SIAM J. MATRIX ANAL. APPL.
Vol. 23, No. 3, pp. 761–798

MORE ACCURATE BIDIAGONAL REDUCTION FOR COMPUTING THE SINGULAR VALUE DECOMPOSITION*

JESSE L. BARLOW†

Abstract. Bidiagonal reduction is the preliminary stage for the fastest stable algorithms for computing the singular value decomposition (SVD) now available. However, the best-known error bounds on bidiagonal reduction methods on any matrix are of the form

$$A + \delta A = UBV^T,$$
$$\|\delta A\|_2 \leq \varepsilon_M f(m,n)\|A\|_2,$$

where B is bidiagonal, U and V are orthogonal, ε_M is machine precision, and $f(m,n)$ is a modestly growing function of the dimensions of A.

A preprocessing technique analyzed by Higham [*Linear Algebra Appl.*, 309 (2000), pp. 153–174] uses orthogonal factorization with column pivoting to obtain the factorization

$$A = Q \begin{pmatrix} C^T \\ 0 \end{pmatrix} P^T,$$

where Q is orthogonal, C is lower triangular, and P is permutation matrix. Bidiagonal reduction is applied to the resulting matrix C.

To do that reduction, a new Givens-based bidiagonalization algorithm is proposed that produces a bidiagonal matrix B that satisfies $C + \delta C = U(B + \delta B)V^T$ where δB is bounded *componentwise* and δC satisfies a columnwise bound (based upon the growth of the lower right corner of C) with U and V orthogonal to nearly working precision. Once we have that reduction, there is a good menu of algorithms that obtain the singular values of the bidiagonal matrix B to relative accuracy, thus obtaining an SVD of C that can be much more accurate than that obtained from standard bidiagonal reduction procedures. The additional operations required over the standard bidiagonal reduction algorithm of Golub and Kahan [*J. Soc. Indust. Appl. Math. Ser. B Numer. Anal.*, 2 (1965), pp. 205–224] are those for using Givens rotations instead of Householder transformations to compute the matrix V, and $2n^3/3$ flops to compute column norms.

Key words. orthogonal reduction, bidiagonal form, singular values, accuracy

AMS subject classifications. 65F15, 65F35, 15A18

PII. S0895479898343541

1. Introduction.

We consider the problem of reducing $A \in \Re^{m\times n}$ to bidiagonal form. Without loss of generality, assume that $m \geq n$. We find orthogonal matrices $U \in \Re^{n\times n}$ and $V \in \Re^{m\times m}$ such that

$$U^T A^T V = \begin{array}{c} \begin{matrix} n & m-n \end{matrix} \\ \begin{pmatrix} B & 0 \end{pmatrix} \end{array}, \tag{1.1}$$

where

$$B = \begin{pmatrix} \gamma_1 & \phi_2 & 0 & \cdot & \cdot & \cdot \\ 0 & \gamma_2 & \phi_3 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \gamma_{n-1} & \phi_n \\ 0 & \cdot & \cdot & \cdot & 0 & \gamma_n \end{pmatrix}. \tag{1.2}$$

*Received by the editors July 27, 1998; accepted for publication (in revised form) by N.J. Higham September 18, 2001; published electronically January 23, 2002. This research was supported by the National Science Foundation under grants CCR–9424435 and CCR–9732081. Part of this work was done while the author was visiting the Department of Mathematics, University of Manchester, Manchester, UK.

http://www.siam.org/journals/simax/23-3/34354.html

†Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802-6106 (barlow@cse.psu.edu, http://www.cse.psu.edu/∼barlow).

To denote B in (1.2) we use the shorthand

$$B = \text{bidiag}(\gamma_1, \gamma_2, \ldots, \gamma_n; \phi_2, \ldots, \phi_n)$$

or use the MATLAB-like [33] form

$$B = \text{bidiag}(\gamma(1\colon n); \phi(2\colon n)).$$

We will also use MATLAB notation for submatrices. Thus $A(i\colon j, k\colon \ell)$ denotes the submatrix of A consisting of rows i through j and columns k through ℓ. Likewise, $A(:, k\colon \ell)$ denotes all of columns k through ℓ and $A(i\colon j, :)$ denotes all of rows i through j. In some of the error analysis proofs and discussions, we use $(A + B)(i\colon j, k\colon \ell)$ to denote that section of $A + B$.

For a matrix $X \in \Re^{m\times n}$, $m \geq n$, let

$$\sigma_i(X)$$

denote the ith singular value of X and let

$$\sigma_{min}(X) = \min_{\|\mathbf{w}\|_2=1} \|X\mathbf{w}\|_2$$

denote the smallest singular value of X. We let $J(i, j, \theta_{ij})$ be a Givens rotation through an angle θ_{ij} applied to columns i and j, that is, a matrix which is the identity except for the four entries (i, i),(j, i),(i, j), and (j, j) given by $\left(\begin{smallmatrix} cn & sn \\ -sn & cn \end{smallmatrix}\right)$, with $cn = \cos\,\theta_{ij}$ and $sn = \sin\,\theta_{ij}$.

The error analysis bounds are stated as

$$error \leq \varepsilon_M g(n) * factor + O(\varepsilon_M^2).$$

This arise out of simplify expressions of the form

$$(1 + \varepsilon_M)^n factor - 1 = n * \varepsilon_M * factor + O(\varepsilon_M^2 factor) = n * \varepsilon_M * factor + O(\varepsilon_M^2).$$

Thus they would be properly stated as

$$error \leq \varepsilon_M g(n) * factor + O(\varepsilon_M^2 factor),$$

but we leave out "factor" in $O(\varepsilon_M^2)$ to make statements less tedious.

The reduction (1.1) is usually done as a preliminary stage for computing the singular value decomposition (SVD) of A. There are now several very good algorithms for computing the SVD of bidiagonal matrices. We know that the "zero-shift" QR algorithm [14], bisection [3, 20], and the dqds algorithm [21] can compute all of the singular values of B to relative accuracy. We also know that it is not reasonable to expect any algorithm to compute all of the singular values of a matrix to relative accuracy unless that matrix has an acyclic graph [13] or is totally sign compound [12].

Thus, it is not surprising that no algorithm can be expected to produce the bidiagonal form of a general matrix to relative accuracy in finite precision arithmetic.

The Jacobi algorithm [28] has a stronger error bound for finding the singular values of a general matrix than any algorithm that requires bidiagonal reduction [15]. Unfortunately, the Jacobi algorithm is usually slower. For simplicity, assume that $m = n$. Bidiagonal reduction followed by the QR algorithm can produce that SVD in

about $20n^3$ flops. One Jacobi sweep requires about $7n^3$ flops. Thus, for Jacobi to be competitive, it must converge in about three sweeps, and that rarely happens.

We note that efforts to improve the performance of Jacobi have been quite successful [17]. However, there have also been significant improvements in speed in obtaining the singular values and vectors of bidiagonal matrices from the dqds algorithms [21, 19, 18, 35, 36]; thus exact complexity comparisons of Jacobi and methods using bidiagonal reduction are a bit elusive. Nonetheless, as of the present writing, algorithms requiring bidiagonal reduction tend to be significantly faster than Jacobi-based algorithms.

In this paper, we present a bidiagonal reduction method that can be expected to preserve more of the accuracy in the SVD.

The reduction is computed in two stages. In the first stage, discussed in section 3, using a Householder factorization method described by Björck [7, pp. 165–169] and analyzed by Cox and Higham [11], it is possible to reduce A to a lower triangular matrix $C \in \Re^{n\times n}$. In floating point arithmetic with machine precision ε_M, for some permutation matrix P and orthogonal matrix $\bar{V}_0$, the first stage reduction satisfies

$$A + \delta A = \bar{V}_0 \begin{pmatrix} C^T \\ 0 \end{pmatrix} P^T, \tag{1.3}$$

where

$$\|\delta A(i,:)\|_2 \le \varepsilon_M \rho_A f(m,n) \|A(i,:)\|_2 + O(\varepsilon_M^2), \tag{1.4}$$

$$i = 1, \ldots, m.$$

There is also a well-known columnwise error bound given in section 3. Here $f(m,n)$ is a modestly sized function and ρ_A is a growth factor given in [11]. A reduction recommended by Demmel and Veselić [15] applied to $A^T A$ before using the Jacobi method produces the same C (in exact arithmetic). The advantage of the row ordering is that if A is well-conditioned after row scalings, the singular values of A will be preserved more accurately in C (see, for instance, [26], or an analysis can be constructed from the discussion in section 2). The discussion below assumes that $\operatorname{rank}(C) = n$, but as shown in section 3, the case $\operatorname{rank}(C) < n$ requires only a postprocessing procedure from Hanson and Lawson [25], and the rest of the algorithm will proceed in the same manner.

Most of the attention in this paper will be focused upon C. Our goal in the above reduction is similar to that in [15], to produce a matrix C that is well-conditioned under column scalings and then apply an SVD routine that has columnwise backward error.

Therefore in the second stage, we apply a new bidiagonal reduction algorithm to C. We show that (see Theorems 6.1 and 6.2) this algorithm produces a bidiagonal matrix B such that for some $\delta B, \delta C \in \Re^{n\times n}$, some modestly sized functions $g_i(n), i = 1{:}\,3$, and matrices U and V, a growth factor $\hat{\rho}_V^{(k)}$ (specified in (5.10)), we have

$$C + \delta C = U(B + \delta B)V^T, \tag{1.5}$$

$$\|U^T U - I\|_2, \|V^T V - I\|_2 \le g_1(n)\varepsilon_M + O(\varepsilon_M^2), \tag{1.6}$$

$$|\delta B| \le \varepsilon_M g_2(n)|B| + O(\varepsilon_M^2), \quad |\cdot| \text{ and } \le \text{ entry-wise}, \tag{1.7}$$

$$\|\delta C(:,j)\|_2 \le g_3(n)\varepsilon_M \hat{\rho}_V^{(j-1)} \|C(:,j\!:\!n)\|_F + O(\varepsilon_M^2), \tag{1.8}$$

where

$$\hat{\rho}_V^{(j-1)} \le \|C\|_F / \|C(:,j\!:\!n)\|_F$$

and is usually much smaller. The upper bound on $\hat{\rho}_V^{(j-1)}$ is just a statement that standard normwise bounds apply. Moreover, the algorithm can be implemented in less than $2n^3 + O(n^2)$ more flops than the Lawson–Hanson–Chan SVD [30, pp. 107–120], [9]. In the worst case, the algorithm produces the same error as the standard algorithm; in practice, it appears to be much better.

Our procedure for bidiagonal reduction of C has three important differences from the Golub–Kahan Householder-transformation-based procedure [22]:

- Givens transformations are used in the construction of the right orthogonal matrix V. (Clearly, 2×2 Householder transformations could also be used.) (See section 5.1.)
- The computation of the matrices U and V are interleaved in a different manner to preserve accuracy in the small columns. (See again section 5.1.)
- The rotations are accumulated in a different manner from that normally used. (See section 5.2.)

In the next section, we give the perturbation theory that justifies (1.5). The appropriate algorithm for producing the factorization (1.3)–(1.4) is discussed in section 3. A template for bidiagonal reduction procedures and an outline for the development of our procedure is given in section 4. The material in section 5 provides justification for the choices made in creating Algorithm 6.1 in section 6 from Algorithm 4.1 in section 4. The motivation for using Givens rotations, the different manner in which they are accumulated, and the way in which the computation of U and V are interleaved is justified in section 5. That motivation includes the error analysis of one major step of the algorithm (see Lemma 5.7 and Theorem 5.8 in section 5.2). A full description of the algorithm is the subject of section 6, followed by the error bounds (1.5)–(1.8), formally stated as Theorem 6.1 in section 6.2. The tedious proofs of Lemma 5.5 and Theorem 5.8 from section 5.2 are in the technical report [2].

In section 7, we give some numerical tests. The tests seem to indicate that our error bounds on bidiagonal reduction of C in (1.3) are pessimistic. We could find no examples where the singular values obtained by our procedure differ from those of the Jacobi method by more than those from the Jacobi method are expected to differ from the exact singular values. The Golub–Kahan Householder-based procedure also seems to produce very accurate singular values of such matrices very often. However, the changes made in this paper to the algorithm in [22] were needed in our proofs of the results in section 6.2. The effect of these changes is demonstrated in an example in section 5.3. Moreover, we produce a class of examples of lower triangular matrices of the same form as C for which our procedure is more accurate than the Golub–Kahan procedure. Our conclusion is given in section 8.

2. Necessary perturbation theory. We now give some perturbation theory results to motivate obtaining error bounds of the form (1.5)–(1.8).

The following result shows that the nonorthogonality of U and V in (1.5) causes only a small relative change in the singular values. This lemma is given in [27, problem 18, pp. 423–424].

LEMMA 2.1. *Let $A \in \Re^{m\times n}$ and let $B \in \Re^{n\times n}$. Then for $i = 1, 2, \ldots, n$,*

$$\sigma_i(A)\sigma_n(B) \le \sigma_i(AB) \le \sigma_i(A)\sigma_1(B).$$

Standard bounds on eigenvalue perturbation [23, Chapter 8] lead to

$$1 - g_1(n)\varepsilon_M + O(\varepsilon_M^2) \le \sigma_n^2(U) \le \sigma_1^2(U) \le 1 + g_1(n)\varepsilon_M + O(\varepsilon_M^2),$$

$$1 - g_1(n)\varepsilon_M + O(\varepsilon_M^2) \le \sigma_n^2(V) \le \sigma_1^2(V) \le 1 + g_1(n)\varepsilon_M + O(\varepsilon_M^2),$$

where $g_1(n)$ is the function in (1.6). Thus the singular values of $B + \delta B$ represent those of $C + \delta C$ to relative accuracy.

The effect of the bound in (1.7) is addressed by a result of Demmel and Kahan [14].

LEMMA 2.2. *Let* $B = \text{bidiag}(\gamma(1\colon n); \phi(2\colon n)) \in \Re^{n\times n}$, *let* $\tilde{B} \equiv B + \delta B = \text{bidiag}(\tilde{\gamma}(1\colon n); \tilde{\phi}(2\colon n)) \in \Re^{n\times n}$, *and let* $\zeta \ge 1$. *If*

$$\frac{1}{\zeta} \le \frac{\tilde{\gamma}_j}{\gamma_j}, \frac{\tilde{\phi}_i}{\phi_i} \le \zeta, \qquad \begin{array}{l} i = 2, \ldots, n, \\ j = 1, 2, \ldots, n, \end{array}$$

with the convention that $0/0 = 1$, *then*

$$\frac{1}{\zeta^{2n-1}} \le \frac{\sigma_j(\tilde{B})}{\sigma_j(B)} \le \zeta^{2n-1}, \quad j = 1, 2, \ldots, n.$$

Thus in (1.5)–(1.7), the singular values of B represent those of $\tilde{B}$ to relative accuracy.

The important effect is that of the error δC which is bounded in (1.5)–(1.8).

To characterize that error, we make the simplifying assumption that C and $C+\delta C$ are both nonsingular. A relaxation of the assumption that $C + \delta C$ is nonsingular is possible; see Barlow and Slapničar [5].

We define two parameters,

$$\tau = \sup_{\mathbf{x}\neq 0} \frac{\|\delta C\mathbf{x}\|_2}{\|C\mathbf{x}\|_2}, \tag{2.1}$$

$$\hat{\tau} = \max\left\{\tau, \sup_{\mathbf{x}\neq 0} \frac{\|\delta C\mathbf{x}\|_2}{\|(C+\delta C)\mathbf{x}\|_2}\right\}. \tag{2.2}$$

Assuming that $\tau < 1$, we have that

$$\hat{\tau} \le \frac{\tau}{1-\tau}.$$

Two simple lemmas illustrate the relative error in the singular values and vectors of C in terms of τ. This result is proved in Demmel and Veselić [15].

LEMMA 2.3. *Let* $C, \delta C \in \Re^{n\times n}$ *be such that both* C *and* $C + \delta C$ *are nonsingular. Let* τ *be given by* (2.1). *Then the singular values of* C *and* $C + \delta C$ *satisfy*

$$\frac{|\sigma_i(C+\delta C) - \sigma_i(C)|}{\sigma_i(C)} \le \tau.$$

We now bound the error in the vectors in terms of $\hat{\tau}$. This is a generalization of a bound in Barlow and Demmel [3] and an elaboration of a bound in Demmel and

Veselić [15]. The analysis is very similar to that used to bound error in subspaces in ULV decomposition in [6]. This result can be generalized to clusters of singular values in a well-known manner; see, for instance, [32, 5].

LEMMA 2.4. *Assume the hypothesis and terminology of Lemma* 2.3. *Let* $(\sigma_i, \mathbf{y}_i, \mathbf{w}_i)$ *denote the ith singular triplet of* C *and let* $(\tilde{\sigma}_i, \tilde{\mathbf{y}}_i, \tilde{\mathbf{w}}_i)$ *denote the ith singular triplet of* $C + \delta C$ *for* $i = 1, \ldots, n$. *Then we have*

$$|\tilde{\mathbf{w}}_j^T \mathbf{w}_i| \leq \hat{\tau} \frac{2\sigma_i \tilde{\sigma}_j}{|\sigma_i^2 - \tilde{\sigma}_j^2|}, \tag{2.3}$$

and

$$|\tilde{\mathbf{y}}_j^T \mathbf{y}_i| \leq \hat{\tau} \frac{\sigma_i^2 + \tilde{\sigma}_j^2}{|\sigma_i^2 - \tilde{\sigma}_j^2|}. \tag{2.4}$$

Proof. To prove (2.3), we simply use the fact that $\mathbf{w}_i$ is an eigenvector of $C^T C$ and $\tilde{\mathbf{w}}_j$ is an eigenvector of $(C + \delta C)^T (C + \delta C)$. Thus

$$\mathbf{w}_i^T (C + \delta C)^T (C + \delta C) \tilde{\mathbf{w}}_j = \tilde{\sigma}_j^2 \mathbf{w}_i^T \tilde{\mathbf{w}}_j,$$

which leads to

$$(\sigma_i^2 - \tilde{\sigma}_j^2) \mathbf{w}_i^T \tilde{\mathbf{w}}_j = -\mathbf{w}_i^T \delta C^T (C + \delta C) \tilde{\mathbf{w}}_j - \mathbf{w}_i^T C^T \delta C \tilde{\mathbf{w}}_j.$$

The use of the Cauchy–Schwarz inequality and the definition of $\hat{\tau}$ in (2.2) yields

$$\begin{aligned} |\sigma_i^2 - \tilde{\sigma}_j^2| |\mathbf{w}_i^T \tilde{\mathbf{w}}_j| &\leq \|\delta C \mathbf{w}_i\|_2 \|(C + \delta C) \tilde{\mathbf{w}}_j\|_2 + \|C \mathbf{w}_i\|_2 \|\delta C \tilde{\mathbf{w}}_j\|_2. \\ &\leq 2\hat{\tau} \sigma_i \tilde{\sigma}_j. \end{aligned}$$

Thus we have (2.3). A nearly identical derivation from $(C + \delta C)\ (C + \delta C)^T$ yields

$$|\sigma_i^2 - \tilde{\sigma}_j^2| |\mathbf{y}_i^T \tilde{\mathbf{y}}_j| \leq \hat{\tau} (\sigma_i^2 + \tilde{\sigma}_j^2).$$

Thus (2.4) holds. ∎

We note that the bound in the error of the right singular vectors is stronger than that for left singular vectors since

$$2\sigma_i \tilde{\sigma}_j \leq \sigma_i^2 + \tilde{\sigma}_j^2.$$

Adding the two bounds together yields Demmel and Veselić's [15] bound

$$|\tilde{\mathbf{w}}_j^T \mathbf{w}_i| + |\tilde{\mathbf{y}}_j^T \mathbf{y}_i| \leq \hat{\tau} \frac{\sigma_i + \tilde{\sigma}_j}{|\sigma_i - \tilde{\sigma}_j|}.$$

The standard error bound on δC is

$$\|\delta C\|_2 \leq \varepsilon_M g_4(n) \|C\|_2 + O(\varepsilon_M^2) \tag{2.5}$$

for some modestly growing function $g_4(n)$. The advantage of (1.8) over (2.5) is a smaller value of τ in (2.1), leading to more accurate singular values and vectors.

A bound on τ is given by

$$\begin{aligned} \tau &= \sup_{\mathbf{x} \neq 0} \frac{\|\delta C \mathbf{x}\|_2}{\|C \mathbf{x}\|_2} = \sup_{\mathbf{y} \neq 0} \frac{\|\delta C C^{-1} \mathbf{y}\|_2}{\|\mathbf{y}\|_2} \\ &= \|\delta C C^{-1}\|_2. \end{aligned}$$

If the only bound that we have on δC is (2.5), then

$$\|\delta C C^{-1}\|_2 \leq \varepsilon_M g_4(n)\kappa_2(C) + O(\varepsilon_M^2), \tag{2.6}$$

where $\kappa_2(C) = \|C\|_2\|C^{-1}\|_2$.

To see the effect of (1.8) we rewrite it as

$$\delta C = E_C R_C D_C, \quad \|E_C\|_2 \leq g_4(n)\varepsilon_M + O(\varepsilon_M^2), \tag{2.7}$$

where

$$D_C = \mathrm{diag}(\|C(:,1)\|_2, \ldots, \|C(:,n)\|_2), \tag{2.8}$$

$$R_C = \mathrm{diag}(\hat{\rho}_V^{(0)}, \ldots, \hat{\rho}_V^{(n-1)}). \tag{2.9}$$

We define

$$\hat{\rho}_V = \|R_C\|_2 = \max_{0\leq k\leq n-1} \hat{\rho}_V^{(k)}. \tag{2.10}$$

A bound of the form (2.7) yields

$$\|\delta C C^{-1}\|_2 \leq \|E_C R_C D_C C^{-1}\|_2 \leq \varepsilon_M \hat{\rho}_V g_4(n)\kappa_2(C D_C^{-1}) + O(\varepsilon_M^2). \tag{2.11}$$

Van der Sluis [38] showed that D_C satisfies

$$\kappa_2(C D_C^{-1}) \leq \sqrt{n}\min\{\kappa_2(CD^{-1}) : D \text{ diagonal and nonsingular}\}. \tag{2.12}$$

Thus the choice of D_C in (2.8) is within a factor of $\sqrt{n}\hat{\rho}_V$ of optimizing the second bound in (2.11) over all nonsingular diagonal scaling matrices.

Noting that the bound in (1.8) implies that in (2.5) the bound in (2.11) cannot be significantly worse than the one in (2.6). Very often it will be significantly better, since it states that the scaling of the columns can be ignored.

3. Reduction to triangular form. Before performing bidiagonal reduction procedure, we assume that A has been reduced to a lower triangular matrix C. We recommend using a Householder-transformation-based procedure discussed by Björck [7, pp. 165–169] and analyzed by Cox and Higham [11]. For more about the use of this procedure in SVD computation, see Higham [26].

The procedure is as follows.

1. Reorder the rows of A so that
$$\|A(1,:)\|_\infty \geq \cdots \geq \|A(m,:)\|_\infty.$$
2. Using the maximal column pivoting algorithm of Businger and Golub [8], factor A into
$$A = V_0 \begin{pmatrix} C^T \\ 0 \end{pmatrix} P^T, \tag{3.1}$$
where $V_0 \in \Re^{m\times m}$ is orthogonal, P is a permutation matrix, and $C \in \Re^{n\times n}$ is lower triangular.

This particular Householder factorization algorithm has very strong numerical stability properties. If we let C be the computed lower triangular factor, then Cox and Higham [11] (based on the analysis of a more complicated pivoting strategy by Powell and Reid [37]) showed that for some orthogonal matrix $\bar{V}_0$ and some matrix δA, we have

$$A + \delta A = \bar{V}_0 \begin{pmatrix} C^T \\ 0 \end{pmatrix} P^T, \tag{3.2}$$

$$\|\delta A(:,j)\|_2 \le c_1(m,n)\|A(:,j)\|_2 \varepsilon_M, \quad j = 1, 2, \ldots, n, \tag{3.3}$$

and

$$\|\delta A(i,:)\|_2 \le c_2(m,n)\rho_A \|A(i,:)\|_2 \varepsilon_M + O(\varepsilon_M^2), \quad i = 1, \ldots, m, \tag{3.4}$$

where ρ_A is a growth factor bounded by $\sqrt{n}2^{n-1}$ and $c_k(m,n) = O(mn)$, $k = 1, 2$. The column oriented error bound (3.3) holds for standard Householder factorization [39, pp. 152–162]. The second rowwise error bound (3.4) can be shown only for algorithms that do some kind of row and column permutations for stability.

Similar results for Givens-based algorithms have been given [1, 4]. Cox and Higham demonstrate that Householder's original version of Householder transformations must be used for these bounds to hold. The bound does not hold if Parlett's version [34] of the Householder transformation is used.

If the matrix C in (3.2) has rank n, we use that matrix. If $\operatorname{rank}(C) = r < n$, then C has the form

$$C = \begin{array}{c} \\ r \\ n-r \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{pmatrix} C_{11} & 0 \\ C_{12} & 0 \end{pmatrix} \end{array},$$

and we can use a procedure of Hanson and Lawson [25] to produce an orthogonal matrix U_0 from a product of Householder transformations such that

$$U_0^T C = \begin{array}{c} \\ r \\ n-r \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{pmatrix} \tilde{C} & 0 \\ 0 & 0 \end{pmatrix} \end{array}, \tag{3.5}$$

where $\tilde{C}$ remains lower triangular. Our algorithms will use $\tilde{C}$ in place of C, and thus we assume that C is square nonsingular matrix.

Since maximal column pivoting assures us that

$$|C(j,j)| \ge \|C(k, j\!:\!n)\|_2, \qquad \begin{array}{l} j = 1, \ldots, n-1, \\ k = j+1, \ldots, n, \end{array}$$

and C is lower triangular, we have that the columns of the matrix C satisfy

$$\|C(:,j)\|_2 \ge \frac{1}{\sqrt{n-j}} \|C(:, j+1\!:\!n)\|_F, \quad j = 1, 2, \ldots, n. \tag{3.6}$$

If we are using $\tilde{C}$ in (3.5) in place of C, this bound is satisfied for n, not $r = \operatorname{rank}(C)$.

For the rest of this paper, any reduction of A that produces C satisfying the property (3.6) will be a suitable preprocessing step and will lead to the results given here.

We now give a class of algorithms for computing the bidiagonal reduction of C.

4. A template for bidiagonal reduction. The usual bidiagonal reduction algorithm is that given by Golub and Kahan [22, Theorem 1, pp. 208–210]; see also Golub and Reinsch [24, pp. 404–405]. It is given below.

ALGORITHM 4.1 (Golub–Kahan bidiagonal reduction).

1. *Construct an orthogonal transformation U_1 such that*

$$U_1^T C(:,1) = \gamma_1 \mathbf{e}_1,$$

$$C \leftarrow U_1^T C;\ U \leftarrow U_1;\ V \leftarrow V_1 \equiv I;$$

2. ***for*** $k = 2 : n-1$
   (a) *If $C(k-1, k\colon n) \neq 0$, construct an orthogonal transformation V_k such that*

$$V_k^T C(k-1, k\colon n)^T = \phi_k \mathbf{e}_1,$$

$$C(k\colon n, k\colon n) \leftarrow C(k\colon n, k\colon n) V_k, \quad V(:, k\colon n) \leftarrow V(:, k\colon n) V_k.$$

   else $\phi_k = 0$, and $V_k = I$ (implicitly).
   (b) *Construct an orthogonal transformation U_k such that*

$$U_k^T C(k\colon n, k) = \gamma_k \mathbf{e}_1,$$

$$C(k\colon n, k\colon n) \leftarrow U_k^T C(k\colon n, k\colon n), \quad U(:, k\colon n) \leftarrow U(:, k\colon n) U_k.$$

3.

$$\gamma_n \leftarrow C(n,n), \quad \phi_n \leftarrow C(n-1, n).$$

The bidiagonal reduction of C is given by

$$C = UBV^T,$$

where

$$B = \text{bidiag}(\gamma_1, \ldots, \gamma_n; \phi_2, \ldots, \phi_n).$$

Golub and Kahan [22] use Householder transformations to describe this algorithm.

Algorithm 4.1 allows a number of options for its implementation. Most important is how we choose U_k and V_k for each step in the transformation. We will follow the convention in [22] and choose $U_k, k = 1, \ldots, n-1$, to be Householder transformations. In section 5.1, we justify choosing $V_k, k = 2, \ldots, n-1$, to be Givens rotations in standard order and show why the application of U_k and V_k must be interleaved differently from described above. In section 5.2, we justify an unusual method for applying V_k.

5. The use and application of Givens rotations in constructing V_k.

5.1. Eliminating columns from the right and element growth. To formulate a new algorithm for bidiagonal reduction, we consider the effect of the orthogonal transformations used to form V in Algorithm 4.1. Let $U_1, \ldots, U_{n-1}$ and $V_1, \ldots, V_{n-1}$ be the orthogonal transformations from Algorithm 4.1. Define

$$\tilde{U}_k = \hat{U}_1 \cdots \hat{U}_k, \quad \hat{U}_k = \operatorname{diag}(I_{k-1}, U_k), \tag{5.1}$$

$$\tilde{V}_k = \hat{V}_1 \cdots \hat{V}_k, \quad \hat{V}_k = \operatorname{diag}(I_{k-1}, V_k), \quad k = 1, \ldots, n-1. \tag{5.2}$$

Define

$$F^{(k)} = \tilde{U}_k^T C, \quad k = 1, \ldots, n-1.$$

By orthogonal equivalence,

$$\|F^{(k)}\mathbf{e}_i\|_2 = \|C\mathbf{e}_i\|_2.$$

If we let

$$C^{(k)} = F^{(k)}\tilde{V}_k = \tilde{U}_k^T C \tilde{V}_k, \tag{5.3}$$

then $C^{(k)}$ has the form

$$C^{(k)} = \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{pmatrix} C_{11}^{(k)} & C_{12}^{(k)} \\ 0 & C_{22}^{(k)} \end{pmatrix} \end{array},$$

where

$$C_{11}^{(k)} = \operatorname{bidiag}(\gamma(1\colon k); \phi(2\colon k)),$$

and $C_{12}^{(k)}$ is zero except for the last row. Therefore, $\tilde{V}_k$, in effect, zeros out the block $F_{21}^{(k)}$.

The following lemma shows the effect of a large class of orthogonal transformations from the right.

LEMMA 5.1. *Let $F \in \Re^{m\times n}$ and $V \in \Re^{n\times n}$ be partitioned according to*

$$F = \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{pmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{pmatrix} \end{array}, \qquad V = \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \end{array}, \tag{5.4}$$

where V_{11} is nonsingular. Let

$$G = \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{pmatrix} G_{11} & G_{12} \\ 0 & G_{22} \end{pmatrix} \end{array} = FV. \tag{5.5}$$

Then

$$F_{21} = -F_{22}V_{21}V_{11}^{-1}, \quad G_{22} = F_{22}\tilde{V}_{22}, \tag{5.6}$$

where

$$\tilde{V}_{22} = V_{22} - V_{21}V_{11}^{-1}V_{12}. \tag{5.7}$$

Proof. Matching blocks in (5.4) and (5.5) yields

$$F_{21}V_{11} + F_{22}V_{21} = 0, \quad F_{21}V_{12} + F_{22}V_{22} = G_{22}.$$

Using the fact that V_{11} is nonsingular, block Gaussian elimination yields (5.6). ∎

The following generalization of the result of Lemma 5.1 provides a key portion of the proof of Theorem 5.4.

LEMMA 5.2. *Let $F \in \Re^{m\times n}$ be partitioned according to*

$$F = \begin{matrix} \\ k \\ m-k \end{matrix} \begin{matrix} \begin{matrix} k & j-k & n-j \end{matrix} \\ \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \end{pmatrix} \end{matrix}.$$

Let $V = \bar{V}_1\bar{V}_2$, where

$$\bar{V}_1 = \begin{matrix} \\ k \\ j-k \\ n-j \end{matrix} \begin{matrix} \begin{matrix} k & j-k & n-j \end{matrix} \\ \begin{pmatrix} V_{11}^{(1)} & V_{12}^{(1)} & 0 \\ V_{21}^{(1)} & V_{22}^{(1)} & 0 \\ 0 & 0 & I_{n-j} \end{pmatrix} \end{matrix}, \qquad \bar{V}_2 = \begin{matrix} \\ k \\ j-k \\ n-j \end{matrix} \begin{matrix} \begin{matrix} k & j-k & n-j \end{matrix} \\ \begin{pmatrix} V_{11}^{(2)} & 0 & V_{12}^{(2)} \\ 0 & I_{j-k} & 0 \\ V_{21}^{(2)} & 0 & V_{22}^{(2)} \end{pmatrix} \end{matrix},$$

and V has the form in Lemma 5.1*, assume $V_{11} = V_{11}^{(1)}V_{11}^{(2)}$ is nonsingular, and let $\tilde{V}_{22}$ be given by* (5.7)*. Let $G = FV$ be partitioned,*

$$G = \begin{matrix} \\ k \\ m-k \end{matrix} \begin{matrix} \begin{matrix} k & j-k & n-j \end{matrix} \\ \begin{pmatrix} G_{11} & G_{12} & G_{13} \\ 0 & G_{22} & G_{23} \end{pmatrix} \end{matrix}. \tag{5.8}$$

Then

$$G_{23} = F_{23}\tilde{V}_{22}^{(2)}, \tag{5.9}$$

where

$$\tilde{V}_{22}^{(2)} = V_{22}^{(2)} - V_{21}^{(2)}[V_{11}^{(2)}]^{-1}V_{12}^{(2)}$$

and $\|\tilde{V}_{22}^{(2)}\|_2 \le \|\tilde{V}_{22}\|_2$.

Proof. We note that

$$V = \bar{V}_1\bar{V}_2 = \begin{matrix} \\ k \\ j-k \\ n-j \end{matrix} \begin{matrix} \begin{matrix} k & j-k & n-j \end{matrix} \\ \begin{pmatrix} V_{11}^{(1)}V_{11}^{(2)} & V_{12}^{(1)} & V_{11}^{(1)}V_{12}^{(2)} \\ V_{21}^{(1)}V_{11}^{(2)} & V_{22}^{(1)} & V_{21}^{(1)}V_{12}^{(2)} \\ V_{21}^{(2)} & 0 & V_{22}^{(2)} \end{pmatrix} \end{matrix}.$$

If we partition V according to (5.4), then

$$V_{11} = V_{11}^{(1)}V_{11}^{(2)}, \qquad V_{12} = \begin{pmatrix} V_{12}^{(1)} & V_{11}^{(1)}V_{12}^{(2)} \end{pmatrix},$$

$$V_{21} = \begin{pmatrix} V_{21}^{(1)} V_{11}^{(2)} \\ V_{21}^{(2)} \end{pmatrix}, \qquad V_{22} = \begin{pmatrix} V_{22}^{(1)} & V_{21}^{(1)} V_{12}^{(2)} \\ 0 & V_{22}^{(2)} \end{pmatrix}.$$

First we note that V_{11} is nonsingular if and only if $V_{11}^{(i)}, i = 1, 2$, are nonsingular. Evaluating $\tilde{V}_{22}$ according to (5.7) leads to

$$\tilde{V}_{22} = \begin{pmatrix} V_{22}^{(1)} - V_{21}^{(1)}[V_{11}^{(1)}]^{-1} V_{12}^{(1)} & 0 \\ -V_{21}^{(2)}[V_{11}^{(2)}]^{-1}[V_{11}^{(1)}]^{-1} V_{12}^{(1)} & V_{22}^{(2)} - V_{21}^{(2)}[V_{11}^{(2)}]^{-1} V_{12}^{(2)} \end{pmatrix}.$$

Thus $\|\tilde{V}_{22}^{(2)}\|_2 \le \|\tilde{V}_{22}\|_2$. Now we have that

$$F^{(1)} = F\bar{V}_1 = \begin{matrix} \\ k \\ m-k \end{matrix} \begin{matrix} \begin{matrix} k & j-k & n-j \end{matrix} \\ \begin{pmatrix} F_{11}^{(1)} & F_{12}^{(1)} & F_{13} \\ F_{21}^{(1)} & F_{22}^{(1)} & F_{23} \end{pmatrix} \end{matrix},$$

where the (1,3) and (2,3) blocks of F are unaffected. We then apply Lemma 5.1 to

$$\begin{pmatrix} F_{11}^{(1)} & F_{13} \\ F_{21}^{(1)} & F_{23} \end{pmatrix} \begin{pmatrix} V_{11}^{(2)} & V_{12}^{(2)} \\ V_{21}^{(2)} & V_{22}^{(2)} \end{pmatrix} = \begin{pmatrix} G_{11} & G_{13} \\ 0 & G_{23} \end{pmatrix}$$

to obtain (5.9). ∎

The following lemma gives us an alternative method for bounding the growth of the $(2,3)$ block.

LEMMA 5.3. *Assume the hypothesis and terminology of Lemma* 5.2. *Assume also that V is orthogonal. Then*

$$\|G_{23}\|_2 \le \rho_V^{(k)} \|F_{23}\|_2, \quad \rho_V^{(k)} \equiv \|\tilde{V}_{22}\|_2 = \left(1 + \|V_{21}V_{11}^{-1}\|_2^2\right)^{1/2}, \tag{5.10}$$

$$\sigma_{min}(\tilde{V}_{22}) \ge 1.$$

Proof. From (5.7), we have

$$\begin{matrix} \\ k \\ \end{matrix} \begin{matrix} \begin{matrix} k & n-k \end{matrix} \\ \begin{pmatrix} 0 & \tilde{V}_{22} \end{pmatrix} \end{matrix} = \begin{pmatrix} 0 & V_{22} - V_{21}V_{11}^{-1}V_{12} \end{pmatrix} = \begin{pmatrix} -V_{21}V_{11}^{-1} & I_{n-k} \end{pmatrix} V.$$

By orthogonal equivalence

$$\|\tilde{V}_{22}\|_2 = \left\| \begin{pmatrix} -V_{21}V_{11}^{-1} & I_{n-k} \end{pmatrix} \right\|_2 = \left(1 + \|V_{21}V_{11}^{-1}\|_2^2\right)^{1/2}.$$

This equivalence is also true for all of the other singular values of $\tilde{V}_{22}$, and thus

$$\sigma_{min}(\tilde{V}_{22}) = \sigma_{min}[\begin{pmatrix} -V_{21}V_{11}^{-1} & I_{n-k} \end{pmatrix}^T] \ge 1.$$

From (5.9) and the result of Lemma 5.2, we have (5.10). ∎

The value of $\hat{\rho}_V^{(k)}$ used in (1.8) and again in (2.9) is given by

$$\hat{\rho}_V^{(k)} = \min\{\rho_V^{(k)}, \|C\|_F / \|C(:, k+1:n)\|_F\}. \tag{5.11}$$

We now assume that the kth right orthogonal transformation $\hat{V}_k$ has the form

$$\hat{V}_k = V_{k,k+1} \cdots V_{k,n}, \quad k = 1, \ldots, n-1, \tag{5.12}$$

where $V_{k,j} = J(k, j, \theta_{k,j})$ is a Givens rotation. The rotation $V_{k,j}$ sets entry $(k-1, j)$ of C (at step k of the algorithm) to zero. We note that we could also assume the use of 2×2 Householder transformations without changing any of the results below. The following is a key result on the growth of the elements of C through the course of Algorithm 4.1.

THEOREM 5.4. *Let Algorithm* 4.1 *be applied to* C *with matrix* V_k *given by the product of Givens rotations in* (5.12). *For* $k = 1, \ldots, n-1$, *let* $\tilde{U}_k$ *be defined by* (5.1), *let* $\tilde{V}_k$ *be defined by* (5.2), *and let* $C^{(k)}$ *be defined by* (5.3). *Let* $\rho_V^{(k)}$ *be defined as in Lemma* 5.3 *and let* $\hat{\rho}_V^{(j-1)}$ *be as defined in* (5.11). *Then for* $k = 1, \ldots, n-1$, *and* $j = k+1, \ldots, n$, *we have*

$$\|C^{(k)}(k+1\colon n, j\colon n)\|_F \le \min\{\rho_V^{(k)}, \hat{\rho}_V^{(j-1)}\} \|C(:, j\colon n)\|_F. \tag{5.13}$$

Proof. Since $\tilde{V}_k$ is the product of Givens rotations in the standard order, we directly apply the results in Lemma 5.2.

Since $\tilde{V}_k = \prod_{i=1}^{k} \prod_{\ell=i+1}^{n} V_{i,\ell}$, for each $j > k$, taking advantage of rotations that commute, we can write

$$\tilde{V}_k = \bar{V}_1 \bar{V}_2,$$

where

$$\bar{V}_1 = \prod_{i=1}^{k} \prod_{\ell=i+1}^{j-1} V_{i,\ell}, \quad \bar{V}_2 = \prod_{i=1}^{k} \prod_{\ell=j}^{n} V_{i,\ell}.$$

Thus, $\bar{V}_1$ and $\bar{V}_2$ have the structure in the hypothesis of Lemma 5.2. Using the terminology of that lemma, we have

$$C^{(k)}(k+1\colon n, j\colon n) = F^{(k)}(k+1\colon n, j\colon n)\tilde{V}_{22}^{(2)} \tag{5.14}$$

and that

$$\begin{aligned} \|C^{(k)}(k+1\colon n, j\colon n)\|_F &\le \|F^{(k)}(k+1\colon n, j\colon n)\|_F \|\tilde{V}_{22}^{(2)}\|_2 \\ &\le \|C(k+1\colon n, j\colon n)\|_F \|\tilde{V}_{22}^{(2)}\|_2 \le \rho_V^{(k)} \|C(:, j\colon n)\|_F, \end{aligned}$$

which is the first half of (5.13). To show that the above bound holds with $\hat{\rho}_V^{(j-1)}$, assume $j > k+1$. We note that

$$\tilde{V}_{j-1} = Z_1 Z_2,$$

where

$$Z_1 = \prod_{i=1}^{j-2} \prod_{\ell=i+1}^{j-1} V_{i,\ell}, \quad Z_2 = \prod_{i=1}^{j-1} \prod_{\ell=j}^{n} V_{i,\ell},$$

and

$$Z_1 = \begin{array}{c} \\ j-1 \\ n-j+1 \end{array} \begin{array}{c} \begin{array}{cc} j-1 & n-j+1 \end{array} \\ \begin{pmatrix} Z_{11}^{(1)} & 0 \\ 0 & I \end{pmatrix} \end{array}, \qquad Z_2 = \begin{array}{c} \\ j-1 \\ n-j+1 \end{array} \begin{array}{c} \begin{array}{cc} j-1 & n-j+1 \end{array} \\ \begin{pmatrix} Z_{11}^{(2)} & Z_{12}^{(2)} \\ Z_{21}^{(2)} & Z_{22}^{(2)} \end{pmatrix} \end{array}.$$

It is easily shown that

$$\rho_V^{(j-1)} = \|\tilde{Z}_{22}\|_2 = \|\tilde{Z}_{22}^{(2)}\|_2.$$

Thus we need only show that $\|\tilde{Z}_{22}^{(2)}\|_2 \geq \|\tilde{V}_{22}^{(2)}\|_2$.

We have that

$$Z_2 = \bar{V}_2 W,$$

where

$$W = \prod_{i=k+1}^{j-1} \prod_{\ell=j}^{n} V_{i,\ell}.$$

Here W has the structure

$$W = \begin{array}{c} \\ k \\ j-k-1 \\ n-j+1 \end{array} \begin{array}{c} \begin{array}{ccc} k & j-k-1 & n-j+1 \end{array} \\ \begin{pmatrix} I & 0 & 0 \\ 0 & W_{11} & W_{12} \\ 0 & W_{21} & W_{22} \end{pmatrix} \end{array}.$$

Therefore the blocks of Z_2 may be written

$$Z_{11}^{(2)} = \begin{pmatrix} V_{11}^{(2)} & V_{12}^{(2)} W_{21} \\ 0 & W_{11} \end{pmatrix}, \quad Z_{12}^{(2)},$$

$$Z_{21}^{(2)} = \begin{pmatrix} V_{21}^{(2)} & V_{22}^{(2)} W_{21} \end{pmatrix}, \quad Z_{22}^{(2)} = V_{22}^{(2)} W_{22}.$$

Some algebra shows that

$$\tilde{Z}_{22}^{(2)} = Z_{22}^{(2)} - Z_{21}^{(2)} [Z_{11}^{(2)}]^{-1} Z_{12}^{(2)}$$

$$= \tilde{V}_{22}^{(2)} \tilde{W}_{22}.$$

We have that

$$\rho_V^{(j-1)} = \|\tilde{Z}_{22}^{(2)}\|_2 \geq \|\tilde{V}_{22}^{(2)}\|_2 \sigma_{min}(\tilde{W}_{22}).$$

By Lemma 5.3, $\sigma_{min}(\tilde{W}_{22}) \geq 1$, so

$$\rho_V^{(j-1)} \geq \|\tilde{V}_{22}^{(2)}\|_2.$$

Using (5.14), we get

$$\|C^{(k)}(k+1\colon n, j\colon n)\|_F \leq \rho_V^{(j-1)} \|C(\colon, j\colon n)\|_F.$$

Since orthogonal equivalence gives us

$$\|C^{(k)}(k+1\colon n, j\colon n)\|_F \leq \|C\|_F$$

we have (5.13). ∎

As will be shown later in Theorem 6.1, if we can control the growth of the elements in the lower corner $C^{(k)}$, then we can obtain a columnwise backward error bound on our implementation of Algorithm 4.1.

Immediately, we make two changes to Algorithm 4.1. The first is the use of Givens rotations in standard order to compose the transformations $V_k, k = 1, \ldots, n-1$, as in (5.12). The second is to note that for the entries of $C(k+1\colon n, k+1\colon n)$ to never exceed the bound on $C^{(k)}(k+1\colon n, k+1\colon n)$ in (5.13), $C^{(k)}$ should be computed by the recurrence

$$C^{(0)} = C, \tag{5.15}$$

$$M^{(k)} \equiv \hat{U}_k^T C^{(k-1)}, \quad k = 1, \ldots, n-1, \tag{5.16}$$

$$C^{(k)} = M^{(k)} \hat{V}_k. \tag{5.17}$$

Thus, even though V_k is formulated before U_k, it must be applied to C after U_k.

If we use the order of application in Algorithm 4.1, then we would produce

$$N^{(k)} \equiv C^{(k-1)} V_k,$$

$$C^{(k)} = U_k^T N^{(k)},$$

but there is the possibility that

$$\|N^{(k)}(k+1\colon n, j\colon n)\|_F \gg \|C^{(k)}(k+1\colon n, j\colon n)\|_F \tag{5.18}$$

for some $j > k \geq 1$, and our error analysis depends upon the ability to bound the growth in these values. In the example in section 5.3, we show that exactly this phenomenon occurs and that it can affect the accuracy of the small singular values.

The main loop of Algorithm 4.1 is modified as follows. If $C(k-1, k\colon n) \neq 0$, construct V_k such that

$$V_k^T C(k-1, k\colon n)^T = \phi_k \mathbf{e}_1,$$

$$\mathbf{y}_k = C(k\colon n, k\colon n) V_k \mathbf{e}_1.$$

Find U_k such that

$$U_k^T \mathbf{y}_k = \gamma_k \mathbf{e}_1, \tag{5.19}$$

$$C(k\colon n, k\colon n) \leftarrow U_k^T C(k\colon n, k\colon n),$$

$$C(k\colon n, k\colon n) \leftarrow C(k\colon n, k\colon n) V_k.$$

To ensure that $C^{(k)}(k\colon n, k) = \gamma_k \mathbf{e}_1$ for some γ_k with appropriate backward error, we modify the manner in which the orthogonal transformation V_k is applied to C. Those modifications are discussed in the next section.

5.2. Application of the U_k and V_k matrices. We now explain how to implement one step of Algorithm 4.1 in the order (5.16)–(5.17) while preserving the identity

$$C^{(k)}(k\colon n, k) = \gamma_k \mathbf{e}_1. \tag{5.20}$$

We assume that $|\phi_k| = \|C^{(k-1)}(k-1, k\colon n)\|_2 \neq 0$, otherwise $V_k = I$, and the steps in this section are unnecessary.

Let $M^{(k)}$ be as defined in (5.16). We also define

$$\mathbf{v}_1^{(k)} = V_k \mathbf{e}_1 \equiv (v_{kk}^{(k)}, \ldots, v_{nk}^{(k)})^T. \tag{5.21}$$

Since, by assumption, $C^{(k-1)}(k-1, k\colon n) \neq 0$, $\mathbf{v}_1^{(k)} = C^{(k-1)}(k-1, k\colon n)^T/\phi_k$, this first column will determine the Givens rotations $V_{k,k+1}, \ldots, V_{k,n}$. We adopt three conventions in our discussions in this section:

- To avoid greater notational complexity, we assume that $M^{(k)}$ and $C^{(k)}$ are the computed values in (5.16)–(5.17) rather than the exact ones. We also use this assumption throughout section 6.
- The dimensions of the matrix $V_{k,j}$ will be different in different contexts, but will always refer to a Givens rotation whose nontrivial portion is applied to columns k and j of C.
- We ignore the rounding errors in applying the rotations in $\hat{V}_k$ to $M^{(k)}$, but not the errors in forming the rotations. Theorem 5.8, at the end of this section, shows that assumption can be lifted and still obtain the necessary error bounds.

The important concern in (5.20) is to maintain

$$C^{(k)}(k+1\colon n, k) = 0, \tag{5.22}$$

and thus row k of $C^{(k)}$ may be computed according to

$$C^{(k)}(k, k\colon n) = M^{(k)}(k, k\colon n) V_{k,k+1} \cdots V_{k,n} \tag{5.23}$$

in the standard manner. Our discussion will center upon how to compute rows $k+1$ through n of $C^{(k)}$.

In exact arithmetic, the statement (5.20) implies that

$$M^{(k)}(k\colon n, k\colon n) V_k \mathbf{e}_1 = M^{(k)}(k\colon n, k\colon n) \mathbf{v}_1^{(k)} = \gamma_k \mathbf{e}_1 \tag{5.24}$$

for some γ_k. Unfortunately, since U_k and V_k are applied in the opposite order from which they are defined in (5.15)–(5.17), we do not expect (5.24) to hold in floating point arithmetic.

We begin the following lemma which is proven in [2]. It shows that a columnwise perturbation of $M^{(k)}$ satisfies (5.24). That result is a building block for developing our procedure for computing $C^{(k)}$ from $M^{(k)}$.

LEMMA 5.5. *Let $M^{(k)} \in \Re^{n\times n}$ be the computed result of applying the Householder transformation U_k to $C^{(k-1)}$ in* (5.16). *Then for some $\delta M_0^{(k)} \in \Re^{n\times n}$ and modestly growing function $g_0(n)$ we have that*

$$(M^{(k)} + \delta M_0^{(k)})(k\colon n, k\colon n) \mathbf{v}_1^{(k)} = \gamma_k \mathbf{e}_1,$$

where, for $j = k, \ldots, n$,

$$\|\delta M_0^{(k)}(:, j)\|_2 \leq \varepsilon_M g_0(n-k+1)\|M^{(k)}(k{:}\,n, j)\|_2 + O(\varepsilon_M^2)$$

and

$$g_0(n) = n^2 + 14n. \tag{5.25}$$

The assumption that U_k is a Householder transformation determines the function $g_0(n)$, but any stable implementation of orthogonal transformations such that (5.19) holds will yield this bound with a different modestly growing function $g_0(n)$.

We note that Lemma 5.5 gives us that

$$M^{(k)}(k{:}\,n, k{:}\,n)\mathbf{v}_1^{(k)} = \gamma_k \mathbf{e}_1 - \delta\mathbf{m}_1, \quad \delta\mathbf{m}_1 = \delta M_0^{(k)}(k{:}\,n, k{:}\,n)\mathbf{v}_1^{(k)}. \tag{5.26}$$

When applying V_k to $M^{(k)}$ the computation (5.26) is not actually performed. Instead we accept $\gamma_k \mathbf{e}_1$ as the correct result for $C^{(k)}(k{:}\,n, k)$, thereby enforcing (5.20).

However, (5.26) tells us that the exact relation between $M^{(k)}$ and $C^{(k)}$ is

$$M^{(k)}(k{:}\,n, k{:}\,n) - \delta\mathbf{m}_1[\mathbf{v}_1^{(k)}]^T = C^{(k)}(k{:}\,n, k{:}\,n)V_k^T$$

or

$$(M^{(k)} + \delta M_1^{(k)})(k{:}\,n, k{:}\,n) = C^{(k)}(k{:}\,n, k{:}\,n)V_k^T, \tag{5.27}$$

where

$$\delta M_1^{(k)}(k{:}\,n, k{:}\,n) = \delta M_0^{(k)}(k{:}\,n, k{:}\,n)\mathbf{v}_1^{(k)}[\mathbf{v}_1^{(k)}]^T.$$

Thus applying $\hat{V}_k$ to $M^{(k)}$ and enforcing (5.20) creates the backward error $\delta M_1^{(k)}$ (*not* $\delta M_0^{(k)}$). Unfortunately, $\delta M_1^{(k)}$ has only the normwise bound

$$\|\delta M_1^{(k)}\|_F \leq \|\delta M_0^{(k)}\|_F \leq \varepsilon_M g_0(n-k+1)\|M^{(k)}\|_F + O(\varepsilon_M^2). \tag{5.28}$$

From the analysis of section 2*, we need columnwise error bounds; thus* (5.28) *is not acceptable.*

Instead, we compute with a matrix $C^{(k,k)}$ such that

$$C^{(k,k)} \equiv M^{(k)} + \delta M_2^{(k)}, \tag{5.29}$$

$$C^{(k,k)}(k+1{:}\,n, k{:}\,n)\mathbf{v}_1^{(k)} = 0, \tag{5.30}$$

and for a modestly growing function $\tilde{g}_0(n)$,

$$\delta M_2^{(k)} = \delta\tilde{\mathbf{m}}_s \mathbf{e}_s^T, \tag{5.31}$$

$$\|\delta\tilde{\mathbf{m}}_s\|_2 \leq \varepsilon_M \tilde{g}_0(n-k+1)\|M^{(k)}(k+1{:}\,n, s)\|_2 + O(\varepsilon_M^2) \tag{5.32}$$

for some $s \in \{k, k+1, \ldots, n\}$. Thus $C^{(k,k)}(:, j) = M^{(k)}(:, j)$ for $j \neq s$, and

$$C^{(k,k)}(:, s) = M^{(k)}(:, s) + \delta\tilde{\mathbf{m}}_s. \tag{5.33}$$

There is no reason to perturb rows $1, 2, \ldots, k$, since there we accept the computed result of (5.16) and (5.17), so $\delta\tilde{\mathbf{m}}_s(1\colon k) = 0$.

We then obtain $C^{(k)}$ from the computation

$$C^{(k,j)} \equiv C^{(k,j-1)} V_{k,j}, \quad j = k+1, \ldots, n, \tag{5.34}$$

$$C^{(k)} = C^{(k,n)} = C^{(k,k)} V_k = (M^{(k)} + \delta M_2^{(k)}) V_k.$$

The perturbation $\delta M_2^{(k)}$ introduces only a columnwise backward error but guarantees (5.20). Below, we show how to obtain a $C^{(k,k)}$ that satisfies (5.30)–(5.32), and thereby a procedure to obtain $C^{(k-1)}$ from $C^{(k)}$ with an appropriate error bound.

First, we show how to get $C^{(k,k)}(k+1\colon n, s)$ for any $s \in \{k, \ldots, n\}$. We then show how to choose s to enforce (5.32). To construct $C^{(k,k)}$, we enforce (5.30). Since

$$C^{(k,j)} = C^{(k,k)} V_{k,k+1} \cdots V_{k,j}, \quad j = k+1, \ldots, n,$$

the equation (5.30) is equivalent to the statement

$$\begin{pmatrix} C^{(k,j)}(k+1\colon n, k) & C^{(k,k)}(k+1\colon n, j\colon n) \end{pmatrix} V_{k,j+1} \cdots V_{k,n} \tag{5.35}$$

$$= \begin{pmatrix} 0 & C^{(k)}(k+1\colon n, j\colon n) \end{pmatrix} \tag{5.36}$$

for $j = k+1, \ldots, n$.

In terms of the components of the Givens rotations, column k of recurrence (5.35)–(5.36) is written

$$C^{(k,j)}(k+1\colon n, k)$$

$$= (cn)_{k,j} C^{(k,j-1)}(k+1\colon n, k) + (sn)_{k,j} C^{(k,k)}(k+1\colon n, j), \tag{5.37}$$

$$j = k+1, \ldots, n,$$

with $C^{(k,n)} = C^{(k)}$ and $C^{(k,n)}(k+1\colon n, k) = 0$.

Writing the recurrence (5.37) in reverse we have

$$C^{(k,n)}(k+1\colon n, k) = C^{(k)}(k+1\colon n, k) = 0, \tag{5.38}$$

$$C^{(k,j-1)}(k+1\colon n, k)$$

$$= (C^{(k,j)}(k+1\colon n, k) - (sn)_{k,j} C^{(k,k)}(k+1\colon n, j))/(cn)_{k,j}, \tag{5.39}$$

$$j = n, \ldots, k+1.$$

To obtain the value of $C^{(k,k)}(k+1\colon n, s)$ we need both the recurrence (5.37) and the recurrence (5.38)–(5.39). We consider two separate cases, $s = k$ and $s > k$.

For $s = k$, we construct $C^{(k,j)}(k+1\colon n, k)$, $j = n, \ldots, k$, using (5.38)–(5.39), thus constructing $C^{(k,k)}(k+1\colon n, k)$.

For $s > k$, we construct $C^{(k,j)}(k+1\colon n,k), j = k,\dots,s-1$, using (5.37), then construct $C^{(k,j)}(k+1\colon n,k), j = n,\dots,s$, using (5.38)–(5.39). Then compute $C^{(k,k)}(k+1\colon n,s)$ from

$$C^{(k,k)}(k+1\colon n,s) = C^{(k,s-1)}(k+1\colon n,s)$$

$$= (C^{(k,s)}(k+1\colon n,k) - (cn)_{k,s}C^{(k,s-1)}(k+1\colon n,k))/(sn)_{k,s}. \tag{5.40}$$

For both cases, the jth column of $C^{(k)}$ is computed from

$$C^{(k)}(k+1\colon n,j) = (cn)_{k,j}C^{(k,k)}(k+1\colon n,j) - (sn)_{k,j}C^{(k,j-1)}(k+1\colon n,k) \tag{5.41}$$

when both terms of (5.41) are available.

The value of $C^{(k,k)}(k\colon n,s)$ may not satisfy (5.32) and (5.33) for every value of s. Fortunately, as Proposition 5.6 shows, there is always at least one value of s such that (5.32) and (5.33) are enforced.

PROPOSITION 5.6. *Assume the hypothesis and terminology of Lemma* 5.5. *Let* $C^{(k,k)}$ *be defined by* (5.29) *and satisfy* (5.30). *If* s *is chosen so that*

$$\|M^{(k)}(k\colon n,s)\|_2|v_{sk}^{(k)}| = \max_{k\le j\le n}\|M^{(k)}(k\colon n,j)\|_2|v_{jk}^{(k)}|, \tag{5.42}$$

then $\delta\tilde{\mathbf{m}}_s$ *satisfies* (5.32) *with* $\tilde{g}_0(n) = ng_0(n)$. *With this choice, excluding the rounding error from applying the Givens rotations, the computed matrix* $C^{(k)}$ *satisfies*

$$C^{(k)} = (M^{(k)} + \delta M_2^{(k)})V_k,$$

where

$$\|\delta M_2^{(k)}(:,j)\|_2 \le \begin{cases} \tilde{g}_0(n-k+1)\varepsilon_M\|M_2^{(k)}(:,s)\|_2 + O(\varepsilon_M^2), & j = s,\\ 0, & j \ne s. \end{cases} \tag{5.43}$$

Proof. From Lemma 5.5 and (5.30) we must have

$$\delta M_0^{(k)}(k+1\colon n,k\colon n)\mathbf{v}_1^{(k)} = \delta M_2^{(k)}(k+1\colon n,k\colon n)\mathbf{v}_1^{(k)},$$

thus

$$\sum_{j=k}^{n}\delta M_0^{(k)}(k+1\colon n,j)v_{jk}^{(k)} = \delta\tilde{\mathbf{m}}_s(k+1\colon n)\mathbf{e}_{s-k+1}^T\mathbf{v}_1^{(k)} = \delta\tilde{\mathbf{m}}_s(k+1\colon n)v_{sk}^{(k)}.$$

Therefore,

$$\|\delta\tilde{\mathbf{m}}_s(k+1\colon n)\|_2|v_{sk}^{(k)}| \le \sum_{j=k}^{n}|v_{jk}^{(k)}|\|\delta M^{(k)}(k+1\colon n,j)\|_2 \tag{5.44}$$

$$\le \varepsilon_M g_0(n-k+1)\sum_{j=k}^{n}|v_{jk}^{(k)}|\|M^{(k)}(k\colon n,j)\|_2 + O(\varepsilon_M^2). \tag{5.45}$$

If we choose s so that (5.42) holds, then (5.45) becomes

$$\|\delta\tilde{\mathbf{m}}_s(k+1\colon n)\|_2 \le \varepsilon_M\tilde{g}_0(n-k+1)\|M^{(k)}(k+1\colon n,s)\|_2 + O(\varepsilon_M^2), \tag{5.46}$$

which satisfies the second expression in (5.31) with $\tilde{g}_0(n) = ng_0(n)$. Thus $\delta M_2^{(k)}$ satisfies (5.43). ∎

Note that Proposition 5.6 requires the computation of the column norms

$$\|M^{(k)}(k\colon n, j)\|_2, \quad j = k, \ldots, n. \tag{5.47}$$

These must be recomputed for each k, thus adding $2(n-k)^2 + O(n)$ flops at each step of the bidiagonal reduction algorithm.

Our procedure for applying V_k is summarized as follows.

ALGORITHM 5.1 (algorithm for applying V_k).

1. *Compute the column norms in* (5.47)*, then determine s that satisfies* (5.42).
2. *Compute $C^{(k)}(k, k\colon n) = M^{(k)}(k, k\colon n)V_k$ in the ordinary manner.*
3. *If $s = k$, compute all $C^{(k,j)}(k+1\colon n, k)$ using the backward recurrence* (5.38)–(5.39)*, thus implicitly replacing $M^{(k)}(k+1\colon n, k)$ with $C^{(k,k)}(k+1\colon n, k)$.*
4. *If $s \neq k$, for $j = k, \ldots, s-1$ compute $C^{(k,j)}(k+1\colon n, j)$ using the forward recurrence* (5.37)*. We then compute $C^{(k,k)}(k+1\colon n, s) = M^{(k)}(k+1\colon n, s) + \delta\tilde{\mathbf{m}}_s$ from* (5.40).
5. *For either* (3) *or* (4)*, compute $C^{(k)}(k+1\colon n, k+1\colon n)$ according to* (5.41) *as the appropriate columns are available.*

The development above provides most of a proof of the following lemma.

LEMMA 5.7. *Let $C^{(k)}, M^{(k)} \in \Re^{n\times n}, k = 1, \ldots, n-1$, be as defined in* (5.16)–(5.17) *and let $C^{(k,j)}, j = k+1, \ldots, n$, be as defined in* (5.34)*. Excluding the rounding error in applying the Givens rotations in V_k to $C^{(k,k)}$, Algorithm* 5.1 *produces a matrix $C^{(k)}$ such that the matrix satisfies* (5.24) *and*

$$C^{(k)}(k\colon n, k\colon n) = U_k^T[(C^{(k-1)} + \delta C_1^{(k-1)})(k\colon n, k\colon n)]V_k,$$

$$\|\delta C_1^{(k-1)}(k\colon n, j)\|_2 \leq \varepsilon_M h(n-k+1)\|C^{(k-1)}(k\colon n, j)\|_2 + O(\varepsilon_M^2),$$

where $h(n) = ng_0(n) + 12n$ and $g_0(n)$ is defined in Lemma 5.5.

Proof. An interpretation of the error bounds due to Wilkinson [39, pp. 152–162, 236] on the formation and application of a single Householder transformation gives us that $M^{(k)}$ and $C^{(k-1)}$ satisfy

$$(C^{(k-1)} + \delta C_0^{(k-1)})(k\colon n, k\colon n) = U_k M^{(k)}(k\colon n, k\colon n),$$

where

$$\|\delta C_0^{(k-1)}(k\colon n, j)\|_2 \leq 12(n-k+1)\varepsilon_M\|C^{(k-1)}(k\colon n, j)\|_2.$$

Since

$$C^{(k)}(k\colon n, k\colon n) = C^{(k,k)}(k\colon n, k\colon n)V_k = (M^{(k)} + \delta M_2^{(k)})(k\colon n, k\colon n)V_k,$$

we have

$$C^{(k)}(k\colon n, k\colon n) = U_k^T[(C^{(k-1)} + \delta C_1^{(k-1)})(k\colon n, k\colon n)]V_k,$$

where

$$\delta C_1^{(k-1)}(k\colon n, k\colon n) = \delta M_0^{(k)}(k\colon n, k\colon n) + U_k\delta M_2^{(k)}(k\colon n, k\colon n).$$

procedure *compute_pivot*$(C, n, \mathbf{v}, \mathbf{cn}, \mathbf{sn}, s)$

```
%
% Input arguments -- v -- First column of orthogonal transformation.
%                    C -- matrix to be transformed
%                    n -- dimension of C, C is n x n
%
% Output arguments -- cn,sn -- defining values for Givens rotations
%                         s -- column to be replaced implicitly in
%                              orthogonal transformations
```

$maxval = |v(1)| * \|C(:,1)\|_2$; $s = 1$;
for $j = 2{:}\, n$
  $newval = |v(j)| * \|C(:,j)\|_2$;
  $rotg(v(1), v(j), \mathbf{cn}(j-1), \mathbf{sn}(j-1))$;
  if $newval > maxval$
    $maxval = newval$;
    $s = j$;
  end;
end;
end*compute_pivot*

FIG. 5.1. *The compute_pivot procedure.*

Thus from Lemma 5.5, Proposition 5.6, and orthogonal equivalence, we conclude that

$$\|\delta C_1^{(k-1)}(k{:}\, n, j)\|_2 \le \|\delta C_0^{(k-1)}(k{:}\, n, j)\|_2 + \|\delta M_2^{(k)}(k{:}\, n, j)\|_2$$

$$\le \varepsilon_M[(n-k+1)g_0(n-k+1) + 12(n-k+1)]\|C^{(k-1)}(k{:}\, n, j)\|_2 + O(\varepsilon_M^2),$$

thereby establishing the necessary result □

We encapsulate Algorithm 5.1 into two procedures given in a MATLAB-like pseudolanguage. The first, called *compute_pivot* (see Figure 5.1), computes the rotations and the value s, thus doing step 1 of Algorithm 5.1. The second, called *rot_back* (see Figure 5.2), implements the above procedure for applying the Givens rotations, thus doing steps 2–5 of Algorithm 5.1.

Unlike MATLAB, we have procedures, and arguments are called by reference.

REMARK 5.1. *We also use routines* rotg *and* rot*, which correspond to the BLAS* 1 *routines of the same names that generate and apply Givens rotations* [31, 16]. *The call* rotg(a, b, cn, sn) *inputs* a *and* b *to produce* cn *and* sn *such that*

$$\begin{pmatrix} cn & sn \\ -sn & cn \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \rho \\ 0 \end{pmatrix}, \qquad \begin{matrix} \rho = \pm\|(a,b)^T\|_2, \\ cn^2 + sn^2 = 1 \end{matrix}$$

except that rotg$(0, 0, cn, sn)$ *produces* $cn = 0$ *and* $sn = 1$. *For two* n*-vectors* $\mathbf{x}$ *and* $\mathbf{y}$ *the call* $rot(\mathbf{x}, \mathbf{y}, cn, sn)$ *performs the rotation*

$$\mathbf{x} \leftarrow (cn)\mathbf{x} + (sn)\mathbf{y}, \quad \mathbf{y} \leftarrow -(sn)\mathbf{x} + (cn)\mathbf{y}.$$

If the condition in Proposition 5.6 and Lemma 5.7 that excludes the rounding error in applying V_k is lifted, we still obtain a good backward error on the step (5.16)–(5.17).

procedure *rot_back*$(C, n, \mathbf{cn}, \mathbf{sn}, s)$

```
%
% Input arguments -- cn,sn -- Vectors defining Givens rotations
%                        C -- matrix to be transformed
%                        n -- dimension of C, C is n x n
%                        s -- column to be modified in
%                             orthogonal transformations
%
% Transform first row in standard fashion as in step 2.
%
```

for $j = 2{:}\, n$
  $rot(C(1,1), C(1,j), \mathbf{cn}(j-1), \mathbf{sn}(j-1))$;
end;
if $s == 1$ % The case $s = k$
  $\mathbf{w} = 0$; % $\mathbf{w}$ computes the backward recurrence for column 1.
  for $j = n{:}\, -1{:}\, 2$
    $\mathbf{w} = (\mathbf{w} - \mathbf{sn}(j-1) * C(2{:}\, n, j))/\mathbf{cn}(j-1)$;
    $C(2{:}\, n, j) = \mathbf{cn}(j-1) * C(2{:}\, n, j) - \mathbf{sn}(j-1) * \mathbf{w}$;
  end;
else % The case $s > k$
  $\mathbf{w} = 0$;
  for $j = n{:}\, -1{:}\, s+1$
    $\mathbf{w} = (\mathbf{w} - sn(j-1) * C(2{:}\, n, j))/\mathbf{cn}(j-1)$;
    $C(2{:}\, n, j) = \mathbf{cn}(j-1) * C(2{:}\, n, j) - \mathbf{sn}(j-1) * \mathbf{w}$;
  end;
  for $j = 2{:}\, s-1$
    $rot(C(2{:}\, n, 1), C(2{:}\, n, j), \mathbf{cn}(j-1), \mathbf{sn}(j-1))$;
  end;
  % Construct column s
  $C(2{:}\, n, s) = (\mathbf{w} - \mathbf{cn}(s-1) * C(2{:}\, n, 1))/\mathbf{sn}(s-1)$;
  % Use new column s
  $C(2{:}\, n, s) = \mathbf{cn}(s-1) * C(2{:}\, n, s) - \mathbf{sn}(s-1) * C(2{:}\, n, 1)$;
end;
end*rot_back*

FIG. 5.2. *The rot_back procedure.*

For that, we state the following theorem, which is proved in [2]. Lemma 5.7 is used in the proof of this theorem.

THEOREM 5.8. *Suppose that* $C^{(k)}$ *is computed from* $C^{(k-1)}$ *as in* (5.16)–(5.17) *using Algorithm* 5.1 *to compute* $C^{(k)}$ *from* $M^{(k)}$. *For* $1 \le k < j \le n$, *let*

$$\beta_{k,j} = \max\{\|C^{(k-1)}(k{:}\, n, j{:}\, n)\|_F, \|C^{(k)}(k+1{:}\, n, j{:}\, n)\|_F\}, \tag{5.48}$$

and $f_1(n) = n^3 + 20n^2 + 12n$. *Then, in floating point arithmetic with machine unit* ε_M, *the computed matrices* $C^{(k-1)}$ *and* $C^{(k)}$ *satisfy*

$$C^{(k)} + \mathbf{e}_{k-1} \begin{pmatrix} \delta\gamma_{k-1} & \delta\phi_k \end{pmatrix} \begin{pmatrix} \mathbf{e}_{k-1}^T \\ \mathbf{e}_k^T \end{pmatrix}$$

$$= E_k \hat{U}_k^T [C^{(k-1)}(k-1\colon n, k\colon n) + \delta C^{(k-1)}] \hat{Q}_k, \tag{5.49}$$

where $\hat{U}_k$ *is orthogonal, the matrices* E_k, $\delta C_k^{(k-1)}$, *and* $\hat{Q}_k$ *and the scalars* $\delta\gamma_{k-1}$ *and* $\delta\phi_k$ *satisfy*

$$\|\delta C^{(k)}(:, j\colon n)\|_F \le \varepsilon_M f_1(n-k+1)\beta_{k,j} + O(\varepsilon_M^2),$$

$$\|E_k - I\|_2 \le 3(n-k+1)^2\varepsilon_M + O(\varepsilon_M^2),$$

$$\|\hat{Q}_k^T\, \hat{Q}_k - I\|_2 \le 6(n-k+1)^2\varepsilon_M + O(\varepsilon_M^2),$$

$$\delta\gamma_0 = \delta\phi_1 = 0,$$

$$|\delta\gamma_{k-1}| \le 3(n-k+1)^2\varepsilon_M + O(\varepsilon_M^2), \quad |\delta\phi_k| \le (n-k+1)\varepsilon_M + O(\varepsilon_M^2), \quad k \ge 2.$$

The diagonal matrix E_k, the slight nonorthogonality of $\hat{Q}_k$, and errors introduced to γ_{k-1} and ϕ_k are the results of the error analysis techniques necessary to get a columnwise backward error bound on $C^{(k-1)}$. Algorithm 5.1 is critical in ensuring this error bound.

In the next section, we give a simple example that shows the effect of using Algorithm 5.1 in the implementation of one step of bidiagonal reduction.

5.3. A 4 × 4 example. The following 4 × 4 example illustrates how this new Givens-based procedure preserves small singular values better than the Golub–Kahan Householder-based procedure.

EXAMPLE 5.1. *Let* A *be the* 4×4 *matrix*

$$A = \begin{pmatrix} 1 & \zeta_1 & \zeta_1 & 2\zeta_1 \\ 0 & 1/\sqrt{3} & \zeta_2 & \zeta_2 \\ 0 & 1/\sqrt{3} & 2\zeta_2 & \zeta_2 \\ 0 & 1/\sqrt{3} & 3\zeta_2 & 3\zeta_2 \end{pmatrix}, \tag{5.50}$$

where ζ_1 *and* ζ_2 *are small parameters. Using the MATLAB value* $\varepsilon_M = 2.2204e-16$, *we chose* $\zeta_1 = 10\varepsilon_M$ *and* $\zeta_2 = \varepsilon_M/1000$. *That yields a matrix that has two singular values clustered at* 1, *and two distinct singular values smaller than* ε_M.

To the digits displayed,

$$A = \begin{pmatrix} 1 & 2.22045e-15 & 2.22045e-15 & 4.44089e-15 \\ 0 & 0.57735 & 2.22045e-19 & 2.22045e-19 \\ 0 & 0.57735 & 4.44089e-19 & 2.22045e-19 \\ 0 & 0.57735 & 6.66134e-19 & 6.66134e-19 \end{pmatrix}.$$

The singular values are well-conditioned under column scalings. If we let

$$\Delta = \mathrm{diag}(\|A(:,1)\|_2, \|A(:,2)\|_2, \|A(:,3)\|_2, \|A(:,4)\|_2),$$

then

$$\kappa_2(A\Delta^{-1}) = \|A\Delta^{-1}\|_2 \|\Delta A^{-1}\|_2 = 5.1962e+04.$$

Thus according to the theory in [15]*, we should expect that the Jacobi method will compute the singular values of* A *to at least* 11 *digit accuracy. The Jacobi method (coded by Yoon for* [6]*) obtains the SVD*

$$A = U\Sigma V^T,$$

where (to the digits displayed)

$$\Sigma = \mathrm{diag}(1, 1, 4.63692e-19, 1.22778e-19),$$

$$U = \begin{pmatrix} 2.22045e-15 & 1 & -1.01546e-51 & -4.70505e-53 \\ 0.57735 & 1.47911e-33 & -0.553113 & -0.600611 \\ 0.57735 & 1.97215e-33 & -0.243588 & 0.779315 \\ 0.57735 & 4.43734e-33 & 0.7967 & -0.178704 \end{pmatrix},$$

$$V = \begin{pmatrix} -4.55449e-33 & 1 & -4.82373e-15 & 1.17624e-15 \\ 1 & 2.22602e-51 & -9.86268e-19 & -1.72585e-19 \\ 7.69185e-19 & 2.22045e-15 & 0.646375 & 0.76302 \\ 6.40988e-19 & 4.44089e-15 & 0.76302 & -0.646375 \end{pmatrix}.$$

For the bidiagonal reduction procedures below, we did none of the preprocessing in section 3*. The singular values of the bidiagonal matrices were obtained by the bisection procedure described in* [3]*; the singular vectors were obtained by using oqd iterations as in* [21] *until the matrices decoupled into* 2×2 *blocks.*

The dramatic difference between the application of our algorithm and the standard bidiagonal reduction procedure occurs with the application of U_2 *and* V_2*. Algorithm* 4.1 *implemented with Householder transformations yields*

$$N^{(2)} = \begin{pmatrix} 1 & 5.438960e-15 & 0 & 0 \\ 0 & -.2357023 & -.2357023 & -.4714045 \\ 0 & -.2357023 & -.2357023 & -.4714045 \\ 0 & -.2357023 & -.2357023 & -.4.714045 \end{pmatrix} = AV_2,$$

where

$$\|N^{(2)}(3\colon 4, 3\colon 4)\|_F = 0.7454.$$

Note that columns of $N^{(2)}(2\colon 4, 2\colon 4)$ *are all nearly multiples of one another. Thus when we compute*

$$C^{(2)} = \begin{pmatrix} 1 & 5.438960e-15 & 0 & 0 \\ 0 & .4082483 & .4082483 & .8164966 \\ 0 & 0 & -2.775558e-17 & -5.551115e-17 \\ 0 & 0 & -2.775558e-17 & -5.551115e-17 \end{pmatrix} = U_2^T N^{(2)},$$

for which

$$\|C^{(2)}(3\colon 4, 3\colon 4)\|_F = 8.7771e-17,$$

the lower 2×2 *corner of* $C^{(2)}$ *is dominated by the rounding error in this step, and its two columns are colinear. Notice that* $\|N^{(2)}(3\colon 4, 3\colon 4)\|_F$ *is much larger than*

$\|C^{(2)}(3\colon 4, 3\colon 4)\|_F$. *This is the concern raised in the text surrounding* (5.18) *and that motivated Algorithm* 5.1.

The final bidiagonal matrix obtained by this procedure is

$$B = \begin{pmatrix} -1 & 5.43896e-15 & 0 & 0 \\ 0 & 0.408248 & -0.912871 & 0 \\ 0 & 0 & -8.77708e-17 & -3.48631e-32 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

It has the singular values

$$\Sigma = \mathrm{diag}(1, 1, 3.58323e-17, 0).$$

The computed singular vector matrices are

$$U = \begin{pmatrix} -0.596931 & -0.802293 & 1.7791e-31 & 0 \\ -0.463204 & 0.344638 & -0.816497 & 0 \\ -0.463204 & 0.344638 & 0.408248 & -0.707107 \\ -0.463204 & 0.344638 & 0.408248 & 0.707107 \end{pmatrix},$$

$$V = \begin{pmatrix} -0.596931 & -0.802293 & -4.96507e-15 & -4.83077e-30 \\ -0.802293 & 0.596931 & -1.11022e-16 & 3.33067e-16 \\ -1.52586e-15 & -1.63233e-15 & 0.447214 & -0.894427 \\ -3.31895e-15 & -3.06585e-15 & 0.894427 & 0.447214 \end{pmatrix}.$$

The invariant subspaces for the double singular value at 1 *and for singular values* 3 *and* 4 *are correct. However, the individual singular vectors for singular values* 3 *and* 4 *are wrong.*

Our algorithm computes the same step

$$M^{(2)} = \begin{pmatrix} 1 & -2.220446e-15 & -2.220446e-15 & -4.440892e-15 \\ 0 & -1 & -7.691851e-18 & -6.409876e-18 \\ 0 & -1.110223e-16 & 8.127397e-19 & -9.384709e-19 \\ 0 & -1.110223e-16 & 3.033186e-18 & 3.502421e-18 \end{pmatrix} = U_2^T A,$$

where

$$\|M^{(2)}(3\colon 4, 3\colon 4)\|_F = 4.7967e-18.$$

Then the procedure of Algorithm 5.1 *produces*

$$C^{(2)} = \begin{pmatrix} 1 & -5.438960e-15 & 0 & 0 \\ 0 & -.4082483 & .7071068 & .5773503 \\ 0 & 0 & -1.778109e-19 & -1.625479e-18 \\ 0 & 0 & 9.242744e-18 & 6.066371e-18 \end{pmatrix} = M^{(2)}V_2,$$

which has

$$\|C^{(2)}(3\colon 4, 3\colon 4)\|_F = 1.1176e-17,$$

and the columns of $C^{(2)}(3\colon 4, 3\colon 4)$ *are linearly independent. Mathematically,* $\|M^{(2)}(3\colon 4, 2\colon 4)\|_F = \|C^{(2)}(3\colon 4, 3\colon 4)\|_F$, *but, in fact,*

$$\|M^{(2)}(3\colon 4, 2\colon 4)\|_F = 1.5708e-16,$$

which is more than 10 *times as large. The value of* s *in Algorithm* 5.1 *is* 2 *for this step. The values in* $M^{(2)}(3\colon 4, 2)$ *are mostly rounding error from computing* $M^{(2)}$*. Step* 3 *of Algorithm* 5.1 *recomputes these components of column* 2 *from columns* 3 *and* 4 *so that*

$$C^{(2,2)}(3\colon 4, 2\colon 4)C^{(2)}(1, 2\colon 4)^T = 0,$$

as is expected.

The resulting bidiagonal matrix for our Givens procedure (to the digits displayed) is

$$B = \begin{pmatrix} -1 & -5.43896e-15 & 0 & 0 \\ 0 & -0.408248 & 0.912871 & 0 \\ 0 & 0 & 1.10577e-18 & -1.01936e-19 \\ 0 & 0 & 0 & -1.26113e-19 \end{pmatrix}.$$

The matrix B *has the same singular values as those obtained by Jacobi to* 15 *significant digits. Thus there can be no important difference in their quality. The corresponding singular vectors are*

$$U = \begin{pmatrix} -0.633989 & -0.773342 & 2.23672e-33 & -1.44416e-34 \\ -0.446489 & 0.366034 & 0.553113 & 0.600611 \\ -0.446489 & 0.366034 & 0.243588 & -0.779315 \\ -0.446489 & 0.366034 & -0.7967 & 0.178704 \end{pmatrix},$$

$$V = \begin{pmatrix} -0.633989 & -0.773342 & 4.82373e-15 & -1.17624e-15 \\ -0.773342 & 0.633989 & -1.10294e-16 & 4.3525e-17 \\ -1.53653e-15 & -1.61158e-15 & -0.646375 & -0.76302 \\ -2.72962e-15 & -3.50472e-15 & -0.76302 & 0.646375 \end{pmatrix}.$$

The first two singular vectors correspond to a cluster at 1*; thus they cannot be expected to be the same, but the subspaces associated with the leading two singular values are computed correctly. The third and fourth singular vectors are the same as obtained by Jacobi to* 15 *digits.*

If $C^{(2)}$ *above is computed from* $M^{(2)}$ *using an ordinary implementation of Givens rotations, we obtain*

$$C^{(2)} = \begin{pmatrix} 1 & -5.43896e-15 & 0 & 0 \\ 0 & -0.408248 & 0.707107 & 0.57735 \\ 0 & -4.53681e-17 & 7.85621e-17 & 6.39977e-17 \\ 0 & -4.49149e-17 & 7.87191e-17 & 6.41258e-17 \end{pmatrix}.$$

Of course, we set $C^{(2)}(3\colon 4, 2) = 0$*, but that neglects a column of* 2*-norm* $6.3841e-17$ *which is of the magnitude of the singular values in lower* 2×2 *corner of* C*.*

If we continue the computation this way, the resulting singular values are

$$\Sigma = \mathrm{diag}(1, 1, 5.85614e-17, 1.62027e-22).$$

Singular values 3 *and* 4 *are wrong. The singular vectors matrices are*

$$U = \begin{pmatrix} -0.615412 & -0.788205 & 0 & 0 \\ -0.455071 & 0.355308 & 0.816496 & 0.00081583 \\ -0.455071 & 0.355308 & -0.407542 & -0.707514 \\ -0.455071 & 0.355308 & -0.408955 & 0.706699 \end{pmatrix},$$

$$V = \begin{pmatrix} -0.615412 & -0.788205 & 4.96505e-15 & -1.37484e-17 \\ -0.788205 & 0.615412 & -1.11022e-16 & 2.1684e-19 \\ -1.49775e-15 & -1.64768e-15 & -0.449689 & -0.893185 \\ -2.64547e-15 & -3.56866e-15 & -0.893185 & 0.449689 \end{pmatrix}.$$

As in the case of the Householder algorithm, the singular vectors associated with the clustered singular value at 1 *are correct, but the individual singular vectors associated with singular values* 3 *and* 4 *are wrong. This suggests that Algorithm* 5.1 *makes a difference in how well the bidiagonal reduction is computed.*

We also tried this example with Householder transformations where the transformation U_k and V_k were performed in reverse order. The results were similar to those for Givens method implemented without using Algorithm 5.1.

We have now specified everything in Algorithm 4.1, and thus we give our proposed bidiagonal reduction algorithm in the next section.

6. A Givens-based bidiagonal reduction algorithm and its backward error.

6.1. Statement of the algorithm. We now present a Givens-based bidiagonal reduction procedure for an $n \times n$ matrix C satisfying (3.6). In section 6.2, we show that this new algorithm will achieve error bounds of the form (1.5)–(1.8).

ALGORITHM 6.1 (new procedure for bidiagonal reduction).

1. *Let U_1 be an orthogonal transformation such that*

$$U_1^T C(:,1) = \gamma_1 \mathbf{e}_1,$$

$$C \leftarrow U_1^T C, \quad U \leftarrow U_1, \quad V \leftarrow V_1 \equiv I.$$

2. ***for*** $k = 2{:}\,n-1$

   (a) *If $C(k-1, k{:}\,n) \neq 0$, let V_k be the product of Givens rotations*

$$V_k = V_{k,k+1} \cdots V_{k,n}, \quad V_{k,j} = J(k, j, \theta_{kj})$$

   that satisfies

$$V_k^T C(k-1, k{:}\,n)^T = \pm \phi_k \mathbf{e}_1,$$

   else $V_k = I$ (implicitly). Compute

$$\mathbf{y}_k = C(k{:}\,n, k{:}\,n) V_k \mathbf{e}_1, \quad V(:, k{:}\,n) \leftarrow V(:, k{:}\,n) V_k.$$

   (b) *Find an orthogonal transformation U_k such that*

$$U_k^T \mathbf{y}_k = \gamma_k \mathbf{e}_1,$$

$$C(k{:}\,n, k{:}\,n) \leftarrow U_k^T C(k{:}\,n, k{:}\,n), \quad U(:, k{:}\,n) \leftarrow U(:, k{:}\,n) U_k.$$

   (c) *If $C(k-1, k{:}\,n) \neq 0$, let $\mathbf{v}_1^{(k)} = C(k-1, k{:}\,n)^T / \phi_k$ and compute*

$$C(k{:}\,n, k{:}\,n) \leftarrow C(k{:}\,n, k{:}\,n) V_k$$

   using the two calls

compute_pivot$(C(k\colon n, k\colon n), n-k+1, \mathbf{v}_1^{(k)}, \mathbf{cn}, \mathbf{sn}, s)$;
rot_back$(C(k\colon n, k\colon n), n-k+1, \mathbf{cn}, \mathbf{sn}, s)$;

3.

$$\gamma_n \leftarrow C(n,n), \quad \phi_n \leftarrow C(n-1,n)$$

The bidiagonal reduction of C is given by

$$C = UBV^T,$$

where

$$B = \text{bidiag}(\gamma_1, \ldots, \gamma_n; \phi_2, \ldots, \phi_n).$$

Table 6.1 summarizes the complexity of the two bidiagonal reduction algorithms. The extra operations in Algorithm 6.1 over Algorithm 4.1 are the computation of the column norms hidden in the routine *compute_pivot*, and the use of Givens rotations instead of Householder transformations.

TABLE 6.1
Complexity of bidiagonal reduction algorithms.

Compute U,V?	Algorithm 4.1(H)	Algorithm 6.1
Yes	$\frac{20}{3}n^3$	$\frac{26}{3}n^3$
No	$\frac{8}{3}n^3$	$4n^3$

H – with Householder transformations

6.2. Error bounds on Algorithm 6.1. The error bounds for this paper are stated in two theorems.

THEOREM 6.1. *Let $C \in \Re^{n\times n}$ and let $B = \text{bidiag}(\gamma(1\colon n); \phi(2\colon n)) \in \Re^{n\times n}$ be the bidiagonal matrix computed by Algorithm* 6.1 *in floating point arithmetic with machine precision ε_M. Let $C^{(k)}, k = 1, \ldots, n-1$, be the contents of C after k passes thorough the main loop of Algorithm* 6.1. *Then there exist $U, V \in \Re^{n\times n}$ and modestly growing functions $g_i(\cdot)$, $i = 1, 2, 3$, such that*

$$C + \delta C = U(B + \delta B)V^T, \tag{6.1}$$

where

$$\|U^TU - I\|_2, \|V^TV - I\|_2 \le g_1(n)\varepsilon_M + O(\varepsilon_M^2), \tag{6.2}$$

$$|\delta B| \le g_2(n)\varepsilon_M|B| + O(\varepsilon_M^2), \quad |\cdot| \text{ and } \le \text{ entry-wise}, \tag{6.3}$$

and for $j = 1, \ldots, n$

$$\|\delta C(:, j\colon n)\|_F \le \varepsilon_M g_3(n) \max_{0\le k<j} \|C^{(k)}(k+1\colon n, j\colon n)\|_F \tag{6.4}$$

$$\le \varepsilon_M g_3(n)\hat{\rho}_V^{(j-1)}\|C(:, j\colon n)\|_F + O(\varepsilon_M^2),$$

where $\hat{\rho}_V^{(j-1)}$ *is defined by* (2.9).

Proof. This proof is an induction argument on Theorem 5.8. Using the notation from that theorem, we let

$$\tilde{U}_k = \hat{U}_1 E_2^{-1} \hat{U}_2 \cdots \hat{U}_k E_k^{-1},$$

$$\bar{U}_k = \hat{U}_k E_k \hat{U}_{k+1} E_{k+1} \cdots \hat{U}_{n-1} E_{n-1},$$

$$U = \tilde{U}_{n-1} = \bar{U}_1,$$

$$\tilde{Q}_k = Q_2^{-T} \cdots Q_k^{-T},$$

$$\bar{Q}_k = Q_{k+1} \cdots Q_{n-1}, \quad \bar{Q}_n = I,$$

$$V = \tilde{Q}_{n-1} = \bar{Q}_1^{-T}.$$

Clearly, by an induction argument on the bounds on Q_k and E_k from Theorem 5.8, we have that

$$\|V^T V - I\|_2, \|U^T U - I\|_2 \leq 2n^3 \varepsilon_M + O(\varepsilon_M^2),$$

which establishes (6.2).

An induction argument on the backward error bound in Theorem 5.8 yields

$$C + \delta C = U(B + \delta B)V^T,$$

where

$$\delta C = \sum_{k=1}^{n-1} \tilde{U}_k \delta C^{(k)} \tilde{Q}_{k-1}^T$$

and

$$\delta B = \sum_{k=1}^{n-1} \bar{U}_k^T \mathbf{e}_{k-1} \begin{pmatrix} \delta\gamma_{k-1} & \delta\phi_k \end{pmatrix} \begin{pmatrix} \mathbf{e}_{k-1}^T \\ \mathbf{e}_k^T \end{pmatrix} \bar{Q}_k.$$

Clearly, $\bar{U}_k$ and $\bar{Q}_k$ have no effect on the terms of the sum constituting δB, so

$$\delta B = \sum_{k=1}^{n-1} \mathbf{e}_{k-1} \begin{pmatrix} \delta\gamma_{k-1} & \delta\phi_k \end{pmatrix} \begin{pmatrix} \mathbf{e}_{k-1}^T \\ \mathbf{e}_k^T \end{pmatrix}$$

$$= \text{bidiag}(\delta\gamma_1, \ldots, \delta\gamma_{n-1}, 0; \delta\phi_2, \ldots, \delta\phi_n),$$

where from Theorem 5.8,

$$|\delta\gamma_k| \leq 3(n-k+2)^2 \varepsilon_M |\gamma_k| + O(\varepsilon_M^2), \quad |\delta\phi_k| \leq 3(n-k+1)\varepsilon_M |\gamma_k| + O(\varepsilon_M^2)$$

for all appropriate k. We may write (conservatively) that

$$|\delta B| \leq 3(n+1)^2 \varepsilon_M |B| + O(\varepsilon_M^2),$$

thereby establishing (6.3).

To prove (6.4), we note that

$$\begin{aligned}\|\delta C(:, j{:}n)\|_F &\leq \sum_{k=1}^{n-1} \|\tilde{U}_k \delta C^{(k)} \tilde{Q}_k^T \begin{pmatrix} 0 \\ I_{n-j+1} \end{pmatrix} \|_F + O(\varepsilon_M^2) \\ &= \sum_{k=1}^{n-1} \|\delta C^{(k)} \tilde{Q}_k^T \begin{pmatrix} 0 \\ I_{n-j+1} \end{pmatrix} \|_F + O(\varepsilon_M^2).\end{aligned}$$

The structure of the Givens rotations in the algorithm give us that

$$\tilde{Q}_k^T = Q_{k-1}^{-1} \cdots Q_1^{-1} \begin{pmatrix} 0 \\ I_{n-j+1} \end{pmatrix} = \begin{matrix} k-1 \\ j-k+1 \\ n-j+1 \end{matrix} \begin{pmatrix} J_1 \\ 0 \\ J_2 \end{pmatrix} = J. \tag{6.5}$$

Noting that

$$\delta C(1{:}k-1, :) = 0, \quad \delta C(:, 1{:}k-1) = 0,$$

we have that

$$\|\delta C^{(k)}(:, j{:}n) Q_{k-1}^T \begin{pmatrix} 0 \\ I_{n-j+1} \end{pmatrix} \|_F = \|\delta C^{(k)}(:, j{:}n) J_2\|_F \leq \|\delta C^{(k)}(:, j{:}n)\|_F.$$

Therefore,

$$\|\delta C(:, j{:}n)\|_F \leq \sum_{k=1}^{n} \|\delta C^{(k)}(:, j{:}n)\|_F.$$

Using the bound on $\|\delta C^{(k)}(:, j{:}n)\|_F$ and the definition of $\beta_{k,j}$ from Theorem 5.8, we have

$$\begin{aligned}\|\delta C(:, j{:}n)\|_F &\leq \varepsilon_M \sum_{k=1}^{n-1} f_1(n-k+1)\beta_{k,j} + O(\varepsilon_M^2) \\ &\leq \varepsilon_M g_3(n) \max_{1 \leq k < j \leq n} \beta_{k,j} + O(\varepsilon_M^2),\end{aligned}$$

where

$$g_3(n) = \frac{n^4}{4} + \frac{20n^3}{3} + 6n.$$

From the definition of $\beta_{k,j}$ we have that

$$\max_{1 \leq k < j} \beta_{k,j} = \max_{1 \leq k < j} \|C^{(k)}(k+1{:}n, j{:}n)\|_F \leq \hat{\rho}_V^{(j-1)} \|C(:, j{:}n)\|_F.$$

That establishes (6.4) and the theorem. ∎

Theorem 6.1 uses none of the properties of C from section 3, the backward error bound above applies to Algorithm 6.1 applied to any nonsingular matrix. Equation (3.6) allows us to obtain a columnwise bound expressed in the next theorem.

THEOREM 6.2. *Let $C \in \Re^{n\times n}$ satisfy* (3.6)*, and let $B = \text{bidiag}(\gamma(1\colon n); \phi(2\colon n)) \in \Re^{n\times n}$ be the bidiagonal matrix computed by Algorithm* 6.1 *in floating point arithmetic with machine precision ε_M. Let $C^{(k)}, k = 1, \ldots, n-1$, be the contents of C after k passes through the main loop of Algorithm* 6.1*, and let $\hat{\rho}_V^{(k)}$ be defined by* (5.11)*. Then there exist $U, V \in \Re^{n\times n}$ satisfying* (6.2) *and a modestly growing function $g_4(\cdot)$ such that B and C satisfy* (6.1)*, δB is as in Theorem* 6.1*, and*

$$\|\delta C(:,j)\|_2 \le \varepsilon_M \hat{\rho}_V^{(j-1)} g_4(n) \|C(:,j)\|_2 + O(\varepsilon_M^2), \tag{6.6}$$

where

$$\hat{\rho}_V^{(j-1)} = \min\{\rho_V^{(j-1)}, \|C\|_F / \|C(:,j\colon n)\|_F\}.$$

Proof. The result follows from our bounds on

$$\|C^{(k)}(k+1\colon n, j\colon n)\|_F.$$

A combination of Theorem 6.1 and (3.6) leads to

$$\begin{aligned}
\|\delta C(:,j)\|_2 &\le \|\delta C(:,j\colon n)\|_F \\
&\le \varepsilon_M g_3(n) \|C^{(k)}(k+1\colon n, j\colon n)\|_F + O(\varepsilon_M^2) \\
&\le \varepsilon_M \hat{\rho}_V^{(j-1)} g_3(n) \|C(:,j\colon n)\|_F + O(\varepsilon_M^2) \\
&\le \varepsilon_M \hat{\rho}_V^{(j-1)} \sqrt{n} g_3(n) \|C(:,j)\|_2 + O(\varepsilon_M^2) \\
&= \varepsilon_M \hat{\rho}_V^{(j-1)} g_4(n) \|C(:,j)\|_2, \quad g_4(n) = \sqrt{n} g_3(n) + O(\varepsilon_M^2). \qquad \square
\end{aligned}$$

From Theorem 6.2, we have the conditional columnwise error bound necessary to establish error bounds on the singular values and vectors as discussed in section 2.

7. Numerical tests. We performed three sets of numerical tests on the bidiagonal reduction algorithms. Three separate routines were used to find the SVD of the matrices in the test sets, as follows.

- *The Jacobi algorithm.* The Jacobi method described in [15]. In Figures 7.1 and 7.2, this is referred to as *Jacobi alg.*
- *The Givens algorithm.* The bidiagonalization method of Algorithm 6.1 followed by the bisection routine of Demmel and Kahan [14]. In Figures 7.1, 7.2, and 7.3, this is referred to as *Givens alg.*
- *The Householder algorithm.* The bidiagonalization method of Algorithm 4.1 using Householder transformations followed by the same bisection routine. In Figures 7.1, 7.2, and 7.3, this is referred to as *Householder alg.*

Our first two sets of examples are constructed using the Kahan matrices [29]. Let $\hat{C}$ be the $n \times n$ lower triangular matrix

$$\hat{C} = (\hat{c}_{ij}), \quad \hat{c}_{ij} = \begin{cases} \alpha^{i-1}, & i = j, \\ -\alpha^{i-1}\beta, & i > j, \end{cases} \tag{7.1}$$

where $\alpha^2 + \beta^2 = 1$ and $\alpha, \beta > 0$. If we chose α bounded away from zero or one, we obtain a matrix that is unaltered by QR factorization with column pivoting, has slowly decaying diagonals, but has a condition number that grows rapidly with n.

This matrix has one isolated small singular value [40], $\sigma_n(\hat{C})$ in (7.3). The other singular values $\sigma_j(\hat{C}), k = 1, \ldots, n-1$, are close enough to $\|\hat{C}\|_2$ that standard SVD software will compute them to nearly machine relative accuracy. The smallest singular value $\sigma_n(\hat{C})$ may be recovered from

$$\sigma_n(\hat{C}) = \frac{|\det(\hat{C})|}{\prod_{j=1}^{n-1}\sigma_j(\hat{C})} = \frac{|\prod_{j=1}^{n}\hat{c}_{jj}|}{\prod_{j=1}^{n-1}\sigma_j(\hat{C})}. \tag{7.2}$$

Since the formula (7.2) uses only multiplications and divisions, in the absence of underflow, it will compute $\sigma_n(\hat{C})$ to nearly machine relative accuracy.

We used the formula (7.2) to construct both of the first two test sets given next.

For each matrix, we also computed the values

$$\rho_V = \max_{1\le k\le n-1}\rho_V^{(k)}, \quad \hat{\rho}_V = \max_{1\le k\le n-1}\hat{\rho}_V^{(k)},$$

$$totratio = \max_{1\le k<j\le n}\frac{\|C^{(k)}(k+1\colon n, j\colon n)\|_F}{\|C(:, j\colon n)\|_F}.$$

Of course, we expect that $\rho_V \ge \hat{\rho}_V \ge totratio$. In practice, the gaps separating these three quantities were very large.

EXAMPLE 7.1 (test set 1). *We let $\hat{C}$ in* (7.1) *be constructed with $\beta = 0.3$ and $n = 50$. Let* $\mathbf{w}$ *be the right singular vector such that*

$$\hat{C}\mathbf{w} = \sigma_n(\hat{C})\mathbf{y}, \quad \|\mathbf{w}\|_2 = \|\mathbf{y}\|_2 = 1, \tag{7.3}$$

where $\sigma_n(\hat{C})$. We then constructed the twenty (20) 51×51 matrices defined by

$$C_j = \begin{pmatrix} \hat{C} & 0 \\ \zeta\mathbf{w}^T & \xi_j \end{pmatrix}, \quad j = 1, \ldots, 20, \tag{7.4}$$

where

$$\zeta = 0.5\hat{c}_{50,50}, \quad \xi_j = \sigma_{50}(\hat{C})/100^{j-1}.$$

The smallest singular value of C_j is also the smallest singular value of the 2×2 matrix

$$D_j = \begin{pmatrix} \sigma_{50}(\hat{C}) & 0 \\ \zeta & \xi_j \end{pmatrix}.$$

Given $\sigma_{50}(\hat{C})$ to nearly machine relative accuracy, we can find the smallest singular value of D_j to nearly machine relative accuracy. We used this value as the exact value of $\sigma_{51}(C_j)$.

We then computed $\sigma_{51}(C_j)$ using the all three of the algorithms given above. The results are represented in Figure 7.1. *The values posted are*

$$\log_{10}\left(\frac{|\tilde{\sigma}_{51}(C_j) - \sigma_{51}(C_j)|}{\sigma_{51}(C_j)}\right),$$

where $\tilde{\sigma}_{51}(C_j)$ and $\sigma_{51}(C_j)$ are the computed and "exact" 51st singular values of C_j. From Figure 7.1, *we can see that the Jacobi and Givens algorithms always compute σ_{51} to at least* 10 *digit accuracy in IEEE double precision. There is no significant*

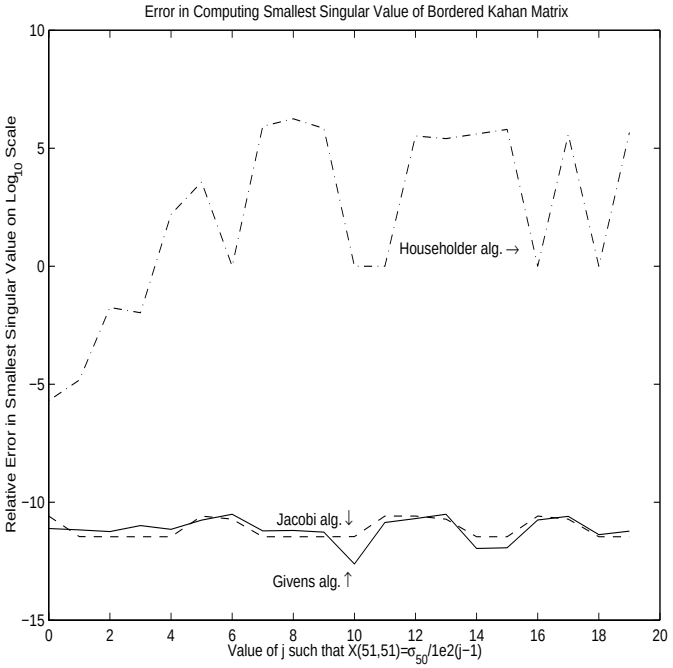

FIG. 7.1. *Relative error from Example* 7.1.

difference in the accuracy of the Jacobi and Givens algorithms on this test set. The Householder algorithm often obtains no accurate digits at all.

The values of ρ_V,$\hat{\rho}_V$ *and* totratio *are given in Table* 7.1. *In this case, the values of* ρ_V *are gross overestimates of the growth factor* totratio *which never exceeds* 4.

EXAMPLE 7.2 (test set 2). *For the second test set, we computed the lower triangular matrices* $C_n, n = 50, 60, \ldots, 200$, *from the QR factorization*

$$\hat{C}_n = QC_n^T,$$

where $\hat{C}_n$ *is a Kahan matrix of size* n *with* $\beta = 0.3$. *Once again this matrix has exactly one small singular value which we compute using the formula* (7.2) *applied to* C_n.

The results represented in Figure 7.2 *are the values*

$$\log_{10}\left(\frac{|\tilde{\sigma}_n(C_n) - \sigma_n(C_n)|}{\sigma_n(C_n)}\right),$$

where $\sigma_n(C_n)$ *is the value computed from the formula* (7.2) *(presumed to be exact) and* $\tilde{\sigma}_n(C_n)$ *computed by one of the three algorithms.*

Similar to the results for Example 7.1, *the Jacobi and Givens algorithms compute* $\sigma_n(C_n)$ *to* 11*-digit accuracy and, once again, there is no significant difference in their accuracy. The Householder algorithm fares a bit better than in Example* 7.1, *but still obtains no more than 5-digit accuracy.*

The values of ρ_V, $\hat{\rho}_V$, *and* totratio *are given in Table* 7.2. *In this class of examples* totratio *was large for some of the smaller problems—it was at* 1.7×10^5 *for* $n = 50$*—but is much smaller for the larger dimensions. Again both* ρ_V *and* $\hat{\rho}_V$ *are gross*

TABLE 7.1
Growth factors for bordered Kahan matrices.

j	ρ_V	$\hat{\rho}_V$	$totratio$
1	1.25434e+009	1.45144e+007	3.86412
2	1.25325e+009	9.7244e+008	3.40234
3	1.25299e+009	8.80033e+006	3.40234
4	1.25299e+009	88066.1	3.40234
5	1.25299e+009	880.661	3.40234
6	1.25299e+009	36.4059	3.40234
7	1.25299e+009	36.4059	3.40234
8	1.25299e+009	36.4059	3.40234
9	1.25299e+009	36.4059	3.40234
10	1.25299e+009	36.4059	3.40234
11	1.25299e+009	36.4059	3.40234
12	1.25299e+009	36.4059	3.40234
13	1.25299e+009	36.4059	3.40234
14	1.25299e+009	36.4059	3.40234
15	1.25299e+009	36.4059	3.40234
16	1.25299e+009	36.4059	3.40234
17	1.25299e+009	36.4059	3.40234
18	1.25299e+009	36.4059	3.40234
19	1.25299e+009	36.4059	3.40234
20	1.25299e+009	36.4059	3.40234

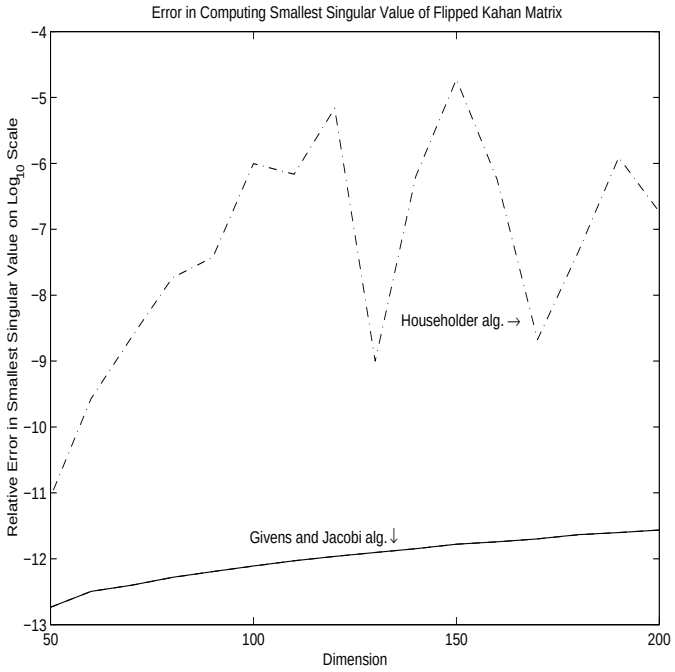

FIG. 7.2. *Relative error from Example* 7.2.

overestimates. For all of the matrices Algorithm 6.1 *computes the smallest singular value correctly.*

The values of $\hat{\rho}_V$ *behave somewhat erratically. It is at about* 10^5 *or* 10^6 *for the first three matrices and hovers about* 10^4 *for the remaining examples. We do not have*

TABLE 7.2
Growth factors for flipped Kahan matrices.

n	ρ_V	$\hat{\rho}_V$	*totratio*
50	840495	279473	171710
60	3.99297e+006	3.99297e+006	89542
70	9.55389e+006	395549	6858.24
80	2.88479e+007	12120.4	507.934
90	5.73345e+008	239.548	25.5923
100	3.13879e+009	404.636	1.39048
110	1.42799e+009	1430.55	1.39033
120	8.87921e+008	2507.13	1.61819
130	5.31115e+008	4181.68	1.61266
140	1.1211e+009	6096.04	1.61983
150	1.06992e+009	11534.8	1.65989
160	1.85538e+009	7503.35	1.58612
170	2.94431e+008	17075.8	1.52791
180	6.5911e+008	20437	1.64403
190	1.03768e+010	33647.4	1.61659
200	8.45255e+008	55320.1	1.55874

a ready explanation for this behavior and it does not seem to affect the accuracy of the algorithm.

These two test sets clearly demonstrate that significant accuracy may be gained through the use of Algorithm 6.1. We found no class of examples where the Jacobi algorithm obtained significantly better accuracy for matrices C of the form (3.1) than did the Givens algorithm.

Interestingly, there were many examples of badly scaled matrices resulting from (3.1) with very small singular values where the Householder algorithm computed the singular values very accurately. The following test set is such a case.

EXAMPLE 7.3 (test set 3). *We used the set*

$$R_k, \quad k = 25, \ldots, 90,$$

where R_k was the Cholesky factor of the Hilbert matrix of dimension k. That is, R_k was the upper triangular matrix with positive diagonals that satisfied

$$H_k = R_k^T R_k,$$

where H_k is a $k \times k$ matrix whose (i, j) entry is $h_{ij} = (i + j - 1)^{-1}$. The matrix R_k is computed from formulas given by Choi [10]. *We then computed the matrices $C_k, k = 25, \ldots, 90$, for input to the three algorithms using the Cox–Higham* [11] *procedure in section* 3 *applied to R_k^T.*

We note that for every matrix C_j in this test set the value $\kappa_2(CD_C^{-1})$ in (2.12) *is less than* 200. *From* [15], *the Jacobi method will compute all of singular values of each C_j to near machine relative accuracy.*

For this test set, we do not present a table of the growth factors ρ_V, $\hat{\rho}_V$, totratio. Instead we simply note that for each matrix in this set, $\rho_V < 212$, $\hat{\rho}_V < 13.5$, and totratio < 4. Thus no significant growth occurred in the columns of these matrices.

For the matrices in Example 7.3, there are no fast formulas for computing the smallest singular value to relative accuracy, thus we use the results of Jacobi algorithm as exact to test the two bidiagonal reduction algorithms.

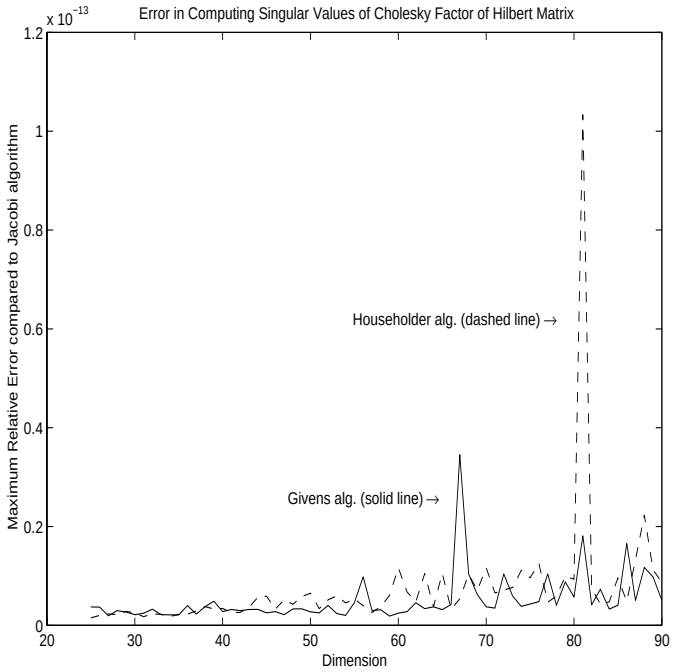

FIG. 7.3. *Relative error from Example* 7.3.

For each matrix in that set, we calculated the two ratios

$$max\_error_G = \max_{1 \le i \le n} \frac{|\sigma_i^G - \sigma_i^J|}{\sigma_i^J},$$

$$max\_error_H = \max_{1 \le i \le n} \frac{|\sigma_i^H - \sigma_i^J|}{\sigma_i^J},$$

where σ_i^J, σ_i^G, and σ_i^H are the ith singular values as calculated by Jacobi, Givens, and Householder algorithms, respectively. Thus we were trying to measure how well the SVDs calculated from the two bidiagonal reduction algorithms agreed with that from the Jacobi algorithm. It can be quickly seen from Figure 7.3 that there is no important difference in the accuracy of the three algorithms on this test set. The good behavior of the Jacobi and Givens algorithms can be explained, but to the author's knowledge there is no satisfactory explanation in the literature for the good behavior of the Householder algorithm.

8. Conclusion. We have constructed a new bidiagonal reduction algorithm (Algorithm 6.1) that allows us to compute the SVD of matrices resulting from (3.1) with more guaranteed accuracy. The accuracy guarantee is not quite as good as the one for Jacobi methods, but we give up little of the speed advantage of bidiagonal reduction methods.

The results given above raise an important unanswered question. The fact that Algorithm 6.1 is based upon 2×2 orthogonal transformations in standard order is used in the proofs of Theorems 5.4 and 6.1. It is not known whether an algorithm

based upon Householder transformations (of size greater than 2×2) or one based on other Givens orderings could yield similar error bounds.

Acknowledgments. The author would like to acknowledge helpful suggestions from Ivan Slapničar, Zlatko Drmač, and Alicja Smoktunowicz. Nick Higham and two anonymous and patient referees made many useful suggestions on earlier versions of this paper.

REFERENCES

[1] J.L. Barlow, *Stability analysis of the G-algorithm and a note on its application to sparse least squares problems*, BIT, 25 (1985), pp. 507–520.

[2] J.L. Barlow, *More Accurate Bidiagonal Reduction for Computing the Singular Value Decomposition*, Technical report, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, 2001; also available online from http://www.cse.psu.edu/∼barlow/bidiag-final4.ps.

[3] J. Barlow and J. Demmel, *Computing accurate eigensystems of scaled diagonally dominant matrices*, SIAM J. Numer. Anal., 27 (1990), pp. 762–791.

[4] J.L. Barlow and S.L. Handy, *The direct solution of weighted and equality constrained least-squares problems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 704–716.

[5] J.L. Barlow and I. Slapničar, *Optimal perturbation bounds for the Hermitian eigenvalue problem*, Linear Algebra Appl., 309 (2000), pp. 19–43.

[6] J.L. Barlow, P.A. Yoon, and H. Zha, *An algorithm and a stability theory for downdating the ULV decomposition*, BIT, 36 (1996), pp. 14–40.

[7] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.

[8] P.A. Businger and G.H. Golub, *Linear least squares solutions by Householder transformations*, Numer. Math., 7 (1965), pp. 269–278.

[9] T.F. Chan, *An improved algorithm for computing the singular value decomposition*, ACM Trans. Math. Software, 8 (1982), pp. 72–83.

[10] M.D. Choi, *Tricks or treats with the Hilbert matrix*, Amer. Math. Monthly, 90 (1983), pp. 301–312.

[11] A.J. Cox and N.J. Higham, *Stability of Householder QR factorization for weighted least squares problems*, in Numerical Analysis 1997, Proceedings of the 17th Dundee Conference, D.F. Griffiths, D.J. Higham, and G.A. Watson, eds., Addison-Wesley-Longman, Harlow, Essex, UK, 1998, pp. 57–73.

[12] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač, *Computing the singular value decomposition with high relative accuracy*, Linear Algebra Appl., 299 (1999), pp. 21–80, 1999.

[13] J.W. Demmel and W.B. Gragg, *On computing accurate singular values and eigenvalues of matrices with acyclic graphs*, Linear Algebra Appl., 185 (1993), pp. 203–217.

[14] J. Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 873–912.

[15] J. Demmel and K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245.

[16] D.S. Dodson and R.G. Grimes, *Remark on algorithm* 539, ACM Trans. Math. Software, 8 (1982), pp. 403–405.

[17] Z. Drmač, *A posteriori computation of the singular vectors in a preconditioned Jacobi SVD algorithm*, IMA J. Numer. Anal., 19 (1999), pp. 191–213.

[18] K.V. Fernando, *Accurate BABE Factorisation of Tri-Diagonal Matrices for Eigenproblems*, Technical Report TR5, Numerical Algorithms Group Ltd., Oxford, UK, 1995.

[19] K.V. Fernando, *On computing an eigenvector of a tridiagonal matrix. Part* I: *Basic results*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 1013–1034.

[20] K.V. Fernando, *Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 373–399.

[21] K.V. Fernando and B.N. Parlett, *Accurate singular values and differential qd algorithms*, Numer. Math., 67 (1994), pp. 191–229.

[22] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, J. Soc. Indust. Appl. Math. Ser. B Numer. Anal., 2 (1965), pp. 205–224.

[23] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[24] G.H. Golub and C. Reinsch, *Singular value decomposition and least squares solutions*, Numer. Math, 14 (1970), pp. 403–20.
[25] R.J. Hanson and C.L. Lawson, *Extensions and applications of the Householder algorithm for solving linear least squares problems*, Math. Comp., 23 (1969), pp. 787–812.
[26] N.J. Higham, *QR factorization with complete pivoting and accurate computation of the SVD*, Linear Algebra Appl., 309 (2000), pp. 153–174.
[27] R.A. Horn and C.A. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
[28] C.G.J. Jacobi, *Über ein Leichtes Verfahren Die in der Theorie der Sacularstorungen Vorkommendern Gleichungen Numerisch Aufzulosen*, Crelle's J., 30 (1848), pp. 51–94.
[29] W. Kahan, *Numerical linear algebra*, Canad. Math. Bull., 9 (1966), pp. 757–801.
[30] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
[31] C.L. Lawson, R.J. Hanson, D.R. Kincaid, and F.T. Krogh, *Basic linear algebra subprograms for FORTRAN usage*, ACM Trans. Math. Software, 5 (1979), pp. 308–325.
[32] R.-C. Li, *Relative perturbation theory:* II. *Eigenspace and singular value variations*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 471–492.
[33] The Mathworks, Inc., *MATLAB: The Language of Technical Computing—Using MATLAB*, The Mathworks, Inc., Natick, MA, 1996.
[34] B.N. Parlett, *Analysis of algorithms for reflectors in bisectors*, SIAM Rev., 13 (1971), pp. 197–208.
[35] B.N. Parlett and I.S. Dhillon, *Relatively robust representations of symmetric tridiagonals*, Linear Algebra Appl., 309 (2000), pp. 121–151.
[36] B.N. Parlett and O.A. Marques, *An implementation of the dqds algorithm (positive case)*, Linear Algebra Appl., 309 (2000), pp. 217–259.
[37] M.J.D. Powell and J.K. Reid, *On applying Householder transformations to linear least squares problems*, in Information Processing, Proceedings of the IFIP Congress, Edinburgh, 1968, Vol. 1, North-Holland, Amsterdam, 1969, pp. 122–126.
[38] A. van der Sluis, *Condition numbers and equilibrium matrices*, Numer. Math., 15 (1969), pp. 74–86.
[39] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.
[40] H. Zha, *Singular values of a classical matrix*, Amer. Math. Monthly, 104 (1997), pp. 172–173.