IEEE Transactions on Power Systems, Vol. 5, No. 2, May 1990

SPARSE MATRIX INVERSE FACTORS

Mark K. Enns
Fellow IEEE
Electrocon International

William F. Tinney
Fellow IEEE
Consultant

Fernando L. Alvarado
Senior Member IEEE
The University of Wisconsin-Madison

Abstract - The inverses of matrix factors lend themselves to parallel operations in the direct solution phase of sparse matrix solutions. These inverse factors, given suitable ordering of the equations, are themselves sparse, if less so than the original factors. Partitioning reduces the build-up of nonzero elements in the inverse factors. All of the multiplications required for repeat solutions may be performed in parallel using the inverse factors, with only as many serial steps as twice the number of factors. KEYWORDS: Sparsity, Parallel computation, Direct Solutions, Partitioning, Inverse Factors.

INTRODUCTION

This paper deals with the structural and computational properties of the sparse matrixes encountered in various power system network analysis problems. Specifically, it is concerned with the inverses of the factors of sparse matrixes produced by factorization or decomposition. These inverse factors are themselves sparse, at least under suitable ordering and partitioning, and lend themselves to parallel operations in the direct or repeat solution phase of sparse matrix problems.

Parallel Processing

This work was motivated by the search for better ways to exploit parallelism in the solution of sparse matrix problems. There are two principal parts to such solutions: matrix factorization or decomposition, and direct or repeat solutions. Sparse matrix factorization seems to be essentially a serial process. The vectors that can be processed simultaneously, and thus in parallel, are too short to achieve much improvement over serial processing [1-3]. Attempts to increase the size of these vectors simply increase the number of operations required instead of decreasing solution times.

Direct solutions (as performed by forward and back substitution), appear to suffer similarly. The vectors that can be processed simultaneously are simply the rows or columns of the factors. We present here an entirely new approach to the parallel direct solution phase based on the factors of the inverse (or the inverse of the factors). These factors can be kept quite sparse while allowing most of the solution to be performed in parallel with only two to perhaps as many as eight serial steps for very large network problems.

This paper deals only with properties of the inverse factors, including partitioning for sparsity enhancement. The actual direct solutions through special-purpose parallel processors will form the subject of future papers.

Importance of Direct Solutions

Obtaining rapid direct or repeat solutions is especially important in problems that require multiple direct solutions for a single matrix factorization. Perhaps the main problems of this class are (static) contingency analysis and the solution of dynamic problems such as transient stability and electromagnetic transients by implicit integration. The recent partial matrix refactorization and factor update methods [4] increase the relative importance of direct solutions because complete factorization is needed less frequently.

88 SM 728-8 A paper recommeded and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1988 Summer Meeting, Portland, Oregon, July 24 - 29, 1988. Manuscript submitted January 22, 1988; made available for printing July 11, 1988.

Prior Work

Since the work reported here deals only indirectly with parallel processing, we shall not review the extensive literature that addresses this problem. However, we shall review briefly recent work that is related to or anticipates some of the developments reported here on inverse factors and their properties.

Alsaç, Stott, and Tinney [5] show that the inverse factors are sparse, but do not exploit this fact for parallel direct solutions. Van Ness and Molina [6] consider the problem of repeat solutions, and also propose a kind of partitioning. It is quite different from the partitioning proposed here, but has the same effect of increasing the number of serial steps. Gomez and Franquelo [7] propose new ordering algorithms that deal with the closely related problem of efficient fast forward and fast backward substitution as used in "sparse vector methods" [8].

Betancourt [9] is concerned with ordering to reduce the "path" lengths for efficient sparse vector solutions, again closely related to the computation and use of inverse factors for direct solutions. He even discusses briefly the inverse factors and notes, without explanation, that it is possible to improve their sparsity by what we refer to as partitioning.

MATHEMATICAL BASIS

We have the usual matrix-vector problem

$$A \cdot x = b$$

with A a sparse matrix. For simplicity in exposition, and with no loss in generality, we assume that A is symmetric. Then we may write

$$A = L \cdot D \cdot L^t \tag{1}$$

and solve

$$L \cdot z = b \tag{2a}$$

for z by forward substitution,

$$D \cdot y = z \tag{2b}$$

for y by division of z by the (diagonal) elements of D, and

$$L^t \cdot x = y \tag{2c}$$

for x by backward substitution.

For sparse A, the computation of L is preceded by an ordering of the axes (rows and columns of the matrix or nodes of a network) to minimize the number of nonzero terms.

Define

$$W = L^{-1}$$

The i^{th} column of W is the forward solution for a unit vector i, a vector with unity in position i and zeros elsewhere. Thus W is lower triangular and sparse. This was shown in [5] where certain terms of W are used to obtain quantities needed for mid-compensation. For a variety of systems ranging from 324 to 2199 nodes (Table III of [5]), the density of nonzero elements in W ranged from 2 to 7 percent, using ordinary minimum-degree ordering. This suggests the use of the W matrix to solve (1) as developed below.

The main idea of this paper is as follows. From equation (1) and the definitions of A and W,

$$x = A^{-1} \cdot b = (L \cdot D \cdot L^t)^{-1} \cdot b$$
$$= (L^t)^{-1} \cdot D^{-1} \cdot L^{-1} \cdot b$$
$$= W^t \cdot D^{-1} \cdot W \cdot b \qquad (3)$$

Equation (3) may be solved by the three serial steps

$$z = W \cdot b \qquad (4a)$$
$$y = D^{-1} \cdot z \qquad (4b)$$
$$x = W^t \cdot y \qquad (4c)$$

All of the multiplications implied by equations (4) may be performed in parallel. The additions cannot all be performed simultaneously. While the problem of additions will not be dealt with further in this paper, it turns out that they can be performed so as to cause essentially no delay in the solution using a practical number of multiplier units.

PARTITIONING

Some experimentation shows that the factor inverse matrix W may have too many nonzero elements for the proposed method to be practical for very large systems. The W matrix for a network based on a 5838-node, 9827-branch model of the Eastern U. S. interconnected system turned out to have nearly 440,000 nonzero elements. This is only 2.6% dense, but a formidable number of elements and operations nonetheless. There is no reason, however, why the solution (3) must be carried out in only two serial steps. (We are not counting the operations with the diagonal elements.) This section shows how the W matrix can be partitioned to enhance sparsity, with the solution carried out in one step per partition.

The matrix L can be expressed as

$$L = L_1 \cdot L_2 \cdots L_n$$

where each L_i is an identity matrix except for the ith column, which contains column i of L. Then

$$W = L^{-1} = (L_1 \cdot L_2 \cdots L_n)^{-1} = L_n^{-1} \cdots L_2^{-1} \cdot L_1^{-1} = W_n \cdots W_2 \cdot W_1$$

where the $L_i^{-1} = W_i$ are simply the L_i with the signs reversed on the off-diagonal elements. Using this expanded form of W in (3),

$$x = (L_1^t)^{-1} \cdot (L_2^t)^{-1} \cdots (L_n^t)^{-1} \cdot D^{-1} \cdot L_n^{-1} \cdots L_2^{-1} \cdot L_{n1}^{-1} \cdot b$$
$$= W_1^t \cdot W_2^t \cdots W_n^t \cdot D^{-1} \cdot W_n \cdots W_2 \cdot W_1 \cdot b \qquad (5)$$

Equation (5) is merely an expression of conventional forward and backward substitution. We are free, however, to combine the adjacent W matrixes in any useful way. Multiplying all of the W_i together produces W; combining all of the factors in (5) produces A^{I}. If the W_i and their transposes are each grouped into two factors then the solution for x may be expressed as

$$x = W_a^t \cdot W_b^t \cdot D^{-1} \cdot W_b \cdot W_a \cdot b \qquad (6)$$

As always, (6) is processed from right to left, in five serial steps, with all multiplications within each step amenable to parallel processing.

When applied to the 5838-node system referenced previously, partitioning W and W^t each into four parts (eight serial steps plus the diagonal operations in the solution) reduced the number of elements in W from 440,000 to 52,000. This compares with 31,000 elements in L itself. Thus the method presented here actually provides a continuum of solutions from conventional forward and backward substitution, as expressed by Eq. (5), through the three-step matrix-vector solution of Eq. (3), to development and use of A inverse. The number of nonzero elements increases with aggregation of the W factors, but the number of serial steps decreases.

Appendix A illustrates a simple example to clarify the mechanism of partitioning and the manner in which sparsity arises within the W factors.

ORDERING TO MINIMIZE NONZEROS IN W

The ordering schemes ordinarily employed in sparse matrix methods are directed toward minimizing the fill-in when factoring A to produce L and U. These schemes will not necessarily minimize the density of W, and in pathological cases can cause its complete fill-in. We therefore look for an ordering scheme that will minimize the number of nonzero elements in W (which will probably be less than optimal for L).

The scheme adopted is a modification of "Scheme 3" where, at each step, the node chosen introduces the fewest additional elements in W (instead of the fewest in L). This is readily done by creating the symbolic structure of W simultaneously with that of L as the ordering progresses. We call this new algorithm "Scheme W."

The following algorithm was used to produce the results presented below in a series of figures and in Table 1.

Scheme W

1. Initialization: For each node, calculate the fills in W that would result if the node were eliminated first. Call these "W-counts." Sort the nodes in increasing order of their W-counts. (This is "Scheme 1" applied to the W matrix. It in fact is the same as Scheme 1 applied to L because the structure of L and of W are exactly the same at this initial stage.)
2. If all nodes have been ordered, stop. Otherwise, select a node with the minimum W-count and symbolically eliminate it. Add fills to the structures of L and W.
3. Recalculate the W-counts for nodes whose connections were changed in step 2. Re-sort the W-counts and go to step 2.

The computational effort for Scheme W is equivalent to that of Tinney Scheme 3, although it necessarily takes longer because of the denser rows in W as the ordering progresses. Selective recalculation and reordering of only the nodes affected by the previous selection and elimination are the keys to efficient implementation. Appendix B discusses further computational complexity issues.

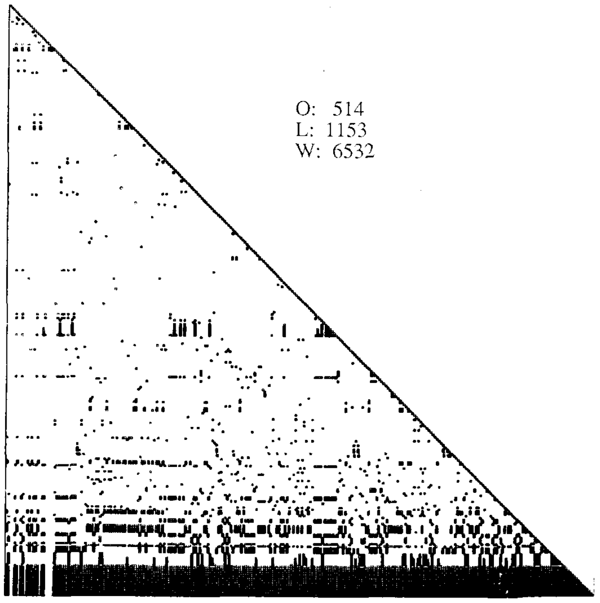


Figure 1a. Scheme 2 applied to dense 284-node system. "O" means nonzeroes in original matrix, "L" means total nonzeroes in L factor and "W" means total nonzeroes in inverse factor(s).

Figures 1 and 2 show the results of applying Scheme 2 and Scheme W to two networks. The first is an unusually dense system resulting from network reduction. The second is a 535-node system of typical structure. That Scheme W produces fewer nonzero elements in W is apparent from a glance at the comparative figures. The element counts given in the figures, and later in Table 1, show that this reduction is achieved at a modest increase in the density of L.

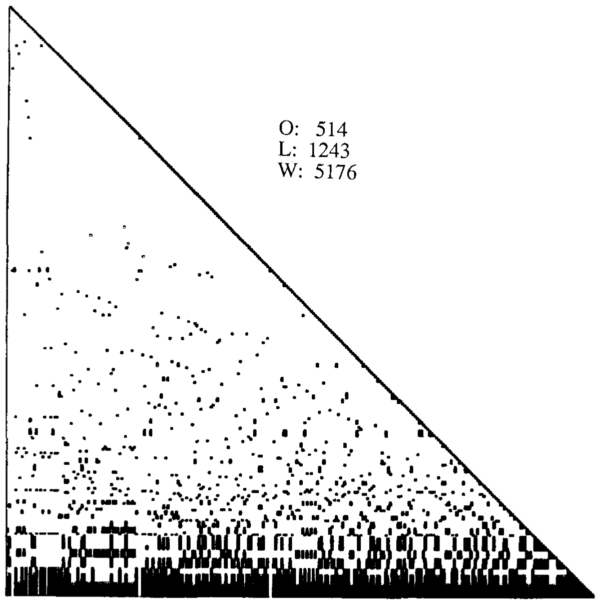


Figure 1b. Scheme W applied to dense 284-node system.

A tridiagonal matrix presents an interesting special case. The W matrix resulting from ordinary Scheme 2 ordering is completely full, although there are only (exactly) 600 nonzero off-diagonal elements in L. Scheme W (no figure shown) works quite well.

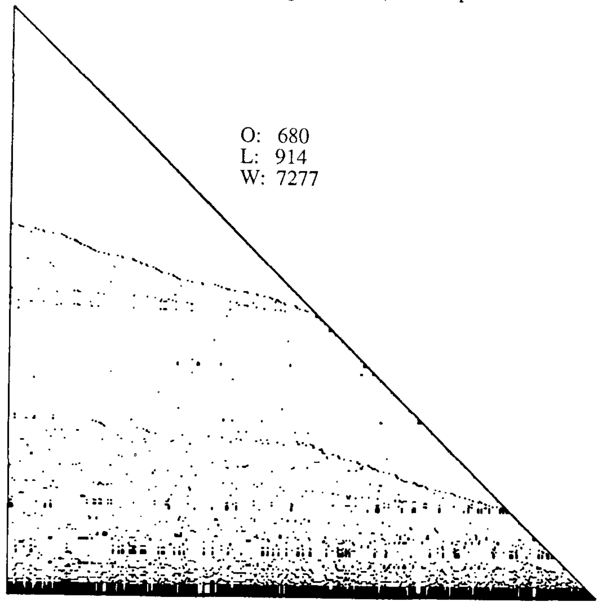


Figure 2a. Scheme 2 applied to normal 535-node system.

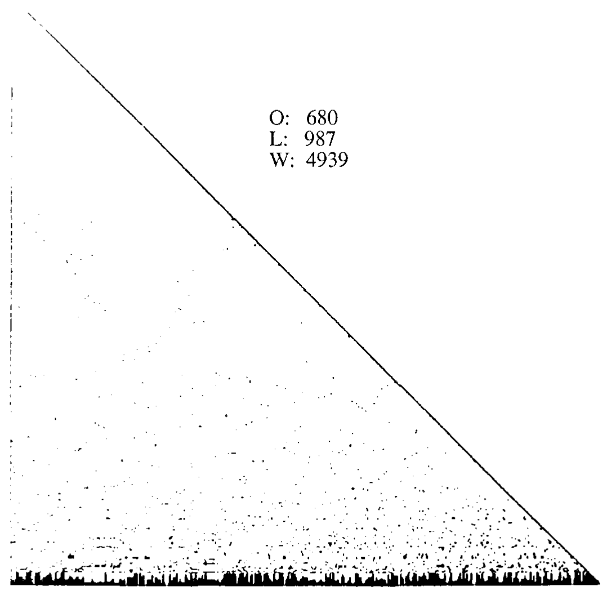


Figure 2b. Scheme W applied to normal 535-node system.

Ordering Scheme W is important for conventional serial processing as well as for parallel processing. The objective of Scheme 2, which is to minimize the number of nonzero elements in L, makes sense for applications where the main computational burden is factorization or solutions with full vectors. For applications where the main burden is repeat solutions with sparse vectors [7], Scheme W is better. The decrease in the number of nonzero elements in W produced by Scheme W over the number produced by Scheme 2 is well worth the modest increase in the number of nonzero elements in L.

Partitioned Scheme W

Scheme W as presented above produces a "sparse" W matrix and works satisfactorily for systems with up to some hundreds of nodes. However, as discussed above, the number of nonzero elements in W becomes excessive for large networks. We show in this section how W is partitioned and the effect on sparsity. The structure of a W partition is just the union of the structures of the adjacent W_i forming it. It differs from the identity matrix only in the columns corresponding to their indexes. For example, if W_i, W_j, and W_k are combined into one partition, then the partition will only have, in addition to the unity diagonal, nonzero elements below the diagonal in columns i, j, and k.

The partitioning rule used is very simple: when the number of nonzero elements in a partition reaches a predetermined limit, start a new factor. The modification is effective in reducing the number of nonzero elements because most of the fill-in occurs precipitately towards the end of the ordering. Each factor starts fresh as an identity matrix and only a few fills are produced by the first nodes selected. There may be superior partitioning criteria, but none was tried.

The effects of partitioning are shown in figures 3 to 5. Figures 3 and 4 may be compared with their unpartitioned counterparts in figures 1(b) and 2(b). In each case, the main difference is in the last few rows. The partitioning stops those rows from filling heavily in the first (of two) partitions. Figure 5 shows the very favorable results obtained by partitioning on a 5838-node system. The reason for its effectiveness is intuitively clear from noting the increasing density of the successive, smaller partitions. The results for the three systems discussed above plus a 600-node tridiagonal system are summarized in Table 1.

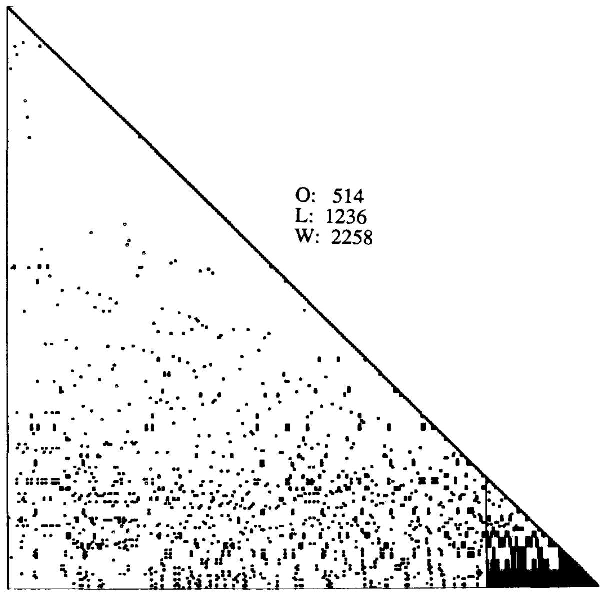


Figure 3. Partitioned Scheme W applied to 284-node system.

Discussion

In each case, and in a number of other cases that were tried but not reported on here, some experimentation was performed in choosing the number of nonzero elements per partition. The goal is to obtain the minimum number of partitions that yields an acceptable total number of nonzero elements, or density of W. This is controlled by choosing a cutoff number of elements for each partition.

Figure 6 shows the results of these experiments on nine different systems, including systems arising from finite-element analysis as well as from power networks. In every case, the density or total number of nonzero elements is monotonically nonincreasing with the number of partitions. Note that in some cases there are significant differences in density for different cutoffs that happened to produce the same number of partitions. This is an artifact of the "end effect" of the size of the last partition, and usually important only for two partitions. For some of the larger systems it was not possible to obtain an unpartitioned W without a clearly excessive number of elements. Finite-element systems are typically much denser than power systems; the count for the 2360-node finite-element system was stopped with 11 partitions.

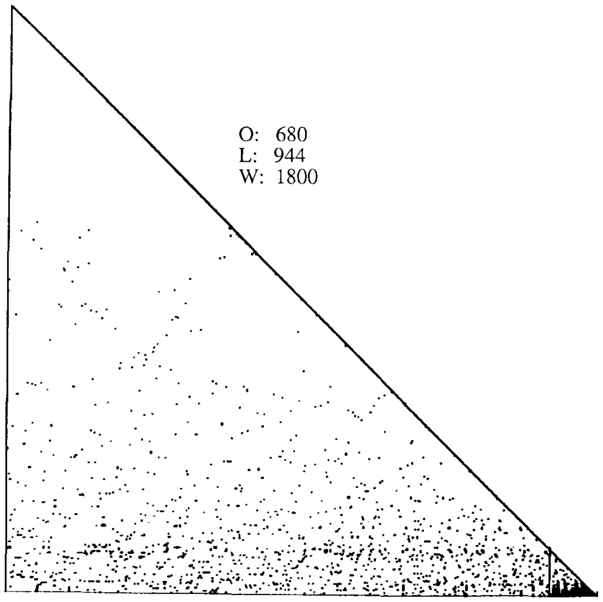


Figure 4. Partitioned Scheme W applied to 535-node system.

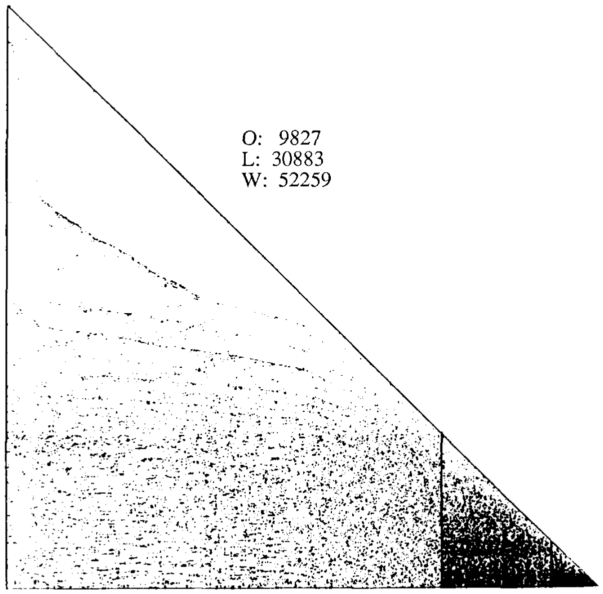


Figure 5. W matrix for 5838-node system with four partitions.

CONCLUSIONS

A new approach has been described that lends itself to the parallel processing of the extremely sparse matrix equations of power network problems. Instead of using the triangular factors of a matrix to compute direct solutions as is now done in serial processing, the new approach uses the inverse of these triangular factors. Unlike the inverse of a sparse matrix, which is full, the inverses of sparse triangular factors are sparse, though less sparse than the factors themselves. Sparsity can be enhanced by partitioning the inverse triangular factors, making the approach practical for even the largest power network problems.

A by-product of the investigation was a new ordering scheme for triangular factorization that tends to minimize the number of nonzero factors in the inverse triangular factors. The new scheme -- Scheme W -- could also be of benefit for serial processing in applications that make many repeat solutions with sparse vectors. The practical use of these methods for parallel processing of sparse matrix problems will be the subject of future papers.

ACKNOWLEDGEMENTS

The authors acknowledge Sao Khai Mong for programming and producing the results given in Figures 1 through 5 and Table 1, and Dr. Donald MacGregor for producing the results of Figure 6.

REFERENCES

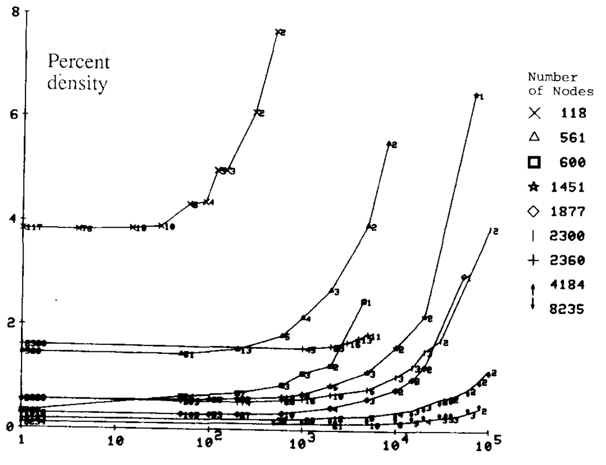
[1] D. A. Calahan, "Parallel Solution of Sparse Simultaneous Linear Equations," Proceedings 11th Allerton Conference on Circuit and System Theory, University of Illinois, October 1983.

Table 1

Comparison of Scheme 2 and Scheme W

System	Matrix	Scheme 2	Scheme W	Partitioned Scheme W	Comments on partitions
Dense 284-bus system system (from network reduction)	O L W W%	514 1,153 6,532 16.31	514 1,243 5,176 12.93	514 1,236 2,258 5.62	230 nodes and 1,500 elements in first of two partitions
Typical 535-bus system (Node to branch ratio = 1.27)	O L W W%	680 914 7,277 5.09	680 987 4,939 3.46	680 944 1,800 1.26	493 nodes and 1,500 elements in first of two partitions
600-node tridiagonal system	O L W W%	600 600 full 100	600 1,677 4,530 2.52	600 1,273 2,059 1.15	493 nodes and 1,500 elements in first of two partitions
5,838-bus model Eastern United States system	O L W W%	9,827 28,620 † †	9,827 29,914 437,588 2.57	9,827 30,883 52,259 0.31	Partitions at 4,285, 5,367 and 5,725 with 16,000 elements in first 3 of 4 partitions

(†) Too many elements exceeded the capacity of the computer.



Maximum number of elements per partition

Figure 6. Density of inverse factors for various matrixes. The numbers along the curves indicate the number of partitions. For all systems, the most dramatic reduction in density takes place during the first few partitions.

[2] H. H. Happ, C. Pottle, and K. A. Wirgau, "An Assessment of Computer Technology for Large Scale Computer Simulation," Proceedings IEEE PICA Conference, pp. 316-24, 1979.

[3] M. K. Enns, D. E. Atkins, and R. M. Howe, "Application of Specialized Digital Computation to Power System Network Analysis Problems," in Exploring Applications of Parallel Processing to Power System Analysis Problems, Seminar Proceedings Report EPRI EL-566-SR, Electric Power Research Institute, Palo Alto, CA, pp. 249-64, Oct 4-7, 1977.

[4] S. M. Chan and V. Brandwajn, "Partial Matrix Refactorization," IEEE Transactions on Power Systems, Vol. PWRS-1, No. 1, pp. 193-200, February 1986.

[5] O. Alsac, B. Stott, and W. F. Tinney, "Sparsity-Oriented Compensation Methods for Modified Network Solutions," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No. 5, pp. 1050-60, May 1983.

[6] J. E. Van Ness and G. Molina, "The Use of Multiple Factoring in the Parallel Solution of Algebraic Equations," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No. 10, pp. 3433-38, October 1983.

[7] A. Gomez and L. G. Franquelo, "Node Ordering Algorithms for Sparse Vector Method Improvements," IEEE PES Winter Meeting, Paper 87 WM 038-3, February 1987.

[8] W. F. Tinney, V. Brandwajn, and S. M. Chan, "Sparse Vector Methods," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 2, pp. 295-301, February 1985.

[9] R. Betancourt, "An Efficient Heuristic Ordering Algorithm for Partial Matrix Refactorization," IEEE PES Summer Meeting, Paper 87 SM 445-0, July 1987.

[10] F. L. Alvarado, "Computational Complexity in Power Systems," IEEE Trans. on Power Apparatus and Systems, Vol PAS-95, No. 4, July/August 1976, pp. 1028-1037.

[11] M. K. Enns, F. L. Alvarado and D. M. McGregor, "A Parallel Repeat Solver for Sparse Matrices," NSF Final Report to Award ECE-8460029, July 31, 1985.

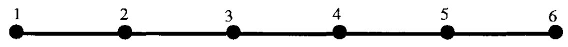
Mark Enns is President of Electrocon International, Inc. in Ann Arbor, Michigan, where he directs the development of the company's software products for electric utilities. His personal research interests are in methods and algorithms for fast solution of large-scale network problems such as short circuit, power flow, and contingency analysis. He has previously taught electrical engineering at the University of Michigan and Carnegie-Mellon University, and prior to that worked for Westinghouse in Pittsburgh, Pa. He received a B. S. degree from Kansas State University, and M.S. and Ph.D. degrees from the University of Pittsburgh, all in electrical engineering.

William F. Tinney received B.S. and M.S. degrees from Stanford University in 1948 and 1949. He worked for the Bonneville Power Administration from 1950 until his retirement in 1979, at which time he was head of System Analysis. Most of his work has been concerned with power system computer applications and he is presently a consultant in the field.

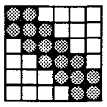
Fernando L. Alvarado was born in Lima, Peru in 1945. He received the BEE and PE degrees from the National University of Engineering in Lima, Peru, the MS degree from Clarkson College (now Clarkson University) in Potsdam, New York, and the Ph.D. degree from the University of Michigan in 1972. He joined the University of Toledo as an assistant professor in 1972. Since 1975 he has been with the University of Wisconsin in Madison, where he is currently a Professor in the Department of Electrical and Computer Engineering. He has authored papers in the area of computer applications to engineering in general and power systems in particular. His main interests are in the area of large scale nonlinear networks, power system economics and control, symbolic computation, computing paradigms, and robotic control.

APPENDIX I: ILLUSTRATIVE SMALL EXAMPLE

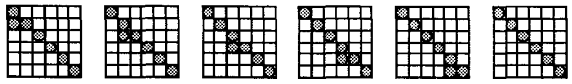
This appendix is intended to clarify the mechanism of partitioning and the manner in which sparsity arises and is enhanced by partitioning. We do this by means of a small example. Consider the following purely radial 6-node system:



The topology of the sparse matrix associated with this system is:

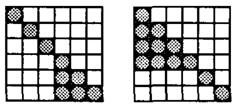


Ordinary factorization of this matrix results in a lower triangular matrix L, a diagonal matrix D, and an upper triangular matrix L^t. The lower triangular matrix L can be expressed as the product of elementary lower triangular matrixes L_i. The structure of the elementary lower triangular matrixes L_i *and of the inverse elementary matrixes* $W_i = L_i^{-1}$ is the same and is given by:



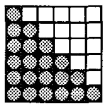
L_1, W_1 L_2, W_2 L_3, W_3 L_4, W_4 L_5, W_5 L_6, W_6

The off-diagonal values of L_i and W_i have opposite signs but are otherwise identical. L_6 and W_6 are identity matrixes and could be omitted. Consider the 3-3 grouping of the these matrixes: combine the last three into a single matrix by carrying out the product of elementary matrixes, and do the same to the first three. The result is:



$W_6 W_5 W_4$ $W_3 W_2 W_1$

These partial products retain a number of nonzeroes. *This is true even if the original matrix A is full.* If we group all elementary matrixes into a single group, we obtain the topology of L^{-1}:



In this pathologically simple case, L^{-1} is full. In general, however, sparsity remains even after all elementary inverse lower triangular factors are grouped into a single inverse factor. Only when the complete product $L^{-1}D^{-1}(L^t)^{-1}$ is carried out does the inverse matrix fill up.

Once these factors are obtained, the solution to Ax=b is obtained from equation (6).

APPENDIX II: COMPUTATIONAL COMPLEXITY ISSUES

The number of nonzeroes after factorization for typical sparse system matrixes grows approximately as a $n^{1+\gamma}$, where n is the number of nodes in the network and γ is approximately 0.2 for typical networks. Under these conditions, a properly programmed forward or back substitution routine exhibits computational behavior of order $n^{1+\gamma}$, factorization exhibits computational complexity behavior of order $n^{1+2\gamma}$, and sparse inversion exhibits behavior of order $n^{2+\gamma}$ [10]. Investigations of algorithmic behavior indicate that proper implementations of the Tinney scheme 2 algorithm exhibit behavior of order $n^{1+2\gamma}$ and proper implementations of the Tinney scheme 3 exhibit behavior of order $n^{1+3\gamma}$.

The algorithms presented in this paper do not depend on the number of nonzeroes in the factors of a sparse matrix, but rather on the number of nonzeroes in the partitioned inverse factors. Assume that the growth in the number of nonzeroes in these inverse factors is of order $n^{1+\xi}$. These inverse factors remain in general sparse, suggesting that ξ is less than 1. These inverse factors are, however, denser than the original factors, suggesting that perhaps ξ is greater than 0.2 for typical systems. A systematic investigation of this computational growth is needed. Regardless of the probable value of ξ, however, the "Scheme W" algorithm presented in this paper exhibits computational complexity of the same order as Tinney Scheme 3, but based on the density of the inverse factors. Thus, the computational complexity of Scheme W is about $n^{1+3\xi}$.

Another computational complexity issue that arises from this paper is the computational complexity associated with the *use* of the inverse factors for repeat solutions. In a serial environment, the computational complexity associated with the use of inverse factors is that of matrix-vector multiplications. The order is equal to the number of nonzero elements in the inverse factors. For full inverses, this is of order n^2. For sparse inverse factors, this computation is of order $n^{1+\xi}$. In a parallel environment, all multiplications of a factor times a vector can be done in parallel. However, the required additions associated with matrix-vector multiplications remain somewhat serial. Simple pair-wise addition algorithms for adding n numbers are known to be of order $\log_2 n$. Assuming that there are p partitions and that no advantage is taken of pipelining and the sparsity of the vectors, the computational complexity for additions can be seen to be of order $p \log_2 n$. For further details on parallel architecture implementations based on these concepts, see [11].

APPENDIX III: COMPARATIVE TESTING OF ORDERING ALGORITHMS

The table below shows ordering times for Scheme 2 and Scheme W. Because specific algorithmic implementations can vary with the machine and compiler used and the skill of the programmer, comparisons of actual computing times for the various algorithms are not very meaningful for an "in principle" paper such as this one. However, it was considered of some value to include a few timing comparisons to give a general idea of the relative times in question. These comparisons were done for a number of examples using a Pascal implementations of both Scheme 2 and Scheme W on an Apollo DN3000. Some explanation is needed about these timing figures. The Scheme W implementation requires the construction of the W matrixes. Scheme 2 does not require the construction of the W matrixes, unless Scheme 2 is used for building W as well. Clearly, if W is constructed along with L, the time for a pure Scheme 2 as it is ordinarily used in sparse matrix computations would appear unnecessarily large. Thus, we elect to illustrate four different times in these tables. The first is for ordering a matrix using Scheme 2 alone (W is not built). The second time is that to build L and W using Scheme 2. The third time is that required to build W using Scheme W. The last time shown is the time to build W if a new W partition is started whenever an arbitrarily chosen number of nonzeroes is reached. As we mention in the paper, much research remains to be done on the optimal partitioning of matrix inverse factors.

	Elements	Time in Seconds			
Matrix size	Scheme 2 Density	Scheme 2 L only	Scheme 2 W built	Scheme W No part.	Scheme W Partitioned
115	394	0.50	0.92	1.03	0.89(2)
233	844	0.80	1.97	3.52	2.35(2)
600	1199	1.14	218**	5.37	4.09(2)
1846	4459	4.17	12.32	21.82	22.48(2)
1447	8441	9.51	200.2	658.2	120.7(3)
3000	19037	26.34	912.0	3214	1043(3)
4184	23333	33.56	2216	*	1149(4)
8235	51603	71.22	*	*	4367(5)

Scheme 2 density means number of elements in L after factorization using Scheme 2 (including the number of diagonal elements).
Number of partitions is shown in parenthesis.
(*) Available memory exceeded, W matrix too dense.
(**) The unpartitioned W matrix for a tridiagonal matrix is full.

Thanks are due to Ms. Zian Wang for helping prepare this table.

Discussion

James E. Van Ness (Northwestern University, Evanston, IL): The number of operations, multiplications, divisions, additions and subtractions, needed either to do a forward and back substitution using the L and U factors or to do the equivalent matrix multiplications in equation 4 of the paper is a linear function of the number of nonzero elements in the matrices involved. Thus the time that will be required for these operations can be measured quite accurately by the number of nonzero elements in the matrix. If this is used as a measure, the partitioning method describe in this paper does not appear to be very effective. The examples given in this paper give a growth in the number of nonzero elements from the L matrix to the final matrix formed using the partitioned scheme W varying between 62% and 91%. The partitioning described in reference 6 of the paper gave a growth that varied from 5% to 12% on similar but different examples. If this was all that there was to the story, the new method would seem to have no advantage.

Many people have tried to solve sparse sets of equation in parallel. The problem invariably is that precedent relationships cause the processors to have a large amount of idle time while they wait for other processors to complete some preceding task. The result is that the expected time saving from parallel computation is not realized. A major advantage of the method presented here is that the numerical operations are independent of each other, and it should be quite straight forward to schedule the tasks into multiple processors. The question is whether the load of the extra elements introduced in forming the W matrix will be offset by the scheduling advantage.

The method describe in reference 6 of the paper was studied further on a simulated network of parallel processors. The results are given in an EPRI report [1]. The losses observed there were: time for the control function, varying from less than 1% to 25%; time for communications, from 1% to 36%; and idle time not in the other categories, 1% to 92%. These losses are functions of the configuration, characteristics of the hardware, and the partitioning scheme used. It will be interesting to see how the method of this paper works on an actual parallel processors.

Reference

1. *Multiple Factoring in the Parallel Solution of Algebraic Equations.* Palo Alto, Calif.: Electric Power Research Institute, March 1985. EL-3893.

Author released without Closure.

Manuscript received June 24, 1988.