MATHEMATICS OF COMPUTATION, VOLUME 30, NUMBER 136
OCTOBER 1976, PAGES 772–795

Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization

By J. W. Daniel, W. B. Gragg, L. Kaufman and G. W. Stewart*

Abstract. Numerically stable algorithms are given for updating the Gram-Schmidt QR factorization of an $m \times n$ matrix A ($m \geqslant n$) when A is modified by a matrix of rank one, or when a row or column is inserted or deleted. The algorithms require $O(mn)$ operations per update, and are based on the use of elementary two-by-two reflection matrices and the Gram-Schmidt process with reorthogonalization. An error analysis of the reorthogonalization process provides rigorous justification for the corresponding ALGOL procedures.

1. Introduction. In many applications, most notably to linear least squares problems, it is important to have the QR factorization of a real $m \times n$ matrix A ($m \geqslant n$) into the product of an $m \times n$ matrix Q with orthonormal columns and an $n \times n$ upper triangular matrix R. When A has full rank n the QR factorization is classically computed, in $O(mn^2)$ multiplications and additions, by the Gram-Schmidt process; the diagonal elements of R may then be taken positive and, with this normalization, the factorization is unique. In cases when the rank of A is nearly deficient the columns of Q, computed by the Gram-Schmidt process in the presence of rounding error, can deviate arbitrarily far from orthonormality.

The purpose of this paper is to provide numerically stable and relatively efficient algorithms for updating the Gram-Schmidt QR factorization of A when a row or column is inserted or deleted, or when A is modified by a matrix of rank one: $A \leftarrow \bar{A} = A + vu^T$, where u and v are (column) vectors. The paper may thus be considered supplementary to the important survey [2] of Gill, Golub, Murray and Saunders. It is emphasized that a principal aim of [2] was to update the *complete* orthogonal decomposition of A. This requires storage of an $m \times m$ orthogonal matrix and $O(m^2)$ arithmetic operations per update. The algorithms presented here arose from the desire to efficiently extend a stable modification of the secant method [4] to nonlinear least squares problems. The knowledge of an $m \times n$ Q then suffices, and we normally have $m >> n$. The storage is thus reduced to $O(mn)$, and we shall show that the same is true of the operation counts.

The principal tools which we shall use are the Gram-Schmidt process, *with*

Received March 17, 1975.

AMS (MOS) subject classifications (1970). Primary 65F05; Secondary 15–04, 15A06, 62–04, 62J05, 65F20, 65F25, 65G05, 90C05, 90C30.

* This research was supported in part by the Office of Naval Research under Contracts N00014-67-A-0126-0015 and N00014-67-A-0314-0018, by the Air Force Office of Scientific Research under Grant AFOSR 71-2006, and by NSF MCS 75-23333.

reorthogonalization, and elementary two-by-two reflectors (or Givens matrices). The Gram-Schmidt process is in essence an algorithm for appending columns. Reorthogonalization is used to insure numerical stability, that is to preserve (near) orthogonality of the columns of the computed Q. There are actually two distinct algorithms for the general rank one update, and each has implications for the special updates. One algorithm, which will not be described in detail, uses Givens matrices to obtain an intermediate problem of appending a column; after the Gram-Schmidt process is applied the initial transformations must be undone. We shall present a slightly more efficient algorithm which uses the Gram-Schmidt process first, and then Givens matrices, to reduce the problem to that of appending a row. We then observe that the algorithm given in [2] for appending a row applies also to the Gram-Schmidt QR factorization. The algorithm for the stable deletion of rows is essentially the reverse of that for appending a row, but the arguments involved seem rather subtle.

In the next section we give a heuristic discussion of the reorthogonalization process. This is followed in Section 3 by a description of the updating algorithms. Section 4 is devoted to an error analysis of the reorthogonalization process. This allows us to set certain parameters in the ALGOL codes of Section 5, where some numerical results are also presented.

We shall use the Householder notational conventions [5], with the addition that $\mathbf{R}^{m\times n}$ denotes the set of real $m \times n$ matrices. Also, $\|\cdot\|$ refers to the Euclidean vector norm, as well as to its induced matrix norm: $\|A\| = \max\{\|Ax\|: \|x\| = 1\}$ for $A \in \mathbf{R}^{m\times n}$.

2. The Gram-Schmidt Process, With Reorthogonalization. We first recall the basic step of the Gram-Schmidt process. Let

$$Q = (q_1, q_2, \ldots, q_n) \in \mathbf{R}^{m\times n} \qquad (m \geqslant n)$$

have orthonormal columns, so that $Q^TQ = I_n$, and let $v \in \mathbf{R}^m$. We seek vectors $q \in \mathbf{R}^m$, $r \in \mathbf{R}^n$ and a scalar ρ so that

$$(Q, v) = (Q, q)\begin{pmatrix} I & r \\ 0 & \rho \end{pmatrix} \quad \text{and} \quad Q^Tq = 0.$$

The last column is $v = Qr + q\rho$ and multiplication by Q^T gives $r = Q^Tv$. Setting $v' \equiv q\rho$, we now have

$$v' = v - Qr = (I - QQ^T)v.$$

If $m > n$, we also insist that $\|q\| = 1$ which gives $\rho = \|v'\|$, and then

$$q = v'/\rho \quad \text{if } \rho \neq 0.$$

The process fails if $\rho = 0$, in which case $v = Qr$ is in the range of Q and $(Q, v) = Q(I, r)$ has rank n. In particular, when $m = n$ the matrix Q is orthogonal so we must have

$$q \equiv 0 \quad \text{and} \quad \rho \equiv 0.$$

The process, without normalization, uses $2mn$ multiplications and about the same number of additions.

If $m > n$ and the process fails, then, theoretically, any unit vector orthogonal to the range of Q could be substituted for q. The corresponding numerical problem is more subtle. If the process is carried out in the presence of rounding error, it is unlikely that ρ would vanish exactly, but it could be quite small. The process is designed to force $Q^T v'$ to be small relative to $\|v\|$, and indeed our error analysis will show this to be true. But even if $\|Q^T v'\| = \epsilon\|v\|$, with ϵ small, if ρ is extremely small, then the normalized vector $q = v'/\rho$ would satisfy only $\|Q^T q\| = \epsilon\|v\|/\rho$; and the error relative to $\|v\|$ could be very large. Thus, there could be a catastrophic loss of orthogonality in the computed q.

To rectify this situation one reasons as follows. If $\|v'\|/\|v\|$ is small, then numerical cancellation has occurred in forming v'. Thus, v', as well as q, are likely to be inaccurate relative to their lengths. If one attempts to correct v' by reorthogonalizing it, that is by applying the process again with v replaced by v', then one gets (approximately)

$$s = Q^T v' \quad \text{and} \quad v'' = v' - Qs = v - Q(r + s).$$

Comparing this with the desired result, $v' = v - Qr$, one sees that v' should be replaced by v'' and r by $r + s$. If $\|v''\|/\|v'\|$ is not too small, then v'' may be safely scaled to give a satisfactory q. Otherwise the process is repeated.

Thus, if η is a parameter satisfying $0 << \eta < 1$, for instance $\eta = 1/\sqrt{2}$, we have the tentative algorithm:

$$r^0 = 0, \quad v^0 = v,$$

for $k = 1, 2, 3, \ldots$ until $\|v^k\| > \eta\|v^{k-1}\|$

$$\begin{bmatrix} s^k = Q^T v^{k-1}, & r^k = r^{k-1} + s^k, \\ u^k = Qs^k, & v^k = v^{k-1} - u^k, \end{bmatrix}$$

$$r = r^k, \quad \rho = \|v^k\|, \quad q = v^k/\rho.$$

It is unlikely that this iterative reorthogonalization process would fail to terminate, for ultimately rounding errors would force some iterate v^k to have substantial components (relative to $\|v^{k-1}\|$) orthogonal to the range of Q. However, this is only a "probabilistic" argument and so in Section 4 we shall give a completely rigorous alternative.

3. The Updating Algorithms.

Givens Matrices. A Givens matrix is a matrix of the form

$$G = \begin{pmatrix} \gamma & \sigma \\ \sigma & -\gamma \end{pmatrix}: \gamma = \cos\theta, \quad \sigma = \sin\theta;$$

G is orthogonal and symmetric and $\det G = -1$. If $x = (\xi_1, \xi_2)^T$, then Gx is the reflection of x in the line which meets the axis $\xi_1 \geqslant 0$ in the angle $\theta/2$. The angle θ can be chosen so that

$$Gx = \tau e_1 = (\tau, 0)^T, \quad \tau = \pm\|x\|.$$

If $\xi_2 = 0$, we take $\theta = 0$ so that $\gamma = 1$ and $\sigma = 0$. Otherwise, we compute

$$\mu = \max\{|\xi_1|, |\xi_2|\}, \qquad |\tau| = \mu \operatorname{sqrt}[(\xi_1/\mu)^2 + (\xi_2/\mu)^2],$$

$$\tau = \pm|\tau|, \qquad \gamma = \xi_1/\tau \quad \text{and} \quad \sigma = \xi_2/\tau.$$

This computation of $|\tau| = \|x\|$ avoids artificial problems of overflow and underflow; a corresponding device will be used to compute the length of any vector. The sign of τ remains unspecified.

The computation of $z = Gy$, $y = (\eta_1, \eta_2)^T$, may be done rather efficiently as follows. First compute $\nu = \sigma/(1 + \gamma)$, and then

$$\begin{pmatrix} \zeta_1 \\ \zeta_2 \end{pmatrix} = \begin{pmatrix} \gamma & \sigma \\ \sigma & -\gamma \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} = \begin{pmatrix} \gamma\eta_1 + \sigma\eta_2 \\ \nu(\eta_1 + \zeta_1) - \eta_2 \end{pmatrix}.$$

If G is applied to a $2 \times n$ matrix in this way, the cost is $3n$ multiplications and additions, instead of the usual $4n$ multiplications and $2n$ additions. Finally, the sign of τ is chosen so no cancellation occurs in the formation of ν:

$$\tau = |\tau| \operatorname{sign} \xi_1, \qquad \operatorname{sign} \xi \equiv \begin{cases} 1, & \xi \geqslant 0, \\ -1, & \xi < 0. \end{cases}$$

Of course, by a trivial modification, G can be chosen so Gx is a scalar multiple of the second axis vector $e_2 = (0, 1)^T$. Also, we shall actually use $n \times n$ Givens matrices $G_{i,j}$ which deviate from the identity only in the submatrix G formed from rows and columns i and j.

General Rank One Updates. Let

$$A = \mathrm{QR} \in \mathbf{R}^{m \times n} \qquad (m > n), \quad u \in \mathbf{R}^n \quad \text{and} \quad v \in \mathbf{R}^m.$$

Observe that

$$\bar{A} \equiv A + vu^T = (Q, v) \begin{pmatrix} R \\ u^T \end{pmatrix}.$$

Step 1. Apply the Gram-Schmidt process (with reorthogonalization) to obtain

$$(Q, v) = (Q, q) \begin{pmatrix} I & r \\ 0 & \rho \end{pmatrix}, \qquad Q^T q = 0, \qquad \|q\| = 1.$$

We then have

$$\bar{A} = (Q, q) \left[\begin{pmatrix} R \\ 0^T \end{pmatrix} + \begin{pmatrix} r \\ \rho \end{pmatrix} u^T \right] \equiv \tilde{Q}\tilde{R},$$

and $\tilde{Q}$ has orthonormal columns.

Step 2. Choose Givens matrices $G_{n,n+1}, G_{n-1,n}, \ldots, G_{1,2}$ so that

$$G \begin{pmatrix} r \\ \rho \end{pmatrix} \equiv G_{1,2} \cdots G_{n,n+1} \begin{pmatrix} r \\ \rho \end{pmatrix} \equiv \tau e_1.$$

That is, choose the $G_{i,i+1}$ $(i = n, n-1, \ldots, 1)$ to successively introduce zeros into the vector from the bottom element through the second. The matrix G is orthogonal. The $(n+1) \times n$ matrix

$$G\begin{pmatrix} R \\ 0^T \end{pmatrix} \equiv G_{1,2} \cdots G_{n,n+1}\begin{pmatrix} R \\ 0^T \end{pmatrix} \equiv R'$$

is upper Hessenberg (= almost triangular), and so is

$$G\tilde{R} = R' + \tau e_1 u^T \equiv \hat{R}.$$

Moreover, by the orthogonality of G, the matrix $\tilde{Q}G^T = \tilde{Q}G_{n,n+1} \cdots G_{1,2} \equiv \hat{Q}$ has orthonormal columns and $\bar{A} = \hat{Q}\hat{R}$.

Step 3. Choose Givens matrices $H_{1,2}, H_{2,3}, \ldots, H_{n,n+1}$ to successively annihilate the subdiagonal elements of $\hat{R}$, giving

$$H\hat{R} \equiv H_{n,n+1} \cdots H_{1,2}\hat{R} \equiv \begin{pmatrix} \bar{R} \\ 0^T \end{pmatrix}$$

with $\bar{R}$ upper triangular. Then

$$\hat{Q}H^T = \hat{Q}H_{1,2} \cdots H_{n,n+1} \equiv (\bar{Q}, \bar{q})$$

has orthonormal columns and

$$\bar{A} = (\bar{Q}, \bar{q})\begin{pmatrix} \bar{R} \\ 0^T \end{pmatrix} = \bar{Q}\bar{R},$$

as required.

This algorithm uses approximately $2(l+3)mn + 3n^2$ multiplications and additions, where l is the number of orthogonalization steps ($l-1$ *re*orthogonalizations). The algorithm appears not to be valid for $m = n$, but in this case it actually simplifies. For then Q is orthogonal, and so

$$\bar{A} = Q(R + ru^T), \qquad r \equiv Q^T v.$$

Steps 2 and 3 apply, with one fewer Givens transformation each, to achieve the result.

Deleting a Column. Let

$$A = QR \in \mathbf{R}^{m \times n} \qquad (m \geqslant n)$$

and let a be the kth column of A, so that

$$A \equiv (A_1, a, A_2) \equiv Q(R_1, r, R_2).$$

Then

$$a = Qr \quad \text{and} \quad \bar{A} \equiv (A_1, A_2) = Q(R_1, R_2) \equiv Q\hat{R}.$$

The matrix $\hat{R}$ is upper Hessenberg. For instance, when $n = 6$ and $k = 3$, we have

$$\hat{R} = \begin{pmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{pmatrix},$$

where x denotes a possibly nonnull element and the elements not indicated are null. Thus, only Step 3 is needed, and this simplifies. We choose Givens matrices $H_{k,k+1}$, $H_{k+1,k+2}, \ldots, H_{n-1,n}$ so that

$$H\hat{R} = H_{n-1,n} \cdots H_{k,k+1}\hat{R} = \begin{pmatrix} \bar{R} \\ 0^T \end{pmatrix}$$

with $\bar{R}$ upper triangular. Then

$$QH^T = QH_{k,k+1} \cdots H_{n-1,n} \equiv (\bar{Q}, \bar{q})$$

has orthonormal columns and $\bar{A} = \bar{Q}\bar{R}$, as required.

This algorithm uses about $3[m + (n-k)/2](n-k)$ multiplications and additions together with mk more if, as will be done in our ALGOL code, the deleted column vector $a = Qr$ is retrieved.

Inserting a Column. Let

$$A \equiv (A_1, A_2) = Q(R_1, R_2) \equiv QR \in \mathbf{R}^{m \times (n-1)} \qquad (m \geqslant n)$$

with $A_1 \in \mathbf{R}^{m \times (k-1)}$ and $A_2 \in \mathbf{R}^{m \times (n-k)}$. If the vector $a \in \mathbf{R}^m$ is inserted "between" A_1 and A_2, it becomes the kth column of

$$\bar{A} \equiv (A_1, a, A_2) = (Q, a)\begin{pmatrix} R_1 & 0 & R_2 \\ 0^T & 1 & 0^T \end{pmatrix}.$$

We apply Step 1 to get

$$(Q, a) = (Q, q)\begin{pmatrix} I & r \\ 0 & \rho \end{pmatrix}, \qquad Q^T q = 0, \qquad \|q\| = 1.$$

Then

$$\bar{A} = (Q, q)\begin{pmatrix} R_1 & r & R_2 \\ 0^T & \rho & 0^T \end{pmatrix} \equiv \tilde{Q}\tilde{R},$$

where $\tilde{R}$ is of the form $(n = 7, k = 3)$

$$\tilde{R} = \begin{pmatrix} x & x & x & x & x & x & x \\ & x & x & x & x & x & x \\ & & x & x & x & x & x \\ & & x & & x & x & x \\ & & x & & & x & x \\ & & x & & & & x \\ & & x & & & & \end{pmatrix}.$$

Step 2 is consequently simplified and Step 3 is not needed. We choose Givens matrices $G_{n-1,n}$, $G_{n-2,n-1}, \ldots, G_{k,k+1}$ so that

$$G\widetilde{R} \equiv G_{k,k+1} \cdots G_{n-1,n}\widetilde{R} \equiv \overline{R}$$

is upper triangular. This fills only the diagonal positions in columns $n, n-1, \ldots, k+1$. Then

$$\widetilde{Q}G^T = \widetilde{Q}G_{n-1,n} \cdots G_{k,k+1} \equiv \overline{Q}$$

has orthonormal columns and $\overline{A} = \overline{Q}\overline{R}$, as required.

This algorithm uses approximately $2lmn + 3[m + (n-k)/2](n-k)$ multiplications and additions.

Inserting a Row. Let

$$A = QR \in \mathbf{R}^{(m-1)\times n} \qquad (m > n) \quad \text{and} \quad a \in \mathbf{R}^n.$$

Without loss of generality we may *append* a^T to A. Then

$$\overline{A} \equiv \begin{pmatrix} A \\ a^T \end{pmatrix} = \begin{pmatrix} Q & 0 \\ 0^T & 1 \end{pmatrix}\begin{pmatrix} R \\ a^T \end{pmatrix} \equiv \widetilde{Q}\widetilde{R}.$$

Here $\widetilde{Q}$ already has orthonormal columns so Step 1 can be avoided. So can Step 2 provided Step 3 is altered only slightly. We choose Givens matrices $H_{1,n+1}$, $H_{2,n+1}$, $\ldots, H_{n,n+1}$ so that

$$H\widetilde{R} \equiv H_{n,n+1} \cdots H_{1,n+1}\widetilde{R} \equiv \begin{pmatrix} \overline{R} \\ 0^T \end{pmatrix}$$

with $\overline{R}$ upper triangular. Then

$$\widetilde{Q}H^T = \widetilde{Q}H_{1,n+1} \cdots H_{n,n+1} \equiv (\overline{Q}, \overline{q})$$

has orthonormal columns and $\overline{A} = \overline{Q}\overline{R}$, as required.

This algorithm uses approximately $3(m + n/2)n$ multiplications and additions.

Deleting a Row. Again, we may delete the last row. Let

$$A \equiv \begin{pmatrix} \overline{A} \\ a^T \end{pmatrix} \equiv \begin{pmatrix} Q \\ q^T \end{pmatrix} R \in \mathbf{R}^{m\times n} \qquad (m > n)$$

with

$$(Q^T, q)\begin{pmatrix} Q \\ q^T \end{pmatrix} = Q^TQ + qq^T = I_n.$$

Now also,

$$\begin{pmatrix} \overline{A} \\ a^T \end{pmatrix} = \begin{pmatrix} Q & 0 \\ q^T & 1 \end{pmatrix}\begin{pmatrix} R \\ 0^T \end{pmatrix}.$$

Apply Step 1, the Gram-Schmidt process (with reorthogonalization), to obtain

$$\begin{pmatrix} Q & 0 \\ q^T & 1 \end{pmatrix} = \begin{pmatrix} Q & \tilde{q} \\ q^T & \sigma \end{pmatrix} \begin{pmatrix} I & r \\ 0 & \rho \end{pmatrix}.$$

The first iteration simplifies, since

$$r = (Q^T, q)\begin{pmatrix} 0 \\ 1 \end{pmatrix} = q,$$

and then

$$v' = \begin{pmatrix} \tilde{q} \\ \sigma \end{pmatrix} \rho = \begin{pmatrix} 0 \\ 1 \end{pmatrix} - \begin{pmatrix} Q \\ q^T \end{pmatrix} q = \begin{pmatrix} -Qq \\ 1 - q^T q \end{pmatrix}.$$

Since $\|\tilde{q}\|^2 + \sigma^2 = 1$, we have

$$\begin{aligned} \rho^2 &= \|Qq\|^2 + (1 - q^T q)^2 = q^T(I - qq^T)q + (1 - q^T q)^2 \\ &= 1 - q^T q = \sigma\rho. \end{aligned}$$

If $\rho \neq 0$, then $\sigma = \rho$. If $\rho = 0$, then $q^T q = 1$; and since $q^T q + \sigma^2 \leqslant 1$, we have $\sigma = \rho$ in any case. Step 1 thus provides

$$\begin{pmatrix} Q & 0 \\ q^T & 1 \end{pmatrix} = \begin{pmatrix} Q & \tilde{q} \\ q^T & \rho \end{pmatrix} \begin{pmatrix} I & q \\ 0 & \rho \end{pmatrix} \quad \text{with } \tilde{q}\rho = -Qq.$$

Hence, we have

$$\begin{pmatrix} \bar{A} \\ a^T \end{pmatrix} = \begin{pmatrix} Q & \tilde{q} \\ q^T & \rho \end{pmatrix} \begin{pmatrix} R \\ 0^T \end{pmatrix}.$$

We now choose Givens matrices $H_{n,n+1}, H_{n-1,n+1}, \cdots, H_{1,n+1}$ so that

$$(q^T, \rho)H^T \equiv (q^T, \rho)H_{n,n+1} \cdots H_{1,n+1} \equiv (0^T, \tau).$$

Moreover, $\tau = \pm 1$ since orthogonal transformations preserve length. Now the matrix

$$\begin{pmatrix} Q & \tilde{q} \\ q^T & \rho \end{pmatrix} H^T \equiv \begin{pmatrix} \bar{Q} & \bar{q} \\ 0^T & \tau \end{pmatrix}$$

has orthonormal columns and so $\bar{q} = 0$. Finally, we have

$$H\begin{pmatrix} R \\ 0^T \end{pmatrix} = H_{1,n+1} \cdots H_{n,n+1}\begin{pmatrix} R \\ 0^T \end{pmatrix} \equiv \begin{pmatrix} \bar{R} \\ \pm \bar{r}^T \end{pmatrix},$$

with $\bar{R}$ upper triangular, and

$$\begin{pmatrix} \bar{A} \\ a^T \end{pmatrix} = \begin{pmatrix} \bar{Q} & 0 \\ 0^T & \pm 1 \end{pmatrix} \begin{pmatrix} \bar{R} \\ \pm \bar{r}^T \end{pmatrix} = \begin{pmatrix} \bar{Q}\bar{R} \\ \bar{r}^T \end{pmatrix},$$

as required.

This algorithm uses about $2(l+1)mn + 3n^2/2$ multiplications and additions, where l is the number of orthogonalization iterations.

4. Construction of the Orthogonalization Code.

LEMMA 4.1 (ON MATRIX BY VECTOR MULTIPLICATION). *Let $A \in \mathbf{R}^{m\times n}$, $x \in \mathbf{R}^n$ and let $y \in \mathbf{R}^m$ be the result of the algorithm*

$$\begin{aligned} &y_0 = 0,\\ &\text{for } k = 1, 2, \ldots, n\\ &\quad y_k = y_{k-1} + a_k\xi_k + e_k,\\ &y = y_n, \end{aligned}$$

in which the (error) vectors $\{e_k\}_1^n$ satisfy

$$\|e_k\| \leqslant \alpha\|y_{k-1}\| + \beta\|a_k\|\,|\xi_k|.$$

Then $y = Ax + e$ with

$$\|e\| \leqslant [(n-1)\alpha + \min\{m^{1/2}, n^{1/2}\}\beta](1+\alpha)^{n-1}\|A\|\,\|x\|.$$

Proof. By induction on k,

$$y_k = Ax_k + \sum_1^k e_j, \qquad x_k \equiv (\xi_1, \ldots, \xi_k, 0, \ldots, 0)^T, \qquad \|y_k\| \leqslant \|Ax_k\| + \sum_1^k \|e_j\|,$$

$$\|e_{k+1}\| \leqslant \alpha\|Ax_k\| + \beta\|a_{k+1}\|\,|\xi_{k+1}| + \alpha\sum_1^k \|e_j\|$$

and

$$\begin{aligned} \sum_1^{k+1}\|e_j\| &\leqslant \alpha\|Ax_k\| + \beta\|a_{k+1}\|\,|\xi_{k+1}| + (1+\alpha)\sum_1^k \|e_j\|\\ &\leqslant \alpha\sum_1^k (1+\alpha)^{k-j}\|Ax_j\| + \beta\sum_1^{k+1}(1+\alpha)^{k+1-j}\|a_j\|\,|\xi_j|\\ &\leqslant (1+\alpha)^k\left(\alpha\sum_1^k\|Ax_j\| + \beta\sum_1^{k+1}\|a_j\|\,|\xi_j|\right). \end{aligned}$$

The result now follows from

$$e = \sum_1^n e_j, \qquad \|Ax_j\| \leqslant \|A\|\,\|x_j\| \leqslant \|A\|\,\|x\|$$

and

$$\sum_1^n \|a_j\|\,|\xi_j| \leqslant \|A\|_F\|x\|, \qquad \|A\|_F \equiv \left(\sum_1^n \|a_j\|^2\right)^{1/2} \leqslant \min\{m^{1/2}, n^{1/2}\}\|A\|.$$

Applications. If y, a and ξ are floating point vectors and scalar, respectively, and $y' = y + a\xi + e'$ is computed in floating point arithmetic [10], [1], [6] then a typical element satisfies

$$\eta' = \eta + \alpha\xi + \epsilon' = \eta(1 + \delta'') + \alpha\xi(1 + \delta)(1 + \delta'),$$

where δ, δ' and δ'' are rounding errors; thus

$$\epsilon' = \eta\delta'' + \alpha\xi(\delta + \delta' + \delta\delta').$$

We assume rounded arithmetic operations and denote the basic machine unit by δ_0 $(=2^{-t})$. Quantities which are in practice only slightly larger than δ_0 will be denoted by $\delta_1, \delta_2, \delta_3, \ldots$; in the same vein we put $\delta_{-1} \equiv \delta_0/(1 + \delta_0)$. When $\xi = 1$ (vector addition), we have $\delta = 0$.

a. (*Weak*) *Single Precision* (possibly large relative error in addition). Here we have $|\delta| \leqslant \delta_{-1}$, $|\delta'| \leqslant 3\delta_0/2$ and $|\delta''| \leqslant 3\delta_0/2$. Thus,

$$|\epsilon'| \leqslant \frac{3}{2}\delta_0|\eta| + \frac{5}{2}\delta_0'|\alpha\xi|, \qquad \delta_0' \equiv \delta_0\left(1 + \frac{1}{5}\delta_{-1}\right).$$

Hence, in Lemma 4.1 we can take

$$\|e_k\| \leqslant \frac{3}{2}\delta_0\|y_{k-1}\| + \frac{5}{2}\delta_0'\|a_k\|\,|\xi_k|$$

to obtain $y = Ax + e$ with

$$\|e\| \leqslant \frac{1}{2}[3(n - 1) + 5\min\{m^{1/2}, n^{1/2}\}]\delta_1\|A\|\,\|x\|$$

and

$$\delta_1 \equiv \delta_0\left(1 + \frac{3}{2}\delta_0\right)^n.$$

b. *Inner Products Computed in Double Precision* (followed by rounding to single precision). If y is the unrounded vector, then Lemma 4.1 applies, with

$$\|e_k\| \leqslant \frac{3}{2}\delta_0^2(\|y_{k-1}\| + \|a_k\|\,|\xi_k|),$$

to provide a bound for $\|y - Ax\|$. The double precision vector y is then rounded to yield the single precision vector z for which $\|z - y\| \leqslant \delta_0\|y\|$. From the triangle inequality it follows that $z = Ax + f$ with

$$\|f\| \leqslant \delta_0\|Ax\| + \frac{3}{2}(n + \min\{m^{1/2}, n^{1/2}\})\delta_2^2\|A\|\,\|x\|$$

and

$$\delta_2^2 \equiv \delta_0^2(1 + \delta_0)\left(1 + \frac{3}{2}\delta_0^2\right)^n.$$

These bounds, which do not appear in [10], [11], [1], [9] for instance, are basic to our further analysis.

In the following the Wilkinson symbols fl and fl_2 indicate the use of (weak) single precision and accumulated inner products, respectively.

THEOREM 4.1. *Let* $Q \in \mathbf{R}^{m\times n}$ $(m > n)$ and $v \in \mathbf{R}^m$ *have floating point entries, and let the vectors* $\{v^k\}_0^\infty$ *be computed from the algorithm*:

$$v^0 = v,$$

for $k = 1, 2, 3, \ldots$

$$\left|\begin{aligned} s^k &= \mathrm{fl}_2(Q^T v^{k-1}), \\ u^k &= \mathrm{fl}(Qs^k), \\ v^k &= v^{k-1} - u^k. \end{aligned}\right.$$

If

$$Q^T Q \equiv I + E, \qquad \|E\| \leqslant \epsilon, \qquad \gamma \equiv (1+\epsilon)^{1/2},$$

$$\alpha \geqslant \frac{3}{2}\gamma\delta \quad \textit{and} \quad \beta \equiv \epsilon + \frac{3}{2}(n^{1/2}+1)^2\gamma^2\delta,$$

where δ *(in practice only slightly larger than the basic machine unit* δ_0*) is defined below, then the following inequalities hold*:

1. $\|Q^T v^{k+1}\| \leqslant \alpha\|v^k\| + \beta\|Q^T v^k\|$;
2. $\|v^{k+1}\| \leqslant [\|v^k\|^2 - (1-\epsilon)\|Q^T v^k\|^2]^{1/2} + \alpha\|v^k\| + \beta\|Q^T v^k\|$;
3. $\|v^{k+1}\| \geqslant [\|v^k\|^2 - (1+\epsilon)\|Q^T v^k\|^2]^{1/2} - \alpha\|v^k\| - \beta\|Q^T v^k\|$
   provided $\gamma\|Q^T v^k\| \leqslant \|v^k\|$.

Likewise, if fl_2 *is replaced by* fl, *and*

$$\alpha \geqslant \frac{3}{2}(m^{1/2}+1)^2\gamma^2\delta.$$

Proof. We may suppose $k = 1$. We elaborate only on the fl_2 case, putting $r \equiv s'$ and $u \equiv u'$. From the applications of Lemma 4.1 we have

$$r = Q^T v + c, \qquad u = Qr + e, \qquad v' = v - u + f,$$

with

$$\|c\| \leqslant \delta_0\|Q^T v\| + \frac{3}{2}(m + n^{1/2})\delta_3^2\|Q\|\,\|v\|,$$

$$\delta_3^2 \equiv \delta_0^2(1+\delta_0)\left(1 + \frac{3}{2}\delta_0^2\right)^m, \qquad \|e\| \leqslant \frac{1}{2}(3n + 5n^{1/2} - 3)\delta_1\|Q\|\,\|r\|$$

and

$$\|f\| \leqslant \frac{1}{2}(\|u\| \leqslant \|v\|).$$

Elimination of u from the above equalities and inequalities gives

$$r = Q^T v + c, \qquad v' = v - Qr + g,$$

with $g \equiv e - f$ and

$$\|g\| \leqslant \frac{3}{2}\delta_0\|v\| + \frac{3n + 5n^{1/2}}{2}\delta_1\|Q\|\,\|r\|.$$

(We have actually used a slightly sharper bound for $\|e\|$ to avoid introducing δ_4.)

Eliminating r in a similar manner, we find $v' = (I - QQ^T)v - h$ with $h \equiv Qc + g$,

$$\|h\| \leqslant \frac{3}{2}\delta_4\|v\| + \frac{3}{2}(n^{1/2}+1)^2\delta_5\|Q\|\,\|Q^Tv\|,$$

$$\delta_4 \equiv \delta_0 + (m+n^{1/2})\delta_3^2\left(1+\frac{3n+5n^{1/2}}{2}\delta_1\right)(1+\epsilon)$$

and

$$\delta_5 \equiv \delta_0(1+\delta_0)\left(1+\frac{3}{2}\delta_0\right)^n.$$

In the single precision case a corresponding rather precise bound is

$$\|h\| \leqslant \frac{3}{2}\delta_6\|Q\|[(m^{1/2}+1)^2\|Q\|\,\|v\| + (n^{1/2}+1)^2\|Q^Tv\|]$$

with

$$\delta_6 \equiv \delta_0\left(1+\frac{3}{2}\delta_0\right)^m\left(1+\frac{3n+5n^{1/2}}{2}\delta_1\right),$$

and we now define

$$\delta \equiv \max\{\delta_4, \delta_5, \delta_6\}.$$

Since

$$Q^Tv' = Q^Tv - (I+E)Q^Tv - Q^Th = -EQ^Tv - Q^Th,$$

we obtain the first bound by taking norms and using $\|Q\| \leqslant \gamma$. The remaining bounds follow from

$$|\|v'\| - \|(I-QQ^T)v\|| \leqslant \|h\|,$$

$$\|(I-QQ^T)v\|^2 = \|v\|^2 - \|Q^Tv\|^2 + (Q^Tv)^TEQ^Tv$$

and

$$|(Q^Tv)^TEQ^Tv| \leqslant \|E\|\,\|Q^Tv\|^2.$$

This completes the proof.

The quantities δ_k/δ_0 $(k \geqslant 0)$ are nondecreasing functions of δ_0, m, n and ϵ. For instance, if $\delta_0 \leqslant 5\cdot 10^{-7}$, $m \leqslant 10^5$, $n \leqslant 10^5$ and $\epsilon \leqslant 1$ we have $\delta < 1.11\delta_0$.

We shall now apply Theorem 4.1 to construct the orthogonalization code. We shall assume that the numbers α, β and ϵ are not extremely large. Restrictions on their size will be expressed in the form of certain "τ-conditions" which will hold for a wide range of practical cases. We shall see that the (provable) attainable limiting precision of the reorthogonalization process is determined by the (minimal) value of α. In practice this is about $3\delta_0/2$ when accumulated inner products are used and $3m\delta_0/2$ otherwise.

If $v \neq 0$ and $\|Q^Tv\|/\|v\| \leqslant \xi \leqslant 1/\gamma$, then Theorem 4.1.3 implies

$$\|v'\|/\|v\| \geqslant [1-(\gamma\xi)^2]^{1/2} - \alpha - \beta\xi.$$

The right side is positive if and only if

$$\psi(\xi) \equiv (\beta^2 + \gamma^2)^{1/2}\xi^2 + 2\alpha\beta\xi - (1 - \alpha^2) < 0.$$

Equivalently, by Descartes' rule, we must have $\alpha < 1$ and

$$\xi < \bar{\xi} \equiv \frac{1 - \alpha^2}{[(1 - \alpha^2)\gamma^2 + \beta^2]^{1/2} + \alpha\beta},$$

the positive zero of ψ. Hence, by Theorem 4.1.1, $\alpha < 1$ and $\xi < \bar{\xi}$ imply

$$\frac{\|Q^T v'\|}{\|v'\|} \leqslant \xi_1 \equiv \varphi(\xi) \equiv \frac{\alpha + \beta\xi}{[1 - (\gamma\xi)^2]^{1/2} - \alpha - \beta\xi}.$$

The function

$$\varphi\colon [0, \bar{\xi}) \longrightarrow [\underline{\xi}, +\infty), \qquad \underline{\xi} \equiv \alpha/(1 - \alpha),$$

is strictly increasing with reciprocal inverse function

$$\frac{1}{\varphi^{(-1)}(\xi)} = \frac{\alpha\beta(1 + \xi)^2 + \xi\{\beta^2(1 + \xi)^2 + \gamma^2[\xi^2 - \alpha^2(1 + \xi)^2]\}^{1/2}}{\xi^2 - \alpha^2(1 + \xi)^2}.$$

The fixed points of φ satisfy

$$\pi(\xi) \equiv \xi^2[(\gamma\xi)^2 - 1] + (1 + \xi)^2(\alpha + \beta\xi)^2 \equiv \xi^2\psi(\xi) + \chi(\xi) = 0,$$

and the polynomial χ has positive coefficients. By Descartes' rule π can have at most two positive zeros. Now $\pi(0) > 0$ and $\pi(\bar{\xi}) > 0$. For the limiting values $\alpha = \beta = 0$ and $\gamma = 1$ we have $\pi(\xi) = \xi^2(\xi^2 - 1)$ and the derivative $\pi'(\xi) = 2\xi(2\xi^2 - 1)$; hence, $\pi'(1/\sqrt{2}) = 0$ and $\pi(1/\sqrt{2}) = -\frac{1}{4} < 0$. For "general" values of α, β and γ we insist that $\pi(1/\sqrt{2}) < 0$, that is

$$\tau_1 \equiv \epsilon + (3 + 2\sqrt{2})(\sqrt{2}\alpha + \beta)^2 < 1.$$

This τ-condition implies that φ has exactly two fixed points ξ^* and ξ^{**} for which $\underline{\xi} < \xi^* < \xi^{**} < \bar{\xi}$. If the algorithm of Theorem 4.1 is started with any vector $v^0 \neq 0$ satisfying

$$\|Q^T v^0\|/\|v^0\| \leqslant \xi_0 < \xi^{**},$$

then it follows from the monotonicity and continuity of φ that

$$\|Q^T v^k\|/\|v^k\| \leqslant \xi_k \equiv \varphi(\xi_{k-1}) \longrightarrow \xi^* \qquad (k \longrightarrow \infty).$$

For practical values of α, β and ϵ the difference equation $\xi_{k+1} = \varphi(\xi_k)$ is extremely "stiff".

To set a sharp termination criterion we shall ultimately need a rather precise upper bound ξ^+ for ξ^*. For now suppose that

$$\xi^+ \equiv \theta\alpha > \xi^* \quad \text{with } \theta > 1$$

as yet unspecified. From the above we may terminate the iteration when

$$[\alpha\|v^{k-1}\| + \beta\|Q^T v^{k-1}\|]/\|v^k\| < \xi^+,$$

since the left side is at most ξ_k. Equivalently, we may terminate when

$$\|v^{k-1}\| + \omega\|Q^T v^{k-1}\| < \theta\|v^k\|, \qquad \omega \equiv \beta/\alpha.$$

The *termination parameters* (ω, θ) will be specified by the user. Increasing α corresponds to decreasing ω, or ϵ, and a weaker accuracy requirement.

We now investigate the possibility of nontermination. If $\|Q^T v^k\|/\|v^k\| \geqslant \xi^+$ then

$$\|Q^T v^{k-l}\|/\|v^{k-l}\| \geqslant \varphi^{(-l)}(\xi^+) \qquad (0 \leqslant l \leqslant n),$$

where $\varphi^{(-l)}$ is the lth compositional power of $\varphi^{(-1)}$. In other words

$$\|v^{k-l}\| \leqslant \|Q^T v^{k-l}\|/\varphi^{(-l)}(\xi^+) \qquad (0 \leqslant l \leqslant n).$$

Then, by induction using Theorem 4.1.1,

$$\xi^+\|v^k\| \leqslant \|Q^T v^0\| \prod_1^k [\beta + \alpha/\varphi^{(-l)}(\xi^+)].$$

In particular

$$\xi^+\|v^k\|/\|v^0\| \leqslant \gamma\rho^k, \qquad \rho \equiv \beta + \alpha/\varphi^{(-1)}(\xi^+).$$

Hence, if $\rho < 1$ and the termination criterion continually fails to be satisfied, then $v^k \longrightarrow 0$.

Our explicit expression for $\varphi^{(-1)}$ provides the means for studying ρ. The apparent difficulty in guaranteeing that $\rho < 1$ lies in the fact that if ξ^+ is extremely close to ξ^* $(\doteq\alpha)$, then the denominator

$$(\xi^+)^2 - \alpha^2(1 + \xi^+)^2 = \alpha^2[\theta^2 - (1 + \theta\alpha)^2]$$

can become extremely small. We now choose θ conveniently to make the term in brackets equal to unity. This gives

$$\theta = ((2 - \alpha^2)^{1/2} + \alpha)/(1 - \alpha^2)$$

which is about $\sqrt{2}$ for small α. Some simple estimates then show that $\rho < 1$ provided

$$\tau_2 \equiv \theta[(1 + \epsilon/2)\alpha + (\theta + 1)(1 + \alpha)\beta] < 1.$$

In fact $\rho < \tau_2$, so in practice ρ is substantially smaller than unity. Finally, the condition that $\xi^+ > \xi^*$ is implied by $\pi(\theta\alpha) < 0$, which reduces to

$$\tau_3 \equiv (1 + \epsilon)\theta^4\alpha^2 + \theta(2 + \theta\alpha)(1 + \theta\alpha)^2\beta < 1.$$

The practical reorthogonalization process will thus either terminate quickly or else we shall soon have $\|v^k\| \leqslant \sigma\|v^0\|$, where $\sigma > 0$ is a parameter somewhat smaller than the basic machine unit δ_0 (for instance $\sigma = \delta_0/10$). In this case v^k is certainly indistinguishable from rounding error; and if $v^k \neq 0$, we can legitimately replace v^k by $\|v^k\|e_l$, where the axis vector e_l is chosen so the lth row $e_l^T Q = (Q^T e_l)^T$ of Q has minimal length. (If $v^k = 0$, we replace v^k by e_l, but put $r = r^k$ and $\rho = 0$.) Our

final, and most stringent, τ-conditions will guarantee the convergence of this alternative procedure.

First, we have

$$\|Q\|_F^2 = \sum_1^m \|Q^T e_i\|^2 \geqslant m\|Q^T e_l\|^2;$$

and then,

$$m^{1/2}\|Q^T e_l\| \leqslant \|Q\|_F \leqslant n^{1/2}\|Q\| \leqslant \gamma n^{1/2}.$$

That is

$$\|Q^T e_l\|/\|e_l\| \leqslant \gamma(n/m)^{1/2}.$$

We now obtain a lower bound, ξ^-, for ξ^{**}. Since $\pi(\xi^{**}) = 0$, we have, from the quadratic formula,

$$(\xi^{**})^2 = \frac{1 + \{1 - 4[(\alpha + \beta\xi^{**})\gamma(1 + \xi^{**})]^2\}^{1/2}}{2\gamma^2}.$$

Since $\xi^{**} < \bar{\xi} < 1/\gamma < 1$, $\gamma < 1 + \epsilon/2$ and $\sqrt{1-\eta} > 1 - \eta$ for $0 < \eta < 1$, we find

$$\xi^{**} > \left[\frac{1 - 4(2 + \epsilon)(\alpha + \beta)^2}{1 + \epsilon}\right]^{1/2} \equiv \xi^-.$$

The alternative procedure thus converges if $\gamma(n/m)^{1/2} \leqslant \xi^-$, or equivalently

$$\tau_4' \equiv (1 + \epsilon)^2 n/m + 4(2 + \epsilon)(\alpha + \beta)^2 \leqslant 1.$$

For practically small values of α, β and ϵ this τ-condition is surely satisfied if m is sufficiently greater than n. However, when reorthogonalization is applied, we *always* have $m > n$ and some trivial rearrangement shows that the alternative procedure converges for *all* such m if

$$\tau_4 \equiv 2(n + 1)[\epsilon + 4(\alpha + \beta)^2] \leqslant 1.$$

For practical values of α and β this is roughly equivalent with $2(n + 1)\epsilon < 1$.

For minimal α (= $3\gamma\delta/2$ or $3(m^{1/2} + 1)^2\gamma^2\delta/2$) the numbers τ_k are increasing functions of δ_0, m, n and ϵ. The τ-conditions are all satisfied if, for instance, $\delta_0 \leqslant 5 \cdot 10^{-7}$, $n \leqslant 10^3$, $n < m \leqslant 10^4$ and $\epsilon < 10^{-4}$. Also, we can choose $\epsilon = n \max|\epsilon_{ij}|$, so in these cases we have $\epsilon \leqslant 10^{-4}$ provided $\max|\epsilon_{ij}| \leqslant n\delta_0$.

Although we shall not pursue the matter in detail, we wish to show how Theorem 4.1.2 can be used to obtain an upper bound for $\|Q^T v\|/\|v\|$ from a lower bound for $\|v'\|/\|v\|$. Thus, assume α, β and ϵ are sufficiently small, fix η so that $\alpha + \beta\gamma \leqslant \eta \leqslant 1$, and suppose

$$\eta \leqslant \|v'\|/\|v\|, \qquad \tilde{\xi} \equiv \|Q^T v\|/\|v\|.$$

Then

$$\eta \leqslant [1 - (1 - \epsilon)\tilde{\xi}^2]^{1/2} + \alpha + \beta\tilde{\xi},$$

or equivalently

$$(1 - \epsilon + \beta^2)\tilde{\xi}^2 - (\eta - \alpha)\beta\tilde{\xi} - [1 - (\eta - \alpha)^2] \leqslant 0.$$

Hence,

$$\frac{\|Q^T v\|}{\|v\|} \leqslant \frac{(\eta - \alpha)\beta + \{(1 - \epsilon)[1 - (\eta - \alpha)^2] + \beta^2\}^{1/2}}{1 - \epsilon + \beta^2} \equiv \xi.$$

As α, β and ϵ tend to zero we have $\xi \longrightarrow (1 - \eta^2)^{1/2}$. Theorem 4.1.1 then yields the bound

$$\frac{\|Q^T v'\|}{\|v'\|} \leqslant \frac{\alpha + \beta\xi}{\eta} \sim \frac{\alpha + (1 - \eta^2)^{1/2}\beta}{\eta}.$$

This *indicates* that our termination criterion with

$$\omega = 0, \quad \theta = 1/\eta, \quad 0 << \eta < 1,$$

is not unreasonable, especially when α and β are of comparable size. On the other hand, when accumulated inner products are used, we may have $\alpha << \beta$ for large n and we would have to take $\eta \doteq 1$ to *guarantee* a limiting precision of $\xi^+ \doteq \sqrt{2}\alpha$.

5. ALGOL Procedures and Numerical Results. The principal practical results of this paper are summarized in the following package of ALGOL procedures.

```
comment: ALGOL procedures for updating Gram-Schmidt QR factorizations;

begin
  integer base; real lnbase;
  real omega, theta, sigma; label fail;
  comment: These are global entities. base and lnbase, the base of the machine arith-
    metic and its natural logarithm, are used in the procedure length. The others are
    relevant to the procedure orthogonalize. omega and theta are used to specify the
    termination criterion and sigma is used to test for restarting. See Section 4. The
    error exit fail is taken if termination is not obtained in a reasonable number of
    iterations;

real procedure length(n, x);
value n; integer n; real array x;
comment: Computes the accumulated Euclidean length of x[1 : n].
  Can be coded in machine language for greater efficiency;
begin
  integer k; real s, t; double ss, tt;
  ss := 0; t := 0;
  for k := 1 step 1 until n do t := max(t, abs(x[k]));
  if t > 0 then
  begin
    t := base ↑ entier(ln(t)/lnbase);
    for k := 1 step 1 until n do
    begin
```

```
      tt := x[k]/t; ss := ss + tt↑2
      end k
   end;
   s := ss; length := t × sqrt(s)
end length;

procedure orthogonalize(m, n, Q, v, r, rho);
value m, n; integer m, n; real rho; real array Q, v, r;
comment: Assuming Q[1 : m, 1 : n] (m ⩾ n) has (nearly) orthonormal columns this
   procedure orthogonalizes v[1 : m] to the columns of Q, and normalizes the result if
   m > n. r[1 : n] is the array of "Fourier coefficients", and rho is the distance from
   v to the range of Q. r and its corrections are computed in double precision. For
   more detail see Sections 2 and 4;
begin
   Boolean restart, null; integer i, j, k; real rho 0, rho 1, t;
   double ss, qq, vv; real array u[1 : m], s[1 : n];
   label again, standardexit;
   restart := null := false;
   for j := 1 step 1 until n do r[j] := 0;
   rho := rho 0 := length(m, v);
   k := 0;
again:
   comment: Take a Gram-Schmidt iteration, ignoring r on later steps
      if previous v was null;
   for i := 1 step 1 until m do u[i] := 0;
   for j := 1 step 1 until n do
   begin
      ss := 0;
      for i := 1 step 1 until m do
      begin
         qq := Q[i, j]; vv := v[i]; ss := ss + qq × vv
      end i;
      s[j] := t := ss;
      for i := 1 step 1 until m do u[i] := u[i] + Q[i, j] × t
   end j;
   if ¬ null then
   for j := 1 step 1 until n do r[j] := r[j] + s[j];
   for i := 1 step 1 until m do v[i] := v[i] − u[i];
   rho 1 := length(m, v); t := length(n, s);
   k := k + 1;
   comment: Treat the special case m = n if necessary;
   if m = n then
   begin
      for i := 1 step 1 until m do v[i] := 0;
```

```
      rho := 0; go to standardexit
   end;
   comment: Test for nontermination;
   if rho 0 + omega × t ⩾ theta × rho 1 then
   begin
      comment: Exit to fail if too many iterations;
      if k > 4 then go to fail;
      comment: Restart if necessary;
      if ¬ restart ∧ rho 1 ⩽ rho × sigma then
      begin
         restart := true;
         comment: Find first row of minimal length of Q;
         for i := 1 step 1 until m do u[i] := 0;
         for j := 1 step 1 until n do
         for i := 1 step 1 until m do u[i] := u[i] + Q[i, j]↑2
         t := 2;
         for i := 1 step 1 until m do
         if u[i] < t then begin k := i; t := u[k] end;
         comment: Take correct action if v is null;
         if rho 1 = 0 then begin null := true; rho 1 := 1 end;
         comment: Reinitialize v and k;
         for i := 1 step 1 until m do v[i] := 0;
         v[k] := rho 1; k := 0
      end;
      comment: Take another iteration;
      rho 0 := rho 1; go to again
   end;
   comment: Normalize v and take the standard exit;
   for i := 1 step 1 until m do v[i] := v[i]/rho 1;
   if ¬ null then rho := rho 1 else rho := 0;
standardexit:
end orthogonalize;

procedure computereflector(x, y, c, s);
real x, y, c, s;
comment: Computes parameters for the Givens matrix G for which
   (x, y)G = (z, 0). Replaces (x, y) by (z, 0);
begin
   real t, u, v, mu;
   u := x; v := y;
   if v = 0 then begin c := 1; s := 0 end
   else
   begin
```

```
      mu := max(abs(u), abs(v));
      t := mu × sqrt((u/mu)↑2 + (v/mu)↑2);
      if u < 0 then t := −t;
      c := u/t; s := v/t; x := t; y := 0
   end
end computereflector;

procedure applyreflector(c, s, k, l, x, y, j);
value c, s, k, l; integer k, l, j; real c, s, x, y;
comment: When called with x := x[j] and y := y[j], this procedure replaces the two
   column matrix (x[k:l], y[k:l]) by (x[k:l], y[k:l])G, where G is the Givens matrix
   determined by c and s. Uses the Jensen device [8];
begin
   real t, u, v, nu;
   nu := s/(1 + c);
   for j := k step 1 until l do
   begin
      u := x; v := y; x := t := u × c + v × s; y := (t + u) × nu − v
   end j
end applyreflector;

procedure rankoneupdate(m, n, Q, R, u, v);
value m, n; integer m, n; real array Q, R, u, v;
comment: Updates the factorization A = Q[1 : m, 1 : n]R[1 : n, 1 : n] (m ⩾ n)
   when the outer product of v[1 : m] and u[1 : n] is added to A;
begin
   integer i, j, k; real c, s, rho; real array t[1 : n];
   orthogonalize(m, n, Q, v, t, rho);
   computereflector(t[n], rho, c, s);
   applyreflector(c, s, n, n, R[n, n], rho, j);
   applyreflector(c, s, 1, m, Q[i, n], v[i], i);
   for k := n − 1 step − 1 until 1 do
   begin
      computereflector(t[k], t[k + 1], c, s);
      applyreflector(c, s, k, n, R[k, j], R[k + 1, j], j);
      applyreflector(c, s, 1, m, Q[i, k], Q[i, k + 1], i)
   end k;
   for j := 1 step 1 until n do R[1, j] := R[1, j] + t[1] × u[j];
   for k := 1 step 1 until n − 1 do
   begin
      computereflector(R[k, k], R[k + 1, k], c, s);
      applyreflector(c, s, k + 1, n, R[k, j], R[k + 1, j], j);
      applyreflector(c, s, 1, m, Q[i, k], Q[i, k + 1], i)
   end k;
```

```
    computereflector(R[n, n], rho, c, s);
    applyreflector(c, s, 1, m, Q[i, n], v[i], i);
end rankoneupdate;

procedure deletecolumn(m, n, Q, R, k, v);
value m, n, k; integer m, n, k; real array Q, R, v;
comment: Updates the factorization A = Q[1 : m, 1 : n]R[1 : n, 1 : n] (m ⩾ n) when
    the kth column of A is deleted. Returns the deleted column in v[1 : m];
begin
    integer i, j, l; real c, s, t;
    for i := 1 step 1 until m do v[i] := 0;
    for l := 1 step 1 until k do
    begin
        t := R[l, k];
        for i := 1 step 1 until m do v[i] := v[i] + Q[i, l] × t
    end l;
    for l := k step 1 until n − 1 do
    begin
        computereflector(R[l, l + 1], R[l + 1, l + 1], c, s);
        applyreflector(c, s, l + 2, n, R[l, j], R[l + 1, j], j);
        applyreflector(c, s, 1, m, Q[i, l], Q[i, l + 1], i)
    end l;
    for j := k step 1 until n − 1 do
    for i := 1 step 1 until j do R[i, j] := R[i, j + 1];
    for i := 1 step 1 until n do R[i, n] := 0;
    for i := 1 step 1 until m do Q[i, n] := 0
end deletecolumn;

procedure insertcolumn(m, n, Q, R, k, v);
value m, n, k; integer m, n, k; real array Q, R, v;
comment: Updates the factorization A = Q[1 : m, 1 : n − 1]R[1 : n − 1, 1 : n − 1]
    (m ⩾ n) when the vector v[1 : m] is inserted between columns k − 1 and k of A;
begin
    integer i, j, l; real c, s; real array u[1 : n];
    for j := n − 1 step −1 until k do
    for i := 1 step 1 until j do R[i, j + 1] := R[i, j];
    for j := k + 1 step 1 until n do R[j, j] := 0;
    orthogonalize(m, n − 1, Q, v, u, u[n]);
    for i := 1 step 1 until m do Q[i, n] := v[i];
    for l := n − 1 step −1 until k do
    begin
        computereflector(u[l], u[l + 1], c, s);
        applyreflector(c, s, l + 1, n, R[l, j], R[l + 1, j], j);
        applyreflector(c, s, 1, m, Q[i, l], Q[i, l + 1], i)
```

```
    end l;
    for i := 1 step 1 until k do R[i, k] := u[i]
end insertcolumn;

procedure insertrow(m, n, Q, R, k, u);
value m, n, k; integer m, n, k; real array Q, R, u;
comment: Updates the factorization A = Q[1 : m - 1, 1 : n]R[1 : n, 1 : n]
    (m > n) when the vector u[1 : n] is inserted between rows k - 1
    and k of A;
begin
    integer i, j, l; real c, s; real array v[1 : m];
    for i := 1 step 1 until m do v[i] := 0; v[k] := 1;
    for l := 1 step 1 until n do
    begin
        for i := m - 1 step -1 until k do Q[i + 1, l] := Q[i, l]; Q[k, l] := 0;
        computereflector(R[l, l], u[l], c, s);
        applyreflector(c, s, l + 1, n, R[l, j], u[j], j);
        applyreflector(c, s, 1, m, Q[i, l], v[i], i)
    end l
end insertrow;

procedure deleterow(m, n, Q, R, k, u);
value m, n, k; integer m, n, k; real array Q, R, u;
comment: Updates the factorization A = Q[1 : m, 1 : n]R[1 : n, 1 : n] (m > n)
    when the kth row of A is deleted. Returns the deleted row in u[1 : n];
begin
    integer i, j, l; real c, s, t; real array v[1 : m];
    for i := 1 step 1 until m do v[i] := 0; v[k] := 1;
    orthogonalize(m, n, Q, v, u, t);
    for i := k step 1 until m - 1 do v[i] := v[i + 1];
    for l := n step -1 until 1 do
    begin
        for i := k step 1 until m - 1 do Q[i, l] := Q[i + 1, l]
        computereflector(t, u[l], c, s);
        applyreflector(c, s, l, n, u[j], R[l, j], j);
        applyreflector(c, s, 1, m - 1, v[i], Q[i, l], i);
        Q[m, l] := 0
    end l;
    for j := 1 step 1 until n do u[j] := t × u[j]
end deleterow;

procedure QRfactor(m, n, A, Q, R);
value m, n; integer m, n; real array A, Q, R;
comment: Computes a Gram-Schmidt QR factorization, Q[1 : m, 1 : n]R[1 : n, 1 : n],
    of A[1 : m, 1 : n] (m ⩾ n);
```

```
begin
  integer i, k; real array v[1 : m];
  for k := 1 step 1 until n do
  begin
    for i := 1 step 1 until m do v[i] := A[i, k]; insertcolumn(m, k, Q, R, k, v)
  end k
end QRfactor;
fail:
end Gram-Schmidt QR updating procedures;
```

Extensive tests with small matrices have been made to check the logic of our codes. We report in detail only the following larger tests with the view of obtaining numerical experience with the inevitable problem of error propagation.

Experiment 1. To test the numerical stability of the procedure orthogonalize, as well as its dependence on the termination criterion, we have constructed numerical Gram-Schmidt QR factorizations, Q_nR_n, of the Hilbert sections

$$H_n \equiv (1/(i+j-1)) \in \mathbf{R}^{100\times n}, \qquad n = 1, 2, \ldots, 100.$$

Since this is done by successively appending columns, no Givens transformations are used and the propagation of rounding errors is due solely to the orthogonalization process. Moreover, the (double precision) Frobenius norms

$$\|Q_nR_n - H_n\|_F \quad \text{and} \quad \|Q_n^TQ_n - I_n\|_F$$

are easily updated. A slight and trivial extension of our error analysis shows that, if the recommended termination pair (ω, θ) is used and the final scale factor ρ is computed using an accumulated inner product, then we have

$$\|Q_n^TQ_n - I_n\|_F \leqslant \kappa n^{1/2}\delta_0$$

with κ essentially independent of m and n. Our experiments were done on a Burroughs 6700 computer, for which $\delta_0 = 0.5/8^{12}$. Table 1 gives an indicative selection of the quantities

$$\bar{d}_n^k \equiv \|Q_nR_n - H_n\|_F/n^{1/2}\delta_0, \qquad \bar{e}_n^k \equiv \|Q_n^TQ_n - I\|_F/n^{1/2}\delta_0,$$

for three different termination criteria ($k = 1, 2, 3$): (1) (ω, θ) as prescribed in Section 4 with minimal α and $\epsilon = \|Q_n^TQ_n - I_n\|_F$, (2) $(\omega, \theta) \equiv (0, \sqrt{2})$ and (3) $(\omega, \theta) \equiv (0, 10)$.

TABLE 1

n	$\bar{d}_n^1$	$\bar{e}_n^1$	$\bar{d}_n^2$	$\bar{e}_n^2$	$\bar{d}_n^3$	$\bar{e}_n^3$
20	0.26	0.90	0.25	1.08	0.25	1.7
40	0.27	1.03	0.26	1.68	0.27	54.0
60	0.26	0.93	0.26	1.42	0.27	44.1
80	0.23	0.90	0.23	1.28	0.24	38.2
100	0.21	0.95	0.21	1.25	0.22	34.2

In cases one and two, for $n \geqslant 2$ and with only one exception, the number of orthogonalization iterations was constant at two for small values of n and three for large n. The jumps from two to three iterations occurred at $n = 12$ and $n = 38$, respectively. Case three was similar except that only one iteration was used for $n = 2$, and the jump then occurred earlier, at $n = 26$. A fourth run was made restricting the number of iterations to two (one reorthogonalization). For this we had

$$\bar{e}_n^4 > 10^{10} \quad \text{for } n \geqslant 45,$$

that is all orthonormality in the computed Q_n had been lost.

In earlier runs we had set the restart parameter σ *equal* to the basic unit δ_0. From about fifty calls to the procedure orthogonalize two restarts were observed to occur. Since restarting is expensive we have thus recommended a somewhat smaller value of σ, in order to give the probabalistic heuristic of Section 2 a fair chance. No restarts were observed in our experience with the code as given ($\sigma = \delta_0/10$).

Experiment 2. We now consider updating the numerical QR factorizations, $Q_m R_m$, of the Hilbert sections

$$H_m \equiv (1/(i + j - 1)) \in \mathbf{R}^{m \times 10}$$

by appending and dropping rows. After obtaining the initial factorization of H_{10} as above, that is with the procedure *QRfactor*, we used *insertrow* to append rows up to $m = 50$ and then *deleterow* to successively drop the last row and return to $m = 10$. The recommended termination parameters were used consistently, with minimal α and $\epsilon = \|Q_m^T Q_m - I_{10}\|_F$. Table 2 lists typical values of the magnified error norms

$$d_m \equiv \|Q_m R_m - H_m\|_F/\delta_0, \qquad e_m \equiv \|Q_m^T Q_m - I_{10}\|_F/\delta_0,$$

which no longer can be computed recursively. For ascending m we have also given the quantities

$$\tilde{d}_m \equiv \ln(d_m)/\ln(m), \qquad \tilde{e}_m \equiv \ln(e_m)/\ln(m).$$

TABLE 2

m	d_m	$\tilde{d}_m$	e_m	$\tilde{e}_m$
10	0.7	−0.13	3	0.54
20	10.4	0.78	37	1.20
30	18.9	0.86	65	1.23
40	32.1	0.94	88	1.21
50	51.9	1.01	123	1.23
40	52.0		122	
30	51.6		120	
20	50.6		118	
10	47.6		106	

For ascending m these results indicate error growth of roughly m in d_m and $m^{5/4}$ in e_m. For descending m both errors are moderately decreasing.

Departments of Computer Science and Mathematics
University of Texas at Austin
Austin, Texas 78712

Department of Mathematics
University of California, San Diego
La Jolla, California 92093

Department of Computer Science
University of Colorado
Boulder, Colorado 80302

Department of Computer Science
University of Maryland
College Park, Maryland 20742

1. GEORGE E. FORSYTHE & CLEVE B. MOLER, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, Englewood Cliffs, N. J., 1967. MR **36** #2306.

2. P. E. GILL, G. H. GOLUB, W. MURRAY & M. A. SAUNDERS, "Methods for modifying matrix factorizations," *Math. Comp.*, v. 28, 1974, pp. 505–535. MR **49** #8299.

3. PHILIP E. GILL, WALTER MURRAY & MICHAEL A. SAUNDERS, "Methods for computing and modifying the *LDV* factors of a matrix," *Math. Comp.*, v. 29, 1975, pp. 1051–1077.

4. W. B. GRAGG & G. W. STEWART, "A stable variant of the secant method for solving nonlinear equations," *SIAM J. Numer. Anal.*, v. 13, 1976.

5. ALSTON S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Blaisdell, New York, 1964. MR **30** #5475.

6. DONALD E. KNUTH, *The Art of Computer Programming.* Vol. 2: *Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1969. MR **44** #3531.

7. CHARLES L. LAWSON & RICHARD J. HANSON, *Solving Least Squares Problems*, Prentice-Hall Ser. in Automatic Computation, Prentice-Hall, Englewood Cliffs, N. J., 1974. MR **51** #2270.

8. HEINZ RUTISHAUSER, *Handbook for Automatic Computation.* Vol. 1/Part a: *Description of ALGOL* 60, Die Grundlehren der math. Wissenschaften, Band 135, Springer-Verlag, New York, 1967.

9. G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

10. J. H. WILKINSON, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, N. J., 1963. MR **28** #4661.

11. J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965. MR **32** #1894.