SPARSE VECTOR METHODS

W.F. Tinney
Fellow, IEEE

Consultant
9101 S.W. 8th Avenue
Portland, OR 97219

V. Brandwajn
Member, IEEE

S.M. Chan
Member, IEEE

Systems Control, Inc.
1801 Page Mill Road
Palo Alto, CA 94304

Abstract - Sparse vector methods enhance the efficiency of matrix solution algorithms by exploiting the sparsity of the independent vector and/or the desire to know only a subset of the unknown vector. This paper shows how these methods can be efficiently implemented for sparse matrices. The efficiency of the sparse vector methods is verified by tests on a 156-bus, a 1598-bus and a 2265-bus system. In all cases tested, the new methods are significantly faster than the established sparse matrix techniques.

INTRODUCTION

Sparse matrix methods are now used for solving almost all large power network problems. On the other hand, sparse vector methods, which extend sparsity exploitation to vectors, appear to be largely neglected. This paper explains sparse vector methods and gives some practical results to demonstrate the benefits of their use for power network problems.

Sparse vector methods are useful for solving a system of simultaneous linear equations when 1) the independent (right-hand-side) vector is sparse, and/or 2) a small number of elements in the unknown vector is wanted. The efficiency of sparse vector methods results from the elimination of all unnecessary matrix operations. These methods can be applied to both full- and sparse-matrix equations.

This paper focuses on the implementation of sparse vector methods in a sparse-matrix setting. It will be shown that it is relatively easy to take advantage of vector sparsity, and the resulting speed-up can be quite dramatic. Investigators seem to have overestimated the difficulty or underestimated the benefits of exploiting vector sparsity; otherwise sparse vector methods would be more widely used.

The primary motivation for using sparse vector methods is to obtain faster solutions to problems having sparse vectors. Such problems are numerous. Sparse vector methods are indispensable for efficient compensation methods [1]. Important uses have also been made in fault analysis [2], optimal power flow [3] and economic dispatch [4]. Other identified applications include contingency analysis, several forms of equivalencing, and partial refactorization. It is probably not an exaggeration to say that appropriate use of sparse vector methods would enhance any large power network application.

84 SM 607-8 A paper recommended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1984 Summer Meeting, Seattle, Washington, July 15 - 20, 1984. Manuscript submitted January 30, 1984; made available for printing May 3, 1984.

The importance of sparse vector methods in power system analysis warrants an in-depth explanation of their derivations, benefits and implementation details. This paper is written to fill this need.

SUMMARY OF SPARSE FACTORIZATION

A general system of linear equations can be symbolized as:

$$\mathbf{A}\, x = b \tag{1}$$

where **A** is a nonsingular matrix of order N, b is a given independent vector, and x is an unknown solution vector. Matrix A can be factored as

$$\mathbf{A} = \mathbf{L}\,\mathbf{D}\,\mathbf{U} \tag{2}$$

where L and **U** are lower- and upper-triangular matrices with unity diagonals, respectively, and D is a diagonal matrix. (Note: the sparse vector methods are equally applicable to the case where **A** is factored as **A** = **LU**.) If A is symmetric, the matrix U is the transpose of L. If **A** is incidence symmetric, the matrix **U** is not the transpose of L, but they are identical in sparsity structure.

Sparse vector methods are applicable to any nonsingular matrix A, but incidence symmetry is assumed to simplify this presentation. Also, for convenience of an example, the matrix **A** is assumed to be a matrix of coefficients for nodal network equations.

The solution vector x for a given vector b is obtained by operating on b with the factors of A:

$$y = \mathbf{D}^{-1}\mathbf{L}^{-1}\, b \tag{3a}$$

$$x = \mathbf{U}^{-1}\, y \tag{3b}$$

Equation (3a) defines a sequence of forward substitution operations on **LD**, while (3b) defines a sequence of back substitution operations on U. The operations can be performed by rows or columns. For sparse vector methods it is essential that the forward solution of (3a) be performed by columns and the back solution of (3b) by rows. (See Appendix A for more details).

Many different schemes can be used for storing and accessing L and **U**. For sparse vector methods it is essential that the number of the lowest-numbered, nonzero, off-diagonal element in each column of L (or row of **U**) be directly accessible without search. This requirement is satisfied by most schemes for storage of **L** and **U**; it is not in any way a handicap.

0018-9510/85/0002-0295$01.00©1985 IEEE

SPARSE VECTOR SOLUTIONS

The independent vector b is sparse in many applications. However, the solution vector x is not sparse in general. The term "sparse vector" in the following refers either to a sparse vector b, or to a subset of vector x containing only the elements of interest. The exact meaning is always clear from the context.

If the vector b is sparse, only a subset of the columns of L is needed for the forward solution of (3a). This is called the fast forward (FF). If only certain elements of vector x are actually wanted, only a subset of the rows of **U** is needed for the back solution of (3b). This is called the fast back (FB). The columns of FF correspond one for one with the nonzeros in vector y of (3a), while the rows of FB correspond one for one with the desired subset of x of (3b). The essential task of sparse vector methods is to efficiently identify the subsets of L and **U** for FF and FB.

The subset of columns for FF is a function of the sparsity structures of L and b. The subset of rows for FB is a function of the sparsity structures of U and x. The sparsity structures of L and U, in turn, are functions of the sparsity structure of **A** and the sparsity ordering algorithm.

A measure of the relative advantage of using FF instead of a full forward solution is the ratio R where:

$$R = \frac{\text{Number of nonzeros in the columns used in FF}}{\text{Number of nonzeros in L and D}} \qquad (4)$$

The ratio R for FB is the same except it applies only to the rows of **U**. Each nonzero element in the columns of FF defines one multiply-add operation in (3a). For some applications the number of nonzeros in vector y may also be a meaningful measure of relative advantage.

FACTORIZATION PATHS

The sparse vector method for forward operations (3a) is an improvement over the following simple, but inefficient, algorithm:

1. Zero all locations in b, and enter the given nonzero elements in b. Let k be the location of the lowest-numbered nonzero element.
2. Perform the forward solution operations defined by column k of L on b.
3. Search forward in b from k to the next higher-numbered nonzero element. Let k be this number.
4. If k = N, exit. Else, return to Step 2.

This algorithm ensures that only the necessary nonzero operation of FF are performed, but it is wasteful because of the zeroing and searching. A similar, but even more wasteful, method is required for FB. More efficient algorithms are based on the concept of factorization paths.

A factorization path (or "path" for short) for a sparse vector is defined as an ordered list of columns of L for FF, or rows of **U** for FB. A path is executed in forward order for FF and in reverse order for FB. The same or different paths may be used for FF and FB in (3) depending on the application.

The path for a singleton is basic to the path concept. A singleton is a vector with only one nonzero element; the nonzero is in location k. The following algorithm determines the path of a singleton:

1. Let k be the first number in the path.
2. Get the number of the lowest-numbered nonzero element in column k of L (or row k of **U**). Replace k with this number, and list it in the path.
3. If k = N, exit. Else, return to Step 2.

The path for a singleton can be determined directly from the indexing arrays without searching or testing. It is a compact description of the sparsity structure of column k of L^{-1} or row k of U^{-1}.

A general sparse vector is the sum of singletons, and its path is the union of the paths of its composite singletons. For any given sparse system represented by (1), a path can be associated with any sparse vector b or x. In a more general way, the path concept can be applied to any group of variables or group of nodes of the system. For example, the path of a subnetwork has the same meaning as the path of the group of nodes contained in the subnetwork, the group of variables associated with these nodes, or a sparse vector whose nonzeros are in locations corresponding to the group of variables.

The properties of paths of general sparse vectors can best be explained by an example. The 20-node network of Figure 1 is used. Figure 2 shows the sparsity structure of an incidence symmetric admittance matrix A of the network. The sparsity structure of its triangular factor L is also shown on Figure 2.

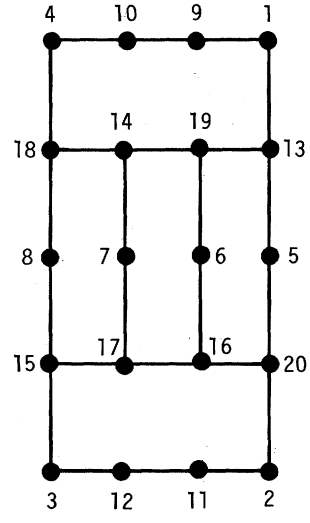

Figure 1: A 20-Node Example Network

A Path Table for the network, which compactly describes all of its singleton paths, is shown in Table I. Any singleton can be traced directly from the Path Table by starting at node k and tracing forward as guided by the column NEXT. For example, the singleton path for k=4 is {4, 10, 13, 18, 19, 20}. It should be noted that the Path Table is never formed in practice because the equivalent information is embedded in the indexing arrays of L (and U).

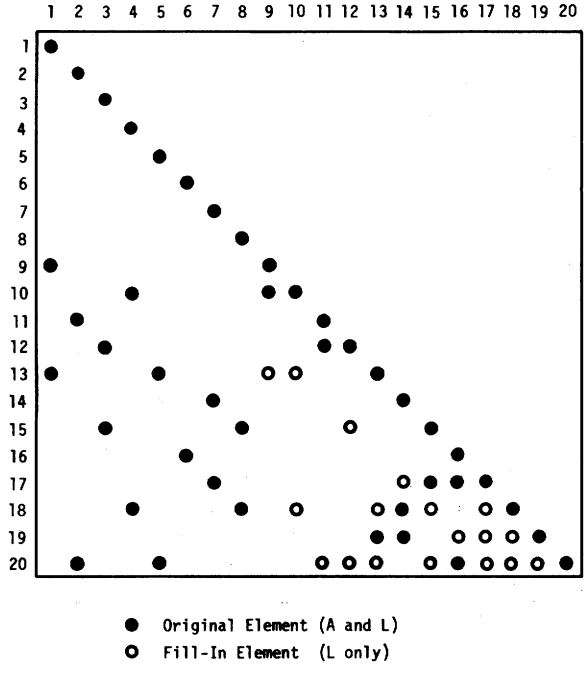

Figure 2: Sparsity Structure of Matrices A and L.

Table I: Path Table

NODE	NEXT	NODE	NEXT
1	9	11	12
2	11	12	15
3	12	13	18
4	10	14	17
5	13	15	17
6	16	16	17
7	14	17	18
8	15	18	19
9	10	19	20
10	13	20	0

A pictorial view of the Path Table is provided by the Path Graph shown in Figure 3. The graph has a direction sense from low to high node for FF and from high to low node for FB.

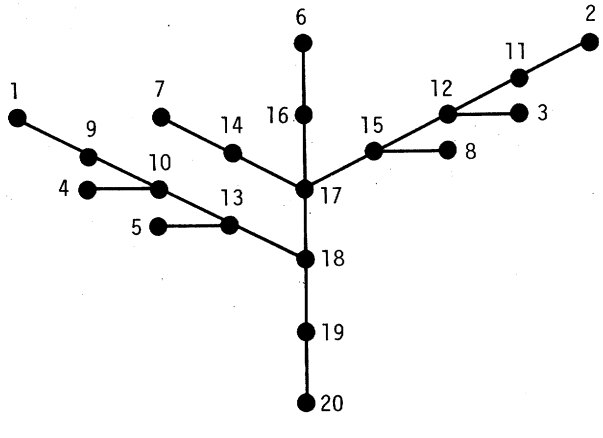

Figure 3. The Path Graph

When working with the indexing arrays -- or conceptually, the Path Table -- a path is traced out in the forward order. The following algorithm is one of several for finding a path for a sparse vector with more than one nonzero.

1. Start with the lowest-numbered nonzero in the vector and find its singleton path as previously explained. This path segment will always end on node N.
2. Identify the next lowest-numbered nonzero node not already in the partially completed path, and trace its path in the same manner until a node already in the partially completed path is encountered.
3. Join this newly formed path segment to the front end of the partially completed path list.
4. If all nodes in the original sparse vector are in the path, exit. Else, return to Step 2.

As an example of how the algorithm works, consider a sparse vector with nonzero in locations 2, 6, 7 and 12. The initial partial path found by Step 1 is {2, 11, 12, 15, 17, 18, 19, 20}. The second path segment is {6, 16} and the third and last is {7, 14}. The total path, assembled from the three segments in the reverse order in which they were found, is {7, 14, 6, 16, 2, 11, 12, 15, 17, 18, 19, 20}.

Pictorally, the path for Node 2 is formed by tracing a path from this node, in ascending node sequence, to the last node (Node 20). The partial paths emanating from nodes 6 and 7 are traced out similarly until they meet the already-existing path at a junction (Node 17). No additional path is required for Node 12 because it is contained in the path of Node 2. The subgraph (Figure 4) containing the union of these paths defines the subset of columns of L for FF, or rows of U for FB.

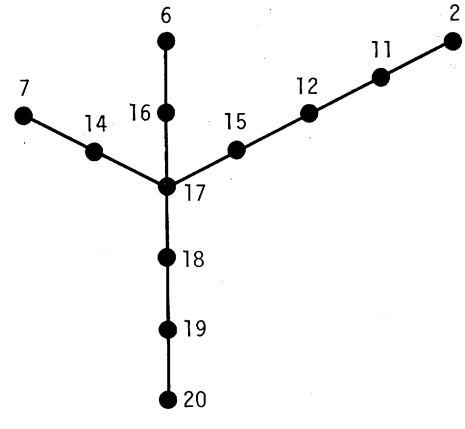

Figure 4: The Example Subgraph

The subset of the columns of L (or rows of U) defined by the path for any given sparse vector is unique, but there are usually many different permutations of the sequence in which the columns and rows can be used for FF or FB. A certain precedence rule must be observed for the sequence to be valid.

For the purpose of deriving the precedence rule, equation (3a) for forward substitution can be interpreted as the nodal equation of a directed network in which each node is affected by only the lower-numbered

nodes in the subgraph. This interpretation implies that the value of a node in FF cannot be computed until the values of all lower-numbered nodes in the subgraph are known. The branches above a junction, however, can be computed in any order. For example in Figure 4, the FF operations for columns 7, 14, 6, 16, 2, 11, 12 and 15 must be finished before the operations for column 17, but the order in which the branches {7, 14}, {6, 16}, and {2, 11, 12, 15} are handled is immaterial as long as the same rules are followed within each branch. Sorted order is always valid.

An analogous precedence rule applies to the back substitution (3b).

Although all possible permutations of the path sequence that fulfill the precedence requirements for FF/FB involve identical arithmetic operations, there are advantages to the sorted sequence. This is because a sorted path makes the most efficient use of virtual memory and, therefore, results in the fastest wall-clock execution time of FF/FB. Highly efficient algorithms for finding sorted paths have been developed based on the scheme outlined. Each path segment is sort-merged into the partially completed path as it is found. Typically, the initial segment is long, and the additional segments are shorter.

INITIALIZATION OF SPARSE VECTOR OPERATIONS

Zeroing of locations outside of the path should be avoided whenever possible. Depending on circumstances, some or all zeroing can be avoided. The requirements for zeroing depend on four possible conditions as follows:

1. FF followed by FB on same path - Zero only the locations in the path.
2. FF on one path followed by FB on another path - Zero the union of the paths.
3. FF followed by full back solution - Zero all locations.
4. Full forward solution followed by FB - No zeroing necessary.

Under condition 3 zeroing of locations outside of the path can be combined with the back solution rather than being done as an initialization step.

EXPERIMENTAL RESULTS

The efficiency of sparse vector methods is investigated using the positive-sequence nodal admittance matrix of three systems: a 156-bus system with 204 branches, a 1598-bus system with 2114 branches, and a 2265-bus system with 3043 branches.

The first scenario assumes that the vector b is a singleton, and all the elements of x are wanted. Only the FF is useful in this case; the full back operation is necessary to compute all the elements in the unknown vector. The number of multiply-adds in this procedure is compared to that of the full-vector methods. Several variants of the full-vector methods are considered, including some that take advantage of the leading zero entries in b. To differentiate among these variants, the notation "(k:N, N:m)" is used to denote a full forward operation that begins at position k and ends at position N, followed by a full back operation that begins at position N and ends at position m. The end point, m, is 1 for this test. In other tests, m may be the larger to avoid the computation of x_1 to x_{m-1}.

The two variants of the full vector-method considered are (1:N, N:1) and (k:N, N:1), where k is the position of the nonzero entry in b. The second version (k:N, N:1) is more efficient because it avoids the unnecessary forward operations associated with the first k-1 columns of L.

The test for each system consists of N runs for every possible singleton b. In each run the following two ratios are calculated:

$$R1 = \frac{\text{total mult-adds in FF and full back}}{\text{total mult-adds in full method (1:N, N:1)}}$$

$$R2 = \frac{\text{total mult-adds in FF and full back}}{\text{total mult-adds in full method (k:N, N:1)}}$$

The mean and standard deviation (s.d.) of R1 and R2 are tabulated in Table II. Inspection of the table shows clearly the advantage of the FF over full-foward methods.

TABLE II

	R1		R2	
	Mean	s.d.	Mean	s.d.
156-bus	50%	3%	70%	16%
1598-bus	43%	1%	60%	14%
2265-bus	46%	1%	61%	13%

The second scenario assumes that the vector b is a singleton and only the value of x_k is required. Both the FF and the FB are applicable to this case. Again, N runs are made for every possible singleton in the system. The ratios of interest in each run are:

$$R3 = \frac{\text{total mult-adds in FF and FB}}{\text{total mult-adds in full method (1:N, N:1)}}$$

$$R4 = \frac{\text{total mult-adds in FF and FB}}{\text{total mult-adds in full method (k:N, N:k)}}$$

The statistics of R3 and R4 are shown in Table III. The combination of FF and FB in this case produces a dramatic improvement over conventional methods. Results for the two large systems indicate that the sparse vector method is on the average 7 to 8 times faster than conventional methods that permit specification of starting and ending points, and 15 to 20 times faster than those that do not.

TABLE III

	R3		R4	
	Mean	s.d.	Mean	s.d.
156-bus	15%	6%	36%	25%
1598-bus	5%	1%	12%	14%
2265-bus	7%	2%	15%	14%

The efficiency of the sparse vector methods is a result of the short factorization paths associated with singletons. Experimental results show that the average path lengths for the 156-, 1598- and 2265-bus systems are 16, 33, and 47, respectively. Test results also indicate that the path length of a singleton depends only weakly on the position of the nonzero element and does not deviate substantially from the mean. Similar results will most probably hold for other positive-sequence power networks. The average path length for zero-sequence networks is expected to be longer.

Three additional scenarios involving sparse vectors with two, five and ten randomly chosen non-zero elements are considered next. The results of interest are:

$$R5 = \frac{\text{average path length}}{\text{average singleton path length}}$$

$$R6 = \frac{\text{average total mult-adds in FF and FB}}{\text{average total mult-adds in FF and FB for singletons}}$$

The statistics of R5 and R6 for each scenario are computed from a total of 100 trials. The results are shown graphically in Figure 5.

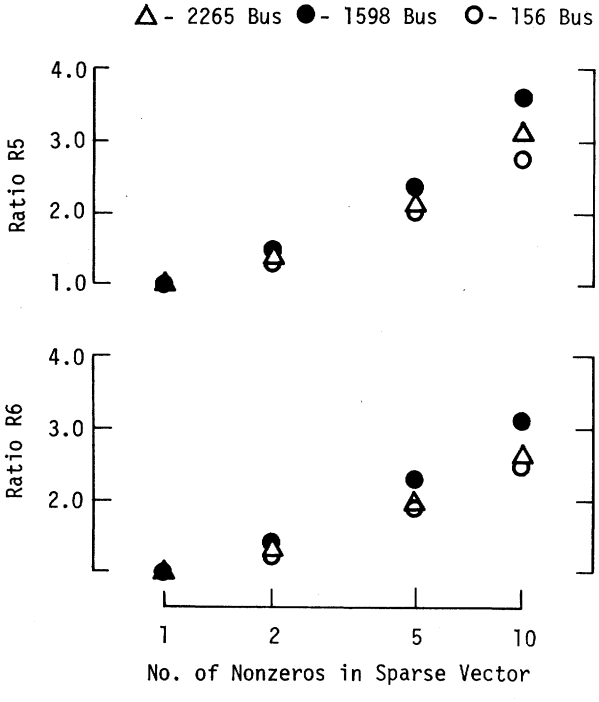

Figure 5: Ratio R5 and R6 for Sparse Vectors with 2, 5 and 10 Randomly Chosen Elements

It can be seen from Figure 5 that the ratios R5 and R6 increase with the number of nonzeros in the sparse vector. The growth rate is relatively slow; nonetheless the positive rate implies that the advantage of sparse vector methods diminishes as the number of random nonzeros becomes large. This is to be expected.

In practice, the nonzeros in the sparse vector are not randomly chosen; they are often close to each other topologically. The nonzeros in b in a compensation method, for example, are typically the end nodes of a transmission line.

An experiment is designed to test the effects of topologically related nodes on the efficiency of sparse vector methods. The relationship among nodes in this experiment can best be explained via the concept of tiers. For a given node, the first tier is defined to be the set of nodes that are directly connected to it. The second tier is the set of nodes that are directly connected to those in the first tier (and not already in the first or inner tiers), and so on. The experiment to be described involves nodes in the first three tiers surrounding node 1148 in the 2265-bus system:

Tier 1: Nodes 1042, 1619
Tier 2: Nodes 1925, 777, 767
Tier 3: Nodes 1327, 1112, 890, 2016

Four runs were performed. In each run, the nonzeros in the sparse vector are as follows:

Run 1: Node 1148
Run 2: Node 1148 and nodes in Tier 1
Run 3: Node 1148 and nodes in Tiers 1 and 2
Run 4: Node 1148 and nodes in Tiers, 1, 2 and 3

The result in terms of path length and number of multiply-adds in FF and FB is shown in Table IV. It can be seen that the growth in path length and operations count with the number of nonzeros is significantly slower than that of the previous randomly-chosen-nodes case. This supports the authors' conjecture that the performance of sparse vector methods is not affected appreciably by the addition of topologically related nodes to the sparse vector.

TABLE IV

Run	No. of Nonzeros	Path Length	Mult-Adds
1	1	38	728
2	3	39	733
3	6	42	748
4	10	45	765

OBSERVATIONS

In this section some observations are given that should be helpful for applications and extensions of FF/FB. Each observation could be expanded and many more could be listed.

Path of a Node Group

The path of the node-pair of a network branch is the path of the lower-numbered node of the pair. The branch node-pair path is important in compensation methods. More generally, the path for any group of nodes that are completely interconnected is the path of the lowest numbered node of the group. The set of nodes comprising the terminals of a group of mutually coupled branches constitute such a group in a nodal admittance matrix formulation.

Effects of Ordering

The customary sparsity-directed ordering used to enhance the sparsity of L has an effect on the efficiency of FF/FB, but it is not yet well-understood. It appears that the ordering could be modified to enhance the sparsity of L^{-1} without sacrificing the sparsity of L. Ordering could also be modified to optimize the tradeoffs between sparsity of L and L^{-1}. Some forms of sparsity-directed ordering are very adverse for FF/FB. For example, banded ordering leads to total fill-in of sparse vectors in FF/FB for all locations higher (in node number) than the leading nonzero. For power networks banded ordering is always a very poor choice anyway.

Factorized Subsystems

Every path defines a reduced subsystem whose triangular factors are contained in the original

factorization. The subsystem factors are sparse but the corresponding subvector is full. The ordering of each such subsystem is as good as the original global ordering. The number of different factorized subsystems contained within any large factorization is very large. These subsystems would seldom correspond exactly with any that would be useful for practical applications, but methods can be developed for efficiently deriving desired subsystems from them.

Partial Refactorization

If matrix A is modified, only certain columns of L and rows of U are affected. The path of a sparse vector whose nonzeros correspond to the modified rows of A defines the affected columns and rows. The factors can be modified to reflect the changes in A by repeating the factorization process only for the affected columns and rows.

Path Extension

After completing the FB for any given path containing the subgraph corresponding to the independent vector b, the back solution can be extended by any desired amount. Additional segments can be added to the FB path starting at any junction with the solved path and continuing the back solution in descending node sequence along the segments.

Parallel Processing

The properties of sparse factorization that make sparse vector methods efficient can be exploited to perform the forward and back solutions for full vectors by parallel instead of sequential operations. The Path Graph of a large network can be broken up in many different ways into subgraphs that can be processed in parallel. The practicality of this idea depends on developments in computers for parallel processing.

CONCLUSIONS

Sparse vector methods enhance matrix solutions by exploiting the sparsity of the independent vector and/or the desire to know only a subset of the unknown vector. Exploiting vector sparsity may seem trivially simple, but this does not diminish its importance. Experimental results show that the sparse vector methods can be as much as twenty times faster than conventional sparse matrix methods. The exact amount of speed-up in an application depends on the number of non-zeros in the vector and their topological relationship to the matrix.

The factorization path concept is the key to the methods' efficiency. This paper gives efficient algorithms for finding the path and provides a graphical interpretation.

Sparse vector methods are completely general and can be applied to any program that requires matrix solution with sparse vectors. Current applications of these methods include compensation techniques, fault analysis, optimal power flow and economic dispatch. Many more applications should follow when the benefits and implementation details of sparse vector methods become better known.

The properties of sparse triangular factorization on which sparse vector methods are based can be exploited for purposes other than fast solutions for sparse vectors. Only a few of the possible applications such as partial refactorization and network equivalencing are mentioned. Extensions of sparse vector methods to these purposes are well worth investigating.

REFERENCES

[1] O. Alsaç, B. Stott and W.F. Tinney, "Sparsity-Oriented Compensation Methods for Modified Network Solutions", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, pp. 1050-1060, May 1983.

[2] V. Brandwajn and W.F. Tinney, "Generalized Method of Fault Analysis", paper submitted to IEEE Transactions on Power Apparatus and Systems, Aug. 1983.

[3] D. Sun, B. Ashley, B.A. Hughes and W.F. Tinney, "Optimal Power Flow by Newton Approach", Ibid, Paper 84WM044-4, IEEE Winter Power Meeting, Dallas, TX, Feb. 1984.

[4] K.W. Edwin and G.G. Wagner, "On-Line Security Constrained Economic Dispatch Using Sensitivities and Convex Linear Programming", IFAC Symposium on Computer Applications in Large Power Systems", New Delhi, India, Aug. 1979.

ACKNOWLEDGEMENT

The authors are grateful to Mr. Walter Powell of Bonneville Power Administration for supplying the 2265-bus data.

APPENDIX A
MATRIX SOLUTION PROCEDURE

This appendix clarifies the statement "the forward solution of (3a) is performed by columns and the back solution (3b) is performed by rows". A 3-by-3 example is used. The matrix A is assumed to have been factored as in (2).

FORWARD SOLUTION

Column 1: $y_1 := b_1 / D_{11}$

$y_2 := b_2 - L_{21}\, y_1$

$y_3 := b_3 - L_{31}\, y_1$

Column 2: $y_2 := y_2 / D_{22}$

$y_3 := y_3 - L_{32}\, y_2$

Column 3: $y_3 := y_3 / D_{33}$

BACK SOLUTION

Row 3: $x_3 := y_3$

Row 2: $x_2 := y_2 - U_{23}\, x_3$

Row 1: $x_1 := y_1 - U_{12}\, x_2 - U_{13}\, x_3$

Note that the elements of L are accessed columnwise in the forward solution, and the elements of U rowwise in the back solution.

William F. Tinney received B.S. and M.S. degrees from Stanford University in 1948 and 1949. From 1950 to 1979 he worked as an electrical engineer in the Bonneville Power Administration and was head of the System Analysis Section when he retired. Most of his work has been in research and development of power system computer applications. He is presently an independent consultant.

Vladimir Brandwajn received B.S. and M.S. degrees in Electrical Engineering from Technion, Israel, in 1969 and 1973, respectively. He received his Ph.D. degree at the University of British Columbia in 1977. From 1977 to 1978, he concentrated on the modeling of synchronous machines in the Electromagnetic Transients Program (EMTP) at the Bonneville Power Administration. In 1978 he joined Ontario Hydro, where he was involved in various projects including torsional vibrations and insulation coordination in AC and HVDC systems. He joined Systems Control, Inc. in 1982.

Sherman M. Chan (S'75 - M'81) was born in Canton, China in 1953. He received his B.S. degree from Harvey Mudd College in 1976, and his S.M., Engineer, and Ph.D. degrees from the Massachusetts Institute of Technology in 1978, 1979, and 1981, respectively, all in electrical engineering. He was a Hertz Foundation Fellow from 1976 to 1981. Dr. Chan was a summer employee of the Bonneville Power Administration from 1973 to 1981, and he has been with Systems Control, Inc., since 1981. His current research interests are in power system computation, and in the effects of dispersed storage and generation systems on utility operations. Dr. Chan is a member of the IEEE System Dynamic Performance Subcommittee.

Discussion

B. Stott and **O. Alsac,** (Power Computer Application, Tempe, AZ): This paper will undoubtedly become a standard reference in the sparse matrix field. It provides important topological insight into sparse matrix factorization paths. It explains how and why solutions with sparse independent vectors involve few elements of the lower triangular factor (fast foward substitution). Likewise, it shows that few elements of the upper triangular factor need be operated on in solutions for "sparse" solution vectors (fast backward substitution). Efficient techniques for realizing the attendant computataional savings are presented.

There are many opportunities for using such techniques in power system analysis, in traditional as well as more recent and advanced applications. The possibility of fast foward substitution seems to have been recognized, though not too widely exploited, much earlier than for fast backward substitution. W.F. Tinney pointed out the latter to us as recently as 1981.

As shown in the paper, fast backward substitution is useful for solving individual buses or small clusters of buses. Another potential application is for the solution of the larger clusters formed by particular areas in multi-area systems. The accompanying table gives the results of one of our original investigations on four different networks, from 113 to 985 nodes. It shows what percentages of the total rows and elments of the upper-triangular factors have to be operated on to solve for indiviual utility areas. Examining the last entry for the 985-node network, for instance, it is seen that only 32% of the normal back substituion work is needed to solve the area consisting of 218 nodes, or 22% of the total nodes.

One of the most valuable combined uses of fast foward and fast backward substitution is to provide a very general method for calculating elements of the matrix inverse. The sparse inverse method [A] can be viewed as a special case of this general scheme. Our own sparse inversion code was implemented with no loss of efficiency in this way. The advantage is that any other inverse elements can also be calulated with similar ease and efficiency.

NUMBER OF NODES IN NETWORK	113			324			502			985		
AREA	NODES	ROWS %	ELE-MENTS %	NODES	ROWS %	ELE-MENTS %	NODES	ROWS %	ELE-MENTS %	NODES	ROWS %	ELE-MENTS %
1	1	14	17	11	15	24	9	5	8	1	3	7
2	2	13	17	12	12	21	19	8	14	1	2	5
3	2	14	15	25	15	25	20	8	14	16	4	9
4	2	14	15	29	18	26	21	9	16	18	5	9
5	18	31	33	43	20	38	33	11	17	20	5	9
6	22	27	31	49	21	38	45	14	19	21	4	11
7	27	32	37	155	66	74	49	15	17	32	6	11
8	39	45	50				65	18	27	35	7	9
9							116	27	23	44	8	12
10							125	29	37	47	9	12
11										61	10	16
12										77	15	25
13										97	13	17
14										107	14	19
15										190	23	29
16										218	24	32

SOLUTION OF INDIVIDUAL AREAS IN A NETWORK, BY FAST BACKWARD SUBSTITUTION

W.F. Tinney, V. Brandwajn and **S.M. Chan**: The test results in the discussion show that significant speedups can be achieved even with vectors that are not very sparse. All that is necessary for sparse vector exploitation is that the nonzeros be topologically clustered rather than randomly scattered.

This paper mainly emphasizes very sparse vectors. As vector sparsity decreass, the path-finding algorithm recommended in the paper may be less efficient than the "simple but inefficient" method described on the second page. Other path-finding schemes could also be developed.

As pointed out in the discussion, the factorization path concept is closely related to the sparse inverse algorithm. The path is also advantageous for the efficient computation of arbitrary subsets of the inverse of a sparse matrix. For any given application requiring selected inverse element, the tradeoff between computing them all in advance in one coordinated operation or comuputing them a few at a time as they are actually needed requires careful evaluation.

Manuscript received September 4, 1984

REFERENCE

[A] K. Takahashi, J. Fagan and M. Chen, "Formulation of a sparse bus impedance matrix and its application in short circuit studies," *Proc. PICA Conf.*, pp. 63-69, Minneapolis, MN, July 1973.

Manuscript received July 27, 1984