141

Sparsity-Directed Decomposition for Gaussian Elimination on Matrices

E. C. OGBUOBIRI, MEMBER, IEEE, WILLIAM F. TINNEY, SENIOR MEMBER, IEEE,

AND JOHN W. WALKER, MEMBER, IEEE

Abstract—This is a concise critical survey of the theory and practice relating to the ordered Gaussian elimination on sparse systems. A new method of renumbering by clusters is developed, and its properties described. By establishing a correspondence between matrix patterns and directed graphs, a sequential binary partition is used to decompose the nodes of a graph into clusters. By appropriate ordering of the nodes within each cluster and by selecting clusters, one at a time, both optimal ordering and a useful form of matrix banding are achieved. Some results pertaining to the compatibility between optimal ordering for sparsity and the usual pivoting for numerical accuracy are included.

INTRODUCTION

THE ORDER in which the Gaussian elimination is performed on sparse matrices affects the total number of new nonzero elements generated in the course of the elimination and hence the total computation time [1], [2]. Power system network problems involving complex valued matrices of order 2000 or more are now becoming common. Obviously it is not practical to store and process the 8×10^6 elements of these matrices which, typically, contain only about 6×10^3 nonzero elements. The exploitation of sparsity cannot be overemphasized in these cases.

Recent investigations of the conservation of sparsity [3]-[16] have considered several practical ordering schemes. The relative performance of these schemes can be conjectured, but what remains to be determined are their inherent characteristics which will aid in making an a priori rating of their respective efficiencies.

Another area of related concern is that of matrix partitioning into blocks (or submatrices) or, analogously, the problem of network decomposition into clusters [15]-[22]. A practial solution to this problem for large random matrices (or networks) is still an open problem. Most researchers, particularly those of diakoptics [16]-[18], [21], tend to assume that such a decomposition is known in general, or at least given. Similar assumptions are encountered in the decompositions for the solution of the classical "traveling salesman problem" (TSP) [23], [24], where the method of diakoptics is also extensively used.

It is observed [21], however, that the overall computational efficiency also depends on the actual decomposition employed in any solution. Computational efficiency is generally greatest when the tearing of networks is carried out at regions of minimum coupling. This type of tearing is the subject of [20], where an optimal scheme is described. This scheme assumes, without loss of generality, that a list of preliminary optimal tearing is known and then proceeds with an iterative method using combinational logic to obtain larger clusters. This scheme might

be quite useful in the packaging of electronic modules where each module constitutes an optimal tearing. Even then the combinatorial logic will render the method impractical for large systems.

Decomposition into subsystems of tree form is another recent contribution [22]. This method does not yield minimally interconnected subsystems and has little to do with sparsity since sparsity has never been a problem in Gauss–Seidel iterative methods. It does show, as observed in [23], that the Gauss–Seidel convergence is order sensitive, if it converges at all. For iterative methods like Newton's which is widely accepted now, the tearing philosophy reported in [22] is not sparsity conserving and may increase solution times.

Judging from experience in allied problems such as the TSP [24]-[28] the subject of optimally ordered Gaussian elimination appears formidable. But to abandon further consideration of this subject is neither science nor engineering. Consequently, the effort to obtain practical optimal ordering algorithms must continue. Although computationally inefficient at face value, they do provide a standard of performance by which the efficiency of near-optimal schemes can be judged.

The objective of this paper will have been met if it helps the reader to 1) select a suitable ordering scheme without further experimentation, and 2) support his selection with sound, mathematical, physical, and economic reasoning. The plan of the paper is guided by the following considerations:

correspondence problem: matrix patterns versus graphs
a survey of existing ordering schemes
network clusters
network decomposition into clusters
a sequential binary partition (SBP) on networks
identification of clusters from SBP
ordering of clusters
optimal ordering
matrix banding schemes
computational results
conclusions.

MATRICES, GRAPHS, AND NETWORKS

In this section a basis is established for discussing indistinguishably 1) the Gaussian elimination on matrices, 2) variable elimination in a system of algebraic equations, and 3) node elimination in graphs or electrical networks.

Definition

An incidence matrix M of a matrix Y is defined to have the following properties:

$$m_{ij} = \begin{cases} 1, & \text{if and only if } y_{ij} \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

Since an element y_{ij} of a matrix Y can equivalently be regarded as the traffic (weight) on a path directed from node i to node j of a directed graph G, one can also talk of the incidence matrix of a graph G provided that the reference node is not included and y_{ii} is the total weight on paths directed away from node i.

Definition

A square matrix Y or a directed graph G is "incidence symmetric" if the associated incidence matrix M is symmetric.

Classes of Matrices

For the purpose of the ensuing discussions, two classes of matrices are recognized, namely:

incidence symmetric
incidence asymmetric.

A special class of incidence-symmetric matrices are the symmetric matrices Y for which $y_{ij} = y_{ji}$ (see Fig. 1).

Node Elimination and Valency

In a connected graph, nodes communicate between one another through the directed paths. Thus if directed paths exist between nodes adjacent to a node k such that if k is removed, flow in the graph is not interrupted, k can be eliminated without affecting the remaining graph. If no equivalent direct paths existed prior to elimination, then new ones have to be created. To illustrate this point, consider the four-node graph of Fig. 2(a) in which node 1 is to be eliminated. The neighbors of node 1 are 2 and 4. There is traffic from 4 to 2 through node 1. So if node 1 is eliminated, a direct path from 4 to 2 must absorb this traffic. Since there is no direct path from 4 to 2, a new one must be created as shown in Fig. 2(b). Since the path 4–1–2 is a one-way path, the newly created path must also be a one-way path. The elimination of node 2 in place of node 1 would have the result shown in Fig. 2(c), while the elimination of 3 in place of 1 or 2 would have the result shown in Fig. 2(d).

Fig. 3 shows more examples to illustrate the effect of eliminating node 1 in a four-node graph.

Definition

The valency of a node (or a group of nodes) in a graph G is the number of new paths added among the remaining set of nodes as a result of the elimination of the node (or the group of nodes).

Definition

The valency of an ordering on a given subset of nodes of a graph G is the total number of new paths generated in the process of performing the node elimination in the order specified.

The reader may verify that the valency of the set of all the nodes in a graph is zero, while the valency of an ordering on the set of all the nodes of the graph may not necessarily be zero. This distinction should serve to elucidate the meaning of the last two definitions.

Lemma 1

There is a one-to-one correspondence between the valency of a node of a given graph G and the number of new nonzero elements introduced into the associated incidence matrix by the Gaussian elimination of that variable, which is associated with the node, from the system of linear algebraic equations associated with the graph G.
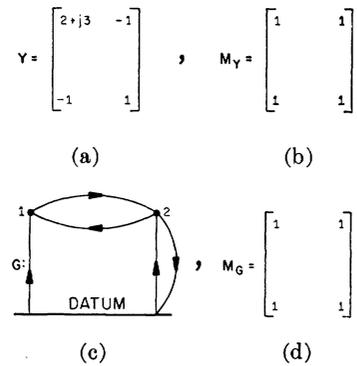
Fig. 1.   Examples of incidence matrices.
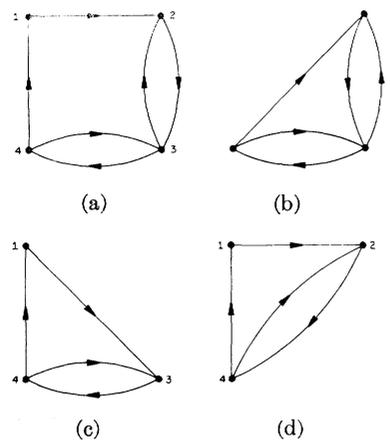
Fig. 2.   Directed graphs.
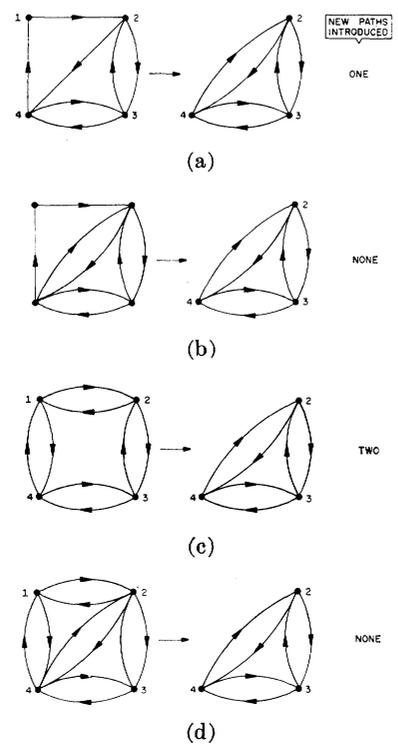
Fig. 3.   Effect of eliminating of node 1.

Proof: To the matrix Y of the linear system of equations $Yv = c$, there corresponds a directed graph G_Y such that the coefficient y_{ij} of the variable v_j in the ith equation of the system is the weight or traffic on the path directed from node i to node j of the graph G_Y. Given the graph G_Y, the matrix Y can also be constructed. There is associated with the matrix Y or the graph G_Y, an incidence matrix M. It is thus sufficient to show that the elimination of node j from the graph G_Y has the same effect on M as the elimination of variable v_j from the system of equations $Yv = c$.

When a node j is eliminated, all paths incident on it are removed from the graph. The traffic going through node j must be redistributed as follows. For each pair of nodes (k,i) which are neighbors of node j, the path $k \rightarrow i$ is created, widened, or unaltered according to whether there existed a directed path $k \rightarrow j \rightarrow i$ and the path $k \rightarrow i$ did not exist, the path $k \rightarrow i$ existed, or there did not exist the path $k \rightarrow j \rightarrow i$. A similar argument holds for the path $i \rightarrow k$. Widened and unaltered paths do not affect the matrix M. But created paths add new nonzero terms in corresponding locations of M.

In a Gaussian elimination, the variable v_j is eliminated by solving for it in equation j in terms of all other variables in equation j, and then substituting for v_j in the lth equation $(l \neq j)$. Now suppose $m_{ki} = 0$ prior to the elimination of v_j. As in the previous paragraph, we seek a condition under which $m_{ki} \neq 0$ after the elimination of v_j. Equation j must contain a term in v_j and v_k; equation k must contain a term in v_j but not in v_i. In terms of neighborhood, we say that equations j and i are neighbors and also that equations j and k are neighbors. This implies that k and i must be neighbors of j, and that there is a coupling from k to i through j in the absence of a direct coupling from k to i. Similar reasoning holds if we consider m_{ik} in place of m_{ki}. Thus the elements of the vector v play the role of nodes of a graph while the coefficients y_{ij} of the elements of v play the role of directed paths of the graph.

Remarks on Lemma 1: This lemma permits us to talk interchangeably of graphs and systems of equations in dealing with ordered Gaussian elimination. If the associated incidence matrix of a graph is symmetric, the graph may be topologically considered to be a set of nodes connected by arcs instead of directed paths. While the use of arcs simplifies an actual computation, it does not alter the basic logic. Most power network problems lend themselves to "arc" analysis. In this paper, incidence-symmetric systems will be assumed. This assumption is also implicit in most ordering schemes in the current literature.

SURVEY OF EXISTING ORDERING SCHEMES

There are two primary objectives for an ordered Gaussian elimination. The first and oldest seeks to control numerical accuracy through a pivoting scheme; the second aims at conservation of matrix sparsity. In general, these objectives are not compatible. However, the physical nature of large-scale electrical networks and the numerical accuracy of modern computers tends to eliminate the need for pivoting and to enhance the need for the exploitation of sparsity. Therefore, this survey is restricted to sparsity-conserving ordering schemes. Although there is appreciable interest and activity in this area, there are few published schemes which represent a definite contribution to the state of the art.

Matrix-Banding Schemes

The objective of the banding schemes is to restrict the elements of a matrix to a narrow band about the major diagonal [1], [5] or, alternatively, about the minor (upper-right-to-lower-left)
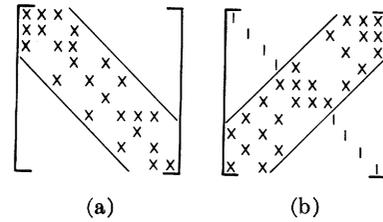
Fig. 4. Forms of matrix banding. (a) Major diagonal bandings. (b) Major diagonal bandings.
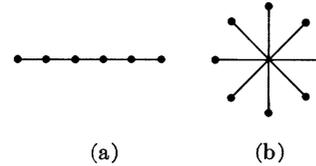
Fig. 5. Basic networks for comparing banding schemes. (a) Simple chain. (b) Simple star.

diagonal [13]. The purpose of these schemes is to isolate the nonzero elements of a matrix into a relatively small region which can be stored in computer memory in place of the entire matrix. Fig. 4 illustrates the general pattern of banded matrices.

Definition

Given a set S of nodes of a connected graph G and a subset S_1 of nodes, the minimal adjacent cut set (MACS) of S_1 are nodes from S not in S_1 but which are neighbors of nodes in S_1.

An algorithm for major-diagonal banding is stated as follows [1]. Starting with an arbitrary node as number 1, the MACS of node 1 are assigned the next sequence of numbers; the MACS of the numbered nodes are subsequently assigned the next sequence of numbers, and the process is continued until all the nodes of the network are enumerated. This algorithm is weak because the choice of the initial node is critical. There is no reason why the initial node cannot be chosen with discretion as in the next algorithm.

An algorithm for minor-diagonal banding is as follows [12]:

1) start with a bus having only one line and list it as node 1
2) list the MACS of node 1 as the last-numbered nodes
3) assign the next lower-valued numbers to the MACS of the numbered nodes
4) assign the next high-valued numbers to the MACS of the numbered nodes
5) return to step 3).

This algorithm has a weak initialization because it presupposes that there always exists a node with only one incident branch. This weakness can be removed by selecting the node with the least number of incident branches as the first node. Assuming that the amended initialization is accepted by the two banding algorithms described above, then these two algorithms can be best compared by giving examples of networks for which they tend to give a characteristically poor result. Consider the simple chain and the simple star networks of Fig. 5.

The major-diagonal banding performs well for the simple chain but not for the simple star, while the minor-diagonal banding scheme gives a good result for the star but not for the simple chain. Thus the minor-diagonal scheme is more suited to radial networks while the major diagonal scheme is more suited to cascade networks.

One major drawback of the banding schemes is that the banded matrix may still be sparse whereas it tends to fill up solidly as

the elimination process is carried out. Perhaps this obvious disadvantage can be reduced by optimal banding which is the subject of [6]. Optimal banding does not however, appear computationally attractive.

Pseudooptimal Ordering Schemes

Apart from the matrix-banding schemes previously described, there are three other schemes [7] which aim at optimum conservation of matrix sparsity during Gaussian elimination. In terms of sparsity conservation, these schemes are generally more efficient than the banding schemes. The usual convention is to refer to these as Scheme 1, Scheme 2, and Scheme 3 [7], [10].

Scheme 1: Number the rows of a matrix in ascending order of the number of off-diagonal nonzero terms; if more than one row has the same number of off-diagonal nonzero terms, select these rows in any order.

Scheme 2: At each stage of an elimination, select that row which has the fewest number of off-diagonal elements; if more than one row has this minimum number, pick any one of them.

Scheme 3: At each stage of an elimination, select that node which has the smallest valency; if more than one node has this property, pick any one of them.

There is, for each of these three schemes, a number of cases for which optimal ordering is achieved. However, it is also possible from the specifications of each scheme to construct examples for which the scheme gives a poor ordering. Such examples serve to bring out the key characteristics of each scheme. All indications of relative efficiency can be deduced from these characteristics except for the actual computation times.

Definition

A network is recognized by an ordering scheme if for any initial numbering of the nodes of the network the scheme produces an optimal ordering on the nodes.

An ordering scheme can therefore be considered as an input–output logic machine, and, according to the last definition, an optimal-ordering scheme has the schematic representation shown in Fig. 6. Fig. 7 shows some elementary networks and the schemes which recognize them. Either from Fig. 7 or from the definition of these schemes, one can construct primitive network which are not recognized by the schemes. Fig. 8 shows these networks.

A primitive network not recognized by Scheme 1 has 5 nodes and 4 branches. A primitive network not recognized by Scheme 2 has 6 nodes and 10 branches or 7 nodes and 8 branches. A primitive network not recognized by Scheme 3 has 9 nodes and 10 branches.

Judging from the number of nodes and branches in the primitive networks, a hierarchy of the recognizing power of the three schemes is apparent. Thus, in general, Scheme 1 will be less optimal (in the sense of sparsity conservation only) than Scheme 3. Exceptions to this rule come from a special class of radial networks which are not typical of power networks. Results reported for typical networks [10] confirm this ranking of the schemes.

A theoretical comparison of the three schemes is complete if some evaluation of the computation times can be made. In general, this will depend on the efficiency of coding and the amount of storage space available. Scheme 1 should be much faster than Scheme 2, and Scheme 2 should be much faster than Scheme 3.

The primitive networks which Schemes 2 and 3 do not recognize are so similar that the sparsity conservation power of Scheme 2 will be expected to be quite close to that of Scheme 3. Thus for
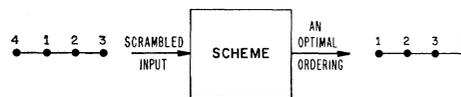
Fig. 6. Ordering scheme considered as input–output logic machine.
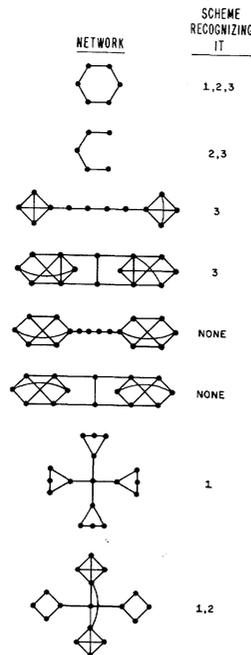
Fig. 7. Simple networks recognized by schemes 1, 2, 3.
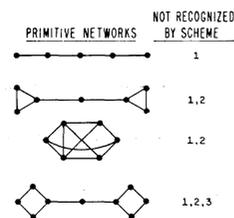
Fig. 8. Primitive networks not recognized by schemes 1, 2, 3.

typical power networks, Scheme 2 will be expected to be superior to Scheme 3 if efficiency is measured in terms of the product of solution time and the valency of the ordering produced. Typical results [10] justify this assertion.

NETWORK CLUSTERS

There are a number of situations where it is advantageous to manipulate subnetworks, one at a time, in place of the entire network.

1) There is a data-processing bottleneck, and overlay techniques might be necessary.

2) Relatively small changes are made in parts of a given network, and the effects of these changes are local to those parts where they are made; for example, the addition or removal of a transmission line in a power system network.

3) Some operations are to be performed on a large network, and these operations are known to be more efficient if applied to subnetworks in succession; for instance, all optimal ordering schemes including the algorithms for the solution of the (TSP).

4) Operations on large sparse matrices involve the use of operations on small submatrices which are both simple and known a priori; for instance, the formulation of the economic-dispatch problem [29] and the static-state estimation problem [30] in such a way that power-flow solution matrices can be employed.

The definition of a cluster is an open matter and will generally depend on a particular application. Informally, a cluster can be defined as a subnetwork whose nodes are more closely related with one another than with any other node outside the subnetwork. Sharper definitions are evoked as the need arises.

NETWORK DECOMPOSITION INTO CLUSTERS

Decomposition Criteria

There are four distinct approaches to network partitioning. These are

1) the method of natural division [5], [15]
2) the method of artificial division [5], [15]
3) minimum flow capacity [22]
4) minimum physical intertie [20].

The method of natural division tries to group the nodes of a given network using functional attributes. For instance, sources may be placed in one group, sinks in another, or perhaps nodes at which certain constraints are imposed such as fixed voltages or power-factor angles. The method of artificial division, on the other hand, may group the nodes in such a way that a value arrived at on the basis of certain considerations is not exceeded. For instance, the nodes may be grouped so that each group contains no more nodes than can be handled in a computer memory using overlay techniques. Because of the admissible variety of definitions for artificial and natural divisions, these methods are too imprecise to evaluate.

The method of minimum flow capacity tries to partition the network in such a way that nodes connected by heavy traffic (high value of flow) are grouped together. The boundaries for this kind of partition will vary considerably with the variation in the birth and death of the sources and sinks. Furthermore, it is generally not compatible with sparsity conservation. This observation does not rule out the possibility of the remarkable effectiveness of this method in certain applications. An algorithm for this method [22] is fast and has no size limitation.

The method of minimum physical intertie tries to reduce the network into minimally intertied subnetworks. Practical algorithms are lacking, but this criterion is sparsity conserving and, therefore, is most relevant to the goals of this paper.

Network Forms in Terms of Clusters

For convenience, four types of basic networks are postulated as shown in Fig. 9. A general network will be a mixed type.

Cluster Identification

The human eye can easily identify clusters from a good layout of the network. In comparison, the effort expended in identifying clusters by digital analyses is sometimes disproportionately high. Thus whenever human judgement can be exercised it should be fully exploited. Perhaps a hybrid computer consisting of a pattern recognizer and a digital computer may be the final answer to problems of efficient cluster identification.

Another possibility for cluster identification is by network solutions using Kirchoff's laws for current and voltage distributions. These methods should be faster than search techniques
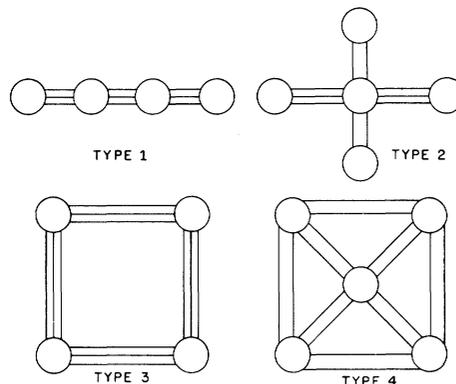
Fig. 9. Basic network forms in terms of clusters.

at comparable efficiency. The main drawback to analog schemes using network solutions is in the choice of reference nodes which appears critical.

An optimal digital scheme for cluster identification is given in [20]. This scheme is not computationally efficient since it involves enumeration of all potential grouping pairs. Thus one is bound to inspect groups in the order of a factorial magnitude.

In the following, a new method is described which not only identifies clusters but also provides enough information for optimal ordering of a Gaussian elimination.

A SEQUENTIAL BINARY PARTITION (SBP) ON NETWORKS

SBP Tableau

In order to facilitate the presentation of the identification method for clusters in a network $\mathfrak{N}$, a special tableau which will be employed in the construction of SBP will now be described.

Suppose we have a connected network $\mathfrak{N}$ whose nodes are numbered $1, 2, \cdots, N$. Without loss of generality, we pick node number 1 as the seed for SBP. The valency of eliminating node 1 is written as a superscript of the number 1. Suppose this valency is v_1, then we enter node 1 together with its valency on the first line (line 0) of the tableau (see Fig. 10), and we also make the same entry on line 1, under the heading $\mathfrak{N}_1$. ($\mathfrak{N}_1$ consists of the list of nodes under the heading $\mathfrak{N}_1$.) The adjacent neighbors of $\mathfrak{N}_1$ are listed on the second line (line 1) under the heading MACS. Suppose these neighbors are nodes 5, 7, and 12. Clearly 5, 7, and 12 comprise the MACS of $\mathfrak{N}_1$. The product of the valency of $\mathfrak{N}_1$ and the cardinality of the MACS of $\mathfrak{N}_1$ is recorded on line 1, under the heading Index. The entry $3v_1$ is thus made.

The valency of eliminating $\mathfrak{N}_1$ together with one of the nodes in the MACS of $\mathfrak{N}_1$ is written as a superscript of that particular node in MACS. Thus we have v_5, v_7, and v_{12} as the valency of eliminating the pairs of nodes $\{1,5\}$, $\{1,7\}$, and $\{1,12\}$, respectively. The node in MACS with the smallest superscript is then entered on line 2 under the heading $\mathfrak{N}_1$, in this case v_7 is the smallest of v_5, v_7, and v_{12}. The smallest valency need not be unique; the choice in that case is arbitrary. The construction continues in this manner until all the nodes of the network are listed in $\mathfrak{N}_1$ (see Fig. 11).

Properties of the SBP Tableau

The SBP tableau so constructed has a number of interesting properties.

1) The right edge of the array under the heading MACS is a wavy pattern with at least one peak. The example of Fig. 10 has only one peak.

INDEX	N_l	MACS			
		1^{V_l}			LINE 0
3_{V_l}	1^{V_l}	5^{V5}	7^{V7}	$12^{V:2}$	LINE 1
	7^{V7}				LINE 2

Fig. 10.   Construction of SBP tableau.

INDEX	N_l	MACS				
		1^3				
9	1^3	2^4	3^4	7^4		
16	2^4	3^4	7^4	4^4	8^4	
16	3^4	7^3	4^2	8^3	5^3	
8	4^2	7^1	8^1	5^1	6^1	
3	7^1	8^0	5^0	6^1		
0	8^0	5^0	6^0			
0	5^0	6^0				
0	6^0					
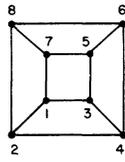
Fig. 11.   Network and its SBP tableau.

INDEX	N_l	MACS					
		1					← SEED
2	1	2	9				
9	9	2	10	14			
24	2	10	14	5	7		
72	7	10	14	5	4	6	8
50	5	10	14	4	6	8	
40	8	10	14	4	6	3	
20	6	10	14	4	3		
9	3	10	14	4			
2	4	10	14				
→12	14	10	11	13	15		
8	15	10	11	13	12		
3	12	10	11	13			
0	13	10	11				
0	11	10					
0	10						

Fig. 12.   SBP tableau and cluster identification.

INDEX	N_l	MACS					
		3					← SEED
9	3	4	6	3			
8	8	4	6	5	7		
3	6	4	5	7			
3	7	4	5	2			
0	4	5	2				
0	5	2					
0	2	1					
0	1	9					
→2	9	10	14				
12	14	10	11	13	15		
8	15	10	11	13	12		
3	12	10	11	13			
0	13	10					
0	11	10					
0	10						

Fig. 13.   SBP tableau for network of Fig. 12
using different seed.

Definition

An initial segment of an optimal ordering is a group of nodes which have the property that their optimally ordered elimination followed by an optimally ordered elimination of the remaining nodes of the network constitutes an optimally ordered elimination of all the nodes of the network.

6) The interior nodes of a cluster constitute an initial segment of an optimal ordering for Gaussian elimination.

Definition

A cluster is recursively defined as follows:

1) it conforms with property 4),
2) it conforms with property 6).

A group of nodes do not belong to a cluster unless by virtue of 1) and 2).

From the tableaux of Figs. 12 and 13, one deduces that the set of nodes {14,15,12,13,11,10} belong to a cluster and also that {9,14,15,12,13,11,10} also belong to a cluster. Clearly, the first cluster is a proper subset of the second, and these two intersecting clusters were produced by two different seeds. The exclusion of node {9} from the first cluster is a "seed effect" and could for certain networks result in partial clusters.

Correction for Seed Effect

To correct for the seed effect, we start with the partial cluster at the end of the tableau. Each node in the MACS of the partial cluster is considered as a candidate for inclusion into the partial cluster. A new partial cluster is created if there exists a node in the MACS which will result in a decrease of the preentry index of the cluster as a result of its inclusion into the partial cluster. This procedure is iterated until there is no further decrease in the preentry index. It ensures that the cluster picked from the SBP tableau has all the desired properties of a cluster as defined.

To illustrate the correction procedure, consider the first partial cluster {14,15,12,13,11,10} given by the first tableau. The MACS of this partial cluster is {9}, the set consisting of node 9. If node 9 is included into the partial cluster, we get {9,14,15,12,13,11,10}. The valency of the complement of {9,14,15,12,13,11,10} is 0, and hence its index is 0. The entry index of the original cluster is 12, and the preentry index is 2. Since 0 < 2, node 9 is made part of the cluster with an entry index 2. Repetition of the procedure yields an index 0 which is not less than the preentry index 0. Thus the correction for seed-effect is complete, and the final cluster is {9,13,15,12,13,11,10} which is the same as the cluster produced when the seed is node 3.

IDENTIFICATION OF CLUSTERS FROM SBP

As has already been outlined, the identification of the clusters defined in the last section consists of 1) identifying the partial cluster from the SBP, and 2) correcting the partial cluster for any seed effect to obtain a cluster. The cluster is then eliminated from the network. The construction of the SBP tableau, the identification of a cluster, and the elimination of the cluster are repeated on the remaining network until the residual network is a cluster by itself, i.e., the SBP curve has only one peak.

This method is very good for tearing a network into two or more minimally interconnected subnetworks. Networks with large well-defined clusters are shown in Fig. 14; all the clusters can be isolated in one pass.

Most electric power networks consists of large clusters containing smaller subclusters. Since a subcluster is a cluster, SBP

2) Each peak is formed by a monotonic rise and fall which, respectively, indicates the entry into and exit from a strongly interconnected group of nodes. Figs. 12 and 13 illustrate this and the previous property. The superscript of valencies has been omitted for simplicity.

If attention is focused on the rise and fall of the last peak, one observes the following.

3) The rise and fall of the last peak corresponds to the last rise and fall of the sequence of indices marked by an arrow.

4) The group of nodes in $\mathfrak{N}_1$ which are subtended by the last rise and fall in the sequence of indices form a cluster.

5) The interface nodes of a cluster are given by the MACS preceding the last rise in the sequence of indices; the interior nodes of the cluster are those nodes which are not interface nodes.

Fig. 14. Network of three clusters.

will tend to isolate the small subclusters for a given seed. This tendency renders SBP inefficient for isolating clusters that consist of many small subclusters. However, approximate methods using SBP appear both promising and practical for tearing large networks into subnetworks that have small interconnections. For most networks, the approximate scheme produces the same results as the exact scheme and is in addition much faster. For the 46-node network of Fig. 14, the approximate method will produce the same result as the exact method.

The approximate scheme consists of constructing the SBP, but the index used is the number of nodes in MACS instead of valency times the number of nodes in MACS. The elimination of the need to find valencies accounts for the great improvement in speed. There is also no correction for seed effect.

Ordering of Clusters

An ordering of clusters can be achieved by any scheme the user wishes to employ. In terms of overall efficiency, Scheme 2 has already been recommended, but if the clusters are in the neighborhood of 100 nodes or less, Scheme 3 may be employed for better ordering.

Optimal Ordering

An ordering is optimal if its valency is less than or equal to that of all other orderings. Since the optimal ordering problem can be formulated as a TSP (if we define the distance d_{ij} of going from node i to node j as the valency of eliminating node i followed by the elimination of node j), it can be solved by such techniques as dynamic programming [31] and integer programming (method of branch and bound [26] in particular). But because the distance matrix of the ordering problem varies with partial orderings, the existing algorithms for TSP are not directly applicable. Furthermore, the existing algorithms are extremely slow and do not appear practical for anything above 50 nodes.

SBP can be exploited for optimal ordering. Consider the SBP for the primitive networks, as shown in Fig. 15.

Theorem 1

The following results are true.

1) There exists no network having less than 5 nodes or having less than 4 branches which is not recognized by Scheme 1.

2) There exists no network having less than 6 nodes or less than 8 branches which is not recognized by Scheme 2.

3) There exists no network having less than 9 nodes or less than 10 branches which is not recognized by Scheme 3.

Proof: (Left to the reader.) Hint: pretend that you can reduce the existing primitive networks by modifying their configurations and then arrive at a contradiction.

Consider the graph of Index versus the line or row number of the SBP tableau. Let this be called an index graph. Since the primitive networks exhibit the essential characteristics of the ordering schemes (this is how they were constructed in the first
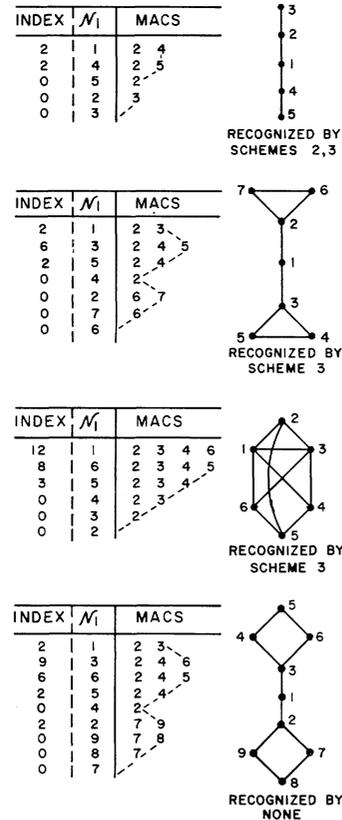
Fig. 15. Primitive networks and their SBP tableau.
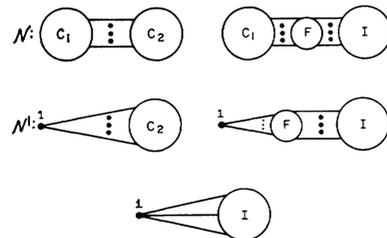
Fig. 16. Decompositions for $\mathfrak{N}$ and $\mathfrak{N}'$.

place) the following conclusion follows from the SBP tableau of the basic networks: If the index graph has no more than one peak and the nodes are either less than 9 or have less than 10 interconnecting branches, then Scheme 3 will order it optimally. When there are more than 9 nodes and more than 10 branches then Scheme 3 must be employed on subgroups of less than 9 nodes which form initial segments. Variants of this principle also exist.

We have so far shown 1) that the interior nodes of clusters constitute an initial segment of an optimal ordering and 2) that networks consisting of exactly one cluster (index graph has one peak) are recognizable by Scheme 3. It remains to be shown how the interior nodes of a cluster which is connected to one or more clusters can be ordered.

Consider a network $\mathfrak{N}$ consisting of connected components C_1 and C_2 which are interconnected as shown in Fig. 16. Suppose C_2 is identified as a cluster from a SBP tableau. Then let F and I be, respectively, the interface and interior nodes of C_3. (Note that the seed of the tableau producing C_2 must be in C_1, and that by construction, C_1 is connected.)

It is easy to verify that $\mathfrak{N}'$ [Fig. 16(c),(d)] is a one-cluster network and hence can be reordered by Scheme 3. The elimination of

$\{1\}$ in $\mathfrak{N}'$ or C_1 in $\mathfrak{N}$ makes the subnetwork, whose nodes are F, a complete network. From this fact and the cascade form of $\mathfrak{N}'$, an optimal elimination of I is consistent with the optimal ordering of $\{1\} \cup F \cup I$. Thus the strategy is to order the nodes of $\mathfrak{N}'$ optimally in the following manner. 1) Select a node by Scheme 3; in case of conflict give priority to $\{1\}$, F, I in that order; if an element of I is selected, let it be the one closest to F and call it I_1. 2) Select another element; if an element of F is selected, call it F_1; note that selection of F_1 need not occur unless $\{1\}$ has been selected in 1)); if an element of I is selected, call it I_2. The final sequence could thus be $(1,F_1,I_1,I_2,F_2,F_3,I_3,I_4)$. The optimal ordering desired is $(I_1,I_2,I_3,I_4,F_3,F_2,F_1,1)$. The original elements of I are eliminated in the order which they appear in this ordering.

Theorem 2

If for a given network, a particular node is restricted to be last in an ordering, an optimal ordering subject to this restriction is also optimal without the restriction.

Proof: Consider that a SBP tableau specifies the decomposition of a network $\mathfrak{N}$ shown in Fig. 16(b) using the restricted node as the seed. (Correction for seed effect must be performed.)

If the restricted node is in $C_1 \cup F$, eliminate I and construct another tableau using the same node as the seed. If the node is in I, eliminate the interior nodes of C_1 (which is a cluster of clusters) and construct another tableau using the restricted node as the seed. If the residual network constitutes one cluster, then we would have the situation of Fig. 16(e). The elimination of I leaves node 1 (the restricted node) as the last node in the ordering.

A direct consequence of Theorem 2 is the following generalization.

Theorem 3

In any graph, the nodes of any complete subgraph can be last eliminated in an optimally ordered elimination.

Matrix Pivoting

There are situations where a particular node of a given network is required to be numbered last. According to Theorem 2 and 3, such a specification is fully compatible with optimal ordering. This compatibility can be exploited for partial pivoting on elements of a matrix. At each stage of an elimination one can require that a certain node be eliminated last; if this node is the one whose diagonal entry is zero, one can see that by postponing its elimination the problem of dividing by zero is avoided. Theorem 3 must be used to determine how many nodes can be eliminated last.

Example: To illustrate the preceding results, consider the network of Fig. 17.

The associated index graph has one peak, and consequently, Scheme 3 recognizes it in the absence of any constraint. Now suppose we constrain node 1 to be the last-numbered node. Using Scheme 3, we get the ordering $(1,7,3,5,4,2,6)$. In the tradition of $\mathfrak{N}'$, we would get the ordering $(3,5,4,7,2,6,1)$ representing $(I,F,1)$.

Optimal-Ordering Algorithm

1) Construct the SBP tableau using any node as the seed.
2) Correct for seed effect.
3) Identify a cluster and construct the associated $\mathfrak{N}'$ network.
4) Eliminate I from an optimal ordering $(I,F,1)$ on $\mathfrak{N}'$.
5) If the remaining nodes consist of $\{1\} \cup F$, then add the ordering of $\{1\} \cup F$; otherwise return to 1).
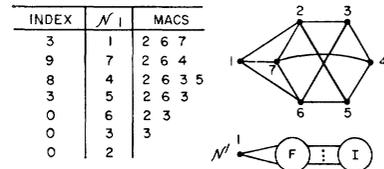
Fig. 17.   Example of one-cluster network $\mathfrak{N}'$.
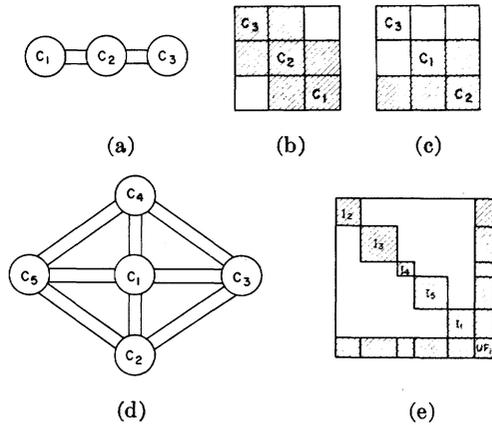
Fig. 18.   Cluster elimination and matrix banding. (a) Type 1 network. (b) Seed is always in C_1 (produces major diagonal banding). (c) Seed is first in C_1, second in C_2, third in C_2. (d) Type 4 network. (e) To obtain I_2, I_3, I_4, I_5, I_1, UF_1 seeds are, respectively, in C_4, F_2, F_3, F_4, F_5, F_1.

This algorithm is optimal and is independent of any existing ordering scheme. Thus the optimal ordering for each set I of interior nodes may be accomplished by the best method available. With suitable adjustments, Scheme 3 is found to give optimal ordering on I.

Matrix Banding

The location of the clusters identified from SBP tableaux can be controlled by sequencing the choice of seeds in a certain way.

Let C denote a cluster, and let F and I denote its interface and interior nodes, respectively. Fig. 18 shows the effect of sequencing seeds for a type 1 network. For a type 4 network the effect of seed selection is more evident. The type of banding which postpones the numbering of interface nodes looks like an anchor and may be described as an anchor banding. It is very desirable for radial-type systems.

Computational Results

Fig. 19 demonstrates the application of the optimal-numbering routine. Its ability to identify clusters for anchor banding of the associated matrices is evident. The networks shown are not recognized by Schemes 1, 2, or 3 alone. Computation times are for CDC 6400 computer.

A 128-node network taken from a real system took 19.070 seconds to renumber.

Conclusion

What is covered in this paper can be correctly described as the theory of optimally ordered Gaussian elimination on sparse systems employing partial pivoting. A critical survey of ordering conventions, philosophies, and practices indicates that Scheme 2 is the most efficient in terms of computation time and optimality. However, Scheme 2 does not identify clusters directly. Since

clusters are shown to be compatible with optimal ordering, ability to renumber cluster by cluster would be a desirable property of a renumbering Scheme.

A method of renumbering by clusters is developed which employs Scheme 3. It is remarkably efficient in identifying well-defined clusters. It also ensures optimal ordering. Partial pivoting is shown to be compatible with optimal ordering, and this may be very useful in situations where the diagonal of matrices tends to vanish in the course of an elimination.

Although the optimal-ordering algorithm reported in this paper solves 100-node problems efficiently (less than 20 seconds in CDC 6400), it is not fast enough to be practical for large systems. But since the idea of tearing networks into minimally interconnected subnetworks is still a useful one, a pseudoclustering Scheme using Scheme 2 is described. This scheme is fast and is applicable in a practical sense to a large system.

While there is clear indication that optimal ordering is not yet practical beyond 200 nodes, the problem of finding fast optimal-ordering schemes is still pressing. Even the inefficient existing methods can be used in small-sized systems to test the relative performance of pseudooptimal schemes.
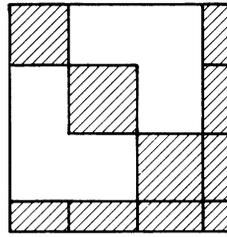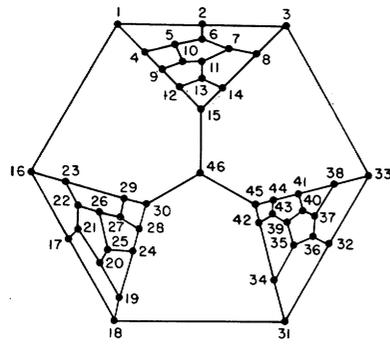
2, 8, 13, 7, 10, 4, 5, 9, 6, 11, 14, 12; 17, 23, 20, 24, 19, 25, 27, 22, 21, 26, 28, 29; 32, 34, 37, 41, 38, 40, 43, 35, 36, 39, 44, 42; 31, 45, 30, 18, 33, 15, 3, 1, 46, 16.
VALENCY 84
TIME ___.948___ S

(a)
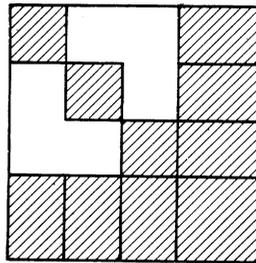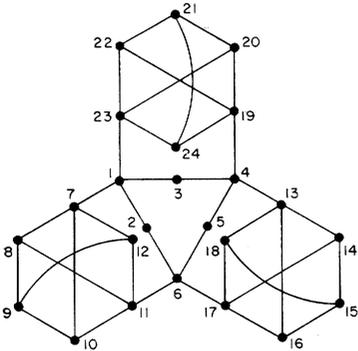
12, 10, 9, 8; 18, 16, 15, 14; 24, 22, 21, 20; 5, 17, 13, 2, 7, 11, 6, 3, 4, 19, 23, 1.
VALENCY 15
TIME ___.302___ S

(b)
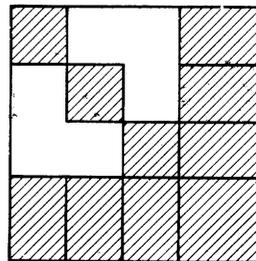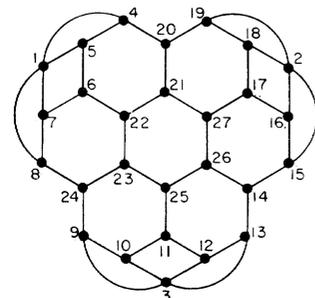
12, 10, 3; 18, 16, 2; 7, 5, 1; 4, 8, 6, 23, 21, 19, 17, 15, 26, 22, 20, 27, 24, 14, 25, 11, 13, 9.
VALENCY 35
TIME ___.444___ S

(c)

Fig. 19. Computational results (semicolon demarcates interior nodes).

REFERENCES

[1] N. Sato and W. F. Tinney, "Techniques for exploiting the sparsity of the network admittance matrix," *IEEE Trans. Power Apparatus and Systems*, vol. 82, pp. 944–950, December 1963.
[2] J. Carpentier, "Ordered eliminations," *Proc. Power System Computation Conf.* (London, 1963).
[3] D. Bree, Jr., "Some remarks on the application of graph theory to the solution of sparse systems of linear equations," Bonneville Power Admin., Portland, Ore., Internal Rept., September 1, 1964.
[4] D. W. Martin, "Matrices," *Internatl. Sci. and Tech.*, pp. 58–104, October 1964.
[5] A. Z. Gamm, L. A. Krumm, and I. A. Sher, "General principles of calculating the steady-state operation of an electric system with division into subsystems," *Izv. Akad. Nauk SSSR, Energ. i Transp.*, pp. 7–15, no. 6, 1965.
[6] G. G. Alway and D. W. Martin, "An algorithm for reducing the bandwidth of a matrix of symmetrical configuration," *Computer J.*, vol. 8, pp. 264–272, October 1965.
[7] W. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proc. IEEE*, vol. 55, pp. 1801–1809, November 1967.
[8] F. Gustavson, W. Liniger, and R. Willoughby, "Symbolic generation of an optimal Crout algorithm for sparse systems of linear equations," T. J. Watson Research Center, IBM Corp., Yorktown Heights, N. Y., Research Rept. RC 1852, June 27, 1967.
[9] W. S. Meyer, "Notes on optimally ordered triangular factorization," University of Minnesota, Madison, Lecture Notes, March 1968.
[10] J. W. Walker, "Renumbering subroutine," Bonneville Power Admin., Portland, Ore., program documentation, July 1967.
[11] W. F. Tinney, "Optimal ordering for sparsely coupled subnetworks," Bonneville Power Admin., Portland, Ore., Rept., Augus 1968.
[12] M. Silverberg, "Near-optimal ordering of electronic circuit equations," Bell Telephone Labs., Whippany, N. J.
[13] B. F. Wallenberg, "Minor diagonal ordering scheme," private communication, Research and Development Center, Leeds and Northrup Co., North Wales, Pa., September 1968.
[14] F. Harary, "A graph theoretic approach to matrix inversion by partitioning," *Numer. Math.*, vol. 4, pp. 128–135, 1962.
[15] A. Z. Gamm, "K voprosu ob uvelichenii effektivnosti algoritmov rascheta rezhima elektrichaskikh sistem," *Izv. Akad. Nauk SSSR, Energ. i Transp.*, no. 3, pp. 3–13, 1968.
[16] H. Edelmann, "Ordered triangular factorization of matrices, or multi-purpose approach for treating problems in large electrical networks," *Proc. 2nd Power Systems Computation Conf.* (Stockholm, Sweden, June 27–July 1, 1966).
[17] A. Brameller, "Application of diakoptics to network analysis," *Proc. 2nd Power Systems Computation Conf.* (Stockholm, Sweden, June 27–July 1, 1966).
[18] H. H. Happ, "Z diakoptics—torn subdivisions radially attached," Paper 4.16, *Proc. 2nd Power Systems Computation Conf.* (Stockholm, Sweden, June 27–July 1, 1966).

[19] C. C. Gotlieb and S. Kumar, "Semantic clustering of index terms," *J. ACM*, vol. 15, pp. 493–513, October 1968.

[20] F. Lucio and M. Sami, "On the decomposition of networks in minimally inter-connected subnetworks," presented at the *IEEE Internatl. Symp. on Circuit Theory*, Miami, Fla., December 4–6, 1968.

21] D. V. Steward, "An improved method for tearing large systems," Atomic Power Equipment Dept., General Electric Co., San Jose, Calif., 1968.

[22] B. A. Carré, "Solution of load-flow problems by partitioning systems into trees," *IEEE Trans. Power Apparatus and Systems*, vol. PAS-87, pp. 1931–1938, November 1968.

[23] E. C. Ogbuobiri and S. Linke, "A unified algorithm for load flow and economic dispatch in electric power systems," *Proc. 1967 IFAC Symp.* (Haifa, Israel).

[24] T. C. Hu, "Decomposition in traveling salesman problems," T. J. Watson Research Center, IBM Corp., Yorktown Heights, N. Y., Research Rept. RC 1527, December 17, 1965.

[25] T. C Hu, "A decomposition algorithm for shortest paths in a network," Mathematics Research Center, University of Wisconsin, Madison, Tech. Summary Rept. 804, September 1967.

[26] J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Operations Res.*, vol. 11, pp. 972–989, November 1963.

[27] J. M. Dobbie, "A survey of search theory," *Operations Res.*, vol. 16, pp. 525–537, May/June 1968.

[28] M. Bellmore and G. L. Nemhauser, "The traveling salesman problem: a survey," *Operations Res.*, vol. 16, pp. 538–558, May/June 1968.

[29] H. W. Dommel and W. F. Tinney, "Optimal power flow solutions," *IEEE Trans. Power Apparatus and Systems*, vol. PAS-87, pp.1866–1876, October 1968.

[30] R. Larson, C. Wells, and J. Peschon, "Efficient computation of state estimates using the inverse of the Jacobian matrix," Wolf Management Services, Palo Alto, Calif., Tech. Memo. 2, August 1968.

[31] R. Bellman, "Dynamic programming treatment of the traveling salesman problem," *J. ACM*, vol. 9, January 1962.

[32] C. Berge, *The Theory of Graphs and Its Applications.* New York: Wiley, 1964.

[33] R. G. Busacker and T. Saaty, *Finite Graphs and Networks.* New York: McGraw-Hill, 1965.

Dynamic Storage and Retrieval in Sparsity Programming

E. C. OGBUOBIRI, MEMBER, IEEE

Abstract—It is shown that sparsity programming is no more than a substitution of a higher level microcompiler for a basic microcompiler in the storage retrieval and processing involving elements of linear and multidimensional arrays. The substitution property of microcompilers permits the coding of a program first in a natural language using formal subscripts and then converting the conventional coding into a sparsity coding after the conventional coding has been fully debugged. This two-stage process not only preserves coding efficiency but also will generally shorten the overall program debugging time. It additionally provides for division of labor between the conventional coder and the sparsity coder. A formal list structuring strategy which has built-in "garbage collection" for sparsity programming is described in detail. This strategy constitutes a conversion guide from conventional to sparsity programming.

INTRODUCTION

MANY industrial problems tax the memory capacity of existing computers. Although more and more storage facilities become available with time, it is well beyond doubt also that problem sizes grow at a faster rate. For a class of problems whose solution involves the manipulation of elements of large but sparse arrays (such as matrices), it has been demonstrated [1] that storage and processing of only the nonzero elements of such arrays not only promotes efficient utilization of available space but also enhances the overall speed with which a solution

is obtained. Thus, even when one has plenty of space for a given class of problems, one should also consider sparsity coding in preference to a conventional coding as a means of reducing computer time.

The author is not immune to the reservations of prospective sparsity programmers. The idea is always there. The goal is around the corner. However the path to it is complicated. The difficulty with sparsity coding in scientific programming appears to lie in the following:

1) there is more to it than knowledge of FORTRAN language,
2) the program logic is more involved, and hence more debugging time is expected.

Certain applications require that the creator of a problem or the inventor of an algorithm play the role of a programmer and vice versa. But this is an ideal combination of talents. Sometimes the creator knows just enough conventional programming to be able to code his problem in a smaller scale. Would it not be helpful to find a programmer who does not understand the totality of the program logic but who has the skill to expand the size of the existing program. This would, in fact, amount to a division of labor that does not involve any loss in efficiency.

This paper purports to formalize sparsity programming in such a way that the subject will become less mysterious to most conventional coders. By introducing and emphasizing a systematic approach, the task of sparsity programming will be much easier both conceptually and in practice. Although rectangular arrays are implied in the text, the extension to higher dimensional arrays is obvious. In the interest of the majority of users, the FORTRAN language is assumed throughout the text. The works of Byrnes [2], Randell and Kuehner [3], and Jodeit [4] are very inspiring.