Numerics of Gram-Schmidt Orthogonalization

Å. Björck

Department of Mathematics
Linköping University
S-581 83 Linköping, Sweden

Submitted by José Dias da Silva

ABSTRACT

The Gram-Schmidt (GS) orthogonalization is one of the fundamental procedures in linear algebra. In matrix terms it is equivalent to the factorization $A = Q_1 R$, where $Q_1 \in \mathbf{R}^{m\times n}$ with orthonormal columns and R upper triangular. For the *numerical* GS factorization of a matrix A two different versions exist, usually called *classical* and *modified* Gram-Schmidt (CGS and MGS). Although mathematically equivalent, these have very different numerical properties. This paper surveys the numerical properties of CGS and MGS. A key observation is that MGS is *numerically* equivalent to Householder QR factorization of the matrix A augmented by an $n \times n$ zero matrix on top. This can be used to derive bounds on the loss of orthogonality in MGS, and to develop a backward-stable algorithm based on MGS. The use of reorthogonalization and iterated CGS and MGS algorithms are discussed. Finally, block versions of GS are described.

1. INTRODUCTION

Let $A \in \mathbf{R}^{m\times n}$, $m \geq n = \operatorname{rank}(A)$. The Gram-Schmidt orthogonalization produces Q_1 and R in the factorization

$$A = (a_1, \ldots, a_n) = Q_1 R, \qquad Q_1 = (q_1, \ldots, q_n),$$

where Q_1 has orthogonal columns and R is upper triangular. The columns of Q_1 in the factorization are obtained by successively orthogonalizing the columns of A. In this paper we survey a number of numerical properties of Gram-Schmidt orthogonalization. We show that in spite of a sometimes bad reputation the Gram-Schmidt algorithm has a number of remarkable properties that make it the algorithm of choice in a variety of applications.

655 Avenue of the Americas, New York, NY 10010 0024-3795/94/$7.00

In Section 2 we give several computational variants of Gram-Schmidt orthogonalization. These different versions have an interesting history. The "modified" Gram-Schmidt algorithm (MGS) was derived long ago by Laplace [15, §2] as an elimination method using weighted row sums; see Farebrother [9, Chapter 4]. However, Laplace did not interpret his algorithm in terms of orthogonalization, nor did he use it for computing least squares solutions. Bienaymé [3] gave a similar derivation of a slightly more general algorithm; see Section 5. (The idea of elimination with weighted row combinations also appears in Bauer [2], but without references to earlier sources.) What is now called the "classical" Gram-Schmidt algorithm (CGS) first appeared explicitly *later*, in a paper by Schmidt [25, p. 61], which treats the solution of linear systems with infinitely many unknowns. The orthogonalization is used here as a theoretical tool rather than a computational procedure.

The bad reputation of Gram-Schmidt orthogonalization as a numerical algorithm has arisen mostly because of the (sometimes catastrophic) loss of orthogonality which can occur. In Section 3 we comment on the remarkable fact that in this respect the classical and modified algorithms behave very differently. In MGS the loss of orthogonality occurs in a predictable manner and is related to the conditioning of the matrix A, which is not the case for CGS.

In Section 4 we outline a roundoff error analysis, which explains the superior stability of MGS. A key observation is that MGS is numerically equivalent to Householder QR factorization applied to the matrix A *augmented with a square matrix of zero elements on top*. Hence the computed $\bar{R}$ from MGS is numerically as good as that from the ordinary Householder QR factorization.

In Section 5 we give a backward-stable algorithms based on MGS for solving the least squares problem $\min_x \|Ax - b\|_2$. This algorithm is not new, and indeed goes back to Laplace [15]. However, the proof of backward stability is of recent origin. For the minimum norm problem $\min \|y\|_2$, subject to $A^T y = c$, a new backward stable algorithm is given.

If MGS is used only as a tool for computing pseudoinverse solutions to linear systems, then the loss of orthogonality in Q causes no problems. However, in some applications, it is essential that the computed Q be orthogonal to working accuracy. Then the Gram-Schmidt algorithm needs to be modified. In Section 6 we consider reorthogonalization and introduce the iterated CGS and MGS algorithms.

To obtain efficient implementations of matrix algorithms on modern computing systems it is necessary to consider block algorithms. In Section 7 we show how a block version of the Gram-Schmidt algorithm can be developed, which to some extent is simpler than the corresponding Householder version.

2. GRAM-SCHMIDT ORTHOGONALIZATION

In this section we review a number of different computational variants of the mathematical Gram-Schmidt orthogonalization, and first consider the modified version. In *row-oriented MGS* a sequence of matrices, $A = A^{(1)}$, $A^{(2)}, \ldots, A^{(n)}$ is computed, where $A^{(n)} = Q_1$, and $A^{(k)} \in \mathbf{R}^{m\times n}$ has the form

$$A^{(k)} = (q_1, \ldots, q_{k-1}, a_k^{(k)}, \ldots, a_n^{(k)}).$$

Here $a_k^{(k)}, \ldots, a_n^{(k)}$ have been made orthogonal to $q_1, \ldots, q_{k-1}$, which are final columns in Q_1. In the kth step we first obtain q_k by normalizing the vector $a_k^{(k)}$:

$$\tilde{q}_k = a_k^{(k)}, \qquad r_{kk} = (\tilde{q}_k^T \tilde{q}_k)^{1/2}, \qquad q_k = \tilde{q}_k / r_{kk}, \tag{1}$$

and then orthogonalize $a_{k+1}^{(k)}, \ldots, a_n^{(k)}$ against q_k:

$$a_j^{(k+1)} = a_j^{(k)} - r_{kj} q_k, \qquad r_{kj} = q_k^T a_j^{(k)}, \qquad j = k+1, \ldots, n. \tag{2}$$

The unnormalized vector $\tilde{q}_k$ is just the orthogonal projection of a_k onto the orthogonal complement of $\operatorname{span}[a_1, a_2, \ldots, a_{k-1}] = \operatorname{span}[q_1, q_2, \ldots, q_{k-1}]$. After n steps we have obtained the factorization $A = Q_1 R$, where the columns of Q_1 are orthonormal by construction.

It is possible to get a column-oriented version of the modified Gram-Schmidt algorithm by interchanging the order of computation in Algorithm 2.1 so that the column a_k is not transformed until the kth major step. We summarize these MGS algorithms below.

ALGORITHM 2.1. [Modified Gram-Schmidt (MGS) row version]. Given $A^{(1)} = A \in R^{m\times n}$ with $\operatorname{rank}(A) = n$, the following algorithm computes the factorization $A = Q_1 R$ using mn^2 flops:

$$
\begin{aligned}
&\text{for } k = 1, 2, \ldots, n\\
&\quad \hat{q}_k := a_k^{(k)}; \quad r_{kk} := (\hat{q}_k^T \hat{q}_k)^{1/2};\\
&\quad q_k := \hat{q}_k / r_{kk};\\
&\quad \text{for } j = k+1, \ldots, n\\
&\qquad r_{kj} := q_k^T a_j^{(k)}; \quad a_j^{(k+1)} := a_j^{(k)} - r_{kj} q_k;\\
&\quad \text{end}\\
&\text{end}
\end{aligned}
$$

ALGORITHM 2.2. [Modified Gram-Schmidt (MGS) column version]. Given $A^{(1)} = A \in R^{m\times n}$ with $\operatorname{rank}(A) = n$, the following algorithm computes Q_1 and R in the factorization $A = Q_1 R$:

$$
\begin{aligned}
&\text{for } k = 1, 2, \ldots, n\\
&\quad \text{for } i = 1, \ldots, k-1\\
&\qquad r_{ik} := q_i^T a_k^{(i)}; \quad a_k^{(i+1)} := a_k^{(i)} - r_{ik} q_i;\\
&\quad \text{end}\\
&\quad \hat{q}_k := a_k^{(k)}; \quad r_{kk} := (\hat{q}_k^T \hat{q}_k)^{1/2};\\
&\quad q_k := \hat{q}_k / r_{kk};\\
&\text{end}
\end{aligned}
$$

REMARK 1. The column- and row-oriented versions of the MGS algorithm are *numerically* equivalent. The operations and rounding errors are the same, and both produce the same numerical results. The row-oriented version is often preferred because it can be combined with column pivoting, which is necessary when treating rank-deficient problems. The column-oriented version of the MGS Algorithm 2.2 was used by Rutishauser [24] (see also Gander [10]) and independently derived by Longley [18]. It is appropriate to use when the columns are A are obtained sequentially, as, e.g., when used in Lanczos type algorithms.

REMARK 2. A square-root-free version of modified Gram-Schmidt orthogonalization results if the normalization of the vectors $\hat{q}_k$ is omitted. The changes in Algorithm 2.2 are to put

$$d_i := \hat{q}_i^T \hat{q}_i, \quad \hat{r}_{ik} := \hat{q}_i^T a_k^{(i)}/d_i$$

and subtract out $\hat{r}_{ik}\hat{q}_i$ instead of $r_{ik}q_i$, to get $A = \hat{Q}_1\hat{R}$ with $\hat{R}$ *unit* upper triangular. It is interesting to note that no corresponding square-root-free Householder orthogonalization method seems possible.

Another way to derive the Gram-Schmidt factorization is as follows. Assume that $q_1, \ldots, q_{k-1}$ have been determined. Then by orthogonality $q_i^T a_k^{(i)} = q_i^T a_k$, and we can compute

$$\hat{q}_k := a_k - \sum_{i=1}^{k-1} r_{ik} q_i, \qquad r_{ik} = q_i^T a_k, \quad i = 1, \ldots, k-1,$$

This leads to the classical Gram-Schmidt algorithm, where the factors Q_1 and R are generated column by column. Note that the column a_k is not used until the kth step.

ALGORITHM 2.3. [Classical Gram-Schmidt (CGS)]. Given $A \in R^{m\times n}$ with $\operatorname{rank}(A) = n$, the following algorithm computes for $k = 1, 2, \ldots, n$ the column q_k of Q_1 and the elements $r_{1k}, \ldots, r_{kk}$ of R in the factorization $A = Q_1 R$:

```
for k = 1, 2, ..., n
   for i = 1, ..., k - 1
      r_ik := q_i^T a_k;
   end
   q̂_k := a_k - Σ_{i=1}^{k-1} r_ik q_i;
   r_kk := (q̂_k^T q̂_k)^{1/2};   q_k := q̂_k / r_kk;
end
```

REMARK 3. In CGS the main work can be performed as a matrix-vector multiplication

$$r_k = Q_{k-1}^T a_k, \qquad \hat{q}_k = a_k - Q_{k-1} r_k,$$

where $Q_{k-1} = (q_1, \ldots, q_{k-1})$, and $r_k \in \mathbf{R}^{k-1}$ is the kth column in R (excluding the diagonal element). Hence CGS is better adapted to parallel computing than MGS.

REMARK 4. The difference between CGS and MGS is that in the modified algorithm the projections $r_{ik}q_i$ are subtracted from a_k before inner products with q_j, $j > i$, are computed. Although mathematically CGS and MGS are equivalent, they differ *numerically*. As a rule, *CGS should not be used numerically without reorthogonalization*. The superiority of the MGS algorithm over CGS for solving least squares problems was first experimentally established by Rice [22].

3. ORTHOGONALITY OF COMPUTED FACTOR

The Gram-Schmidt algorithms explicitly computes the matrix Q_1, which theoretically provides an orthogonal bases for $\mathcal{R}(A)$. This is in contrast to other numerical methods for computing the QR decomposition, in which Q is implicitly defined as a product of Householder or Givens matrices. However, due to round off, there will in general be a gradual loss of orthogonality in the computed vectors q_k, and the computed matrix $\bar{Q}_1$ will not be orthogonal to working accuracy. The reason for this is that cancellation takes place when the orthogonal projection on q_i is subtracted in

$$a_k^{(i+1)} := a_k^{(i)} - r_{ik} q_i.$$

We will show that this may lead to a serious loss of orthogonality in case

$$\|a_k^{(i+1)}\|_2 \ll \|a_k^{(i)}\|_2$$

To exhibit the loss of orthogonality it suffices to consider a case of orthogonalizing *two* vectors. Given vectors q_1 ($\|q_1\|_2 = 1$) and a_2, we want to compute

$$\hat{q}_2 = a_2 - r_{12}q_1, \qquad r_{12} = q_1^T a_2. \tag{3}$$

We assume the standard model for floating point computation

$$\mathrm{fl}(x \text{ op } y) = (x \text{ op } y)(1+\epsilon), \qquad |\epsilon| \le u,$$

where fl denotes floating point computation, op $= \{+, -, \cdot, /\}$, and u is the unit roundoff. Then it can be shown (see [5]) that

$$\|\mathrm{fl}(\hat{q}_2) - \hat{q}_2\|_2 < cu\|a_2\|_2, \qquad c = 1.06(2m+3).$$

(This estimate holds even when the normalization error is included.) Since $q_1^T\hat{q}_2 = 0$, it follows that $|q_1^T\mathrm{fl}(\hat{q}_2)| < cu\|a_2\|_2$. Hence the loss of orthogonality is proportional to

$$\frac{\|a_2\|_2}{\|\mathrm{fl}(\hat{q}_2)\|_2} \approx \frac{\|a_2\|_2}{\|\hat{q}_2\|_2} = \frac{1}{\sin\phi(q_1, a_2)},$$

where $\phi(q_1, a_2)$ is the angle between q_1 and a_2.

Remarkably, for MGS the loss of orthogonality can be bounded in terms of the condition number $\kappa(A)$ also for $n > 2$. (Note that for $n = 2$ MGS and CGS are the same.) In [5] it was proved that if $\bar{c}_2\kappa u < 1$, then

$$\|I - \bar{Q}_1^T\bar{Q}_1\|_2 \le \frac{\bar{c}_1}{1 - \bar{c}_2\kappa u}\kappa u. \tag{4}$$

Here, and in the following, $\bar{c}_i$ and c_i, $i = 1, 2, \ldots$, denote constants depending on m, n, and the details of the arithmetic. Hence the loss of orthogonality in $(q_1, \ldots, q_k)$ depends on $\kappa(a_1, \ldots, a_k)$, $k = 1, 2, \ldots, n$, and column pivoting may be used in maintaining orthogonality as long as possible.

In contrast, the computed vectors q_k from CGS may depart from orthogonality to an almost arbitrary extent. As pointed out by Gander [10], even computing Q_1 via the Cholesky decomposition $A^TA = R^TR$, $Q_1 = AR^{-1}$ seems to give better orthogonality than CGS. The more gradual loss of orthogonality in the computed vectors q_i for MGS is illustrated in the example below from Björck [5].

EXAMPLE 5. Consider the matrix in Läuchli [17],

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \epsilon & & \\ & \epsilon & \\ & & \epsilon \end{pmatrix},$$

and assume that $|\epsilon|$ is so small that $\mathrm{fl}(1+\epsilon^2) = 1$. (This assumption implies that $|\epsilon| \leq \sqrt{u}$, where u is the unit roundoff.) If no other rounding errors are made, then the orthogonal matrices computed by CGS and MGS respectively are

$$\hat{Q}_{\mathrm{CGS}} = \begin{pmatrix} 1 & 0 & 0 \\ \epsilon & -\epsilon & -\epsilon \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{pmatrix}, \qquad \hat{Q}_{\mathrm{MGS}} = \begin{pmatrix} 1 & 0 & 0 \\ \epsilon & -\epsilon & -\epsilon/2 \\ 0 & \epsilon & -\epsilon/2 \\ 0 & 0 & \epsilon \end{pmatrix},$$

where, for simplicity, we have omitted the normalization of $\hat{Q}$. It is easily verified that the maximum deviations from orthogonality of the computed columns are

$$\mathrm{CGS}: \ |q_3^T q_2| = \frac{1}{2}, \quad \mathrm{MGS}: \ |q_3^T q_1| = \left(\frac{2}{3}\right)^{1/2} |\epsilon| < \frac{\sqrt{2}}{3}\kappa(A)u,$$

The last inequality follows from $\kappa(A) = |\epsilon|^{-1}(3+\epsilon^2)^{1/2} \approx \sqrt{3}|\epsilon|^{-1}$. For CGS orthogonality has been completely lost. Note that in this example $\kappa(a_1, a_2) \approx \kappa(a_1, a_2, a_3)$. For MGS cancellation occurs only when orthogonalizing against q_1, and $q_3^T q_2 = 0$.

4. MGS AS A HOUSEHOLDER METHOD

A key observation for understanding the numerical properties of the modified Gram-Schmidt algorithm is that it can be interpreted as Householder QR factorization applied to the matrix A *augmented with a square matrix of zero elements on top*. These two algorithms are not only mathematically (see [16, Problem 19.39]), but also numerically equivalent. This key observation, apparently by Charles Sheffield, was relayed to the author in 1968 by Gene Golub. This relationship was studied in detail by Björck and Paige in [6]. We now outline the main results.

In the MGS method the columns are transformed by

$$a_j^{(k+1)} = M_k a_j^{(k)}, \qquad M_k = I - q_k q_k^T, \tag{5}$$

where M_k is the orthogonal projection onto the complement of q_k. In the Householder method one computes the factorization

$$P^T \begin{pmatrix} 0 \\ A \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad P^T = P_n \cdots P_2 P_1, \tag{6}$$

where

$$P_k = I - v_k v_k^T, \qquad v_k = \begin{pmatrix} -e_k \\ q_k \end{pmatrix}, \qquad \|v_k\|_2^2 = 2, \tag{7}$$

are Householder transformations. Because of the special structure of the augmented matrix the vectors v_k have a special form. Since the first n rows are initially zero, the scalar products of the vector v_k with later columns will only involve q_k, and it is easily verified that the quantities r_{kj} and q_k are *numerically* equivalent to the quantities in the modified Gram-Schmidt method (6)–(7).

Ideally, if $q_i^T q_j = 0$, $i \neq j$, the orthogonal matrix P in (6) has the form

$$P = P_1 \cdots P_n = I - \sum_{k=1}^{n} \begin{pmatrix} -e_k \\ q_k \end{pmatrix} (-e_k^T \; q_k^T) = \begin{pmatrix} O & Q_1^T \\ Q_1 & I - Q_1 Q_1^T \end{pmatrix},$$

and P is fully defined by its $(1,2)$ block $Q_1 = (q_1, \ldots, q_n)$. Using the computed $\bar{Q}_1 = (\bar{q}_1, \ldots, \bar{q}_n)$ the corresponding matrix $\bar{P}$ has the form (see [6, Theorem 4.1])

$$\bar{P} = \begin{pmatrix} \bar{P}_{11} & \bar{P}_{12} \\ \bar{P}_{21} & \bar{P}_{22} \end{pmatrix} = \begin{pmatrix} \bar{P}_{11} & (I - \bar{P}_{11})\bar{Q}_1^T \\ \bar{Q}_1(I - \bar{P}_{11}) & I - \bar{Q}_1(I - \bar{P}_{11})\bar{Q}_1^T \end{pmatrix},$$

where

$$\bar{P}_{11} = \begin{pmatrix} 0 & \bar{q}_1^T\bar{q}_2 & \bar{q}_1^T\bar{M}_2\bar{q}_3 & \cdots & \bar{q}_1^T\bar{M}_2\bar{M}_3 \cdots \bar{M}_{n-1}\bar{q}_n \\ 0 & 0 & \bar{q}_2^T\bar{q}_3 & \cdots & \bar{q}_2^T\bar{M}_3\bar{M}_4 \cdots \bar{M}_{n-1}\bar{q}_n \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \bar{q}_{n-1}^T\bar{q}_n \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

is upper triangular, and $\bar{M}_i = I - \bar{q}_i\bar{q}_i^T$, $i = 1, \ldots, n-1$.

The equivalence of MGS and Householder QR gives us a way to understand the numerical behavior of MGS. From Wilkinson's classical error analysis of Householder QR in [29] it follows that there exists an *exactly* orthogonal $\tilde{P}$ (not equal to $\bar{P}$) such that

$$\begin{pmatrix} E_1 \\ A + E_2 \end{pmatrix} = \tilde{P} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} = \begin{pmatrix} \tilde{P}_{11} \\ \tilde{P}_{21} \end{pmatrix} \bar{R}, \qquad \|E_i\|_2 \le c_i u \|A\|_2, \tag{8}$$

where c_i, $i = 1, 2$, are constants. Using the CS decomposition (see [11, p. 77]), we can write

$$\tilde{P}_{11} = U_1 C W^T, \qquad \tilde{P}_{21} = V_1 S W^T,$$

where $U = (U_1, U_2)$ and $V = (V_1, V_2)$ are square orthogonal matrices and $C > 0, S > 0$ diagonal with $C^2 + S^2 = 1$. Then it follows (see [6]) that

$$A + E = \tilde{Q}_1 \bar{R}, \qquad \|E\|_2 \le (c_1 + c_2) u \|A\|_2, \tag{9}$$

where $\tilde{Q}_1 = V_1 W^T$ is the closest orthogonal matrix to $\tilde{P}_{21}$ in any unitarily invariant norm. This shows that the computed $\bar{R}$ from MGS is numerically as good as that from the ordinary Householder QR factorization.

From (8) it follows that $\tilde{P}_{11} = E_1 \bar{R}^{-1}$, and assuming $c_3 u\kappa < 1$, where $c_3 = c_1 + c_2$, it is easily shown that

$$\|\tilde{P}_{11}\|_2 \le \frac{c_1}{1 - c_3 u\kappa} u\kappa.$$

This could be used to obtain a bound for the deviation from orthogonality of $\bar{Q}_1$. More simply, we can derive a slightly less sharp bound using the bound from [5],

$$A + E_0 = \bar{Q}_1 \bar{R}, \qquad \|E_0\|_2 \le c_0 u \|A\|_2, \tag{10}$$

involving the computed $\bar{Q}_1$ and $\bar{R}$. Combining this with (9), we get $\bar{Q}_1 = \tilde{Q}_1 + (E_0 + E)\bar{R}^{-1}$, giving

$$\|\tilde{Q}_1 - \bar{Q}_1\|_2 \le \frac{c_4}{1 - c_3 u\kappa} u\kappa, \qquad c_4 = c_0 + c_3.$$

From this we obtain a bound roughly the same as (4),

$$\|I - \bar{Q}_1^T \bar{Q}_1\|_2 \le \frac{2c_4}{1 - c_3 u\kappa} u\kappa + O((u\kappa)^2). \tag{11}$$

5. PSEUDOINVERSE SOLUTIONS WITH MGS

We first consider the use of the modified Gram-Schmidt algorithm for solving linear least squares problems. It is important to note that because of the loss of orthogonality in Q_1 that takes place also in MGS, computing $c_1 = Q_1^T b$ and then x from $Rx = c_1$ will not give an accurate solution. A backward-stable algorithm can be derived by applying the MGS algorithm to the matrix (A, b) to compute the factorization

$$(A, b) = (Q_1, q_{n+1}) \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}. \tag{12}$$

From this we have

$$Ax - b = (A, b) \begin{pmatrix} x \\ -1 \end{pmatrix} = (Q_1, q_{n+1}) \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix} \begin{pmatrix} x \\ -1 \end{pmatrix} = Q_1(Rx - z) - \rho q_{n+1}.$$

If q_{n+1} is orthogonal to Q_1, then the solution of $\min_x \|Ax - b\|_2$ is obtained from

$$Rx = z, \qquad r = \rho q_{n+1}.$$

(Note that it is not necessary to assume that Q_1 is orthogonal for this conclusion to hold.) The resulting algorithm can be written as follows:

ALGORITHM 5.1. (Linear Least Squares Solution by MGS) Carry out MGS on $A \in R^{m \times n}$ to give $Q_1 = (q_1, \ldots, q_n)$ and R, and put $b^{(1)} = b$. Then the least squares solution x is computed by:

for $k = 1, 2, \ldots, n$
  $\delta_k = q_k^T b^{(k)}; \quad b^{(k+1)} = b^{(k)} - \delta_k q_k;$
end
$Rx = (\delta_1, \ldots, \delta_n)^T;$

In [6] it is shown that Algorithm 5.1 is numerically equivalent to the Householder-Golub method

$$P_n \cdots P_2 P_1 \begin{pmatrix} 0 & 0 \\ A & b \end{pmatrix} = \begin{pmatrix} R & c_1 \\ 0 & c_2 \end{pmatrix}.$$

This follows by taking the special form (7) of the matrices P_k into account. Using this equivalence, it is proved in [6] that *Algorithm 5.1 is a backward stable method for solving the linear least squares problem*, which is a new result. However, from the numerical experiments of Jordan [14] and Wampler [27, 28], it is known that MGS often gives slightly more accurate least squares solutions than other orthogonalization methods. In [21] Wilkinson writes "Evidence is accumulating that the modified Gram-Schmidt method gives better results than Householder.... The reasons for this phenomenon appear not to have been elucidated yet." Note, however, that MGS is somewhat more expensive in terms of operations and storage.

We now show how Algorithm 5.1 can be derived from Gaussian elimination applied to the normal equations, with the computation of inner products deferred as long as possible. Bienaymé [3] gave a constructive derivation of an algorithm of Cauchy for solving more general systems of the form $Z^T A x = Z^T b$, where $Z = (z_1, \ldots, z_n)$, which we illustrate for the case $n = 3$,

$$\begin{pmatrix} z_1^T a_1 & z_1^T a_2 & z_1^T a_3 \\ z_2^T a_1 & z_2^T a_2 & z_2^T a_3 \\ z_3^T a_1 & z_3^T a_2 & z_3^T a_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} z_1^T b_1 \\ z_2^T b_2 \\ z_3^T b_3 \end{pmatrix}.$$

In the first elimination step the i, jth element is transformed

$$z_i^T a_j - \frac{z_i^T a_1}{z_1^T a_1} z_1^T a_j = z_i^T \left(a_j - \frac{z_1^T a_j}{z_1^T a_1} a_1 \right) \equiv z_i^T a_j^{(2)}, \qquad 2 \le i, j \le 3.$$

The reduced system has the form

$$\begin{pmatrix} z_2^T a_2^{(2)} & z_2^T a_3^{(2)} \\ z_3^T a_2^{(2)} & z_3^T a_3^{(2)} \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} z_2^T b_2^{(2)} \\ z_3^T b_3^{(2)} \end{pmatrix},$$

where we have defined

$$a_j^{(2)} = a_j - \frac{z_1^T a_j}{z_1^T a_1} a_1, \qquad b^{(2)} = b - \frac{z_1^T b}{z_1^T b} a_1.$$

Since the vectors z_2, z_3 are not used in the first step, only z_1 need be chosen at this stage. The reduced system is of similar form, and the reduction can be

continued by chosing z_2 and eliminating x_2 by Gaussian elimination to obtain the single equation

$$z_3^T a_3^{(3)} x_3 = z_3^T b^{(3)}.$$

Taking the first equation from each step now gives a triangular system defining the solution.

If $Z = A$, we are solving the normal equations. However, it suffices that $\mathcal{R}(Z) = \mathcal{R}(A)$ for the algorithm to produce a least squares solution. In particular, taking $Z = Q = (q_1, q_2, \ldots, q_n)$, where

$$q_1 = a_1, \quad q_2 = a_2^{(2)}, \ldots, q_n = a_n^{(n)},$$

this condition is satisfied, since span $[q_1, \ldots, q_n]$ = span $[a_1, \ldots, a_n]$. With this choice we have

$$q_2 = a_2 - \frac{q_1^T a_2}{q_1^T q_1} q_1, \quad q_3 = a_3^{(2)} - \frac{q_2^T a_3^{(2)}}{q_2^T q_2} q_2, \ldots,$$

which leads exactly to Algorithm 5.1. As remarked in the introduction, this algorithm was derived originally by Laplace [15].

Using the numerical equivalence of MGS with the special Householder factorization we can also derive a backward stable algorithm for computing the minimum norm solution of an *underdetermined* linear system

$$\min \|y\|_2, \qquad A^T y = c, \tag{13}$$

The algorithm is derived by using the Householder factorization to solve the augmented problem

$$\min \left\| \begin{pmatrix} w \\ y \end{pmatrix} \right\|_2, \qquad (0 \;\; A^T) \begin{pmatrix} w \\ y \end{pmatrix} = c.$$

If we solve $R^T z = c$ for z, the solution y ($w = 0$) is obtained from

$$\begin{pmatrix} w \\ y \end{pmatrix} = P_1 P_2 \cdots P_n \begin{pmatrix} z \\ 0 \end{pmatrix}.$$

Again, on simplification using the special form of the matrices P_k, this gives the following backward-stable MGS algorithm for the problem (13) (see [6]):

ALGORITHM 5.2. (Minimum norm solution by MGS). Carry out MGS on $A^T \in R^{m\times n}$ to give $Q_1 = (q_1, \ldots, q_n)$ and R. Then the minimum norm solution $y = y^{(0)}$ is obtained from:

$$
\begin{aligned}
&R^T(\zeta_1, \ldots, \zeta_n)^T = c; \quad y^{(n)} = 0;\\
&\text{for } k = n, \ldots, 2, 1\\
&\quad \omega_k = q_k^T y^{(k)}; \quad y^{(k-1)} = y^{(k)} - (\omega_k - \zeta_k)q_k;\\
&\text{end}
\end{aligned}
$$

If the columns of Q_1 were orthogonal, then $\omega_n = \ldots = \omega_1 = 0$. Otherwise ω compensates for the lack of orthogonality. Hence this new algorithm is the correct dual to Algorithm 5.1. It does not seem likely that this algorithm could have been discovered without using the interpretation of MGS as a Householder method.

6. ITERATED GRAM-SCHMIDT ORTHOGONALIZATION

In the *orthogonal basis problem* one wants to compute Q_1 and R such that $A = Q_1R$ and Q_1 is accurately orthogonal. Perhaps the most important application is in updating a QR factorization when rows/columns are added or deleted; see Daniel et al. [8]. To solve the orthogonal basis problem we must use Gram-Schmidt with reorthogonalization. It can be shown, in a sense made more precise below, that one reorthogonalization always suffices. This will at most double the cost of the Gram-Schmidt method.

We first consider the case $n = 2$ and describe the Kahan-Parlett algorithm; see Parlett [20, pp. 105–110]. This algorithm is based on unpublished notes of Kahan on the fact that "twice is enough". Given vectors q_1 ($\|q_1\|_2 = 1$) and a_2, we want to compute $\hat{q}_2 = a_2 - r_{12}q_1$, $r_{12} = q_1^T a_2$. Assume that we can perform this computation with an error $\|\bar{q}_2 - \hat{q}_2\|_2 \le \epsilon\|a_2\|_2$ for some small positive ϵ independent of q_1 and a_2. (As remarked in Section 2, this holds with $\epsilon = cu$ for standard floating point arithmetic.) Let α be a fixed value chosen in the range $[1.2\epsilon, 0.83 - \epsilon]$. Then a vector $\check{q}_2$ is computed as follows: Take

$$\bar{r}_{12} = \mathrm{fl}(q_1^T a_2), \quad \bar{q}_2 = \mathrm{fl}(\bar{a}_2 - \bar{r}_{12}q_1),$$

where fl denotes floating point computation. If $\|\bar{q}_2\|_2 \ge \alpha\|a_2\|_2$, then put $\check{q}_2 = \bar{q}_2$, else reorthogonalize $\bar{q}_2$;

$$\delta r_{12} := \mathrm{fl}(q_1^T \bar{q}_2), \quad \tilde{q}_2 := \mathrm{fl}(\bar{q}_2 - \delta r_{12}q_1).$$

If $\|\tilde{q}_2\|_2 \geq \alpha\|\tilde{q}_2\|_2$, then accept $\check{q}_2 := \tilde{q}_2$, else accept $\check{q}_2 := 0$. The computed $\check{q}_2$ satisfies

$$\|\check{q}_2 - \hat{q}_2\|_2 \leq (1+\alpha)\epsilon\|a_2\|_2, \quad \|q_1^T\check{q}_2\| \leq \epsilon\alpha^{-1}\|\check{q}_2\|_2. \tag{14}$$

When α is large, say 0.5, then the bounds (14) are very good but reorthogonalization will occur more frequently. If α is small, reorthogonalization will be rarer, but the bound on orthogonality less good. Rutishauser [24] used $\alpha = 0.1$, but there seems to be a good case for recommending the more stringent value $\alpha = 0.5$; see below.

It should be noted that in the algorithm above we may end up with a zero column, if the matrix is sufficiently close to being rank-deficient. This is not good enough for some applications, where a full set of orthogonal columns is needed also when A is numerically rank-deficient. In this case two orthogonalizations are not enough. The analysis of this situation is surprisingly intricate (see [8]), and we will not include a discussion of it here.

We now consider the more general case when we are given the matrix $Q_1 = (q_1, \ldots, q_{k-1})$ with $\|q_1\|_2 = \ldots = \|q_{k-1}\|_2 = 1$, together with the vector a_k, and want to compute a vector $\hat{q}_k \in \text{span}\,(Q_1, a_k) \perp Q_1$. The solution equals $\hat{q}_k = a_k - Q_1 r_k$, where r_k solves the least squares problem

$$\min_{r_k} \|a_k - Q_1 r_k\|_2. \tag{15}$$

For solving this problem when the columns of Q_1 need not be accurately orthogonal we consider *iterated* Gram-Schmidt methods, where the CGS or MGS algorithm is repeatedly applied as follows.

ALGORITHM 6.1. (Iterated CGS Algorithm). Given $Q_1 \in R^{m\times(k-1)}$ with $\text{diag}\,(Q_1^TQ_1) = I$, and a vector $a_k \notin \mathcal{R}(Q_1)$, the following algorithm computes r_k and $\hat{q}_k = a_k - Q_1 r_k \perp \mathcal{R}(Q_1)$:

$\hat{q}_k^0 := a_k; \quad r_k^0 := 0;$
for $p = 0, 1, \ldots,$ until $\hat{q}_k^p \perp \mathcal{R}(Q_1)$
  $s_k^p := Q_1^T\hat{q}_k^p; \quad \hat{q}_k^{(p+1)} := \hat{q}_k^p - Q_1 s_k^p;$
  $r_k^{(p+1)} := r_k^p + s_k^p;$
end

The first step of this algorithm is the usual CGS algorithm, and each step is a reorthogonalization. The iterated MGS algorithm is similar, except that each projection is subtracted as soon as it computed:

$$
\begin{aligned}
&\hat{q}_k := \hat{q}_k^{(p)};\\
&\text{for } i = 1, \ldots, k-1\\
&\quad s_{ik}^p := q_i^T \hat{q}_k; \quad \hat{q}_k := \hat{q}_k - s_{ik}^p q_i;\\
&\quad r_{ik}^{(p+1)} := r_{ik}^p + s_{ik}^p;\\
&\text{end}\\
&\hat{q}_k^{(p+1)} := \hat{q}_k;
\end{aligned}
$$

Iterated Gram-Schmidt algorithms have been considered and analyzed by Daniel et al. [8], Ruhe [23], and Hoffmann [12] . A rounding error analysis for the CGS algorithm with reorthogonalization has also been given by Abdelmalek [1].

Ruhe [23] shows that the iterated CGS and MGS algorithms correspond to the Jacobi and Gauss-Seidel iterative methods for the system of normal equations

$$Q_1^T Q_1 r_k = Q_1^T a_k \tag{16}$$

for (15). Hence with the splitting $Q_1^T Q_1 = I + L + L^T$, the rate of convergence for iterated CGS and MGS is related to the spectral radius of $L + L^T$ and $(I + L)^{-1} L^T$ respectively. Ruhe also points out that in case Q_1 is far from orthogonal, fewer iterations may be needed if the system (16) is instead solved by the conjugate gradient method.

Hoffmann [12] has given a very detailed error analysis for iterated CGS and MGS. He shows that, as in the Kahan-Parlett algorithm, the stopping criterion $\hat{q}_k^p \perp \mathcal{R}(Q_1)$ in iterated CGS can be chosen as $\|\hat{q}_k^{(p+1)}\|_2 > \alpha \|\hat{q}_k^p\|_2$. Hoffmann considers in particular the case when iterated Gram-Schmidt is used recursively, adding one column a_k at a time, to compute the factorization $A = Q_1 R$. He concludes that if A has full column rank, then with the choice $\alpha = 0.5$ both iterated CGS and MGS give a factor Q_1 which is orthogonal to almost full working precision, *using at most one reorthogonalization*. Hence in this case iterated CGS is *not* inferior to the iterated MGS.

If less than full precision orthogonality is wanted, values of $\alpha < 0.5$ may be used. For iterated MGS orthogonality of Q_1 will then be bounded roughly by $\alpha^{-1}\sqrt{n}u$. No such result holds for iterated CGS. Note that if α is chosen small enough, then no reorthogonalization will occur and we have the bound (4) for MGS. For more details we refer to [12]. We finally note that the solution of the orthogonal basis problem with Householder's method requires $2(mn^2 - n^3/3)$ flops. Hence if the average number of reorthogonalizations needed is ν, then the Gram-Schmidt method requires less operations when $\nu < 2 - 2n/(3m)$. More important is that for updating the QR factorization the Householder method requires storage of a full $Q \in \mathbf{R}^{m \times m}$, whereas iterated CGS or MGS only needs to store $Q_1 \in \mathbf{R}^{m \times n}$.

7. BLOCK ORTHOGONALIZATION METHODS

Many current computing architectures require code that is dominated by matrix-matrix multiplication in order to attain near-peak performance. This explains the current interest in developing block algorithms, such as the block Householder QR by Schreiber and Van Loan [26]. We now show that a block version of the Gram-Schmidt algorithm can be developed using similar ideas.

Assume that the matrix A is partitioned into blocks of columns

$$A = (A_1, A_2, \ldots, A_N), \qquad A_j \in \mathbf{R}^{m\times p},$$

where $n = Np$, and $p \ll n$. The block QR algorithm proceeds in N steps, $k = 1, \ldots, N$. For $k = 1$ we first compute by MGS the factorization

$$A_1 = Q_1 R_{11}, \qquad Q_1 = (q_1, \ldots, q_p),$$

and then update the remaining $N - 1$ block columns of A through premultiplication by P_1:

$$A_j^{(2)} = P_1 A_j, \qquad j = 2, \ldots, N, \qquad P_1 = (I - q_p q_p^T) \cdots (I - q_1 q_1^T).$$

We want to perform this updating as a sequence of matrix multiplications, and therefore express P_1 in the form

$$P_1 = I - Q_1 L_1 Q_1^T, \tag{17}$$

where $L_1 \in \mathbf{R}^{p\times p}$ is lower triangular. Here Q_1 and L_1 can be recursively generated as follows: Let $Q_1 := q_1$, $L_1 := 1$, and for $i = 2, \ldots, p$, compute

$$l_i := -L^T(Q_1^T q_k), \qquad L_1 := \begin{pmatrix} L_1 & 0 \\ l_i^T & 1 \end{pmatrix}, \qquad Q_1 := (Q_1\ q_i).$$

This requires about $\frac{1}{2}p^2(m + \frac{1}{3}p)$ flops.

We can now express the update in (17) as two matrix-matrix multiplications and one rank p update:

$$A_j^{(2)} = P_1 A_j = (I - Q_1 L_1 Q_1^T) A_j = A_j - Q_1[L_1(Q_1^T A_j)]. \tag{18}$$

This requires $2(n - p)(mr + \frac{1}{4}p^2)$ flops, and constitutes the main work in the first step.

In the next step, $k = 2$, we compute the QR factorization of $A_2^{(2)}$, Q_2, and L_2, and premultiply the rest of the block columns by $P_2 = I - Q_2 L_2 Q_2^T$. All

the remaining steps, $k = 3, \ldots, N$, are similar, and the operation count for the complete QR factorization becomes:

(1) Compute the factorizations $A_k^{(k)} = Q_k R_{kk}$: mnp flops.
(2) Generate the matrices L_k: $\frac{1}{2}np(m + \frac{1}{3}p)$ flops.
(3) Apply projections: $P_k A_j^{(k)}$: $(n^2 - np)(m + \frac{1}{4}p)$ flops.

In practice, typically $p = 16$ or $p = 32$ and $p < 0.1n$. Then, as for the corresponding Householder version, the overhead $3mnp/2 + n^2p/4$ flops in steps 1 and 2 is small compared to the dominating term n^2m flops in step 3.

An even simpler version of the block Gram-Schmidt algorithm can be developed by noting that if the matrix Q_k in the factorization $A_k = Q_k R_{kk}$ is orthogonal, then $L_k = I$ and $P_k = I - Q_k Q_k^T$. This can be achieved by using iterated Gram-Schmidt with $\alpha = 0.5$, as described in Section 5, to compute the QR factorization of the column blocks. This will at most increase the operation count in step 1 with mnp flops. On the other hand, we save about $\frac{1}{2}mnp$ flops in step 2, and $n^2p/4$ flops in step 3. Also, the storage space $np/2$ for the matrices L_k is saved. As remarked in Section 5, with the choice $\alpha = 0.5$, CGS is as good as MGS. Since in classical Gram-Schmidt the factorization of A_k can be performed by matrix-vector multiplication, CGS should be preferred to MGS. This block method, which has recently been analyzed by Jalby and Philippe [13] and Malard [19], has no corresponding Householder version.

The traditional column pivoting strategy cannot be used with the block algorithm, since it requires the update of all remaining columns as one column is processed. We note that Bischof [4] has suggested a local pivoting strategy based on an incremental condition estimator. Columns which are found to be nearly linearly dependent on the space spanned by previously chosen columns are permuted to the end of the matrix.

The author is grateful for several comments from an anonymous referee which greatly improved the presentation.

REFERENCES

1 N. N. Abdelmalek, Roundoff error analysis for Gram-Schmidt method and solution of linear least squares problems, *BIT* 11:345–368 (1971).
2 F. L. Bauer, Elimination with weighted row combinations for solving linear equations and least squares problems, *Numer. Math.* 7:338–352 (1965).
3 I. J. Bienaymé, Remarques sur les différences qui distinguent l'interpolation de M. Cauchy de la méthode des moindre carrés, et qui assurent la supériorité de cette méthode, *C. R. Acad. Sci. Paris* 37:5–13 (1853).

4 C. H. Bischof, A block *QR* factorization algorithm using restricted pivoting, in *Supercomputing 89*, ACM Press, 1989, pp. 248–256.
5 Å. Björck, Solving linear least squares problems by Gram-Schmidt orthogonalization, *BIT* 7:1–21 (1967).
6 Å. Björck and C. C. Paige, Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm, *SIAM J. Matrix Anal. Appl.* 13:176–190 (1992).
7 A. L. Cauchy, Mémoire sur L'Interpolation, Lithograph, Prague, 1835, and *J. Math. Pures Appl.* 2:193–205 (1837); reprinted in *Oeuvres*, Ser. 2, Vol. 2, Gauthier-Villars, Paris, 1958, pp. 5–17.
8 J. Daniel, W.B. Gragg, L. Kaufman, and G. W. Stewart, Reorthogonalization and stable algorithms for updating the Gram-Schmidt *QR* factorization, *Math. Comp.* 30:772–795 (1976).
9 R.W. Farebrother, *Linear Least Squares Computations*, Marcel Dekker, New York, 1988.
10 W. Gander, Algorithms for the *QR*-decomposition, Technical Report 80–02, Angewandte Mathematik, ETH, 1980.
11 G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed. Johns Hopkins U.P., Baltimore, 1989.
12 W. Hoffman, Iterative algorithms for Gram-Schmidt orthogonalization, *Computing* 41:335–348 1989.
13 W. Jalby and B. Philippe, Stability analysis and improvement of the block Gram-Schmidt algorithm, *SIAM J. Sci. Statist. Comput.* 12:1058–1073 (1991).
14 T. L. Jordan, Experiments on error growth associated with some linear least-squares procedures, *Math. Comp.* 22:579–588 (1968).
15 P. S. Laplace, *Théorie Analytique des Probabilités. Premier Supplément*, Mme. Courcier, Paris, 1816.
16 C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, N.J., 1974.
17 P. Läuchli, Jordan-Elimination und Ausgleichung nach kleinsten Quadraten, *Numer. Math.* 3:226–240 (1961).
18 J. W. Longley, Modified Gram-Schmidt process vs. classical Gram-Schmidt, *Comm. Statist. Simulation Comput. B10* 5:517–527 (1981).
19 J. Malard, Block Solvers for Dense Linear Systems on Local Memory Multiprocessors, Ph.D. Thesis, School of Computer Science, McGill Univ., Montréal, Aug. 1992.
20 B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, N.J., 1980.
21 G. Peters and J. H. Wilkinson, The least squares problem and pseudo-inverses, *Comput. J.* 13:309–316 (1970).
22 J. R. Rice, Experiments on Gram-Schmidt orthogonalization, *Math. Comp.* 20:325–328 (1966).
23 A. Ruhe, Numerical aspects of Gram-Schmidt orthogonalization of vectors, *Linear Algebra Appl.* 52/53:591–601 (1983).
24 H. Rutishauser, *Description of Algol 60*, Handbook Automat. Comput. 1a, Springer-Verlag, Berlin, 1967.

25 E. Schmidt, Über die Auflösung linearer Gleichungen mit unendlich vielen Unbekannten, *Rend. Circ. Mat. Palermo (1)* 25:53–77 (1908).
26 R. Schreiber and C. F. Van Loan, A storage efficient *WY* representation for products of Householder transformations, *SIAM J. Sci. Statist. Comput.* 10:53–57 (1989).
27 R. H. Wampler, A report on the accuracy of some widely used least squares computer programs, *J. Amer. Statist. Assoc.* 65:549–565 (1970).
28 R. H. Wampler, Solutions to weighted least squares problems by modified Gram–Schmidt with iterative refinement, *ACM Trans. Math. Software* 5:457–465 (1979).
29 J. H. Wilkinson, Error analysis of transformations based on the use of matrices of the form $I - 2ww^{H}$, in *Error in Digital Computation* L. B. Rall, Ed. Wiley, New York, 1965, pp. 77–101.

Received 13 December 1992; final manuscript accepted 20 April 1992