A Storage Efficient WY Representation for Products of Householder Transformations

Robert Schreiber*
Charles Van Loan†

87-864

September 1987

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501

*Partially supported by ONR contract N00014-86-K-0610 and by ARO Contract DAAL03086-K-0112.

†Partially supported by ONR contract N00014-83-K-640 and NSF contract DCR 86-2310, and the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University.

A STORAGE EFFICIENT WY REPRESENTATION FOR PRODUCTS OF HOUSEHOLDER TRANSFORMATIONS

Robert Schreiber*
The SAXPY Computer Corporation
255 San Geronimo Way
Sunnyvale, California 94086

Charles Van Loan**
Department of Computer Science
Upson Hall
Cornell University
Ithaca, New York 14853

*Partially supported by ONR Contract N00014-86-K-0610 and by ARO Contract DAAL03086-K-0112.

**Partially supported by ONR contract NOOO14-83-K-640 and NSF contract DCR 86-2310.

Abstract

A product $Q = P_1 \cdots P_r$ of m-by-m Householder matrices can be written in the form $Q = I + WY^T$ where W and Y are each m-by-r. This is called the WY representation of Q. It is of interest when implementing Householder techniques in high-performance computing environments that "like" matrix-matrix multiplication. In this note we describe a storage efficient way to implement the WY representation. In particular, we show how the matrix Q can be expressed in the form $Q = I + YTY^T$ where $Y \in R^{m \times r}$ and $T \in R^{r \times r}$ with T upper triangular. Usually $r << m$ and so this "compact" WY representation requires less storage. When compared with the recent block-reflector strategy proposed by Schreiber and Parlett the new technique still has a storage advantage and involves a comparable amount of work.

Introduction

Many important eigenvalue and least square methods rely on Householder matrices, i.e., matrices of the form $H = I - 2vv^T$ where v has unit 2-norm. (The normalization of the Householder vector v is unnecessary in practice but convenient for exposition.) Householder matrices are orthogonal and can be used to zero selected portions of a matrix, see Golub and Van Loan (1983). Important implementations are discussed in the LINPACK or EISPACK manuals.

The QR factorization of an m-by-n matrix A is a good setting for describing Householder matrix use. In this application Householder matrices $H_1,...,H_n$ are generated such that $H_n \cdots H_2 H_1 A = R$ is upper triangular:

Algorithm 1 (Householder QR)

For k = 1:n
    Determine Householder H_k such that if $z = H_k A(1{:}m,k{:}k)$
        then $z(k+1{:}m) = 0$.
    $A \leftarrow H_k A$
end k

The Householder update $A \leftarrow H_k A$ is "rich" in matrix-vector multiplication. Unfortunately, many of the new supercomputing architectures require code that is rich in matrix-matrix multiplication in order to attain near-peak performance. This accounts for the current interest in the level-3 BLAS and the increasingly vigorous search for efficient block algorithms.

Along these lines there has been some recent work on "block" Householder methods. Two such techniques are of interest to us. They are the WY representation due to Bischof and Van Loan (1987) and the block reflector method of Schreiber and Parlett (1987). Both of these techniques can be used to solve what we shall refer to as problem **P**:

P: Given $B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$ find an orthogonal Q so $Q^T B = C = \begin{bmatrix} C_1 \\ 0 \end{bmatrix}$

Here, $B,C \in R^{mxr}$ and $B_1,C_1 \in R^{rxr}$. In the WY approach the solution to problem **P** is represented in the form

$$(1) \qquad Q = I + WY^T \qquad W,Y \in R^{mxr}$$

where Y is lower trapezoidal, i.e., $y_{ij} = 0$ if $i < j$. The submatrix C_1 is upper triangular. Q is a rank r correction to the identity and so can be regarded as a generalization of a Householder matrix.

A different solution to problem **P** is obtained by the block reflector approach. It has the form

$$(2) \qquad Q = I - GG^T \qquad G \in R^{mxr}.$$

and obviously can be regarded as a block Householder matrix. It requires

m·r storage locations to represent Q in this fashion, about half of that required by the WY representation. The block reflector solution to problem **P** does not in general yield an upper triangular C_1.

The point of this note is to show how to modify the WY representation so that only m·r storage is necessary. In particular, we show how to write the matrix Q in (1) as

$$Q = I + YTY^T \tag{3}$$

where $Y \in R^{m \times r}$ (lower trapezoidal) and $T \in R^{r \times r}$ (upper triangular). We refer to (3) as the *compact WY representation* of Q and we discuss its use in the design of a block Householder QR procedure.

Block QR Procedures

Any of the above block Householder representations can be used to implement a block QR factorization procedure that is rich in matrix multiplication. Suppose $A \in R^{m \times n}$ and that n = rN.

Algorithm 2 (Block Householder QR)

```
For k = 1:N
    s ← (k-1)·r+1
    Determine block Householder Q such that if
        Q^T A(s:m,s:s+r-1) = C  then  C is zero below row r .
    A(s:m,s:n) ← Q^T A(s:m,s:n)
end
```

Note that the update of B = A(s:m,s:n) is rich in matrix multiplication if Q is represented as $I + WY^T$ or $I - GG^T$ or $I + YTY^T$, i.e.,

$$B \leftarrow (I + WY^T)^T B = B + Y \cdot (W^T B)$$

In the case of the WY representation, the k-th W and Y are generated from the Householder matrices $P_j = I - 2v_jv_j^T$, j = 1:r, that upper triangularize the submatrix A(s:m,s:s+r-1). The procedure is simple.

Algorithm 3 (WY Generation)

```
For j = 1:r
    If j = 1
        W ← [ - 2v1 ] ; Y ← [ v1 ]
    else
        z ← -2(I + WY^T)vj ; W ← [ W z ] ; Y ← [ Y vj ]
    endif
end
```

It follows that $I + WY^T = P_1 \cdots P_r$ and so A is reduced to upper triangular form when Algorithm 2 is implemented with block Householders in WY form. Note that each Y is just the aggregation of the Householder vectors and so is lower trapezoidal. Thus, all the Y matrices fit into the zeroed portion of A as Algorithm 2 proceeds. If the W-factors are saved then a workspace of size

$$\sum_{k=1}^{N} (m - (k-1)\cdot r)\cdot r \approx mn - n^2/2$$

is required. This is a serious storage overhead although in many applications it is not necessary to store all the W matrices.

These storage concerns do not arise if the block reflector approach is taken. If we chose to write Q as $I - GG^T$ then the Schreiber-Parlett procedure can be used to generate the G matrices in Algorithm 2 as follows.

Algorithm 4 (G Generation)

- Compute Householders $P_1,\cdots,P_r$ such that $P_r\cdots P_1 A(s{:}m,s{:}s+r-1)$ is upper triangular

- Let $\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$ be the first r columns of $P_1\cdots P_r$ with $U_1 \in R^{r\times r}$

- Compute orthogonal $Z \in R^{r\times r}$ and symmetric nonnegative definite $M \in R^{r\times r}$ such that $U_1 = ZM$, the polar decomposition.

- Compute Cholesky factorization $LL^T = I + M$.

- Set $G = \begin{bmatrix} ZL \\ U_2 L^{-T} \end{bmatrix}$

A block reflector implementation of Algorithm 2 results in a block upper triangular reduction of A. Subsequent computations are necessary to obtain true triangular form. If all the G matrices are to be saved, then a workspace of size $Nr^2 = nr$ is required. (There is no room in the A array for the top r-by-r portions of each G.)

To sum up, the WY representation is simpler to compute but it requires a much larger workspace if all the block Householder factors are saved ($mn - n^2/2$ vs. nr).

The Compact WY Representation

The idea behind the compact WY representation is to exploit a connection between the W and Y matrices in (1). Here is the main result that enables us to build up Y and T just as W and Y are constructed in Algorithm 3.

Theorem 1. (Compact WY update)

Suppose $Q = I + YTY^T \in R^{m\times m}$ is orthogonal with $Y \in R^{m\times j}$ $(m > j)$ and $T \in R^{j\times j}$ (upper triangular). Suppose $P = I - 2vv^T$ is a Householder matrix with $v \in R^m$ and $\| v \|_2 = 1$. If

$$Q_+ = QP$$

then

$$Q_+ = I + Y_+ T_+ Y_+^T$$

where $Y_+ = [\, Y \; v \,] \in R^{m\times(j+1)}$ and

$$T_+ = \begin{bmatrix} T & z \\ 0 & \rho \end{bmatrix}$$

with $\rho = -2$ and $z = -2TY^Tv$.

Proof.

Note that

$$I + Y_+ T_+ Y_+^T = I + [\, Y \; v \,] \begin{bmatrix} T & z \\ 0 & \rho \end{bmatrix} \begin{bmatrix} Y^T \\ v^T \end{bmatrix}$$

$$= I + [\, Y \; v \,] \begin{bmatrix} TY^T + zv^T \\ \rho v^T \end{bmatrix}$$

$$= I + YTY^T + Yzv^T + \rho vv^T .$$

This equals

$$Q_+ = QP = (I + YTY^T)(I - 2vv^T) = I + YTY^T - 2YTY^TVV^T - 2vv^T$$

so long as we set $\rho = -2$ and $z = -2YT^Tv$. □

Returning to Algorithm 2, if $P_j = I - 2v_jv_j^T$, $j = 1{:}r$, are the Householder matrices that upper triangularize $A(s{:}m,s{:}s+r-1)$ during the k-th step then here is how T and Y are determined so $P_1 \cdots P_r = I + YTY^T$:

Algorithm 5 (YT Generation)

For $j = 1{:}r$
  if $j = 1$ then
    $Y \leftarrow [\, v_1 \,]$; $T \leftarrow [\, -2 \,]$
  else
    $z \leftarrow -2 \cdot T\, Y^T v_j$
    $Y \leftarrow [\, Y \;\; v_j \,]$

$$T \leftarrow \begin{bmatrix} T & z \\ 0 & -2 \end{bmatrix}$$

  endif
end j

Remarks

We conclude with a number of remarks about the compact WY representation and its use in block Householder schemes like Algorithm 2. To begin with, if each triangular T matrix is saved then a workspace of size $N \cdot r^2/2 = nr/2$ is required. As with WY, the Y matrices fit in the zeroed portion of A and so no additional workspace is required for their

storage. Thus, compact WY is the most storage efficient of the three representations discussed.

From the standpoint of actually generating the representation factors, {W,Y} , {G}, or { T,Y}, the compact WY representation is also the most efficient. This can be seen by comparing Algorithms 3, 4, and 5.

We next compare the cost of the update $B \leftarrow Q^T B$ where Q is in one of our three block Householder forms. If $B \in R^{n_1 \times n_2}$ and rank(Q - I) = r , then simple flop counts reveal that the updates

$$\text{(a)} \quad B \leftarrow (I + WY^T)^T B = B + Y \cdot (W^T B)$$

$$\text{(b)} \quad B \leftarrow (I - GG^T)^T B = B - G \cdot (G^T B)$$

$$\text{(c)} \quad B \leftarrow (I + YTY^T)^T B = B + (Y \cdot T^T) \cdot (Y^T \cdot B)$$

$$\text{(d)} \quad B \leftarrow (I + YTY^T)^T B = B + Y \cdot (T^T \cdot (Y^T \cdot B))$$

each require about the same amount of arithmetic: $2n_1n_2r$ flops. (If Y's trapezoidal structure is exploited then the flops thus saved offset those needed for the T multiplications.) Of course, counting flops is a crude predictor of performance. There is some penalty associated with the compact WY form as it involves three matrix-matrix operations instead of just two. However, if $r << n_1$ or n_2, as is often the case, then the multiplications involving T are not significant. Note that there are two possible strategies for compact WY updates, see (c) and (d) above. The proper choice depends upon n_1, n_2, r, and the underlying architecture.

We mention that a QR factorization routine based on the compact WY representation is part of the new level-3 BLAS LINPACK.

Last, but not least, the compact WY representation is stable. We delete the proof as it closely follows the demonstration of stability given in Bischof and Van Loan (1987) for the ordinary WY representation.

Acknowledgement

We would like to thank Ralph Schmidt of SAXPY Corporation for participating in early formulations of this work.

References

C. Bischof and C. Van Loan (1987), "The WY representation for products of Householder matrices," *SIAM J. Scientific and Statistical Computing*, 8, s2-s13.

G.H. Golub and C. Van Loan (1987), *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md.

R. Schreiber and B.N. Parlett (1987), "Block reflectors: theory and computation," *SIAM J. Numer. Analysis*, to appear.