HOUSEHOLDER REFLECTIONS VERSUS GIVENS ROTATIONS IN SPARSE ORTHOGONAL DECOMPOSITION*

J.A. George
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

J.W.H. Liu
Department of Computer Science
York University
Downsview, Ontario, Canada
M3J 1P3

Research Report CS-84-42
November 1984

Householder Reflections versus Givens Rotations in Sparse Orthogonal Decomposition*

Alan George

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3Gl

Joseph W.H. Liu

Department of Computer Science
York University
Downsview, Ontario
Canada M3J 1P3

ABSTRACT

It has been generally assumed that the use of Givens rotations provides significant advantages over the use of Householder transformations for the orthogonal decomposition of sparse matrices. It is also generally acknowledged that the opposite is true for dense matrices. In this paper, a way of applying Householder reflections is described which is competitive or superior to the use of Givens rotations for sparse orthogonal decomposition. In other words, the advantage of Householder over Givens for dense matrices can carry over to the sparse case, provided that the implementation of the Householder scheme is done in a certain way. The approach relies heavily on the idea of general row merge trees developed by Liu [12]. Results of numerical experiments are provided which demonstrate the advantages of the new scheme. The method also appears to be attractive for use on vector computers.

*Research supported in part by Canadian Natural Sciences and Engineering Research Council under grants A8111 and A5509, by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems Inc., and by the U.S. Air Force Office of Scientific Research under contract AFOSR-ISSA-84-00056.

Table of Contents

1. Introduction

Householder reflections and Givens rotations are standard basic computational operations which are used to compute the orthogonal decomposition of matrices. For a given m by n matrix A, a sequence of $n-1$ Householder transformations can be applied to reduce A to upper triangular form. Alternatively, a sequence of $m-1$ Givens row rotations (that is, a sequence of $mn-n(n+1)/2$ actual rotations) can be used to achieve the reduction.

When the matrix A is dense, Householder transformations have been employed almost exclusively (LINPACK [2]). However, if the matrix is large and sparse, Givens rotations are generally used or recommended [4,5,9]. The scheme proposed by George and Heath [5] makes use of Givens rotations and has the advantage of using a static predetermined data structure for the upper triangular factor matrix. The rows of A are processed one by one, gradually forming the factor matrix, the storage of which has been pre-allocated. "Intermediate fills" are restricted to the working row, and they are annihilated during the processing of this row.

On the other hand, Householder transformations have been generally regarded as quite *inappropriate* for sparse QR decomposition [3,8,9]. The application of Householder column reflections can cause severe intermediate fills. Although they will eventually be annihilated, temporary storage is required to accomodate them, which often turns out to exceed greatly the number of nonzeros in the final factor matrix.

The following 8 by 4 matrix example serves to illustrate the problem with Householder transformations. The letter "i" is used to denote intermediate fills.

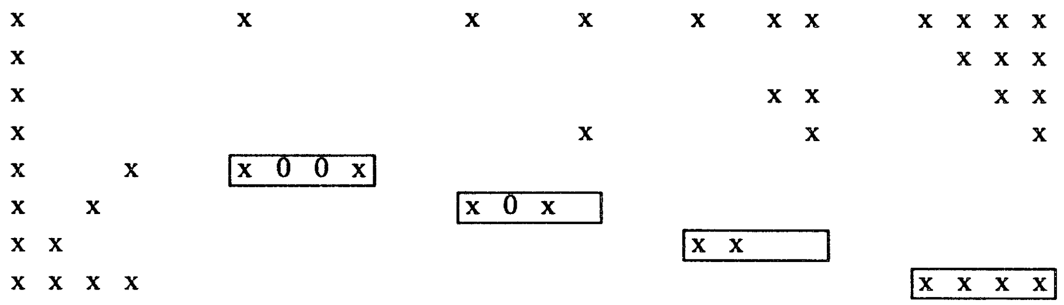


Figure 1.1: Sequence of Givens row rotations

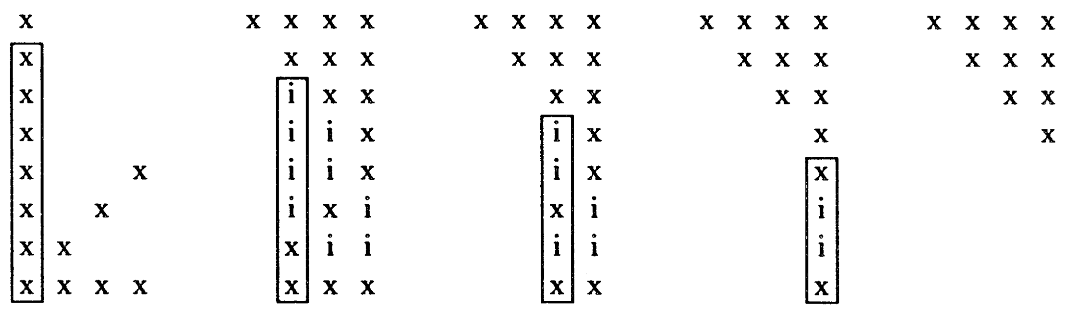


Figure 1.2: Sequence of Householder column reflections

One of the main objectives of this paper is to propose a way of applying Householder transformations so that it becomes competitive with Givens rotations for sparse orthogonal decomposition. We hope that this will lead to a re-examination of the role of Householder transformations in sparse computations. Indeed, if this new approach is used, most of the advantages of Householder reflections for the dense case now carry over to sparse systems, while its main disadvantage is removed.

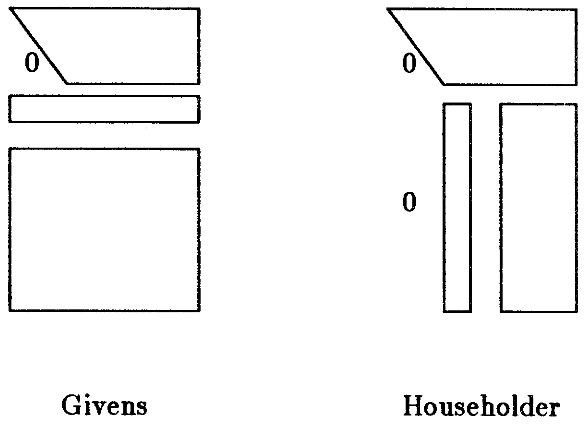
The approach uses the idea of ***general row merge trees*** as developed by the second author in [12]. In section 2, a matrix interpretation of the general row merge scheme is given. We also relate the use of the ***submatrix annihilation*** technique to some previous work in the literature.

The basic algorithm is described briefly in section 3. The main difference between this algorithm and that of [12] is the use of Householder transformations instead of Givens rotations in the core of the numerical *QR* factorization module. A ***row-oriented*** version of Householder reflection is presented to adapt to the computational scheme. A minor modification to the overall merging scheme, motivated by the use of Householder transformations, is also given in this section.

Numerical experiments are provided in section 4 to compare Householder to Givens. Based on the experimental results reported, Householder reflections do have a role to play in sparse orthogonal factorization. It is consistently faster (in terms of operation counts), and more accurate, in exchange for a very modest increase in working storage. Section 4 also contains our concluding remarks. It is interesting to point out that the row-oriented version of Householder transformations can be adapted to vector computation. Its performance for vector machines, however, will be explored elsewhere.

2. Selective Submatrix Annihilation

A conventional Givens method is usually implemented as a row-oriented scheme, in which rows are annihilated one by one using the partially formed upper triangular factor. On the other hand, the Householder method is always treated as a column-oriented scheme, and the lower triangular portion of each column is annihilated, column by column.



The proposed scheme in this paper can be viewed as one using a ***submatrix annihilation*** technique. Instead of annihilating an entire row or an entire column, a sequence of submatrices within the given sparse matrix is annihilated one after another so that eventually the matrix is reduced to upper triangular.

The added flexibility in the choice of objects to annihilate can lead to major savings in terms of intermediate fills and arithmetic operations. In order to achieve this saving, care must be exercised in the choice of the sequence of submatrices, so that zeros created at one point will not become nonzero again as an intermediate fill at a later stage.

The example in Figure 2.1 is designed to illustrate possible gains by using submatrix annihilation. The submatrix processed at each step is enclosed by rectangles. Only two intermediate fills (labelled by "i") occur in this example for this sequence of submatrices. An "f" is used in the figure to represent actual fill in the factor matrix.

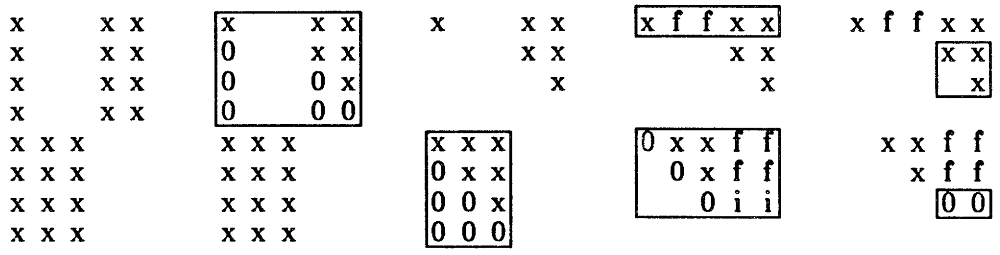


Figure 2.1: Selective submatrix annihilation sequence.

This new approach of submatrix annihilation originates from the *general row merging scheme* [12] for the sparse QR decomposition using Givens rotations. It was shown there that the numerical computation in the row merge scheme can be organized as a sequence of reductions of two upper triangular (strictly speaking, trapezoidal) full submatrices into another upper triangular full matrix. Indeed, submatrix annihilation is simply another *interpretation* of this scheme, whereby the reduction of two upper triangular submatrices is performed by Householder reflections. For details of the row merge scheme, the reader is referred to [12].

It is interesting to note that this idea of submatrix annihilation has been used implicitly in previous work in the literature. Reid [14] provides an efficient scheme to perform the QR decomposition of a banded system by Householder reductions. It may be interpreted as a wise choice of submatrix annihilation sequence, based on the structure of the band. Figure 2.2 provides an example.

In [11, Chapter 27], Lawson and Hanson consider an algorithm for the QR decomposition without requiring the entire matrix be in computer storage at one time. That again can be interpreted as a sequence of submatrix annihilations, and in this case the choice depends on the size of the matrix and the amount of available core storage.

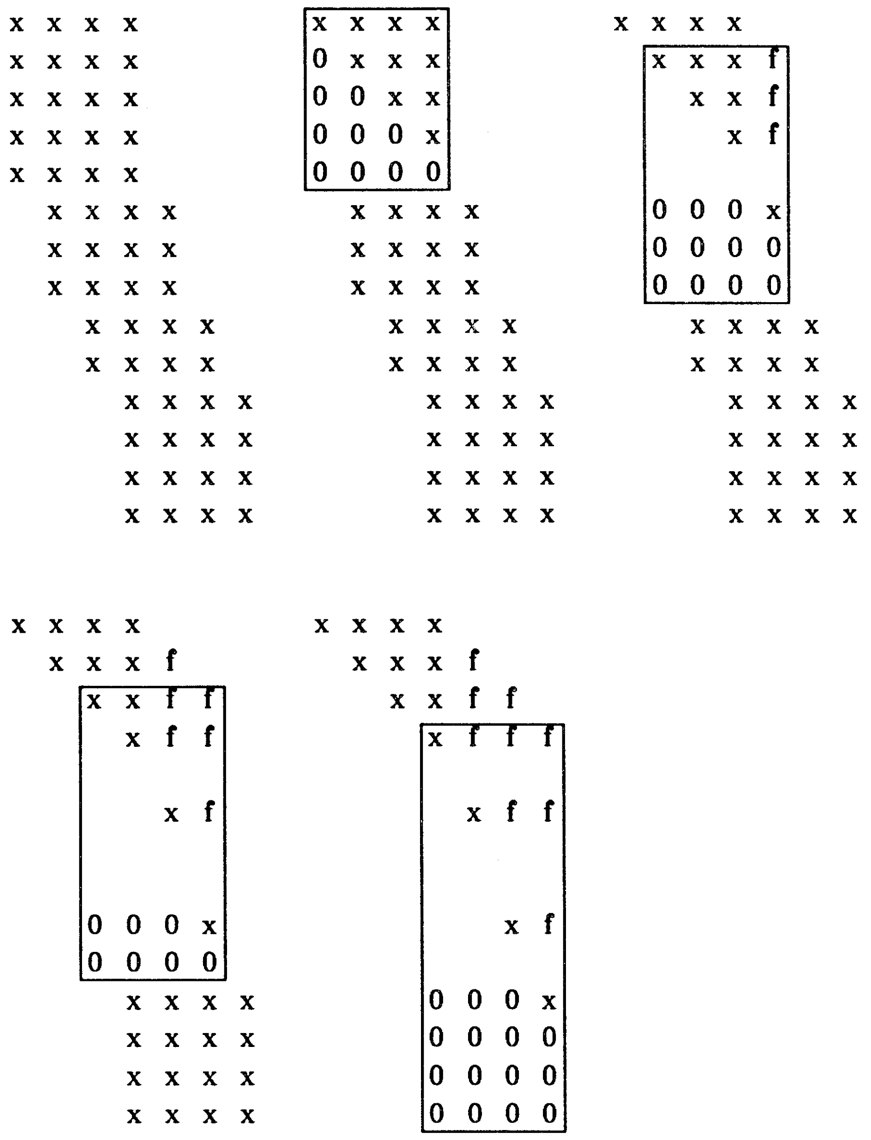


Figure 2.2: Selective submatrix annihilation for a band matrix.

3. The Algorithm

3.1. Row Merge Tree

The crucial factors in the proposed approach of submatrix annihilation are the choice of the submatrix sequence and the implementation (or data organization) of the annihilation process.

The notion of row merge trees is introduced in [12]. For an m by n matrix A, a *row merge tree* is defined to be a strictly binary tree with m leaves, each corresponding to a row in the matrix. It can be used as a basic structure to determine a submatrix sequence, where each submatrix corresponds to a (rooted) subtree. It has the desirable property of preserving zeros created in previous annihilations. Of course, different row merge trees induce different submatrix sequences.

The algorithm proposed here is basically the same as the one in [12], except that Householder reflections are used instead of Givens rotations. The defining row merge trees will be generated by the same heuristic algorithm as described in [12]. In view of the different nature of column reflections and row rotations, there are basic implementational or organizational differences in the process of "merging" or reduction of two upper triangular submatrices. They will be considered in the next subsection.

3.2. Row-oriented Version of Householder Transformation

Householder transformations have been described and implemented almost exclusively in the form of a sequence of inner products (LINPACK [2]). This is not suitable for our purpose of merging two upper trapezoidal matrices, since in our scheme, it is more appropriate to store an upper trapezoidal matrix row by row as shown in Figure 3.1.

```
a a a a a     a a a a a b b b b c c c
  b b b b
    c c c

Trapezoidal Matrix          Storage Array
```

Figure 3.1: Storage of an upper trapezoidal matrix.

In this section, we employ an observation in [10] to show how to re-organize the determination and application of Householder reflections in a row-oriented manner. This new scheme is ideally suited for our computation of reducing two upper trapezoidal matrices stored row by row. It should be noted that it is also suitable for general orthogonal decomposition by Householder transformations on *vector machines*.

Consider the following m by n matrix

$$A = \begin{pmatrix} d & v^T \\ u & E \end{pmatrix} ,$$

where u and v are $(m-1)-$ and $(n-1)-$ vectors respectively. Let σ be the 2-norm of the first column of A. That is, let

$$\sigma = \sqrt{(d^2+u^T u)}.$$

Assume that σ is nonzero. Then, a Householder reflection can be used to annihilate the vector u in the matrix A, where the *Householder vector* h is given by

$$h = \begin{pmatrix} \beta \\ w \end{pmatrix}$$

with
$$\beta = 1 + \frac{d}{\sigma_d},$$

$$w = \frac{u}{\sigma_d},$$

and $\sigma_d = sgn(d) * \sigma.$

Here $sgn(d)$ is a function whose value is $+1$ if d is non-negative, and -1 otherwise.

The Householder transformation is then given by

$$P = I - \frac{hh^T}{\beta} \quad .$$

It can be readily verified that

$$h^T \begin{pmatrix} d \\ u \end{pmatrix} = \beta \sigma_d .$$

Theorem 3.1:

$$P\,A = \begin{pmatrix} d' & v'^T \\ O & E' \end{pmatrix}$$

where $d' = -\sigma_d$

$$v'^T = -(\frac{d}{\sigma_d} v^T + w^T E)$$

$$E' = E - w(\beta v^T + w^T E)/\beta \quad .$$

Proof: Consider the application of the Householder transformation to the first column of A:

$$P \begin{pmatrix} d \\ u \end{pmatrix} = \begin{pmatrix} d \\ u \end{pmatrix} - \frac{hh^T}{\beta} \begin{pmatrix} d \\ u \end{pmatrix}$$

$$= \begin{pmatrix} d \\ u \end{pmatrix} - h\sigma_d = \begin{pmatrix} -\sigma_d \\ 0 \end{pmatrix} \quad .$$

On the other hand, applying P to the remaining columns of A, we have

$$P \begin{pmatrix} v^T \\ E \end{pmatrix} = \begin{pmatrix} v^T \\ E \end{pmatrix} - \frac{hh^T}{\beta} \begin{pmatrix} v^T \\ E \end{pmatrix}$$

$$= \begin{pmatrix} v^T \\ E \end{pmatrix} - \frac{h}{\beta} (\beta v^T + w^T E)$$

$$= \begin{pmatrix} v^T \\ E \end{pmatrix} - \frac{1}{\beta} \begin{pmatrix} \beta \\ w \end{pmatrix} (\beta v^T + w^T E)$$

$$= \begin{pmatrix} v^T - (\beta v^T + w^T E) \\ E - w(\beta v^T + w^T E)/\beta \end{pmatrix} \quad .$$

Hence, $E' = E - w(\beta v^T + w^T E)/\beta,$

and $v'^T = v^T - (\beta v^T + w^T E)$

$$= -(\frac{d}{\sigma_d} v^T + w^T E) \quad .$$

□

The results of Theorem 3.1 suggest the following version of Householder transformations that access the entries of the matrix in a row by row manner.

Algorithm: *Row-oriented Version of Householder Transformation*

Step 1: Computation of σ_d

$$\sigma = \sqrt{d^2+(u^T u)}$$

$$\sigma_d = sgn(d) * \sigma$$

Step 2: Computation of factor row (d', v'^T)

a) $d' = -\sigma_d$

b) Form $w = \frac{u}{\sigma_d}$

c) Form the linear combinations from rows of A (except first column)

$$v'^T = -(\frac{d}{\sigma_d},\ w^T) \begin{pmatrix} v^T \\ E \end{pmatrix}$$

Step 3: Determination of the "pivot row" by computing

$$(1+\frac{d}{\sigma_d}\ ,\ v^T - v'^T) = (\beta\ ,\ \beta v^T + w^T E)$$

Step 4: Gaussian elimination using pivot row from step 3

$$\begin{pmatrix} \beta & \beta v^T + w^T E \\ w & E \end{pmatrix} \rightarrow \begin{pmatrix} \beta & \beta v^T + w^T E \\ 0 & E' \end{pmatrix}$$

where $E' = E - w(\beta v^T + w^T E)/\beta$.

Here, we distinguish the *factor row* as the row in the resulting factor matrix R, from the *pivot row* as the row used in the elimination step. It should be pointed out that one temporary vector is required to store the computed pivot row in step 3.

It is interesting to note that this formulation of Householder transformations can be regarded as a special kind of Gaussian elimination, where the pivot row is computed from a linear combination of the rows of the matrix, rather than being taken directly from it. The numerical stability of the process can be seen from the fact that $|w_i| \leq 1 \leq \beta$ for every entry w_i of the vector w . As usual, care must be exercised in computing σ in order to avoid overflows. The standard method involving the scaling of $(d,\ u)^T$ was employed [2].

The reduction of two upper triangular or trapezoidal submatrices into another upper trapezoidal matrix can now be organized as a sequence of Householder column reflections, where the submatrices are stored in a row by row manner. The basic steps involved can be best illustrated by an example. The following are two such submatrices with column subscript sets {1,3,4,6,8} and {1,3,6,9} respectively.

The merging involves three steps, which are given below.

$$\begin{array}{ccccc} 1 & 3 & 4 & 6 & 8 \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \end{array} \qquad \begin{array}{cccc} 1 & 3 & 6 & 9 \\ x & x & x & x \\ & x & x & x \\ & & x & x \\ & & & x \end{array}$$

Step 1: Union of column subscripts:

$$\begin{array}{cccccc} 1 & 3 & 4 & 6 & 8 & 9 \end{array}$$

Step 2: Extension of the submatrices:

$$\begin{array}{cccccc} 1 & 3 & 4 & 6 & 8 & 9 \\ x & x & x & x & x & \\ & x & x & x & x & \\ & & x & x & x & \\ x & x & & x & & x \\ & x & & x & & x \\ & & & x & & x \\ & & & & & x \end{array}$$

Step 3: Sequence of Householder reflections:

$$\begin{array}{cccccc} \hline \multicolumn{1}{|c}{r} & r & r & r & r & \multicolumn{1}{c|}{r} \\ \hline & x & x & x & x & \\ & & x & x & x & \\ \hline \multicolumn{1}{|c}{0} & f & f & x & f & \multicolumn{1}{c|}{x} \\ \hline & x & & x & & x \\ & & & x & & x \\ & & & & & x \end{array} \qquad \begin{array}{cccccc} r & r & r & r & r & r \\ \cline{2-6} & \multicolumn{1}{|c}{r} & r & r & r & \multicolumn{1}{c|}{r} \\ \cline{2-6} & & x & x & x & \\ \cline{2-6} & \multicolumn{1}{|c}{0} & f & x & f & \multicolumn{1}{c|}{x} \\ & \multicolumn{1}{|c}{0} & f & x & f & \multicolumn{1}{c|}{x} \\ \cline{2-6} & & & x & & x \\ & & & & & x \end{array} \qquad \begin{array}{cccccc} r & r & r & r & r & r \\ & r & r & r & r & r \\ \cline{3-6} & & \multicolumn{1}{|c}{r} & r & r & \multicolumn{1}{c|}{r} \\ & & \multicolumn{1}{|c}{0} & x & f & \multicolumn{1}{c|}{x} \\ & & \multicolumn{1}{|c}{0} & x & f & \multicolumn{1}{c|}{x} \\ \cline{3-6} & & & x & & x \\ & & & & & x \end{array}$$

$$\begin{array}{cccccc} r & r & r & r & r & r \\ & r & r & r & r & r \\ & & r & r & r & r \\ \cline{4-6} & & & \multicolumn{1}{|c}{r} & r & \multicolumn{1}{c|}{r} \\ & & & \multicolumn{1}{|c}{0} & f & \multicolumn{1}{c|}{x} \\ & & & \multicolumn{1}{|c}{0} & f & \multicolumn{1}{c|}{x} \\ \cline{4-6} & & & & & x \end{array} \qquad \begin{array}{cccccc} r & r & r & r & r & r \\ & r & r & r & r & r \\ & & r & r & r & r \\ & & & r & r & r \\ \cline{5-6} & & & & \multicolumn{1}{|c}{r} & \multicolumn{1}{c|}{r} \\ & & & & \multicolumn{1}{|c}{0} & \multicolumn{1}{c|}{x} \\ \cline{5-6} & & & & & x \end{array} \qquad \begin{array}{cccccc} r & r & r & r & r & r \\ & r & r & r & r & r \\ & & r & r & r & r \\ & & & r & r & r \\ & & & & r & r \\ \cline{6-6} & & & & & \multicolumn{1}{|c|}{r} \\ & & & & & \multicolumn{1}{|c|}{0} \\ \cline{6-6} \end{array}$$

3.3. Preliminary Submatrix Reduction

It is well known that the reduction of a matrix with two rows to upper trapezoidal by a Householder reflection is computationally equivalent to the reduction by a Givens rotation [7]. There is, from a practical point of view, no advantage of Householder over Givens in such situations.

On the other hand, Householder reflections will be comparatively more effective if the matrix to be reduced has many rows. In this case, one reflection can be used to reduce all nonzeros under the diagonal in the first column as opposed to the use of many rotations. In view of this, the row merging sequence given in [12] is modified so that the algorithm will accumulate as many rows as possible before reduction by Householder reflections is performed. The criterion used is that the algorithm will take the next incoming row if it does not enlarge the subscript set.

In Figure 3.2, a row merge tree is specified for the given 8 by 5 matrix example. This means that seven reductions of submatrices are to be performed. Since the first four rows have the same set of nonzero subscripts, it will be more advantageous to reduce them together by Householder transformations. The same applies to the last four rows. The row merge tree structure can hence be depicted in Figure 3.3. Note that if one is using Givens rotations, the two row merge structures are equivalent from a computational point of view.

$$\begin{array}{cccccc}
r1 & x & & & x & x \\
r2 & x & & & x & x \\
r3 & x & & & x & x \\
r4 & x & & & x & x \\
r5 & x & x & x & & \\
r6 & x & x & x & & \\
r7 & x & x & x & & \\
r8 & x & x & x & &
\end{array}$$

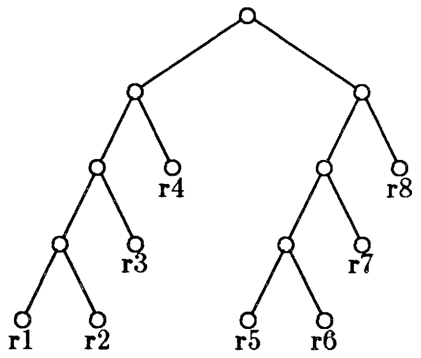


Figure 3.2: A matrix and its row merge tree

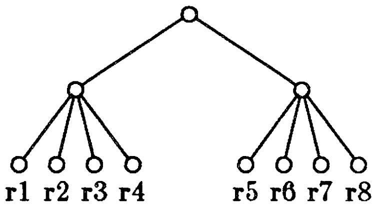


Figure 3.3: Modified row merge tree

4. Experimental Results and Concluding Remarks

In this section, experimental results for sparse orthogonal decomposition using Householder reflections and Givens rotations are provided. The times reported in the tables are in seconds on a VAX 11/780 having floating point hardware. Only multiplicative operations are accounted for in the operation counts. The method labelled "Preproc Hholder" in the tables refers to the one with the preliminary submatrix reduction as described in Section 3.3.

Our experiments involved two sets of test problems which display a considerable variety of structures. For test set #1, the matrix values were obtained directly from the application, while for test set #2, the numerical values of the nonzeros in the matrices were uniform random numbers from [-1,1]. In all cases, the solution was arranged to be a vector of all ones by setting the right hand side equal to the sum of the columns of the matrix, computed in double precision.

The first test set consists of the ten problems used by George, Heath and Ng in their comparison paper on methods for solving sparse linear least squares systems [6]. Readers are referred to it for details about the problems. We list them in Table 4.1 for reference.

	---number of ---			
Problem	Rows	Cols	Nonz	Problem Description
1	313	176	1557	Sudan survey data
2	1033	320	4732	Analysis of gravity-meter observations (well-conditioned)
3	1033	320	4719	Analysis of gravity-meter observations (ill-conditioned)
4	1850	712	8755	Similar to Problem 2, but larger
5	1850	712	8638	Similar to Problem 4, but larger
6	784	225	3136	15x15 grid problem
7	1444	400	5776	20x20 grid problem
8	1512	402	7152	3x3 geodetic network with 2 observations per node
9	1488	784	7040	4x4 geodetic network with 1 observation per node
10	900	269	4208	Geodetic network problem provided by U.S. National Geodetic Survey (ill-conditioned)

Table 4.1: Matrix Problems for Test Set #1

Various performance results of the different methods applied to test set #1 are tabulated in Tables 4.2 and 4.3. The column ordering used was that provided by the minimum degree algorithm, as suggested in [5]. A modified form of the minimum degree algorithm due to Liu [13] was actually used in the experiments.

	Factorization Opcount			Factorization Time		
Problem	Givens	Hholder	Preproc Hholder	Givens	Hholder	Preproc Hholder
1	51732	51306	50556	2.13	2.39	2.42
2	141744	149068	121936	6.68	7.16	6.31
3	143764	149049	121778	6.36	7.19	6.73
4	472198	440872	398964	14.27	16.55	14.99
5	477920	445960	404826	14.17	16.17	15.42
6	138320	120002	109066	5.17	5.58	4.64
7	357444	285182	262640	10.71	12.42	9.36
8	330884	333970	249058	10.76	14.67	9.38
9	382936	365286	315420	11.84	14.98	11.12
10	561156	465598	455772	11.05	13.33	11.48

Table 4.2: Comparison of Factorization Operation Counts and Time for the Different Schemes Applied to the Problems in Test Set #1.

	Relative Error			Residual		
Problem	Givens	Hholder	Preproc Hholder	Givens	Hholder	Preproc Hholder
1	2.86E-6	1.67E-6	1.19E-6	4.90E-6	5.61E-6	5.51E-6
2	0.89E-5	0.68E-5	1.43E-5	2.99E-6	3.13E-6	3.30E-6
3	0.35E-3	0.27E-3	1.01E-3	3.21E-6	3.73E-6	3.04E-6
4	3.81E-6	3.52E-6	1.79E-6	4.43E-6	4.95E-6	4.36E-6
5	1.04E-4	0.99E-4	0.84E-4	4.78E-6	5.13E-6	4.58E-6
6	1.25E-6	1.19E-6	0.83E-6	7.07E-6	7.43E-6	7.48E-6
7	1.31E-6	1.25E-6	1.43E-6	1.03E-5	1.09E-5	1.10E-5
8	1.79E-6	1.79E-6	1.19E-6	1.48E-5	1.56E-5	1.23E-5
9	2.15E-6	3.46E-6	1.67E-6	1.20E-5	1.37E-5	1.14E-5
10	3.87E-3	8.18E-3	5.64E-3	0.76E 0	1.01E 0	0.95E 0

Table 4.3: Comparison of Numerical Results for the Different Methods Applied to the Problems in Test Set #1

The second set of test problems are typical of those that arise in the ***natural factor formulation*** of finite element methods [1]. Consider the k by k regular grid with $(k-1)^2$ small squares. Associated with each of the k^2 grid nodes is a variable, and associated with each square is a set of ***four*** equations involving the four variables at the corners of the square. The assembly of these equations results in a large sparse overdetermined system of equations

$$Ax = b$$

where the matrix A has k^2 columns and $4(k-1)^2$ rows. The columns of the matrix were ordered by the minimum degree scheme as in [5]. The different schemes were tested on values of k=10, 20, 30, 40, and 50, and the results are tabulated in Tables 4.4 and 4.5.

	Factorization Opcount			Factorization Time		
k	Givens	Hholder	Preproc Hholder	Givens	Hholder	Preproc Hholder
10	38624	37036	33378	1.93	2.08	1.80
20	357436	285182	262640	10.62	11.08	9.34
30	1177632	863562	810704	28.51	28.57	24.64
40	2897088	1987730	1890948	57.57	57.35	49.83
50	5692656	3742930	3591612	100.48	100.28	87.02

Table 4.4: Comparison of Factorization Operation Counts and Time for the Different Schemes Applied to the Problems of Test Set #2

	Relative Error		
k	Givens	Hholder	Preproc Hholder
10	1.07E-6	1.19E-6	0.72E-6
20	1.67E-6	1.19E-6	1.07E-6
30	1.43E-6	1.07E-6	1.07E-6
40	2.26E-6	1.25E-6	1.79E-6
50	2.03E-6	1.67E-6	1.31E-6

Table 4.5: Comparison of Numerical Results for the Different Schemes Applied to the Problems of Test Set #2

The results of the numerical experiments demonstrate that the use of Householder reflections in sparse matrix decomposition can be organized so that they are very competitive with Givens rotations. For the variety of problems in test set #1, their overall performance is at least as good as that of Givens. The factorization operation count is always smaller, while the actual CPU time is always comparable.

For test set #2, there is a more substantial reduction in the operation count. Indeed, for $k=50$, the operation count was reduced by a factor of more than one third. It should be noted that the reduction in arithmetic is not reflected in a proportional decrease in execution time. This is probably because of larger computational overhead incurred in the Householder version. Nevertheless, for $k=50$, the preprocess version of the new scheme yields a reduction in execution time of nearly 25 percent.

In summary, the experiments do confirm that the widely-held view that Householder transformations are inappropriate for sparse orthogonal decomposition should be re-assessed. In particular, we have shown here that the preprocess version is competitive and often superior to the use of Givens rotations.

It would appear that the row-oriented version of Householder reflections described in this paper is readily adaptable to vectorization. We are currently attempting to conduct such experiments on a Cray computer, and we will report the results of those experiments when they have been completed.

References

[1] J.H. ARGYRIS AND O.E. BRONLUND, "The natural factor formulation of the stiffness matrix displacement method", *Comput. Meth. Appl. Mech. Engrg.*, **5** (1975), pp. 97-119.

[2] J.J. DONGARRA, C.B. MOLER, J.R. BUNCH, AND G.W. STEWART, *LINPACK users' guide*, SIAM, Philadelphia (1980).

[3] I.S. DUFF AND J.K. REID, "A comparison of some methods for the solution of sparse overdetermined systems of linear equations", *J. Inst. Maths. Appl.*, **17** (1976), pp. 267-280.

[4] W.M. GENTLEMAN, "Row elimination for solving sparse linear systems and least squares problems", in *Proc. 1975 Dundee Conference on Numerical Analysis*, ed. G.A. Watson, Lecture Notes in Mathematics (506), Springer-Verlag (1975), pp. 122-133.

[5] J.A. GEORGE AND M.T. HEATH, "Solution of sparse linear least squares problems using Givens rotations", *Linear Algebra and its Appl.*, **34** (1980), pp. 69-83.

[6] J.A. GEORGE, M.T. HEATH, AND E.G.Y. NG, "A comparison of some methods for solving sparse linear least squares problems", *SIAM J. Sci. Stat. Comput.*, **4** (1983), pp. 177-187.

[7] P.E. GILL, G.H. GOLUB, W. MURRAY, AND M.A. SAUNDERS, "Methods for modifying matrix factorizations", *Math. Comp.*, **28** (1974), pp. 505-535.

[8] P.E. GILL AND W. MURRAY, "The orthogonal factorization of a large sparse matrix", in *Sparse matrix computations*, ed. J.R. Bunch and D.J. Rose, Academic Press, New York (1976), pp. 201-212.

[9] M.T. HEATH, "Numerical methods for large sparse linear least squares problems", *SIAM J. Sci. Stat. Comput.*, **26** (1984), pp. 497-513.

[10] J. JOHNSSON, "A computational array for the *QR* method", in *1982 Conference on Advanced Research in VLSI*, , MIT Press (1982), pp. 123-129.

[11] C.L. LAWSON AND R.J. HANSON, *Solving least squares problems*, Prentice-Hall Inc., Englewood Cliffs, N.J. (1974).

[12] J.W.H. LIU, "On general row merging schemes for sparse Givens transformations", Technical Report No. 83-04, Department of Computer Science, York University, Downsview, Ontario (1983).

[13] J.W.H. LIU, "On multiple elimination in the minimum degree algorithm", Technical Report No. 83-03, Department of Computer Science, York University, Downsview, Ontario (1983).

[14] J.K. REID, "A note on the least squares solution of a band system of linear equations by Householder reductions", *Comput. J.*, **10** (1967), pp. 188-189.