SIAM J. SCI. STAT. COMPUT.
Vol. 9, No. 1, January 1988

010

IMPLEMENTATION OF THE GMRES METHOD USING HOUSEHOLDER TRANSFORMATIONS*

HOMER F. WALKER†

Abstract. The standard implementation of the GMRES method for solving large nonsymmetric linear systems involves a Gram-Schmidt process which is a potential source of significant numerical error. An alternative implementation is outlined here in which orthogonalization by Householder transformations replaces the Gram-Schmidt process. This implementation requires slightly less storage but somewhat more arithmetic than the standard one; however, numerical experiments suggest that it is more stable, especially as the limits of residual reduction are reached. The extra arithmetic required may be less significant when products of the coefficient matrix with vectors are expensive or on vector and, in particular, parallel machines.

Key words. GMRES method, iterative methods, matrix-free methods, nonsymmetric linear systems, Householder transformations.

AMS(MOS) subject classifications. 65F10, 65N20

1. Introduction. Of interest here is the generalized minimal residual (GMRES) method of Saad and Schultz [8]. This is an iterative method for solving large linear systems of equations

$$Ax = b \tag{1.1}$$

in which $A \in \mathbf{R}^{n\times n}$ is nonsymmetric. For a full description of this method and its standard implementation, see [8]. For examples of its successful application to problems arising from the numerical solution of ordinary and partial differential equations, see Brown and Hindmarsh [2] and Wigton, Yu and Young [10].

In brief, the GMRES method begins with an initial approximate solution x_0 and initial residual $r_0 = b - Ax_0$; at the mth iteration, a correction z_m is determined in the *Krylov subspace*

$$K_m(v) \equiv \text{span}\{v, Av, \ldots, A^{m-1}v\}$$

with $v = r_0$ which solves the least-squares problem

$$\min_{z\in K_m(r_0)} \|b - A(x_0 + z)\|_2. \tag{1.2}$$

The mth iterate is then $x_m = x_0 + z_m$. The correction z_m is chosen to reduce the residual norm as much as possible among the space of allowable corrections, and it is clear that the residual norm is nonincreasing from one iteration to the next. Furthermore, one can show that if exact arithmetic were used, then the solution would be reached in no more than n iterations [8, Corollary 3, p. 865].

* Received by the editors December 22, 1986; accepted for publication (in revised form) May 3, 1987. This research was supported in part by the United States Department of Energy under contract DE-FG02-86ER25018 with Utah State University and in part by the Computing and Mathematics Research Division, Lawrence Livermore National Laboratory, Livermore, California 94550.

† Department of Mathematics, Utah State University, Logan, Utah 84322-3900.

A natural approach to determining z_m is to determine a basis of $K_m(r_0)$ and then to solve an m-dimensional least-squares problem for the coefficients of the linear combination of basis elements that solves (1.2). Since the method is likely to proceed for a number of iterations, a basis should be determined for each m in a way which allows economizing on the arithmetic incurred in incrementing m, e. g., by allowing updating of the matrix factorizations used to solve the least-squares problems for increasing values of m. An obvious choice of a basis for each m is $\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$, and the solution of the least-squares problems for these bases can be carried out economically as m is incremented (see Walker [9]). However, in many applications these least-squares problems are likely to be very ill-conditioned even for fairly small values of m, with a resulting degradation of the performance of the method. The GMRES implementation of [9], which uses these bases in conjunction with very stable Householder transformations, was observed by Hindmarsh and Walker [5] to perform significantly worse in stiff ODE solving experiments than the standard implementation of [8]. We also found this implementation to perform far worse on the two numerical problems outlined below than either the standard implementation or the implementation using Householder transformations given in the sequel, although details of the performance of this implementation are not shown here. Consequently, the use of the bases $\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$ is not recommended.

The implementation of GMRES given by Saad and Schultz [8] determines z_m by maintaining for each m a particular orthonormal basis $\{v_1, \ldots, v_m\}$ of $K_m(r_0)$ together with an upper-Hessenberg $(m+1) \times m$ matrix H_m. These are generated through *Arnoldi's method*, the basic form of which is the following:

ARNOLDI'S METHOD. Suppose v_1 is given with $\|v_1\|_2 = 1$.

For $m = 1, 2, \ldots,$ do:

a. Set
$h_{im} = (Av_m)^T v_i$, $i = 1, 2, \ldots, m$,
$\hat{v}_{m+1} = Av_m - \sum_{i=1}^{m} h_{im} v_i$,
$h_{m+1,m} = \|\hat{v}_{m+1}\|_2$,

b. If $h_{m+1,m} = 0$, then stop; otherwise, set
$v_{m+1} = \hat{v}_{m+1} / h_{m+1,m}$.

Clearly $\{v_1, \ldots, v_m\}$ generated by Arnoldi's method is an orthonormal basis of $K_m(v_1)$ for each m. In the GMRES implementation of [8], $v_1 = r_0/\|r_0\|_2$ and the h_{ij}'s constitute the (possibly) nonzero elements of the matrices H_m. Note that

$$A(v_1, \ldots, v_m) = (v_1, \ldots, v_{m+1})H_m$$

for each m.

With these orthonormal bases and upper-Hessenberg matrices, z_m is determined as follows: Setting $z = (v_1, \ldots, v_m)y$ for $y \in \mathbf{R}^m$ gives

$$\begin{aligned} \min_{z \in K_m(r_0)} \|b - A(x_0 + z)\|_2 &= \min_{y \in \mathbf{R}^m} \|r_0 - A(v_1, \ldots, v_m)y\|_2 \\ &= \min_{y \in \mathbf{R}^m} \|r_0 - (v_1, \ldots, v_{m+1})H_m y\|_2. \end{aligned} \tag{1.3}$$

Since v_1 is $r_0/\|r_0\|_2$ and since $\{v_1, \ldots, v_{m+1}\}$ is orthonormal, one has

$$\min_{y \in \mathbf{R}^m} \|r_0 - (v_1, \ldots, v_{m+1})H_m y\|_2 = \min_{y \in \mathbf{R}^m} \left\| \|r_0\|_2 e_1 - H_m y \right\|_2, \tag{1.4}$$

where e_1 is the first canonical basis vector in $\mathbf{R}^{m+1}$. It follows from (1.3) and (1.4) that $z_m = (v_1, \ldots, v_m)y_m$, where y_m solves

$$\min_{y \in \mathbf{R}^m} \left\| \|r_0\|_2 e_1 - H_m y \right\|_2. \tag{1.5}$$

In fact, this characterization of z_m is valid even when $h_{m+1,m} = 0$, which occurs if and only if the solution y_m of (1.5) yields zero residual. Thus the GMRES method is forced to stop at the mth iteration if and only if x_m is the solution of (1.1).

Since H_m is upper-Hessenberg, (1.5) is easily solved. In actual practice, H_m is maintained in factored form, and the factors are updated with each increment of m. Typically, H_m is factored as $H_m = Q_m R_m$, where Q_m is a product of Givens rotations and R_m is upper-triangular. In this case (1.5) is equivalent to

$$\min_{y \in \mathbf{R}^m} \left\| \|r_0\|_2 Q_m^T e_1 - R_m y \right\|_2,$$

and y_m is obtained by solving an upper-triangular system. Note that the absolute value of the last coordinate of $\|r_0\|_2 Q_m^T e_1$ is just $\|b - Ax_m\|_2$, and so the residual at the mth iteration can be determined without actually having to compute the correction.

At the heart of Arnoldi's method is a Gram-Schmidt process, and so we refer to the GMRES implementation of [8] as the Gram-Schmidt implementation. The basic form of Arnoldi's method given above employs the classical Gram-Schmidt process, which is numerically untrustworthy. Because of roundoff, there may be severe loss of orthogonality among the computed v_m's. In practice, it is usual to implement Arnoldi's method using the modified Gram-Schmidt process (see Golub and Van Loan [4]). Mathematically, this is just a rearrangement of the classical process; computationally, it has superior properties.

Even the modified Gram-Schmidt process can fail to perform well if the vectors on which it operates are not sufficiently independent. Indeed, if $S = (s_1, \ldots, s_m)$ is an $n \times m$ matrix the columns of which are to be orthonormalized and if $Q = (q_1, \ldots, q_m)$ is the computed result of applying modified Gram-Schmidt to the columns of S using floating point arithmetic with unit rounding error $\mathbf{u}$, then Bjorck [1] has shown that

$$Q^T Q = I + E, \qquad \|E\|_2 \approx \mathbf{u}\kappa_2(S), \tag{1.6}$$

where the condition number $\kappa_2(S)$ is the ratio of the largest singular value of S to the smallest. It follows that at the mth step of Arnoldi's method using modified Gram-Schmidt, v_{m+1} may have a significantly nonzero component in the span of $\{v_1, \ldots, v_m\}$ if $\kappa_2\big((v_1, \ldots, v_m, Av_m)\big)$ is large, i. e., if Av_m is nearly in the span of $\{v_1, \ldots, v_m\}$. Saad [7, p. 214] has suggested that the Gram-Schmidt process in Arnoldi's method may be an important source of errors in the full and incomplete orthogonalization methods ([6],[7]), which are related to GMRES.

There is an alternative orthogonalization procedure based on the use of *Householder transformations* which is reliable even if the vectors to be orthonormalized are not very independent. A Householder transformation is of the form $P = I - 2uu^T$, where I is the identity matrix and $\|u\|_2 = 1$. We refer to u as the *Householder vector* which determines P. Note that $P^{-1} = P^T = P$. Also, note that the action of P on a vector or matrix can be easily determined using u; in particular, one need not explicitly form or store P in the applications of interest here. If two vectors of the same norm are given, then it is easy to determine P so that it takes one vector into the other;

furthermore, if the first k components of the two vectors are the same, then the u which determines P can be chosen so that its first k components are zero. For more on the properties and uses of Householder transformations, see Golub and Van Loan [4].

To orthonormalize the columns of $S = (s_1, \ldots, s_m)$, one determines Householder transformations $P_1, \ldots, P_m$ such that $P_m \ldots P_1 S = R$, an upper-triangular matrix. It is natural to determine $P_1, \ldots, P_m$ inductively by the requirement that $P_k \ldots P_1(s_1, \ldots, s_k)$ be upper-triangular for $k = 1, \ldots, m$; if $P_m \ldots P_1 S = R$, then this requirement is met if and only if for $k = 2, \ldots, m$ the first $k-1$ components of the Householder vector determining P_k are zero. Since $S = P_1 \ldots P_m R$, the matrix Q consisting of the first m columns of $P_1 \ldots P_m$ gives the desired orthonormalization of the columns of S. If Q is computed in floating point arithmetic with unit rounding error $\mathbf{u}$, then (Bjorck [1])

$$Q^T Q = I + E, \qquad \|E\|_2 \approx \mathbf{u}. \tag{1.7}$$

In view of the greater reliability of orthogonalization based on Householder transformations, as reflected in (1.6) and (1.7), it seems worthwhile to consider implementations of the GMRES method in which this orthogonalization replaces the Gram-Schmidt process in the implementation of [8]. We outline one such Householder implementation here. It is based on a particular method of using Householder transformations to generate orthonormal bases of Krylov subspaces. This method is essentially that used by Golub, Underwood and Wilkinson [3] in their implementation of the Lanczos algorithm; see also Golub and Van Loan [4, pp. 334-335]. An analogous implementation can be formulated without difficulty for the full orthogonalization method of Saad [6], but we do not give this here.

Our experiments indicate that the Householder implementation given here has better numerical properties than the Gram-Schmidt implementation, especially in the final iterations when the limits of residual reduction are neared. This Householder implementation uses slightly less storage than the Gram-Schmidt implementation; however, it requires additional arithmetic. The increase in arithmetic is always less than a factor of three, and we mention several important circumstances in which it may not be an overriding concern. We also give a variation of the Householder implementation of GMRES which may be useful in some circumstances on parallel computers. This variation is obtained by using simple formulas to express products of Householder transformations acting on vectors as operations involving matrix-vector products. In our experiments, this variation performed well while analogous variations of the Gram-Schmidt implementation, which use the classical Gram-Schmidt process, did not perform as effectively.

Notational conventions are as follows: Matrices are denoted by capital letters whereas vectors and scalars are denoted by lower case letters. Vector components and matrix entries are indicated by superscripts in parentheses, e. g., $v^{(i)}$ denotes the ith component of the vector v. With or without subscripts or other distinguishing marks, the letters H, J, L, P, and R always indicate matrices of the following respective types: upper-Hessenberg, Givens rotation, lower-triangular, Householder transformation, and upper-triangular. (See Golub and Van Loan [4] for definitions and properties.) The ith canonical basis vector, i. e., the ith column of the identity matrix I, is denoted by e_i. Dimensions of vectors and matrices and, when appropriate, their (possibly) nonzero elements are implicit from the contexts in which they appear. For example, if R is $p \times q$ upper-triangular and we write $R = (ce_1, H, h)$,

then H must be $p \times (q-2)$ upper-Hessenberg and h must be a p-vector with zero components after the qth.

2. The Householder implementation. We first formulate an algorithm which uses Householder transformations to generate orthonormal bases of Krylov subspaces. Similar use of Householder transformations is made in the references [3], [4] cited in the introduction.

ALGORITHM 2.1. Suppose v_1 is given with $\|v_1\|_2 = 1$.

1. Choose P_1 such that $P_1 v_1 = e_1$.
2. For $m = 1, 2, \ldots$, do:
   a. Set $v_m = P_1 \ldots P_m e_m$.
   b. If $(v_1, Av_1, \ldots, Av_m)$ has rank m, then stop;
   otherwise, choose P_{m+1} such that
   $P_{m+1} \ldots P_1 (v_1, Av_1, \ldots, Av_m)$ is upper-triangular.

LEMMA 2.1. *A set $\{v_1, \ldots, v_m\}$ generated by Algorithm 2.1 is an orthonormal basis of $\mathcal{K}_m(v_1)$.*

Proof. The lemma is trivially true if $m = 1$, so suppose $m > 1$. The facts that $P_1 v_1 = e_1$ and $P_k \ldots P_1 (v_1, Av_1, \ldots, Av_{k-1})$ is upper-triangular for $k = 2, \ldots, m$ imply that for $k = 2, \ldots, m$ the first $k-1$ components of the Householder vector determining P_k are zero. It follows that for $1 \leq k \leq m$, $v_j = P_1 \ldots P_k e_j$ for $j = 1, \ldots, k$, i.e., that

$$P_1 \ldots P_k = (v_1, \ldots, v_k, \ldots), \quad 1 \leq k \leq m. \tag{2.1}$$

In particular, $P_1 \ldots P_m = (v_1, \ldots, v_m, \ldots)$, and so $\{v_1, \ldots, v_m\}$ is an orthonormal set.

It remains to show that $\mathcal{K}_m(v_1) = \operatorname{span}\{v_1, \ldots, v_m\}$. Certainly $\mathcal{K}_1(v_1) = \operatorname{span}\{v_1\}$. Suppose that

$$\mathcal{K}_k(v_1) \equiv \operatorname{span}\{v_1, Av_1, \ldots, A^{k-1}v_1\} = \operatorname{span}\{v_1, \ldots, v_k\}$$

for some k such that $1 \leq k < m$. Then

$$\mathcal{K}_{k+1}(v_1) \equiv \operatorname{span}\{v_1, Av_1, \ldots, A^k v_1\} = \operatorname{span}\{v_1, Av_1, \ldots, Av_k\}. \tag{2.2}$$

Since $k < m$, $(v_1, Av_1, \ldots, Av_k)$ has rank $k+1$, and so $P_{k+1} \ldots P_1 (v_1, Av_1, \ldots, Av_k)$ is an upper-triangular matrix which also has rank $k+1$. Consequently, the span of $\{v_1, Av_1, \ldots, Av_k\}$ is the span of the first $k+1$ columns of $P_1 \ldots P_{k+1}$. It follows from (2.2) and (2.1) that $\mathcal{K}_{k+1}(v_1) = \operatorname{span}\{v_1, \ldots, v_{k+1}\}$, and the induction is complete.

To use Algorithm 2.1 in implementing the GMRES method, we take $v_1 = \pm r_0 / \|r_0\|_2$, where for numerical soundness the sign is chosen so that the first component of v_1 is positive. At the mth step of Algorithm 2.1, if $(v_1, Av_1, \ldots, Av_m)$ is of full rank, then we choose P_{m+1} according to Algorithm 2.1 and set

$$P_{m+1} \ldots P_1 (Av_1, \ldots, Av_m) = H_m.$$

Writing $z = (v_1, \ldots, v_m)y$ for $y \in \mathbf{R}^m$, one has

$$\begin{aligned} \min_{z \in \mathcal{K}_m(r_0)} \|b - A(x_0 + z)\|_2 &= \min_{y \in \mathbf{R}^m} \|r_0 - A(v_1, \ldots, v_m)y\|_2 \\ &= \min_{y \in \mathbf{R}^m} \left\| \pm \|r_0\|_2 e_1 - H_m y \right\|_2. \end{aligned}$$

It follows that $z_m = (v_1, \ldots, v_m)y_m$, where y_m solves

$$\min_{y \in \mathbf{R}^m} \left\| \pm \|r_0\|_2 e_1 - H_m y \right\|_2. \tag{2.3}$$

As in the GMRES implementation of [8], the least-squares problem (2.3) is solved by maintaining a factorization $H_m = Q_m R_m$, where Q_m is a product of Givens rotations and R_m is upper-triangular; these factors are updated with each increment of m. Then (2.3) is equivalent to

$$\min_{y \in \mathbf{R}^m} \left\| \pm \|r_0\|_2 Q_m^T e_1 - R_m y \right\|_2,$$

and y_m is found by solving an upper-triangular system. Note that the absolute value of the last coordinate of $\pm\|r_0\|_2 Q_m^T e_1$ is $\|b - Ax_m\|_2$, and as in the Gram-Schmidt implementation the residual can be found without having to compute the correction.

We give a complete outline of our Householder implementation of the GMRES method below. This implementation is of a "restarted" version of the method, in which the iteration proceeds for no more than a preset maximum number of times, which we assume to be less than or equal to n.

ALGORITHM 2.2. Suppose an initial approximate solution x_0, a tolerance TOL, and an iteration limit $MAXIT$ are given.

1. Compute $r_0 = b - Ax_0$, and determine P_1 such that $P_1 r_0 = \pm\|r_0\|_2 e_1 \equiv w$.
2. For $m = 1,2, \ldots, MAXIT$, do:
   a. Evaluate $v \equiv P_m \ldots P_1 A P_1 \ldots P_m e_m$.
   b. If $v^{(m+1)} = \cdots = v^{(n)} = 0$, then proceed to step (e); otherwise, continue.
   c. Determine P_{m+1} with a Householder vector having first m components zero such that $P_{m+1}v$ has zero components after the $(m+1)$st.
   d. Overwrite $v \leftarrow P_{m+1}v$.
   e. If $m > 1$, overwrite $v \leftarrow J_{m-1} \ldots J_1 v$.
   f. If $v^{(m+1)} = 0$, proceed to step (i); otherwise, continue.
   g. Determine J_m acting on components m and $m+1$ such that $(J_m v)^{(m+1)} = 0$.
   h. Overwrite $v \leftarrow J_m v$ and $w \leftarrow J_m w$.
   i. Set

$$R_m = \begin{cases} (v), & \text{if } m = 1; \\ (R_{m-1}, v), & \text{if } m > 1. \end{cases}$$

   j. If $|w^{(m+1)}| \leq TOL$ or $m = MAXIT$, then solve for y_m and overwrite x_0 with x_m; otherwise, increment m.
3. Solve for y_m and overwrite x_0 with x_m.
   a. Determine y_m which minimizes $\|w - R_m y\|_2$ by solving an $m \times m$ upper-triangular system with the first m rows of R_m as the coefficient matrix and the first m components of w as the right-hand side.
   b. For $k = 1, \ldots, m$, do:
      Overwrite $x_0 \leftarrow x_0 + y_m^{(k)} P_1 \ldots P_k e_k$.
   c. If $|w^{(m+1)}| \leq TOL$, accept x_0 as the solution; otherwise, return to 1.

In Algorithm 2.2, the Householder vectors determining $P_1, \ldots, P_m$ are stored instead of the orthonormal basis vectors $v_1, \ldots, v_m$. Each basis vector $v_m = P_1 \ldots P_m e_m$ is generated when needed as part of larger computations in steps (2.a) and (3.b). Of

course, the basis vectors generated in step (2.a) can be stored for later use in step (3.b) if this is preferable to regenerating them in step (3.b). The Householder vectors are of decreasing length, and they together with the upper-triangular matrices generated by the algorithm can be stored in an array of dimension $n \times (MAXIT + 1)$. This is somewhat less storage than is required by the Gram-Schmidt implementation, which stores full-length orthonormal basis vectors as well as upper-triangular matrices. Since $MAXIT \ll n$ in most applications, this savings in storage is unlikely to be a major advantage.

Algorithm 2.2 requires more arithmetic than the Gram-Schmidt implementation because of the need to generate basis vectors in steps (2.a) and (3.b). This arithmetic is not necessary in the Gram-Schmidt implementation since orthonormal basis vectors are available from storage. Except for this, the arithmetic needed by the Householder and Gram-Schmidt implementations is about the same, and both require the same number of products of A with vectors. For a more detailed comparison, we count the number of multiplications and products of A with vectors required by the two algorithms, assuming that $MAXIT \ll n$ and ignoring terms less than $O(n)$.

In Algorithm 2.2, step (1) requires about $2n$ multiplications and one A-product. For each m, steps (2.a) and (2.d) require about $4nm$ multiplications and one A-product. If a total of m iterations are performed in step (2), then about $2nm(m+1)$ multiplications and m A-products are required. For each k, step (3.b) requires about $2n(k-1)+n = 2nk-n$ multiplications, and so step (3.b) requires a total of about nm^2 multiplications. Thus Algorithm 2.2 requires about $n(3m^2 + 2m + 2)$ multiplications and $m+1$ A-products to carry out m GMRES iterations and to compute the resulting approximate solution. The Gram-Schmidt implementation requires one A-product to compute the initial residual and about $2n$ multiplications to normalize it. At the mth GMRES iteration, one A-product and about $2n(m+1)$ multiplications are needed to produce the next orthonormal basis vector. If a total of m iterations are performed, then about $nm(m+3)$ multiplications and m A-products are required. Computing the approximate solution after m iterations requires about nm additional multiplications. The Gram-Schmidt implementation then requires about $n(m^2+4m+2)$ multiplications and $m+1$ A-products to carry out m GMRES iterations and compute the resulting approximate solution. There is a variation of the Gram-Schmidt implementation in which successive basis vectors are "reorthogonalized", i. e., subjected again to the Gram-Schmidt process, in the hope of reducing loss of orthogonality due to numerical error. If reorthogonalization is carried out before normalization at every GMRES iteration, then this variation requires about $n(2m^2+5m+2)$ multiplications and $m+1$ A-products to carry out m GMRES iterations and compute the resulting approximate solution.

The cost of the additional arithmetic required by Algorithm 2.2 may be significant; however, it may not be prohibitive if $MAXIT$ is small or if computation is done on vector computers which effectively reduce the expense of applying Householder transformations to vectors. Also, it may not be a very significant part of the overall computational expense in applications in which A-products are relatively costly or in which the GMRES iterations themselves are only a small part of a larger computation. One such application is that given by Brown and Hindmarsh [2] to stiff ODE solving. There, A-products are approximated by forming difference quotients of values of nonlinear functions and the iterative solution of linear systems is itself only a part of the total effort of solving stiff ODE initial-value problems. Finally, our numerical experiments suggest that the variation of the Householder implementation given in the next section may offer an opportunity for effectively exploiting parallelism in

some instances. Analogous variations of the Gram-Schmidt implementation use the classical Gram-Schmidt process and are not as numerically sound.

A numerical experiment. We compared the performance of the Householder and Gram-Schmidt implementations of the GMRES method on a nonsymmetric linear system arising from the discretization of the boundary value problem

$$\begin{aligned} \Delta u + cu + d\frac{\partial u}{\partial x} &= f \qquad \text{in } D, \\ u &= 0 \qquad \text{on } \partial D, \end{aligned}$$

where $D = [0,1] \times [0,1]$ and $c \geq 0$ and d are constants. In our trials, we took $f(x) \equiv 1$ and used a 100×100 mesh of discretization points on D. We used a variety of values of c and d ranging from 1 to 1000, and the maximum number of GMRES iterations allowed before restarting ($MAXIT$ in Algorithm 2.2) ranged from 5 to 50. No preconditioning was used in order to keep the issues of interest clear. Computing was done in single precision on a Digital Equipment Corporation MicroVAX II running Ultrix and using the f77 Fortran compiler.

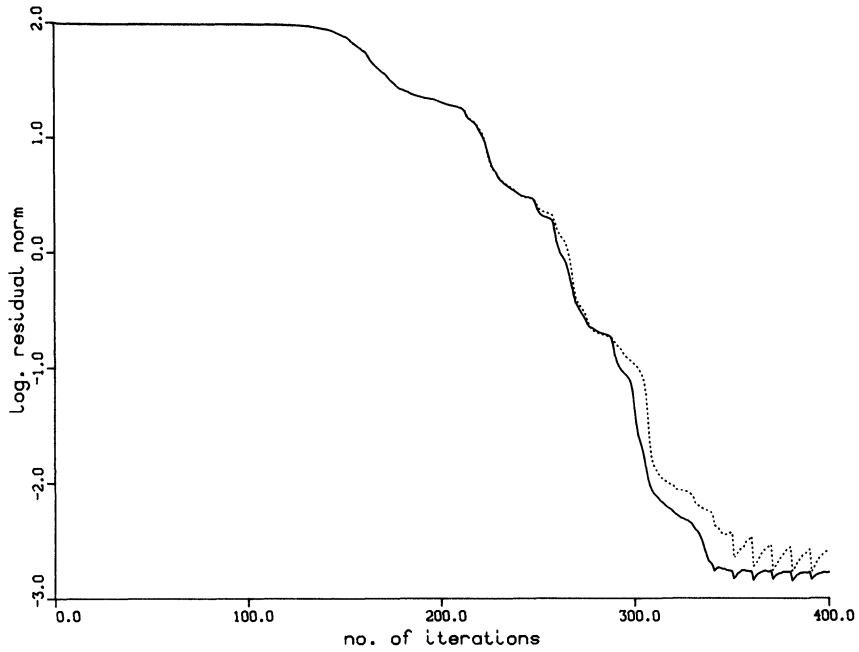

FIG. 1. *The solid line is from the Householder implementation. The dotted line is from the Gram-Schmidt implementation.*

For each trial, we plotted the logarithm of the residual norm versus the number of iterations for the two implementations. It is important to note that in these trials *the residual norms were determined "from scratch,"* i.e., at each iteration, the current approximate solution was computed and multiplied by the coefficient matrix, then the right-hand side vector was subtracted and the norm taken. In particular, the residual norms were *not* taken to be the values maintained by the GMRES implementations. An important aspect of these trials is that they show the unreliability of these values as the limits of residual reduction are neared.

The results shown in Figure 1 are typical. They were obtained with $c = d = 100$ and with the maximum number of iterations allowed before restarting equal to 10. Note

the "sawtooth" pattern of residual-norm levels which appears as the limits of residual reduction are reached. This pattern continues unabated as additional iterations are made. It is exhibited by both implementations, but it is much more pronounced for the Gram-Schmidt implementation. We also tested the Gram-Schmidt implementation with reorthogonalization in this case and found the results to be visually indistinguishable from those produced by the Gram-Schmidt implementation without reorthogonalization.

3. A variation. We now give a variation of Algorithm 2.2 in which the products of Householder transformations acting on vectors are replaced by operations which to a large extent can be performed concurrently. Our hope is that this variation will provide an effective way of exploiting parallelism in circumstances which favor evaluating matrix-vector products in parallel. We count as matrix-vector products such operations as computing the inner products of a vector with the vectors in some set and accumulating a sum of scalar multiples of vectors in some set. The modified Gram-Schmidt process is inherently sequential, and the only apparent variations of the Gram-Schmidt implementation of GMRES which can take advantage of parallelism similarly use the potentially unstable classical Gram-Schmidt process. We have no proof of the numerical superiority of the variation of the Householder implementation given here, but we describe a numerical experiment below in which it performs significantly better than the Gram-Schmidt implementation using the classical Gram-Schmidt process, even when reorthogonalization is done.

The variation of Algorithm 2.2 is based on the following: If $v, w_1, z_1, \ldots, w_m, z_m$ are vectors, then one can write

$$(I + w_m z_m^T) \ldots (I + w_1 z_1^T) v = v + d^{(1)} w_1 + \cdots + d^{(m)} w_m,$$

where $d^{(1)}, \ldots, d^{(m)}$ are determined by

$$\begin{aligned} d^{(1)} &= z_1^T v, \\ d^{(k)} &= z_k^T v + d^{(1)} z_k^T w_1 + \cdots + d^{(k-1)} z_k^T w_{k-1}, \qquad k = 2, \ldots, m. \end{aligned} \tag{3.1}$$

Note that (3.1) is equivalent to specifying $d = (d^{(1)}, \ldots, d^{(m)})^T$ by solving $Ld = (z_1, \ldots, z_m)^T v$ using forward substitution, where

$$L = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ -z_2^T w_1 & 1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ -z_m^T w_1 & \ldots & -z_m^T w_{m-1} & 1 \end{pmatrix}. \tag{3.2}$$

Then one can write

$$(I + w_m z_m^T) \ldots (I + w_1 z_1^T) v = \left\{ I + (w_1, \ldots, w_m) L^{-1} (z_1, \ldots, z_m)^T \right\} v,$$

with L given by (3.2). It follows in particular that if one has Householder transformations $P_k = I - 2u_k u_k^T$, $k = 1, \ldots, m$, and a vector v, then

$$P_m \ldots P_1 v = \left\{ I - 2U_m L_m^{-1} U_m^T \right\} v, \tag{3.3}$$

where

$$U_m = (u_1, \ldots, u_m) \qquad \text{and} \qquad L_m = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 2u_2^T u_1 & 1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 2u_m^T u_1 & \ldots & 2u_m^T u_{m-1} & 1 \end{pmatrix}. \tag{3.4}$$

Similarly,

$$P_1 \dots P_m v = \{I - 2U_m (L_m^T)^{-1} U_m^T\} v, \tag{3.5}$$

where U_m and L_m are given by (3.4). It is understood that the steps in evaluating $P_m \dots P_1 v$ using the right-hand side of (3.3) are the following:

1. Evaluate $U_m^T v$.
2. Solve $L_m d = -2U_m^T v$ by forward substitution.
3. Evaluate

$$v + U_m d = (v, U_m) \begin{pmatrix} 1 \\ d \end{pmatrix}.$$

Similar remarks hold for evaluating the right-hand side of (3.5).

The full outline of the variation of Algorithm 2.2 is given in the following algorithm.

ALGORITHM 3.1. Suppose an initial approximate solution x_0, a tolerance TOL, and an iteration limit $MAXIT$ are given.

1. Compute $r_0 = b - Ax_0$, determine $P_1 = I - 2u_1 u_1^T$ such that $P_1 r_0 = \pm \|r_0\|_2 e_1 \equiv w$, and set $U_1 = (u_1)$ and $L_1 = (1)$.
2. For $m = 1,2, \dots, MAXIT$, do:
   a. Evaluate $v \equiv \{I - 2U_m L_m^{-1} U_m^T\} A \{I - 2U_m (L_m^T)^{-1} U_m^T\} e_m$.
   b. If $v^{(m+1)} = \cdots = v^{(n)} = 0$, then proceed to step (e); otherwise, continue.
   c. Determine $P_{m+1} = I - 2u_{m+1} u_{m+1}^T$ with u_{m+1} having first m components zero such that $P_{m+1} v$ has zero components after the $(m+1)st$.
   d. Overwrite $v \leftarrow P_{m+1} v$.
   e. If $m > 1$, overwrite $v \leftarrow J_{m-1} \dots J_1 v$.
   f. If $v^{(m+1)} = 0$, proceed to step (i); otherwise, continue.
   g. Determine J_m acting on components m and $m+1$ such that $(J_m v)^{(m+1)} = 0$.
   h. Overwrite $v \leftarrow J_m v$ and $w \leftarrow J_m w$.
   i. Set

$$R_m = \begin{cases} (v), & \text{if } m = 1; \\ (R_{m-1}, v), & \text{if } m > 1. \end{cases}$$

   j. If $|w^{(m+1)}| \le TOL$ or $m = MAXIT$, then solve for y_m and overwrite x_0 with x_m; otherwise, continue.
   k. Set $U_{m+1} = (U_m, u_{m+1})$, evaluate

$$L_{m+1} = \begin{pmatrix} L_m & 0 \\ 2u_{m+1}^T U_m & 1 \end{pmatrix},$$

   and increment m.
3. Solve for y_m and overwrite x_0 with x_m.
   a. Determine y_m which minimizes $\|w - R_m y\|_2$ by solving an $m \times m$ upper-triangular system with the first m rows of R_m as the coefficient matrix and the first m components of w as the right-hand side.
   b. Overwrite $x_0 \leftarrow x_0 + \{I - 2U_m (L_m^T)^{-1} U_m^T\} (e_1, \dots, e_m) y_m$.
   c. If $|w^{(m+1)}| \le TOL$, accept x_0 as the solution; otherwise, return to 1.

A numerical experiment. We compared the performance of Algorithm 3.1 with that of the Gram-Schmidt implementation of GMRES using the classical Gram-Schmidt process with and without reorthogonalization. There was essentially no difference in

the performance of these implementations on the problem considered in the preceding section. However, the difference was significant on a contrived problem (1.1) in which

$$A = \begin{pmatrix} 1 & 0 & \dots & 0 & \alpha \\ 0 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & n \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

As in the numerical experiment in the preceding section, we plotted the logarithm of the residual norm versus the number of iterations for these implementations, with the residual norm computed "from scratch." The computing environment and precision were the same as before, and again no preconditioning was used.

Algorithm 3.1 performed markedly better than the Gram-Schmidt implementations in our trials. The results shown in Figure 2 are typical. These results were obtained with $n = 100$, $\alpha = 2,000$, and with the maximum number of iterations allowed before restarting equal to 32. The Gram-Schmidt implementation without reorthogonalization performed very poorly. The Gram-Schmidt implementation with reorthogonalization performed considerably better but did not achieve the final residual reduction of Algorithm 3.1 and appeared to suffer greater instability as the limits of residual reduction were approached. Although not shown, the performance of Algorithm 2.2 was about the same as that of Algorithm 3.1 in this and other trials.

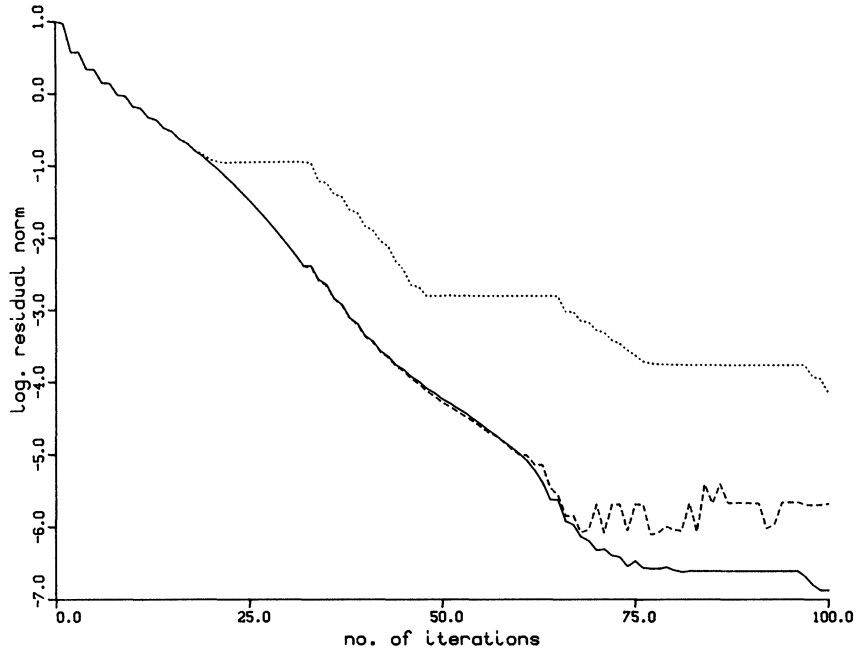

FIG. 2. *The solid line is from the variation of the Householder implementation. The dashed and dotted lines are from the Gram-Schmidt implementations using the classical Gram-Schmidt process with and without reorthogonalization, respectively.*

Acknowledgment. The author thanks Youcef Saad for providing the two test problems and the Gram-Schmidt implementation of the GMRES method used in the numerical experiments.

REFERENCES

[1] A. BJORCK, *Solving linear least-squares problems by Gram-Schmidt orthogonalization*, BIT, 7 (1967), pp. 1–21.

[2] P. N. BROWN AND A. C. HINDMARSH, *Matrix-free methods for stiff systems of ODE's*, SIAM J. Numer. Anal., 23 (1986), pp. 610–638.

[3] G. H. GOLUB, R. UNDERWOOD AND J. H. WILKINSON, *The Lanczos algorithm for the symmetric $Ax = \lambda Bx$ problem*, Computer Science Dept. Report STAN-CS-72-270, Stanford Univ., Stanford, CA, 1972.

[4] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983.

[5] A. C. HINDMARSH AND H. F. WALKER, *A note on a Householder transformation implementation of the GMRES method*, Lawrence Livermore Nat. Lab. Tech. Report UCID-20899, 1986.

[6] Y. SAAD, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105–126.

[7] ———, *Practical use of some Krylov subspace methods for solving indefinite and nonsymmetric linear systems*, this Journal, 5 (1984), pp. 203–228.

[8] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, this Journal, 7 (1986), pp. 856–869.

[9] H. F. WALKER, *Implementation of the GMRES and Arnoldi methods using Householder transformations*, Lawrence Livermore Nat. Lab. Tech. Report UCRL-93589, 1985.

[10] L. B. WIGTON, N. J. YU, AND D. P YOUNG, *GMRES acceleration of fluid dynamics codes*, Proceedings of the American Institute of Aeronautics and Astronautics 7th Computational Fluid Dynamics Conference, Cincinnati, OH, July 15–17, 1985, pp. 67–74.