A NODE ORDERING ALGORITHM TO SPEED UP THE SOLUTION OF SPARSE MATRIX AND SPARSE VECTOR LINEAR EQUATION SYSTEMS

Antonio Gómez and Leopoldo G. Franquelo
Depto. de Ingen. Eléctrica, Electrónica y Automática
Universidad de Sevilla, Spain

ABSTRACT.

Recently, more attention has been devoted to sparse vector methods in order to reduce the computational burden when solving sparse systems of linear equations. These methods exploit the sparsity of the independent vector and/or the desire to know only a subset of the unknown vector. They are also applicable when refactorization of a slightly modified matrix is required.

This paper proposes a scheme to order the nodes with the purpose of reducing the number of operations when applying sparse vector methods.

INTRODUCTION.

Sparse matrix equations of the form:

$$Ax = b \qquad (1)$$

appear in many engineering fields. The standard solution procedure performs a sparsity-oriented LDU decomposition of A, followed by forward and back operations on the independent vector, b.

This paper deals with two kinds of problems:

a) Repeated solution of (1) when matrix A is slightly modified (partial matrix refactorization problem) [1].
b) Solution of (1) when either vector b has only a few non-zero elements or a small number of elements in the unknown vector x is needed (sparse vector problem) [2].

The main interest is focused on numerous power system-related topics where the mentioned problems arise, like the following ones: compensation methods, short-circuit analysis, optimal power flow, economic dispatch, contingency analysis, equivalencing, etc. Some of these aspects are strongly related to real time simulation and operation control of electrical power systems.

A recent paper by Tinney et al. [2] emphasized the interest of sparse vector methods in power system analysis, showing a dramatic reduction in operations count compared to conventional methods. In that paper, the authors noted that new node ordering algorithms had to be developed in order to enhance the sparsity of L^{-1} (U^{-1}) without sacrificing the sparsity of L (U).

In this paper, one of such algorithms is presented. Results of applying it to eight test systems are shown and compared to those provided by Tinney's scheme 2 (T-2) ordering, also known as minimum degree algorithm [3].

BASIC CONCEPTS.

For simplicity this paper will be restricted to sparse symmetric matrices or matrices that are symmetric in pattern of non zero off-diagonal terms.

Given a symmetric matrix $A=(a_{ij})$ an associated undirected graph $G=(V,E)$ can be defined, where V is the set of vertices $(v_1,v_2,\ldots,v_n)$ and E is a collection of unordered pairs of elements of V (edges), such that $(v_i, v_j)\in E$ if and only if $a_{ij}\neq 0$ and $i\neq j$. The vertices are also called nodes, points, buses, etc, and other names for edges are branches, arcs or lines. If $(v_i, v_j)\in E$, then v_i and v_j are said to be adjacent. The degree of a vertex is the number of vertices adjacent to it. If G has n vertices, then a one-to-one map, f, from V onto the set $(1,2,\ldots,n)$ is called a numbering of G.

If element a_{ii} is chosen as pivot and is eliminated, then the reduced graph associated with the reduced matrix is obtained from the original graph by deleting vertex i and incident arcs and adding arcs

between any pair of vertices which are neighbors of i but are not neighbors of each other.

Consider the symmetric factorization of a sparse matrix A into U^tU. The graph associated to the matrix $F=U^t+U$ is called the filled graph G_F.

Once A has been factorized into U^tU, interest focuses on the solution of (1), where either b is sparse or only a few elements of x are needed. Following [2] some definitions related to sparse vectors will be introduced in order to make the paper self-sufficient.

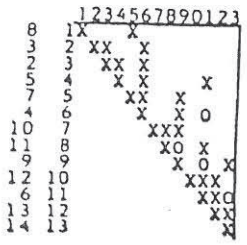
A singleton is a vector with only one nonzero element. (e.g. in location k). A path for a singleton is defined as an ordered list of rows of U which are strictly necessary for the forward solution of (1). The same path is executed in reverse order when only the k-th element of x is wanted. Such a path is easily determined from the sparse structure of U as follows [2]:

1) Let k be the first row in the path.
2) Get the number of the lowest-numbered nonzero element in row k of U. Replace k with this number and include it in the path.
3) If k is the last row, exit. Otherwise, return to step 2.

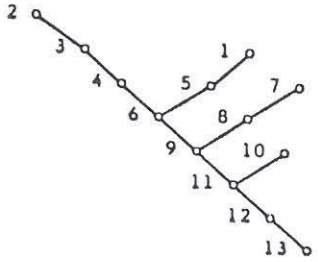
When row k of A is modified, only the rows belonging to the path graph of the singleton k-th must be taken into account during the refactorization of A. Hence, partial matrix refactorization and forward solution process are aborded in the same way. From now on, only the forward and backward processes on the independent vector will be considered.

A general sparse vector is the sum of singletons, and its path is the union of the paths of its composite singletons. If only the rows of a given path are involved in the solution of (1) the corresponding process is called the Fast Forward (FF) or the Fast Backward (FB) respectively.

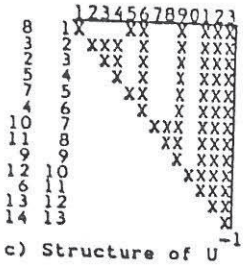
As an example consider the 14 node IEEE test system [4], whose matrix U is shown in Figure 1.a. The ordering algorithm used is T-2,



a) Structure of U

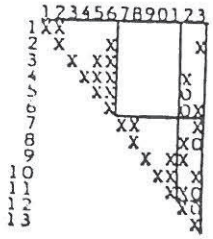


b) Path Graph

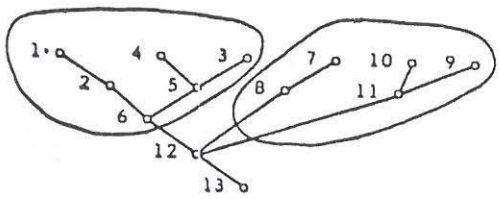


c) Structure of U^{-1}

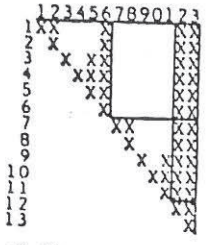
Fig. 1 Minimum degree ordering on 14-node system



a) Structure of U



b) Path Graph



c) Structure of U^{-1}

Fig. 2 Alternative ordering on 14-node system

and node 1 (the reference node in the Load Flow problem) is disregarded. A path graph for this network, which compactly describes all of its singleton paths is shown in Figure 1.b. The graph is a tree which has a direction sense from low to high node for FF and from high to low node for FB.

There is a relationship between this path graph and the sparsity structure of U^{-1}. The nodes belonging to the k-th node path coincide with the nonzero columns of the k-th row of U^{-1}. Matrix U^{-1} for the former example is given in Figure 1.c.

As may be noted, the number of nonzero elements of U^{-1} is directly related to the average singleton path length. Also the number of nonzero elements of U is directly related to the average mult-adds needed to perform the forward or back substitution process on each row. Thus, as pointed out in [2], what is required is that both U and U^{-1} remain as sparse as possible.

DESCRIPTION OF THE ALGORITHM.

The basic idea which motivated the algorithm is quite simple. Let us observe again the T-2 ordering applied to the 14-node system shown in Figure 1. In this case the average path length of the tree is 4.92. This high value is due mainly to the existence of a long branch consisting of 8 nodes (2,3,4,6,9,11,12,13). Assume now that the same system is reordered as can be seen in Figure 2. The resultant average path length is 3.61 and the longest branch is composed only of 5 nodes. This better result could be expected by simple inspection of the tree, where two sets of nodes are apparent (encircled) whose paths have only two common nodes. This shape has been achieved by using a clustering algorithm in order to obtain two weakly interconnected clusters.

The reduction in the average path length implies a more sparse U^{-1} matrix. It may be observed, however, that this is done at the cost of a slightly larger fill-in of U (one extra element appears).

Hence, motivated by the above results, the following heuristic procedure is proposed:

a) Perform a clustering of the matrix A, which yields the typical Bordered Block Diagonal Form (BBDF). The optimum number of clusters is not known "a priori". In general, some trials must be carried out, because a large number of clusters may degrade substantially the sparsity of U, overriding the gain attained with a shorter path length. Sometimes, the matrix structure itself suggests the number of clusters to be adopted.

b) Reorder the nodes belonging to each cluster following the minimum degree algorithm (T-2).

A few clustering algorithms have appeared in the literature (see for instance [5,6,7]). All of them have the drawback of seriously degrading the sparsity of U, as the size of the border grows up. Recently a clustering algorithm has been proposed [8] which has proved to avoid this problem almost completely. This is the one adopted in the next section.

DISCUSSION OF EXPERIMENTAL RESULTS.

The admittance matrix of many electrical power systems has been used to test the proposed algorithm. The results corresponding to the eight larger systems are tabulated in Tables I and II for the T-2 algorithm and the one proposed here respectively. The Appendix describes briefly the characteristics of these test systems, whose sizes range from 118 to 661 nodes.

Each table contains the number of nonzero elements of U and U^{-1}, as well as the number of mult-adds required to perform the factorization of the matrix (actually, the matrix has one node less than the number indicated in the first column, as the reference or slack node is not included).

For every possible singleton the path length and the total mult-adds in FF were computed. The mean of these quantities is

			MUL			RANDOM SINGLETONS						CLUSTERED SINGLETONS									
		-1	ADD	1 SINGL		2 SINGL		5 SINGL		10 SING		2 SINGL		5 SINGL		10 SING					
NOD	U	U	FAC	AP	AO	AP	AO	AP	AO	AP	AO	AP	AO	AP	AO	AP	AO	R1	R2	R3	R4
118	253	990	425	9.5	21	14.1	33	25.1	62	35.9	88	10.6	23	13.1	29	17.5	40	54	72	8	24
175	399	2761	710	16.9	50	22.6	66	33.2	94	47.3	130	19.0	56	19.8	57	24.2	68	56	74	12	31
265	549	3622	972	14.7	49	21.3	70	36.2	111	51.5	148	16.1	53	18.1	56	23.8	70	54	71	9	23
293	684	5435	1229	19.6	57	28.5	85	44.1	129	60.4	175	20.1	58	23.6	68	29.8	84	54	72	8	26
383	1038	8971	2503	24.5	137	32.4	181	49.7	260	66.7	324	26.3	145	28.2	149	34.6	173	57	71	13	28
448	1189	10667	2850	24.8	141	33.9	189	50.9	268	70.7	344	25.8	143	28.3	152	34.3	172	56	70	12	25
596	1710	15688	4707	27.4	188	37.1	257	58.9	392	83.5	512	28.6	194	30.9	203	36.6	220	56	68	11	23
661	1851	17638	4972	27.7	190	40.3	278	61.8	406	86.0	523	27.8	187	32.1	210	37.3	221	55	68	10	21

Table I. Results obtained with T-2 method.

			MUL			RANDOM SINGLETONS						CLUSTERED SINGLETONS										
		-1	ADD	1 SINGL		2 SINGL		5 SINGL		10 SING		2 SINGL		5 SINGL		10 SING						C
NOD	U	U	FAC	AP	AO	AP	AO	AP	AO	AP	AO	AP	AO	AP	AO	AP	AO	R1	R2	R3	R4	L
118	256	968	436	9.3	21	13.8	33	23.5	59	35.4	89	10.6	24	12.2	27	17.7	41	54	74	8	25	9
175	413	1967	788	12.3	33	18.3	54	32.0	98	46.1	138	13.8	37	16.1	43	20.8	56	54	73	8	28	4
265	552	3426	987	14.0	45	20.4	65	34.9	110	50.8	149	14.6	46	17.5	53	22.7	65	54	74	8	27	2
293	721	3963	1434	13.6	42	21.8	72	37.9	130	58.1	196	15.2	47	16.8	51	21.9	65	53	72	6	18	5
383	1048	7012	2602	19.4	97	28.1	147	45.9	237	66.8	328	20.9	104	23.1	111	29.1	134	55	73	9	24	3
448	1197	8416	2916	19.8	101	28.0	146	46.9	246	70.1	344	21.9	111	23.6	114	29.6	134	54	73	8	23	4
596	1714	13819	4755	24.2	158	36.3	245	58.7	392	81.9	510	25.3	162	28.7	179	34.9	202	55	74	9	26	3
661	1855	15314	5020	24.2	156	36.6	243	60.7	397	84.8	519	24.8	155	28.6	175	35.4	203	54	74	8	24	3

AP: Average path length.
AO: Average total mult-adds.

Table II. Results obtained with the proposed algorithm.

tabulated, as well as the ratios R1, R2, R3, and R4 defined in [2] which give a measure of the relative advantage of using FF instead of the full forward process.

Further tests were made involving sparse vectors with two, five and ten randomly chosen nonzero elements. For each case 100 trials were performed, and the average path length as well as the average total mult-adds in FF are tabulated here. From these results the ratios R5 and R6 defined in [2] may also be computed. These ratios indicate the loss of efficiency of sparse vector methods as the number of random nonzeros grows.

In practice, the nonzeros in the sparse vector are not randomly chosen. To test the effects of topologically related nodes another set of 100 trials was done. This time the two, five and ten nonzero elements of the sparse vector were chosen in a similar way as is done in diagonal band ordering. A node is randomly chosen as the first one. The adjacent nodes to those already numbered are consecutively chosen, and so on until the required number of nodes is achieved. The average results from these tests are also given.

It can be seen, as pointed out in [2], that the growth in path length and operations count is significantly smaller if the nonzero entries in the sparse vector correspond to topologically related nodes.

The last column in Table II shows the adopted number of clusters for each network.

These results suggest the following conclusions:

- The fill-in of the matrix U is slightly increased with respect to T-2 algorithm, as was expected (about 1%).
- Oppositely, the sparsity of U^{-1} is significantly improved (about 15%) which means, as was explained previously, that the average path length for one singleton is proportionally decreased (column 5).
- Besides, the average total mult-adds in FF for 1 singleton (column 6) is about 20% less than with T-2 algorithm.

- As the number of singletons grows, both algorithms tend to behave similarly, though for 10 clustered singletons the proposed algorithm still saves about 14% of mult-adds in FF compared to T-2 (column 18).
- When the singletons are randomly chosen, the advantage of the proposed algorithm over T-2 is less important.

CONCLUSIONS.

Recently, more interest has been devoted to sparse vector methods, which enhance forward and backward substitution processes by exploiting the sparsity of the independent vector and/or the need to know only a subset of the unknown vector. The same techniques can be used in the partial matrix refactorization process. Both types of problems arise frequently in many engineering fields, particularly in electric power systems analysis.

The speedup in a particular application depends not only on the number of nonzeros in the vector but on the sparsity of U and U^{-1}, which may be enhanced with a proper node ordering.

In this paper, an algorithm is proposed which clearly improves the sparsity of U^{-1} compared to the minimum degree algorithm. The improvement is translated into a reduction on the operations count besides 15%.

REFERENCES.

[1] - Chan S.M., Brandwajn V., Partial matrix refactorization. IEEE Transac. on PWRS-1, pp. 193-200, 1986.

[2] - Tinney W.F., Brandwajn V., Chan S.M., Sparse Vector Methods. IEEE Trans. on PAS-104, pp. 295-301, 1985.

[3] - Tinney W.F., Walker J.W., Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization. Procee. IEEE vol. 55 pp. 1801-1809, 1967

[4] - IEEE Committee Report, IEEE Reliability Tests Systems. IEEE Transac. on PAS-98, pp. 2047-2054, 1979.

[5] - Ogbuobiri E.C., Tinney W.F., Walker J.W., Sparsity Directed Decomposition for Gaussian Elimination on Matrices. IEEE Transac. on PAS-89, pp. 141-150, 1970

[6] - Sangiovanni-Vincentelli A. et al., An efficient Heuristic Cluster Algorithm for Tearing Large Scale Networks. IEEE Transac. on PAS-89, pp. 141-150, 1970.

[7] - Gómez A., Franquelo L.G., Multi-processor Architectures for Solving Sparse Linear Systems. Application to the Load Flow Problem. LCA'86. IFAC 1986.

[8] - Gómez A., Franquelo L.G., A New Contribution to the Cluster Problem. IEEE Transactions on CAS.

APPENDIX.

To test the proposed algorithm, several networks have been tried. The data in Tables I and II refer to the following systems:

Nodes	Comment
118	- IEEE test system.
175	- Obtained by joining 2 IEEE test systems together.
265	- Andalusian network covering 50,66,132 Kv.
293	- Obtained by joining 3 IEEE test systems together.
383	- Spanish 132,220,380 Kv. network.
448	- The former and the same levels of Portugal.
596	- Spanish 132,220,380 Kv. levels and 50,66 Kv. Andalusian network.
661	- The former and 132,220,380 Kv. levels of Portugal.

All the above mentioned systems, except the IEEE ones, are derived from the 661 node Spanish network in a peak load condition of about 20 GW, 8 GVAr. Smaller networks, like the IEEE 14, 30 and 57 [4] among others, have been succesfully tested, but the results are not reported because of space limitations.