UMENTATION PAGE

Form Approved
OMB No. 0704-0188

(2)

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Reprint

4. TITLE AND SUBTITLE

Title shown on Reprint

5. FUNDING NUMBERS

DAAL03-91-G-0032

6. AUTHOR(S)

Author(s) listed on Reprint

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Yale Univ.
New Haven, CT 06520

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARO 28007.2-MA

11. SUPPLEMENTARY NOTES

The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

ABSTRACT ON REPRINT

14. SUBJECT TERMS	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

SIAM J. SCI. COMPUT.
Vol. 14, No. 1, pp. 253–257, January 1993

EXPLOITING STRUCTURAL SYMMETRY IN A SPARSE PARTIAL PIVOTING CODE*

STANLEY C. EISENSTAT† AND JOSEPH W. H. LIU‡

Abstract. This short communication shows how to take advantage of structural symmetry to improve the performance of a class of partial pivoting codes for the LU factorization of large sparse unsymmetric matrices. Experimental results demonstrate the effectiveness of this technique in reducing the overall factorization time.

Key words. sparse LU factorization, partial pivoting, structural symmetry

AMS(MOS) subject classifications. 65F05, 65F50

1. Introduction. Many implementations of sparse LU factorization with partial pivoting compute the factors one row or column at a time. Each step involves both symbolic operations (to determine the nonzero structure) and numeric operations. With the development of fast floating-point hardware and vector processors, the symbolic operations have come to represent a nontrivial fraction of the overall factorization time. Thus any sizable reduction in this symbolic overhead would have a significant impact.

The technique of *symmetric reduction* [4] exploits structural symmetry to decrease the amount of structural information required for the symbolic factorization of a sparse unsymmetric matrix (i.e., for obtaining the nonzero structures of the factor matrices). This has the practical advantage of decreasing the run-time.

In this short communication, we show how to use symmetric reduction to improve the performance of a class of partial pivoting codes for the LU factorization of large sparse unsymmetric matrices, in particular, Sherman's NSPFAC (a more recent version of NSPIV [8]) and a code of Gilbert and Peierls [7]. For some problems the speedup is more than a factor of two.

Notation. For an $n \times n$ matrix M and two sets I and J of subscripts, we let M_{IJ} denote the submatrix of M determined by the rows in I and the columns in J. As a special case, we let M_{I*} denote the submatrix of M determined by the rows in I.

We let $G(M)$ denote the associated directed graph. Here edges are directed from row to column; i.e., (r, c) is an edge in $G(M)$ if and only if m_{rc} is nonzero. We use the notation $r \xrightarrow{M} c$ to indicate the existence of an edge from r to c in $G(M)$, and $r \overset{M}{\Longrightarrow} c$ to indicate the existence of a path from r to c. We also adopt the conventic that $i \overset{M}{\Longrightarrow} i$ for any i.

2. Unsymmetric symbolic factorization. Let A be a sparse unsymmetr $n \times n$ matrix that can be decomposed (without pivoting) into $L \cdot U$, where L is lowe triangular with unit diagonal and U is upper triangular. Let F denote the *filled matri* $L + U$.

*Received by the editors February 15, 1992; accepted for publication June 23, 1992.

†Department of Computer Science and Research Center for Scientific Computation, Yale Uni versity, New Haven, Connecticut 06520. The research of this author was supported in part by U. S Army Research Office contract DAAL03-91-G-0032.

‡Department of Computer Science, York University, North York, Ontario, Canada M3J 1P3 The research of this author was supported in part by Natural Sciences and Engineering Researcl Council of Canada grant A5509, and by the Institute for Mathematics and its Applications, witl funds provided by the National Science Foundation.

Assume that we have determined the nonzero structures of the first $k-1$ rows of L and U; i.e., letting $K = \{1, \ldots, k-1\}$ and $\overline{K} = \{k, \ldots, n\}$, we know the structure of

$$F_{K*} = L_{K*} + U_{K*} = \begin{pmatrix} L_{KK} & 0 \end{pmatrix} + \begin{pmatrix} U_{KK} & U_{K\overline{K}} \end{pmatrix}.$$

The following result relates the structures of the rows L_{k*} and U_{k*} to the existence of certain paths in $G(U_{K*})$.

THEOREM 2.1 (see [7]). *$k \xrightarrow{F} i$ if and only if $k \xrightarrow{A} m \xrightarrow{U_{K*}} i$ for some m.*

Thus, to determine the nonzero structure of $F_{k*} = L_{k*} + U_{k*}$, we can search $G(U_{K*})$ for nodes reachable from some node m for which $a_{km} \neq 0$.

3. Two sparse partial pivoting codes. We focus on two implementations of sparse LU factorization with partial pivoting: Sherman's NSPFAC (a descendant of NSPIV [8]) and Gilbert and Peierls's code [7] (referred to here as GP).

NSPFAC factors A by rows using column partial pivoting. While computing F_{k*}, it represents the structure of the current, partially formed row by an ordered, linked list of subscripts corresponding to nonzero columns. The linked list is initialized to the nonzero columns in A_{k*}. For each nonzero f_{kj} (in increasing column order), the structural and numeric updates from U_{j*} to F_{k*} are applied in a single loop, one element at a time. The numeric update involves two levels of indirection.

Gilbert and Peierls [7] observed that it is not necessary to apply the row updates in increasing order—*any* order consistent with a *topological order* of $G(U_{KK})$ would suffice. They also noted that a depth-first search of $G(U_{K*})$ starting from the nonzero columns of A_{k*} gives the nonzero structure of F_{k*}, and that a topological ordering can be obtained as a byproduct, without additional work. Using this result, they show that GP runs in time proportional to the number of floating-point operations, a property not shared by other sparse partial pivoting codes.

In computing F_{k*},[1] GP first does a depth-first search to compute the structure of L_{k*} (but not U_{k*}) as above. Then, for each nonzero f_{kj} (in topological order), it applies the structural updates from U_{j*} to U_{k*} and the numeric updates from U_{j*} to F_{k*} in a single loop, one element at a time.

To estimate the time NSPFAC and GP spend in nonnumeric computations, we wrote a sparse LU factorization code (called NF) that uses a predetermined pivot sequence and precomputed factor structures.[2] By using the same pivot sequence and factor structures as computed by NSPFAC or GP, we can measure how much time would be spent if the nonnumeric operations involving symbolic factorization and pivot selection were removed.

Table 2 gives the run-times[3] for ten problems from the Harwell-Boeing collection [3]. For each test matrix A, the rows of the matrix were preordered by a minimum degree ordering of AA^t, as suggested by George and Ng [5]. The results for the Sun SparcStation/1 show that the nonnumeric overhead can exceed 50 percent. For the

[1] Although GP computes the LU factorization by columns using row partial pivoting, to be consistent we describe the Gilbert–Peierls approach by rows. In the numerical experiments, GP factored A^t rather than A.

[2] NSPFAC scales rows by multiplying by the reciprocal of the pivot; GP scales columns by dividing by the pivot. To make the comparisons fair, we used two versions of NF.

[3] All programs were written in Fortran; use double-precision arithmetic; and were compiled with optimization enabled (`f77 -O` (SC1.0 Fortran V1.4) on the SparcStation/1, `xlf -O` (XL FORTRAN Compiler/6000 Version 2.2) on the RS/6000).

TABLE 1
Nonzeros in original and filled matrices.

Problem	n	$nz(A)$	$nz(F_{NSP})$	$nz(F_{GP})$
GEMAT11	4929	33185	79774	79757
JPWH991	991	6027	134741	131502
LNS3937	3937	25407	403017	403520
LNSP3937	3937	25407	383313	383340
MCFE	765	24382	68288	68288
ORANI678	2529	90158	262250	262365
ORSREG1	2205	14133	374957	374957
SAYLR4	3564	22316	624742	624742
SHERMAN3	5005	20033	409475	409475
SHERMAN5	3312	20793	242556	242556

TABLE 2
Time (in seconds) for NSPFAC/GP *and* NF *with the same pivot sequence.*

	SparcStation/1				RS/6000			
Problem	NSP	NF	GP	NF	NSP	NF	GP	NF
GEMAT11	2.61	1.44	3.23	1.59	1.75	0.48	1.98	0.50
JPWH991	29.79	17.75	32.54	18.75	19.53	5.20	18.97	5.10
LNS3937	68.58	39.76	73.79	42.86	43.27	11.88	45.02	12.38
LNSP3937	63.34	35.16	65.42	37.92	38.38	10.58	39.77	10.98
MCFE	7.18	4.16	7.89	4.63	4.83	1.28	4.65	1.32
ORANI678	32.17	15.64	29.41	16.81	21.82	4.87	17.78	5.13
ORSREG1	92.43	56.26	103.94	60.40	60.23	16.33	62.20	17.33
SAYLR4	168.39	102.90	189.65	110.18	110.07	29.80	118.82	30.78
SHERMAN3	97.31	58.60	107.58	62.70	62.88	17.10	65.18	17.32
SHERMAN5	42.50	26.01	47.98	27.88	27.90	7.73	29.25	7.92

IBM RS/6000 Model 320, which has relatively faster (with respect to the speed of its integer unit) floating-point hardware, the nonnumeric overhead can exceed 70 percent.

4. Symmetric reduction. Theorem 2.1 characterizes the nonzero structure of F_{k*} in terms of the structure of A_{k*} and paths in the graph $G(U_{K*})$. But by removing from $G(U_{K*})$ edges that are not needed to preserve the set of paths, a process called *transitive reduction* [1], we can decrease the amount of searching required to determine the structure.

If we remove all such redundant edges, then we get the *elimination dag* (*directed acyclic graph*) [6], the minimal subgraph that preserves paths. However, if we remove fewer redundant edges, we will still preserve the set of paths. The search time will be larger than for the elimination dag, but the total time (including the time for the reduction) may be less.

Symmetric reduction [4] is based on structural symmetry in the filled matrix F. The symmetric reduction of $G(U_{K*})$ is obtained by deleting all edges (i, m) for which $\ell_{ji} * u_{ij} \neq 0$ for some $j < \min\{k, m\}$. In effect, all nonzeros to the right of the first symmetric nonzero are deleted; if no such symmetric nonzero exists, then all nonzero entries are kept. We denote the resulting symmetrically reduced matrix by $\widetilde{U}_{K*}$.

Figure 1 shows the structures of two partial factor matrices $F_{K_4,*}$ and $F_{K_5,*}$, where $K_4 = \{1,2,3,4\}$ and $K_5 = \{1,2,3,4,5\}$. We use "•" to indicate a nonzero entry in the original matrix, and "o" an entry that fills in. Since $\ell_{41} * u_{14}$ is the only symmetric nonzero pair in $F_{K_4,*}$, only the nonzeros to the right of u_{14} are pruned from $U_{K_4,*}$ to get $\widetilde{U}_{K_4*}$. On the other hand, there are two more symmetric nonzero pairs in $F_{K_5,*}$, $\ell_{52} * u_{25}$ and $\ell_{54} * u_{45}$, so that nonzeros are pruned in rows 2 and 4 to get $\widetilde{U}_{K_5,*}$.

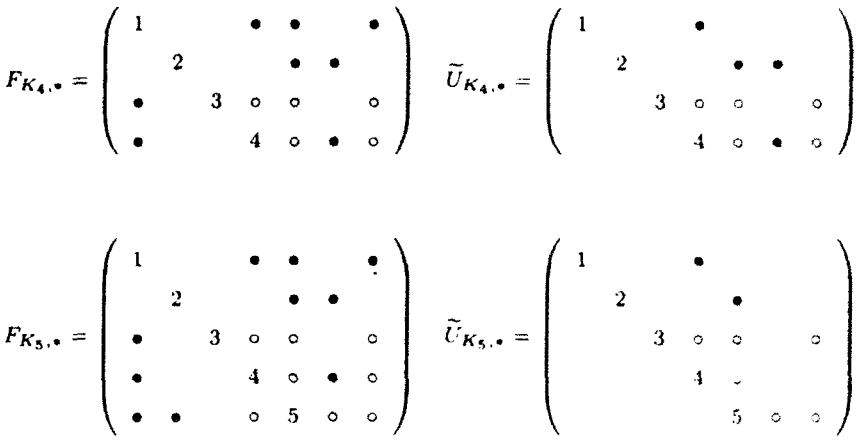

FIG. 1. *An example to illustrate symmetric reduction.*

TABLE 3
Normalized time for the original and two modified versions of NSPFAC/GP.

	SparcStation/1						IBM RS/6000					
Problem	NSP	Red	Mod	GP	Red	Mod	NSP	Red	Mod	GP	Red	Mod
GEMAT11	1.81	1.76	1.56	2.03	1.69	1.64	3.65	2.50	2.29	3.96	2.64	2.50
JPWH991	1.68	1.27	1.09	1.74	1.09	1.09	3.76	1.58	1.19	3.72	1.24	1.21
LNS3937	1.72	1.30	1.12	1.72	1.12	1.12	3.64	1.67	1.27	3.64	1.36	1.33
LNSP3937	1.80	1.32	1.12	1.73	1.14	1.13	3.63	1.68	1.29	3.62	1.36	1.33
MCFE	1.73	1.39	1.21	1.70	1.19	1.17	3.77	1.86	1.43	3.52	1.42	1.38
ORANI678	2.06	1.79	1.60	1.75	1.25	1.21	4.48	2.86	2.51	3.47	1.62	1.49
ORSREG1	1.64	1.27	1.07	1.72	1.08	1.08	3.69	1.58	1.17	3.59	1.20	1.18
SAYLR4	1.64	1.26	1.07	1.72	1.07	1.07	3.69	1.59	1.20	3.86	1.26	1.24
SHERMAN3	1.66	1.30	1.08	1.72	1.09	1.08	3.68	1.59	1.18	3.76	1.27	1.25
SHERMAN5	1.63	1.29	1.09	1.72	1.11	1.11	3.61	1.63	1.23	3.69	1.33	1.28
Harmonic Mean	1.73	1.37	1.17	1.75	1.16	1.15	3.75	1.78	1.37	3.68	1.40	1.36

Symmetric reduction preserves the set of paths in $G(U)$ (see [4]). The argument can be adapted to show that it also preserves the set of paths in $G(U_{K*})$. The following result is an immediate corollary of this observation and Theorem 2.1.

COROLLARY 4.1. *$k \xrightarrow{F} i$ if and only if $k \xrightarrow{A} m \xrightarrow{\widetilde{U}_{K*}} i$ for some m.*

5. Numerical experiments. We incorporated symmetric reduction into NSPFAC and GP. In the process, we made a number of small modifications to the codes.

In NSPFAC, we split the innermost loop so that, when applying the update from U_{j*} to F_{k*}, we complete the structural update *before* performing the numeric update. Furthermore, we removed one of the two levels of indirection from the numeric update.

In GP, we removed the structural update to U_{k*} from the innermost loop and disabled the test for accidental cancellation, for otherwise symmetric reduction might not preserve paths. Furthermore, we combined the symbolic computation of L_{k*} and U_{k*} into a single depth-first search that computes the structure of F_{k*} using Corollary 4.1.

Table 3 presents the ratios of the run-times of the original and two modified versions of NSPFAC and GP to the corresponding NF using the same pivot sequence. The versions labeled "Red" include only those changes needed to incorporate symmetric reduction; the versions labeled "Mod" also include the changes that remove one

level of indirection (NSPFAC) or combine the depth-first searches (GP). As in Table 2, the rows of each test matrix A were preordered by a minimum degree ordering on AA^t.

The results show a dramatic decrease in the overall factorization time. The reduction is more pronounced on the RS/6000 due to the relatively faster floating-point hardware. An even more dramatic reduction would be expected on a vector processor.

There are other ways to improve these sparse partial pivoting codes. One is to use path-symmetric or partial path-symmetric reduction, as described in [4]. Another is to switch from nodal to supernodal elimination [2], which we expect will give a substantial improvement. A code with these features is currently under development by the authors.

Acknowledgment. The authors thank John Gilbert for making available a prerelease of sparse Matlab, which was used to generate the row orderings for the test problems, and for suggesting the notation used for edges and paths.

REFERENCES

[1] A. V. AHO, M. R. GAREY, AND J. D. ULLMAN, *The transitive reduction of a directed graph*, SIAM J. Comput., 1 (1972), pp. 131–137.

[2] 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. Supercomputer Appl., 1 (1987), pp. 10–30.

[3] I. S. DUFF, R. GRIMES, AND J. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[4] S. C. EISENSTAT AND J. W. H LIU, *Exploiting structural symmetry in sparse unsymmetric symbolic factorization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 202–211.

[5] J. A. GEORGE AND E. NG, *An implementation of Gaussian elimination with partial pivoting for sparse systems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 390–409.

[6] J. R. GILBERT AND J. W. H. LIU, *Elimination structures for unsymmetric sparse LU factors*, Tech. Report CS-90-11, Department of Computer Science, York University, North York, Ontario, Canada, 1990.

[7] J. R. GILBERT AND T. PEIERLS, *Sparse partial pivoting in time proportional to arithmetic operations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 862–874.

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