A Partial Pivoting Implementation of Gaussian Elimination for Sparse Systems

Alan George
Esmond Ng

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

CS-83-15
May 1983

A partial pivoting implementation of Gaussian elimination for sparse systems+

Alan George

Esmond Ng

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

Research report CS-83-15
May 1983

ABSTRACT

In this paper, we consider the problem of solving a sparse nonsingular system of linear equations. We show that the structures of the triangular matrices obtained in the LU-decomposition of a sparse nonsingular matrix A using Gaussian elimination with partial pivoting are contained in those of the Cholesky factors of A^TA, provided that the diagonal elements of A are nonzero. Based on this result, a method for solving sparse linear systems is then described. The main advantage of this method is that the numerical computation can be done using a static data structure. Numerical experiments comparing this method with other implementations of Gaussian elimination for solving sparse linear systems are presented and the results indicate that the method proposed in this paper is quite competitive with other approaches.

+ Research was supported in part by the Canadian Natural Sciences and Engineering Research Council under Grant A8111.

1. Introduction

In this paper, we consider the direct solution of the system of linear equations

$$Ax = b \quad ,$$

where A is a given $n \times n$ matrix, b is a given n-vector, and x is the n-vector to be computed. We assume that A is sparse, nonsymmetric and nonsingular. One of the most popular techniques for solving such a linear system involves computing an LU-decomposition of A using Gaussian elimination with partial pivoting. That is, A is decomposed into

$$A = P_1 L_1 P_2 L_2 \cdots P_{n-1} L_{n-1} U \quad ,$$

where P_k is an $n \times n$ permutation matrix corresponding to the row interchanges in step k, L_k is an $n \times n$ unit lower triangular matrix whose $k-th$ column contains the multipliers, and U is an $n \times n$ upper triangular matrix. Then the solution to the original linear system is obtained by solving the systems

$$P_1 L_1 P_2 L_2 \cdots P_{n-1} L_{n-1} y = b \quad ,$$

and

$$Ux = y \quad .$$

Given a sparse matrix A, it is clear that *fill-in* will occur during the decomposition process; that is, nonzeros may be created in positions where there are zeros in A. Thus, space has to be allocated not only for the nonzeros in A, but also for the fill-in. Note that the row interchanges depend on both the *structure* and the *numerical values* of A (and of the subsequent reduced matrices). Furthermore, the structures of the triangular matrices L_k's and U also depend on the structure of A and the row interchanges. Thus, one cannot predict where fill-in will occur before the numerical decomposition begins, since the row interchanges are not known beforehand. Consequently, it is common practice in the implementation of sparse LU-decomposition with row interchanges to allocate space for any fill-in *during* the numerical computation phase. (That is, one uses a dynamic data structure.) Most computer packages that compute an LU-decomposition of a sparse matrix with row interchanges use a dynamic data structure. This usually results in substantial overhead in both storage requirements and execution time.

In this paper, we show that the structures of the triangular matrices L_k and U are contained in the structures of the Cholesky factors of the symmetric positive definite matrix A^TA, as long as the diagonal elements of A are nonzero. Suppose A^TA is sparse. Since it is possible to determine the structure of the Cholesky factor of A^TA efficiently from the structure of A^TA, this gives us a scheme for implementing the LU-decomposition of A using Gaussian elimination with partial pivoting. The attractive feature of this scheme is that a dynamic data structure is *not* needed. By analyzing the structure of A^TA, we determine the structure of the Cholesky factor of A^TA and set up a storage scheme. Then we simply use that static storage scheme in the numerical decomposition of A using Gaussian elimination with partial pivoting.

The proposed scheme assumes that A^TA is sparse if A is sparse, but there are examples in which A^TA is dense even though A is sparse. This usually occurs when a relatively small number of the rows of A are dense. We propose a technique to handle these dense rows so that we only have to compute the LU-decomposition of a sparse submatrix of

A.

An outline of the paper is as follows. In Section 2 we derive the main results which show that the structures of the triangular matrices L_k and U are contained in those of the Cholesky factors of A^TA. The effect of permuting the columns of A is examined in Section 3. In Section 4 the proposed method is described and in Section 5 it is compared with other implementations of Gaussian elimination for solving sparse linear systems. Numerical experiments are also provided to compare the performance of the various implementations in Section 5. We consider the effect of dense rows and propose a technique to handle them in Section 6. Finally, some concluding remarks are provided in Section 7.

The structure of the Cholesky factor of A^TA is also used in the solution of sparse least squares problems $\min_x \|Ax-b\|_2$ [14] and underdetermined systems of linear equations $A^Tx=b$ [15], where A is $m\times n$, with $m\geq n$.

2. Preliminary results

Lemma 2.1 (Duff [1])

Let A be a sparse nonsingular matrix. Then there exists a permutation matrix Q such that the diagonal elements of QA are all nonzero.

The matrix QA is then said to have a *zero-free diagonal*. The problem of finding such a permutation matrix is a well-known one and it is sometimes called the *assignment problem*. See [4] for a description of this problem and [5] for an efficient algorithm for finding Q.

Since Lemma 2.1 is true for every nonsingular matrix A, we may therefore assume that the rows of the given matrix A have already been permuted. That is, in the remaining of this section, we assume that A has a zero-free diagonal.

The following notation will be used throughout the discussion in this section. Let A be a sparse $n\times n$ matrix. The (i,j)-element of A is denoted by A_{ij}. The set of subscripts of the nonzeros of A is denoted by $NONZ(A)$. That is,

$$NONZ(A) = \{(i,j) \mid A_{ij}\neq 0\} \ .$$

Furthermore, we will assume that exact cancellation does not occur.

The following result, which we state without proof, is an immediate consequence of the fact that A has a zero-free diagonal. It says that $NONZ(A)$ is contained in $NONZ(A^TA)$.

Lemma 2.2

Assume A has a zero-free diagonal and let $B=A^TA$. If $A_{ij}\neq 0$, then $B_{ij}\neq 0$. That is, $NONZ(A)\subseteq NONZ(A^TA)$.

Now consider applying the first step of Gaussian elimination to A with partial pivoting.

$$P_1A = \begin{pmatrix} \alpha & u^T \\ v & E \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{v}{\alpha} & I \end{pmatrix} \begin{pmatrix} \alpha & u^T \\ 0 & F \end{pmatrix}$$

Here P_1 is an $n \times n$ permutation matrix chosen so that

$$|\alpha| \geq \|v\|_\infty \ .$$

Assume P_1 interchanges rows 1 and s of A $(1 \leq s \leq n)$. Then we have

$$\alpha \neq 0 \ .$$

For simplicity, we assume that

$$u^T = \left(u_2, u_3, \cdots, u_n\right)$$

and

$$v^T = \left(v_2, v_3, \cdots, v_n\right) \ .$$

We also assume that

$$E = \begin{pmatrix} E_{22} & E_{23} & \cdots & E_{2n} \\ E_{32} & E_{33} & \cdots & E_{3n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ E_{n2} & E_{n3} & \cdots & E_{nn} \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} F_{22} & F_{23} & \cdots & F_{2n} \\ F_{32} & F_{33} & \cdots & F_{3n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ F_{n2} & F_{n3} & \cdots & F_{nn} \end{pmatrix} \ .$$

Note that if $s \neq 1$, then, because A has a zero-free diagonal and since partial pivoting is used,

$$u_s = A_{ss} \neq 0 \quad \text{and} \quad v_s = A_{11} \neq 0 \ .$$

It is easy to see that

$$F = E - \frac{1}{\alpha} v u^T \ .$$

Thus,

$$NONZ(F) = NONZ(E) \bigcup NONZ(vu^T) \ .$$

We now consider the structure of F more carefully. First suppose $s=1$. Then $P_1 = I$ and $E_{ii} = A_{ii} \neq 0$ for $2 \leq i \leq n$, since A has a zero-free diagonal. Hence $F_{ii} \neq 0$ for $2 \leq i \leq n$. If $s \neq 1$, then $E_{ii} = A_{ii} \neq 0$ for $2 \leq i \leq n$ and $i \neq s$, but $E_{ss} = A_{1s}$ may be zero. For $2 \leq i \leq n$ and $i \neq s$, clearly

$$F_{ii} = E_{ii} - \frac{1}{\alpha} v_i u_i \neq 0 \ .$$

For $i = s$,

$$F_{ss} = E_{ss} - \frac{1}{\alpha} v_s u_s \neq 0 \ ,$$

since u_s and v_s are both nonzero. Thus we have proved that all the diagonal elements of F are nonzero.

Theorem 2.3

The $(n-1)\times(n-1)$ matrix F has a zero-free diagonal.

Corollary 2.4

$NONZ(F) \subseteq NONZ(F^TF)$.

Consider the $n\times n$ symmetric positive definite matrix $B=A^TA$.

$$B = A^TA = A^TP_1^TP_1A = (P_1A)^T(P_1A)$$

$$= \begin{pmatrix} \alpha & v^T \\ u & E^T \end{pmatrix} \begin{pmatrix} \alpha & u^T \\ v & E \end{pmatrix} = \begin{pmatrix} \alpha^2+v^Tv & \alpha u^T+v^TE \\ \alpha u+E^Tv & uu^T+E^TE \end{pmatrix} = \begin{pmatrix} \beta & w^T \\ w & G \end{pmatrix} ,$$

where $\beta=\alpha^2+v^Tv$, $w=\alpha u+E^Tv$, and $G=uu^T+E^TE$. Applying the first step of Cholesky decomposition to B, we obtain

$$B = \begin{pmatrix} \beta & w^T \\ w & G \end{pmatrix} = \begin{pmatrix} \sqrt{\beta} & 0 \\ \frac{w}{\sqrt{\beta}} & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} \sqrt{\beta} & \frac{w^T}{\sqrt{\beta}} \\ 0 & I \end{pmatrix} ,$$

where

$$H = G - \frac{1}{\beta}ww^T .$$

The following results show the relationship between the structures of F, H, u, v and w.

Theorem 2.5

$NONZ(F) \subseteq NONZ(H)$.

Proof:

First note that

$$H = G - \frac{1}{\beta}ww^T = E^TE + uu^T - \frac{1}{\beta}(\alpha^2uu^T + \alpha E^Tvu^T + \alpha uv^TE + E^Tvv^TE) .$$

Thus, assuming exact cancellation does not occur,

$$NONZ(H) =$$

$$NONZ(E^TE) \bigcup NONZ(uu^T) \bigcup NONZ(E^Tvu^T) \bigcup NONZ(uv^TE) \bigcup NONZ(E^Tvv^TE) .$$

Also note that

$$F^TF = (E - \frac{1}{\alpha}vu^T)^T(E - \frac{1}{\alpha}vu^T) = E^TE - \frac{1}{\alpha}uv^TE - \frac{1}{\alpha}E^Tvu^T + \frac{v^Tv}{\alpha^2}uu^T .$$

Hence,

$$NONZ(F^TF) = NONZ(E^TE) \bigcup NONZ(uv^TE) \bigcup NONZ(E^Tvu^T) \bigcup NONZ(uu^T) ,$$

$$\subseteq NONZ(H) .$$

Using this observation, together with Corollary 2.4, we can now conclude that

$$NONZ(F) \subseteq NONZ(H) \ .$$

Theorem 2.6

(1) $NONZ(u) \subseteq NONZ(w)$.

(2) $NONZ(v) \subseteq NONZ(w)$.

Proof:

It is obvious that $NONZ(u) \subseteq NONZ(w)$, since $w = \alpha u + E^T v$.

Now consider the structure of v. First assume that $s=1$. Then $E_{ii}=A_{ii} \neq 0$ for $2 \leq i \leq n$. Note that

$$w_i = \alpha u_i + \sum_{k=2}^{n} E_{ki} v_k = \alpha u_i + E_{ii} v_i + \sum_{\substack{k=2 \\ k \neq i}}^{n} E_{ki} v_k \ , \qquad \text{for } 2 \leq i \leq n \ .$$

Thus, $w_i \neq 0$ if $v_i \neq 0$.

On the other hand, suppose $s \neq 1$. For $i \neq s$, $E_{ii}=A_{ii} \neq 0$ and

$$w_i = \alpha u_i + E_{ii} v_i + \sum_{\substack{k=2 \\ k \neq i}}^{n} E_{ki} v_k \ .$$

Thus, if $v_i \neq 0$, then $w_i \neq 0$. For $i=s$, E_{ss} may be zero, but $v_s \neq 0$ and $u_s \neq 0$. Hence

$$w_s = \alpha u_s + \sum_{k=2}^{n} E_{ks} v_k \neq 0 \ .$$

This shows that $NONZ(v) \subseteq NONZ(w)$.

Theorems 2.5 and 2.6 show that, at least for the first step of the LU-decomposition of A, the structures of u, v and F are contained in those of w and H. These two results can be extended to cover the complete LU-decomposition of A using Gaussian elimination with partial pivoting. Recall that F is $(n-1)\times(n-1)$ and has a zero-free diagonal, and H is $(n-1)\times(n-1)$, symmetric and positive definite. Thus we can consider the LU-decomposition of F and the Cholesky decomposition of H. Indeed, by applying Theorems 2.5 and 2.6 recursively to the matrices F and H, we obtain Theorem 2.7 which is an extension of Theorems 2.5 and 2.6.

Before stating Theorem 2.7, we first introduce more notation. Consider the two sequences of matrices:

$$\left\{F^{(0)}, F^{(1)}, F^{(2)}, \cdots, F^{(n-1)}\right\}$$

and

$$\left\{H^{(0)}, H^{(1)}, H^{(2)}, \cdots, H^{(n-1)}\right\} \ ,$$

where $F^{(0)}=A$ and $H^{(0)}=A^T A$. For $k=1,2, \cdots, n-1$, $F^{(k)}$ is obtained by applying one step of Gaussian elimination to $F^{(k-1)}$ with partial pivoting:

$$\bar{P}_k F^{(k-1)} = \begin{pmatrix} \alpha_k & (u^{(k)})^T \\ v^{(k)} & E^{(k)} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ \frac{v^{(k)}}{\alpha_k} & I_{n-k} \end{pmatrix} \begin{pmatrix} \alpha_k & (u^{(k)})^T \\ 0 & F^{(k)} \end{pmatrix} = \bar{L}_k \begin{pmatrix} \alpha_k & (u^{(k)})^T \\ 0 & F^{(k)} \end{pmatrix} ,$$

where $\bar{P}_k$ is a permutation matrix of order $(n-k+1)$, I_{n-k} is the identity matrix of order $(n-k)$ and $F^{(k)}=E^{(k)}-\frac{1}{\alpha_k}v^{(k)}(u^{(k)})^T$. Similarly, for $k=1,2,\cdots,n-1$, $H^{(k)}$ is obtained by applying one step of Cholesky decomposition to $H^{(k-1)}$:

$$H^{(k-1)} = \begin{pmatrix} \beta_k & (w^{(k)})^T \\ w^{(k)} & G^{(k)} \end{pmatrix}$$

$$= \begin{pmatrix} \sqrt{\beta_k} & 0 \\ \frac{w^{(k)}}{\sqrt{\beta_k}} & I_{n-k} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & H^{(k)} \end{pmatrix} \begin{pmatrix} \sqrt{\beta_k} & \frac{(w^{(k)})^T}{\sqrt{\beta_k}} \\ 0 & I_{n-k} \end{pmatrix} = \hat{R}_k^T \begin{pmatrix} 1 & 0 \\ 0 & H^{(k)} \end{pmatrix} \hat{R}_k .$$

where $H^{(k)}=G^{(k)}-\frac{1}{\beta_k}w^{(k)}(w^{(k)})^T$. It can be shown that

$$A = P_1L_1P_2L_2 \cdots P_{n-1}L_{n-1}U ,$$

where

$$P_k = \begin{pmatrix} I_{k-1} & O \\ O & \bar{P}_k \end{pmatrix} ,$$

$$L_k = \begin{pmatrix} I_{k-1} & O \\ O & \bar{L}_k \end{pmatrix} = \begin{pmatrix} I_{k-1} & O & O \\ O & 1 & O \\ O & \frac{v^{(k)}}{\alpha_k} & I_{n-k} \end{pmatrix} , \quad 1 \le k \le n-1 ,$$

and

$$U = \begin{pmatrix} \alpha_1 & & (u^{(1)})^T & & \\ & \alpha_2 & & (u^{(2)})^T & \\ O & & \alpha_3 & (u^{(3)})^T & \\ & O & & \cdot & \\ & & O & & \cdot \\ & & & & \cdot \end{pmatrix} .$$

Also, we have

$$B = R_1^TR_2^T \cdots R_{n-1}^TR_n^TR_nR_{n-1} \cdots R_2R_1 ,$$

where

$$R_k = \begin{pmatrix} I_{k-1} & O \\ O & \hat{R}_k \end{pmatrix} = \begin{pmatrix} I_{k-1} & O & O \\ O & \sqrt{\beta_k} & \frac{(w^{(k)})^T}{\sqrt{\beta_k}} \\ O & O & I_{n-k} \end{pmatrix} , \quad \text{for } 1 \le k \le n-1 ,$$

and

$$R_n = \begin{pmatrix} I_{n-1} & O \\ O & \sqrt{H^{(n-1)}} \end{pmatrix} .$$

The proof of Theorem 2.7 is the same as those of Theorems 2.5 and 2.6, and hence is omitted.

Theorem 2.7

For $k=1,2,\cdots,n-1$,

(1) $F^{(k)}$ has a zero-free diagonal.

(2) $NONZ(u^{(k)}) \subseteq NONZ(w^{(k)})$.

(3) $NONZ(v^{(k)}) \subseteq NONZ(w^{(k)})$.

(4) $NONZ(F^{(k)}) \subseteq NONZ(H^{(k)})$.

Because of the way in which $H^{(k)}$'s are generated, it is not hard to see that

$$\bigcup_{k=0}^{n-1} NONZ(H^{(k)}) = \bigcup_{k=1}^{n} NONZ(R_k + R_k^T) .$$

Thus, Theorem 2.7 essentially says that the structures of the triangular matrices obtained in the LU-decomposition of A are contained in those of the Cholesky factors of A^TA. Consequently, if we allocate space for the nonzero structure of the Cholesky factor of A^TA, then that data structure will *always* have space to accommodate any fill-in that occurs during the LU-decomposition of A. These results are important since it allows the LU-decomposition of A using Gaussian elimination *with partial pivoting* to be computed in a *predictable* amount of space. One may regard this result as saying that the structures of the Cholesky factors of A^TA predicts the *worst possible* structures of the triangular matrices L_k's and U, for *any* pivotal sequence $\{P_1,P_2,\cdots,P_{n-1}\}$.

Suppose B is a sparse symmetric positive definite matrix and denote its Cholesky factor by R_B. It is well known that the structure of R_B can be *predicted* from that of the matrix B. Thus one can allocate space for the nonzeros of R_B *before* the numerical computation begins. Furthermore, there are algorithms that accomplish these tasks efficiently. See [17] for details.

Hence, by setting $B=A^TA$ and finding a data structure for the Cholesky factors, R_B and R_B^T, of B, we know from Theorems 2.5, 2.6 and 2.7 that there will always be space in that (static) data structure to accommodate any fill-in created during the LU-decomposition of A using Gaussian elimination with partial pivoting. Note that the symmetric positive definite matrix is assumed to be sparse. This may not be true in some cases and we will address this problem in a later section.

3. Effect of permuting the columns of A

Let P_c be an $n \times n$ permutation matrix and consider the matrix AP_c. Assume for the moment that AP_c has a zero-free diagonal. Denote the LU-decomposition of AP_c by

$$AP_c = P_1L_1P_2L_2 \cdots P_{n-1}L_{n-1}U .$$

The results in the previous section show that the nonzero structures of the triangular matrices are contained in those of the Cholesky factors of $(AP_c)^T(AP_c)=P_c^TA^TAP_c=P_c^TBP_c$, where $B=A^TA$. We assume that B is sparse. It is well known that, for a sparse symmetric and positive definite matrix B, the choice of the permutation matrix P_c can drastically affect the sparsity of the Cholesky factor of $P_c^TBP_c$ [17]. Hence it is desirable to find a P_c so that $P_c^TBP_c$ has a sparse Cholesky decomposition.

The problem of finding a P_c that yields minimal fill-in is an NP-complete problem [19]. However, there are efficient heuristic algorithms that produce a P_c so that $P_c^TBP_c$ has a reasonably sparse Cholesky decomposition. Examples include the nested dissection algorithm and the minimum degree algorithm [17].

It should be noted that even though A has a zero-free diagonal, the matrix AP_c may not necessarily have one. We illustrate this by a small example. Consider the following 4×4 matrix.

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

Thus

$$A^TA = \begin{pmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \\ & & & \times \end{pmatrix}$$

and $NONZ(A) \subseteq NONZ(A^TA)$. Let

$$P_c = \begin{pmatrix} & & 1 & \\ & 1 & & \\ 1 & & & \\ & & & 1 \end{pmatrix} .$$

Now

$$AP_c = \begin{pmatrix} & \times & \times & \\ & \times & & \\ \times & & & \\ & & & \times \end{pmatrix}$$

and

$$P_c^TA^TAP_c = \begin{pmatrix} \times & & & \\ & \times & \times & \\ & \times & \times & \\ & & & \times \end{pmatrix} .$$

Note that AP_c does not have a zero-free diagonal and the nonzero structure of $P_c^TA^TAP_c$ does not even contain that of AP_c.

One way to preserve the property that A has a zero-free diagonal is as follows. Instead of permuting the columns of A by P_c, we permute *both* the columns and rows of A symmetrically by P_c. That is, we would consider the matrix $P_c^TAP_c$. Note that

$$(P_c^TAP_c)^T(P_c^TAP_c) = P_c^TA^TAP_c \quad .$$

Thus premultiplying AP_c by P_c^T does not affect the structure of $P_c^TA^TAP_c$ at all. However, $P_c^TAP_c$ now has a zero-free diagonal (as long as A has one). To illustrate this, consider the previous 4×4 example again.

$$P_c^TAP_c = \begin{pmatrix} \times & & & \\ & \times & & \\ & \times & \times & \\ & & & \times \end{pmatrix} ,$$

and the structure of $P_c^TA^TAP_c$ indeed contains that of $P_c^TAP_c$.

4. Proposed method

The results and discussions in Sections 2 and 3 provide us with a scheme for solving a sparse system of linear equations

$$Ax = b \quad .$$

In general, the coefficient matrix A that is provided may not have a zero-free diagonal. Thus it is necessary to find a row permutation Q so that the diagonal elements of QA are nonzero. This can be achieved by using the algorithm described in [5].

We now summarize the solution scheme below.

(1) Find a permutation matrix Q so that QA has a zero-free diagonal.

(2) Determine the structure of $B=A^TA$.

(3) Find a symmetric permutation P_c so that $P_c^TBP_c$ has a sparse Cholesky factor. Denote the Cholesky factorization by $P_c^TBP_c = \hat{R}^T\hat{R}$.

(4) Determine the structure of the Cholesky factor $\hat{R}$ of $P_c^TBP_c$, and set up a storage scheme that exploits the sparsity of $\hat{R}$ and $\hat{R}^T$.

(5) Input the numerical values of A, storing it as $P_c^TQAP_c$.

(6) Compute the LU-decomposition of $P_c^TQAP_c$ using Gaussian elimination with partial pivoting. Store the triangular factors in the storage structure for $\hat{R}$ and $\hat{R}^T$.

(7) Solve $P_c^TQAP_cP_c^Tx=P_c^TQb$ using the LU-decomposition.

A few remarks on the implementation are in order. First, we only work with the *structures* of A and A^TA in Steps (1), (3) and (4). Second, efficient algorithms are available for performing Steps (1), (3) and (4). In the experiments which we will describe in the next section, we use the code given in [5] to find the permutation Q in Step 1. We use the minimum degree algorithm from SPARSPAK to find the symmetric permutation P_c in Step (3) and also the symbolic factorization routine from SPARSPAK to carry out Step (4) [16]. Third, the approach we have employed assumes that A^TA is sparse if A is sparse. However there are some instances in which A^TA may be dense even though A is sparse. We will deal with this situation in Section 6.

5. Comparison with other methods for solving sparse linear systems

There are many codes available for solving sparse systems of linear equations $Ax=b$ using an LU-decomposition of A. The ones we have considered include SPARSPAK [16], MA28 from Harwell [2], NSPIV [18], and an implementation of the method proposed in Section 4. In this section, we first consider the basic methodology used by each package and examine its advantages and disadvantages.

SPARSPAK --

This package uses the structure of $A+A^T$ and computes an LU-decomposition of A without using partial pivoting. A symmetric row and column ordering is chosen and a data structure is set up before computing the decomposition. It then uses that static data structure in the numerical computation phase. Since there are no row interchanges, numerical stability may be a problem. Furthermore, it requires the diagonal elements of A to be nonzero. Of course, this can be circumvented by finding an assignment before choosing the symmetric ordering. However, there is still a chance that some of the diagonal elements may become zeros during the decomposition process. This is illustrated in our experiments. The symmetric ordering we have used in the numerical experiments was a minimum degree ordering.

MA28 from Harwell --

In this code, column and row permutations are chosen to maintain numerical stability and preserve sparsity simultaneously. That is, the permutations will depend on the numerical values of the nonzeros of the elements of A and the pivotal sequence. Thus one cannot predict how much space is needed before numerical computation begins, and storage has to be allocated during the numerical computation phase. Experience shows that the overhead, both in terms of execution time and storage, can be quite significant. A threshold pivoting technique is used in the search of pivot; that is, at the $k-th$ step of the decomposition process, an element in the reduced matrix, say A_{ki}, will be chosen as the pivot if it satisfies

$$|A_{ki}| \geq u \max_j |A_{kj}| \quad ,$$

where u is a user specified parameter satisfying $0 \leq u \leq 1$ (see [2,10] for details). Increasing the threshold parameter may improve the accuracy, but this may also increase both the storage requirements and execution times. In our experiments, we have set $u=0.1$.

NSPIV --

This code computes an LU-decomposition of A using partial pivoting. The elimination is carried out row by row. Storage for fill-in is allocated during the numerical decomposition phase. The user is responsible for the choice of initial row and column orderings. In our experiments, the initial orderings were those suggested by Sherman [18]. The column ordering was the original ordering of the variables, and the rows were arranged in increasing number of nonzeros. Experience indicates that it can be very efficient. However, the lower triangular matrix L is not saved. Thus, a potential drawback is that if it is used to solve several systems which have the same coefficient matrix, the factorization must be repeated for each new right hand side.

The method proposed in Section 4 --

The method we propose computes an LU-decomposition of A using Gaussian elimination with partial pivoting. Data structures for storing L_k, $1 \le k \le n-1$, and U can be set up before the numerical computation begins by finding the structures of the Cholesky factors of $A^T A$. The numerical computation is then performed using the static data structure. The triangular matrices L_k, $1 \le k \le n-1$, and U are saved so that solution of several systems with the same coefficient matrix is very convenient and efficient. A potential weakness is that it makes use of the structure of the matrix $A^T A$ which could be dense or severely overestimate the storage for L_k and U. (More on this can be found in Section 6). A good column ordering (that would yield low fill-in in the Cholesky factor of $A^T A$) is chosen prior to the numerical computation. In our experiments, we have used a minimum degree ordering as the column ordering.

There are other packages which we have not considered. Examples include MA32 [3, 6] and MA37 [12] from Harwell and the Yale sparse matrix package [13]. The package MA37 should be of particular interest. It computes an LU-decomposition of A using the so-called *multi-frontal technique* and is based on the ideas used in MA27 [7, 11] for solving sparse symmetric indefinite systems. Based on our experience with MA27, we expect MA37 to be effective in terms of storage and execution times. Unfortunately we do not have a copy of MA37 available.

We now provide some numerical experiments to compare the performance of the various implementations. The experiments were carried out on an IBM 4341. The programs were written in ANSI standard FORTRAN and compiled using an IBM VS FORTRAN Optimizing compiler. Single precision arithmetic was used. The test problems include thirteen finite element and nine non-finite element problems. (The non-finite element problems were obtained from Harwell.) Their characteristics are given in Table 5.1. For the finite element problems, the numerical values for the coefficient matrices were generated using a uniform random number generator. For each of the twenty-two test problems, the right-hand side vector was chosen so that the solution vector contained all ones.

Tables 5.2 and 5.3 contain respectively the storage and execution times required by the various methods. Storage requirements are given in terms of number of storage locations required, including space for pointers, subscripts, etc, and execution times are in seconds. Table 5.4 shows the accuracy achieved. We have used $\|x-\bar{x}\|_\infty$ as a measure of the accuracy, where $\bar{x}$ denotes the computed solution. Other terms used in the tables are explained below.

SPARSPAK --

analysis storage and analysis time - amount of space and time required to find an assignment for A, to find a symmetric ordering for $A+A^T$ and to allocate space for the LU-decomposition of A.

solution storage and solution time - amount of space required to store the LU-decomposition and the time required for the computation.

total time - sum of analysis and solution times.

MA28 and NSPIV --

total storage - amount of space required to compute the LU-decomposition.

total time - amount of time to compute the LU-decomposition (including the times required to do any structure analysis for MA28).

New method --

Problem number	Number of unknowns	Number of nonzeros	Remarks
1	265	1753	Graded-L finite element mesh.
2	406	2716	Graded-L finite element mesh.
3	577	3889	Graded-L finite element mesh.
4	778	5272	Graded-L finite element mesh.
5	936	6266	Finite element mesh -- a hollow square (small hole).
6	1009	6865	Finite element mesh -- a graded-L problem.
7	1089	7361	Finite element mesh -- a square problem.
8	1440	9504	Finite element mesh -- a hollow square (large hole).
9	1180	7750	Finite element mesh -- a +-shaped problem.
10	1377	8993	Finite element mesh -- an H-shaped problem.
11	1138	7450	Finite element mesh -- a 3-hole problem.
12	1141	7465	Finite element mesh -- a 6-hole problem.
13	1349	9101	Finite element mesh -- a pinched hole problem.
14	113	655	Matrix pattern supplied by Morven Gentleman.
15	54	291	Matrix pattern supplied by Curtis.
16	57	281	Matrix pattern supplied by Willoughby.
17	199	701	Matrix pattern supplied by Willoughby.
18	130	1037	Matrix from laser problem (A.R. Curtis).
19	363	3279	Matrix from linear programming problem.
20	541	4282	Facsimile convergence matrix.
21	991	6027	Matrix from Philips Ltd (J.P. Whelan).
22	192	2992	Matrix from parabolic pde.

Table 5.1: Characteristics of test problems

analysis storage and analysis time - amount of space and time required to find an assignment for A, to determine a symmetric ordering for A^TA and to allocate space for the Cholesky factors of A^TA.

solution storage and solution time - amount of space required to store the Cholesky factors of A^TA and the time required to compute the LU-decomposition of A.

total time - sum of analysis time and solution time.

Problem	SPARSPAK		MA28	NSPIV	New method	
	analysis	solution	total	total	analysis	solution
1	5362	9795	21985	17326	10902	18765
2	8275	17184	37716	38747	17007	35239
3	11818	26688	71700	55152	24461	55672
4	15991	38347	93449	96863	33267	84716
5	19081	42874	119831	81527	39257	96581
6	20794	51712	134336	123914	43422	119039
7	22346	52803	147364	141046	46406	114967
8	29089	61818	155042	106907	59081	125702
9	23761	42966	98066	71675	48061	76957
10	27626	47739	107349	82694	55566	83870
11	22867	46194	108391	113869	46123	94512
12	22918	46294	120558	85876	46266	98366
13	27646	64876	192496	133844	57306	143317
14	3050	3900	3897	2932	5494	4229
15	983	1002	1851	1609	1835	1575
16	1022	935	1781	1422	1730	1310
17	3716	6268	6493	7240	5044	7435
18	3903	3285	5057	4401	12247	7305
19	13948	18799	17219	19934	20280	23132
20	14722	23308	39846	99696	31798	45252
21	20471	73540	178779	164498	57220	279483
22	7329	9271	21136	21011	16033	15894

Table 5.2: Storage requirements (in number of storage locations)

Following are some remarks on the results.

(1) Even though SPARSPAK requires the least amount of space and execution times in most cases, its accuracy is usually poor. This is expected since there is no pivoting for stability performed.

(2) The results indicate that the method proposed in this paper is quite competitive with both MA28 and NSPIV in most cases. In particular, for finite element problems, our method is certainly better than MA28, in terms of storage requirements, execution time and accuracy. The method may occasionally require a little more storage and execution time than NSPIV. However, it should be pointed out that NSPIV only stores the upper triangular matrix U, whereas in our case, we store *both* the lower and upper triangular matrices in the LU-decomposition. Furthermore, we have ignored the problem of finding "good" initial column and row orderings (if they exist) for NSPIV. Thus, taking these comments into account, it is fair to say that our method performs at least as well as NSPIV for these finite element problems.

(3) Note that we have only proved that the structures of the triangular matrices obtained in the LU-decomposition of A *are contained in* the structures of the Cholesky factors of A^TA. There is a possibility that the amount of space allocated for the Cholesky factors of A^TA may be much larger than that required to store the LU-decomposition.

Problem	SPARSPAK			MA28	NSPIV	New method		
	analysis	solution	total	total	total	analysis	solution	total
1	0.327	0.450	0.777	3.113	3.040	0.783	3.530	4.313
2	0.617	1.070	1.687	6.663	9.083	1.410	9.123	10.533
3	0.810	1.967	2.776	27.038	15.822	1.800	17.779	19.579
4	1.137	3.290	4.426	36.261	33.225	2.703	31.168	33.871
5	1.237	3.170	4.406	60.196	21.845	2.996	34.401	37.398
6	1.377	4.566	5.943	81.238	53.970	3.840	51.167	55.006
7	1.467	4.156	5.623	111.190	56.263	3.496	42.627	46.124
8	1.950	3.836	5.786	77.828	28.263	4.596	36.628	41.224
9	1.557	1.890	3.446	30.765	9.716	3.560	13.672	17.232
10	1.803	1.973	3.776	24.312	11.426	4.120	13.176	17.296
11	1.590	2.606	4.196	50.317	32.701	3.480	24.309	27.788
12	1.680	2.613	4.293	78.435	20.159	3.780	27.978	31.758
13	1.733	5.046	6.780	154.237	47.307	3.923	50.820	54.743
14	0.380	0.160	0.540	0.233	0.070	0.930	0.290	1.220
15	0.053	0.013	0.067	0.087	0.050	0.110	0.077	0.187
16	0.053	0.013	0.067	0.077	0.023	0.110	0.053	0.163
17	0.350	0.210	0.560	0.410	0.547	0.480	0.637	1.117
18	1.837	0.070	1.907	0.250	0.080	2.710	0.863	3.573
19	3.370	fail		1.280	2.823	4.366	3.606	7.973
20	11.363	1.503	12.866	5.043	1721.269	14.062	10.196	24.258
21	5.633	17.819	23.452	147.264	184.002	19.692	302.144	321.836
22	0.527	0.620	1.147	3.116	3.883	1.620	3.313	4.933

Table 5.3: Execution times (in seconds)

In order to explore this question, we have shown in Table 5.5 the percentage of storage that is actually utilized. The results indicate that, for the finite element problems, the utilization is approximately 50% for the lower triangular portion and 66% for the upper triangular portion. For the non-finite element problems, the corresponding percentages are roughly 40% and 50% respectively. Moreover, there are examples in which the percentages of utilization are very low.

(4) One nice feature about the new method is that the amount of space allocated to store the LU-decomposition is not sensitive to the numerical values in A and the row interchanges. In other words, the same amount of space (or the same data structure) can be used for different coefficient matrices as long as they have the same structure. In fact, after the first system has been solved, it is not necessary to determine the data structure again when subsequent systems having the same structure are to be solved. This is not true in MA28 and NSPIV, in which the amount of space depends on the numerical values and the row interchanges (unless one wants to decompose a new matrix having the same structure using the pivotal sequence obtained during the decomposition of the old matrix). To illustrate this, we have generated, for each of problems 1 and 2, three matrices that have different numerical values but the same nonzero structure. The results are given in Table 5.6.

Problem	SPARSPAK	MA28	NSPIV	New method
1	1.23-1	1.53-2	3.16-3	9.44-4
2	2.78-2	4.60-2	6.26-4	8.47-4
3	4.42-1	1.76-1	4.33-3	2.56-3
4	1.15-1	1.31-1	2.75-3	4.31-3
5	1.58+0	7.68-1	1.32-2	8.57-3
6	5.34-1	1.18+0	3.36-3	6.29-3
7	8.66-2	1.09-1	5.81-3	1.62-3
8	1.88+0	5.77+0	4.55-2	4.49-2
9	5.06-2	2.88-1	3.71-3	5.95-3
10	5.76-2	1.67-1	2.16-3	2.76-3
11	2.22+0	1.17+0	7.43-2	1.41-2
12	3.76-1	5.26+0	4.59-3	1.20-2
13	2.25-1	5.95+0	1.19-2	2.66-3
14	1.18+0	5.43-3	1.02-2	1.23-1
15	3.90-3	1.39-3	3.19-4	2.26-3
16	2.93-3	6.03-3	9.70-4	3.72-4
17	5.75+0	5.98-3	1.99-3	1.52-3
18	6.69-1	7.50-1	5.91-1	3.05-1
19	fail	1.30-3	1.90-3	1.27-2
20	4.29-5	7.63-6	6.68-6	1.34-5
21	2.22-4	6.42+0	3.17-4	5.02-4
22	1.30-5	2.43-3	1.56-5	1.16-4

Table 5.4: Errors (in l_∞ norm)

(5) Note that the threshold pivoting technique in MA28 does not necessarily give satisfactory results for some of our test problems. In our experiments, we have assigned the threshold parameter u the value 0.1. Obviously, one can use a larger threshold parameter so as to obtain more accurate results. The tradeoffs are increases in storage requirements and execution times. This is illustrated in Table 5.7 in which we varied the threshold parameter from 0.1 to 1.0. We have used problems 1 and 2 in this experiment. Note the increases in space requirements and execution times when u is increased.

(6) Finally, the amount of space reported for MA28 is the *minimum amount* required in order to solve a given problem. With the minimum amount of space, MA28 has to perform numerous data compressions and consequently requires substantial execution time. Thus, in order to reduce the execution time, it is common practice to provide more space than the minimum amount. The extra space is sometimes known as the *elbow room*. In our experiments, we have provided a lot of elbow room so that data compressions do not occur. To illustrate the effect of data compressions on execution time, we ran MA28 for the first two problems using the minimum amount of space. The results are given in Table 5.8. Note the change in execution times. Also note that the new method and NSPIV do not need any elbow room.

Problem	L	U
1	51.8	62.7
2	49.7	62.1
3	49.7	66.1
4	48.7	66.8
5	50.2	66.2
6	49.9	67.6
7	49.9	66.4
8	49.9	66.6
9	49.7	62.4
10	48.5	63.5
11	50.5	64.8
12	49.9	66.5
13	48.6	65.5
14	20.5	63.4
15	45.1	68.9
16	40.4	65.4
17	40.6	54.5
18	13.1	29.3
19	30.2	42.4
20	40.4	44.2
21	47.1	48.9
22	57.7	66.6

Table 5.5: Data structure utilization by L and U in the new method (percentage)

Method	n	265	265	265	406	406	406
MA28	total storage	21985	22702	20503	37716	40743	37870
	total time	3.113	3.650	2.750	6.663	11.100	8.570
	error	1.53-2	3.70-2	2.11-1	4.60-2	3.34-1	2.60-2
NSPIV	total storage	17326	19146	19906	38747	40743	37870
	total time	3.040	3.043	3.670	9.083	7.057	7.973
	error	3.16-3	1.51-4	3.01-3	6.26-4	5.31-3	1.85-3
New method	analysis storage	10902	10902	10902	17007	17007	17007
	solution storage	18765	18765	18765	35239	35239	35239
	analysis time	0.783	0.783	0.783	1.410	1.410	1.410
	solution time	3.530	3.537	3.557	9.123	9.270	9.287
	total time	4.313	4.320	4.340	10.533	10.680	10.697
	error	9.44-4	3.35-4	6.40-3	8.47-4	6.06-4	5.52-4

Table 5.6: Solution of systems with same nonzero structure.

Problem	u	total storage	total time	error
1	0.1	21985	3.113	1.53-2
1	0.4	22581	3.730	3.31-3
1	0.7	26525	6.390	3.37-4
1	1.0	27365	7.093	2.19-4
2	0.1	37716	6.663	4.60-2
2	0.4	45302	10.920	5.44-4
2	0.7	46044	12.530	5.97-4
2	1.0	46784	18.643	3.89-4

Table 5.7: Effect of varying the threshold parameter in MA28

Problem	total storage	total time (with elbow room)	total time (without elbow room)
1	21985	3.113	8.110
2	37716	6.663	26.126

Table 5.8: Effect of elbow room on the performance of MA28

6. Handling of dense rows

Throughout our discussion in the previous sections, we have assumed that the symmetric matrix A^TA is sparse whenever A is sparse. However, there are some instances in which A^TA is dense even though A is sparse. An example is given below.

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

Clearly A^TA is a dense matrix, but the LU-decomposition of A is as sparse as the original matrix A.

This example illustrates the main disadvantage of the method we propose in this paper. The structures of the Cholesky factors of A^TA may overestimate the structures of the triangular matrices obtained in the LU-decomposition of A. Fortunately this is usually caused by a relatively small number of dense rows of A. (In the previous 4×4 example, the first row is dense.) One way to handle this situation is as follows. For convenience, let A be partitioned into

$$A = \begin{pmatrix} B \\ C \end{pmatrix} ,$$

where B and C contain respectively the sparse and dense rows of A. Assume B is $p\times n$ and C is $(n-p)\times n$. We also assume that B has a "zero-free diagonal"; that is, $B_{ii}\neq0$ for $1\leq i\leq p$. Suppose a sparse LU-decomposition of B is given by

$$B = P_1L_1P_2L_2 \cdots P_{p-1}L_{p-1}\begin{pmatrix} R & S \end{pmatrix} ,$$

where P_k is a $p\times p$ permutation matrix, L_k is $p\times p$ unit lower triangular, R is $p\times p$ upper triangular, and S is $p\times(n-p)$. This can be achieved using the method we have proposed earlier. For simplicity, let

$$\bar{L} = P_1L_1P_2L_2 \cdots P_{p-1}L_{p-1} \ .$$

That is,

$$B = \bar{L}\begin{pmatrix} R & S \end{pmatrix} = \begin{pmatrix} \bar{L}R & \bar{L}S \end{pmatrix} \ .$$

Partition C into

$$C = \begin{pmatrix} C_1 & C_2 \end{pmatrix} \ ,$$

where C_1 and C_2 are respectively $(n-p)\times p$ and $(n-p)\times(n-p)$. Then we have

$$A = \begin{pmatrix} B \\ C \end{pmatrix} = \begin{pmatrix} \bar{L}R & \bar{L}S \\ C_1 & C_2 \end{pmatrix} = \begin{pmatrix} \bar{L} & O \\ O & I \end{pmatrix}\begin{pmatrix} R & S \\ C_1 & C_2 \end{pmatrix} \ .$$

Now we can eliminate C_1 using block elimination. That is, we find an $(n-p)\times p$ matrix V so that

$$\begin{pmatrix} R & S \\ C_1 & C_2 \end{pmatrix} = \begin{pmatrix} I & O \\ V & I \end{pmatrix}\begin{pmatrix} R & S \\ O & W \end{pmatrix} \ .$$

It is not hard to see that $V=C_1R^{-1}$ and $W=C_2-C_1R^{-1}S$. Then we can decompose the $(n-p)\times(n-p)$ matrix W using Gaussian elimination with partial pivoting:

$$W = \hat{P}_1\hat{L}_1\hat{P}_2\hat{L}_2 \cdots \hat{P}_{n-p-1}\hat{L}_{n-p-1}T \ ,$$

where $\hat{P}_k$ is an $(n-p)\times(n-p)$ permutation matrix, $\hat{L}_k$ is $(n-p)\times(n-p)$ unit lower triangular, and T is $(n-p)\times(n-p)$ upper triangular. Let

$$\hat{L} = \hat{P}_1\hat{L}_1\hat{P}_2\hat{L}_2 \cdots \hat{P}_{n-p-1}\hat{L}_{n-p-1} \ .$$

Then

$$W = \hat{L}T \ ,$$

and

$$\begin{pmatrix} R & S \\ O & W \end{pmatrix} = \begin{pmatrix} I & O \\ O & \hat{L} \end{pmatrix}\begin{pmatrix} R & S \\ O & T \end{pmatrix} \ .$$

Combining all identities, we obtain the following decomposition.

$$A = \begin{pmatrix} B \\ C \end{pmatrix} = \begin{pmatrix} \bar{L} & O \\ O & I \end{pmatrix}\begin{pmatrix} I & O \\ V & I \end{pmatrix}\begin{pmatrix} I & O \\ O & \hat{L} \end{pmatrix}\begin{pmatrix} R & S \\ O & T \end{pmatrix} = \begin{pmatrix} \bar{L} & O \\ V & \hat{L} \end{pmatrix}\begin{pmatrix} R & S \\ O & T \end{pmatrix} \ .$$

Let the right-hand side vector b be partitioned into

$$b = \begin{pmatrix} c \\ d \end{pmatrix} \ ,$$

where c and d are respectively p- and $(n-p)$-vectors. Similarly, partition the solution vector x into

$$x = \begin{pmatrix} u \\ v \end{pmatrix} ,$$

where u and v are respectively p- and $(n-p)$-vectors. Then x is obtained by solving

$$\begin{pmatrix} \bar{L} & O \\ V & \hat{L} \end{pmatrix} \begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix} ,$$

and

$$\begin{pmatrix} R & S \\ O & T \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} e \\ f \end{pmatrix} .$$

This approach will be effective if $(n-p)$ is small. In that case, the matrices C, V and W can be stored and processed as small dense matrices.

Suppose A has a zero-free diagonal. In practice, it may not be possible to partition A into

$$\begin{pmatrix} B \\ C \end{pmatrix}$$

such that B contains the sparse rows of A and at the same time has a zero-free diagonal. To illustrate this, consider the following example.

$$A = \begin{pmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{pmatrix} .$$

Thus,

$$B = \begin{pmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{pmatrix}$$

and it does not have a zero-free diagonal. To solve this problem, we use the following approach in our implementation. Let B be the $n \times n$ matrix obtained from A by *replacing* the dense rows by null rows. Thus, in the previous example,

$$B = \begin{pmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{pmatrix} .$$

Then we perform the LU-decomposition on B using Gaussian elimination with partial pivoting, skipping any step where we encounter a zero pivot. The resulting $n \times n$ upper triangular matrix, denoted by U, will be a (row) permuted form of

$$\begin{pmatrix} R & S \\ O & O \end{pmatrix} .$$

Finally, to eliminate C_1, all we have to do is to identify those rows in U whose diagonal elements are nonzero.

7. Concluding remarks

In this paper, we have considered the solution of the sparse linear system $Ax=b$ using Gaussian elimination with partial pivoting. We have proved that the structures of the triangular matrices obtained in the LU-decomposition of the $n \times n$ matrix A are contained in the structures of the Cholesky factors of the symmetric positive definite matrix $A^T A$, regardless of the choice of the row interchanges. These results are important since they allow us to implement Gaussian elimination with partial pivoting using a static data structure which is obtained by analyzing the structure of $A^T A$. The latter can be achieved efficiently using techniques developed for solving sparse symmetric positive definite systems. As a result, the overhead involved in the numerical computation is smaller than that in most existing methods which usually employ dynamic data structures. Preliminary numerical experiments indicate that, in general, the method we have proposed can be quite competitive with existing methods for solving general sparse systems of linear equations.

Clearly, our approach will perform poorly if the matrix $A^T A$ is dense. Fortunately this usually occurs when A has a relatively small number of dense rows. We have also derived an algorithm to cope with this situation.

It is possible to further improve the performance of our method. First, suppose the matrix A is reducible. That is, there exist permutation matrices P and Q so that PAQ is block triangular. For definiteness, assume PAQ is block lower triangular.

$$PAQ = \begin{pmatrix} A_{11} & & & & & \\ A_{21} & A_{22} & & & & \\ A_{31} & A_{32} & A_{33} & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ A_{p1} & A_{p2} & A_{p3} & \cdot\cdot\cdot & & A_{pp} \end{pmatrix} .$$

Here A_{ij} is the (i,j)-block in PAQ. If we partition the right-hand side vector b and the solution vector x conformally,

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \cdot \\ \cdot \\ \cdot \\ b_p \end{pmatrix} , \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_p \end{pmatrix} ,$$

then the solution x is obtained simply by solving

$$A_{kk} x_k = b_k - \sum_{j=1}^{k-1} A_{kj} x_j \ , \qquad k=1,2, \cdots ,p \ .$$

Thus, all we need are the LU-decompositions of the matrices A_{kk} which can be obtained using the approach proposed in this paper. Note that the order of each A_{tt} is smaller than that of A. Consequently, the space required to store the LU-factorization of A_{tt}'s should be smaller than that for A, as long as A is reducible. This idea has been used in MA28 and its incorporation into our scheme is currently under investigation. Note that there are efficient algorithms for permuting a reducible matrix A into block triangular form. See [8, 9] for details.

Finally, note that the actual number of nonzeros obtained in the LU-decomposition of A is usually smaller than the number of nonzeros in the Cholesky factors of A^TA. Thus, an open problem is whether there exists a scheme that would compress the data structure for the Cholesky factors of A^TA so that the compressed structure provides a more efficient data structure to store the LU-factorization, regardless of the row interchanges used.

8. References

[1] I.S. DUFF, "Analysis of sparse systems", D. Phil. Thesis, Oxford University (1972).

[2] I.S. DUFF, "MA28 - A set of FORTRAN subroutines for sparse unsymmetric linear equations", Tech. Report AERE R-8730, Harwell (1977).

[3] I.S. DUFF, "MA32 - A set of FORTRAN subroutines for sparse unsymmetric linear equations", Report AERE R 10079, Harwell (1981).

[4] I.S. DUFF, "On algorithms for obtaining a maximum transversal", *ACM Trans. on Math. Software*, **7** (1981), pp. 315-330.

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

[6] I.S. DUFF, "The design and use of a frontal scheme for solving sparse unsymmetric equations", in *Proceedings of the Third IIMAS Workshop on Numerical Analysis (1981)*, ed. J.P. Hennart, Lecture Notes in Mathematics (909), Springer-Verlag (1982), pp. 240-247.

[7] I.S. DUFF, "MA27 - A set of FORTRAN subroutines for solving sparse symmetric sets of linear equations", Report AERE R 10533, Harwell (1982).

[8] I.S. DUFF AND J.K. REID, "An implementation of Tarjan's algorithm for the block triangularization of a matrix", *ACM Trans. on Math. Software*, **4** (1978), pp. 137-147.

[9] I.S. DUFF AND J.K. REID, "Algorithm 529. Permutations to block triangular form", *ACM Trans. on Math. Software*, **4** (1978), pp. 189-192.

[10] I.S. DUFF AND J.K. REID, "Some design features of a sparse matrix code", *ACM Trans. on Math. Software*, **5** (1979), pp. 18-35.

[11] I.S. DUFF AND J.K. REID, "The multifrontal solution of indefinite sparse symmetric linear systems", Report CSS 122, Harwell (1982).

[12] I.S. DUFF AND J.K. REID, "The multifrontal solution of unsymmetric sets of linear equations", (Submitted to SISCC) (1983).

[13] S.C. EISENSTAT, M.C. GURSKY, M.H. SCHULTZ, AND A.H. SHERMAN, "Yale sparse matrix package, II. the nonsymmetric codes", Research report 114, Dept. of Computer Science, Yale University (1977).

[14] 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.

[15] J.A. GEORGE, M.T. HEATH, AND E.G.Y. NG, "Solution of sparse underdetermined systems of linear equations", (In preparation) (1983).

[16] J.A. GEORGE AND J.W.H. LIU, "The design of a user interface for a sparse matrix package", *ACM Trans. on Math. Software*, **5** (1979), pp. 134-162.

[17] J.A. GEORGE AND J.W.H. LIU, *Computer solution of large sparse positive definite systems*, Prentice-Hall Inc., Englewood Cliffs, N.J. (1981).

[18] A.H. SHERMAN, "Algorithm 533. NSPIV, a FORTRAN subroutine for sparse Gaussian elimination with partial pivoting", *ACM Trans. on Math. Software*, **4** (1978), pp. 391-398.

[19] M. YANNAKKIS, "Computing the minimum fill-in is NP-complete", *SIAM J. Alg. Disc. Meth.*, **2** (1981), pp. 77-79.