Substructuring and Ordering: Graph–Theoretical Methods

A. Kaveh[1] and G.R. Roosta[2]

In this paper an efficient algorithm is designed for substructuring, for use in parallel computing, employing simple concepts of graph theory. This algorithm partitions the graph model of a structure into subgraphs with an equal or nearly equal number of internal nodes, while keeping the interface nodes to the smallest possible number. Further refinement of the selected substructures are also made by recursive application of the algorithm. A simplified method is also presented and a nodal ordering algorithm is provided.

INTRODUCTION

Substructuring has been attractive to engineers for various reasons. At early stages of its development, limitation of storage in computers compelled its use for the analysis of large scale problems. Later, its efficiency in analysis encouraged its development. Recently its suitability for parallel processing has been studied. Soon it may become an important tool for topology optimization of structures.

Substructuring has been developed under different names, such as the tearing and interconnecting of Kron's [1] K-partitioning, factored-form technique, doubling technique, elimination-backsubstitution, divide-and-conquer, divide-and-decomposition, dissection, nested dissection, multi-level dissection and subdomaining. Although these methods have some differences, the underlying basic idea does not differ much from the tearing of the main structure into substructures and interconnecting them for a solution. Such an idea dates back to developments made by Whitehead [2] in topology known as "an expansion process" for studying the properties of topological spaces. The generalization of such an expansion process [3] made feasible the study of many topological properties of general skeletal structures.

In many engineering applications, particularly in the analysis and design of large systems, it is convenient to allocate the design of certain components (substructures) to individual design groups. The study of each substructure is carried out more or less independently and the dependencies between the substructures resolved after the study of individual substructures is completed. The dependencies among the components may, of course, require redesign of some of the substructures, so the above procedure may be iterated several times.

In order to make these remarks specific, suppose for a structural model S we choose a set of nodes I and their incident members which, if removed from S, disconnect it into two substructures. If the variables associated with each substructure are numbered consecutively,

1. Dept. of Civil Engineering, Iran University of Science and Technology, Tehran, I.R. Iran
2. Building and Housing Research Center, P.O. Box 13145–1696, Tehran, I.R. Iran

Scientia Iranica, Vol. 1, No. 2, © Sharif University of Technology, July 1994.

followed by the variables associated with I, then the following partitioning of the stiffness matrix A of the entire structure will be induced.

$$A = \begin{bmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{13}^t & A_{23}^t & A_{33} \end{bmatrix}.$$

The Cholesky factor L of A correspondingly partitioned. is given as

$$L = \begin{bmatrix} L_{11} & & \\ 0 & L_{22} & \\ W_{13}^t & W_{23}^t & L_{33} \end{bmatrix},$$

where $A_{11} = L_{11}L_{11}^t$, $A_{22} = L_{22}L_{22}^t$, $W_{13} = L_{11}^{-1}A_{13}$, $W_{23} = L_{22}^{-1}A_{23}$ and $L_{33}L_{33}^t = A_{33} - A_{13}^t A_{11}^{-1}A_{13} - A_{23}^t A_{22}^{-1}A_{23}$. Here A_{11} and A_{22} correspond to each substructure and the matrices A_{13} and A_{23} represent the "glue" which relates the substructures through the nodes of I.

Since the factors of A_{11} and A_{22} are independent, they can be computed in either order, or in parallel if two processes are available. Finally, in some design applications, several substructures may be identical; for example, have the same configuration and properties, and each substructure may be regarded as a super-element which is constructed once and used repeatedly in the design of several structures. In the above example, A_{11} and A_{22} could be identical.

Methods of substructuring are well documented in literature [1, 4–10, among many others]. Although the methods are fully described, not enough attention has been paid to an automatic formation of substructuring. The existing methods are confined to the work of George [7, 11] who employs the properties of level structures (SRT).

In this paper, a simple algorithm is developed for partitioning and ordering the nodes of a structure, which can be incorporated into any program available for analysis. The algorithm is recursively applied, and each selected substructure is decomposed into smaller ones, yielding a nested dissection. A nodal ordering algorithm is also included for bandwidth reduction.

DEFINITIONS AND CONCEPTS FROM GRAPH THEORY

A simple graph S is defined as a set $N(S)$ of nodes and a set $M(S)$ of members together with a relation of incidence which associates two distinct nodes with each member, known as its ends. A member is incident with a node if it is an end node of that member. The valency of a node is the number of members incident with that node. A node n_i is called adjacent to a subgraph S_j, if it is adjacent to a node of S_j.

A tree T of S is a subgraph of S which has no cycle, a cycle being a closed path. A shortest route tree (SRT) rooted from a specified node (starting node) is a tree T containing all the nodes of S and the distance between any node of T and its root is minimum [12].

Let S be decomposed into q subgraphs $S_1, S_2, S_3, \cdots S_q$. A set of nodes of S containing one distinct node from each subgraph $S_i (i = 1, \cdots, q)$ is called the transversal of the subgraphs.

The distance $d(n_i, n_j)$ between nodes n_i and n_j is defined to be the length of the shortest path between these nodes. The following algorithm is used for finding the distance between each pair of the nodes of S. One can, however, employ any other available approach for this purpose.

A matrix $\boldsymbol{NA}$ is considered, which is the same as the node adjacency matrix of S at the beginning of the processes, and its typical entry $NA(i,j)$ denotes the distance between nodes i and j at the end of the processes, where $1 \leq i, j \leq N(S)$. Due to the symmetry of $\boldsymbol{NA}$, i.e. $NA(i,j) = NA(j,i)$, the domains of variations of i and j in the program are restricted to $1 \leq i \leq N(S) - 1$ and $i + 1 \leq j \leq N(S)$. In an adjacency matrix for two nodes i and j of S, if $NA(i,j) = 1$ (or $NA(i,j) = 0$), it means that these nodes are adjacent (or disjointed), but here, $NA(i,j) = 1$ means that the distance between i and j is equal to unity, and $NA(i,j) = 0$ means that the distance between i and j has not yet been calculated.

In Step 1 of the following algorithm, each pair of nodes of S with distance equal to 2 are

determined, in Step 2, those pairs of nodes with distance equal to 3 are obtained. Subsequently, in Step k, the pair of nodes with distance equal to $k+1$ are detected.

Step p ($p = 1$ to $\delta - 1$), should be carried out as follows, with δ being the diameter of S. In step p, for each pair of nodes i and j with $NA(i,j) = 0$, find a node k for which $NA(i,k) + NA(j,k) = p+1$ such that $NA(i,k) \neq 0$ and $NA(j,k) \neq 0$. If such a node is found, then let $NA(i,j) = p+1$.

The eccentricity of a node n_i is taken as:

$$e(n_i) = \text{Max}\, d(n_i, n_j) \quad \text{for} \quad j = 1, 2, \cdots, N(S).$$

Further graph theoretical definitions and concepts used in this paper, may be found in [11].

ALGORITHM FOR SUBSTRUCTURING

Let S be the graph model of a structure. The following algorithm is designed to decompose S into q subgraphs with an equal or nearly equal number of nodes (support nodes are not counted) with the least number of interface nodes.

Step 1. Delete all the support nodes with their incident members and denote the remaining subgraph as S_r.

Step 2. Determine the distance between each pair of nodes of S_r and evaluate the eccentricities of its nodes.

Step 3. Sort the remaining nodes (RN) in ascending order of their eccentricities.

Step 4. Select the first node of RN as the representative node of the subgraph S_1 to be determined and find a second node as the representative node of subgraph S_2 with a maximum distance from S_1.

Step 5. Find the third representative node with the maximum least distance from S_1 and S_2, and denote it by S_3.

Step 6. Subsequently select a representative node of subgraph k for which the least distance from $S_1, S_2, \cdots, S_{k-1}$ is maximum. Repeat this process until q representative nodes of the subgraphs to be selected are found.

Step 7. For each subgraph $S_j (j = 1, \cdots, q)$ add an unselected node n_i of RN, if it is adjacent only to S_j and its least distance from all nodes of other subgraphs is maximum.

Step 8. Continue the process of Step 7, without the restriction of transforming one node to each subgraph S_j, until no further node can be transferred. The remaining nodes in RN are interface nodes.

Step 9. Transfer the support nodes to the nearest subgraphs.

Once the nodes for each subgraph S_j are found, the incident members can easily be specified.

This algorithm is recursively applied to the selected substructures, decomposing each substructure into smaller ones, resulting in a further refinement.

A SIMPLIFIED ALGORITHM

In the following, a simplified algorithm is presented which requires less storage and computer time than the main algorithm, in the expense of selecting subgraphs with a slightly higher number of interface nodes for some structural models. In this approach, the number of distances to be considered and compared for finding the nodes of substructures is far less than the main algorithm, where the distances between each pair of nodes of S had been required. This algorithm consists of the following steps:

Step 1. Form an SRT, rooted from an arbitrary node, in order to find a representative node of S_1 with maximum distance from the root. The selected node is also denoted by S_1.

Step 2. Form an SRT, rooted from S_1, to calculate the distance between each node of S and S_1, and find the representative node S_2 in a maximum distance from S_1.

Step 3. Form an SRT, rooted from S_2, to calculate the distance between each node of S and S_2 and find the representative node S_3 in a maximum least distance from the selected

nodes. Repeat this process until q representative nodes $S_1, S_2, \cdots, S_q$, forming a transversal, are selected.

Step 4. For each subgraph S_i, find a node adjacent to S_i only, with maximum least distance from other representative nodes in turn.

Step 5. Continue the process of Step 4, without the restriction of transforming one node to each subgraph S_i, until no further node can be transferred.

In this algorithm, support nodes are dealt with in a similar way to Steps 1 and 9 of the main algorithm.

EXAMPLES

Example 1.

A cross-shaped single layer grid, as shown in Figure 1, is considered and partitioned into $q = 2, 4, 6$ and 8 substructures. The partitions obtained and the corresponding node adjacency matrices are presented in Figure 2a–h. The selected interface nodes are shown by solid circles. For the case $q = 2$, the selected substructures are further refined with $q' = 2$ and 3 and the corresponding matrices are illustrated in Figure 3a–b.

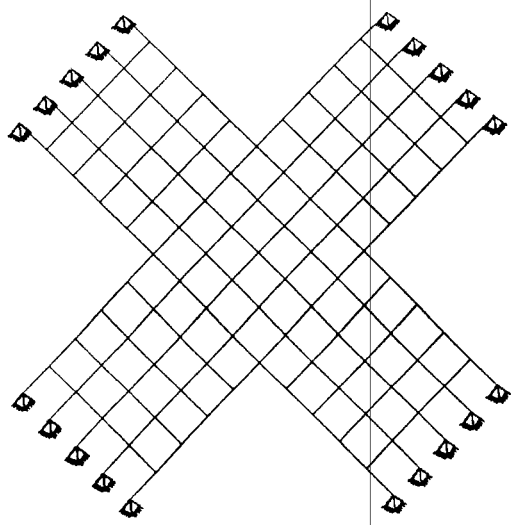

Figure 1. A cross-shaped single layer grid S.

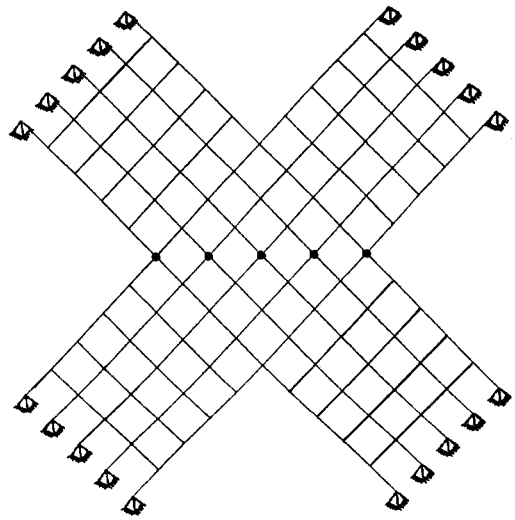

Figure 2a. Partitioned model for $q = 2$.

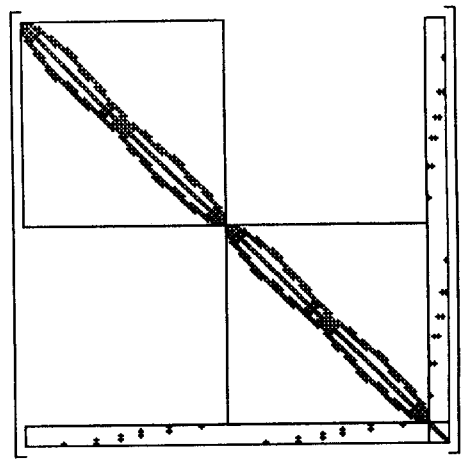

Figure 2b. Adjacency matrix for $q = 2$.

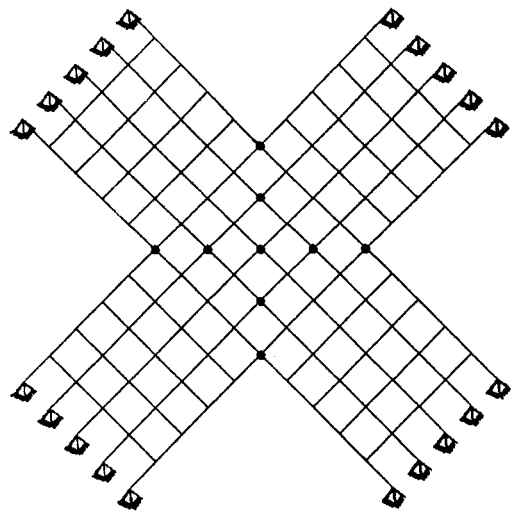

Figure 2c. Partitioned model for $q = 4$.

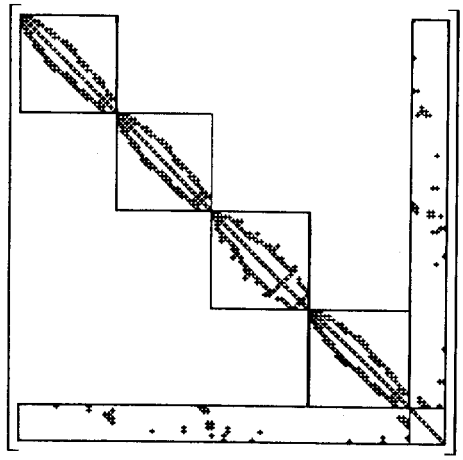

Figure 2d. Adjacency matrix for $q = 4$.

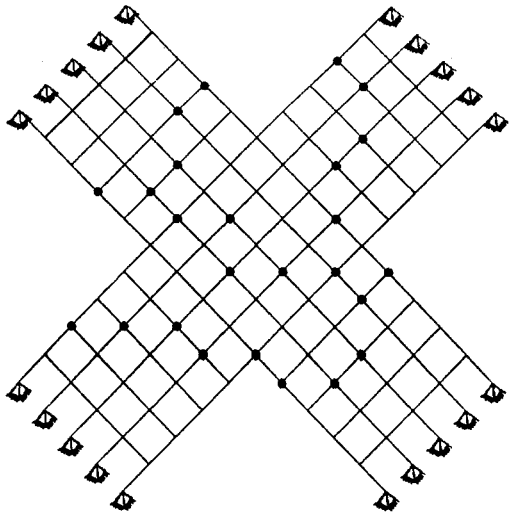

Figure 2e. Partitioned model for $q = 6$.

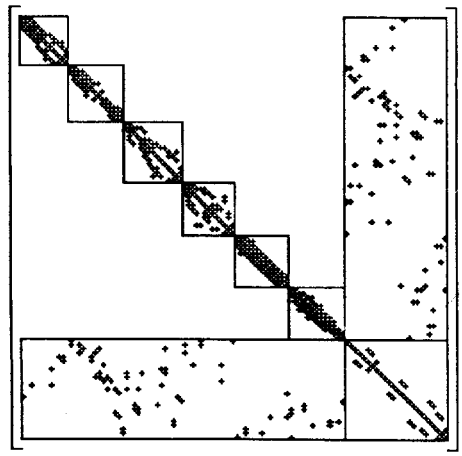

Figure 2f. Adjacency matrix for $q = 6$.

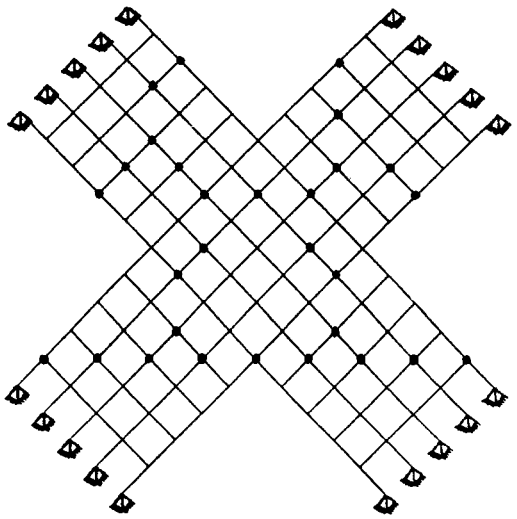

Figure 2g. Partitioned model for $q = 8$.

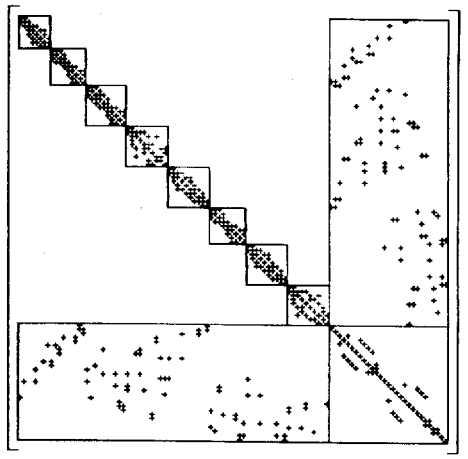

Figure 2h. Adjacency matrix for $q = 8$.

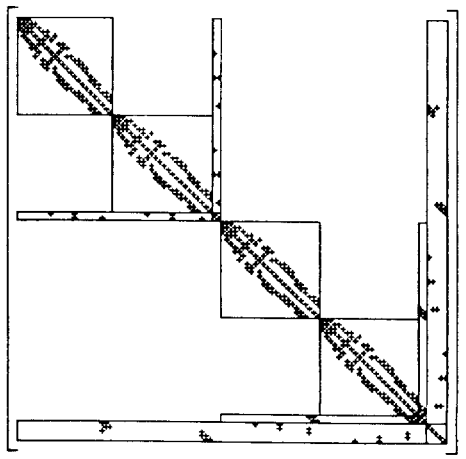

Figure 3a. Adjacency matrix for $q = 2$ and $q' = 2$.

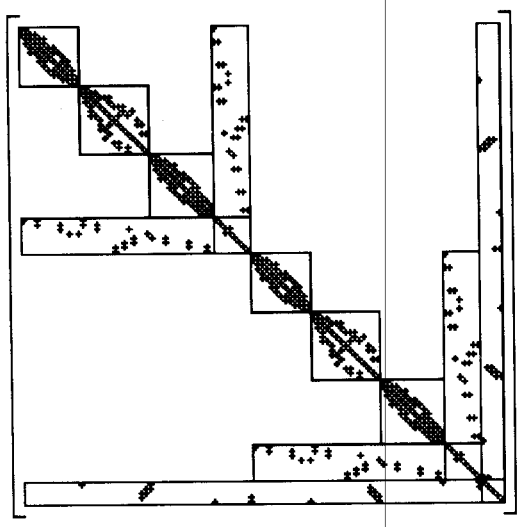

Figure 3b. Adjacency matrix for $q = 2$ and $q' = 3$.

Example 2.

A double layer grid, supported at four corner nodes, as shown in Figure 4, is considered and partitioned into $q = 2$ and 4 substructures. The corresponding node adjacency matrices are presented in Figure 5a–b. For the case of $q = 2$, the selected substructures are further refined with $q' = 2$ and 3 and the corresponding matrices are illustrated in Figure 6a–b.

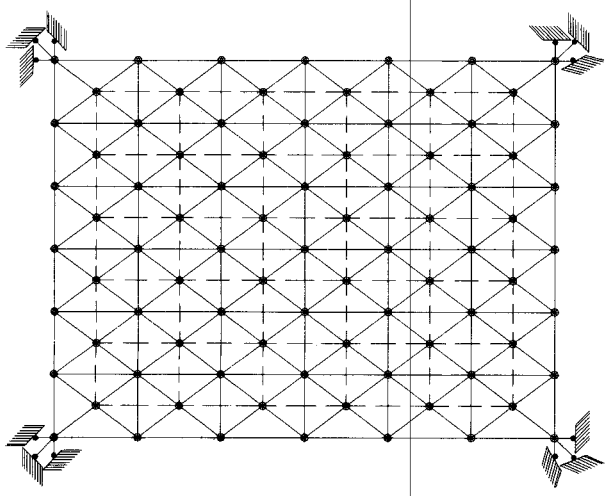

Figure 4. A double layer grid.

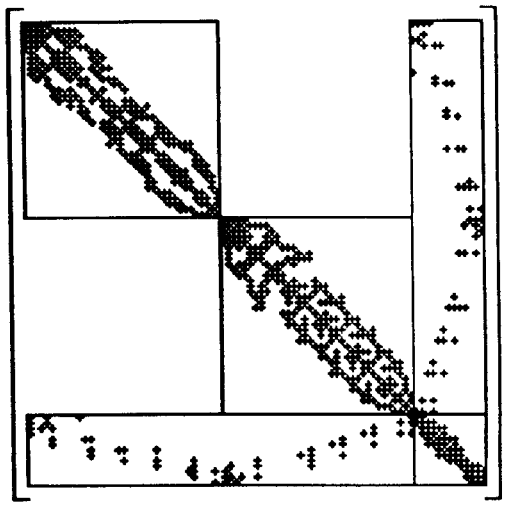

Figure 5a. Adjacency matrix for $q = 2$.

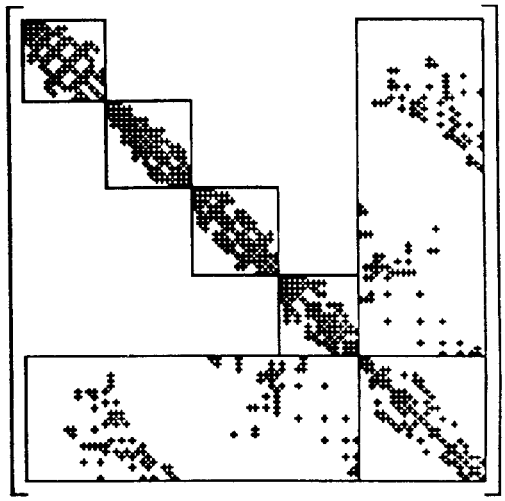

Figure 5b. Adjacency matrix for $q = 4$.

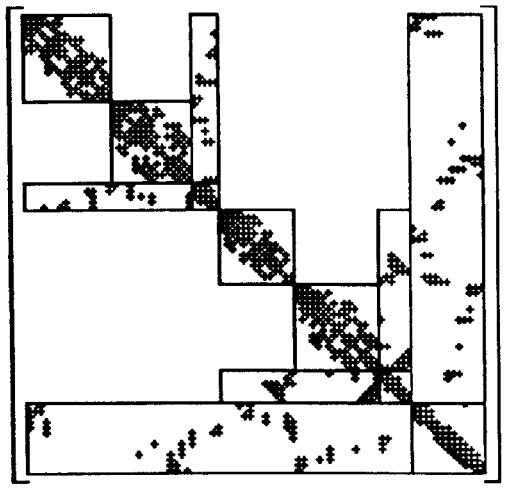

Figure 6a. Adjacency matrix for $q = 2$ and $q' = 2$.

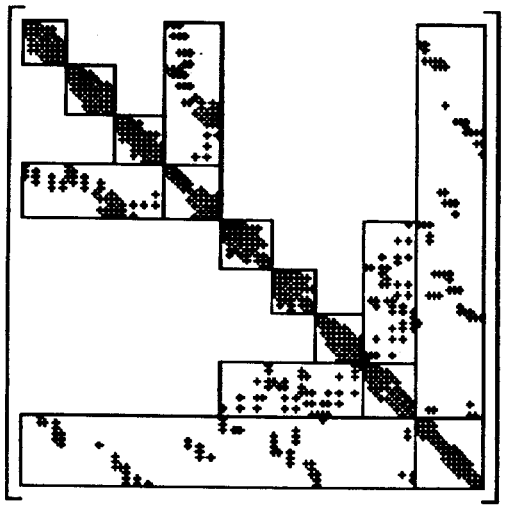

Figure 6b. Adjacency matrix for $q = 2$ and $q' = 3$.

Example 3.

A dome-type space structure S, as shown in Figure 7, is considered and partitioned into $q = 2$, 3, 4 and 5 substructures. The corresponding node adjacency matrices are presented in Figure 8a–d. For the case of $q = 2$, the selected substructures are further refined with $q' = 2$ and 3 and the corresponding matrices are illustrated in Figure 9a–b.

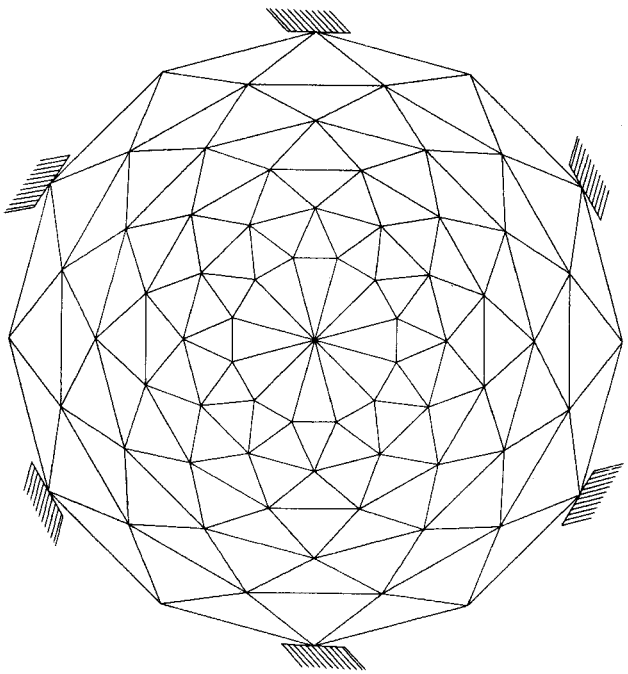

Figure 7. A space structure S.

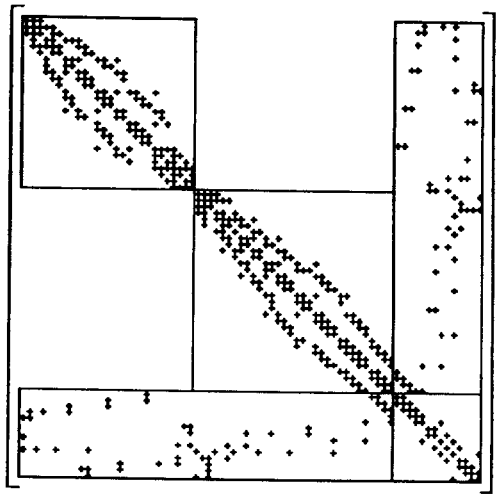

Figure 8a. Adjacency matrix for $q = 2$.

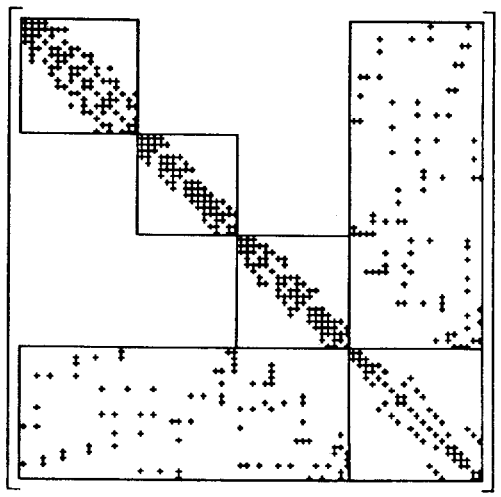

Figure 8b. Adjacency matrix for $q = 3$.

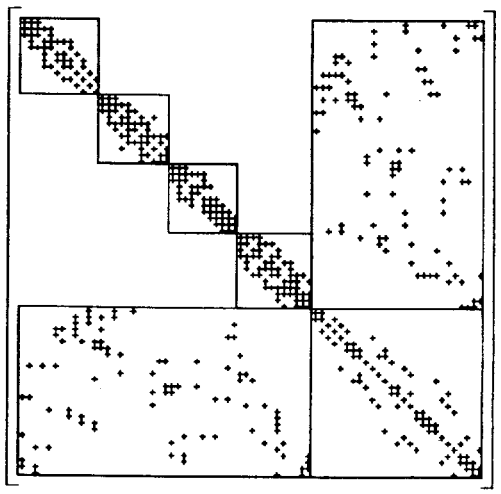

Figure 8c. Adjacency matrix for $q = 4$.

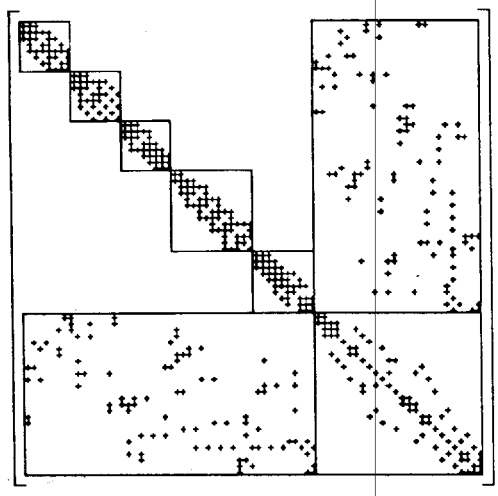

Figure 8d. Adjacency matrix for $q = 5$.

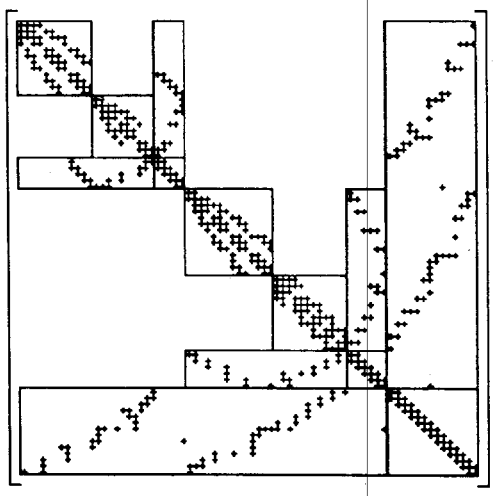

Figure 9a. Adjacency matrix for $q = 2$ and $q' = 2$.

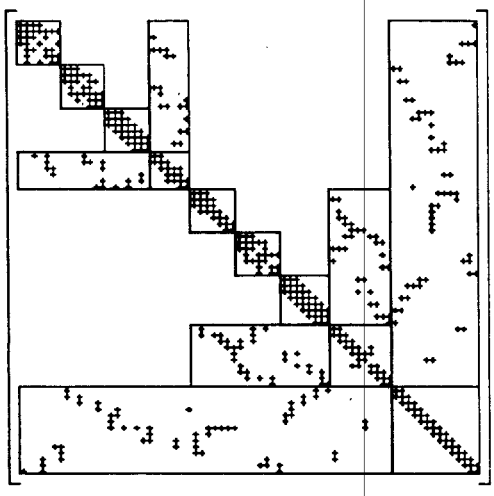

Figure 9b. Adjacency matrix for $q = 2$ and $q' = 3$.

ORDERING FOR BANDWIDTH REDUCTION

Once the substructures and the interface nodes are specified, the following algorithm can be used for the nodal numbering.

Step 1. Select a good starting node O for S and order the substructures according to their distance from O. The distance of a substructure is taken to be the length of the shortest path between the nearest node of the substructure and O.

Step 2. Find a good starting node of S_1 and use any nodal ordering routine available (e.g., the algorithm of [13]) for numbering the internal nodes of S_1 .

Step 3. Repeat Steps similar to that of Step 1 for all the other substructures, in turn.

Step 4. Select a node of minimal valency from interface node-set adjacent to S_1 and order the interface nodes using a nodal ordering algorithm. In the process of numbering, when possible, priority is given to the nodes adjacent to lower numbered substructures. For this numbering, the underlying topology of S is used.

Step 5. Support nodes are numbered after the numbering of the internal and interface nodes is completed.

In this algorithm, one can also order the representative nodes of the selected transversal for ordering the substructures.

CONCLUDING REMARKS

The algorithms developed in this paper are simple and can be used for partitioning the graph model of systems. The application can easily be extended to finite element models. For this purpose an associate graph $A(S)$, of the model S can easily be constructed, as defined in [12]. The algorithm can then be applied to $A(S)$, and the interface nodes specify the elements corresponding to separators of the FE model. Ordering the nodes of each subgraph will specify the order of the corresponding sub-domain. Once the order of elements is known, an ordering within each element can easily be

performed by specifying certain priority for the nodes.

The proposed method is extended to multi-level substructuring, by applying the same process to each selected substructure, resulting in a more refined substructuring.

The application of the presented method is by no means restricted to the analysis and design of structures. It can be used for the analysis of any other system, such as electrical and hydraulic networks, and also the study of those sparse matrices for which a graph model can be associated.

REFERENCES

1. Kron, G. "Solving complex elastic structures in easy stages", *J. Appl. Mech.*, **22**, pp 235–244 (1955).
2. Whitehead, J.H.C. "Simplicial spaces, nuclei and m-groups", *Proc. Lond. Soc.*, **45**, pp 243–327 (1939).
3. Kaveh, A. *Application of Topology and Matroid Theory to the Analysis of Structures*, Ph.D. thesis, London University, IC (1974).
4. Roth, J.P. "An application of algebraic topology: Kron's method of tearing", *Quart. Appl. Math.*, **17**, pp 1–23 (1959).
5. Noor, A.K., Kamel, H.A. and Fulton, R.E. "Substructuring technique-status and projections", *Comput. Struct.*, **8**, pp 621–632 (1978).
6. Spillers, W.R. "Application of topology in structural analysis", *Struc. Div.*, ASCE, **89**, pp 301–313 (1963).
7. George, A. "Nested dissection of a regular finite element mesh", *SIAM J. Numer. Anal.*, **10**, pp 345–363 (1973).
8. George, A. and Liu, J.W.H. "Algorithms for partitioning and numerical solution of finite element systems", *SIAM J. Numer. Anal.*, **15**, pp 297–327 (1978).
9. Przemienecki, J.S. "Matrix structural analysis of substructures", *AIAA J.*, **1**, pp 138–147 (1963).
10. Adeli, H. and Kamal, O. *Parallel Processing in Structural Engineering*, Elsevier Appl. Sci, London (1993).
11. George, A. "Solution of linear system of equations; direct methods for finite element problems", *Lect. Notes Math.*, A. Dold and E. Eckmann, Eds., Springer Verlag, **572**, pp 52–101 (1977).
12. Kaveh, A. *Structural Mechanics: Graph and Matrix Methods*, RSP Ltd. (John Wiley), Exeter, UK (1992).
13. Kaveh, A. "Ordering for bandwidth reduction", *Comput. Struct.*, **24**, pp 413–420 (1986).