1986

NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

MARSHALL SPACE FLIGHT CENTER
THE UNIVERSITY OF ALABAMA

ELIMINATION SEQUENCE OPTIMIZATION FOR SPAR

Prepared By:	Harry A. Hogan
Academic Rank:	Assistant Professor
University and Department:	Louisiana Tech University Mechanical Engineering Dept.
NASA/MSFC:	
Laboratory:	Systems Dynamics
Division:	Structural Dynamics
Branch:	Systems Analysis
MSFC Colleague:	Larry Kiefling
Date:	11 July 1986
Contract No.:	NGT 01-002-099 The University of Alabama

ELIMINATION SEQUENCE OPTIMIZATION FOR SPAR

by

Harry A. Hogan
Assistant Professor of Mechanical Engineering
Louisiana Tech University
Ruston, LA 71272

ABSTRACT

SPAR is a large-scale computer program for finite element structural analysis. The program allows user specification of the order in which the joints of a structure are to be eliminated since this order can have significant influence over solution performance, in terms of both storage requirements and computer time. An efficient elimination sequence can improve performance by over 50% for some problems. Obtaining such sequences, however, requires the expertise of an experienced user and can take hours of tedious effort to affect. Thus, an automatic elimination sequence optimizer would enhance productivity by reducing the analysts' problem definition time and by lowering computer costs.

Two possible methods for automating the elimination sequence specification have been examined. Several algorithms based on graph theory representations of sparse matrices have been studied with mixed results. Significant improvement in program performance has been achieved, but sequencing by an experienced user still yields substantially better results. Effort has also been directed toward developing a "rational" approach whereby the sequencing routine attempts to mimic as closely as possible the actions of an experienced analyst. Preliminary tests with simple example problems have provided guardedly promising results; performance near that of user-specified sequences has been achieved. In order to obtain sufficient generality to handle a wide variety of problems, however, extensive developmental work will likely be required. Nevertheless, these initial results provide encouraging evidence that the potential benefits of such an automatic sequencer would be well worth the effort.

ACKNOWLEDGEMENTS

I thank Dr. L. M. Freeman, Mrs. Ernestine Cothran, and Dr. Fred Speer for their respective roles in making this fellowship an enjoyable and stimulating experience. Sincere gratitude is also expressed to Mr. Larry Kiefling for motivating and guiding this project. I have benefited immensely from collaboration with a person of his caliber.

The interaction with others at MSFC is also worthy of mention. In particular, informal discussions of both a technical and non-technical nature with Messrs. Wayne Ivey and Stan Fuller have made my time at MSFC more informative and interesting. Getting a glimpse of the day-to-day activities of the people and programs of MSFC has been not only educational but rewarding as well. I also owe many thanks to Mr. Wayne Ivey for so graciously tolerating my sharing his office.

INTRODUCTION

Much of the structural design and analysis work done throughout NASA, as well as in industry and universities, depends heavily on the use of large-scale finite element programs like NASTRAN, SPAR/EAL, and ANSYS. The power and capability of these codes makes them indispensable tools to workers in this area. A unique feature of SPAR/EAL+ is that the user can influence program performance by providing supplemental input data that specifies the order in which the joints of the structure are to be eliminated. Improvements in storage requirements, solution time, and in some cases solution accuracy can be realized by appropriate specification of joint elimination sequences. For example, a recent problem being studied by Larry Kiefling at MSFC showed a 35% decrease in solution cost when provided with a "good" joint elimination sequence. The tradeoff involved is the time required by the analyst to devise such a sequence. Two and one-half to three hours were taken in selecting the sequence for the example cited above. The ability to identify efficient sequences also requires extensive experience by the user. A compeling case can thus be made that an automatic elimination sequencer would be of significant benefit by reducing the analysts' problem definition time and lowering computer costs.

+SPAR and EAL were both developed by Whetstone [1,2] and are quite similar in capability. SPAR was developed under contract to NASA's MSFC and LaRC, wheras EAL is commercially marketed by Engineering Information Systems, Inc. of San Jose, California.

OBJECTIVES

The objectives pursued in this project are summarized as follows:

-- to become more familiar with the storage and solution procedures employed in SPAR/EAL, with special attention to understanding the effects of the joint elimination sequence

-- to research possible ways to automate the specification of efficient joint elimination sequences

-- to begin developing, implementing, and evaluating promising methods for automatic sequencing

-- to periodically assess the progress and future prospects of such endeavors

Because of the rather uncertain nature of this work, these objectives were pursued with a combination of sequential and simultaneous effort.

THE SPAR/EAL COMPUTER PROGRAM

Background

SPAR was originally developed under contract by W. D. Whetstone for NASA's Marshall Space Flight Center and Langley Research Center. The program allows for both static and dynamic analysis of large-scale (greater than 10^4 degrees of freedom) structural analysis problems. Typical, low-order interpolation elements ranging from one-dimensional beam and bar elements to three-dimensional solid elements are available. The program also contains two- and three-dimensional fluid elements. EAL was developed subsequent to SPAR and is commercially marketed by Engineering Information Systems, Inc., of San Jose, California. Its capabilities are basically the same as that of SPAR with some improvements and minor modifications. Hence, the two will be referred to as a single entity except when distinction is needed.

Overview

SPAR/EAL actually consists of a series of processors and subprocessors that can be independently executed by appropriate commands. Problem data is stored in large data tables that are accessed by the processors and subprocessors as needed. A schematic of the primary processors involved in problem definition is depicted in Figure 1 (taken from EAL Reference Manual [2]).

Processor TAB contains subprocessors that allow specification of joint location information (TAB/JLOC), material properties (TAB/MATC), and motion constraints (TAB/CON). Subprocessor TAB/JSEQ provides for alternate joint elimination sequences; the default is for the joints to be eliminated in the order in which they were generated by TAB/JLOC. Processor ELD is needed to define the particular finite elements of choice and their connection to the joints. Processor TAN is then typically invoked to analyze the joint and element information and calculate statistics characterizing computer storage and solution time requirements. System stiffness and mass matrices are generated by other processors from "E-State" data as indicated in Figure 1. This E-State is produced by intermediate processors utilizing the basic joint and element data. The other major processor shown in Figure 1

is RSI, which factors the system stiffness matrix. Processors TAN and RSI are known as TOPO and INV, respectively, in the SPAR version of the program. Subsequent processors (not shown) define the loading case, or cases, to be considered and provide solutions for the particular load cases and constraint conditions chosen. Processors for manipulating the data in the data tables and for post-processing of results are also available.

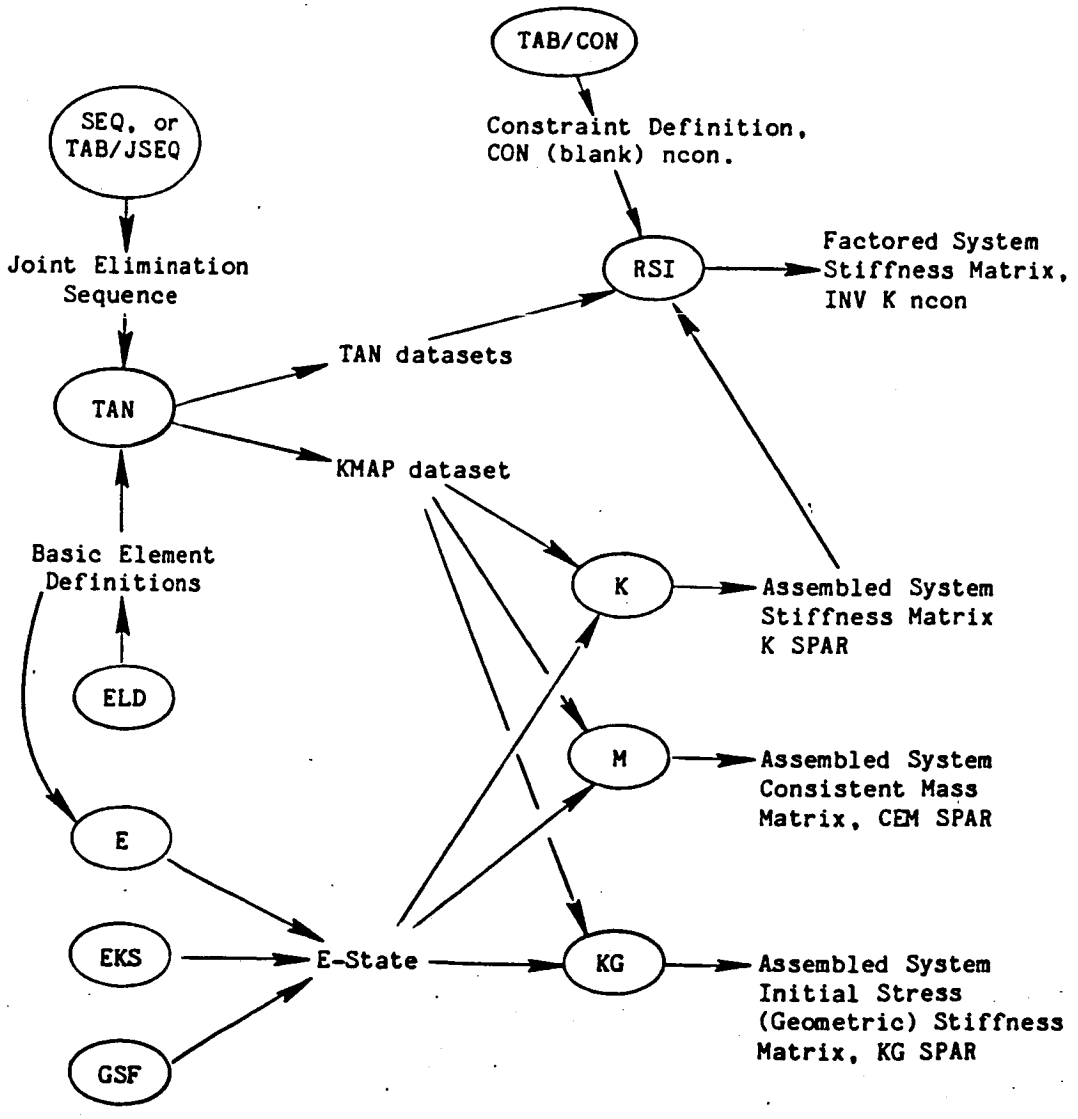


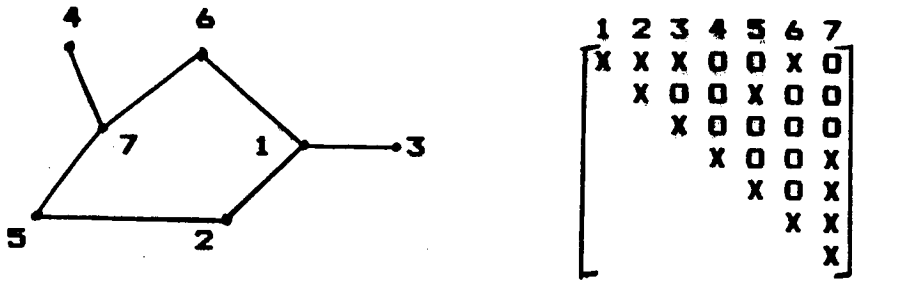
Figure 1. Problem definition processors.

Joint Elimination Sequence

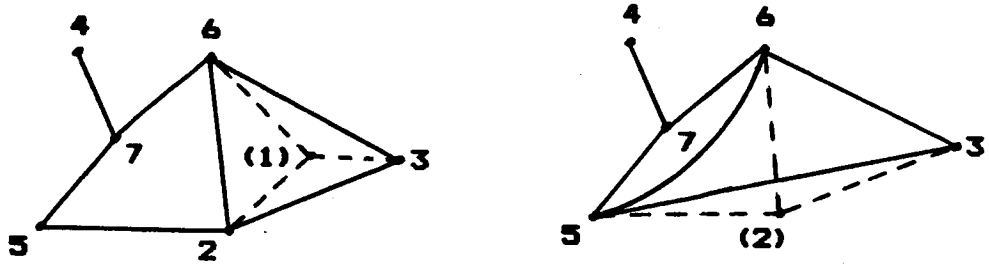
The fundamental concepts underlying the storage and solution scheme used by SPAR/EAL was described in a technical paper by Whetstone [3]. A critical aspect of this scheme is the order in which the joints of a structure are eliminated during solution. Stiffness matrices produced by the overriding majority of problems are symmetric and sparse (many zero-valued entries). As factorization of such matrices proceeds, the process of matrix "fill-in" occurs whereby the sparsity of the portion of the matrix remaining to be factored is diminished. This fill-in increases the storage requirement and number of operations involved in factorization. Hence, computer costs increase. Proper specification of a joint elimination sequence can significantly reduce the amount of fill-in incurred and consequently reduce computer costs.

Figure 2 will aid in beginning to understand the basic mechanism of matrix fill-in. Consider the lower portion of part (a.) to be a structure composed of one-dimensional finite elements with the corresponding upper-triangular stiffness matrix depicted above. Note that X represents non-zero terms and 0 zero terms. Also recall that non-zero terms appear in row-column locations corresponding to interconnected joint numbers. For example, joint 1 is connected to joints 2, 3, and 6, so non-zero terms appear in columns 2, 3, and 6 of row 1. The lower portion of part (b.) contains a "graphical analog" of the joint elimination process. The elimination of joint 1 creates new interconnections between joints 3 & 6, 3 & 2, and 2 & 6 that did not previously exist. Thus, new non-zero terms enter into the matrix accordingly, as seen in the upper portion of part (b.). Part (c.) contains the same information for the elimination of joint 2. Note that the total number of new non-zeroes associated with the elimination of these two joints is five. It can be verified easily that if the first two joints eliminated were joint 3 and then joint 1, then the total number of new non-zeroes introduced would be one. Even a simple example like this serves to demonstrate the substantial benefits offered by a "good" joint elimination sequence.

Recognizing the benefits derived from proper joint elimination sequencing is the easy part of the problem; devising sequences that exploit these benefits is the difficult part. In the 1978 version of the Reference Manual for SPAR [1] mention is made of an automatic elimination sequencer planned for future development. Guidelines are then given for manual sequencing. They can be summarized as follows:



(a.)



(b.) (c.)

Figure 2. Joint elimination and matrix fill-in.

(1) Carefully study Whetstone's original paper [3] and learn how to determine favorable sequences.

(2) Consult an experienced user of SPAR.

(3) Use a numbering sequence appropriate for band-matrix or wavefront procedures.

Comment is included with these recommendations that band-matrix and wavefront methods are different from those employed in SPAR yet results may be satisfactory for problems of small to moderate size. Experience indicates that results are erratic and indeed generally not very good for larger problems. Nevertheless, current versions of EAL have a new processor called SEQ (see Figure 1) that implements the widely-used GPS [4] bandwidth minimizing method. For review, bandwidth and profile storage schemes are presented in Figure 3. Only the terms enclosed by the curves are stored and operated upon. Wavefront methods utilize profile storage and and can realize improved performance by profile-reducing algorithms.

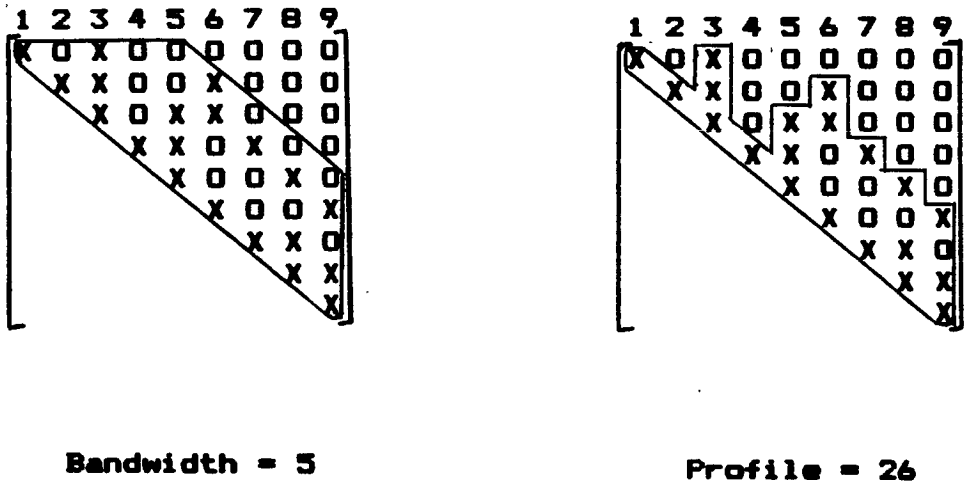


Figure 3. Bandwidth and profile storage schemes.

Automatic Sequencing

Need remains for an automatic joint elimination sequencer that will:

-- generate a favorable sequence with minimum user input

-- handle a wide variety of problem types and configurations.

Two seemingly promising approaches for tackling this problem have been identified and will be described in the

next two sections. Algorithms based on graph theory have been developed in recent years for manipulating sparse matrices. Moderate success has been reported for some problems, but no such methods exist for exploiting the particular storage and solution methods used in SPAR/EAL. A "rational" method is also being pursued whereby the actions of an experienced user are imitated as closely as possible.

GRAPH THEORY METHODS

Fundamental Concepts

Graph theory is an abstract mathematical theory that has found application in areas as diverse as map coloring, route planning for traveling salesmen, and sparse matrix manipulation. Only a cursory introduction to the very basic concepts and definitions will be attempted here.

A graph is defined as a finite set of nodes or vertices together with a finite set of edges. An edge is simply an unordered pair of vertices. A labelled or ordered graph has a one-to-one mapping of successive integers (1, 2, 3, ..., N) onto the set of N vertices. Graphs are commonly represented pictorially by points or small circles for vertices and lines or curves between points for edges.

A pair of vertices is adjacent if the pair form an edge of the graph. If X is the set of vertices of a graph and Y is some set of vertices that is a subset of X, then the adjacency set of Y is the set of vertices that are members of X and are adjacent to at least one vertex in Y. In the simplest case, the set Y is a single vertex and the adjacency set consists of all nodes adjacent to this vertex. The degree of Y is defined to be the number of members in the adjacency set.

Application to Sparse Matrices

The particular application of graph theory to sparse matrices [5,6] that will be examined is that for sparse matrices arising from finite element analysis. Figure 4 illustrates the interrelationships between sparse matrices, their corresponding labelled graphs, and the finite element structures giving rise to such matrices.

The lowermost portion of part (a.) depicts a one-dimensional finite element mesh with a schematic representation of its stiffness matrix immediately above. As before, X simply indicates a non-zero term and 0 a zero term. The structure of the matrix again takes its form from the connectivity of the finite element structure. The relationship between a labelled graph and its corresponding matrix representation is defined in a quite similar manner.

In this case, non-zero terms enter each row of the matrix in columns matching the labels of the vertices in the adjacency set of the vertex labelled with the row number. Non-zero terms also appear on the diagonal for each vertex. The labelled graph for part (a.) of Figure 4 is the uppermost item. As apparent from the figure, the pictorial representations of the graph and finite element mesh for a given matrix are essentially the same for one-dimensional finite elements. Part (b.) of Figure 4 contains the respective three items for a mesh of two four-noded two-dimensional finite elements. The only difference of note is that the pictorial representations of the finite element mesh and the graph are not the same. Any two nodes common to an element are interconnected structurally and the matrix contains non-zero terms indicating so. The labelled graph indicates this same interconnection by having edges between corresponding pairs of vertices.

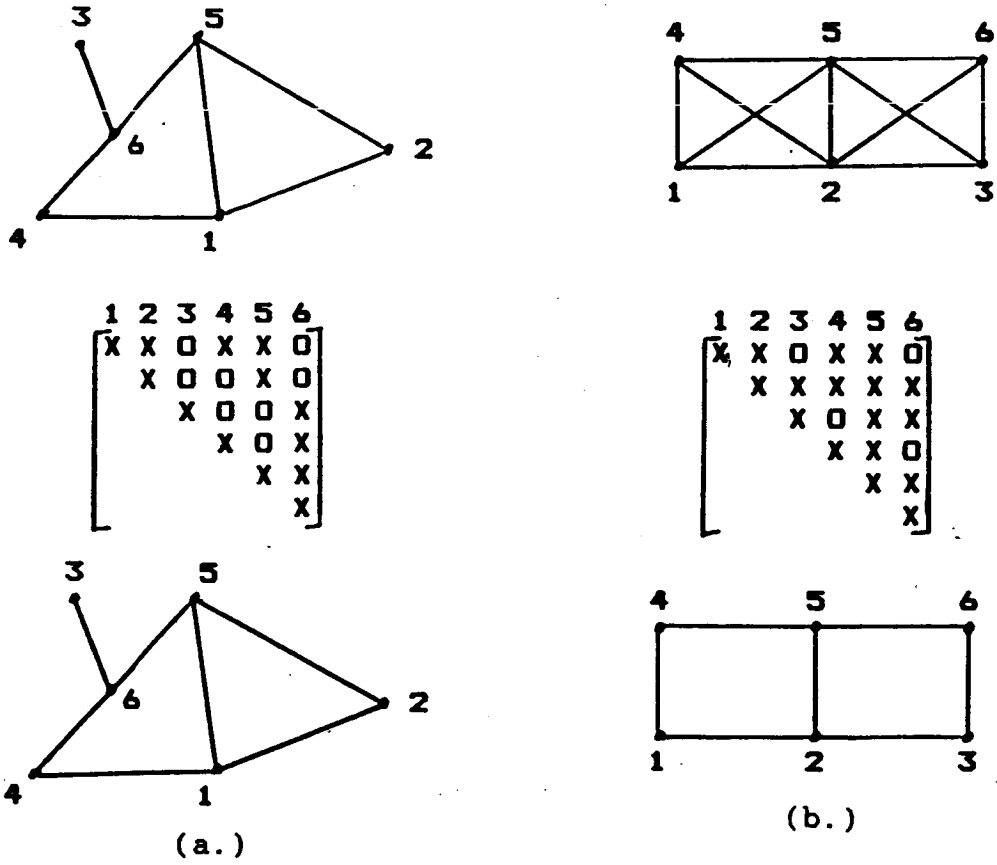


Figure 4. Matrices and labelled graphs for finite elements.
(a.) One-dimensional finite element case.
(b.) Two-dimensional finite element case.

The computer representation of a labelled graph consists primarily of two vectors. One vector contains the entries in the adjacency set of each vertex arranged sequentially. The other contains pointers to the locations in the first vector of the beginning of the adjacency set of each vertex. The adjacency sets and degrees for each vertex in the graphs of Figure 4 are presented in Figure 5 below.

vertex	adjacency	degree	vertex	adjacency	degree
(1)	2 4 5	3	(1)	2 4 5	3
(2)	1 5	2	(2)	1 3 4 5 6	5
(3)	6	1	(3)	2 5 6	3
(4)	1 6	2	(4)	1 2 5	3
(5)	1 2 6	3	(5)	1 2 3 4 6	5
(6)	3 4 5	3	(6)	2 3 5	3

Figure 5. Adjacency and degree of labelled graphs.

A comprehensive treatment of sparse matrix manipulation algorithms based upon graph theory concepts is presented by George and Liu [6]. Several of the algorithms described in this book have been incorporated into an experimental processor for EAL known as RSEQ. The development of RSEQ is spearheaded largely by Sue McCleary and William Greene of NASA's Langley Research Center. Their generosity in making this processor available for the present work is gratefully acknowledged. Having these algorithms already implemented and operational has been extremely helpful.

The three algorithms that have been included in this study are the Reverse-Cuthill-McKee, Minimum Degree, and Nested Dissection algorithms. The Reverse-Cuthill-McKee algorithm is basically a profile minimizing method. A labelling is first determined by the Cuthill-McKee method and then reversed. The Cuthill-McKee approach involves successively numbering the unnumbered neighbors of vertices in order of increasing degree. It is very sensitive to the choice of a starting vertex. Fairly successful algorithms exist for finding such a vertex.

Both the Minimum Degree and Nested Dissection methods attempt to reduce matrix fill-in. The Minimum Degree approach involves the use of elimination graphs, which are graph theory representations corresponding to the reduced structures depicted in Figure 2. Labelling consists of choosing the vertex in each elimination graph having the minimum degree to be the one eliminated next. Nested Dissection makes use of a separator, which is simply a set of vertices whose removal from a graph forms two graphs. Removing separators from graphs affects a partitioning of the accompanying matrix representation of the graph. The Nested Dissection method seeks to find separators that partition the matrix such that all-zero submatrices remain so throughout factorization.

Such brief descriptions of these methods belie their true complexity. The reader is referred to the books by Pissanetsky [5] and George and Liu [6] for more thorough presentations of these methods. It should also be emphasized that these algorithms were not developed to exploit the particular features of the storage and solution scheme employed by SPAR/EAL. In fact, they include recommendations for storage and solution schemes suitable for each.

A RATIONAL APPROACH

Initial Ideas

The basic approach decided upon for pursuing a method to imitate an experienced user involves evaluating the structural topology of the finite element mesh and identifying all "branches" and "holes" in the mesh. The shaded region shown in Figure 6 represents a structure containing such features. Numbering nodes would then proceed so as to eliminate the branches first and then reduce the remaining structure by eliminating nodes "radially" thereby progressing "around" the hole. Initial efforts have been concentrated on developing methods for identifying and eliminating branches.

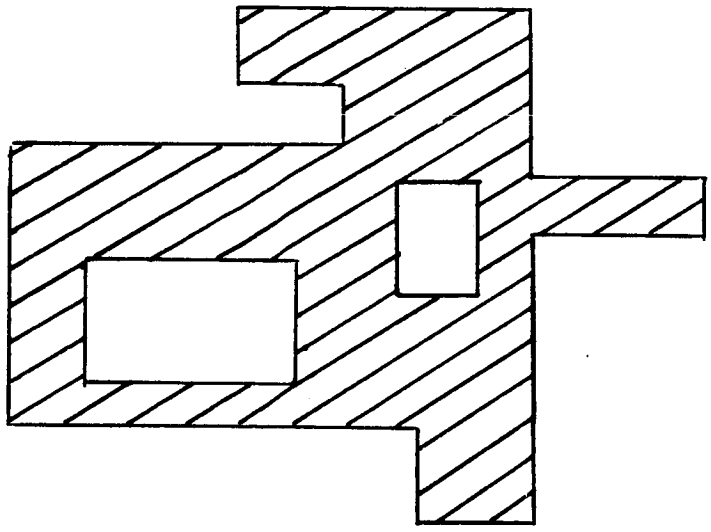


Figure 6. A region with branches and holes.

Branch Elimination

The branch elimination procedure is summarized in the functional flowchart of Figure 7. For this phase of development, the finite element mesh is assumed to be free

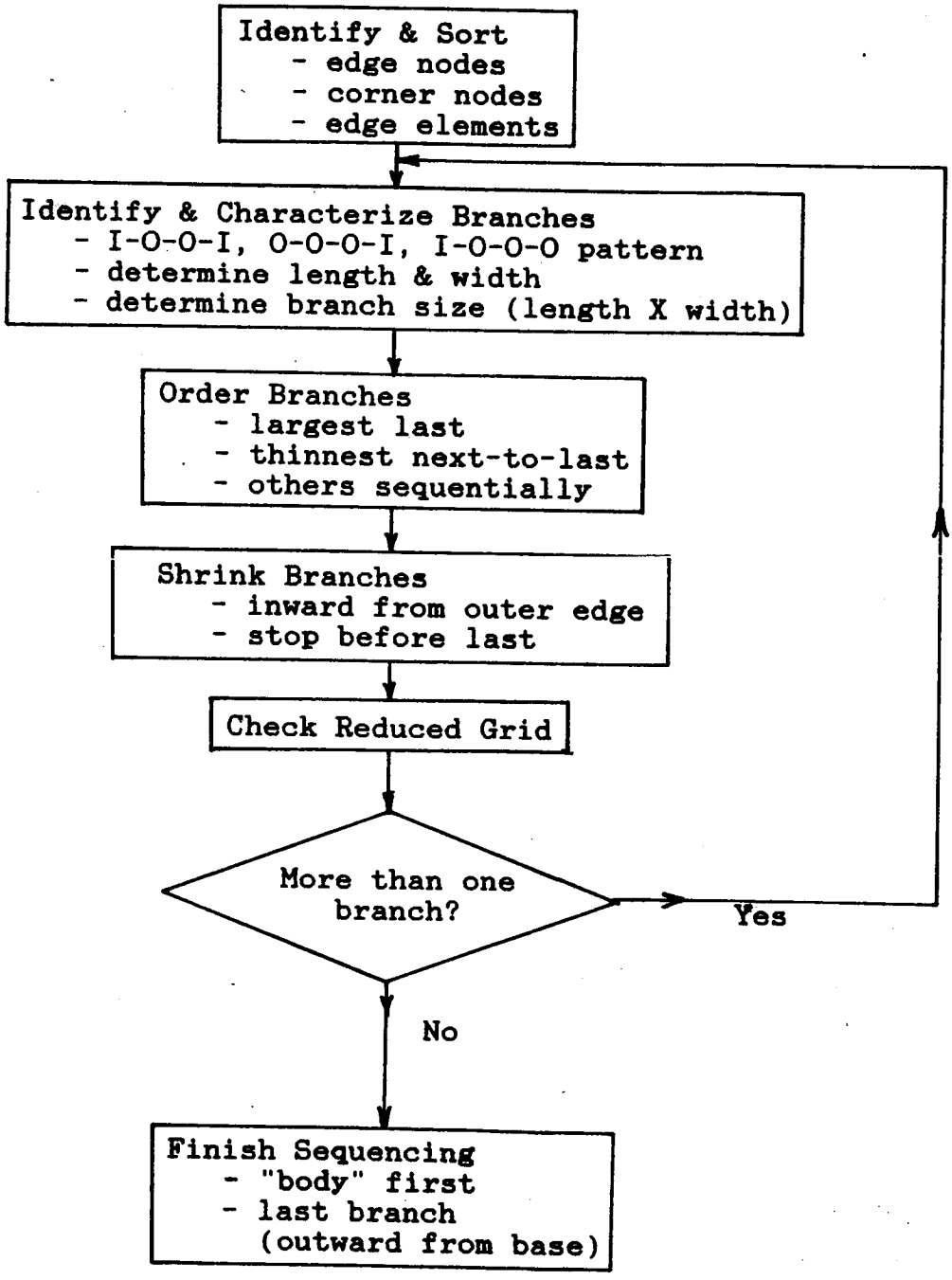


Figure 7. Branch shrinker flowchart.

of any "holes" and composed of four-node two-dimensional elements. The first block involves characterization of the edge of the finite element mesh. To accomplish this, all nodes located along the edge are identified by counting the number of elements to which each node is common. For four-node elements, all nodes common to three or fewer elements will be on the edge of the domain. This information is then used to identify all edge elements, i.e. elements containing edge nodes. Corner nodes are those common to either one or three elements, with the former being marked as "outer" corners and the latter as "inner" corners. Edge nodes are also sorted in order of occurrence around the periphery of the mesh.

The next main task is that of identifying branches. This is simply done by scanning the sorted list of corner nodes and defining a branch to be represented by one of the following patterns of inner(I) and outer(O) corners: I-O-O-I, O-O-O-I, or I-O-O-O. The length of each branch is determined as the minimum number of elements from the "base" of the branch to its end, while the width is the number of elements between the two outside corners at its end. An indication of the "size" of each is also calculated by taking the product of the length and width.

Branch elimination order is determined by first reserving the largest branch for elimination last. The branch with the smallest width is designated to be eliminated second-to-last. The remaining branches are then ordered with the same sequence with which they were initially identified. Next, each branch is eliminated in order by numbering the nodes as the branch is scanned across its width from the outer end to the base. All are eliminated in this manner except the last (largest) branch. At this point, the reduced mesh is rechecked for branches and all branches are eliminated in order, except the last. This procedure is repeated until only a rectangular domain remains with the last branch forming three sides. This rectangle is then sequenced by scanning across its width but proceeding from the region at the base of the last branch toward the outer end of this branch.

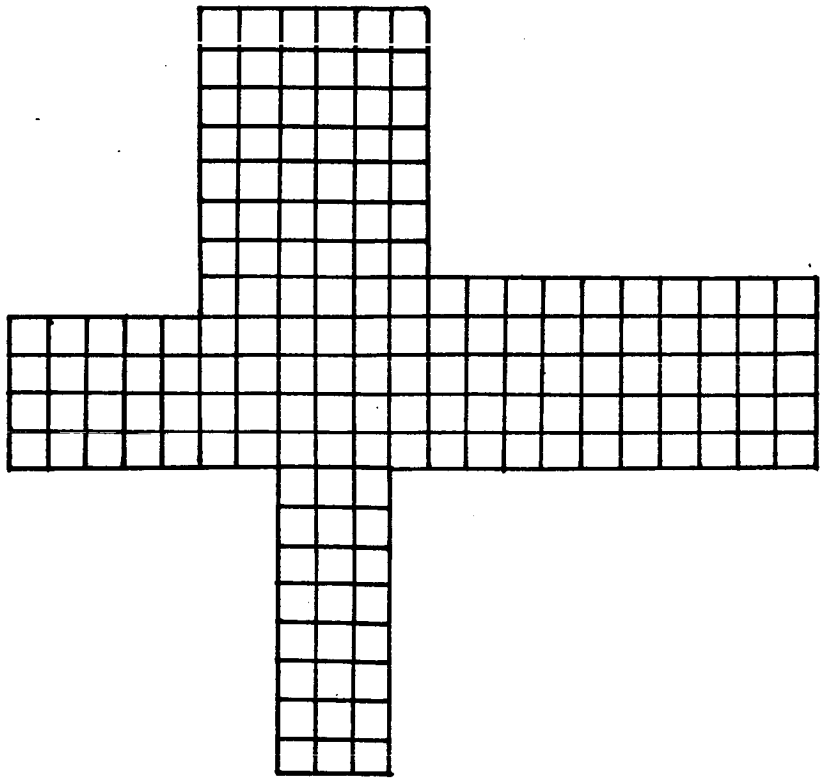
EXAMPLE PROBLEMS

The graph theory algorithms and rational approach were evaluated for two examples. Both examples are relatively simple two-dimensional problems with four-node finite elements. Each also contains a variety of branches but no holes. The jobs were run using EAL on a VAX 11/785 super minicomputer. This version of EAL also contains the experimental processor RSEQ. For the automatic sequencer implementing the rational approach, joint elimination data was generated independent of EAL and then incorporated into the problem input data using the subprocessor TAB/JSEQ.

Figure 8 contains a diagram of the finite element mesh along with a summary of results for this problem. The data denoted "none" refer to the case of no particular sequence being specified. The joints are eliminated in whatever sequence they happen to be generated by TAB/JLOC. SEQ indicates results from the bandwidth minimizing algorithm included with recent versions of EAL. The cases designated RCM, MD, and ND refer to the RSEQ methods of Reverse-Cuthill-McKee, Minimum Degree, and Nested Dissection, respectively. Results from the automatic sequencer using the rational approach are denoted by "Auto" wheras "L.K." indicates results from sequencing by an experienced user.

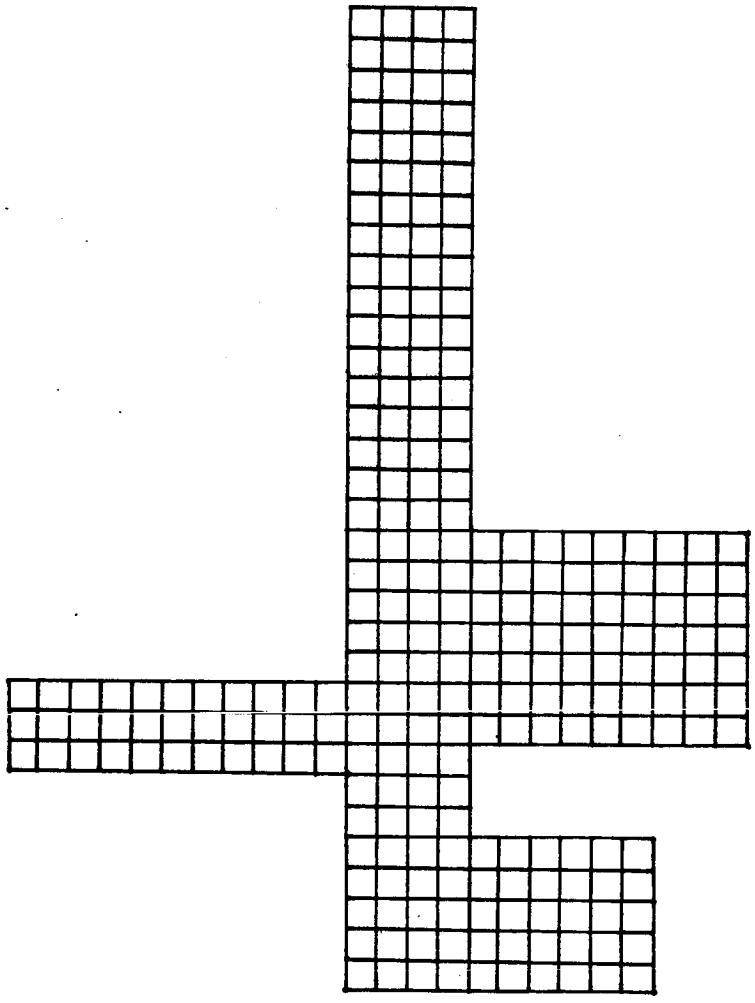
The first two columns of results contain the maximum and average interconnectivity encountered during the factorization process. Recall that interconnectivity generally increases as joints are eliminated; hence, lower numbers indicate less matrix fill-in. The last two columns are cost indices associated with computer storage and the number of operations required to factor the matrix and then solve the factored system, respectively. Note that the methods of MD, Auto, and L.K. show particularly better results than the others. Further, all four parameters are significantly lower for Auto and L.K. compared to MD. These encouraging results suggest that further pursuit of more general rational methods should continue.

Figure 9 contains the finite element mesh and summarized results for the other example studied. The results are quite similar with two apparent differences. First, no results were available for an experienced user. Also, the differences between the top three performers, Auto, MD, and RCM, are not as large as with the first example. Nevertheless, the trend is similarly encouraging.



Method	Interconnect Max	Interconnect Avg	IC1	IC2
none	24	15.2	31621	3161
SEQ	20	11.3	17178	2360
RCM	23	10.8	16205	2252
MD	19	8.5	9511	1766
ND	31	11.2	17502	2338
Auto	13	7.9	7690	1639
L.K.	12	7.8	7449	1617

Figure 8. Mesh layout and results for example 1.



Method	Interconnect Max	Interconnect Avg	IC1	IC2
none	41	22.8	119152	7223
SEQ	20	9.3	17550	2949
RCM	18	8.8	15006	2778
MD	16	8.3	13179	2626
ND	33	11.1	25953	3532
Auto	13	7.8	11428	2469

Figure 9. Mesh layout and results for example 2.

CONCLUSIONS AND RECOMMENDATIONS

The encouraging results from this preliminary study of automatic joint elimination sequencing methods for SPAR/EAL indicate that further investigation is warranted. Admittedly, the problems examined are limited in scope and complexity, but the potential benefits of a successful scheme seem substantial enough to spur further effort.

One promising method that was encountered in this study, though not implemented yet, is the one-way dissection method [7]. This method has found some acceptance and use in finite element analysis programs and may prove useful for SPAR/EAL as well. Other avenues that may prove worthy of pursuit include structural pattern recognition [8], computer graphics algorithms, and even perhaps expert system applications. With enough time and attention, the prospects for developing a useful automatic joint elimination sequencer appear fairly bright at this juncture.

REFERENCES

[1] Whetstone, W. D., SPAR Structural Analysis System Reference Manual, Volume 1, Program Execution, Engineering Information Systems, Incorporated: San Jose, California, 1978.

[2] Whetstone, W. D., EISI-EAL Engineering Analysis Language Reference Manual, Volume 2, Structural Analysis - Primary Processors, Engineering Information Systems, Incorporated: San Jose, California, 1983.

[3] Whetstone, W. D., "Computer Analysis of Large Linear Frames," Journal of the Structural Division, ASCE, Vol. 95, No. ST11, Proc. Paper 6897, November, 1969, pp. 2401-2417.

[4] Gibbs, N. E., Poole, Jr., W. G. and Stockmeyer, P. K., "An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix," SIAM Numer. Anal., Vol. 13, 1976, pp. 236-250.

[5] Pissanetsky, Sergio, Sparse Matrix Technology, Academic Press: Orlando, Florida, 1984.

[6] George, Alan, and Liu, Joseph W-H, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall: Englewood Cliffs, New Jersey, 1981.

[7] George, Alan, "An Automatic One-Way Dissection Algorithm for Irregular Finite Element Problems," SIAM J. Numer. Anal., Vol. 17, No. 6, December, 1980, pp. 740-751.

[8] Pavlidis, T., Structural Pattern Recognition, Springer-Verlag: Berlin, 1977.