive normalized number)
n = dimension of the matrix
tol = relative error tolerance (currently 100ε)
maxit = maximum number of QR inner loops (currently $3n^2$)

Bidiagonal Singular Value Decomposition

```
Compute σ ≤ σ_min(B) using (2.4)
σ̄ = max(|s_i|, |e_i|)
thresh = max(tol·σ̄, maxit·λ)
/* any e_i less than thresh in magnitude may be set to zero */

Loop:
    /* Find bottommost nonscalar unreduced block diagonal submatrix of B */
    let ī be the smallest i such that |e_i| through |e_{n-1}| are at most thresh, or n if no such i exists
    if ī = 1, goto Done
    let i' be the largest i less than ī such that |e_i| ≤ thresh, or 0 if no such i exists
    i̲ = i'+1

    /* Apply algorithm to unreduced block diagonal submatrix from i̲ to ī */
    if ī = i̲+1, then
        /* 2 by 2 submatrix, handle specially */
        compute SVD of 2 by 2 submatrix, setting e_i̲ to 0
        goto Loop
    endif

    if submatrix from i̲ to ī disjoint from submatrix of last pass through Loop, then
        /* Choose bulge chasing direction */
        if |s_i̲| ≥ |s_ī|, then
            direction = "down"
        else
            direction = "up"
        endif
    endif
```

```
    /* Apply convergence criteria */
    if direction = "down", then
        Apply convergence criterion 1b to e_{i-1}
        Apply convergence criterion 1a
        Apply convergence criterion 2a to e_{i-1} and possibly e_{i-2}
    else
        Apply convergence criterion 1a to e_i
        Apply convergence criterion 1b
        Apply convergence criterion 2b to e_i and possibly e_{i+1}
    endif

    /* Compute shift */
    if fudge*tol*σ/σ̄ ≤ ε, then
        /* Use zero because tiny singular values present */
        shift = 0
    else
        if direction = "down", then
            s = s_i
            shift = smallest singular value of bottom 2 by 2 corner
        else
            s = s_i
            shift = smallest singular value of top 2 by 2 corner
        endif
        if (shift/s)^2 ≤ eps, then
            /* Use zero shift, since shift rounds to 0 */
            shift = 0
        endif
    endif

    /* Perform QR iteration */
    if shift = 0, then
        if direction = "down", then
            do implicit zero-shift QR downward
            if |e_{i-1}| ≤ thresh, set e_{i-1} = 0
        else
            do implicit zero-shift QR upward
            if |e_i| ≤ thresh, set e_i = 0
        endif
    else
        if direction = "down", then
            do standard shifted QR downward
            if |e_{i-1}| ≤ thresh, set e_{i-1} = 0
        else
            do standard shifted QR upward
            if |e_i| ≤ thresh, set e_i = 0
        endif
    endif
    goto Loop

Done: sort singular values
```

6. Other Methods for Computing Accurate Singular Values

In this section we discuss other methods for computing the singular values of a bidiagonal matrix B to high relative accuracy. These methods include bisection, Rayleigh Quotient Iteration, and iterative refinement. They are not competitive in speed with QR for computing all the singular values on a serial machine, but can efficiently verify whether a computed singular value is accurate or not. We have used it to verify the numerical results presented in section 7. However, they are extremely easy to parallelize and will probably be among the best parallel algorithms for this problem.

All the algorithms are based on bisection for the symmetric tridiagonal eigenproblem, which we discuss first. Bisection is in turn based on Sylvester's Law of Inertia, or equivalently, Sturm sequences [15, p. 52]. As explained in section 2, the number of negative d_i in recurrence (2.1) is the number of eigenvalues less than x, a quantity we will denote by $\nu(x)$. $\nu(x_2)-\nu(x_1)$ is therefore the number of eigenvalues in the interval $[x_1,x_2)$, so this method can easily verify whether an interval contains any eigenvalues. The identification of the singular value problem for the bidiagonal matrix B with the eigenvalue problem for a symmetric tridiagonal matrix with zero diagonal later in section 2 makes it clear that we can use the same method to count the number of singular values in any interval $[x_1,x_2)$.

What remains is an error analysis to show that the function $\nu(x)$ is accurate. This is provided in [12, p.35]:

> Let $\nu(x)$ be the computed number of eigenvalues less than x for a symmetric tridiagonal matrix A. Barring over/underflow, the computed value of $\nu(x)$ is the exact value of $\nu(x)$ for a perturbed matrix $A+\delta A$ where $|\delta A| \le 2\varepsilon|\mathrm{offdiag}(A)| + \varepsilon x I$. Here, offdiag($A$) refers to the offdiagonal part of A. If A has a zero diagonal, this bound may be improved to $|\delta A| \le 1.5\cdot\varepsilon|A|$.

Therefore, by Theorem 2 or Corollary 2, if the computed value of $\nu(x)$ is k, there must be at least k singular values of B less than $x/(1-(3n-1.5)\varepsilon)$ and no more than k singular values less than $x\cdot(1-(6n-2)\varepsilon)/(1-(3n-1.5)\varepsilon)$; we assume $n\varepsilon 1$. If the computed value of $\nu(x_2)-\nu(x_1)$ is j, there must be at least j singular values in the interval $[x_1\cdot(1-(6n-2)\varepsilon)/(1-(3n-1.5)\varepsilon\ ,\ x_2/(1-(3n-1.5)\varepsilon))$.

There is one other important feature of the computed $\nu(x)$. In exact arithmetic, since $\nu(x)$ is the number of eigenvalues less than x, $\nu(x)$ must be a monotonic increasing function of x. It is by no means clear that the computed values of $\nu(x)$ should also be monotonic. This is significant because a failure in monotonicity could cause an algorithm to misestimate the number of eigenvalues in an interval, although a bisection routine which begins with an interval $[x_1,x_2)$ where $\nu(x_2)-\nu(x_1)$ is positive can always maintain an interval over which the computed value of ν increases. It turns out, however, that as long as the arithmetic is monotonic, the computed value of $\nu(x)$ will be monotonic [12, p. 27]. By monotonic arithmetic we mean that if $a\circ b \ge c\circ d$ in exact arithmetic, then $fl(a\circ b)\ge fl(c\circ d)$ as well. This holds in any well-designed arithmetic, such as the IEEE Floating Point Standard 754 [10]. We have only shown monotonicity holds if the recurrence is computed exactly as follows, with the order of evaluation respecting parentheses:

$$d_i = (a_i - (b_{i-1}^2/d_{i-1})) - x \quad .$$

Now we briefly consider Rayleigh Quotient Iteration and iterative refinement. Both algorithms begin with a small interval containing a singular value, and refine it as does bisection. The major difference from bisection is in the zerofinder used to refine the intervals. As long as the zerofinders are implemented in a componentwise backward stable way (i.e. they compute the correct result for a bidiagonal having only small relative perturbations in each entry), then Corollary 2 and Theorem 2 guarantee the relative accuracy of the computed singular values.

7. Numerical experiments

The numerical experiments we discuss here compare the algorithm of sections 3 through 5 with the LINPACK SVD [5]. Both codes were run in double precision on a SUN 3/60 with an MC68881 floating point coprocessor, which implements IEEE standard 754 floating point arithmetic [10]; the machine precision $\varepsilon = 2^{-53} \approx 1.1 \cdot 10^{-16}$ and the range is approximately $10^{\pm 308}$. In order to guarantee reliable timings, each matrix tested was run sufficiently often that the total elapsed time was about 10 seconds. Singular vectors were computed by identical calls to *drot* [5] in both algorithms.

The codes were compared with respect to

accuracy,
total number of passes through the inner loop of QR iteration,
(half the number of Givens rotations performed)
elapsed time when computing singular values only,
elapsed time when computing both left and right singular vectors as well,
elapsed time including bidiagonalizing the input matrix, and
elapsed time excluding bidiagonalizing the input matrix.

Also, the dependence of the new algorithm on the parameter *tol* (see section 4) was investigated. At the end we comment on the implications of our results for the "perfect shift" strategy for computing singular vectors.

The LINPACK code was modified to explicitly use the machine precision ε in the stopping criteria rather than implicitly as in (4.1) and (4.2). Specifically,

$$\text{if } (|s_i| \le \varepsilon * (|e_i| + |e_{i-1}|)), \quad \text{set } s_i \text{ to } 0 \tag{7.1}$$

was used in place of (4.1) and

$$\text{if } (|e_{i-1}| \le \varepsilon * (|s_i| + |s_{i-1}|)), \quad \text{set } e_{i-1} \text{ to } 0 \tag{7.2}$$

was used in place of (4.2). Thus, since both the new algorithm and modified LINPACK code use stopping criteria with ε appearing explicitly, there is no danger that the extended precision registers on the MC68881 would cause tests like (4.1) and (4.2) to be executed with a smaller effective ε than expected, which could slow convergence.

The LINPACK code also used a corrected shift calculation rather than the erroneous one in [5]. The version in [5] computes $f = (sl + sm) * (sl - sm) - shift$; this should be $f = (sl + sm) * (sl - sm) + shift$ instead (the corrected version is distributed by netlib [4]).

It turns out that the results depend strongly on the form of the bidiagonal matrix. For example, the standard SVD behaves entirely differently on matrices graded from top to bottom than on matrices graded in the opposite direction. Therefore, we present our results on 12 separate classes of bidiagonal matrices, since this seems to be the only fair way to compare results. The classes are as follows:

Class 1:
These 8 matrices are graded in the usual way from large at the upper left to small at the lower right. All matrices have a 1 in the upper corner, and each superdiagonal entry $B_{i,i+1}$ equals its neighbor B_{ii} on the diagonal. Four of the matrices are 10 by 10 and have a constant multiple between adjacent entries on the diagonal and superdiagonal: 10^{10}, 10^5, 10^2 and 10. The other four are 20 by 20 and are obtained from the first four by simply repeating each entry once, e.g. a diagonal containing 1, 10^{-10}, 10^{-20}, ... , 10^{-90} becomes 1, 1, 10^{-10}, 10^{-10}, 10^{-20}, 10^{-20}, ... , 10^{-90}, 10^{-90}.

Class 2:
This class is identical to class 1 except the order of the entries on the diagonal and superdiagonal are reversed. Thus these matrices are graded from small at the upper left corner to large at the lower right.

Class 3:
These eight 20 by 20 and 40 by 40 matrices are obtained by abutting those in class 1 with

their reversals in class 2. Thus each matrix is small at the upper left, large in the middle, and small again at the lower right.

Class 4:

These eight 20 by 20 and 40 by 40 matrices are obtained by abutting those in class 2 with their reversals in class 1. Thus each matrix is large at the upper left, small in the middle, and large again at the lower right.

Class 5:

These 8 matrices are obtained from class 1 by reversing the order of the superdiagonals. Thus the diagonal is graded from large at the upper left to small at the lower right, and the superdiagonal is graded in the opposite direction.

Class 6:

These 8 matrices are obtained from class 5 by reversing the order of both the diagonals and superdiagonals. Thus the diagonal is graded from small at the upper left to large at the lower right, and the superdiagonal is graded in the opposite direction.

Class 7:

These 16 matrices are all small on the diagonal and mostly large on the offdiagonal. Eight of them are 10 by 10 with 1's on the off diagonal and a constant diagonal, equaling 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, 10^{-12}, 10^{-14}, and 10^{-16}, respectively. The other eight 20 by 20 matrices are obtained by putting two copies of each of the first eight together, and "connecting" them by setting the middle offdiagonal entry $B_{10,11}$ to be 10^{-15} times the value of the diagonal entries.

Classes 8-11:

The ten 20 by 20 matrices in each class are generated by letting each bidiagonal entry be a random number of the form $r \cdot 10^{i}$, where r is a random number uniformly distributed between $-.5$ and .5, and i is a random integer. In class 8, i is uniformly distributed from 0 to -15. In class 9, i is uniformly distributed from 0 to -10. In class 10, i is uniformly distributed from 0 to -5. In class 11, i is identically 0. Thus, in class 11 each matrix entry is simply uniformly distributed on $[-.5, .5]$.

Class 12:

This one 41 by 41 matrix is graded in as in class 1, with the ratio of adjacent entries being $10^{-.1} \approx .79$. Each offdiagonal entry is identical to the diagonal entry below it. This very dense grading leads to different convergence properties than for the matrices in class 1, which is why we put this example in a separate class.

Thus classes 1-6 and 12 consist of graded matrices, class 7 consists of matrices larger on the offdiagonal than the diagonal, and classes 8-11 consist of random matrices with random exponents.

First we discuss the accuracy of computed singular values. With $tol = 100\varepsilon \approx 10^{-14}$ the new algorithm always converged in fewer than $maxit = 3n^2$ passes through the QR inner loop and computed all singular values to nearly full accuracy. Accuracy was determined using the method in section 6: If σ is a computed singular value, the number of singular values in the interval $[\sigma(1-n\varepsilon)\ ,\ \sigma(1+n\varepsilon))$ were counted. Overlapping intervals were joined into larger intervals. The number of computed singular values in each interval was then compared with the true number of singular values in each interval. This accuracy test was passed in all cases but one singular value out of 2041 singular values of all 105 matrices. In other words, 2040 singular values were computed with a relative error of about 10^{-14} or better; the exceptional singular value (in class 11) had a relative error a little less than $3 \cdot 10^{-14}$.

The accuracy of the singular values computed by the LINPACK SVD were determined by comparison with the singular values from the new algorithm. This data is presented in Table 1. We called agreement to at least 14 digits with the verified correct results of the new algorithm "all digits correct"; the notation "% all digits" in Table 1 means the percentage of such singular values. The notation "% $m-n$ digits" in Table 1 means the percentage of singular values computed with m to n correct digits. 0 digits means that the order of magnitude is still correct. -1

digits means correct to within a factor of 10. The column "% nonzero, no digits" gives the percentage of computed singular values which were nonzero and had incorrect orders of magnitude. The column "% zero, no digits" gives the percentage of computed singular values which were exactly zero, even though the matrix was nonsingular. The * in row 4 indicates that the algorithm did not converge for one of the test matrices (this matrix was not counted in computing the percentages).

Table 1: Accuracy of Singular Values from LINPACK SVD

Class	% all digits	% 12-14 digits	% 8-12 digits	% 4-8 digits	% 0-4 digits	% -1 digits	% nonzero, no digits	% zero, no digits
1	100	0	0	0	0	0	0	0
2	42	14	11	0	29	1	0	3
3	99.5	.5	0	0	0	0	0	0
4*	59	11	4	2	21	1	0	2
5	94	0	0	0	0	0	0	6
6	94	0	0	0	0	0	0	6
7	90	1	.5	0	.5	0	1	7
8	80.5	4.5	5	3	1	.5	.5	5
9	80	6.5	7	2.5	1	0	0	3
10	91	4.5	3	1.5	0	0	0	0
11	98	2	0	0	0	0	0	0
12	100	0	0	0	0	0	0	0

One striking feature about this table is the difference between classes 1 and 2. The only difference between the matrices in classes 1 and 2 is the order of the entries. When the entries are graded from large to small, the standard SVD gets all the singular values correct. Indeed, it was constructed to perform well on matrices graded in this fashion. When they are graded in the opposite way, only 42% are fully correct and another third have fewer than 4 digits correct. 3% are computed as 0 even though all matrices tested were nonsingular. This happens because the standard SVD always "chases the bulge" from top to bottom. When the matrix is graded from large to small, this works well, but when it is graded in the opposite way as in class 2, the algorithm must "reorder" all the matrix entries, and in doing so must combine tiny entries with large entries, thereby losing precision. The same thing happens for class 4. The new algorithm avoids the need to reorder by always "chasing the bulge" from the large to the small end of the matrix. This is also done in the algorithm in [17]; see section 5 for details. The nonzero singular values which are not even order of magnitude correct are off by factors of 10^{-53} and 10^{-57} (class 7) and 10^{-5} (class 8). The last column indicates how often the computed singular values were exactly zero, when in fact none of the test matrices were singular.

We evaluated the computed singular vectors by computing the norm of the residual $BV - U\Sigma$, where B is the bidiagonal matrix, V contains the right singular vectors, U the left singular vectors, and Σ the singular values. The norm was the maximum absolute matrix entry. In all cases for both new and old SVD this measure never exceeded $1.1 \cdot 10^{-14} \approx 100\varepsilon$, which is quite good and as expected from both algorithms (it is easy to show the convergence criteria for both algorithms leave the residual near the roundoff level). We do not yet have a complete perturbation theory or better accuracy tests for the singular vectors; see section 9 for further discussion.

Table 2 provides a measure of the difficulty of the different problem classes which is independent of matrix dimension. The usual rule of thumb for the number of QR sweeps it takes to compute the SVD is two sweeps per singular value [15, p. 165]. If convergence always takes place at the end of the matrix, this means there will be 2 sweeps on a matrix of length i, for $i = n, n-1, ..., 3$ (two by two matrices are handled specially). Here, n is the dimension of the original matrix. Thus, counting one QR sweep on a matrix of length i as i "QR inner loops," we expect an average of about $n(n+1)$ "QR inner loops" for the entire SVD. Thus, the quantity

"QR inner loops" divided by $n(n+1)/2$ should be a measure of the difficulty of computing the SVD of a matrix which is independent of dimension, and we expect it to equal 2 on the average. For each of the twelve problem classes, and for the three algorithms old SVD (LINPACK), new SVD without singular vectors, and new SVD with singular vectors, the minimum, average and maximum of the quantity "QR inner loops" divided by $n(n+1)/2$ are given in Table 2. Recall that we use different convergence criteria depending on whether or not we compute singular vectors, which is why we have different columns for these two cases.

Table 2: QR Inner Loops / $(n(n+1)/2)$ for Old and New SVD Algorithms with $tol = 100\varepsilon \approx 10^{-14}$

Class	Old SVD			New SVD without vectors			New SVD with vectors		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	.60	.90	1.33	.09	.36	.91	.09	.49	1.11
2	.60	1.94	3.07	.09	.36	.91	.09	.49	1.11
3	.61	.85	1.19	.56	.82	1.19	.56	.82	1.19
4	.32	1.04	1.80	.31	.58	1.00	.35	.60	1.04
5	.07	.45	1.11	.09	.54	1.29	.09	.57	1.42
6	.07	.40	.93	.09	.54	1.29	.09	.57	1.42
7	.10	1.32	2.31	.10	1.04	1.85	.10	1.04	1.85
8	.41	.64	.95	.25	.47	.75	.26	.49	.77
9	.79	.94	1.29	.51	.73	.89	.57	.75	.93
10	1.07	1.29	1.57	.98	1.19	1.47	1.04	1.22	1.48
11	1.97	2.26	2.52	2.07	2.20	2.38	2.06	2.20	2.41
12	1.53	1.53	1.53	2.96	2.96	2.96	2.96	2.96	2.96

It is interesting to note in Table 2 that only in class 11 is our expectation of 2 QR sweeps per singular value for the standard SVD nearly fulfilled. Recall that class 11 has matrices all of whose entries are uniformly distributed between $\pm .5$. Otherwise, either the average is much lower or there is a great variability in the number of QR sweeps needed (class 2). The same comments hold for the new algorithm, except for class 12 which was chosen to make the new algorithm look as bad as possible. Even so, it is within a factor of two of the old algorithm.

Table 3 gives timing comparisons between the old and new algorithms. The results depend on whether singular vectors are computed (Job= v in Table 3) or not (Job= nv). There were several statistics collected. First, the number of QR inner loops for each algorithm was counted, and the ratio of QR inner loops for the new algorithm to QR inner loops for the old algorithm computed; these statistics (minimum, average and maximum ratios, the same for the other statistics) are shown in columns 3-5 of Table 3. The timings also depend on whether we count the time to bidiagonalize or not. The time to bidiagonalize is quite large and can swamp the second, iterative part. Therefore we computed timing ratios (new algorithm to old algorithm) both with and without the initial bidiagonalization. The identical bidiagonalization code was used for the old and new algorithms. We performed the bidiagonalization part of the algorithm on a different, dense matrix, so that the algorithm and floating point hardware would not recognize they were dealing with a bidiagonal input matrix and so bypass some of the work. Columns 6-8 of Table 3 include the bidiagonalization phase, and columns 9-11 exclude it.

Table 3: Timing Comparisons of Old and New SVD Algorithms with $tol = 100\varepsilon \approx 10^{-14}$

Class	Job	Ratio of Inner Loops New SVD / Old Svd			Ratio of Times (with bidiagonalization) New SVD / Old SVD			Ratio of Times (without bidiagonalization) New SVD / Old SVD		
		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	nv	.15	.37	.77	.69	.77	.85	.29	.41	.63
	v	.15	.48	.84	.67	.75	.87	.26	.49	.77
2	nv	.10	.18	.34	.37	.58	.82	.14	.23	.37
	v	.10	.23	.36	.37	.51	.66	.13	.26	.34
3	nv	.86	.96	1.02	.91	.94	.96	.73	.77	.80
	v	.86	.96	1.03	.95	.97	1.01	.88	.95	1.03
4	nv	.44	.60	.99	.72	.85	1.02	.40	.58	1.15
	v	.44	.63	1.10	.69	.78	1.03	.46	.63	1.10
5	nv	.67	1.23	2.00	.97	1.06	1.12	.94	1.76	3.42
	v	.67	1.26	2.00	1.01	1.09	1.16	1.06	1.33	1.77
6	nv	.67	1.29	2.00	1.03	1.07	1.11	1.11	1.72	3.16
	v	.67	1.33	2.00	.99	1.11	1.24	.98	1.33	1.56
7	nv	.10	.80	1.00	.51	.91	1.12	.16	1.03	3.32
	v	.10	.80	1.00	.44	.89	1.14	.15	.93	2.04
8	nv	.38	.79	1.61	.81	.92	1.07	.45	.75	1.32
	v	.40	.82	1.64	.74	.91	1.20	.47	.82	1.47
9	nv	.52	.78	.97	.78	.89	.96	.48	.69	.87
	v	.52	.81	1.02	.72	.89	1.01	.54	.80	1.00
10	nv	.62	.94	1.19	.78	.90	.98	.53	.77	.94
	v	.67	.96	1.21	.76	.95	1.09	.63	.93	1.18
11	nv	.89	.98	1.12	.88	.93	1.00	.79	.87	.99
	v	.86	.98	1.13	.87	.96	1.06	.83	.94	1.09
12	nv	1.93	1.93	1.93	1.13	1.13	1.13	1.37	1.37	1.37
	v	1.93	1.93	1.93	1.55	1.55	1.55	1.87	1.87	1.87

Whenever a number less than 1 appears in the table, it means the new algorithm was faster, and numbers greater than 1 indicate the old algorithm is faster. An examination of the table shows that on the whole the performance of the two algorithms is comparable. Counting bidiagonalization, the new algorithm varies from over 2.7 times faster (class 2) to 1.55 times slower (class 12). Not counting bidiagonalization the extremes are 7.7 times faster to 3.42 times slower; the extra overhead of bidiagonalization moderates the extremes. On simply graded matrices (classes 1-4) and on random matrices (classes 8-11) the new algorithm always did better than the old on the average. With the diagonal and offdiagonal being graded differently (classes 5-6), the old algorithm was generally a little faster. In classes 5-7 the largest ratios occurred in examples where convergence was very fast with both algorithms, the old SVD's faster convergence criterion winning out over the new algorithm's more careful but more expensive convergence test. In class 7 without bidiagonalization and without computing singular vectors, there were only two matrices where the old algorithm beat the new (by factors of 2.42 and 3.32); in both cases both algorithms converged after a *single* QR sweep. Thus the difference in times can be attributed to the slower convergence criteria of the new algorithm; in both cases convergence was nearly immediate. Similarly, in classes 5 and 6 without bidiagonalization and without computing singular vectors, whenever the old algorithm beat the new algorithm by more than 32%, the ratio "QR inner loops"/$(n(n+1)/2)$ was less than .17. In class 7 and many examples in classes 5-6 there were generally a few very small singular values and the rest large and evenly spaced over a range of at most a few factors of 10; the new algorithm deflated out the smallest singular values after 1 or 2 sweeps and spent the rest of the time working on the closely spaced singular values. It appears our criterion for choosing between zero and nonzero shift chooses the zero shift quite often, sometimes sacrificing cubic convergence until many

singular values have been deflated. The single matrix in Class 12 was therefore chosen with very closely spaced singular values in order to make the new algorithm perform as poorly as possible; in this example the average number of (mostly zero shift) QR sweeps per singular value was 2.96 for the new algorithm, whereas the average number of (shifted) QR sweeps per singular value was 1.53 for the old algorithm, which still computed them all correctly. We are not currently able to find another criterion permitting more frequent nonzero shifts while still guaranteeing high relative accuracy. Nonzero shifts for fairly small singular values frequently do not cause inaccuracy in practice because small rotation angles prevent mixing large and small magnitude matrix entries; unfortunately this phenomenon seems hard to exploit systematically.

From Table 3, it appears that Convergence Criteria 2a and 2b are not very effective, since the ratio of inner loops (columns 3-5) does not change very much when Job= nv (singular vectors are not computed and Criteria 2a and 2b are used) and when Job= v (singular vectors are computed and Criteria 2a and 2b are *not* used). This is somewhat misleading, however. Closer inspection of the test cases shows that in classes 1 and 2, Criteria 2a and 2b cut the ratio of inner loops in half for matrices which have constant ratios between adjacent diagonal matrix entries. In these cases, the algorithm converges in a single QR sweep, instead of two QR sweeps. But for the other test matrices in classes 1 and 2, where matrix entries come in equal pairs, Criteria 2a and 2b have no effect at all. The excellent performance on the first set of test matrices is watered down in the statistics presented. Of course, since this speed up is only for matrices for which the algorithm is already quite fast, we could simply omit Criteria 2a and 2b altogether; this would have the advantage of computing identical singular values independent of whether we also compute singular vectors.

Another interesting feature of Table 3 is the difference between classes 1 and 2. Recall that these matrices differ only in the order of the data. In class 1, the old and new algorithms are always chasing the bulge in the same direction; in class 2 they always chase the bulge in the opposite direction, which degrades the accuracy of the old algorithm as mentioned above. It also degrades the performance by about a factor of 2: in class 1 (without bidiagonalization and without computing singular vectors) the new algorithm is about twice as fast as the old on the average, and in class 2 four times as fast.

We next present some timings for our algorithm with $tol = 10^{14}\varepsilon \approx 10^{-2}$ compared to the new algorithm with $tol = 100\varepsilon \approx 10^{-14}$. This low accuracy requirement speeds up the algorithm while still providing order-of-magnitude correct singular values; thus it may be of use for rank determination. Only "Job= nv" (singular values only) cases were run. The new algorithm with $tol = 10^{14}\varepsilon$ was always faster than the new algorithm with $tol = 10^{2}\varepsilon$ except for two matrices in class 4 and one in class 5. In all cases the computed singular values were good to at least 2 figures as expected.

Table 4: Timing Comparisons of New SVD Algorithm with $tol = 10^{14}\varepsilon \approx 10^{-2}$ and $tol = 10^{2}\varepsilon \approx 10^{-14}$

Class	Ratio of Inner Loops SVD ($tol \approx 10^{-2}$) / SVD ($tol \approx 10^{-14}$)			Ratio of Times (with bidiagonalization) SVD ($tol \approx 10^{-2}$) / SVD ($tol \approx 10^{-14}$)			Ratio of Times (without bidiagonalization) SVD ($tol \approx 10^{-2}$) / SVD ($tol \approx 10^{-14}$)		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	.19	.58	1.00	.80	.91	1.00	.33	.65	.95
2	.19	.58	1.00	.79	.91	1.00	.32	.66	.95
3	.47	.70	.84	.88	.95	.98	.68	.81	.92
4	.56	.86	1.05	.94	.99	1.08	.71	.93	1.24
5	.07	.35	1.00	.61	.85	1.00	.19	.54	1.00
6	.07	.35	1.00	.61	.85	1.00	.19	.53	1.00
7	.09	.33	1.00	.46	.70	1.00	.13	.37	1.00
8	.10	.23	.46	.80	.88	.96	.26	.45	.71
9	.07	.20	.40	.77	.83	.91	.21	.36	.57
10	.04	.17	.30	.68	.74	.81	.15	.29	.44
11	.27	.41	.48	.63	.71	.76	.31	.46	.53
12	.14	.14	.14	.68	.68	.68	.21	.21	.21

As mentioned at the end of section 4 on convergence criteria, we may use the much less stringent criteria (4.5) and (4.6) if only absolute accuracy rather than relative accuracy in the singular values is desired. In Table 5 we show timing comparisons between new algorithm where each singular value is computed to an absolute accuracy of $tol \cdot A = 100\varepsilon A \approx 10^{-14} \cdot A$, and the new algorithm with a relative accuracy tolerance $tol = 100\varepsilon \approx 10^{-14}$ as in Table 3. The format is the same as in Table 3. As can be seen from Table 5, the absolute convergence criterion almost always leads to faster convergence than the relative convergence criterion.

Table 5: Timing Comparisons with Absolute Accuracy $tol = 100\varepsilon \cdot A \approx 10^{-14} A$ Versus Relative Accuracy $tol = 100\varepsilon \approx 10^{-14}$

Class	Job	Ratio of Inner Loops SVD (absolute *tol*) / SVD (relative *tol*)			Ratio of Times (with bidiagonalization) SVD (absolute *tol*) / SVD (relative *tol*)			Ratio of Times (without bidiagonalization) SVD (absolute *tol*) / SVD (relative *tol*)		
		Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	nv	.21	.48	1.00	.87	.91	.97	.32	.51	.92
	v	.17	.40	.77	.76	.83	.89	.32	.47	.74
2	nv	.21	.48	1.00	.87	.91	.97	.32	.52	.92
	v	.17	.40	.77	.76	.83	.89	.31	.46	.75
3	nv	.06	.39	.87	.81	.88	.97	.13	.42	.85
	v	.06	.39	.86	.59	.73	.92	.09	.41	.85
4	nv	.03	.25	.70	.82	.88	.94	.11	.30	.71
	v	.03	.25	.73	.66	.73	.85	.07	.28	.72
5	nv	.14	.47	.92	.75	.89	.94	.21	.47	.87
	v	.14	.46	.86	.68	.85	.91	.26	.50	.85
6	nv	.14	.47	.92	.75	.89	.94	.21	.47	.87
	v	.14	.46	.86	.68	.85	.91	.25	.50	.85
7	nv	.82	.94	1.00	.92	.96	.99	.41	.83	.98
	v	.82	.94	1.00	.92	.96	1.00	.66	.88	1.00
8	nv	.26	.40	.64	.82	.88	.93	.32	.43	.55
	v	.25	.40	.62	.66	.78	.89	.29	.43	.62
9	nv	.50	.68	.95	.88	.91	.97	.54	.67	.87
	v	.51	.68	.93	.78	.85	.96	.53	.68	.91
10	nv	.60	.83	1.00	.88	.93	.98	.65	.82	.96
	v	.61	.83	.97	.78	.90	.97	.62	.83	.95
11	nv	.97	1.00	1.00	1.00	1.01	1.02	.99	1.01	1.04
	v	.97	1.00	1.05	.98	1.00	1.04	.97	1.00	1.05
12	nv	.88	.88	.88	.95	.95	.95	.89	.89	.89
	v	.88	.88	.88	.91	.91	.91	.87	.87	.87

Finally, we discuss the implications of our results for the "perfect shift" strategy for computing singular vectors (or eigenvectors). This strategy advocates computing the singular values (or eigenvalues) by the quickest available method without accumulating singular vectors, and then using these computed singular values as "perfect shifts" in the QR iteration to compute the singular vectors in 1 or possibly 2 QR sweeps. The hope is that by avoiding the work of accumulating vectors while converging to accurate singular values, time will be saved by computing the singular vectors afterwards in one or two sweeps each. Unfortunately, our numerical results indicate this approach will not work in general. For when our hybrid algorithm chooses to do an implicit zero shift, it is in fact doing a perfect shift within the limits of roundoff error. Depending on the distribution of singular values, this can take more or less time to converge. Therefore one cannot assume 1 or 2 sweeps with the "perfect shift" will results in converged singular vectors, and we could well end up doing as many sweeps to compute the singular vectors as the singular values. This will not happen in general, and a clever algorithm might be able to decide when perfect shifts are useful and then use them, perhaps by keeping track of which deflated subblocks of the matrix do not require zero shifts and using the perfect shift strategy on them.

8. Detailed Error Analysis

In this section we present a detailed error analysis of the implicit zero-shift QR algorithm (Theorems 6 and 7). We begin with the assumptions used in the error analyses. Our model of error in floating point arithmetic was given above in (3.5). It implies that overflow and underflow do not occur; we discuss susceptibility to overflow and underflow briefly at the end. In our analysis εs with numerical subscripts denote quantities bounded in magnitude by ε, where ε is the machine precision. Our analysis will be linearized in the sense that we will replace quantities like $(1+\varepsilon_1)*(1+\varepsilon_2)$ by $1+(\varepsilon_1+\varepsilon_2)$ and $(1+\varepsilon_1)/(1+\varepsilon_2)$ by $1+(\varepsilon_1-\varepsilon_2)$; such approximations can be made rigorous by assuming all ε_i are less than .1 in magnitude and increasing the final error bound by a factor 1.06 [18, p. 113].

Lemma 5: Let cosθ, sinθ and ρ denote the exact outputs of *ROT* for inputs f and g and exact arithmetic. Now consider the floating point version of *ROT* applied to the perturbed inputs $\hat{f} = f(1+\varepsilon_f)$ and $\hat{g} = g(1+\varepsilon_g)$, and let $cs = (1+\varepsilon_{cs})\cos\theta$, $sn = (1+\varepsilon_{sn})\sin\theta$, and $r = (1+\varepsilon_r)\rho$ denote the computed results, where we assume neither overflow nor underflow occurs. Then we may estimate the relative errors ε_{cs}, ε_{sn} and ε_r as follows:

$$\varepsilon_{cs} = (\varepsilon_f-\varepsilon_g)\sin^2\theta + \varepsilon_{cs}' \text{ , where } |\varepsilon_{cs}'| \le \frac{21}{4}\varepsilon$$

$$\varepsilon_{sn} = (\varepsilon_g-\varepsilon_f)\cos^2\theta + \varepsilon_{sn}' \text{ , where } |\varepsilon_{sn}'| \le \frac{21}{4}\varepsilon$$

$$\varepsilon_r = \varepsilon_g\sin^2\theta + \varepsilon_f\cos^2\theta + \varepsilon_r' \text{ , where } |\varepsilon_r'| \le \frac{13}{4}\varepsilon$$

Proof: We only consider the case $|\hat{f}| > |\hat{g}|$; the other case is analogous. In the following εs with numeric subscripts indicate quantities bounded by ε which may be functions of previous ε_is as described in the first paragraph of this section. Then applying (3.5) systematically to the expressions in *ROT*, and using the fact that $|\hat{g}/\hat{f}| < 1$, yields

$$t = \frac{g}{f}(1+\varepsilon_g-\varepsilon_f+\varepsilon_1)$$

$$\begin{aligned} tt &= (1+\varepsilon_4)\cdot[(1+\varepsilon_3)\cdot(1 + t^2(1+\varepsilon_2))]^{1/2} \\ &= (1+7\varepsilon_5/4)[1+t^2]^{1/2} \\ &= (1+7\varepsilon_5/4)[1+(g/f)^2(1 + 2(\varepsilon_g-\varepsilon_f+\varepsilon_1))]^{1/2} \\ &= (1+7\varepsilon_5/4)(1 + (\varepsilon_g-\varepsilon_f+\varepsilon_1)(g/f)^2/(1+(g/f)^2))[1+(g/f)^2]^{1/2} \\ &= (1 + 9\varepsilon_6/4 + (\varepsilon_g-\varepsilon_f)\sin^2\theta)\sec\theta \end{aligned}$$

$$cs = (1+\varepsilon_7)/tt = (1 + 13\varepsilon_8/4 + (\varepsilon_f-\varepsilon_g)\sin^2\theta)\cos\theta$$

$$sn = (1+\varepsilon_9)t\cdot cs = (1 + 21\varepsilon_{10}/4 + (\varepsilon_g-\varepsilon_f)\cos^2\theta)\sin\theta$$

$$r = (1+\varepsilon_{11})f\cdot tt = (1 + 13\varepsilon_{12}/4 + \varepsilon_g\sin^2\theta + \varepsilon_f\cos^2\theta)\rho$$

□

To analyze the errors in the implicit zero-shift QR algorithm, we need to investigate how the errors accumulate from one pass through the loop to the next. It turns out the errors in f and *oldcs* are the essential ones:

Lemma 6: Let f_i and $oldcs_i$ denote the true values of f and *oldcs* at the entry to the i-th iteration of the loop in the implicit zero-shift QR algorithm. Let θ_{1i} and θ_{2i} be the true values of the two rotation angles in the i-th iteration of the loop. In other words, f_i, $oldcs_i$, θ_{1i} and θ_{2i} are the values that would have been computed had all arithmetic been exact. Let $f_i(1+\varepsilon_{f_i})$ and

$oldcs_i(1+\varepsilon_{oldcs_i})$ denote the actual floating point values of f and $oldcs$ at the top of the loop, with all previous loop iterations having been done in floating point without any overflows or underflows. Then

$$\begin{bmatrix} |\varepsilon_{f_{i+1}}| \\ |\varepsilon_{oldcs_{i+1}}| \end{bmatrix} \le \begin{bmatrix} \sin^2\theta_{1i} & 0 \\ 2\cos^2\theta_{1i}\cdot\sin^2\theta_{2i} & \sin^2\theta_{2i} \end{bmatrix} \cdot \begin{bmatrix} |\varepsilon_{f_i}| \\ |\varepsilon_{oldcs_i}| \end{bmatrix} + \begin{bmatrix} 25\varepsilon\ /4 \\ 21\varepsilon\ /4 + 21\varepsilon\cdot\sin^2\theta_{2i}\ /2 \end{bmatrix} \tag{8.1}$$

In terms of these expressions, we can bound the errors in the computed values of e_i and s_i:

$$e_i = \text{"}true\ e_i\text{"}\cdot(1 + \varepsilon_{e_i}) \quad \text{and} \quad s_i = \text{"}true\ s_i\text{"}\cdot(1 + \varepsilon_{s_i})$$

where

$$\varepsilon_{e_i} = -\varepsilon_{oldcs_i}\cdot\cos^2\theta_{2i} - 2\varepsilon_{f_i}\cdot\cos^2\theta_{1i}\cdot\cos^2\theta_{2i} + \varepsilon_{f_{i+1}}\cdot\cos^2\theta_{1,i+1} + 20\varepsilon \tag{8.2}$$

and

$$\varepsilon_{s_i} = \varepsilon_{f_i}\cdot\cos^2\theta_{1i}\cdot\cos2\theta_{2i} + \varepsilon_{oldcs_i}\cdot\cos^2\theta_{2i} + \frac{15}{2}\varepsilon_{14} + \frac{25}{4}\sin^2\theta_{2i}\varepsilon_{15} \quad . \tag{8.3}$$

ε_{e_i} and ε_{s_i} may be further bounded by:

$$|\varepsilon_{e_i}| \le |\varepsilon_{oldcs_i}| + 2|\varepsilon_{f_i}| + |\varepsilon_{f_{i+1}}| + 20\varepsilon \tag{8.4}$$

and

$$|\varepsilon_{s_i}| \le |\varepsilon_{f_i}| + |\varepsilon_{oldcs_i}| + 25\varepsilon\sin^2\theta_{2i}\ /4 + 15\varepsilon/2 \quad . \tag{8.5}$$

Proof: We apply (3.5) and Lemma 5 systematically to the expressions in the algorithm. As before, εs with numeric subscripts denote expressions bounded in magnitude by ε.

Note that at the top of the loop, g is known exactly. Therefore, after the first call to *ROT* we have

$$cs = \cos\theta_{1i}\cdot(1 + \varepsilon_{f_i}\cdot\sin^2\theta_{1i} + \frac{21}{4}\varepsilon_1)$$

$$sn = \sin\theta_{1i}\cdot(1 - \varepsilon_{f_i}\cdot\cos^2\theta_{1i} + \frac{21}{4}\varepsilon_2)$$

$$r = \text{"}true\ r\text{"}\cdot(1 + \varepsilon_{f_i}\cdot\cos^2\theta_{1i} + \frac{13}{4}\varepsilon_3)$$

The errors in f, g and h are given by

$$f = (1+\varepsilon_4)\cdot oldcs\cdot r = \text{"}true\ f\text{"}\cdot(1 + \varepsilon_{oldcs_i} + \varepsilon_{f_i}\cdot\cos^2\theta_{1i} + \frac{17}{4}\varepsilon_5)$$

$$g = (1+\varepsilon_6)\cdot s_{i+1}\cdot sn = \text{"}true\ g\text{"}\cdot(1 - \varepsilon_{f_i}\cdot\cos^2\theta_{1i} + \frac{25}{4}\varepsilon_7)$$

$$h = (1+\varepsilon_8)\cdot s_{i+1}\cdot cs = \text{"}true\ h\text{"}\cdot(1 + \varepsilon_{f_i}\cdot\sin^2\theta_{1i} + \frac{25}{4}\varepsilon_9)$$

After the second call to *ROT* we have

$$cs = \cos\theta_{2i}\cdot(1 + \varepsilon_{oldcs_i}\cdot\sin^2\theta_{2i} + 2\varepsilon_{f_i}\cdot\cos^2\theta_{1i}\cdot\sin^2\theta_{2i} + \frac{21}{2}\varepsilon_{10}\cdot\sin^2\theta_{2i} + \frac{21}{4}\varepsilon_{11})$$

$$sn = \sin\theta_{2i}\cdot(1 - \varepsilon_{oldcs_i}\cdot\cos^2\theta_{2i} - 2\varepsilon_{f_i}\cdot\cos^2\theta_{1i}\cdot\cos^2\theta_{2i} + \frac{21}{2}\varepsilon_{12}\cdot\cos^2\theta_{2i} + \frac{21}{4}\varepsilon_{13})$$

$$r = \text{"}true\ r\text{"}\cdot(1 + \varepsilon_{f_i}\cdot\cos^2\theta_{1i}\cdot\cos2\theta_{2i} + \varepsilon_{oldcs_i}\cdot\cos^2\theta_{2i} + \frac{15}{2}\varepsilon_{14} + \frac{25}{4}\varepsilon_{15}\cdot\sin^2\theta_{2i})$$

Since $oldcs = cs$ and $f = h$ at the bottom of the loop, we have shown that at the start of the next loop

$$f = f_{i+1}(1 + \varepsilon_{f_i}\cdot\sin^2\theta_{1i} + \frac{25}{4}\varepsilon_9)$$

$$oldcs = oldcs_{i+1}(1 + \varepsilon_{oldcs_i}\cdot\sin^2\theta_{2i} + 2\varepsilon_{f_i}\cdot\cos^2\theta_{1i}\cdot\sin^2\theta_{2i} + \frac{21}{2}\varepsilon_{10}\cdot\sin^2\theta_{2i} + \frac{21}{4}\varepsilon_{11})$$

as desired. The expressions for the errors in e_i and s_i follow from plugging the above bounds into the expressions $e_{i-1} = oldsn*r$ and $s_i = r$. □

From (8.4) and (8.5) we see that the errors in the computed e_i and s_i are governed by the errors ε_{f_i} and ε_{oldcs_i}, and that the growths of these errors are governed by the recurrence (8.1). A simple but somewhat pessimistic bound on these errors is given by

Lemma 7: Let ε_{f_i}, ε_{oldcs_i}, ε_{e_i} and ε_{s_i} be as in Lemma 6. Then

$$|\varepsilon_{f_i}| \le 25(i-1)\varepsilon/4$$
$$|\varepsilon_{oldcs_i}| \le 113(i-1)\varepsilon/4$$
$$|\varepsilon_{e_i}| \le (138i-53)\varepsilon/4$$
$$|\varepsilon_{s_i}| \le (113i-58)\varepsilon/4$$

Proof: Replace the recurrence (8.1) by

$$E_{i+1} = A_i\cdot E_i + F_i$$

where

$$A_i = \begin{bmatrix} \sin^2\theta_{1i} & 0 \\ 2\cos^2\theta_{1i}\cdot\sin^2\theta_{2i} & \sin^2\theta_{2i} \end{bmatrix} \quad , \quad F_i = \begin{bmatrix} 25\varepsilon/4 \\ 21\varepsilon/4 + 21\varepsilon\cdot\sin^2\theta_{2i}/2 \end{bmatrix}$$

and the entries of E_i are upper bounds for $|\varepsilon_{f_i}|$ and $|\varepsilon_{oldcs_i}|$. Taking $E_1 = 0$, this recurrence has the solution

$$E_{i+1} = \sum_{j=1}^{i} \left(\prod_{k=j+1}^{i} A_k \right) F_j \tag{8.6}$$

Trivial bounds for A_i and F_i are

$$|A_i| \le \begin{bmatrix} \sin^2\theta_{1i} & 0 \\ 2\cos^2\theta_{1i} & 1 \end{bmatrix} \quad \text{and} \quad |F_i| \le \begin{bmatrix} 25\varepsilon/4 \\ 63\varepsilon/4 \end{bmatrix} \tag{8.7}$$

A simple induction shows that

$$\prod_{k=j+1}^{i} \begin{bmatrix} \sin^2\theta_{1k} & 0 \\ 2\cos^2\theta_{1k} & 1 \end{bmatrix} = \begin{bmatrix} \prod_{k=j+1}^{i} \sin^2\theta_{1k} & 0 \\ 2\left(1 - \prod_{k=j+1}^{i} \sin^2\theta_{1k}\right) & 1 \end{bmatrix}$$

and the rest of the proof is a straightforward computation. □

In other words, the relative errors in the computed e_i and s_i are bounded by a linear function of i, and so the largest relative error is bounded by a linear function of the matrix dimension n. We can now apply Theorem 2 of section 2 to bound the errors in the singular values of the transformed matrix:

Theorem 6: Let B be an n by n bidiagonal matrix and B' the matrix obtained by running the implicit zero-shift QR algorithm on B. Let the singular values of B be $\sigma_1 \ge \cdots \ge \sigma_n$, and the singular values of B' be $\sigma_1' \ge \cdots \ge \sigma_n'$. Then if

$$\omega \equiv 69n^2\varepsilon < 1 \quad , \tag{8.8}$$

the relative differences between the singular values of B and the singular values of B' are bounded as follows:

$$|\sigma_i - \sigma_i'| \leq \frac{\omega}{1-\omega}\sigma_i \quad .$$

Let B_k be the matrix obtained after k repetitions of the implicit zero-shift QR algorithm, and let $\sigma_{k1} \geq \cdots \geq \sigma_{kn}$ be its singular values. Then if condition (8.8) holds we have

$$|\sigma_i - \sigma_{ki}| \leq \left(\frac{1}{(1-\omega)^k} - 1\right) \cdot \sigma_i \approx 69kn^2\varepsilon \cdot \sigma_i \quad ,$$

where the approximation to the last upper bound holds if $k\omega \ll 1$.

Proof: Plug the bounds of Lemma 7 into Theorem 2. □

Actually, Lemma 7 and Theorem 6 are quite pessimistic, since the upper bounds in (8.7) are unattainable. In fact, as we approach convergence, we expect the rotation angles θ_{1i} and θ_{2i} to approach zero, which means the matrix A_i should approach zero. We can use this fact to obtain a much stronger error bound in the region of convergence:

Lemma 8: Let ε_{f_i}, ε_{oldcs_i}, ε_{e_i} and ε_{s_i} be as in Lemma 6. Assume further that all the rotation angles θ during the course of the algorithm satisfy $\sin^2\theta \leq \tau < 1$. Then

$$|\varepsilon_{f_i}| \leq \frac{25\varepsilon}{4(1-\tau)}$$

$$|\varepsilon_{oldcs_i}| \leq \frac{50\tau\varepsilon}{4(1-\tau)^2} + \frac{21\tau\varepsilon}{2(1-\tau)} + \frac{21\varepsilon}{4(1-\tau)}$$

$$|\varepsilon_{e_i}| \leq \frac{50\tau\varepsilon}{4(1-\tau)^2} + \frac{21\tau\varepsilon}{2(1-\tau)} + \frac{24\varepsilon}{1-\tau} + 20\varepsilon$$

$$|\varepsilon_{s_i}| \leq \frac{50\tau\varepsilon}{4(1-\tau)^2} + \frac{21\tau\varepsilon}{2(1-\tau)} + \frac{23\varepsilon}{2(1-\tau)} + \frac{25\tau\varepsilon}{4} + \frac{15\varepsilon}{2}$$

Proof: Use the bounds

$$|A_i| \leq \begin{bmatrix} \tau & 0 \\ 2\tau & \tau \end{bmatrix} \quad \text{and} \quad |F_i| \leq \begin{bmatrix} 25\varepsilon/4 \\ 21\varepsilon/4 + 21\varepsilon\tau/2 \end{bmatrix} \quad .$$

The rest of the proof is a straightforward computation from (8.6). □

These bounds permit us state the following improvement to Theorem 6:

Theorem 7: Let B be an n by n bidiagonal matrix and B' the matrix obtained by running the implicit zero-shift QR algorithm on B. Assume that all the rotation angles θ during the course of the algorithm satisfy $\sin^2\theta \leq \tau < 1$. Let the singular values of B be $\sigma_1 \geq \cdots \geq \sigma_n$, and the singular values of B' be $\sigma_1' \geq \cdots \geq \sigma_n'$. Then if

$$\omega \equiv \frac{88n\varepsilon}{(1-\tau)^2} < 1 \quad , \tag{8.9}$$

the relative differences between the singular values of B and the singular values of B' are bounded as follows:

$$|\sigma_i - \sigma_i'| \leq \frac{\omega}{1-\omega}\sigma_i \quad .$$

Let B_k be the matrix obtained after k repetitions of the implicit zero-shift QR algorithm, where we assume all rotation angles θ satisfy $\sin^2\theta \leq \tau < 1$. Let $\sigma_{k1} \geq \cdots \geq \sigma_{kn}$ be the singular values of B_k. Then if condition (8.9) holds we have

$$|\sigma_i - \sigma_{ki}| \leq \left(\frac{1}{(1-\omega)^k} - 1\right) \cdot \sigma_i \approx \frac{88kn\varepsilon}{(1-\tau)^2} \cdot \sigma_i \quad ,$$

where the approximation to the last upper bound holds if $k\omega \ll 1$.

Proof: Plug the bounds of Lemma 8 into Theorem 2. □

Thus, if the rotation angles are all bounded away from $\pi/2$, the error after k iterations of the implicit zero-shift QR algorithm can grow essentially only as the product kn. The algorithm can easily compute τ as it proceeds, and so compute its own error bound if desired. In the numerical experiments in section 7, we observed no error growth at all, and so as is often the case an algorithm behaves much better in practice than rigorous error bounds can guarantee.

Now we briefly consider over/underflow. Most of the error analyses presented here can be extended to take over/underflow into account. Techniques for error analysis in the presence of underflow are discussed in [2]. If over/underflow is handled as suggested in the IEEE Floating Point Standard [10], then using Sylvester's theorem to count the number of eigenvalues less than x (2.1) can be made completely impervious to over/underflow [12]: If some $d_i = \pm 0$, then $d_{i+1} = \pm \infty$ and $d_{i+2} = a_{i+2}$, and we count the number of d_i whose sign bit is negative (i.e. including -0 and $-\infty$). Rules for arithmetic with ± 0 and $\pm \infty$ are described in detail in [10].

9. The Accuracy of the Computed Singular Vectors

In this section we assess the accuracy of the computed singular vectors. Just as with the standard SVD, the new algorithm guarantees a small residual in the sense that both $B\hat{v}-\hat{\sigma}\hat{u}$ and $B^T\hat{u}-\hat{\sigma}\hat{v}$ are on the order of εB, where $\hat{\sigma}$ is the computed singular value and $\hat{u}$ and $\hat{v}$ are the computed singular vectors. However, in contrast to the singular values, high relative accuracy in the bidiagonal matrix entries does not guarantee high relative accuracy in the singular vectors; we will give a 2 by 2 example to illustrate this. It also turns out to be impossible to guarantee a tiny componentwise relative backwards error, where each computed singular vector of B would be the exact singular vector of a small componentwise relative perturbation $B+\delta B$ of B, with $|\delta B| \leq \eta |B|$, η on the order of machine precision. We will also demonstrate this with a small example.

In place of such simple a priori forward or backward error bounds, our bounds will depend on the singular value distribution. Briefly, the closer together singular values are, the less accurately their corresponding singular vectors can be computed. This dependency is captured in the well known "gap" theorem [15, p. 222] which can be used to show that the angular error in the computed singular vectors corresponding to σ_i is bounded by the largest roundoff error committed divided by the "gap" or difference between σ_i and its nearest neighbor $\sigma_{i\pm 1}$. This well known bound holds for the standard SVD applied to dense matrices as well as the new algorithm.

Numerical experience leads us to make the following conjecture for the new algorithm applied to bidiagonal matrices which would significantly strengthen the bound in the last paragraph: the "gap" $\min|\sigma_i-\sigma_{i\pm 1}|$ in the denominator of the above error bound can be replaced by the "relative gap" $\min(|\sigma_i-\sigma_{i\pm 1}|/\sigma_i)$. Since the relative gap can be much larger than the gap, the resulting error bound can be much smaller. For example, if B is 3 by 3 bidiagonal matrix with singular values $\sigma_1=1$, $\sigma_2=2\cdot 10^{-20}$ and $\sigma_3=10^{-20}$, the old error bounds for the vectors corresponding to the two tiny singular values are on the order of $10^{20}\varepsilon$ since the gap is 10^{-20}. However, the conjectured bounds are both ε since the relative gap between $2\cdot 10^{-20}$ and 10^{-20} is 1. Proving this conjecture rigorously remains an open problem, although a supporting result appears in [1].

Now we present a two by two example showing that small relative perturbations in the entries of a bidiagonal matrix can cause large perturbations in the singular vectors:

$$A(\eta) = \begin{bmatrix} 1+\eta & \varepsilon \\ 0 & 1 \end{bmatrix} .$$

As η varies from 0 to ε, an easy computation shows that both left and right singular vectors rotate by 22.5 degrees.

The same example can be used to show that no tiny componentwise relative backward error bound can hold. Specifically, let $\hat{u}_i$ and $\hat{v}_i$ be the left and right unit singular vectors of

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

computed by the new algorithm (for ease of presentation we ignore the fact that 2 by 2 matrices are handled specially by the algorithm; this same phenomenon holds for larger examples as well). Suppose that $\hat{u}_i$ and $\hat{v}_i$ differ by at most ε_1 from the exact unit singular vectors u_i and v_i of a componentwise relative perturbation $A+\delta A$ of A, where $|\delta A| \leq \varepsilon_2 |A|$. Then if $\varepsilon_2 = o(\varepsilon^{2/3})$, $\varepsilon_1 \cdot \varepsilon_2 = \Omega(\varepsilon)$. In other words, ε_1 is $\Omega(\varepsilon^{1/3})$. Therefore, any attempt to prove a small componentwise relative backward error $o(\varepsilon^{2/3})$ must permit errors in the computed vectors at least as large as $\varepsilon^{1/3} >> \varepsilon$.

The proof goes as follows. Applying the new algorithm to A (and ignoring the fact that 2 by 2 matrices are handled specially), we set $A_{1,2}$ to 0 and get the columns of the identity matrix as left and right singular vectors. Now we make relative perturbations of size at most $\varepsilon_2 = \varepsilon^{\alpha}$ ($\alpha > 2/3$) in each entry of A (here, $|\varepsilon_{2i}| \leq \varepsilon_2$):

$$B = A + \delta A = \begin{bmatrix} (1+\varepsilon)(1+\varepsilon_{21}) & \varepsilon(1+\varepsilon_{22}) \\ 0 & 1+\varepsilon_{23} \end{bmatrix}$$

Compute

$$B^T B = I + \begin{bmatrix} 2\varepsilon+2\varepsilon_{21} & \varepsilon \\ \varepsilon & 2\varepsilon_{23} \end{bmatrix} + O(\varepsilon^2+\varepsilon_2^2)$$

$$\equiv I + \varepsilon \begin{bmatrix} x & 1 \\ 1 & y \end{bmatrix} + O(\varepsilon^2+\varepsilon_2^2)$$

$$\equiv I + \varepsilon C + O(\varepsilon^2+\varepsilon_2^2)$$

where $|x| \leq 2+2\varepsilon_2/\varepsilon$ and $|y| \leq 2\varepsilon_2/\varepsilon$. We consider the eigenproblem for C. Suppose $[1\ \eta]^T$ is an eigenvector of C; we will show

$$\frac{1}{3 + 4\varepsilon_2/\varepsilon} \leq |\eta| \leq 3 + 4\varepsilon_2/\varepsilon$$

implying the angle between an eigenvector of C and $[1\ 0]^T$ is $\Omega(\varepsilon/\varepsilon_2) = \Omega(\varepsilon^{1-\alpha})$. We compute as follows. If $[1\ \eta]^T$ is an eigenvector of C, η must satisfy $\eta^2+(x-y)\eta-1=0$ or

$$\eta = \frac{(y-x)}{2} \pm \left(\left(\frac{(y-x)}{2}\right)^2 + 1\right)^{1/2} .$$

Since $|(y-x)/2| \leq 1+2\varepsilon_2/\varepsilon$, it is easy to see both $|\eta|$ and $|\eta^{-1}|$ are bounded by

$$1+2\varepsilon_2/\varepsilon + ((1+2\varepsilon_2/\varepsilon)^2+1)^{1/2} \leq 3+4\varepsilon_2/\varepsilon$$

as desired.

Now consider the so far ignored perturbation $O(\varepsilon^2+\varepsilon_2^2)$. The gap between the eigenvalues of C is computed to be $((x-y)^2+4)^{1/2} \geq 2$. Thus the perturbation $O(\varepsilon^2+\varepsilon_2^2)$ can change the eigenvectors by at most $O(\varepsilon+\varepsilon_2^2/\varepsilon)$. When $\varepsilon_2=\varepsilon^\alpha$, this is a perturbation of at most $O(\varepsilon^{2\alpha-1})$. But when $\alpha > 2/3$, $2\alpha-1 > 1-\alpha$ and so the perturbation cannot change the lower bound $\Omega(\varepsilon^{1-\alpha})$ on $|\eta|$.

Thus, a relative perturbation of size ε^α ($\alpha > 2/3$) to A means the right singular vectors are least $\Omega(\varepsilon^{1-\alpha}) \equiv \Omega(\varepsilon_1)$ away from the computed right singular vectors. Thus $\varepsilon_1 \cdot \varepsilon_2 = \Omega(\varepsilon)$ as desired.

Since our algorithm handles 2 by 2 matrices as special cases, a 4 by 4 matrix like

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & \varepsilon & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

could be used in the proof, but the computations are more complicated.

As stated above, rigorous error bounds on the computed singular vectors depend on the singular value distribution, and that the closer together singular values are, the less accurately their corresponding singular vectors can be computed. The "gap" theorem [15, p. 222] expresses this dependency by bounding the angular error in the computed singular vectors by the residual $B\hat{v}_i - \hat{\sigma}_i\hat{u}_i$, $B^T\hat{u}_i - \hat{\sigma}_i\hat{v}_i$ (the norm of the n by 2 matrix $[B\hat{v}_i - \hat{\sigma}_i\hat{u}_i$, $B^T\hat{u}_i - \hat{\sigma}_i\hat{v}_i]$) divided by the gap (here $\hat{u}_i$, $\hat{v}_i$ and $\hat{\sigma}_i$ are the computed singular vectors and singular value). Standard backward error analysis shows that the residual may be bounded by largest roundoff error committed (which is $p(n)\varepsilon B$, $p(n)$ a modest function of n and ε the machine precision). This yields the error bound

$$\max(\theta(\hat{u}_i, u_i)\ ,\ \theta(\hat{v}_i, v_i)) \leq\ p(n)\varepsilon\ B\ /\ gap \equiv\ p(n)\varepsilon\ B\ /\ \min\ |\sigma_i - \sigma_{i\pm 1}|\ ; \qquad (9.1)$$

here u_i and v_i are the exact singular vectors.

The bound (9.1) is true for the standard SVD of a dense matrix as well as the new algorithm. A natural question is whether the bound can be improved for the new algorithm applied to the bidiagonal singular value problem. Numerical experience and Proposition 7 in [1] support the following

Conjecture: Let B be an unreduced bidiagonal matrix with singular values σ_i and left and right singular vectors u_i and v_i. Let the singular vectors computed by the new algorithm be $\hat{u}_i$ and $\hat{v}_i$. Then the errors in $\hat{u}_i$ and $\hat{v}_i$ are bounded by

$$\max(\theta(\hat{u}_i, u_i)\ ,\ \theta(\hat{v}_i, v_i)) \leq\ p(n)\varepsilon\ /\ \textit{relative gap} \equiv\ p(n)\varepsilon\ /\ \min\ (|\sigma_i - \sigma_{i\pm 1}|/\sigma_i)\ . \qquad (9.2)$$

The justification for this conjecture is as follows. In section 8 we proved that the zero-shift part of the algorithm is forward stable across a single QR sweep; numerical experience indicates that it is actually forward stable across many QR sweeps. (It is straightforward but tedious to show that after k sweeps, rounding errors can grow by at most a factor which is $O(k)$, but it appears difficult to estimate the constant.) This forward stability means the accumulated transformation matrices are computed accurately. Thus, the only serious errors are committed on convergence: setting an offdiagonal to zero. If we use a "conservative" convergence criterion, where only offdiagonals smaller than $\varepsilon \cdot \sigma_{\min}$ are set to zero, the numerator in (9.1) is reduced from $p(n)\varepsilon B$ to $p(n)\varepsilon\sigma_{\min}$, which implies the conjecture. Extending this argument to the stopping criterion described in section 4 appears difficult, and it is possible that with the more conservative stopping criterion the algorithm will occasionally compute more accurate vectors than the criterion of section 4.

10. Conclusions and Open Problems

We have described a method for computing all the singular values of a bidiagonal matrix to nearly full machine precision, and showed it to be comparable in speed to the LINPACK SVD algorithm. This computation is justified because small relative errors in the bidiagonal entries (from roundoff in the algorithm or from previous computations) can only cause small relative errors in the singular values, independent of their magnitudes. The technique can be extended to computing the eigenvalues of symmetric positive definite tridiagonal matrices with high relative accuracy as well [1].

A number of open questions remain. First, how accurate are the singular vectors computed by this algorithm? We cannot generally expect high relative accuracy in all singular vectors, because clustered singular values can have arbitrarily ill-conditioned singular vectors. Still, singular vectors might be computable fully accurately so long as the *relative* differences between corresponding singular values and their neighbors are big enough, at least if we use a stopping criterion more conservative than the one in section 4. When in practice is it necessary to compute such accurate singular vectors for tiny clustered singular values? Do applications demand accurate singular vectors, or are tiny residuals sufficient, and if so, how tiny?

Second, since we have shown that it is possible to obtain accurate singular values from accurate bidiagonal matrices, we may ask when it is possible to guarantee accuracy in the reduction to bidiagonal form. This is clearly not possible in general, but for some special classes of matrices (such as positive definite symmetric tridiagonal matrices [1]) reduction to bidiagonal form is accurate. It may also be possible for graded matrices arising from integral equations. For what classes is this true?

Third, how generally can our implicit zero-shift technique be employed to guarantee accurate singular values and eigenvalues? As mentioned in section 3, a similar technique was used in root-free versions of symmetric tridiagonal QR; can it be modified to produce a tridiagonal symmetric QR algorithm which guarantees accurate eigenvalues for at least some interesting classes of symmetric tridiagonal matrices? This question is addressed in [13].

Finally, what is the best parallel algorithm for high accuracy singular values? As mentioned in section 3, zero-shift QR can be parallelized, but it is not as easy to see how to incorporate shifts and convergence testing. In section 6, we showed that bisection and its refinements could be used to compute high accuracy singular values. Such a technique has been successfully parallelized for finding eigenvalues of symmetric tridiagonal matrices [14]. Another possibility is an algorithm based on divide and conquer [11], although it appears difficult to guarantee high accuracy. The answer will probably depend on whether all or just some singular values are desired; in the latter case bisection will likely be superior.

The code is available electronically from the first author. It will also be incorporated in the LAPACK linear algebra library [3].

References

[1] J. Barlow, J. Demmel, *Computing Accurate Eigensystems of Scaled Diagonally Dominant Matrices*, submitted to SIAM J. Num. Anal.; Courant Institute, Computer Science Dept. Report 421, Dec 1988

[2] J. Demmel, *Underflow and the Reliability of Numerical Software*, SIAM J. Sci. Statist. Comput., vol. 5, no. 4, 1984

[3] J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, D. Sorensen, *Prospectus for the Development of a Linear Algebra Library for High Performance Computers*, Argonne National Laboratory, Mathematics and Computer Science Division, Technical Memorandum No. 97, September 1987

[4] J. Dongarra, E. Grosse, "Distribution of Mathematical Software via Electronic Mail," Comm. of the ACM, vol. 30, no. 5, July 1987, pp 403-407

[5] J. Dongarra, C. Moler, J. Bunch, G. W. Stewart, *LINPACK User's Guide*, SIAM, Philadelphia, 1979

[6] F. R. Gantmacher, *The Theory of Matrices*, Vol. 1, Chelsea, New York, 1959

[7] G. Golub and W. Kahan, *Calculating the Singular Values and Pseudo-inverse of a Matrix*, SIAM J. Num. Anal. Ser. B, Vol. 2, No. 2, 1965, pp. 205-224

[8] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins, Baltimore, 1983

[9] N. J. Higham, *Efficient algorithms for computing the condition number of a tridiagonal matrix*, SIAM J. Sci. Stat. Comput. 7 (1986), pp. 150-65

[10] IEEE Standard for Binary Floating Point Arithmetic, ANSI/IEEE Std 754-1985, IEEE 1985

[11] E. R. Jessup and D. C. Sorensen, *A Parallel Algorithm for Computing the Singular Value Decomposition of a Matrix*, Argonne National Laboratory, Mathematics and Computer Science Division, Technical Memorandum No. 102, December 1987

[12] W. Kahan, *Accurate Eigenvalues of a Symmetric Tridiagonal Matrix*, Technical Report No. CS41, Computer Science Dept., Stanford University, July 22, 1966 (revised June 1968)

[13] J. Le and B. Parlett, *On the Forward Instability of the QR Transformation*, submitted to SIAM J. Mat. Anal. Appl.; also Report PAM-419, Center for Pure and Applied Mathematics, University of California, Berkeley, July 1988

[14] S-S. Lo, B. Phillipe, A. Sameh, *A Multiprocessor Algorithm for the Symmetric Tridiagonal Eigenvalue Problem*, SIAM J. Sci. Stat. Comp., Vol. 8, No. 2, March 1987, pp s155-s165

[15] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, New Jersey, 1980

[16] G. W. Stewart, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Review, Vol. 15, No. 4, October 1973

[17] S. Van Huffel, J. Vandewalle, A. Haegemans, *An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values*, J. Computational and Appl. Math. 19, (1987) 313-30

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

Acknowledgements

W. Kahan was supported at times by grants from the Research Offices of the U.S. Army, Navy and Air Force under contracts numbered respectively DAA629-85-K-0070, N00014-85-K-0013 and AFOSR-84-0158. J. Demmel has been supported as a Presidential Young Investigator and under NSF grant ASC-8715728. The authors acknowledge the suggestions of the referees as well as various colleagues who over the years have pointed out deficiencies of the standard SVD, in particular Augustin Debrulle, Cleve Moler, Beresford Parlett, and G. W. Stewart. This paper was originally presented at Gatlinburg X, Fairfield Glade, Tennessee, in October 1987.