Compact Structural Representation of Sparse Cholesky, QR and LU Factors*

Alan George†
Joseph Liu††

CS-85-41

October 1985

* 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.
† Department of Computer Science, University of Waterloo, Waterloo, Ontario
†† Department of Computer Science, York University, Downsview, Ontario

Compact Structural Representation of Sparse Cholesky, QR and LU Factors

Alan George

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

Joseph W.H. Liu

Department of Computer Science
York University
Downsview, Ontario, Canada

*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.

ABSTRACT

The computer storage used for a sparse matrix involves two components: *primary storage* and *overhead storage.* Primary storage refers to that which is used for the actual numerical values, while overhead storage refers to that which is used to represent the *structure* of the matrix. That is, the latter information provides the locations in the matrix where the numbers contained in the primary storage belong, along with a mechanism for accessing those numbers. Ideally, the overhead storage should be small, and the accessing mechanism should be rapid and convenient to use. In this paper, we describe some recent developments which allow very efficient storage schemes for the factors of sparse matrices obtained via Gaussian elimination and orthogonal factorization. In particular, the overhead storage required is bounded by the number of nonzeros in the *original matrix*, irrespective of the amount of fill-in that occurs during the factorization. The algorithms considered include Cholesky's method, Gaussian elimination with partial pivoting, and orthogonal factorization via Householder transformations of non-rectangular matrices.

Table of Contents

1. Introduction

In this article we consider the problem of efficiently storing the factors obtained in connection with solving large sparse systems of linear equations. We will denote the coefficient matrix of the system by A, and depending on the context, it may or may not be square, symmetric, or positive definite. Normally, when A is decomposed (factored) using Gaussian elimination or orthogonal reduction, its factors suffer *fill-in*; that is, they normally have some nonzeros in positions that are zero in A. Nevertheless, if the rows and columns of A are appropriately ordered, its factors are usually sparse, and it is therefore highly desirable to exploit sparsity in storing those factors.

The computer storage used for a sparse matrix involves two components: *primary storage* and *overhead storage*. Primary storage refers to storage used for the actual numerical values, while overhead storage refers to that used to represent the *structure* of the matrix. That is, the latter information provides the locations in the matrix where the numbers contained in the primary storage belong, along with a mechanism for accessing those numbers. Ideally, the overhead storage should be small, and the accessing mechanism should be rapid and convenient to use.

In this paper, we describe some recent developments which provide very efficient storage schemes for the factors of sparse matrices obtained via Gaussian elimination and orthogonal factorization. In particular, the amount of overhead storage required is bounded by the number of nonzeros in the matrix A, and is therefore independent of the number of nonzeros in the factors themselves. Our objective here is to describe the key ideas that allow for efficient structural representation. Details about the actual data structures and efficient access to elements in them can be found in the references.

An outline of the paper is as follows. In section 2 we review some basic ideas dealing with sparse Cholesky factorization, and introduce the notion of an *elimination tree* which is a key element in the development of the storage schemes in this paper. Section 2 also describes a method for storing the (sparse) Cholesky factor L of a symmetric positive definite matrix A using overhead storage bounded by the number of nonzeros in A. Section 3 deals with the problem of efficiently storing the orthogonal factor Q associated with the QR factorization of a general rectangular matrix A. In particular, it is shown that if one knows the structure of R, then Q can be stored using only $m+n$ overhead elements, where m and n are respectively the number of rows and columns in the matrix A. Note that this is independent of the number of nonzeros in either Q or R.

In section 4, results of section 2 are used to show that the matrix R associated with the QR factorization of A can be stored using overhead that is bounded by the number of nonzeros in A. This is different from the results of section 2, since R^T is (mathematically) the Cholesky factor of A^TA rather than A. We believe that the results of section 4 are new. The results in sections 3 and 4 together imply that the structure of the QR factorization of a sparse matrix A can be represented using storage that is bounded by the number of nonzeros in A; that is, independent of the number of nonzeros in Q and R.

Section 5 deals with the problem of representing the structure of the triangular factors of a general n by n sparse nonsingular matrix A. The representation is done in the context of a new partial pivoting implementation of Gaussian elimination in which the data structure used is large enough to accommodate all possible interchanges that might occur during the factorization. It is shown that the results of sections 3 and 4 are applicable.

Section 6 contains our concluding remarks.

2. Cholesky Factorization and Elimination Trees

In this section we assume that the sparse matrix A is symmetric and positive definite, and we denote its Cholesky factorization by LL^T. We implicitly assume that the rows and columns of A have been permuted so that its Cholesky factor L is sparse.

Consider the structure of the matrix L. For each column $j \leq n$, define $\gamma[j]$ by

$$\gamma[j] \;=\; \min \left\{ i \;\middle|\; L_{ij} \neq 0 \right\} \quad ;$$

that is, $\gamma[j]$ is the row subscript of the first off-diagonal nonzero in column j of L. If column j has no off-diagonal nonzero, we set $\gamma[j]=j$. (Hence $\gamma[n]=n$.) We shall use the quantities $\gamma[1]$, $\gamma[2]$, $\cdots$, $\gamma[n]$ to characterize a structural representation of sparse Cholesky factors and Householder transformations.

We now introduce the notion of an *elimination tree* corresponding to the structure of the Cholesky factor L. The tree has n nodes, labelled from 1 to n. For each j, if $\gamma[j]>j$, then node $\gamma[j]$ is the *parent* of node j in the elimination tree, and node j is one of possibly several *child* nodes of node $\gamma[j]$. We assume that the matrix A is *irreducible*, so that n is the only node with $\gamma[n]=n$ and it is the *root* of the tree. Thus, for $1 \leq k < n$, $\gamma[k] > k$. (If A is reducible, then the elimination tree defined above is actually a forest which consists of several trees.) There is exactly one *path*

from each node to the root of the tree. If node i lies on the path from node j to the root, then node i is an *ancestor* of node j, and node j is a *descendant* of node i.

An example to illustrate the notion of elimination trees is provided by the 6 by 6 matrix shown in Fig. 1. The structure of the Cholesky factor of the matrix in Fig. 1 is shown in Fig. 2, and the elimination tree associated with L is illustrated in Fig. 3. Elimination trees have been used either implicitly or explicitly in numerous articles dealing with sparse symmetric factorization [3,4,5,6,7,11,12,15,16].

$$A = \begin{pmatrix} x & & x & & x & \\ & x & & & x & x \\ x & & x & x & x & \\ & & x & x & & x \\ x & x & x & & x & \\ & x & & x & & x \end{pmatrix} .$$

Fig. 1: A sparse matrix A.

$$L = \begin{pmatrix} x & & & & & \\ & x & & & & \\ x & & x & & & \\ & & x & x & & \\ x & x & x & x & x & \\ & x & & x & x & x \end{pmatrix} .$$

Fig. 2: Structure of the Cholesky factor of the matrix in Fig. 1.

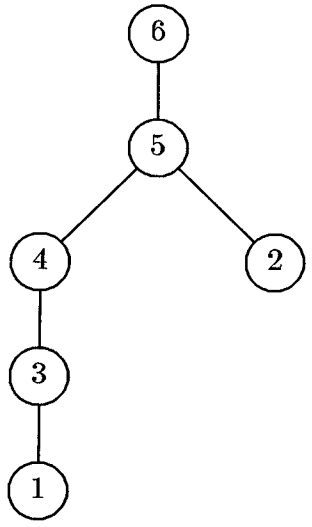


Fig. 3: The elimination tree associated with the matrix in Fig. 2.

The elimination tree has simple structure and can be economically represented using γ, as shown in Fig. 4.

i	1	2	3	4	5	6
$\gamma[i]$	3	5	4	5	6	6

Fig. 4: Computer representation of the tree of Fig. 3.

Thus, the representation requires only a single vector of size n.

By the *subtree rooted at* node i we mean the subgraph consisting of node i together with *all* its descendants in the tree. A *pruned subtree* at node i is a subtree rooted at node i where some of its own subtrees have been pruned or removed. Each pruned subtree is characterized uniquely by its root and its set of *leaf nodes*. The results that follow in this section are aimed at identifying such a pruned subtree that can be used to represent the structure of row i of L.

We begin with a basic Lemma due to Parter [14].

Lemma 2.1 Let $i > j$. Then $L_{ij} \neq 0$ if and only if at least one of the following conditions hold:

a) $A_{ij} \neq 0$

b) For some $k < j$, $L_{ik} \neq 0$ and $L_{jk} \neq 0$.

□

In words, the above Lemma states that L_{ij} is nonzero as a result of one *or both* of the following: a) the corresponding element of A is already nonzero, or b) some step of the factorization previous to step j caused fill-in to occur in position (i,j).

Note that if L_{ik} and L_{jk} are nonzero, then L_{ij} will be nonzero *irrespective of whether* A_{ij} *is or is not zero.* In this case, A_{ij} can be set to zero without changing the *structure* of the corresponding Cholesky factor L. We shall refer to the matrix A^{-}, derived from A by setting as many of its off-diagonal elements as possible to zero while at the same time preserving the structure of its Cholesky factor, as the *skeleton matrix* of A. We return to this notion later in this section.

The following is a small but important modification to Lemma 2.1.

Lemma 2.2: Let $i > j$. Then $L_{ij} \neq 0$ if and only if at least one of the following conditions holds:

a) $A_{ij} \neq 0$

b) There exists a k such that $j = \gamma[k]$ and $L_{ik} \neq 0$.

Proof: Note that from the definition of γ, $j = \gamma[k]$ implies that $k < j$ and $L_{jk} \neq 0$. Thus, most of the proof is a direct consequence of Lemma 2.1. The only part that must be confirmed is that when b) in Lemma 2.1 holds, k can be *chosen* such that $j = \gamma[k]$.

Let k be the *maximum* of those for which $L_{ik} \neq 0$ and $L_{jk} \neq 0$, and suppose $\gamma[k] \neq j$. This implies that there is a $t = \gamma[k]$ satisfying $k < t < j$ such that $L_{tk} \neq 0$. But since $L_{jk} \neq 0$ and $L_{ik} \neq 0$, by Lemma 2.1, this implies that $L_{jt} \neq 0$ and $L_{it} \neq 0$, contradicting the assumption that k was the largest numbered column having nonzeros in rows i and j.

□

The b) part of Lemma 2.2 can be regarded as a mechanism by which we can proceed *down* the elimination tree, beginning at a node j for which $L_{ij} \neq 0$. Specifically, given $L_{ij} \neq 0$, it says that either $A_{ij} \neq 0$ or node j has a child node $k < j$ for which $L_{ik} \neq 0$.

Note that in applying Lemma 2.2, *both* a) *and* b) may hold. Note also that when b) fails to hold, there may still exist a k for which $j = \gamma[k]$. When no such k exists, node j is a leaf node of the elimination tree.

The b) part of Lemma 2.2 can also be regarded as a mechanism by which we can proceed *up* the elimination tree, beginning at a node k for which $L_{ik} \neq 0$ and $j = \gamma[k]$.

Theorem 2.3: Let $i > k$ and $L_{ik} \neq 0$. Then node i is an ancestor of node k in the elimination tree.

Proof: If $i = \gamma[k]$, there is nothing to prove. Otherwise, $\gamma[k] < i$ and by repeated application of b) in Lemma 2.2 we generate an ascending sequence of nodes (subscripts) bounded above by i:

$$k < \gamma[k] < \gamma[\gamma[k]] < \gamma^3[k] < \cdots < i.$$

Thus, there exists an integer p such that $\gamma^p[k] = i$.

□

Theorem 2.3 implies that node i is an ancestor of all nodes k for which $L_{ik} \neq 0$. Thus, in considering the structure of row i, it is sufficient to consider the subtree of the elimination tree rooted at node i. The following Theorem shows that the subscripts of *all* nonzeros in row i of L can be generated by using paths in this subtree that begin at nodes k for which $A_{ik} \neq 0$.

Theorem 2.4: Let $i > j$ and $L_{ij} \neq 0$. Then node j is an ancestor of some node k for which $A_{ik} \neq 0$.

Proof: If $j = k$, there is nothing to prove. Otherwise, we repeatedly apply Lemma 2.2, generating a *decreasing* sequence of nodes (column subscripts) as long as condition a) fails to be satisfied. But this sequence is necessarily bounded below, so it must terminate at some node k for which $A_{ik} \neq 0$.

□

Thus, the structure of row i is completely specified by the paths in the subtree which originate at nodes k for which $A_{ik} \neq 0$ and terminate at the root i. The subtree defined

by these paths will be referred to as the *i-th row subtree.*

Note that not all nodes k for which $A_{ik} \neq 0$ will be leaves of this subtree. Consider distinct nodes j and k where $A_{ij} \neq 0$, $A_{ik} \neq 0$, and node j is an ancestor of node k in the elimination tree. Obviously the path from node j to node i is a proper subpath of that from node k to node i, so node j is not a leaf of the i-th row subtree. This observation leads immediately to the following Corollary, which specifies which nodes are leaf nodes of the i-th row subtree.

Corollary 2.5: Node j is a leaf node of the i-th row subtree if and only if $A_{id}=0$ for all descendant nodes d of node j.

□

Thus, we can associate a pruned subtree of the elimination tree with the structure of each row of the factor L. We illustrate this using the matrix example of Fig. 1. In Fig. 5 , we provide the row subtrees for each row of L shown in Fig. 2. The reader will want to consult the corresponding elimination tree shown in Fig. 3.

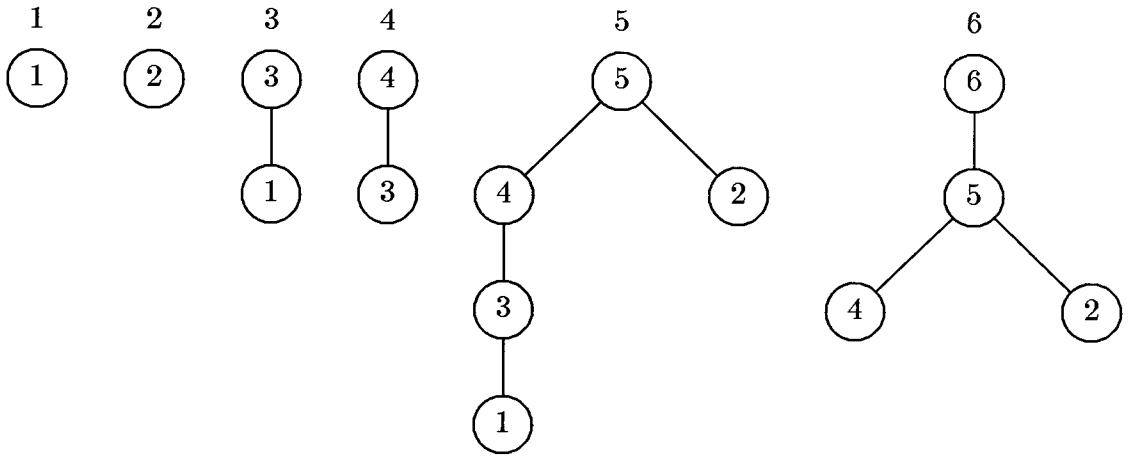


Fig. 5: Row subtrees associated with the matrix in Fig. 2.

Note that the discussion following Theorem 2.4 provides the key to identifying the elements of A that can be removed without affecting the structure of the Cholesky factor of the matrix. Specifically, any element A_{ij} that does not correspond to a leaf node of row i can be removed, since its position will be filled in as a consequence of being on a path from some leaf node of row i to node i. On the other hand, no leaf

node of row i can be removed without affecting its structure.

Thus, the *skeleton matrix* A^{-} of A is defined by the set of leaf nodes in each of the row subtrees. For $i > k$, $A^{-}_{ik} = A_{ik}$ if node k is a leaf node of the pruned subtree at node i; otherwise, it is zero. For the diagonal entries, $A^{-}_{ii} = A_{ii}$. As mentioned above, the skeleton matrix A^{-} is the minimal matrix structure which has the same Cholesky structure as the matrix A.

Using the skeleton matrix of A, it is possible to set up a very efficient scheme to represent the structure of L in an *implicit* form by storing the elimination tree using γ, along with the set of leaf nodes for each row subtree. A vector *LEAF* can be used to store the lists of leaves for each row subtree, along with an index vector *XLEAF* containing the positions in *LEAF* where each row list begins. Note that the storage needed for the vector *LEAF* is bounded by the number of nonzeros in the lower triangular part of A, and is usually much less than that quantity.

Thus, altogether, the overhead storage required to represent the structure of L using this approach is $3n+2$ (for γ and *XLEAF*) plus the storage required for *LEAF*. For important details and enhancements, the reader should consult [13].

3. A Structural Representation of Sparse Householder Vectors

In this section we describe the use of elimination trees in connection with a very efficient scheme for representing the structure of the (factored form of the) orthogonal matrix Q obtained during the reduction of a general rectangular matrix to upper triangular form using Householder reflections. We shall assume that the given m by n ($m \geq n$) sparse matrix A has full column rank. Hence A^TA is symmetric and positive definite.

Consider the orthogonal decomposition of A into $Q\begin{bmatrix} R \\ O \end{bmatrix}$, where Q is orthogonal and R is upper triangular. It is well known that R^T is mathematically the same as the Cholesky factor L of A^TA (apart from possible sign differences in some rows). However, it is also well known that the *structure* of the Cholesky factor obtained by a *symbolic* factorization of the structure of A^TA may overestimate that of R. In [1], Coleman et. al. have shown that if the matrix A has the *strong Hall property*, then the symbolic Cholesky factorization of A^TA will correctly provide the structure of R. For the purpose of this paper, we shall assume that the matrix A has this property so that

we can refer to the structure of the Cholesky factor of A^TA and the structure of the factor matrix R^T interchangeably. This assumption is reasonable since if the matrix A does not have this property, it can be permuted to block upper triangular form where each diagonal block submatrix has the strong Hall property.

We define γ exactly as we did in the previous section, except it is done in terms of $R=L^T$ rather than L. For each row $i \leq n$, define $\gamma[i]$ by

$$\gamma[i] \;=\; \min \{\, j \mid R_{ij} \neq 0 \,\} \quad ;$$

that is, $\gamma[i]$ is the column subscript of the first off-diagonal nonzero in row i of R. If row i does not have any off-diagonal nonzero, we set $\gamma[i]=i$. (Hence $\gamma[n]=n$.)

We briefly review the use of Householder transformations in the orthogonal decomposition of a matrix in order to establish the necessary notation. Consider the m by n matrix

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

where u and v are $(m-1)$- and $(n-1)$-vectors respectively.

For appropriately chosen scalar β and $(m-1)$-vector w, the Householder transformation can be expressed as

$$P \;=\; I - \frac{1}{\beta} \begin{pmatrix} \beta \\ w \end{pmatrix} \begin{pmatrix} \beta \\ w \end{pmatrix}^T \quad .$$

The vector $\begin{pmatrix} \beta \\ w \end{pmatrix}$ is referred to as the *Householder vector.*

After the transformation, the matrix becomes

$$PA \;=\; \begin{pmatrix} -\sigma_d & v^T - y^T \\ 0 & E - wy^T/\beta \end{pmatrix} \quad ,$$

where $y^T=\beta v^T+w^TE$; that is, y^T is a linear combination of v^T and the rows from E. The vector y plays an important role in terms of structural modifications in the vector v^T and the submatrix E.

Thus, the matrix A can be reduced to upper triangular form by a sequence of Householder transformations

$$P_1 , P_2 , \cdots , P_n$$

defined by the corresponding Householder vectors

$$\begin{pmatrix} \beta_1 \\ w_1 \end{pmatrix}, \begin{pmatrix} \beta_2 \\ w_2 \end{pmatrix}, \cdots , \begin{pmatrix} \beta_n \\ w_n \end{pmatrix} \quad ,$$

where β_i is a scalar, and w_i is a vector of size $m-i$.

Since we are interested in the structure of this set of Householder vectors when the given matrix A is sparse, we define the following m by n *Householder matrix* H to be

$$H = \begin{pmatrix} \beta_1 & 0 & 0 & & 0 \\ & \beta_2 & 0 & & 0 \\ w_1 & & \beta_3 & & 0 \\ & w_2 & & \cdot\ \cdot & \cdot \\ & & w_3 & \cdot\ \cdot & \beta_n \\ & & & \cdot\ \cdot & w_n \end{pmatrix} \quad .$$

In what follows, we shall relate the structure of the matrix H to that of the upper triangular factor matrix R. We shall also assume that the diagonal elements of A are nonzero. This assumption is reasonable since the rows of a sparse matrix with full rank can be reordered so that the diagonal elements of the permuted matrix are nonzero [2].

We now examine more carefully the effect of applying the Householder transformation P to A. As usual, we assume throughout that exact cancellations do not occur. Recall that $\gamma[i]$ denotes the column subscript of the first off-diagonal nonzero in row i of R.

Observation 1: The first row of the factor matrix R is given by

$$R_{1*} = (-\sigma_d \ , \ v^T - y^T) \quad .$$

Observation 2: The *structure* of R_{1*} is the *union* of the row structures of A_{i*} where $A_{i1} \neq 0$, assuming A has a zero-free diagonal.

Observation 3: $R_{1,\gamma[1]} \neq 0$, and $R_{1,j} = 0$ for $1 < j < \gamma[1]$.

Observation 4: If $A_{i1} \neq 0$, then

$$(PA)_{i,\gamma[1]} \neq 0$$

and

$$(PA)_{i,j} = 0 \ , \qquad \text{for } 1 < j < \gamma[1] \quad .$$

In fact, the structure of $(PA)_{i*}$ is the same as that of R_{1*}. Furthermore, the diagonal of PA remains zero-free.

Observation 4 implies that if $A_{i1} \neq 0$, then after the Householder transformation is applied to A the next nonzero entry in row i of PA appears in column $\gamma[1]$. Since the algorithm is to be applied recursively to the submatrix $E - wy^T/\beta$, observation 4 can be used repeatedly and the next theorem then follows.

Theorem 3.1: If f_i is the column subscript of the first nonzero in row A_{i*}, then the locations of the nonzeros in row H_{i*} are given by

$$f_i \ , \ \gamma[f_i] \ , \ \gamma[\gamma[f_i]] \ , \ \cdots \ , \ t \quad ,$$

where t is defined as follows. If $i \leq n$, then $t = i$; otherwise, t is some column subscript with $\gamma[t] = t$.

□

Theorem 3.1 says that the structure of each row of the Householder matrix H corresponds to a *chain* in the elimination tree of R. Since the structure of this tree is already given implicitly in the row structure of R (by the γ's), it is sufficient to know for each row of H where the associated chain starts in the tree (that is, the f's). This can be viewed as a special case of the row structures in the previous section, which in general correspond to subtrees of the elimination tree.

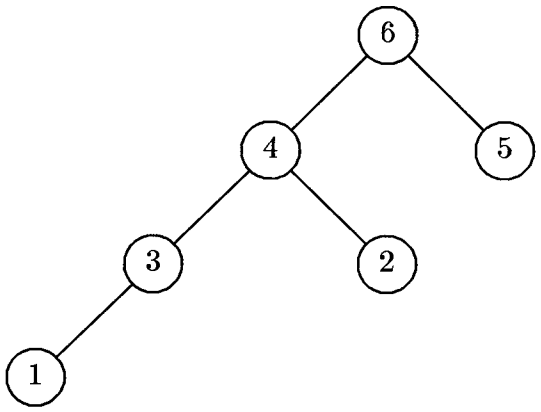
We conclude this section by providing an example to illustrate Theorem 3.1 above.

$$A = \begin{pmatrix} x & & & & & x \\ & x & & x & & \\ x & & x & & & \\ & x & & x & & x \\ & & & & x & x \\ x & & & & & x \\ & & x & x & & \\ & x & & x & & \end{pmatrix} .$$

The structures of the corresponding Householder matrix H and triangular factor R are given by

$$H = \begin{pmatrix} x & & & & & \\ & x & & & & \\ x & & x & & & \\ & x & & x & & \\ & & & & x & \\ x & & x & x & & x \\ & & x & x & & x \\ & x & & x & & x \end{pmatrix} \qquad \begin{pmatrix} R \\ O \end{pmatrix} = \begin{pmatrix} x & & x & & & x \\ & x & & x & & x \\ & & x & x & & x \\ & & & x & & x \\ & & & & x & x \\ & & & & & x \\ & & & & & \\ & & & & & \end{pmatrix} .$$

The elimination tree associated with R is given below.



As an example, note that the column subscripts of the nonzeros in row 6 of H are 1, 3, 4, 6, which is a chain in the elimination tree.

Thus, the structure of the rows of H can be represented implicitly using only γ and a single array of length m containing the f_i's. Since this representation is necessarily row oriented, in order to use it effectively, the computation must be

correspondingly row oriented. Such a version of Householder transformations has been described in [8]. For more details about an associated data structure and implementation, the reader is referred to [9].

4. Representation of the Cholesky factor R of A^TA.

In this section, we show how the structure of the sparse factor R can be represented in storage bounded by the number of nonzeros in the original matrix A. The scheme is based on the structural scheme for the representation of Cholesky factors using the *skeleton matrix* introduced in Section 2. Indeed, the result is to bound the number of nonzeros in the skeleton matrix of M $(= A^TA)$ by the number of nonzeros in A. This will give an efficient structural representation of the factor matrix R.

Consider the structure of the Cholesky factor R of M. Without loss of generality, we assume that the matrix M is irreducible. This implies that each row of R has an off-diagonal nonzero except for the last row. As before, let $\overline{M}$ denote the *skeleton matrix* of M.

In Section 2 it was shown that the structure of the Cholesky factor R can be represented implicitly using the structure of the skeleton matrix $\overline{M}$. The representation requires storage overhead bounded by the number of nonzeros in $\overline{M}$. Moreover, the explicit column structure (subscript sequence) can be generated easily and efficiently when needed. Since the structure of $\overline{M}$ is a subset of that of M, we have a structural representation of R using storage bounded by the number of nonzeros in the matrix M.

In what follows, we shall characterize the structure of the skeleton matrix $\overline{M}$ in terms of the original matrix A. We shall use i, j, and k as subscripts, and unless otherwise specified, we assume that $k < j < i$.

Lemma 4.1: $M_{ij} \neq 0$ if and only if $A_{ri} \neq 0$ and $A_{rj} \neq 0$ for some row r of A.

□

Lemma 4.2: Let r be a row subscript of A such that both A_{ri} and A_{rj} are nonzero. If $A_{rk} \neq 0$ for some $k < j$ in this row, then $\overline{M}_{ij} = 0$.

Proof: It follows from Lemma 4.1 that M_{ij}, M_{ik}, and M_{jk} are all nonzero. Since $k < j < i$, in the elimination tree of M, by Theorem 2.3, node i is an ancestor of node j, which is in turn an ancestor of node k. Moreover, since M_{ik} is nonzero and is a descendant of M_{ij}, the latter cannot be a leaf node so $M^{\frown}_{ij}$ must be zero.

□

Theorem 4.3: If $M^{\frown}_{ij} \neq 0$ then $M_{ij} \neq 0$ and for every row r of A with nonzero A_{ri} and A_{rj}, $A_{rk} = 0$ for $k = 1, \ldots, j-1$.

Proof: Assume that $M^{\frown}_{ij}$ is nonzero. Clearly, M_{ij} must be nonzero, and it follows from Lemma 4.2 that every row of A with nonzeros in the i-th and j-th locations must have $j-1$ leading zeros.

□

The condition of Theorem 4.3 is sufficient but not necessary, as the following 3 by 3 matrix example illustrates.

$$A = \begin{pmatrix} x & x & \\ x & & x \\ & x & x \end{pmatrix} \qquad M = \begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \end{pmatrix} \qquad M^{\frown} = \begin{pmatrix} x & x & x \\ x & x & \\ x & & x \end{pmatrix}$$

In the corresponding skeleton matrix, $M^{\frown}_{32} = 0$. However, Theorem 4.3 is clearly satisfied.

Theorem 4.4: $\eta_t(M^{\frown}) \leq \eta(A) - m$,

where $\eta_t(M^{\frown})$ is the number of off-diagonal nonzeros in the upper triangular part of the symmetric matrix $M^{\frown}$, and $\eta(A)$ is the number of nonzeros in the matrix A.

Proof: The result is proved by identifying a unique nonzero in the matrix A for each nonzero of $M^{\frown}$. Consider a nonzero $M^{\frown}_{ij}$ in the skeleton matrix $M^{\frown}$. By Lemma 4.1, there must exist a row r in A such that A_{ri} and A_{rj} are nonzero. We map this nonzero $M^{\frown}_{ij}$ to the nonzero A_{ri} of the original matrix A. We shall show that this mapping is injective.

Assume that the nonzeros $M^{\frown}_{ij}$ and $M^{\frown}_{ik}$ are both mapped into the same nonzero A_{ri} of A. By Theorem 4.3, we must have $j = k$, so that the mapping is injective.

Moreover, the first nonzero in each row of A will never be mapped into, and there are m rows. The result therefore follows.

□

5. Representation of the Triangular Factors Obtained via Gaussian Elimination with Partial Pivoting

We now consider an application of the results of the previous sections to the problem of representing the structure of the triangular factors of an n by n, sparse, nonsymmetric and nonsingular matrix A obtained using Gaussian elimination with partial pivoting. Following [10], we denote the triangular decomposition by

$$A = P_1 M_1 P_2 M_2 \cdots P_{n-1} M_{n-1} U \quad ,$$

where P_k is a permutation matrix that performs the row interchange at step k of the partial pivoting algorithm, M_k is an elementary lower triangular matrix with the k-th column containing the multipliers used at step k, and U is the final upper triangular matrix.

In [10], George and Ng have provided an implementation of sparse Gaussian elimination with partial pivoting. Their scheme determines from the structure of A a data structure that will accommodate the nonzeros in the factors M_k and U for *all* possible partial pivoting sequences.

In what follows, we briefly review this new scheme, and show that the structural modifications one has to apply to A are the same as those in computing the orthogonal decomposition of A using Householder transformations. Hence the structural representation scheme we have described in Section 3 is equally applicable to sparse Gaussian elimination with partial pivoting. We shall assume that the matrix A is irreducible and has a zero-free diagonal.

As in Section 3, we partition A into

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

where u and v are $(n-1)$-vectors. Regardless of the actual row interchange at step 1, the final structure of the pivot row (or row 1 of U) must be contained in the structure of the following row vector

$$(d \ , \ \bar{v}^T) \ = \ (d \ , \ v^T+u^TE) \quad .$$

Thus, instead of considering A at step 1, one applies one step of Gaussian elimination *without* row interchange to the structure of the following modified matrix

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

$$= \ \begin{pmatrix} 1 & 0 \\ u/d & I \end{pmatrix} \begin{pmatrix} d & \bar{v}^T \\ 0 & E-u\bar{v}^T/d \end{pmatrix} \ = \ \bar{M}_1\bar{A}_1 \quad .$$

It is clear that the structure of $\bar{M}_1$ must contain that of M_1, and the structure of the matrix $(E-u\bar{v}^T/d)$ must also contain the structure of the matrix that would result from modifying E in Gaussian elimination with partial pivoting, *irrespective of the actual row interchange that occurs.* One of the attractive features of this approach is that the structure of the matrix $(E-u\bar{v}^T/d)$ can be computed readily from the structure of A without knowing the actual pivoting sequence.

If the same idea is applied recursively to the structure of $(E-u\bar{v}^T/d)$, we obtain a lower triangular matrix, say $\bar{L}$, and an upper triangular matrix, say $\bar{U}$, such that the structure of U is contained in that of $\bar{U}$, and the structure of column k of M_k is contained in that of column k of $\bar{L}$, regardless of the partial pivoting sequence. An efficient algorithm has been given in [10] for determining the structures of $\bar{L}$ and $\bar{U}$.

Now observe that the structure of $\begin{pmatrix} 1 \\ u/d \end{pmatrix}$ is identical to that of $\begin{pmatrix} \beta \\ w \end{pmatrix}$, the Householder vector at step 1 in Section 3, and the structure of $(E-u\bar{v}^T/d)$ (or $\bar{A}_1$) is also identical to that of $(E-wy^T/\beta)$ (or PA) in Section 3. That is, structurally, $\bar{L}$ and $\bar{U}$ are identical respectively to the Householder matrix H and the upper triangular factor R described in Section 3. (Of course, here we assume both H and R are n by n.) Thus, the structural representation scheme described in Section 3 for H is also applicable for $\bar{L}$, and the results of sections 2 and 4 are applicable to the representation of $\bar{U}$.

For more details about an associated data structure and implementation, along with some numerical experiments, the reader is referred to [9].

6. Concluding Remarks

In this paper, we have described a basic structure called an elimination tree that is useful in representing the structures of triangular and orthogonal factors arising in numerical linear algebra computations involving sparse matrices. The representations are very efficient in that they require storage that is in all cases bounded by the number of nonzeros in the original matrix A, and is independent of the amount of fill-in that might occur during the factorization process.

Results in section 2 provide a structural representation of the Cholesky factor L of a positive definite matrix A in terms of the structure of its skeleton matrix $\bar{A}$.

In section 3 we showed that the orthogonal matrix Q in the sparse QR factorization of a general rectangular matrix A can be represented in a factored form using the elimination tree structure of R together with an additional $m + n$ overhead items.

Theorem 4.4 in section 4 shows that the number of nonzeros in $\bar{M}$, where $M=A^TA$, is bounded by the number of nonzeros in the original matrix A. It follows from the results in section 2 that R can be represented structurally using overhead bounded by that of the given matrix A. Combining this with the result in section 3, we have a very efficient scheme for the structural representation of the orthogonal factors Q and R of A. The storage overhead will be no more than the number of nonzeros in A together with a few vectors of length m and n.

Finally, in section 5 we showed that the results of sections 2 through 4 are also applicable in the context of a static storage implementation of Gaussian elimination with partial pivoting.

7. References

[1] T.F. COLEMAN, A. EDENBRANDT, AND J.R. GILBERT, "Predicting fill for sparse orthogonal factorization", Technical Report 83-578, Dept. of Computer Science, Cornell University (1983).

[2] I.S. DUFF, "Algorithm 575. Permutations for a zero-free diagonal", *ACM Trans. on Math. Software* **7**, pp. 387-390 (1981).

[3] I.S. DUFF, "Full matrix techniques in sparse Gaussian elimination", in *Lecture Notes in Mathematics (912)*, ed. G. A. Watson, Springer-Verlag (1982).

[4] I.S. DUFF AND J.K. REID, "The multifrontal solution of indefinite sparse symmetric linear equations", *ACM Trans. on Math. Software* **9**, pp. 302-325 (1983).

[5] S.C. EISENSTAT, M.H. SCHULTZ, AND A.H. SHERMAN, "Applications of an element model for Gaussian elimination", in *Sparse Matrix Computations*, ed. J.E. Bunch and D.J. Rose, Academic Press, pp. 85-96 (1976).

[6] S. C. EISENSTAT, M. H. SCHULTZ, AND A. H. SHERMAN, "Software for sparse Gaussian elimination with limited core storage", in *Sparse Matrix Proceedings*, ed. I.S. Duff and G.W. Stewart, SIAM Press, pp. 135-153 (1979).

[7] J.A. GEORGE AND J.W-H. LIU, "An optimal algorithm for symbolic factorization of symmetric matrices", *SIAM J. Comput.* **9**, pp. 583-593 (1980).

[8] J.A. GEORGE AND J.W-H. LIU, "Householder reflections versus Givens rotations in sparse orthogonal decomposition", *Linear Algebra and its Appl.*, (1986). (To appear.)

[9] J. A. GEORGE, J. W. H. LIU, AND E. NG, "A data structure for sparse QR and LU factors", Research Report CS 85-16, Dept of Computer Science, University of Waterloo (1985).

[10] J.A. GEORGE AND E.G-Y. NG, "Symbolic factorization for sparse Gaussian elimination with partial pivoting", Research Report CS-84-43, Department of Computer Science, University of Waterloo (1984).

[11] J.A.G. JESS AND H.G.M. KEES, "A data structure for parallel L/U decomposition", *IEEE Trans. Comput.* **C-31**, pp. 231-239 (1982).

[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, "A compact row storage scheme for sparse Cholesky factors using elimination trees", *ACM Trans. on Math. Software*, (1985). (To appear)

[14] S.V. PARTER, "The use of linear graphs in Gaussian elimination", *SIAM Review* **3**, pp. 364-369 (1961).

[15] F. J. PETERS, "Sparse matrices and substructures ", Mathematical Centre Tracts 119, Mathematisch Centrum, Amsterdam, The Netherlands (1980).

[16] R. SCHREIBER, "A new implementation of sparse Gaussian elimination", *ACM Trans. on Math Software* **8**, pp. 256-276 (1982).