ornl

ORNL/TM-11836

OAK RIDGE NATIONAL LABORATORY

MARTIN MARIETTA

Supernodal Symbolic Cholesky Factorization on a Local-Memory Multiprocessor

Esmond Ng

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

ORNL/TM-11836

Engineering Physics and Mathematics Division

Mathematical Sciences Section

SUPERNODAL SYMBOLIC CHOLESKY FACTORIZATION ON A LOCAL-MEMORY MULTIPROCESSOR

Esmond Ng

Mathematical Sciences Section
Oak Ridge National Laboratory
P.O. Box 2007, Bldg. 6012
Oak Ridge, TN 37831-6367
(esmond@msr.epm.ornl.gov)

DATE PUBLISHED: June 1991

Research was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
managed by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC-05-84OR21400

Contents

SUPERNODAL SYMBOLIC CHOLESKY FACTORIZATION ON A LOCAL-MEMORY MULTIPROCESSOR

Esmond Ng

Abstract

In this paper, we consider the symbolic factorization step in computing the Cholesky factorization of a sparse symmetric positive definite matrix on distributed-memory multiprocessor systems. By exploiting the supernodal structure in the Cholesky factor, the performance of a previous parallel symbolic factorization algorithm is improved. Empirical tests demonstrate that there can be drastic reduction in the execution time required by the new algorithm on an Intel iPSC/2 hypercube.

1. Introduction

Let $\boldsymbol{A}$ be a large sparse symmetric positive definite matrix of order n and $\boldsymbol{b}$ be an n-vector. Consider the solution of the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ using Cholesky factorization. Denote the Cholesky factor of $\boldsymbol{A}$ by $\boldsymbol{L}$. It is often desirable to determine the *structure* of $\boldsymbol{L}$ before computing it numerically, since the information allows a data structure to be set up prior to the numerical factorization. Then numerical factorization can proceed with a fixed storage structure. The determination of the structure of $\boldsymbol{L}$ is often called the *symbolic factorization* of $\boldsymbol{A}$. In this note, we are concerned with computing the structure of $\boldsymbol{L}$ on a multiprocessor system in which each processor has its own private memory.

In [8], an algorithm was proposed for performing the symbolic factorization step on a local-memory multiprocessor system. The goal of this paper is to describe an improvement to that algorithm by exploiting the supernodal structure in the Cholesky factor. Preliminary numerical experiments on a hypercube indicate that the improvement leads to more than 50% reduction in the time required by the symbolic factorization step for matrices of order greater than 5000 on 16 or more processors.

An outline of the paper is as follows. In Section 2, a symbolic factorization algorithm for serial machines is presented. The parallel version of the sequential algorithm from [8] and the improved algorithm are described in Sections 3 and 4, respectively. Some numerical experiments and concluding remarks are provided in Section 5.

2. A sequential symbolic factorization algorithm

Throughout this paper, we will use $Struct[\boldsymbol{M}, k]$ to denote the set of row indices of the nonzeros in column k of the lower triangular part of the matrix $\boldsymbol{M}$. That is,

$$Struct[\boldsymbol{M}, k] = \{i > k : \boldsymbol{M}_{ik} \neq 0\}.$$

Consider the Cholesky factor $\boldsymbol{L}$ of a symmetric and positive definite matrix $\boldsymbol{A}$. When $Struct[\boldsymbol{L}, k] \neq \emptyset$, we define $f(k)$ to be the row index of the *first* off-diagonal nonzero in column k of $\boldsymbol{L}$. If $Struct[\boldsymbol{L}, k] = \emptyset$, we let $f(k) = k$. Using this notation, the structure of column k of $\boldsymbol{L}$ can be characterized as follows [22]:

$$Struct[\boldsymbol{L}, k] = Struct[\boldsymbol{A}, k] \cup \left(\bigcup_{\substack{i<k \\ f(i)=k}} Struct[\boldsymbol{L}, i] \right) - \{k\}. \tag{2.1}$$

That is, the structure of column k of $\boldsymbol{L}$ is given by the structure of column k of $\boldsymbol{A}$ (excluding the portion above the diagonal), together with the structures of those columns of $\boldsymbol{L}$ whose first off-diagonal nonzeros are in row k. An example demonstrating the result is provided in Figure 2.1. The structure of column 4 of $\boldsymbol{L}$ is given by the union of the structure of column 4 of $\boldsymbol{A}$ and the structures of columns 2 and 3 of $\boldsymbol{L}$.

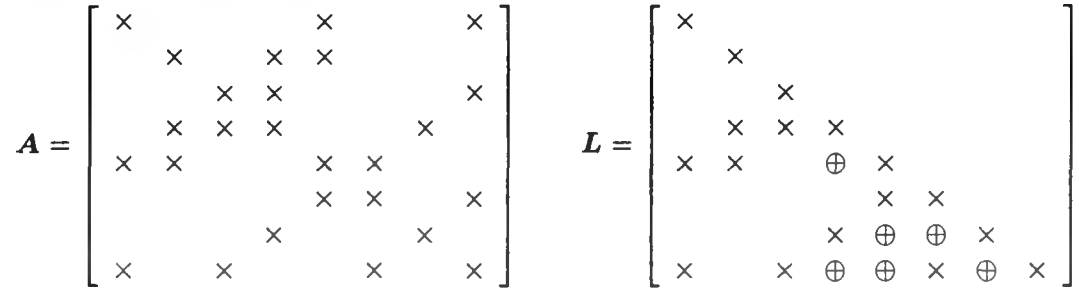

Figure 2.1: The structure of a matrix and its Cholesky factor. ($\times$ denotes a nonzero and $\oplus$ denotes a fill due to factorization.)

An algorithm for computing the structure of $\boldsymbol{L}$ can be formulated using Equation (2.1) and is presented in Figure 2.2. In the algorithm, the set $\mathcal{R}_k$ is used to record

```
for k = 1 to n do
    Set R_k ← ∅.
end for
for k = 1 to n do
    Set Struct[L, k] ← Struct[A, k].
    for i ∈ R_k, do
        Set Struct[L, k] ← Struct[L, k] ∪ Struct[L, i] − {k}.
    end for
    Determine f(k)
    if f(k) > k, set R_f(k) ← R_f(k) ∪ {k}.
end for
```

Figure 2.2: A sequential symbolic factorization algorithm.

the columns of $\boldsymbol{L}$ whose first off-diagonal nonzeros are in row k. It is constructed during the execution of the algorithm. When $Struct[\boldsymbol{L}, k]$ has been computed, k is added to the set $\mathcal{R}_{f(k)}$ to indicate that column k of $\boldsymbol{L}$ is needed to compute the structure of $f(k)$ of $\boldsymbol{L}$. This symbolic factorization algorithm can be implemented efficiently; see [12]

for a detailed discussion. Efficient implementations of the sequential algorithm can be found in SPARSPAK [4,11] and the Yale Sparse Matrix Package [6].

It is worth noting that the set of indices $\{f(1), f(2), \cdots, f(n)\}$ plays an important role in sparse matrix computations. Define the graph $\mathcal{T}$ as follows. Let $\{1, 2, \cdots, n\}$ be the vertex set of $\mathcal{T}$, and let there be an edge between i and j in $\mathcal{T}$ if and only if $j = f(i)$ and $j \neq i$. It is easy to verify that $\mathcal{T}$ is a collection of trees, which is referred to as the *elimination tree* or *elimination forest* of $\boldsymbol{L}$ [16,21]. The elimination tree associated with the Cholesky factor in Figure 2.1 is depicted in Figure 2.3. There is exactly one

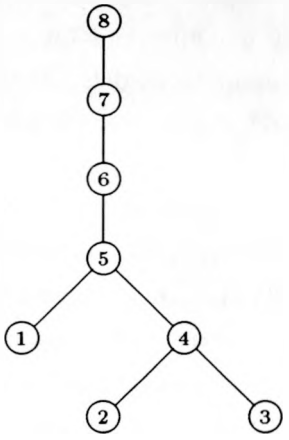

Figure 2.3: The elimination tree associated with the Cholesky factor in Figure 2.1.

tree in $\mathcal{T}$ if and only if the matrix $\boldsymbol{A}$ is *irreducible*. When $\boldsymbol{A}$ is reducible, it is possible to permute the rows and columns of $\boldsymbol{A}$ symmetrically so that the permuted matrix is block diagonal. In this case, each tree in $\mathcal{T}$ corresponds to a diagonal block in the permuted matrix. Thus, without loss of generality, we will assume from now on that the given matrix $\boldsymbol{A}$ is irreducible, so that $\mathcal{T}$ has exactly one tree.

In the elimination tree $\mathcal{T}$, n is the only vertex such that $f(n) = n$ and it is referred to as the *root*. Moreover, given any vertex i in $\mathcal{T}$, there is a unique path between i and n. If k is a vertex on the path joining i and n, then k is an *ancestor* of i and i is a *descendant* of k. In particular, if $k = f(i)$, k is the *parent* of i and i is a *child* of k. Thus, at step k of the symbolic factorization algorithm, the members of $\mathcal{R}_k$ are exactly the children of vertex k in $\mathcal{T}$. Finally, although the elimination tree is defined in terms of the structure of $\boldsymbol{L}$, it can in fact be computed from the structure of $\boldsymbol{A}$. An efficient algorithm is given in [16]. A parallel implementation of the algorithm on a

distributed-memory machine can be found in [23].

3. A parallel symbolic factorization algorithm

The solution of a sparse symmetric positive definite system typically involves several stages, and it is often the case that the numerical factorization and the symbolic factorization are, respectively, the most and the least expensive phases. Thus, much effort has been spent on parallelizing the numerical factorization phase. There are, however, reasons for parallelizing symbolic factorization, particularly on local-memory multiprocessor systems, even though the resulting parallel symbolic factorization algorithm may not be much faster than its sequential counterpart. The most compelling reason is that, on a distributed-memory machine and for large problems, there may not be enough memory on a single processor to hold the entire problem to perform the symbolic factorization sequentially. As the problem is partitioned and distributed among the processors in a local-memory multiprocessor, it is natural to develop as efficient an algorithm as possible to perform the symbolic factorization on such architectures.

In [8], a parallel version of the symbolic factorization algorithm described in the previous section was developed for distributed-memory multiprocessor systems. It is assumed that the columns of the matrix $\boldsymbol{A}$ and its Cholesky factor $\boldsymbol{L}$ are distributed among the processors according to some predetermined mapping strategy. As the numerical factorization tends to be the most time-consuming phase in the solution of a sparse linear system, the mapping is often chosen in an attempt to minimize the factorization time by reducing the amount of communication required and balancing the load among the processors during numerical factorization. Detailed discussion of the mapping issue can be found in [9]. In this paper, we will use $map[k]$ to denote the processor to which column k of $\boldsymbol{L}$ is assigned. Naturally, we assume that column k of $\boldsymbol{A}$ is also assigned to processor $map[k]$. In performing the symbolic factorization on a local-memory multiprocessor, the structure of column k of $\boldsymbol{L}$ has to be made available to processor $map[f(k)]$ when it has been computed. If $map[f(k)] \neq map[k]$, this will result in a message (containing $Struct[\boldsymbol{L}, k]$) being sent from processor $map[k]$ to processor $map[f(k)]$ on most of the local-memory multiprocessor systems available today. In Figure 3.1, we summarize the parallel algorithm in [8]. The parallel algorithm will be executed on each processor.

In the algorithm, $smod[k]$ is the number of structure modifications that have to be applied to column k. Since $smod[k]$ is the same as the number of children of vertex k in the elimination tree, it can be computed by traversing $\mathcal{T}$ once before the symbolic factorization proceeds. Here we assume that $\mathcal{T}$ is computed before the start of symbolic factorization, for example, using the algorithm from [23]. Two communication

primitives are used: **send** for sending a message from one processor to another processor and **recv** for receiving a message. The algorithm in Figure 3.1 is *data-driven*, since the data is made available to another processor once the data is generated. A detailed description of the parallel algorithm can be found in [8].

---

for each column, say column k, of $\boldsymbol{A}$ assigned to this processor **do**
  Set $Struct[\boldsymbol{L}, k] \leftarrow Struct[\boldsymbol{A}, k]$.
  if $smod[k] = 0$ **then**
    if $|Struct[\boldsymbol{L}, k]| \geq 1$ **then**
      Determine $f(k)$.
      send $Struct[\boldsymbol{L}, k]$ to processor $map[f(k)]$.
    end if
  end if
end for
while there are columns of $\boldsymbol{L}$ to be computed in this processor **do**
  recv $Struct[\boldsymbol{L}, i]$, for some i (defined in the message).
  Determine $f(i)$.
  Set $Struct[\boldsymbol{L}, f(i)] \leftarrow Struct[\boldsymbol{L}, f(i)] \cup Struct[\boldsymbol{L}, i] - \{f(i)\}$.
  Decrement $smod[f(i)]$ by 1.
  if $smod[f(i)] = 0$ **then**
    if $|Struct[\boldsymbol{L}, f(i)]| \geq 1$ **then**
      Determine $f(f(i))$.
      send $Struct[\boldsymbol{L}, f(i)]$ to processor $map[f(f(i))]$.
    end if
  end if
end while

Figure 3.1: A parallel symbolic factorization algorithm for distributed-memory multiprocessor systems.

---

4. An improvement to the parallel symbolic factorization algorithm

It is often the case that multiple columns in the Cholesky factor $\boldsymbol{L}$ share the same sparsity structure. Such a grouping of columns is referred to as a *supernode*. To be more precise, $K = \{s_1, s_2, \cdots, s_m\}$, with $s_1 < s_2 < \cdots < s_m$, is a supernode if and only if $Struct[\boldsymbol{L}, s_i] = Struct[\boldsymbol{L}, s_m] \cup \{s_{i+1}, \cdots, s_m\}$, for $1 \leq i \leq m-1$. As an example, columns 5-8 of the Cholesky factor $\boldsymbol{L}$ in Figure 2.1 form a supernode and each of the first four columns of $\boldsymbol{L}$ is in a supernode of size one. The notion of supernodes (and its variants) has been used extensively in sparse matrix computations [1,3,5,13,15,19,

20,22]. The set of supernodes can sometimes be identified in the reordering phase. For example, the set of indistinguishable nodes in the minimum degree algorithm [13] or a minimal separator in the nested dissection algorithm [10] forms a supernode in $\boldsymbol{L}$. Alternatively, the algorithm in [18] can be used to compute the supernode partitioning.

Without loss of generality and for convenience, we assume that columns in the same supernode are numbered consecutively. Such supernodes can be obtained by computing a postordering of the elimination tree [17]. (See [18] for more discussion on the numbering of columns in a supernode.) Moreover, we assume that the supernodes in $\boldsymbol{L}$ are *fundamental* supernodes [2]. Let $K = \{j, j+1, \cdots, j+r-1\}$ be a supernode. Then K is a fundamental supernode if it is a maximal contiguous column subset such that $j+i-1$ is the *only* child of $j+i$ in the elimination tree, for $1 \leq i \leq r-1$.

The improvement to the parallel symbolic factorization algorithm in Figure 3.1 is obtained by exploiting the supernodal structure of the Cholesky factor. Since the columns in the same supernode share basically the same structure, it is sufficient to compute the structure of the *first* column in each supernode. This observation is actually exploited in existing sequential symbolic factorization algorithms [12,22].

We can exploit the observation made above in the parallel setting as well. Let $K = \{j, j+1, \cdots, j+r-1\}$ be a fundamental supernode in $\boldsymbol{L}$. We use the notation $f(K)$ to stand for $f(j+r-1)$. Suppose $Struct[\boldsymbol{L}, j]$ has been computed by processor $map[j]$. For the parallel algorithm in Figure 3.1, $Struct[\boldsymbol{L}, j]$ will be sent to processor $map[f(j)] = map[j+1]$ (due to the way in which columns in a supernode are numbered and the fact that columns j and $j+1$ are in the same supernode) so that processor $map[j+1]$ can compute $Struct[\boldsymbol{L}, j+1]$. In particular, processor $map[f(j+r-1)]$ would not be able to finish computing $Struct[\boldsymbol{L}, f(j+r-1)]$ until $Struct[\boldsymbol{L}, j+r-1]$ has been computed by processor $map[j+r-1]$. However, since columns $j+1, \cdots, j+r-1$ are in the supernode containing column j, there is no need to *compute* $Struct[\boldsymbol{L}, j+i]$, for $1 \leq i \leq r-1$; $Struct[\boldsymbol{L}, j+i]$ is simply given by $Struct[\boldsymbol{L}, j] - \{j+1, \cdots, j+i\}$. Thus, processor $map[f(j+r-1)]$ does not have to wait for $Struct[\boldsymbol{L}, j+r-1]$; it really needs $Struct[\boldsymbol{L}, j]$. However, as the columns belonging to the same supernode are generally assigned to different processors, processor $map[j+i]$ still needs to receive $Struct[\boldsymbol{L}, j]$ from processor $map[j]$, even though no structure computation is required for column $j+i$, for $1 \leq i \leq r-1$. Because of this observation, we will distinguish between two types of messages: primary and secondary.

When $Struct[\boldsymbol{L}, j]$ has been computed by processor $map[j]$, it is clearly desirable to send the structure to processor $map[f(j+r-1)]$ first, so that processor $map[f(j+r-1)]$ can proceed with the computation of $Struct[\boldsymbol{L}, f(j+r-1)]$. From the definition of fundamental supernodes, it should be clear that column $f(j+r-1)$ (i.e., $f(K)$) must be the first column of some fundamental supernode K' in $\boldsymbol{L}$. The message sent from

the first column of a supernode to the first column of another supernode is referred to as a *primary* message.

After sending the structure of column j to processor $map[f(j+r-1)]$, processor $map[j]$ sends $Struct[\boldsymbol{L}, j]$ to processors $map[j+i]$, where $1 \leq i \leq r-1$, with the understanding that only one copy of $Struct[\boldsymbol{L}, j]$ should be sent to a processor even if several columns from the same supernode are assigned to it. Messages sent from the first column of a supernode to other columns in the same supernode are referred to as *secondary* messages.

It is important for a processor to consume as many primary messages as it can before considering any secondary messages, since this will allow the structure of the Cholesky factor to be computed as soon as possible. A processor will consume the secondary messages only when no primary messages are available in the message queue. An improved parallel symbolic factorization algorithm that exploits the supernodal structure is given in Figures 4.1 and 4.2. In the algorithm, we make use of an additional communication primitive `iprobe(type)`, which is used to check if there is any message of type `type` waiting in the message queue.

In the description of the algorithm, the notation $smod[K]$ denotes the number of children of vertex j in the elimination tree, where j is the first column in K. Thus, $smod[K]$ is the number of structure updates that supernode K will expect. The number of fundamental supernodes in $\boldsymbol{L}$ is denoted by N. Moreover, the set $\mathcal{R}_K$ records the supernodes J such that k_f and j_f are assigned to the same processor, where k_f and j_f denote, respectively, the first columns of K and J. That is, $\mathcal{R}_K$ keeps track of *local* structure modifications that supernode K expects to receive. The variable *myid* refers to the processor number of the processor executing the algorithm.

Finally, the variable *ktrol* in Figure 4.2 is used to control the maximum number of secondary messages a processor will process before looking for primary messages again; it is set to 3 in Figure 4.2. Intuitively, a large value for *ktrol* implies that a processor may process more secondary messages between the processing of two primary messages. This may cause delay in computing the structures of the first columns of the supernodes. On the other hand, a small value for *ktrol* means that each processor will give priority to the primary messages. However, for the problems in our numerical experiments, we have found that the performance of the improved parallel symbolic factorization algorithm is not very sensitive to the choice of *ktrol*. This suggests that the queues for the primary messages tend to be non-empty, so that the processors will handle them first before examining the secondary message queues. In any case, in the experiments reported in Section 5, *ktrol* was set to 3.

```
{The following algorithm is to be executed on each processor.}
for each supernode K = 1 to N do
    Set R_K ← ∅.
end for
for each supernode K = 1 to N do
    Let k_f and k_l be the first and the last columns in supernode K, respectively.
    if map[k_f] = myid then
        Struct[L, K] ← Struct[A, K].
        for I ∈ R_K do
            Struct[L, K] ← Struct[L, I] − {1, 2, ···, k_f}.
            Decrement smod[K].
        end for
        if smod[K] ≠ 0 then
            perform external updates (see Figure 4.2).
        end if
        if k_l is not the root of the elimination tree then
            Let j_f be the parent of k_l in the elimination tree.
            Suppose j_f is in supernode J.
            if map[j_f] ≠ map[k_f] then
                send primary message of type J to map[j_f] containing Struct[L, K].
            else
                R_J ← R_J ∪ {K}
            end if
            for i ∈ K and i ≠ k_f do
                if map[i] ≠ map[k_f] then
                    send secondary message to map[i] containing Struct[L, K].
                end if
            end for
            for i ∈ K and i ≠ k_f do
                if map[i] = map[k_f] then
                    Set up pointer information for the structure of column i
                end if
            end for
        end if
    end if
end for
while there are more secondary messages to arrive do
    recv secondary message from supernode K
    Set up pointer information for columns in K
end while
```

Figure 4.1: An improved parallel symbolic factorization algorithm for distributed-memory multiprocessor systems that exploits the supernodal structure.

```
External updates for supernode K:
while true do
    while iprobe(K) > 0 do
        { Process primary messages. }
        recv Struct(L, I), for some supernode I (defined in the message).
        Struct(L, K) ← Struct(L, K) ∪ Struct(L, I) − {1, 2, ···, k_f}.
        Decrement smod[K].
        if smod[K] = 0 then exit from external updates.
    end while
    ktrol ← 3.
    while ktrol > 0 and iprobe(secondary) > 0 do
        { Process secondary messages. }
        recv Struct(L, I), for some supernode I (defined in the message).
        Set up pointer information for columns in I
        ktrol ← ktrol − 1.
    end while
end while
```

Figure 4.2: Procedure "External updates".

5. Numerical experiments and concluding remarks

In this section, we present the results of some preliminary numerical experiments comparing the improved algorithm described in this section with the parallel algorithm in [8]; these two algorithms are referred to as the `new` and `old` algorithms, respectively, in the tables. All experiments were performed on an Intel iPSC/2. The programs were written in Fortran and compiled with optimization turned on.

There were two sets of test problems. The first set contains a sequence of matrices, each of which is obtained by applying a nine-point operator to a $k \times k$ grid ordered by the nested dissection algorithm [7]. That is, $n = k^2$. The second set contains matrices obtained from triangulations of an L-shaped domain as illustrated in [10]. The mesh points were ordered using a parallel version of an automatic nested dissection algorithm [9,10] The columns of $\boldsymbol{A}$ and $\boldsymbol{L}$ are assigned to the processors using the subtree-to-subcube mapping [14], which is known to reduce communication and balance the load, particularly for the numerical factorization phase. See [8,14] for details.

The timing statistics are provided in Tables 5.1 and 5.2. The improvement due to the exploitation of the supernodal structure in the Cholesky factor is obvious. The large reduction in the time required to perform symbolic factorization using the new algorithm comes from two sources. First, by processing the primary messages first, the

n	$\|A\| - n$	method	$p = 8$	$p = 16$	$p = 32$	$p = 64$
900	6844	**new**	.036	.038	.050	.038
		old	.055	.052	.056	.056
1225	9384	**new**	.044	.039	.045	.050
		old	.070	.066	.070	.070
1600	12324	**new**	.055	.047	.053	.053
		old	.088	.081	.083	.084
2025	15664	**new**	.071	.054	.058	.061
		old	.112	.103	.101	.102
2500	19404	**new**	.085	.063	.064	.073
		old	.137	.120	.121	.121
3025	23544	**new**	.099	.074	.068	.079
		old	.158	.138	.136	.134
3600	28084	**new**	.115	.080	.075	.084
		old	.183	.156	.155	.153
4225	33024	**new**	.134	.099	.095	.122
		old	.216	.185	.179	.177
4900	38364	**new**	.148	.103	.091	.097
		old	.246	.208	.200	.197
5625	44104	**new**	.170	.115	.096	.107
		old	.277	.234	.223	.219

Table 5.1: Time in seconds for new and old parallel symbolic factorization algorithms for $k \times k$ grid problems.

new algorithm attempts to compute the structures of the first columns of the supernodes as soon as possible. Second, since the structures of the columns in a supernode are given essentially by the structure of the first column in the same supernode, there is no need to compute the structure of every column in a supernode. Thus, the new algorithm has avoided some redundant computation by exploiting the supernodal structure and consequently it further reduces the time required to compute the structure of a Cholesky factor.

n	$\|A\| - n$	method	$p = 8$	$p = 16$	$p = 32$	$p = 64$
1009	5856	new	.038	.037	.047	.066
		old	.073	.075	.079	.081
1270	7398	new	.045	.039	.048	.050
		old	.089	.090	.096	.097
1561	9120	new	.053	.047	.051	.055
		old	.106	.105	.108	.110
1882	11022	new	.062	.051	.055	.067
		old	.124	.122	.124	.126
2233	13104	new	.071	.055	.068	.066
		old	.145	.140	.144	.145
2614	15366	new	.084	.064	.065	.076
		old	.201	.192	.193	.195
3025	17808	new	.093	.075	.069	.079
		old	.228	.218	.218	.222
3466	20430	new	.107	.080	.078	.088
		old	.257	.238	.234	.238
3937	23232	new	.118	.087	.080	.093
		old	.286	.266	.266	.265
4438	26214	new	.131	.096	.084	.099
		old	.326	.304	.298	.301
4969	29376	new	.149	.104	.090	.106
		old	.358	.325	.320	.325
5530	32718	new	.164	.119	.105	.108
		old	.390	.368	.356	.357
6121	36240	new	.178	.122	.111	.115
		old	.430	.384	.375	.376

Table 5.2: Time in seconds for new and old parallel symbolic factorization algorithms for a sequence of L-shaped problems.

6. References

[1] C. Ashcraft, S.C. Eisenstat, J.W.H. Liu, B.W. Peyton, and A.H. Sherman. A compute-ahead implementation of the fan-in sparse distributed factorization scheme. Technical Report ORNL/TM-11496, Oak Ridge National Laboratory, Oak Ridge, TN, 1990.

[2] C. Ashcraft and R. Grimes. The influence of relaxed supernode partitions on the multifrontal method. *ACM Trans. Math. Software*, 15:291–309, 1989.

[3] C.C. Ashcraft, R.G. Grimes, J.G. Lewis, B.W. Peyton, and H.D. Simon. Progress in sparse matrix methods for large linear systems on vector supercomputers. *Internat. J. Supercomp. Appl*, 1:10–30, 1987.

[4] E.C.H. Chu, A. George, J. W-H. Liu, and E. G-Y. Ng. User's guide for SPARSPAK-A: Waterloo sparse linear equations package. Technical Report CS-84-36, University of Waterloo, Waterloo, Ontario, 1984.

[5] I.S. Duff and J.K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Trans. Math. Software*, 9:302–325, 1983.

[6] S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A. H. Sherman. The Yale sparse matrix package I. the symmetric codes. *Internat. J. Numer. Meth. Engrg.*, 18:1145–1151, 1982.

[7] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.

[8] A. George, M.T. Heath, J. W-H. Liu, and E. G-Y. Ng. Symbolic Cholesky factorization on a local-memory multiprocessor. *Parallel Computing*, 5:85–95, 1987.

[9] A. George, M.T. Heath, J. W-H. Liu, and E. G-Y. Ng. Solution of sparse positive definite systems on a hypercube. *J. Comp. Appl. Math.*, 27:129–156, 1989.

[10] A. George and J. W-H. Liu. An automatic nested dissection algorithm for irregular finite element problems. *SIAM J. Numer. Anal.*, 15:1053–1069, 1978.

[11] A. George and J. W-H. Liu. The design of a user interface for a sparse matrix package. *ACM Trans. Math. Software*, 5:134–162, 1979.

[12] A. George and J. W-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.

[13] A. George and J. W-H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989.

[14] A. George, J. W-H. Liu, and E. G-Y. Ng. Communication results for parallel sparse Cholesky factorization on a hypercube. *Parallel Computing*, 10:287–298, 1989.

[15] J.G. Lewis, B.W. Peyton, and A. Pothen. A fast algorithm for reordering sparse matrices for parallel factorization. *SIAM J. Sci. Stat. Comput.*, 10:1156–1173, 1989.

[16] J. W-H. Liu. A compact row storage scheme for Cholesky factors using elimination trees. *ACM Trans. Math. Software*, 12:127–148, 1986.

[17] J. W-H. Liu. The role of elimination trees in sparse factorization. *SIAM J. Matrix Anal. Appl.*, 11:134–172, 1990.

[18] J.W.H. Liu, E. Ng, and B.W. Peyton. On finding supernodes for sparse matrix computations. Technical Report ORNL/TM-11563, Oak Ridge National Laboratory, Oak Ridge, TN, 1990.

[19] E. Ng and B. Peyton. A supernodal Cholesky factorization algorithm for shared-memory multiprocessors. Technical Report ORNL/TM-11814, Oak Ridge National Laboratory, Oak Ridge, TN, 1991.

[20] E. Rothberg and A. Gupta. Fast sparse matrix factorization on modern workstations. Technical Report STAN-CS-89-1286, Stanford University, Stanford, California, 1989.

[21] R. Schreiber. A new implementation of sparse Gaussian elimination. *ACM Trans. Math. Software*, 8:256–276, 1982.

[22] A.H. Sherman. On the efficient solution of sparse systems of linear and nonlinear equations. Technical Report 46, Dept. of Computer Science, Yale University, 1975.

[23] E. Zmijewski and J.R. Gilbert. A parallel algorithm for sparse symbolic Cholesky factorization on a multiprocessor. *Parallel Computing*, 7:199–210, 1988.

INTERNAL DISTRIBUTION

1. B. R. Appleton
2–3. T. S. Darland
4. E. F. D'Azevedo
5. J. J. Dongarra
6. G. A. Geist
7. E. R. Jessup
8. M. R. Leuze
9–13. E. G. Ng
14. C. E. Oliver
15. B. W. Peyton
16–20. S. A. Raby
21. C. H. Romine
22. T. H. Rowan
23–27. R. F. Sincovec
28–32. R. C. Ward
33. P. H. Worley
34. Central Research Library
35. ORNL Patent Office
36. K-25 Plant Library
37. Y-12 Technical Library / Document Reference Station
38. Laboratory Records - RC
39–40. Laboratory Records Department

EXTERNAL DISTRIBUTION

41. Cleve Ashcraft, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346

42. Donald M. Austin, 6196 EECS Bldg., University of Minnesota, 200 Union St., S.E., Minneapolis, MN 55455

43. Robert G. Babb, Oregon Graduate Institute, CSE Department, 19600 N.W. von Neumann Drive, Beaverton, OR 97006-1999

44. Lawrence J. Baker, Exxon Production Research Company, P.O. Box 2189, Houston, TX 77252-2189

45. Jesse L. Barlow, Department of Computer Science, Pennsylvania State University, University Park, PA 16802

46. Edward H. Barsis, Computer Science and Mathematics, P. O. Box 5800, Sandia National Laboratories, Albuquerque, NM 87185

47. Chris Bischof, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

48. Ake Bjorck, Department of Mathematics, Linkoping University, S-581 83 Linkoping, Sweden

49. Jean R. S. Blair, Department of Computer Science, Ayres Hall, University of Tennessee, Knoxville, TN 37996-1301

50. Roger W. Brockett, Wang Professor of Electrical Engineering and Computer Science, Division of Applied Sciences, Harvard University, Cambridge, MA 02138

51. James C. Browne, Department of Computer Science, University of Texas, Austin, TX 78712

52. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307

53. Donald A. Calahan, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109

54. John Cavallini, Acting Director, Scientific Computing Staff, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20585

55. Ian Cavers, Department of Computer Science, University of British Columbia, Vancouver, British Columbia V6T 1W5, Canada

56. Tony Chan, Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90024

57. Jagdish Chandra, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709

58. Eleanor Chu, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

59. Melvyn Ciment, National Science Foundation, 1800 G Street N.W., Washington, DC 20550

60. Tom Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853

61. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720

62. Andy Conn, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598

63. John M. Conroy, Supercomputer Research Center, 17100 Science Drive, Bowie, MD 20715-4300

64. Jane K. Cullum, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598

65. George Cybenko, Center for Supercomputing Research and Development, University of Illinois, 104 S. Wright Street, Urbana, IL 61801-2932

66. George J. Davis, Department of Mathematics, Georgia State University, Atlanta, GA 30303

67. Tim A. Davis, Computer and Information Sciences Department, 301 CSE, University of Florida, Gainesville, Florida 32611-2024

68. John J. Dorning, Department of Nuclear Engineering Physics, Thornton Hall, McCormick Road, University of Virginia, Charlottesville, VA 22901

69. Iain Duff, Numerical Analysis Group, Central Computing Department, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England

70. Patricia Eberlein, Department of Computer Science, SUNY at Buffalo, Buffalo, NY 14260

71. Stanley Eisenstat, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

72. Lars Elden, Department of Mathematics, Linkoping University, 581 83 Linkoping, Sweden

73. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742

74. Albert M. Erisman, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346

75. Geoffrey C. Fox, Northeast Parallel Architectures Center, 111 College Place, Syracuse Universitiy, Syracuse, NY 13244-4100

76. Paul O. Frederickson, NASA Ames Research Center, RIACS, M/S T045-1, Moffett Field, CA 94035

77. Fred N. Fritsch, L-300, Mathematics and Statistics Division, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550

78. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650

79. K. Gallivan, Computer Science Department, University of Illinois, Urbana, IL 61801

80. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47405

81. Feng Gao, Department of Computer Science, University of British Columbia, Vancouver, British Columbia V6T 1W5, Canada

82. David M. Gay, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

83. C. William Gear, Computer Science Department, University of Illinois, Urbana, IL 61801

84. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Road, Ottawa, Ontario, Canada K1A 0R8

85. J. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

86. John R. Gilbert, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto CA 94304

87. Gene H. Golub, Department of Computer Science, Stanford University, Stanford, CA 94305

88. Joseph F. Grcar, Division 8331, Sandia National Laboratories, Livermore, CA 94550

89. John Gustafson, Ames Laboratory, Iowa State University, Ames, IA 50011

90. Per Christian Hansen, UCI*C Lyngby, Building 305, Technical University of Denmark, DK-2800 Lyngby, Denmark

91. Richard Hanson, IMSL Inc., 2500 Park West Tower One, 2500 City West Blvd., Houston, TX 77042-3020

92. Michael T. Heath, Center for Supercomputing Research and Development, 305 Talbot Laboratory, University of Illinois, 104 South Wright Street, Urbana, IL 61801-2932

93. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P.O. Box 481, Houston, TX 77001

94. Nicholas J. Higham, Department of Mathematics, University of Manchester, Grt Manchester, M13 9PL, England

95. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332

96. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550

97. Ilse Ipsen, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

98. Lennart Johnsson, Thinking Machines Inc., 245 First Street, Cambridge, MA 02142-1214

99. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309

100. Barry Joe, Department of Computer Science, University of Alberta, Edmonton, Alberta T6G 2H1, Canada

101. Bo Kagstrom, Institute of Information Processing, University of Umea, 5-901 87 Umea, Sweden

102. Malvyn H. Kalos, Cornell Theory Center, Engineering and Theory Center Bldg., Cornell University, Ithaca, NY 14853-3901

103. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Bldg. 221, Argonne, IL 60439

104. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

105. Robert J. Kee, Applied Mathematics Division 8331, Sandia National Laboratories, Livermore, CA 94550

106. Kenneth Kennedy, Department of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77001

107. Thomas Kitchens, Department of Energy, Scientific Computing Staff, Office of Energy Research, ER-7, Office G-236 Germantown, Washington, DC 20585

108. Richard Lau, Office of Naval Research, 1030 E. Green Street, Pasadena, CA 91101

109. Alan J. Laub, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106

110. Robert L. Launer, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709

111. Charles Lawson, MS 301-490, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109

112. Peter D. Lax, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012

113. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815

114. John G. Lewis, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346

115. Jing Li, IMSL Inc., 2500 Park West Tower One, 2500 City West Blvd., Houston, TX 77042-3020

116. Heather M. Liddell, Center for Parallel Computing, Department of Computer Science and Statistics, Queen Mary College, University of London, Mile End Road, London E1 4NS, England

117. Arno Liegmann, c/o ETH Rechenzentrum, Clausiusstr. 55, CH-8092 Zurich, Switzerland

118. Joseph Liu, Department of Computer Science, York University, 4700 Keele Street, North York, Ontario, Canada M3J 1P3

119. Robert F. Lucas, Supercomputer Research Center, 17100 Science Drive, Bowie, MD 20715-4300

120. Franklin Luk, Electrical Engineering Department, Cornell University, Ithaca, NY 14853

121. Thomas A. Manteuffel, Department of Mathematics, University of Colorado - Denver, Campus Box 170, P.O. Box 173364, Denver, CO 80217-3364

122. Paul C. Messina, Mail Code 158-79, California Institute of Technology, 1201 E. California Blvd., Pasadena, CA 91125

123. James McGraw, Lawrence Livermore National Laboratory, L-306, P.O. Box 808, Livermore, CA 94550

124. Neville Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green Street, Urbana, IL 61801

125. Cleve Moler, The Mathworks, 325 Linfield Place, Menlo Park, CA 94025

126. Brent Morris, National Security Agency, Ft. George G. Meade, MD 20755

127. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742

128. James M. Ortega, Department of Applied Mathematics, Thornton Hall, University of Virginia, Charlottesville, VA 22901

129. Chris Paige, McGill University, School of Computer Science, McConnell Engineering Building, 3480 University Street, Montreal, Quebec, Canada H3A 2A7

130. Roy P. Pargas, Department of Computer Science, Clemson University, Clemson, SC 29634-1906

131. Beresford N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720

132. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706

133. Robert J. Plemmons, Departments of Mathematics and Computer Science, Box 7311, Wake Forest University Winston-Salem, NC 27109

134. Jesse Poore, Department of Computer Science, Ayres Hall, University of Tennessee, Knoxville, TN 37996-1301

135. Alex Pothen, Department of Computer Science, Pennsylvania State University, University Park, PA 16802

136. Yuanchang Qi, IBM European Petroleum Application Center, P.O. Box 585, N-4040 Hafrsfjord, Norway

137. Giuseppe Radicati, IBM European Center for Scientific and Engineering Computing, via del Giorgione 159, I-00147 Roma, Italy

138. John K. Reid, Numerical Analysis Group, Central Computing Department, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England

139. Werner C. Rheinboldt, Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA 15260

140. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907

141. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore Laboratory, Livermore, CA 94550

142. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706

143. Edward Rothberg, Department of Computer Science, Stanford University, Stanford, CA 94305

144. Axel Ruhe, Dept. of Computer Science, Chalmers University of Technology, S-41296 Goteborg, Sweden

145. Joel Saltz, ICASE, MS 132C, NASA Langley Research Center, Hampton, VA 23665

146. Ahmed H. Sameh, Center for Supercomputing R&D, 1384 W. Springfield Avenue, University of Illinois, Urbana, IL 61801

147. Michael Saunders, Systems Optimization Laboratory, Operations Research Department, Stanford University, Stanford, CA 94305

148. Robert Schreiber, RIACS, Mail Stop 230-5, NASA Ames Research Center, Moffet Field, CA 94035

149. Martin H. Schultz, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

150. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006

151. Lawrence F. Shampine, Mathematics Department, Southern Methodist University, Dallas, TX 75275

152. Andy Sherman, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

153. Kermit Sigmon, Department of Mathematics, University of Florida, Gainesville, FL 32611

154. Horst Simon, Mail Stop T045-1, NASA Ames Research Center, Moffett Field, CA 94035

155. Anthony Skjellum, Lawrence Livermore National Laboratory, 7000 East Ave., L-316, P.O. Box 808 Livermore, CA 94551

156. Danny C. Sorensen, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston, TX 77251

157. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742

158. Paul N. Swartztrauber, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307

159. Philippe Toint, Dept. of Mathematics, University of Namur, FUNOP, 61 rue de Bruxelles, B-Namur, Belgium

160. Bernard Tourancheau, LIP, ENS-Lyon, 69364 Lyon cedex 07, France

161. Hank Van der Vorst, Dept. of Techn. Mathematics and Computer Science, Delft University of Technology, P.O. Box 356, NL-2600 AJ Delft, The Netherlands

162. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853

163. Jim M. Varah, Centre for Integrated Computer Systems Research, University of British Columbia, Office 2053-2324 Main Mall, Vancouver, British Columbia V6T 1W5, Canada

164. Udaya B. Vemulapati, Dept. of Computer Science, University of Central Florida, Orlando, FL 32816-0362

165. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665

166. Phuong Vu, Cray Research, Inc., 655F Lone Oak Drive, Eagan, MN 55121

167. Daniel D. Warner, Department of Mathematical Sciences, O-104 Martin Hall, Clemson University, Clemson, SC 29631

168. Mary F. Wheeler, Rice University, Department of Mathematical Sciences, P.O. Box 1892, Houston, TX 77251

169. Andrew B. White, Computing Division, Los Alamos National Laboratory, P.O. Box 1663, MS-265, Los Alamos, NM 87545

170. Margaret Wright, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

171. David Young, University of Texas, Center for Numerical Analysis, RLM 13.150, Austin, TX 78731

172. Earl Zmijewski, Department of Computer Science, University of California, Santa Barbara, CA 93106

173. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P.O. Box 2001 Oak Ridge, TN 37831-8600

174–183. Office of Scientific & Technical Information, P.O. Box 62, Oak Ridge, TN 37831