

ORNL/TM-11045

OAK RIDGE
NATIONAL
LABORATORY



Parallel Direct Solution of
Sparse Linear Systems

Esmond Ng

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
OPERATING SECTION
1000 EIGHTH AVENUE
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this
report, send us name with request and
the library will arrange a loan.

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

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A03; Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-11045

Engineering Physics and Mathematics Division

Mathematical Sciences Section

PARALLEL DIRECT SOLUTION OF SPARSE LINEAR SYSTEMS

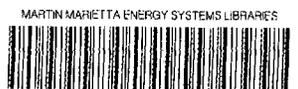
Esmond Ng

Oak Ridge National Laboratory
Mathematical Sciences Section
P.O. Box 2009, Bldg. 9207-A
Oak Ridge, TN 37831-8083

Date Published: January, 1989

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
operated by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC-05-84OR21400



3 4456 0288917 7

Contents

1	Introduction	1
2	Sequential Algorithms for Sparse Linear Systems	1
3	Parallel Sparse Matrix Factorization Algorithms	4
4	Identifying Parallelism and Scheduling Independent Subtasks	6
5	Numerical Experiments and Concluding Remarks	11
6	References	14

PARALLEL DIRECT SOLUTION OF SPARSE LINEAR SYSTEMS

Esmond Ng

Abstract

In this paper the direct solution of sparse linear systems on multiprocessor systems is considered. Elimination trees are used as a tool for identifying and exploiting parallelism in the parallel numerical factorization of the coefficient matrix. Some open problems are described and results of some numerical experiments are provided.

This paper is based on a talk by the author at a Workshop on Methods and Algorithms for PDE's on Advanced Processors held in Austin, Texas on October 17-18, 1988.

1. Introduction

In this paper, we consider direct methods for solving large sparse linear systems

$$Ax = b$$

on multiprocessor systems, where A is an $n \times n$ nonsingular matrix. The basic approach is to decompose A into triangular factors

$$Q_r A Q_c = LU,$$

where Q_r and Q_c are some permutation matrices chosen to preserve sparsity and/or maintain numerical stability, and L and U are respectively lower and upper triangular matrices. With the triangular factorization, the solution to the linear system can be obtained by first solving $Lu = Q_r b$ and $Uv = u$, and then setting $x = Q_c v$.

In general the most expensive part of the solution process is the factorization of A . Thus much effort has been spent in designing efficient factorization algorithms for both sequential and parallel computers. The objective of this paper is to provide an overview of some of the approaches and to discuss some of the issues in the design of effective parallel factorization algorithms. An outline of the paper is as follows. In Section 2, we briefly survey some of the effective sequential algorithms for solving sparse linear systems. Parallel algorithms are then described in Section 3. Tools for identifying and exploiting parallelism are introduced in Section 4, together with a discussion of related issues. Finally, some concluding remarks are provided in Section 5.

2. Sequential Algorithms for Sparse Linear Systems

There has been extensive research in the design of efficient sequential algorithms for the solution of large sparse linear systems. For example, [17] contains an excellent discussion of most of the state-of-the-art methods for solving sparse symmetric positive definite systems and [5] has a detailed description of some methods for handling sparse nonsymmetric problems. The approach we consider in this paper can be summarized as follows. There are basically four steps in the solution process:

1. *Ordering:*

Compute permutations Q_r and Q_c so that L and U are sparse, where $LU = Q_r A Q_c$.

2. *Symbolic factorization:*

Compute the structures of L and U . Set up a compact data structure for storing the nonzeros of L and U .

3. *Numerical factorization:*

Input $Q_r A Q_c$ and compute L and U numerically. (Pivoting may be needed to ensure stability.) Store the nonzeros in the fixed data structure determined at step 2.

4. *Triangular solution:*

Solve $Lu = Q_r b$ and $Uv = u$. Set $x = Q_c v$.

The approach stated above has been widely adopted for solving sparse symmetric positive definite systems [7,8,17], in which case Cholesky factorization is employed and is numerically stable by choosing the diagonal elements as pivots [37]. Moreover, $Q_c = Q_r^T$ and $U = L^T$. Because of the fact that the factorization is stable without pivoting, the structure of L can therefore be determined solely from the structure of $Q_r A Q_r^T$, if we make the assumption that exact cancellation does not occur during numerical factorization. Once the structure of L is known, a compact data structure can then be set up to exploit the sparsity of L . There are efficient symbolic factorization algorithms for computing the structure of L and setting up the data structure [35]. Then the numerical factorization and triangular solution can be performed using the fixed data structure.

The set of nonzeros introduced into L during numerical factorization is referred to as *fill*. The role of the permutation Q_r is to control the amount of fill in L . It is well known that the choice of Q_r can affect the sparsity of L drastically. This is illustrated by an example in Figure 2.1, in which $\hat{Q}_r A \hat{Q}_r^T$ is obtained from $Q_r A Q_r^T$ by reversing the ordering of rows and columns. Unfortunately, the general problem of finding the

$$\begin{aligned}
 Q_r A Q_r^T &= \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & & & \\ \times & & \times & & \\ \times & & & \times & \\ \times & & & & \times \end{pmatrix} & L &= \begin{pmatrix} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \end{pmatrix} \\
 \hat{Q}_r A \hat{Q}_r^T &= \begin{pmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & & \times \\ & & & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} & \hat{L} &= \begin{pmatrix} \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ \times & \times & \times & \times & \times \end{pmatrix}
 \end{aligned}$$

Figure 2.1. An example illustrating the effect of permutations on the sparsity of the Cholesky factors.

permutation that minimizes the number of nonzeros in L is NP-complete [38]. Thus, we have to rely on heuristic strategies for finding permutations that reduce fill in L . Some of the well-known strategies are the nested dissection algorithm [10,16] and the minimum degree algorithm [18]. Efficient implementations of these ordering algorithms and algorithms for the other three steps can be found, for example, in the SPARSPAK package [2].

For sparse nonsymmetric problems, it is well known that pivoting is necessary to ensure stability during numerical factorization [37]. Since the choice of pivot at each step of the numerical factorization depends on both the structure and the numerical values of the active matrix, it is not clear how steps 1 and 2 can be performed prior to step 3. In fact, in almost all implementations of sparse triangular factorization with

pivoting, steps 1, 2 and 3 are often combined together [3,25,27,36]. For example, at each elimination step in MA28 (during which a column is eliminated), the pivot is chosen to preserve sparsity and to maintain stability [3]. Then storage for the nonzeros is allocated immediately before the elimination of the column is performed numerically.

However, if we relax somewhat the condition that only nonzeros are stored, then it is possible to apply the previous four-step approach to nonsymmetric problems. Suppose $Q_r = Q_c = I$ for the moment. Consider computing a triangular factorization of A using Gaussian elimination with partial pivoting (i.e., row interchanges):

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

where P_i corresponds to the row interchange that occurs at step i and L_i is a Gauss transformation at step i . Define $L = \sum_{i=1}^{n-1} L_i - (n-2)I$. In [21], George and Ng have presented a symbolic factorization algorithm that will generate a lower triangular matrix \bar{L} and an upper triangular matrix \bar{U} from the *structure* of A alone so that the structures of \bar{L} and \bar{U} contain respectively those of L and U , *irrespective of the choice of P_i , $1 \leq i \leq n-1$* . Thus, we can use the structures of \bar{L} and \bar{U} as bounds on the structures of L and U respectively in step 2 of the solution process. Using this approach, an effective static data structure for Gaussian elimination with partial pivoting can be set up [20]. Preserving the sparsity of \bar{L} and \bar{U} is important for the effectiveness of this scheme. It was demonstrated in [21] that the sparsity of \bar{L} and \bar{U} depends on the column ordering of A , and furthermore a good symmetric reordering of $A^T A$ appears to be a good column reordering of A .

We conclude the discussion in this section by presenting results of some numerical experiments. The objective is to illustrate the cost of performing each step in the solution process. There are two sets of test problems, all of which are finite element problems defined on L-shaped domains with triangular elements. The problems in the first set are symmetric positive definite and those in the second set are nonsymmetric (with symmetric structures). The experiments were performed on (one processor of) a Sequent Balance 8000 using single-precision floating-point arithmetic and execution times are in seconds. The symmetric positive definite problems were solved using the SPARSPAK package, and the nonsymmetric problems were solved using the approach described in [20]. The results are provided in Tables 2.1 and 2.2.

n	3025	3466	3937	4438	4969
ordering	8.550	9.917	11.533	13.183	15.100
symbolic factorization	1.917	2.150	2.450	2.783	3.117
numerical factorization	59.533	73.317	89.167	105.817	126.567
triangular solution	4.867	5.733	6.550	7.617	8.667

Table 2.1. Execution time statistics (in seconds) for symmetric positive definite problems. (n is the order of the matrix.)

n	3025	3466	3937	4438	4969
ordering	48.800	57.266	69.883	78.483	89.367
symbolic factorization	4.717	5.450	6.117	7.017	7.933
numerical factorization	502.050	646.283	776.300	915.917	1165.133
triangular solution	12.300	14.650	16.950	19.350	22.733

Table 2.2. Execution time statistics (in seconds) for nonsymmetric problems. (n is the order of the matrix.)

3. Parallel Sparse Matrix Factorization Algorithms

It is clear from the numerical results in the previous section that the ordering, symbolic factorization and triangular solution phases are relatively inexpensive; the numerical factorization phase is usually the most expensive part of the solution process. Thus, when multiprocessor systems became available, much effort was spent on parallelizing numerical factorization. In this section, we discuss the potential sources of parallelism in sparse numerical factorization.

We begin with a description of a sequential numerical factorization algorithm, in which columns are eliminated, and we will ignore sparsity for the moment.

```

for  $k = 1$  to  $n$ 
    perform row and/or column interchanges, if necessary
    compute multipliers at step  $k$ 
    for  $j = k + 1$  to  $n$ 
        modify row/column  $j$  by row/column  $k$ 

```

Whether we use a row-oriented algorithm or a column-oriented algorithm will depend on the choice of data structure for the matrix A . For example, if the elements of A are stored by columns, then it may be beneficial to use the column-oriented algorithm to facilitate access of the matrix elements. At any rate, we see from the algorithm above that the computation at each major step can be broken up into subtasks: $cdiv(k)$ and $mod(j, k)$. The subtask $cdiv(k)$ refers to the computation of the multipliers at step k , which includes the row and/or column interchanges that may be necessary to ensure numerical stability, and the subtask $mod(j, k)$ is the modification of row/column j by row/column k . Thus, we can express the computation in a compact way.

```

for  $k = 1$  to  $n$ 
     $cdiv(k)$ 
    for  $j = k + 1$  to  $n$ 
         $mod(j, k)$ 

```

It is important to note that for a given k , each $mod(j, k)$ subtask uses data from row/column k to update row/column j . Hence, the $mod(j, k)$ subtasks are *independent* subtasks for a fixed k . Suppose there are several processors available in a multiprocessor system. As long as $cdiv(k)$ has been performed and row/column k is available to each processor, the independent subtasks $mod(j, k)$'s may therefore be performed concurrently. Note that the operations within a mod subtask are also independent. Such

independence may be exploited, for example, if the processors have vector processing capability or if there are enough processors in the multiprocessor system. However, we will not consider such fine-grain parallel algorithms in this paper; we are interested in medium-grain parallel algorithms. Furthermore, note that the $cdiv(k)$ subtask cannot begin until $mod(k, i)$ has been performed for all $i < k$. Thus, if the matrix is dense, the $cdiv$ subtasks will be executed sequentially even though some of the mod subtasks can be carried out in parallel.

Sparsity of the matrix can enhance the amount of parallelism available. Following is an illustration.

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

For definiteness, suppose we are using a row-oriented factorization algorithm. Consider the first three steps of the factorization, and for simplicity ignore the necessity of pivoting. Because of the nonzero pattern, the first three columns are independent. This implies that $cdiv(i)$, $1 \leq i \leq 3$, can be carried out simultaneously, provided that there are enough processors available. This small example illustrates the fact that because of sparsity in the matrix, not only can some of the mod subtasks be executed in parallel, but some of $cdiv$ subtasks may become independent during factorization, and they can be performed concurrently.

Parallel sparse numerical factorization algorithms have been developed for various classes of multiprocessor systems [4,6,12,14,15,22,23]. Of course, the crucial issue is how the independence among the $cdiv$ and mod subtasks can be identified and how such independent computations can be scheduled in such a way that the computational load is balanced and the amount of synchronization or communication is kept low. We will discuss these issues in the next section.

We have concentrated our discussion in this section on the potential for parallelism in sparse numerical factorization. We conclude the section by making a few remarks about parallel algorithms for the other phases in the solution process. Although the numerical factorization phase is usually the most expensive phase in the solution process on sequential machines, if we are able to reduce the factorization time by employing multiple processors on a multiprocessor system, the cost of doing the remaining three phases (sequentially) may become significant. Thus, research on designing efficient parallel algorithms for the ordering, symbolic factorization and triangular solution phases has been initiated. Some of the work has been reported in [1,12,13,14,24,26,39,40]. However, since the amount of computing in ordering, symbolic factorization or triangular solution is often relatively small, and the sequential algorithms for each of these three phases are extremely efficient, it is in general difficult to devise parallel algorithms with good efficiencies for these three phases. On the other hand, there are situations in which parallel algorithms for ordering, symbolic factorization and triangular solu-

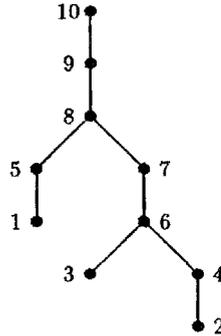


Figure 4.2. The elimination tree associated with the matrix in Figure 4.1.

the subtasks *cdiv*'s and *mod*'s in sparse numerical factorization. It serves as a tool for identifying and exploiting parallelism in sparse matrix factorization. Consider Cholesky factorization and suppose we are using a column-oriented algorithm. We note that in order to perform *cdiv*(*k*), *mod*(*k*, *i*) has to be performed first, for all *i* < *k* such that $L_{ki} \neq 0$. It is easy to show that node *i* must be a descendant of node *k* in the elimination tree; the proof follows from the way the Cholesky factor *L* is computed. A corollary of this result is that column *k* of *L* depends explicitly on column *k* of *A* and a subset of the columns of *L* that are associated with the subtree rooted at node *k*. In other words, *cdiv*(*k*) cannot be executed unless

1. *cdiv*(*i*)'s have been performed, for all nodes *i* in the subtree rooted at node *k*, and
2. *mod*(*k*, *i*) has been applied, for appropriate nodes *i* in the subtree rooted at node *k*.

In general, column *k* of *L* depends either explicitly or implicitly on the columns of *L* that are associated with the subtree rooted at node *k*. Hence, for $k_1 \neq k_2$, if nodes k_1 and k_2 are in two *disjoint* subtrees, then columns k_1 and k_2 are independent, since the two sets of columns on which columns k_1 and k_2 depend are disjoint. For the example in Figures 4.1 and 4.2, columns 1, 2 and 3 are therefore independent. Also, columns 5 and 7 are independent, but column 7 depends either explicitly or implicitly on columns 2, 3, 4 and 6. In summary, dependencies among subtasks in sparse numerical factorization can be identified by analyzing the structure of the elimination tree associated with the matrix *A*.

Since the elimination tree provides information on the dependency among the subtasks in sparse numerical factorization, the tree can be used to schedule the independent computations on a multiprocessor system. One strategy is to schedule the columns by pruning the elimination tree. The idea is to schedule the columns so that the *cdiv*'s can be performed as soon as possible. For example, for the matrix in Figure 4.1, we will first schedule columns 1, 3 and 2. This amounts to removing the leaves from the elimination tree. Then based on the pruned tree, we schedule columns 5 and 6, and

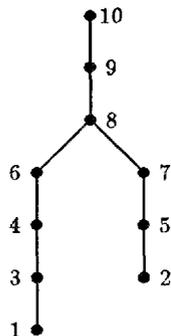


Figure 4.4. The elimination tree associated with the matrix \bar{A} in Figure 4.3.

but also the fill is small. This suggests the following heuristic. We first determine a reordering P_r that attempts to minimize fill in L , where $LL^T = P_r A P_r^T$. Thus, there is an elimination tree associated with $P_r A P_r^T$. Then the tree is restructured in such a way that the new tree has a different shape and hopefully has a smaller height, but the fill and operation count are preserved. This approach is proposed by Liu [30,32]. It amounts to finding another reordering \bar{P}_r for $P_r A P_r^T$ with the constraint that both fill and operation count are preserved. When fill and operation count are preserved, the reordering \bar{P}_r is said to be *equivalent* to P_r .

If A is a finite element matrix arising from a two-dimensional problem, and P_r is a nested dissection reordering, then our experience is that the resulting elimination tree is often short and balanced, and more importantly, P_r often adequately reduces fill. Thus, nested dissection reorderings are often good reorderings for parallel sparse numerical factorization, at least for certain classes of problems. For general sparse symmetric positive definite problems, a minimum degree reordering is in general a much better reordering in terms of fill-reduction. Unfortunately, the resulting elimination tree is often tall and unbalanced, and hence, may not be suitable for parallel factorization. The example in Figure 4.5 is an elimination tree associated with a minimum degree reordering on a 7×7 grid, with a nine-point operator. When we apply Liu's heuristic to the elimination tree in Figure 4.5, we obtain the elimination tree in Figure 4.6, which is again not balanced, although the height of the new tree is somewhat smaller than that of the old tree. In fact, it is our experience that such a phenomenon is typical for minimum degree reorderings.

There is a different way of finding an elimination tree with a short height. Suppose we first find a reordering P_r that attempts to minimize fill in L , where $LL^T = P_r A P_r^T$. There may be equivalent reorderings which preserve fill and operation count. Each equivalent reordering will result in a Cholesky factor with different structure from that of L , and consequently a different elimination tree. Thus, a possibility is to choose from the set of equivalent reorderings the one that has an elimination tree of the shortest height. The scheme for finding the equivalent reordering was first described by Jess and Kees [28], but it was Liu who proved that the resulting elimination tree indeed has the

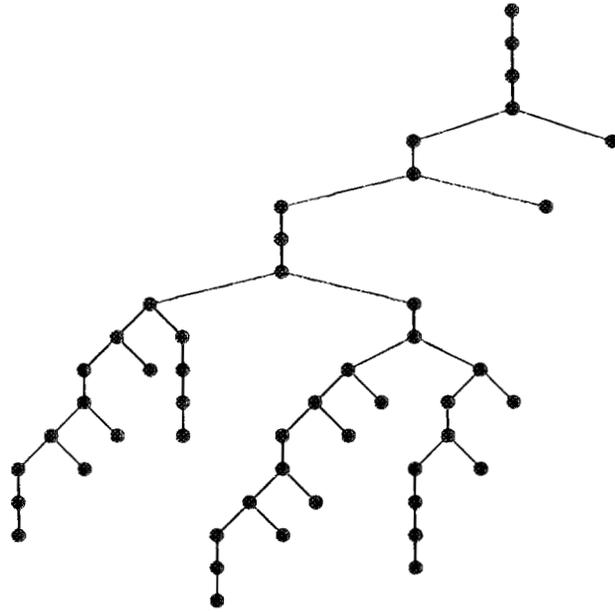


Figure 4.5. The elimination tree associated with a minimum degree reordering on a 7×7 grid.

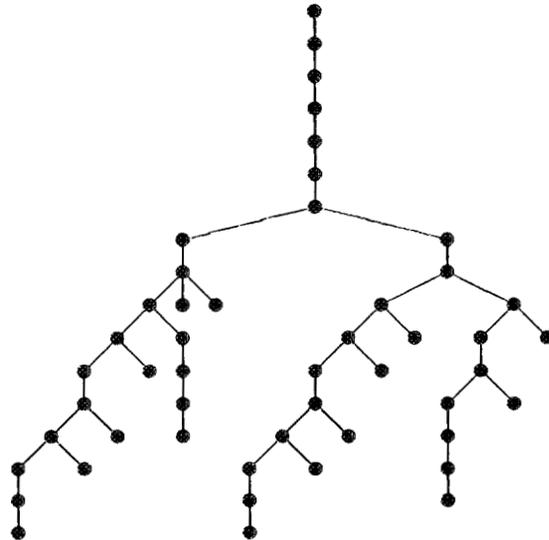


Figure 4.6. The elimination tree obtained by applying Liu's heuristic to the elimination tree in Figure 4.5.

shortest height [30]. Implementations of this scheme were described in [29] and [33]. Unfortunately, we do not have much numerical experience with this approach, since the codes are not available to us. However, even though the new reordering gives an elimination tree that has the smallest height, there is no guarantee that the new tree is balanced or has many branches. Investigation into the effectiveness of this technique in terms of parallel sparse numerical factorization is underway.

The height of an elimination tree is not the only criterion for effective parallel sparse numerical factorization. The shape of the elimination tree is also important. It is our experience that elimination trees which are balanced and have many branches, such as those corresponding to nested dissection orderings, appear to be desirable. However, for most reorderings, their elimination trees are not balanced. Thus, for unbalanced elimination trees, another issue is how to schedule the columns so that the computational work is balanced and the synchronization or communication requirements are reduced. Based on the structure of a *weighted* elimination tree, Geist and Ng have proposed a heuristic for assigning the columns so that part of the computation can be distributed evenly among the processors, and at the same time the amount of synchronization or communication is reduced [9].

Finally, although our discussions in this section are on the solution of sparse symmetric positive definite systems, they apply equally well to nonsymmetric problems. When A is nonsymmetric, the elimination tree is defined in terms of the structure of the Cholesky factor of $A^T A$ [20,22,23].

5. Numerical Experiments and Concluding Remarks

In this paper, we have provided an overview of the current state of affairs in the direct solution of sparse linear systems on multiprocessor systems. Clearly one of the open problems is how to characterize and determine an appropriate reordering for parallel sparse numerical factorization. Note that there are several constraints to be satisfied. It is important to find a reordering so that fill is reduced, and it is desirable to choose the reordering so that the height of the resulting elimination tree is minimized. Moreover, it is also important to have an elimination tree that contains many branches and is balanced, so that there is a high degree of parallelism. Of course, to make the problem even harder, it is desirable to be able to compute the reordering itself in parallel. There are other problems to consider as well, such as the design of efficient parallel algorithms for performing symbolic factorization and triangular solution. These problems are under investigation.

We conclude this paper by providing some numerical results for parallel sparse numerical factorization we have obtained on two different parallel computers. The test problems are the finite element problems we have used in Section 2. Tables 5.1 and 5.2 contain respectively the performance statistics for sparse symmetric positive definite problems and sparse nonsymmetric problems on a Sequent Balance 8000, which is a multiprocessor system with shared-memory. Table 5.3 contains the performance statistics for sparse symmetric positive definite problems on an Intel/iPSC-2, which is distributed-memory parallel machine. For each sparse symmetric positive definite problem, a nested dissection reordering was computed to reduce fill. For each sparse

nonsymmetric problem, a minimum degree reordering with multiple elimination was used [18]. In all tables, n is the order of the matrix, and the second column (“sequential”) contains the execution times in seconds required by the sequential algorithm.

n	sequential	$p = 2$	$p = 4$	$p = 6$
3025	59.533	37.700	19.967	14.333
		1.58	2.98	4.15
		78.96%	74.54%	69.23%
3466	73.317	45.967	24.200	17.183
		1.59	3.03	4.27
		79.75%	75.74%	71.11%
3937	89.167	55.783	29.383	20.700
		1.60	3.03	4.31
		79.92%	75.87%	71.79%
4438	105.817	66.233	34.633	24.467
		1.60	3.06	4.32
		79.88%	76.38%	72.08%
4969	126.567	78.533	41.133	29.033
		1.61	3.08	4.36
		80.58%	76.93	72.66%

Table 5.1. Performance results on a Sequent Balance 8000 for sparse symmetric positive definite systems. For each problem/processor pair, the three entries are respectively the execution time in seconds, the speed-up ratio and the efficiency.

n	sequential	$p = 2$	$p = 4$	$p = 6$
3025	502.050	290.967	151.417	104.800
		1.73	3.32	4.79
		86.27%	82.89%	79.84%
3466	646.283	376.300	194.933	133.833
		1.72	3.32	4.83
		85.87%	82.89%	80.48%
3937	776.300	449.050	231.700	160.983
		1.73	3.35	4.82
		86.44%	83.76%	80.37%
4438	915.917	529.800	273.333	186.867
		1.73	3.35	4.90
		86.44%	83.77%	81.69%
4969	1165.133	666.150	343.517	235.750
		1.75	3.39	4.94
		87.45%	84.79%	82.37%

Table 5.2. Performance results on a Sequent Balance 8000 for sparse non-symmetric systems. For each problem/processor pair, the three entries are respectively the execution time in seconds, the speed-up ratio and the efficiency.

Like most parallel algorithms, we see from the tables that, for a fixed number of processors, the efficiency increases as the size of the problem increases. However, for

n	sequential	$p = 8$	$p = 16$	$p = 32$
2233	17.803	6.500	4.695	3.513
		2.74	3.79	5.07
		34.24%	23.70%	15.84%
2614	22.512	8.480	6.198	4.400
		2.65	3.63	5.12
		33.18%	22.70%	15.99%
3025	28.183	10.532	7.456	5.292
		2.68	3.78	5.33
		33.45%	23.62%	16.64%
3466	34.607	13.141	8.980	6.313
		2.63	3.85	5.48
		32.92%	24.09%	17.13%
3937	42.368	15.175	10.670	7.565
		2.79	3.97	5.60
		34.90%	24.82%	17.50%
4438	50.379	18.224	12.551	8.437
		2.76	4.01	5.97
		34.56%	25.09%	18.66%
4969	59.932	21.868	14.498	9.872
		2.74	4.13	6.07
		34.26%	25.84%	18.97%

Table 5.3. Performance results on an Intel/iPSC-2 for sparse symmetric positive definite systems. For each problem/processor pair, the three entries are respectively the execution time in seconds, the speed-up ratio and the efficiency.

a fixed problem, the efficiency decreases as the number of processors increases. The efficiencies on the Intel/iPSC-2 are poor; this is mainly because the communication overhead is relatively high compared to the speed of computation.

6. References

1. G. Alaghband and H. F. Jordan. *Multiprocessor sparse L/U decomposition with controlled fill-in*. Technical Report 85-48, ICASE, NASA Langley Research Center, Hampton, Virginia, 1985.
2. E. C. H. Chu, J. 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, Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, 1984.
3. I. S. Duff. *MA28 - A set of FORTRAN subroutines for sparse unsymmetric linear equations*. Technical Report AERE R-8730, Harwell, 1977.
4. I. S. Duff. Parallel implementation of multifrontal schemes. *Parallel Computing*, 3:193-204, 1986.
5. I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, Oxford, England, 1987.
6. I. S. Duff, N. I. M. Gould, M. Lescrenier, and J. K. Reid. *The multifrontal method in a parallel environment*. Technical Report CSS 211, Computer Science and Systems Division, Harwell, 1987.
7. I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Trans. on Math. Software*, 9:302-325, 1983.
8. 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.
9. G. A. Geist and E. G-Y. Ng. *A partitioning strategy for parallel sparse Cholesky factorization*. Technical Report ORNL/TM-10937, Oak Ridge National Laboratory, Oak Ridge, TN, 1988.
10. J. A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345-363, 1973.
11. J. A. George, M. T. Heath, and J. W-H. Liu. Parallel Cholesky factorization on a shared-memory multiprocessor. *Linear Algebra and its Appl.*, 77:165-187, 1986.
12. J. A. George, M. T. Heath, J. W-H. Liu, and E. G-Y. Ng. Solution of sparse positive definite systems on a shared memory multiprocessor. *Internat. J. Parallel Programming*, 15:309-325, 1986.

13. J. 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.
14. J. A. George, M. T. Heath, J. W-H. Liu, and E. G-Y. Ng. *Solution of sparse positive definite systems on a hypercube*. Technical Report ORNL/TM-10865, Oak Ridge National Laboratory, Oak Ridge, TN, 1988.
15. J. A. George, M. T. Heath, J. W-H. Liu, and E. G-Y. Ng. Sparse Cholesky factorization on a local-memory multiprocessor. *SIAM J. Sci. Stat. Comput.*, 9:327-340, 1988.
16. J. 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.
17. J. A. George and J. W-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.
18. J. A. George and J. W-H. Liu. On the evolution of the minimum degree algorithm. 1989. (To appear in SIAM Review.)
19. J. A. George, J. W-H. Liu, and E. G-Y. Ng. Communication reduction in parallel sparse Cholesky factorization on a hypercube. In M. T. Heath, editor, *Hypercube Multiprocessors*, pages 576-586, SIAM Publications, Philadelphia, PA, 1987.
20. J. A. George, J. W-H. Liu, and E. G-Y. Ng. A data structure for sparse QR and LU factors. *SIAM J. Sci. Stat. Comput.*, 9:100-121, 1988.
21. J. A. George and E. G-Y. Ng. Symbolic factorization for sparse Gaussian elimination with partial pivoting. *SIAM J. Sci. Stat. Comput.*, 8:877-898, 1987.
22. J. A. George and E. G-Y. Ng. *Parallel sparse Gaussian elimination with partial pivoting*. Technical Report ORNL/TM-10866, Oak Ridge National Laboratory, Oak Ridge, TN, 1988.
23. J. R. Gilbert. *An efficient parallel sparse partial pivoting algorithm*. Technical Report CMI No. 88/45052-1, Centre for Computer Science, Dept. of Science and Technology, Chr. Michelsen Institute, Bergen, Norway, 1988.
24. J. R. Gilbert and H. Hafsteinsson. *A parallel algorithm for finding fill in a sparse symmetric matrix*. Technical Report TR 86-789, Dept. of Computer Science, Cornell University, Ithaca, New York, 1986.
25. J. R. Gilbert and T. Peierls. Sparse partial pivoting in time proportional to arithmetic operations. *SIAM J. Sci. Stat. Comput.*, 9:862-874, 1988.
26. J. R. Gilbert and E. Zmijewski. *A parallel graph partitioning algorithm for a message-passing multiprocessor*. Technical Report TR 87-803, Dept. of Computer Science, Cornell University, Ithaca, New York, 1987.

27. P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Maintaining LU factors of a general sparse matrix. *Linear Algebra and its Appl.*, 88/89:239–270, 1987.
28. J. A. G. Jess and H. G. M. Kees. A data structure for parallel L/U decomposition. *IEEE Trans. Comput.*, C-31:231–239, 1982.
29. J. G. Lewis, B.W. Peyton, and A. Pothen. *A fast algorithm for reordering sparse matrices for parallel factorization*. Technical Report, Oak Ridge National Laboratory, Oak Ridge, TN, 1988. (Submitted to *SIAM J. Stat. Sci. Comput.*)
30. J. W-H. Liu. *Reordering sparse matrices for parallel elimination*. Technical Report CS-87-01, Dept. of Computer Science, York University, Downsview, Ontario, 1987. (To appear in *Parallel Computing*.)
31. J. W-H. Liu. *The role of elimination trees in sparse factorization*. Technical Report CS-87-12, Dept. of Computer Science, York University, Downsview, Ontario, 1987. (To appear in *SIAM J. Matrix Anal. & Appl.*)
32. J. W-H. Liu. Equivalent sparse matrix reordering by elimination tree rotations. *SIAM J. Sci. Stat. Comput.*, 9:424–444, 1988.
33. J. W-H. Liu and A. Mirzaian. *A linear reordering algorithm for parallel pivoting of chordal graphs*. Technical Report CS-87-02, Dept. of Computer Science, York University, Downsview, Ontario, 1987. (To appear in *SIAM J. Disc. Math.*)
34. A. Pothen. *The complexity of optimal elimination trees*. Technical Report CS-88-16, Dept. of Computer Science, The Pennsylvania State University, University Park, PA, 1988.
35. A. H. Sherman. *On the efficient solution of sparse systems of linear and nonlinear equations*. Technical Report 46, Dept. of Computer Science, Yale University, New Haven, Connecticut, 1975.
36. A. H. Sherman. Algorithm 533. NSPIV, a FORTRAN subroutine for sparse Gaussian elimination with partial pivoting. *ACM Trans. on Math. Software*, 4:391–398, 1978.
37. J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, 1965.
38. M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981.
39. E. Zmijewski. *Sparse Cholesky Factorization on a Multiprocessor*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York 14853-7501, August 1987.
40. 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 | 26-30. R. C. Ward |
| 2. J. B. Drake | 31. P. H. Worley |
| 3. G. A. Geist | 32. A. Zucker |
| 4-5. R. F. Harbison | 33. J. J. Dorning (Consultant) |
| 6. M. T. Heath | 34. R. M. Haralick (Consultant) |
| 7-11. J. K. Ingersoll | 35. Central Research Library |
| 12. M. R. Leuze | 36. ORNL Patent Office |
| 13-17. F. C. Maienschein | 37. K-25 Plant Library |
| 18-22. E. G. Ng | 38. Y-12 Technical Library |
| 23. G. Ostrouchov | /Document Reference Station |
| 24. B. W. Peyton | 39. Laboratory Records - RC |
| 25. C. H. Romine | 40-41. Laboratory Records Department |

EXTERNAL DISTRIBUTION

42. Dr. Donald M. Austin, Office of Scientific Computing, Office of Energy Research, ER-7, Germantown Building, U.S. Department of Energy, Washington, DC 20545
43. Dr. Robert G. Babb, Department of Computer Science and Engineering, Oregon Graduate Center, 19600 N.W. Walker Road, Beaverton, OR 97006
44. Lawrence J. Baker, Exxon Production Research Company, P.O. Box 2189, Houston, TX 77252-2189
45. Dr. Jesse L. Barlow, Department of Computer Science, Pennsylvania State University, University Park, PA 16802
46. Dr. Chris Bischof, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
47. Prof. Ake Bjorck, Department of Mathematics, Linkoping University, Linkoping 58183, Sweden
48. Dr. James C. Browne, Department of Computer Sciences, University of Texas, Austin, TX 78712
49. Dr. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307

50. Dr. Donald A. Calahan, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109
51. Dr. Tony Chan, Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90024
52. Dr. Jagdish Chandra, Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709
53. Dr. Eleanor Chu, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
54. Prof. Tom Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853
55. Dr. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720
56. Prof. Andy Conn, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
57. Dr. Jane K. Cullum, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598
58. Dr. George Cybenko, Computer Science Department, University of Illinois, Urbana, IL 61801
59. Dr. George J. Davis, Department of Mathematics, Georgia State University, Atlanta, GA 30303
60. Dr. Jack J. Dongarra, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
61. Dr. Iain Duff, CSS Division, Harwell Laboratory, Didcot, Oxon OX11 0RA, England
62. Prof. Pat Eberlein, Department of Computer Science, SUNY/Buffalo, Buffalo, NY 14260
63. Dr. Stanley Eisenstat, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
64. Dr. Lars Elden, Department of Mathematics, Linköping University, 581 83 Linköping, Sweden
65. Dr. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742
66. Dr. Albert M. Erisman, Boeing Computer Services, 565 Andover Park West, Tukwila, WA 98188

67. Dr. Peter Fenyes, General Motors Research Laboratory, Department 15, GM Technical Center, Warren, MI 48090
68. Prof. David Fisher, Department of Mathematics, Harvey Mudd College, Claremont, CA 91711
69. Dr. Geoffrey C. Fox, Booth Computing Center 158-79, California Institute of Technology, Pasadena, CA 91125
70. Dr. Paul O. Frederickson, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
71. Dr. Fred N. Fritsch, L-300, Mathematics and Statistics Division, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550
72. Dr. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650
73. Dr. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47405
74. Dr. David M. Gay, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
75. Dr. C. William Gear, Computer Science Department, University of Illinois, Urbana, Illinois 61801
76. Dr. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Road, Ottawa, Ontario, Canada K1A 0R8
77. Dr. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
78. Dr. John Gilbert, Xerox Palo Alto Research Center 3333 Coyote Hill Road Palo Alto, CA 94304
79. Dr. Jacob D. Goldstein, The Analytic Sciences Corporation, 55 Walkers Brook Drive, Reading, MA 01867
80. Prof. Gene H. Golub, Department of Computer Science, Stanford University, Stanford, CA 94305
81. Dr. Joseph F. Grcar, Division 8331, Sandia National Laboratories, Livermore, CA 94550
82. Dr. Per Christian Hansen, Technical University of Denmark, Danish University Computing Center, UCI-C Lyngby, Building 305, DK-2800 Lyngby, Denmark
83. Dr. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P.O. Box 481, Houston, TX 77001

84. Dr. F. J. Helton, GA Technologies, P.O. Box 81608, San Diego, CA 92188
85. Dr. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332
86. Dr. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550
87. Dr. Ilse Ipsen, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
88. Ms. Elizabeth Jessup, Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520
89. Prof. Barry Joe, Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1
90. Dr. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309
91. Dr. Bo Kagstrom, Institute of Information Processing, University of Umea, 5-901 87 Umea, Sweden
92. Dr. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
93. Dr. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
94. Dr. Robert J. Kee, Applied Mathematics Division 8331, Sandia National Laboratories, Livermore, CA 94550
95. Ms. Virginia Klema, Statistics Center, E40-131, MIT, Cambridge, MA 02139
96. Dr. Richard Lau, Office of Naval Research, 1030 E. Green Street, Pasadena, CA 91101
97. Dr. Alan J. Laub, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106
98. Dr. Robert L. Launer, Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709
99. Dr. Charles Lawson, MS 301-490, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109
100. Prof. Peter D. Lax, Director, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012
101. Dr. John G. Lewis, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346

102. Dr. Heather M. Liddell, Director, Center for Parallel Computing, Department of Computer Science and Statistics, Queen Mary College, University of London, Mile End Road, London E1 4NS, England
103. Dr. Joseph Liu, Department of Computer Science, York University, 4700 Keele Street, Downsview, Ontario, Canada M3J 1P3
104. Dr. Franklin Luk, Electrical Engineering Department, Cornell University, Ithaca, NY 14853
105. James G. Malone, General Motors Research Laboratories, Warren, Michigan 48090-9055
106. Dr. Thomas A. Manteuffel, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
107. Dr. Bernard McDonald, National Science Foundation, 1800 G Street, NW, Washington, DC 20550
108. Dr. Paul C. Messina, California Institute of Technology, Mail Code 158-79, Pasadena, CA 91125
109. Dr. Cleve Moler, Ardent Computers, 550 Del Ray Avenue, Sunnyvale, CA 94086
110. Dr. Brent Morris, National Security Agency, Ft. George G. Meade, MD 20755
111. Dr. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742
112. Maj. C. E. Oliver, Office of the Chief Scientist, Air Force Weapons Laboratory, Kirtland Air Force Base, Albuquerque, NM 87115
113. Dr. James M. Ortega, Department of Applied Mathematics, University of Virginia, Charlottesville, VA 22903
114. Prof. Chris Paige, Department of Computer Science, McGill University, 805 Sherbrooke Street W., Montreal, Quebec, Canada H3A 2K6
115. Dr. John F. Palmer, NCUBE Corporation, 915 E. LaVieve Lane, Tempe, AZ 85284
116. Prof. Roy P. Pargas, Department of Computer Science, Clemson University, Clemson, SC 29634-1906
117. Prof. Beresford N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720
118. Prof. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706
119. Dr. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650

120. Dr. Alex Pothan, Department of Computer Science, Pennsylvania State University, University Park, PA 16802
121. Dr. John K. Reid, CSS Division, Building 8.9, AERE Harwell, Didcot, Oxon, England OX11 0RA
122. Dr. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907
123. Dr. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore Laboratory, Livermore, CA 94550
124. Dr. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706
125. Dr. Ahmed H. Sameh, Computer Science Department, University of Illinois, Urbana, IL 61801
126. Dr. Michael Saunders, Systems Optimization Laboratory, Operations Research Department, Stanford University, Stanford, CA 94305
127. Dr. Robert Schreiber, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180
128. Dr. Martin H. Schultz, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
129. Dr. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006
130. Dr. Lawrence F. Shampine, Mathematics Department, Southern Methodist University, Dallas, TX 75275
131. Dr. Kermit Sigmon, Department of Mathematics, University of Florida, Gainesville, FL 32611
132. Dr. Horst Simon, Mail Stop 258-5, NASA Ames Research Center, Moffett Field, CA 94035
133. Dr. Danny C. Sorensen, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
134. Prof. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742
135. Dr. Kosmo D. Tatalias, Atlantic Aerospace Electronics Corporation, 6404 Ivy Lane, Suite 300, Breenbelt, MD 20770-1406
136. Prof. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853

137. Dr. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665
138. Dr. Andrew B. White, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
139. Dr. Arthur Wouk, Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709
140. Dr. Margaret Wright, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
141. Dr. A. Yeremin, Department of Numerical Mathematics of the USSR Academy of Sciences, Gorki Street 11, Moscow, 103905, USSR
142. 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
- 143-152. Office of Scientific & Technical Information, P.O. Box 62, Oak Ridge, TN 37831