On Accumulating Householder Transformations

Thierry Joffrain*
Tze Meng Low*
Enrique S. Quintana-Ortí†
Robert van de Geijn*
Field Van Zee *

FLAME Working Note #13

October 12, 2004

Abstract

A theorem related to the accumulation of Householder transformations into a single orthogonal transformation, known as the compact WY transform, is presented. It provides a simple characterization of the computation of this transformation and suggests an alternative algorithm for computing it. It also suggests an alternative transformation, the UT transform, with the same utility as the compact WY Transform, which requires less computation and has similar stability properties. That alternative transformation was first published over a decade ago, but has gone unnoticed by the community.

1 Introduction

Given a nonzero vector $u \in \mathbb{R}^m$, a *Householder transformation* (or *reflector*) is defined by $H = I - \frac{uu^T}{\tau}$, where I denotes the (square) identity matrix and $\tau = \frac{u^T u}{2}$ [4]. It is an orthogonal matrix ($H^T H = HH^T = I$) and its own transpose ($H^T = H$). This transformation has wide application in the solution of linear least-squares problems, the computation of orthonormal bases, and the solution of the algebraic eigenvalue problem.

Two transforms that capture the action of multiple Householder transformations and cast it in terms of high-performance matrix-matrix products were proposed in the late 1980s, the WY transform [1] and the compact WY transform [6] (CWY). A third such transform was proposed and published by Walker in 1988 [8], in the setting of a GMRES algorithm based on Householder transformations, and rediscovered by Puglisi in 1992 in the setting of the QR factorization [5]. Yet few in the numerical analysis community appear to be aware of these results as they relate to the CWY [7]. It was a brief brainstorming session involving the authors of this paper that independently rediscovered this result once again. We believe the result to be of sufficient importance that it warrants republishing.

In Section 2 we review the traditional way in which the CWY is computed. In Section 3 we present the main theorem that characterizes the accumulation of Householder transformations. In Section 4 we discuss opportunities that appear due to the alternative characterization. Remarks on how to modify LAPACK to accommodate the insights are given in Section 5. Experimental results are presented in Section 6, followed by concluding remarks in the final section.

*Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712, {joffrain,ltm,rvdg,field}@cs.utexas.edu.

†Departamento de Ingeniería y Ciencia de Computadores, Universidad Jaume I, Campus Riu Sec, 12.071 – Castellón, Spain, quintana@icc.uji.es.

Partition $U \rightarrow \left(U_L \,\middle|\, U_R \right)$, $t \rightarrow \left(\frac{t_T}{t_B} \right)$, $S \rightarrow \left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right)$
where U_L has 0 columns, t_T has 0 rows, and S_{TL} is 0×0

while $m(S_{TL}) < m(S)$ **do**
Repartition
$\left(U_L \,\middle|\, U_R \right) \rightarrow \left(U_0 \,\middle|\, u_1 \,\middle|\, U_2 \right)$,
$\left(\begin{array}{c} t_T \\ \hline t_B \end{array} \right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array} \right)$, $\left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} S_{00} & s_{01} & S_{02} \\ \hline & \sigma_{11} & s_{12}^T \\ \hline & & S_{22} \end{array} \right)$
where u_1 has one column, and τ_1, σ_{11} are scalars

$s_{01} := U_0^T u_1$
$\sigma_{11} := 1/\tau_1 \ (= 2/u_1^T u_1)$
$s_{01} := -S_{00} s_{01} \sigma_{11}$

Continue with
$\left(U_L \,\middle|\, U_R \right) \leftarrow \left(U_0 \,\middle|\, u_1 \,\middle|\, U_2 \right)$,
$\left(\begin{array}{c} t_T \\ \hline t_B \end{array} \right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array} \right)$, $\left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} S_{00} & s_{01} & S_{02} \\ \hline & \sigma_{11} & s_{12}^T \\ \hline & & S_{22} \end{array} \right)$
endwhile

Figure 1: Traditional algorithm for computing S.

2 Computing the compact WY transform

The following theorem presents the traditional formulae for accumulating Householder transformations into a CWY:
Theorem 1 *Let the matrix $U_{k-1} \in \mathbb{R}^{m \times k}$ be partitioned by columns as*

$$U_{k-1} = \left(u_0 \,\middle|\, u_1 \,\middle|\, \cdots \,\middle|\, u_{k-1} \right),$$

and consider the vector $t = (\tau_0, \tau_1, \ldots, \tau_{k-1})^T$. Then, there exists a unique $k \times k$ nonsingular upper triangular matrix S_{k-1} such that

$$\left(I - \frac{u_0 u_0^T}{\tau_0} \right) \left(I - \frac{u_1 u_1^T}{\tau_1} \right) \cdots \left(I - \frac{u_{k-1} u_{k-1}^T}{\tau_{k-1}} \right) = \left(I - U_{k-1} S_{k-1} U_{k-1}^T \right).$$

Matrix S_j can be computed via the recurrence

$$S_0 = 1/\tau_0 \quad \textit{and} \quad S_j = \left(\begin{array}{c|c} S_{j-1} & -S_{j-1} U_{j-1}^T u_j / \tau_j \\ \hline 0 & 1/\tau_j \end{array} \right), \quad 1 \leq j < k. \tag{1}$$

Proof: By induction on k. An algorithm for computing this transformation based on (1) is given in Fig. 1.

3 Central result

We now state a theorem that will give a simpler characterization of the relation between U and S.
Theorem 2 *Let $U \in \mathbb{R}^{m \times k}$ be a matrix of full column rank. There exists a unique nonsingular upper triangular matrix $S \in \mathbb{R}^{k \times k}$ such that $I - USU^T$ is an orthogonal matrix. This matrix S satisfies $S = T^{-1}$ with $T + T^T = U^T U$, where T is itself a unique upper triangular matrix.*

Proof: Since $I - USU^T$ is orthogonal,

$$\begin{aligned} 0 &= I - (I - USU^T)(I - USU^T)^T = I - (I - USU^T)(I - US^TU^T) \\ &= I - (I - US^TU^T - USU^T + USU^TUS^TU^T) = U\left[(S^T + S) - SU^TUS^T\right]U^T. \end{aligned}$$

Thus, $S^T + S = SU^TUS^T$ since U has full column rank. Now, as matrix S is required to be nonsingular, $S^{-1}(S^T + S)S^{-T} = S^{-1}SU^TUS^TS^{-T}$ and therefore

$$S^{-1} + S^{-T} = U^TU. \tag{2}$$

Finally, replacing S^{-1} by T in (2) we find that $T = \text{striu}(U^TU) + \frac{1}{2}\text{diag}(U^TU)$ uniquely defines the upper triangular matrix T. Here striu(A) denotes the part of matrix A that lies strictly above the diagonal of that matrix, and diag(A) equals the diagonal matrix that has the same diagonal as A.

Under the assumptions of the above theorem, S can be computed by the steps

1. $S :=$ the upper triangular part of U^TU.
2. Divide the diagonal elements of S by two.
3. $S := S^{-1}$.

An algorithm for the first step is given in the top part of Fig. 2 while an algorithm that combines the last two steps is given in the bottom part of that figure.

Note 1 *Puglisi arrived at the result in Theorem 2 by applying the Woodbury-Morrison formula to $I - USU^T$. We believe our proof to simpler and more revealing.*

The two algorithms in Fig. 2 together implement *exactly* the same computation as the traditional algorithm in Fig. 1, except that rather than computing σ_{11} in three steps ($\sigma_{11} := u_1^Tu_1$; $\sigma_{11} := \sigma_{11}/2$; $\sigma_{11} := 1/\sigma_{11}$) the traditional algorithm simply sets σ_{11} to τ_1, which has the same net result:

Update in Fig. 1	Update in Fig. 2
$s_{01} := U_0^Tu_1$	$s_{01} := U_0^Tu_1$
$\sigma_{11} := 1/\tau_1 \left(= 2/(u_1^Tu_1)\right)$	$\begin{cases} \sigma_{11} := u_1^Tu_1 \\ \sigma_{11} := \sigma_{11}/2 \\ \sigma_{11} := 1/\sigma_{11} \end{cases}$
$s_{01} := -S_{00}s_{01}\sigma_{11}$	$s_{01} := -S_{00}s_{01}\sigma_{11}$

Other than one additional recomputation of $u_1^Tu_1/2$ per diagonal element of S, the two algorithms perform the same operations. Therefore, they will have very similar cost and numerical stability. This additional computation is an artifact of the fact that the level-3 Basic Linear Algebra Subprograms (BLAS) routine DSYRK [2], which would typically be used to compute U^TU, also recomputes the diagonal of the result. Clearly, σ_{11}, the diagonal element of S, could simply be set to $1/\tau_1$ in Fig. 2. The computation of U^TU and the inversion of S can be implemented using any algorithm for those operations, not just the ones in Fig. 2.

Note 2 *Puglisi makes the same connection between the traditional algorithm for computing S and the separate steps mentioned above.*

4 Opportunities

While the result in the previous section provides a simple theoretical characterization of the relation between the Householder vectors and the CWY, we now show how it provides opportunities for performance and numerical stability.

Partition $U \rightarrow \left(U_L \,\|\, U_R \right)$, $S \rightarrow \left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right)$

  where U_L has 0 columns and S_{TL} is 0×0

while $m(S_{TL}) < m(S)$ **do**

  Repartition

$$\left(U_L \,\|\, U_R \right) \rightarrow \left(U_0 \,\|\, u_1 \,|\, U_2 \right), \left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} S_{00} & s_{01} & S_{02} \\ \hline & \sigma_{11} & s_{12}^T \\ \hline & & S_{22} \end{array} \right)$$

  where u_1 is a column and σ_{11} is a scalar

$s_{01} := U_0^T u_1$
$\sigma_{11} := u_1^T u_1$

  Continue with

$$\left(U_L \,\|\, U_R \right) \leftarrow \left(U_0 \,|\, u_1 \,\|\, U_2 \right), \left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} S_{00} & s_{01} & S_{02} \\ \hline & \sigma_{11} & s_{12}^T \\ \hline & & S_{22} \end{array} \right)$$

endwhile

Partition $S \rightarrow \left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right)$

  where S_{TL} is 0×0

while $m(S_{TL}) < m(S)$ **do**

  Repartition

$$\left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} S_{00} & s_{01} & S_{02} \\ \hline & \sigma_{11} & s_{12}^T \\ \hline & & S_{22} \end{array} \right)$$

  where σ_{11} is a scalar

$\sigma_{11} := \sigma_{11}/2$
$\sigma_{11} := 1/\sigma_{11}$
$s_{01} := -S_{00} s_{01} \sigma_{11}$

  Continue with

$$\left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} S_{00} & s_{01} & S_{02} \\ \hline & \sigma_{11} & s_{12}^T \\ \hline & & S_{22} \end{array} \right)$$

endwhile

Figure 2: Computing S as proposed in Section 3. Top: Compute $S := U^T U$ (upper triangular part only). Bottom: Divide the diagonal elements of S by 2 and compute $S := S^{-1}$.

4.1 Potential impact on performance

The traditional algorithm in Fig. 1 is rich in matrix-vector products, a level-2 BLAS [3] operation. By contrast, Steps 1–3 in Section 3 can inherently attain high performance: Step 1 can be implemented by a call to an optimized implementation of the level-3 BLAS routine DSYRK while the LAPACK routine DTRTRI can be used for Step 3. Typically, k is small enough so that the inversion of the $k \times k$ matrix in Step 3 will keep that matrix in cache memory, making that operation inherently efficient.

Note 3 *Puglisi makes the same observation.*

4.2 The UT transform

$I - UT^{-1}U^T$ represents an alternative expression for the accumulation of the Householder transformations. This formulation eliminates the need for the k^3 floating-point operations (flops) required to compute $S := T^{-1}$. We call this formulation *the UT transform.*

The CWY is typically formed so that it can be applied to a matrix $A \in \mathbb{R}^{m \times n}$, as in the computation $A := (I - USU^T)A$. One can instead compute $(I - UT^{-1}U^T)A$. The parentheses in the following expressions indicate the order in which operations in these two approaches are typically performed:

$$A := A - U[S \; \underbrace{[U^T A]}_{W} \;] \quad \text{versus} \quad A := A - U[T^{-1} \; \underbrace{[U^T A]}_{W} \;].$$

The computation of SW and $T^{-1}W$, via the level-3 BLAS routines DTRMM and DTRSM, respectively, requires *exactly the same number* of flops. Thus, avoiding the inversion of matrix T translates directly into k^3 fewer flops being performed.

Note 4 *Puglisi makes the same observation.*

For different implementations of the BLAS, DTRSM may attain better or worse performance than DTRMM. This would influence whether to compute and use the UT transform or the CWY.

4.3 Potential impact on numerical stability

Householder transformations are inherently used because of their exceptional stability properties. The CWY is known to inherit these properties. Nonetheless, it is also well known that computing $W := T^{-1}W$ as a triangular solve with multiple right-hand sides is numerically more stable than computing $W := SW$ after explicitly inverting $S := T^{-1}$. Thus, the UT transform is at least as stable as the CWY, and possibly more stable.

Note 5 *Puglisi makes a similar comment regarding stability.*

5 Modifications to LAPACK

We now give details of how minor modifications to LAPACK can be made to incorporate the insights in this paper.

A detail that is not made obvious in the previous discussion is that the matrix U that stores the Householder vectors as they are computed during a QR factorization has the form $U = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$, where U_1 is unit lower triangular. Thus, the computation $S = U^T U$ can be broken down into $S := U_1^T U_1$ followed by $S := S + U_2^T U_2$, computing only the upper triangular part. The term $U_2^T U_2$ is a simple call to DSYRK. The problem is that there is no routine in the BLAS or LAPACK that computes only the upper triangular part of $S = U_1^T U_1$ while taking advantage of the special structure of U_1.

$S := \mathrm{triu}(U_2^T U_2)$

Partition $U_1 \rightarrow \left(\begin{array}{c|c} U_L & U_R \end{array} \right)$, $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array} \right)$, $S \rightarrow \left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right)$

  where U_L has 0 columns, t_T has 0 rows, and S_{TL} is 0×0

while $m(S_{TL}) < m(S)$ **do**

  Repartition

$$\left(\begin{array}{c|c} U_L & U_R \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} U_0 & u_1 & U_2 \end{array} \right),$$

$$\left(\begin{array}{c} t_T \\ \hline t_B \end{array} \right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array} \right), \left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} S_{00} & s_{01} & S_{02} \\ \hline & \sigma_{11} & s_{12}^T \\ \hline & & S_{22} \end{array} \right)$$

  where u_1 has one column, and τ_1, σ_{11} are scalars

Compute S:	**Compute $T = S^{-1}$ in S:**
$s_{01} := s_{01} + U_0^T u_1$	$s_{01} := s_{01} + U_0^T u_1$
$\sigma_{11} := 1/\tau_1$	$\sigma_{11} := \tau_1$
$s_{01} := -S_{00} s_{01} \sigma_{11}$	

  Continue with

$$\left(\begin{array}{c|c} U_L & U_R \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} U_0 & u_1 & U_2 \end{array} \right),$$

$$\left(\begin{array}{c} t_T \\ \hline t_B \end{array} \right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array} \right), \left(\begin{array}{c|c} S_{TL} & S_{TR} \\ \hline & S_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} S_{00} & s_{01} & S_{02} \\ \hline & \sigma_{11} & s_{12}^T \\ \hline & & S_{22} \end{array} \right)$$

endwhile

Figure 3: Modification of traditional algorithm for computing S and T.

To overcome this, let us examine routine DLARFT from LAPACK, which computes the matrix S via the algorithm in Fig. 1. Now, S can be computed by initializing it to the upper triangular part of $U_2^T U_2$ changing the update $s_{01} := U_0^T u_1$ to $s_{01} := s_{01} + U_0^T u_1$ in Fig. 1, and executing this modified algorithm with U_1 rather than all of U! Thus, first S is set to $U_2^T U_2$, after which the remaining computations are all accomplished by the modification given in Fig. 3 (left). This approach casts most computations in terms of $U_2^T U_2$ (DSYRK) and, in one sweep, performs the remaining computation with matrices that are small enough to remain in cache. This is coded by modifying DLARFT, adding a call to DSYRK with U_2 before the loop, changing the upper limit of the loop from `N` (the row dimension of U) to `K` (the row dimension of U_1), and changing a `ZERO` to a `ONE` in the call to DGEMV so that the result of the matrix-vector multiply is added to s_{01}. Let us call the result DLARFT_NEW.

The new routine DLARFT_NEW can then be turned into a computation of T by further changing the algorithm in Fig. 1, replacing $\sigma_{11} = 1/\tau_1$ by $\sigma_{11} = \tau_1$ and deleting the update $s_{01} = -S_{00} s_{01} \sigma_{11}$, as illustrated in Fig. 3 (right). This translates to a change in one line of DLARFT_NEW and the deletion of one call to DTRMV. Applying the UT transform so computed requires only that a single call to DTRMM be changed to a call to DTRSM in DLARFB.

6 Experiments

We demonstrate the potential of the alternative approaches by modifying the LAPACK routines for computing and applying the CWY, DLARFT and DLARFB, and measuring its effect on the LAPACK QR factorization routine, DGEQRF.

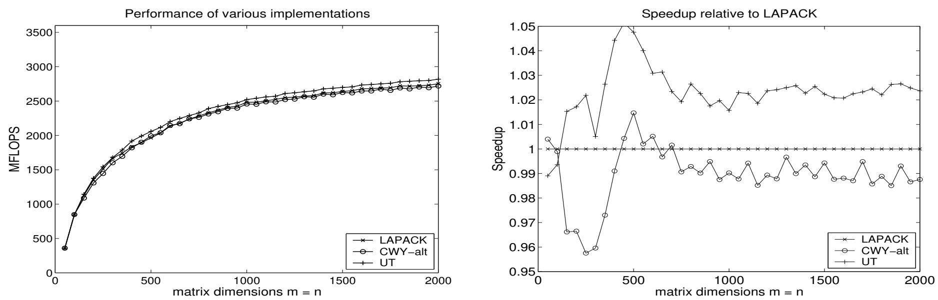

Figure 4: Performance of the various implementations.

6.1 Implementation

Three different implementations were examined: **LAPACK:** The standard LAPACK implementation; **CWY-alt:** The modified LAPACK implementation based on the algorithm in Fig. 3 (left); and **UT:** The modified LAPACK implementation based on the UT transform as described in Fig. 3 (right).

6.2 Performance

The impact of the described modifications was measured by computing the QR factorization of matrices of various sizes and using the result to solve a linear system. The target platform was an Intel Itanium2 (900MHz) processor based workstation, using a BLAS library implemented by Kazushige Goto[1]. The results are reported in Fig. 4. In all experiments, a blocksize of $k = 32$ was used.

Casting most computation in DLARFT in terms of DSYRK yields a slight degradation in performance. We speculate that is due to inefficiencies in the implementation of that routine for the specific matrix dimensions that are encountered in our computation. By switching to the implementation based on the UT transform, modest performance improvements are observed. As part of a QR factorization the amount of computation that is performed in the routines that were optimized constitutes a lower order term, so that modest improvements are all that can be expected.

Note 6 *Puglisi also comments on the inefficiency of the* DSYRK *operation.*

6.3 Numerical stability

The effects of the modifications on numerical stability were also examined. Since the matrices S, as computed from the matrices U encountered in the QR factorization, are extremely well-conditioned, no meaningful improvements or degradations in the quality of the residual were observed.

7 Conclusion

In this paper an alternative characterization of the compact WY transform was given. The characterization suggests a simpler approach to computing that transform and an alternative way of accumulating Householder transformations, the UT transform, which eliminates the cost of the inversion of a triangular matrix. This alternative transform was already discovered by Puglisi, a result that appears to have been lost to the community. Modest performance gains were demonstrated when the insights were incorporated into the corresponding LAPACK routines.

[1]Visit the URL `http://www.cs.utexas.edu/users/kgoto/`.

Acknowledgments

This research was partially sponsored by NSF grants ACI-0305163 and CCF-0342369 and an equipment donation from Hewlett-Packard.

References

[1] Christian Bischof and Charles Van Loan. The WY representation for products of Householder matrices. *SIAM J. Sci. Stat. Comput.*, 8(1):2–13, Jan. 1987.

[2] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.

[3] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Richard J. Hanson. An extended set of FORTRAN basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 14(1):1–17, March 1988.

[4] Alston S. Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM*, 5:339–42, 1958.

[5] C. Puglisi. Modification of the Householder method based on the compact wy representation. *SIAM J. Sci. Stat. Comput.*, 13:723–726, 1992.

[6] Robert Schreiber and Charles Van Loan. A storage-efficient WY representation for products of Householder transformations. *SIAM J. Sci. Stat. Comput.*, 10(1):53–57, Jan. 1989.

[7] Xiaobai Sun. Aggregations of elementary transformations. Technical Report Technical report DUKE–TR–1996–03, Duke University, 1996.

[8] H. F. Walker. Implementation of the GMRES method using Householder transformations. *SIAM J. Sci. Stat. Comput.*, 9(1):152–163, 1988.