IMPROVING THE PERFORMANCE OF SPARSE LU MATRIX FACTORIZATION USING A SUPERNODAL ALGORITHM

Bogdan OANCEA
PhD, Associate Professor, Artifex University, Bucharest, Romania

E-mail: oanceab@ie.ase.ro

Abstract: *In this paper we investigate a method to improve the performance of sparse LU matrix factorization used to solve unsymmetric linear systems, which appear in many mathematical models. We introduced and used the concept of the supernode for unsymmetric matrices in order to use dense matrix operations to perform the LU factorization for sparse matrices. We describe an algorithm that uses supernodes for unsymmetric matrices and we indicate methods to locate these supernodes. Using these ideas we developed a code for sparse LU matrix factorisation. We conducted experiments to evaluate the performance of this algorithm using several sparse matrices. We also made comparisons with other available software packages for sparse LU factorisation.*

Key words: *sparse matrices; linear algebra; LU factorisation*

1. Introduction

The numerical solution for large sparse linear systems lies at the heart of many engineering and scientific applications like macroeconometric models, linear programming, semiconductor device simulations, computational fluid dynamics.

The problem of solving sparse symmetric positive definite systems of linear equations on sequential processors is fairly well understood. Normally, the solution process is performed in two phases:

- First, the matrix A is factorized, $A = LU$ where *L* is a lower triangular matrix with 1s on the main diagonal and *U* is an upper triangular matrix; in the case of symmetric positive definite matrices, we have A=*LL*^{t}.
- Second, we have to solve two linear systems with triangular matrices: $Ly = b$ and $Ux = y$.

While the problem of solving sparse symmetric positive definite systems of linear equations is well understood, for unsymmetric systems it is difficult to design high performance algorithms because the pivoting needed in LU factorization generates dynamic and unpredictable amount of work and intermediate results.

For positive definite systems, the solution is computed in three phases:

- Symbolic factorization to determine the nonzero structure of the Cholseky factor;
- Numeric factorization;

- Solution of two triangular systems;

Elimination trees (Liu, 1990) are the standard way to reduce the time and space for symbolic factorization. For numeric factorization there are two high performance solutions: the supernodal method and the multifrontal method (Duff, 1983). Supernodal and multifrontal methods allow the use of dense vector operations for nearly all of the floating-point computation, thus reducing the symbolic overhead in numeric factorization. Overall, the Megaflop rates of modern sparse Cholesky codes are nearly comparable to those of dense solvers.

2. The generalization of supernodes for unsymmetric LU factorization

For unsymmetric systems, where pivoting is required to maintain numerical stability, the performances of the software packages are below the ones for symmetric systems.

The research has concentrated on two basic approaches for unsymmetric systems: submatrix-based methods and column-based methods.

Submatrix methods typically use some form of Markowitz ordering with threshold pivoting, in which each stage's pivot element is chosen from the uneliminated submatrix by criteria that attempt to balance numerical quality and preservation of sparsity.

Column methods typically use ordinary partial pivoting. The pivot is chosen from the current column according to numerical considerations alone; the columns may be preordered before factorization to preserve sparsity. In column methods, the preordering for sparsity is completely separate from the factorization, just as in the symmetric case. This is an advantage when several matrices with the same nonzero structure but different numerical values must be factored. However, symbolic factorization cannot be separated from numeric factorization, because the nonzero structures of the factors depend on the numerical pivoting choices. Thus column codes must do some symbolic factorization at each stage.

In this paper we describe a solution of sparse LU factorization using a left looking column method with partial pivoting.

In our method, preordering for preserving sparsity is completely separate from the factorization process, but symbolic factorization cannot be separated from numeric factorization because the structure of factors changes dynamically due to the pivoting. In order to speedup the numerical factorization we use a generalized version of supernodes for unsymmetric matrices.

The idea of a supernode is to group together columns with the same nonzero structure, so they can be treated as a dense matrix for storage and computation. Supernodes were originally used for symmetric sparse Cholesky factorization. In the factorization $A = LL^T$ (or $A = LDL^T$), a supernode is a range (r:s) of columns of L with the same nonzero structure below the diagonal; that is, L(r:s; r:s) is full lower triangular and every row of L(s:n; r:s) is either full or zero.

Using the supernodes improves the LU factorisation because of the following (we considered the influence of supernodes on the left-looking LU factorization) (Gilbert, 1993):

1. The inner loop has no indirect addressing and thus sparse Level 1 BLAS is replaced by dense Level 1 BLAS;
2. The outer loop can be unrolled to save memory references and Level 1 BLAS can be replaced by level 2 BLAS operations;

3. Elements of the source supernode can be reused in multiple columns of the destination supernode to reduce cache misses - level 2 BLAS is replaced by Level 3 BLAS.

For a symmetric positive definite matrix, a formal definition of a supernode can be given in terms of elimination tree (Liu, 1990). A supernode is a maximal set of contiguous nodes {j, j+1, … j+w} such that

$$Adj_G(T[j]) = \{j+1, j+2, \ldots, j+w\} \cup Adj_G(T[j+w])$$

In matrix terms, $Adj_G(T[j])$ indicates the nonzero elements in column *j* of the factor *L*. In other words, a supernode is a maximal block of contiguous columns in the Cholesky factor *L* where these columns have identical nonzero structure below the diagonal and the corresponding block diagonal is full triangular.

It is possible to use a generalization of the supernode concept for the unsymmetric matrices. In this case, a supernode is a range *(k:t)* of columns of *L* with the triangular diagonal block full and the same structure below the diagonal block. In figure 1 we have represented a sparse matrix *A* and its factors *F = L + U*. Using the above definition of supernodes, matrix A has 7 supernodes : {1, 2, 3}, {4}, {5}, {6}, {7, 8}, {9}, 10, 11, 12}.

The effect of supernodes in Cholesky factorization is that all updates from columns belonging to a supernode are aggregated into a dense vector before the sparse update of the current column. This process of columns update can be implemented using Level 2 BLAS matrix-vector multiplications. It is even possible to use supernode-supernode update which can be implemented with Level 3 BLAS operations.

$$\begin{array}{c|cccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
1 & x & x & x & & & & & & & & & \\
2 & x & x & & & & & & & & & & \\
3 & x & & x & & & & & & & & & \\
4 & & & & x & & & & & & x & & \\
5 & & & & & x & & & & & & & \\
6 & x & & & & & x & x & & & x & & \\
7 & & & & & & & x & x & & & & \\
8 & x & x & x & & & & x & x & & x & x & x \\
9 & & & & & & x & & & x & & & \\
10 & & & & & & & x & & x & x & & \\
11 & & & & x & & & & & & & x & \\
12 & & & & & & x & & & & & & x \\
\hline
\end{array}$$

A

$$\begin{array}{c|cccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
1 & x & x & x & & & & & & & & & \\
2 & x & x & x & & & & & & & & & \\
3 & x & x & x & & & & & & & & & \\
4 & & & & x & & & & & & x & & \\
5 & & & & & x & & & & & & & \\
6 & x & x & x & & & x & x & & & x & & \\
7 & & & & & & & x & x & & & & \\
8 & x & x & x & & & & x & x & & x & x & x \\
9 & & & & & & x & x & x & x & x & x & x \\
10 & & & & & & & x & x & x & x & x & x \\
11 & & & & x & & & & & & x & x & x \\
12 & & & & & & x & x & x & & x & x & x \\
\hline
\end{array}$$

F

Figure 1. An example of a sparse matrix and its factors *L* and *U*

Supernodes are only useful if they actually occur in practice. The occurrence of symmetric supernodes is related to the clique structure of the chordal graph of the Cholesky factor, which arises because of fill during the factorization. Unsymmetric supernodes are harder to characterize, but they also are related to dense submatrices arising from fill.

In sparse Cholesky factorization, supernodes can be determined during the initial symbolic factorization which is not the case of unsymmetric *LU* factorization where the nonzero structure cannot be determined before numeric factorization. In order to obtain larger supernodes we must permute the columns of *L* to bring toghether columns with the same nonzero structure.

In Cholesky factorization this reordering can de obtained by a postorder traversal of the elimination tree. The postorder etree is used to locate the supernodes before numerical factorization. The analog for etree in the case of an unsymmetric matrix is the *column elimination tree* (Gilbert, 1993). The column etree of A is the symmetric elimination tree of the column intersection graph of A, or equivalently the elimination tree of A^TA provided there is no cancellation in computing A^TA (Gilbert, 1993). The column etree can be computed from A in time almost linear in the number of nonzeros of A by a variation of an algorithm of Liu (Liu, 1990).

It can be shown that column elimination tree represents the dependence between columns of *U* and *L*, and for strong Hall matrices there is no other information obtainable from the structure of the original matrix A (Gilbert, 1993). Before we factor the matrix A we determine its column elimination tree and permute its columns according to a postorder on the tree.

In figure 2 we present a description of the algorithm used for unsymmetric *LU* factorization. The advantage of the supernode-column updates consists in using efficient Level 2 BLAS operations, while in the simple column algorithm there are only Level 1 BLAS operations. The presence of the pivoting process precludes the separation of symbolic from the numeric factorization. Thus, the symbolic factorization for each column is computed just before the numeric factorization of that column. The structure of the column j can be determined by a traversal of the directed graph G associated with matrix $L(:, 1:j-1)^t$ (Gilbert, 1993). The depth-first traversal of G determine the structure of *U(:, j)* and the columns of L that updates the column *j*. At the end of the symbolic factorization of column *j* we can determine very easy if column *j* is part of the same supernode as column *j-1*.

```
for each column j = 1 to n do
      c = A(:,j); c is the current column
      Determine the set S of the supernodes from L which
      updates column j;
      Verify if column j is part of the same supernode as
      column j-1;
      for each supernode in S do
             c(k:t) = L(k:t, k:t)^{-1}c(k:t);
             c(k+1:n)=c(k+1,n)- L(k+1:n, k:t) c(k:t);
      endfor
      find the index i of the pivot, c(i) = max_{i=j:n}c(i)
      swap c(i) and c(j);
      U(1:j,j)= c(1:j), L(j:n) = c(j:n)/c(j);
endfor
```

Figure 2. The supernodal LU factorization algorithm

3. Experimental results

We have implemented the supernodal LU factorization using the C programing language and we have conducted a series of experiments on several sparse matrices. We have used a system with an INTEL CORE 2 processor at 1.6 GHz with 1 GB of main memory. The experimental matrices are from the Harwell Boeing collection. These matrices are presented in table 1 and they are factored using the natural order and using an approximate minimum degree ordering algorithm (Davis, 2002).

Table 1. Test matrices: **n** is the number of lines/columns and **nnz** is the number of nonzero elements

Matrix	n	nnz(A)	nnz(A)/n
SHERMAN3	5005	20033	4
SHERMAN5	3312	20793	6.27
FS7601	760	5976	7.86
FS7602	760	5976	7.86
FS7603	760	5976	7.86
ORSIRR2	886	5970	6.73

In our implementation, we used the ATLAS (Automatically Tuned Linear Algebra Software) library as a high performance implementation of the BLAS.

Table 2 presents the performance of the supernodal LU factorization with natural ordering of the matrix A and table 3 presents the case with matrix A reordered with the approximate minimum degree algorithm. This results show that the approximate minimum degree ordering improves the sparsity of factors as we expected. The ***mflops*** rate of sparse supernodal factorization is about 70% of the dense LU factorization. The dense factorization performance was measured with the standard ATLAS - BLAS implementation and LAPACK package.

Table 2. Performance of supernodal LU factorization using natural ordering

Matrix	MFLOPS	nnz(L)	nnz(U)
SHERMAN3	212.98	548571	548571
SHERMAN5	280.06	409095	998665
FS7601	291.77	201627	207257
FS7602	291.78	207378	208907
FS7603	296.08	207378	208024
ORSIRR2	200.01	66027	79763

Table 3. Performance of supernodal LU factorization using the aproximate minimum degree ordering

Matrix	MFLOPS	nnz(L)	nnz(U)
SHERMAN3	279.57	184086	184086
SHERMAN5	251.80	93357	118436
FS7601	161.9	16535	18206
FS7602	161.08	17094	19479
FS7603	157.80	17532	52003
ORSIRR2	200.01	66027	79763

We compared the performance of the supernodal LU factorization implementation with a multifrontal factorization package – UMFPACK 5.2. UMFPACK uses a multifrontal algorithm. Where the outer loop of a left-looking algorithm like supernodal LU is over columns of the factors being computed, the outer loop of a multifrontal algorithm is over pivots being eliminated. All the updates created when a block is eliminated are computed at once and stored as a dense update matrix. Before a block of pivots is eliminated, all the update matrices contributing to that block are summed into a frontal matrix. The elimination step can use Level 2 or Level 3 BLAS because the arithmetic is carried out on the dense frontal matrix. Some extra intermediate storage is needed to record update matrices that have not yet been assembled into frontal matrices, and some extra data movement is needed for the assembly. UMFPACK does not use a column preordering; rather, it chooses row and column pivots to balance considerations of stability and sparsity by using approximate Markowitz counts with a pivot threshold. In principle, the pivot threshold can lead to a less accurate solution than strict partial pivoting; in practice, the lost accuracy can usually be retrieved by iterative refinement of the solution. In principle, the freedom to choose both row and column pivots dynamically could lead to sparser factors than strict partial pivoting; in practice, some matrices have sparser factors by one method and some by the other.

UMFPACK does not include an initial column ordering step. For the initial column ordering in supernodal LU factorization, we ran the aproximate minimum degree algorithm (Davis, 1997) on the structure of A^TA. We reported times for ordering and factorization separately. In applications where many matrices with the same nonzero structure but different values are factored, the cost of column ordering can be amortized over all the factorizations; in applications where only a single matrix is to be factored, preordering is part of the solution cost.

Table 4 and 5 shows the time needed for LU factorisation using our supernodal algorithm and the UMFPACK. Table 6 shows the memory requirements for the two approaches.

Table 4. Performance of supernodal LU factorization and UMFPACK using natural ordering

Matrix	Supernodal Factorization (msec)	UMFPACK Factorization (msec)
SHERMAN3	560	710
SHERMAN5	670	780
FS7601	48	57
FS7602	51	55
FS7603	45	72
ORSIRR2	54	53

Table5. Performance of supernodal LU factorization and UMFPACK using the aproximate minimum degree ordering

Matrix	Supernodal Factorization (msec)	UMFPACK Factorization (msec)
SHERMAN3	832	1002
SHERMAN5	932	1210
FS7601	75	89
FS7602	78	92
FS7603	70	81
ORSIRR2	80	93

Table 6. Memory requirements for LU factorization using the supernodal approach and UMFPACK.

Matrix	Supernodal Factorization (MB)	UMFPACK Factorization (MB)
SHERMAN3	3.67	5.87
SHERMAN5	3.83	6.01
FS7601	1.22	2.23
FS7602	1.25	2.32
FS7603	1.22	2.24
ORSIRR2	1.44	2.98

It can be observed from this figures that the supernodal approach performs slightly better for five from the six matrices used for tests. Although there is not a clear difference betwenn two aproaches, the supernodal algorithm can be used with succes for large sparse systems.

The supenodal LU factorization algorithm can be improved further on machines with a memory hierarchy by changing the data access pattern. The data we are accessing in the inner loop include the destination column j and all the updating supernodes to the left of column j. Column j is accessed many times, while each supernode is used only once. In practice, the number of nonzero elements in column j is much less than that in the updating supernodes. Therefore, the access pattern given by this loop provides little opportunity to reuse cached data. In particular, the same supernode may be needed to update both columns j and j+1. But when we factor the (j+1)st column, we will have to fetch the same supernode again from memory, instead of from cache (unless the supernodes are small compared to the cache). To exploit memory locality, we factor several columns (say s of them) at a time in the outer loop, so that one updating supernode can be used to update as many of the s columns as possible.

4. Conclusions

In this paper we presented an algorithm for sparse LU factorization using a generalization of the supernode concept. The supernode for the unsymmetric matrices is a range (*k:t*) of columns of *L* with the triangular diagonal block full and the same structure below the diagonal block. We developed an algorithm for matrix factorization using supernodes based on the clasical left-looking LU factorization.

We implemented this algorithm in a software package using the C programming language and ATLAS library as a high performance implementation of BLAS. We tested our implementation using some sparse matrices from the Harwell-Boeing collection. The results of the tests shows that the sparse LU factorization using supernodes can achieve about 70% of the performance for the dense LU factorization.

We also compared the sparse LU factorization using supernodes with a multifrontal factorization package –UMFPACK. The results shows that the supernodal approach performs slightly better than the multiforntal approach.

References

1. Davis, T.A., Gilbert, J.R., Larimore, S.I. and Ng, E. **Approximate minimum degree ordering for unsymmetric matrices,** SIAM Symposium on Applied Linear Algebra, October, 1997
2. Davis, T.A. **A column pre-ordering strategy for unsymmetric-pattern multifrontal method,** Technical Report TR-02-001, University of Florida, January, 2002
3. Demmel, W.D., Eisenstat, S.C., Gilbert, J.R., Li, X.S. and Liu, J.W.H. **A supernodal approach to sparse partial pivoting,** SIAM J. Matrix, Anal. Appl. Vol 20, no. 3, 1999
4. Duff, I., and Reid, J. **The multifrontal solution of indefinite sparse symmetric linear equations***,* ACM Trans. Mathematical Software, 9, 1983
5. Eisenstat, S.C., Gilbert, J.R. and Liu, W. **A supernodal approach to sparse partial pivoting code***,* Householder Symposium XII, 1993

6. George, J.A. and Liu, J. W. H. **Computer Solution of Large Sparse Positive Definite Systems**, Prentice-Hall, Englewood Cliffs, 1981
7. Golub, G. H. and Van Loan, Ch., **Matrix Computations**, Third Edition, The Johns Hopkins University Press, 1996
8. Liu, J. W. H. **The role of elimination trees in sparse factorization**, SIAM J. Matrix Analysis and applications, 11,1990, pp. 134-172
9. Gilbert, J. R. and Ng, E. **Predicting structure in nonsymmetric sparse matrix factorizations**, in George, A., Gilbert, J.R. and Liu, J. W. H. (editors) "Graph Theory and Sparse Matrix Computation", Springer-Verlag, 1993
10. Gilbert, J. R., and Liu, J. W. H. **Elimination structures for unsymmetric sparse LU factors**, SIAM J. Matrix Anal. Appl., 14, 1993
11. Grigori, L. and Li, X.S. **Performance Analysis of Parallel Right-Looking Sparse LU Factorization on Two Dimensional Grids of Processors**, Para'04 workshop on state-of-the-art in scientific computing, 2004
12. ***, http://math-atlas.sourceforge.net/