INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 37, 1511–1530 (1994)

NODE AND ELEMENT RESEQUENCING USING THE LAPLACIAN OF A FINITE ELEMENT GRAPH: PART I—GENERAL CONCEPTS AND ALGORITHM

GLAUCIO H. PAULINO[*]

School of Civil and Environmental Engineering, Cornell University, Ithaca, NY, 14853, U.S.A.

IVAN F. M. MENEZES[*]

Department of Civil Engineering, PUC-Rio, Rua Marquês de São Vicente, 225, 22453, Rio de Janeiro, R.J., Brazil

MARCELO GATTASS[†]

Department of Computer Science, PUC-Rio, Rua Marquês de São Vicente, 225, 22453, Rio de Janeiro, R.J., Brazil

SUBRATA MUKHERJEE[‡]

Department of Theoretical and Applied Mechanics, Kimball Hall, Cornell University, Ithaca, NY, 14853, U.S.A.

SUMMARY

A Finite Element Graph (FEG) is defined here as a nodal graph (G), a dual graph (G^*), or a communication graph ($G^\bullet$) associated with a generic finite element mesh. The Laplacian matrix (($\mathbf{L}(G)$, $\mathbf{L}(G^*)$ or $\mathbf{L}(G^\bullet)$), used for the study of spectral properties of an FEG, is constructed from usual vertex and edge connectivities of a graph. An automatic algorithm, based on spectral properties of an FEG (G, G^* or $G^\bullet$), is proposed to reorder the nodes and/or elements of the associated finite element mesh. The new algorithm is called Spectral FEG Resequencing (SFR). This algorithm uses global information in the graph, it does not depend on a pseudoperipheral vertex in the resequencing process, and it does not use any kind of level structure of the graph. Moreover, the SFR algorithm is of special advantage in computing environments with vector and parallel processing capabilities.

Nodes or elements in the mesh can be reordered depending on the use of an adequate graph representation associated with the mesh. If G is used, then the nodes in the mesh are properly reordered for achieving profile and wavefront reduction of the finite element stiffness matrix. If either G^* or $G^\bullet$ is used, then the elements in the mesh are suitably reordered for a finite element frontal solver. A unified approach involving FEGs and finite element concepts is presented. Conclusions are inferred and possible extensions of this research are pointed out.

In Part II of this work,[1] the computational implementation of the SFR algorithm is described and several numerical examples are presented. The examples emphasize important theoretical, numerical and practical aspects of the new resequencing method.

1. INTRODUCTION

Heuristic node and element resequencing techniques in the Finite Element Method (FEM) have been a subject of research for a long time. Recognized books on finite elements, such as Desai and Abel,[2] Irons and Ahmad[3], Bathe,[4] Hughes[5] and Zienkiewicz and Taylor,[6] observe the importance of this subject. Duff[7] has surveyed sparse matrix research until 1976. Everstine[8] has

[*] Ph.D. Student
[†] Associate Professor
[‡] Professor

CCC 0029-5981/94/091511-20$9.00
© 1994 by John Wiley & Sons, Ltd.

Received 30 March 1993
Revised 2 June 1993

presented a review of the resequencing algorithms published until 1978. Chinn *et al.*[9] has written a survey paper emphasizing the bandwidth problem for graphs and matrices, and covering the literature until 1981. George[10] has reviewed resequencing of nodes and elements in the context of automatic mesh generation and finite elements.

Some recent papers have been published on resequencing techniques in the FEM. Shephard *et al.*[11] have presented a *node queue* algorithm which directly uses the mesh data structure (finite quadtree and finite octree mesh generators) to reorder nodes or elements in the mesh. The advantage of using the mesh data structure instead of the standard connectivity tables to obtain adjacency information is a substantial reduction of storage requirements. Sloan[12] has presented an algorithm and a FORTRAN program for profile and wavefront reduction of sparse symmetric matrices. Livesley and Sabin[13] have studied numbering procedures to reduce the bandwidth of grids derived from 2-D and 3-D finite element meshes of rectangular form. They have reported that the algorithm by Gibbs *et al.*[14] performs badly on some of these grids, particularly the ones that are approximately cubic in form. They have also presented a new algorithm which generates near-optimal numberings for this class of grids. Kaveh[15] has presented a connectivity co-ordinate system for node and element ordering. In Kaveh's paper, four algorithms are presented for selecting a good starting node for nodal numbering of a structure. J.-C. Luo[16] has presented an algorithm for reducing bandwidth and profile of a symmetric and sparse matrix. The idea of Luo's algorithm is to decompose a graph associated with the matrix into a group of isolated sets by general level structures. These are used to construct a maximal-depth partitioned structure, each level of which has as equal width as possible. Compared to Gibbs *et al.*[14] algorithm, Luo's algorithm is more complicated, and, in general, it produces smaller bandwidths but bigger profiles than the former algorithm. Koo and Lee[17] have developed a profile reduction algorithm based on the frontal ordering scheme and graph theory. They have developed a two-step approach where the finite elements are ordered first by the Cuthill and McKee[18] algorithm and then the nodes are reordered based on the concept of frontal ordering and the adjacency measure of graph theory. An earlier version of the two-step approach for finite element ordering has been presented by Fenves and Law.[19] Medeiros *et al.*[20] have just published an algorithm for profile and wavefront reduction which is based on some ideas of the Gibbs,[21] King[22] and Sloan[12] methods. They have shown that, in the numerical tests, their algorithm has a better performance than the reverse Cuthill–Mckee[23] Gibbs–King[24] and Sloan[12] algorithms.*

In general, the FEM leads to a linear system of coupled algebraic equations of the form

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

where $\mathbf{A} = [a_{ij}]_{N\times N}$ is a known sparse symmetric positive-definite matrix of order N, $\mathbf{b}$ is a known N-vector and the unknown N-vector $\mathbf{x}$ is sought. The system of equations (1) can be solved by Gauss elimination (see, for example, References 3, 4). To take advantage of the sparsity of the coefficient matrix $\mathbf{A}$ requires that the equations be organized in a special order, which depends on the solution scheme (bandwidth, profile or frontal) being utilized. The bandwidth and the profile methods construct the coefficient matrix explicitly, while the frontal methods alternates the assembly and elimination processes—the coefficient matrix is never assembled explicitly. In terms of the FEM, efficiency of a bandwidth or profile solver depends upon the ordering of the nodes, while efficiency of a frontal solver depends upon the ordering of the elements (equations are processed by element numbering). Therefore, in this paper, bandwidth and profile methods are classified as *node-based* ones, while a frontal method is classified as an *element-based* one.

*The last three references relate to the computational implementation of the respective algorithms

Nodal resequencing algorithms are designed to produce a permutation matrix $\mathbf{P} = [p_{ij}]_{N \times N}$ such that each row or column has only one non-null component which is equal to unity. The identity matrix ($\mathbf{I}$) is the simplest permutation matrix. Moreover $\mathbf{P}^{\mathrm{T}} = \mathbf{P}^{-1}$. The problem of reordering the system of linear equations (1) can be viewed as determining a permutation matrix that produces a convenient ordering of the system matrix $\mathbf{A}$ by the following transformation:

$$(\mathbf{PAP}^{\mathrm{T}})(\mathbf{Px}) = \mathbf{Pb} \tag{2}$$

where the orthogonality of the matrix $\mathbf{P}$ has been utilized. Since it is impractical to verify each of the $N!$ possible sequences associated with a given system matrix $\mathbf{A}$ (equation (2)), fast heuristic nodal resequencing algorithms for reducing bandwidth, profile or wavefront have been developed. Moreover, it has been shown that the general nature of optimal resequencing algorithms is NP complete.[25-27]

There is an intrinsic correspondence between a finite element stiffness matrix and the topology of the mesh. These topological properties can be explored by associating graphs with meshes and matrices.[23,28,29] Permutation of rows and columns in a matrix corresponds to relabelling the vertices of the associated graph or to resequencing the nodes of the mesh. This approach is suitable for studying nodal resequencing techniques.

Association of graphs with finite elements is also useful to provide an adequate element ordering for a frontal solution method.[30,3] A convenient approach consists of associating the elements in the original mesh to the vertices of the graph. Also, the connectivity among adjacent elements of the original mesh (common nodes, sides or faces among elements) is used to determine the edges of the graph. Relabelling the vertices of this graph is equivalent to resequencing the elements of the associated mesh. This approach is suitable for studying element resequencing techniques. In this case, the ordering of the nodal variables may be irrelevant.

In this paper, the terms *nodes* and *elements* relate to finite element meshes, and the terms *vertices* and *edges* relate to graphs. This terminology has been motivated in the previous paragraphs and it will be used from now on.

Consider a Finite Element Graph (FEG), i.e. a graph associated with a finite element mesh. A resequencing method, based on spectral properties of an FEG, is proposed here to renumber nodes and/or elements of a mesh. Moreover, numerical techniques for treating non-connected FEGs are also presented. The proposed method can handle large and *generic* (arbitrary) finite element meshes, i.e. meshes without any geometrical or topological regularity, with 1-D, 2-D and/or 3-D finite elements, with finite elements of various shapes or different interpolation orders.

The remaining sections of this paper are organized as follows. First, some graph-theoretical background is given in order to establish the basic concepts and definitions to the proposed method. Second, a preliminary discussion about the proposed spectral resequencing method is presented. Next, a connection between FEGs and finite element methodology is presented. The spectral resequencing method is then described and its relevant aspects are discussed. Afterwards, a special version of the subspace iteration method is presented. Finally, conclusions are inferred and recommendations for future research are pointed out.

2. GRAPH-THEORETICAL BACKGROUND

The basic graph theory background to this paper can be found in the classic book by Harary[31] or the more recent one by Buckley and Harary.[32] For further details about spectral techniques applied to graph theory, see the book by Cvetković *et al.*[33] Only the concepts and definitions of prime relevance to this paper are mentioned here.

Let $G = (V, E)$ be an *unidirected graph.* $V = \{v_1, v_2, \dots, v_n\}$ is a set of *vertices* with $|V| = n$, where $|\cdot|$ denotes the cardinality of the set. $E = \{e_1, e_2, \dots, e_m\}$ is a set of *edges* with $|E| = m$, and edges are unordered pairs of distinct vertices of V.

Given a graph G, properly *labelled*, there are several matrices which can be associated to G. Of special interest here are the adjacency (**A**), degree (**D**), vertex-edge incidence (**M**), and Laplacian (**L**) matrices. In particular, the spectral properties of the Laplacian matrix will be examined in detail.

The *adjacency matrix* $\mathbf{A}(G) = [a_{ij}]_{|V| \times |V|}$ of a labelled graph G is defined as

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is adjacent to } v_j, \text{ i.e. } \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The *degree matrix* $\mathbf{D}(G) = [d_{ij}]_{|V| \times |V|}$ is the diagonal matrix of vertex degrees

$$d_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

in which $\deg(v_i)$ is the number of edges incident with v_i or, equivalently,

$$\deg(v_i) = |\mathrm{Adj}(v_i)| \tag{5}$$

where $\mathrm{Adj}(v_i)$ is the set of vertices adjacent to v_i.

The *Laplacian matrix* $\mathbf{L}(G) = [l_{ij}]_{|V| \times |V|}$ is defined as

$$\mathbf{L}(G) = \mathbf{D}(G) - \mathbf{A}(G) \tag{6}$$

therefore, the components of $\mathbf{L}(G)$ are given by

$$l_{ij} = \begin{cases} -1 & \text{if } v_i \text{ is adjacent to } v_j, \text{ i.e. } \{v_i, v_j\} \in E \\ \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The *vertex-edge incidence matrix* $\mathbf{M}(G) = [m_{ij}]_{|V| \times |E|}$ is defined as follows. First, orient the edges of G arbitrarily. Then

$$m_{ij} = \begin{cases} 1 & \text{if } e_j \text{ points toward } v_i \\ -1 & \text{if } e_j \text{ points away from } v_i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Note that two distinct rows of **M** will have non-zero entries in the same column if and only if an edge joins the corresponding vertices. These entries will be 1 and -1. It can be shown that

$$\mathbf{L} = \mathbf{M}\mathbf{M}^{\mathrm{T}} \tag{9}$$

Also, one can easily verify that **L** is independent of the orientation of the edges in **M**.

The quadratic form associated with **L** can be obtained as

$$\mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\mathbf{M}\mathbf{M}^{\mathrm{T}}\mathbf{x} = (\mathbf{M}^{\mathrm{T}}\mathbf{x})^{\mathrm{T}}(\mathbf{M}^{\mathrm{T}}\mathbf{x}) = \sum_{\substack{\{v_i, v_j\} \in E \\ i, j \leqslant |V|}} (x_i - x_j)^2 \tag{10}$$

where x_i denotes the ith vector component. From equation (10), it follows that **L** is positive semi-definite. In summary, the Laplacian matrix $\mathbf{L}(G)$ is symmetric, singular (each of the row or column sums is zero), positive semi-definite, and it determines G up to *isomorphism.*†

The Laplacian matrix, associated with a graph $G(V, E)$, has interesting properties that are worth mentioning (for details, see References 34–37).

(a) If λ is an eigenvalue of $\mathbf{L}(G)$ then

$$0 \leqslant \lambda \leqslant |V| \tag{11}$$

(b) The multiplicity of the eigenvalue $\lambda = 0$ equals the number of *connected components* of G.
(c) If $\lambda = |V|$ is an eigenvalue of $\mathbf{L}(G)$ then G is connected.
(d) If G is connected, then $\lambda_2 > 0$, where λ_2 is the second smallest eigenvalue.
(e) λ_2 has an upper bound given by

$$\lambda_2 \leqslant \left(\frac{|V|}{|V|-1}\right) \min_i l_{ii}, \quad \text{(no sum)} \tag{12}$$

(f) Let $\rho(\mathbf{L}(G))$ denote the *spectral radius* of the Laplacian matrix $\mathbf{L}(G)$. Then

$$\rho(\mathbf{L}(G)) \leqslant \max_{i,j}(\deg(v_i) + \deg(v_j)) \tag{13}$$

considering all pairs of vertices $\{v_i, v_j\}$ joined by an edge.

(g) Let G be a connected graph. Then

$$\rho(\mathbf{L}(G)) \leqslant 2\max_i(\deg(v_i)), \quad 1 \leqslant i \leqslant |V| \tag{14}$$

Consider an undirected and connected graph G. Let the eigenvalues of $\mathbf{L}(G)$ be ordered such that

$$0 = \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n \tag{15}$$

The smallest eigenvalue is $\lambda_1 = 0$ and the associated eigenvector $\mathbf{y}_1$ has all its normalized components equal to 1. The special properties of the second smallest eigenvalue λ_2 and its corresponding eigenvector $\mathbf{y}_2$ have been studied by Fiedler.[34, 36] He designates λ_2 as the *algebraic connectivity of the graph* G which is related to usual vertex and edge connectivities of G. If the graph has a simple pattern, then there are analytical solutions for λ_2.[34, 37] For example, $\lambda_2 = 1$ for a star graph, $\lambda_2 = 2$ for a D-dimensional cube, and $\lambda_2 = 2[1 - \cos(2\pi/|V|)]$ for a circuit graph.

The components of $\mathbf{y}_2$ can be assigned to the vertices of G and can be considered as weights for them. Fiedler[36] designates this weighting process as the *characteristic valuation* of G. It is determined uniquely up to a non-zero factor if λ_2 is a simple eigenvalue of $\mathbf{L}(G)$, i.e. with multiplicity equal to 1.

3. SPECTRAL VERSUS PREVIOUS RESEQUENCING METHODS

In this paper, the method for resequencing nodes or elements is based on spectral properties of the Laplacian matrix of an appropriate FEG associated with a given finite element mesh. The name 'Laplacian matrix' comes from a discrete analogy with the Laplacian operator in numerical

† Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a one-to-one function f: $V_1 \rightarrow V_2$ that preserves adjacency, i.e., $E_2 = \{\{f(v_i), f(v_j)\} | \{v_i, v_j\} \in E_1\}$

analysis.[37] Consider, for example, the standard discrete five-point finite difference approximation for the Laplace equation on a rectangular grid subjected to Neumann boundary conditions; in this case the discrete Laplacian and the Laplacian matrices coincide. For further details on finite differences, see Forsythe and Wasow,[38] Section 20.7, or Lapidus and Pinder,[39] Section 5.2.1

The Spectral FEG Resequencing (SFR) method has some features that distinguish it from previous algorithms and are worth mentioning: (1) it uses *global* information in the graph; (2) it does not use the *pseudoperipheral vertex* concept; (3) it does not use any kind of *level structure* of the graph; (4) it is based on an *eigenanalysis* for determining the second eigenpair of the Laplacian matrix; and (5) its implementation is *simple* and based on a well-studied algebraic theory.

Previous reordering algorithms such as CM[‡] (Cuthill–McKee),[18] RCM (Reverse Cuthill–Mckee),[40] Levy,[41] GPS (Gibbs–Poole–Stockmeyer),[14] GK (Gibbs–King),[21, 22] Snay,[42] Sloan and Randolph,[43] Sloan[44, 12] and Medeiros *et al.*[20] make use of *local* information in the graph, namely information of neighbours of a vertex to compute the new ordering scheme. Conceptually, the SFR method differs from previous resequencing approaches in the sense that it finds a new ordering scheme based on *global* properties of the graph. Two other resequencing algorithms based on a global approach, but still using local information in the graph, have been proposed by Armstrong.[45, 46] He has used simulated annealing techniques to obtain the minimal or the near-minimal matrix bandwidth[45] or matrix profile and wavefront.[46]

Most reordering algorithms, based on graph-theoretical methods, use a *pseudo-peripheral vertex* as an appropriate starting vertex for numbering the graph (RCM,[§][23] GPS,[14] GK,[21, 22] etc.). Since the location of peripheral vertices in graphs is computationally expensive,[47, 48] several papers have been specifically written on efficient algorithms for finding pseudo-peripheral vertices, e.g. George and Liu,[49] Pachl[50] and Grimes *et al.*[51] Other reordering algorithms that use the pseudo-peripheral vertex concept are the refined quotient tree, the one-way dissection, and the nested dissection in the sparse matrix package SPARSPAK.[23] The SFR method does not need the step for finding a pseudoperipheral vertex in the graph. Moreover, although the SFR does not use the pseudoperipheral vertex, it naturally provides a pseudoperipheral vertex, which can be obtained as the vertex corresponding to the smallest (or largest) component in $\mathbf{y}_2$. There is no extra computation required to obtain this information!

Most reordering algorithms, based on graph-theoretical methods, use the concept of a *level structure* of the graph, e.g. References 12–17, 19–23, 41–44 and 52 among many others. The SFR method does not use any kind of level structure of the graph.

The major computational effort in the SFR method is the *eigenanalysis* where the second eigenvector ($\mathbf{y}_2$) of the Laplacian matrix has to be determined. In this paper, the eigenanalysis is performed by the subspace iteration method, but any equivalent algorithm can be used for this purpose. Therefore, most of the computation in the SFR is based upon standard matrix operations using floating point arithmetic. This computational aspect distinguishes the SFR from previous reordering algorithms based on integer arithmetic such as GPS,[14] GK,[21, 22] Sloan[44, 12] and Medeiros *et al.*[20]

The computational implementation of the SFR method is very *simple* and it is based on a well-studied algebraic theory.[33, 34, 37] Once a reliable eigensolver is available, the implementation procedure is straightforward. Moreover, nowadays, a finite element software usually has an eigensolver available as part of the finite element library. Also, there exist software libraries such

[‡] The abbreviations used in this paper have been named by the authors and may differ from other references. However, we have tried to use standard abbreviations from the technical literature

[§] Note that this implementation of the RCM algorithm [23] differes from the one cited in the previous paragraph [40] as IMSL (formerly International Mathematical and Statistical Libraries) and International Business Machines (IBM) Engineering and Scientific Subroutine Library (ESSL), which have eigensystem analysis subroutines available. Some numerical aspects and implementation issues related to the SFR will be discussed later in this paper.

4. FINITE ELEMENT MESH AND FINITE ELEMENT GRAPHS

The purpose of this section is to provide a practical and unified approach involving a finite element mesh, its associated graphs (FEGs), and corresponding matrices. The present approach has a much broader range of applications other than using it in resequencing methods, e.g. this approach can be advantageously used in conjunction with substructuring methods.[28] Resequencing of finite element meshes (nodes and/or elements) is one among many fields where graph theory has found several applications.[28, 53-56]

As mentioned in the introduction of this paper, the terms *nodes* and *elements* relate to a finite element mesh, and the terms *vertices* and *edges* relate to a graph, more specifically, an FEG. A finite element mesh is a collection of nodes and elements where adjacent elements are connected by common boundaries or nodes (note that a finite element may have nodes at the corners, along the sides, on the faces, or within the element itself). On the other hand, an FEG is the mathematical object that describes the connectivity among nodes (G) or elements (either G^* or $G^\bullet$) in a generic finite element mesh.

Consider, for example, the finite element mesh of Figure 1(a) with three-noded triangular elements (T3), four-noded quadrilateral elements (Q4), and no boundary conditions. The nodal connectivity of the mesh in Figure 1(a) can be represented by the nodal graph G shown in Figure 1(b). Consider now a labelling of the nodal graph G defined as the function $f\colon V(G) \to S$, where S is some domain of labels. Assume that $S = \{1, 2, \ldots, |V|\}$ and denote the graph labelled by S as $G^S = (V^S, E)$. The correspondence between the nodal graph G^S and an *associated matrix* can be established by a symmetric matrix $\mathbf{K}$ of order $|V|$ with non-null diagonal components k_{ii} (no sum). The *ordered* and *undirected* graph of $\mathbf{K}$ is denoted by $G^K = (V^K, E^K)$. In this case $\{\{v_i, v_j\} \in E^K \Leftrightarrow k_{ij} = k_{ji} \neq 0,\ i \neq j\}$, where v_i denotes the vertex of V^K with label i. This simple discussion implies that the configuration of the matrix $\mathbf{K}$ is topologically similar to the configuration of the finite element stiffness matrix. Therefore, each vertex of the nodal graph in Figure 1(b) is associated with a node in the finite element mesh of Figure 1(a) and with a row, column, equation or unknown of the system matrix. Each edge of the graph is associated with the conventional finite element connectivity table and with a non-zero entry in the matrix $\mathbf{K}$, excluding the diagonal entries k_{ii} (no sum), which are assumed to be non-zero.

The representation of element connectivity in a finite element mesh can also be expressed in terms of graphs. Two additional *element graphs*, associated with the original finite element mesh, will be presented here: the *dual graph* G^* and the *communication graph* $G^\bullet$. The dual graph representation of the connectivity among finite elements has been used by Bykat[57] and Fenves and Law[19] for element reordering. Both the dual and the communication graphs have been used by Venkatakrishnan *et al.*[58] in a parallel computing environment for domain decomposition of large 2-D meshes with triangular elements.

The geometric aspect of the *dual graph* is employed here for representing the *finite element connectivity graph* of a *generic* mesh. The vertices in the dual graph G^* represent finite elements in the original mesh. The edges in the dual graph represent adjacent finite elements that share a common boundary in the original mesh. In this definition, a finite element of dimension

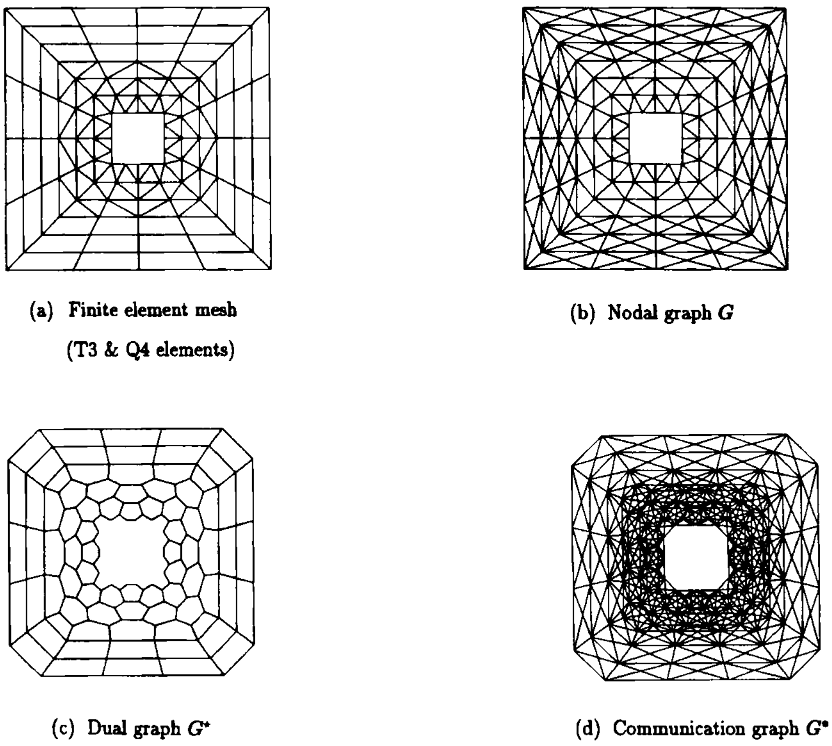


Figure 1. Finite element mesh and associated FEGs

$D(D = 1, 2$ or $3)$ has boundaries of dimension $(D - 1)$. Note that, whenever applicable, boundaries of dimension less than $(D - 1)$ are not considered. Application of the above definition to the finite element mesh of Figure 1(a) leads to the dual graph G^* of Figure 1(c). Note that for the 2-D continuum discretized by the finite element mesh of Figure 1(a), the planar elements are bounded and interconnected by their 1-D boundary segments.

The main advantage of the *dual graph* is its considerable economy in terms of data storage because the number of edges in the connectivity graph is generally reduced (cf. Figures 1(b) and 1(c)). When higher-order elements are used, the number of edges in G increases combinatorially with the number of nodes per element, while the number of edges in G^* remains unchanged for a given mesh discretization. For example, in Figure 1(a) suppose that instead of T3 and Q4, the elements are changed to T6 (six-noded triangular) and Q8 (eight-noded quadrilateral), respectively. In this case, the number of edges in G (Figure 1(b)) increases substantially, while the number of edges in G^* (Figure 1(c)) remains unchanged. A disadvantage of the dual graph approach for *generic* finite element meshes is that in some cases the connectivity of the finite elements cannot be completely represented. One case where this occurs is when D-dimensional finite elements are connected through boundaries of dimensions less than $(D - 1)$. Another case is when adjacent finite elements do not have the same geometric dimensions, e.g. a mesh with plane elements connected by beam elements. In these cases, non-connected dual graphs may be generated and each connected component may be treated separately.

Another convenient way of representing the element–element connectivity relationship in a *generic* finite element mesh is by means of a *communication graph.* The geometric aspect of the communication graph is also employed here as another tool to represent finite element connectivity. The vertices in the communication graph $G^{\bullet}$ represent finite elements in the original mesh (as before, in G^*). The edges in the communication graph represent adjacent finite elements that share a common node in the original mesh. Application of this definition to the finite element mesh of Figure 1(a) leads to the communication graph $G^{\bullet}$ of Figure 1(d). Note that the communication graph consists of *cliques* corresponding to the degree of each vertex. The communication graph overcomes the above-mentioned problems with the dual graph. However, the communication graph is dense (cf. Figures 1(c) and 1(d)) and makes the computation more intensive.

It is possible to define an element graph with intermediate requirements between G^* and $G^{\bullet}$. However, in the authors' point of view, the FEGs G^* and $G^{\bullet}$ are the most natural graphs for describing element connectivity in a generic finite element mesh. To the best of the authors' knowledge, the communication graph has not been explored yet in resequencing algorithms. In some situations, e.g. to avoid non-connected components in the graph or to improve the results, it may be advantageous to use the communication graph instead of the dual graph.

5. THE SPECTRAL FEG RESEQUENCING METHOD

Consider a consistent finite element mesh,[59] i.e. a mesh in which there is no inconsistency in the geometrical or topological model (e.g. repeated nodes or overlapped elements). Starting from a given consistent and generic finite element mesh, Figure 2 presents a diagram that summarizes the proposed spectral resequencing method. An important feature of this method is that ordering the nodal variables and ordering the finite elements can be separated into two completely independent tasks (Figure 2). Previous algorithms in the literature (e.g. References 17 and 19) do not reach this level of independence.

The method in Figure 2 enables one to perform a one-step or a two-step resequencing process. For simplicity, only the one-step resequencing process is illustrated by Figure 2. For a one-step process, there are two choices: (1) direct node resequencing (using G); or (2) direct element resequencing (using G^* or $G^{\bullet}$). For a two-step process, there are three choices: (1) direct node

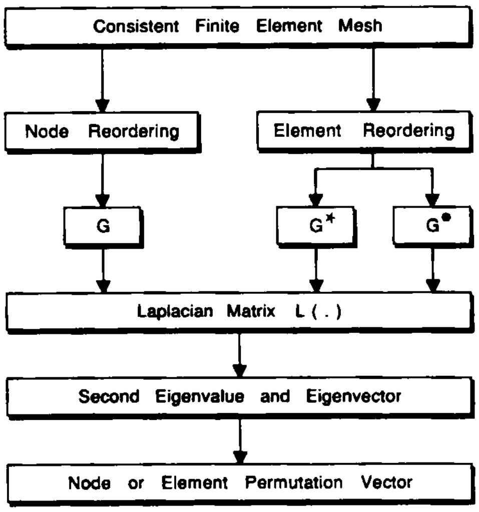

Consistent Finite Element Mesh
Node Reordering
Element Reordering
G
G*
G•
Laplacian Matrix L (.)
Second Eigenvalue and Eigenvector
Node or Element Permutation Vector

Figure 2. The spectral FEG resequencing method

resequencing–direct element resequencing; (2) direct node resequencing–indirect element resequencing; and (3) direct element resequencing–indirect node resequencing. Here the direct resequencing schemes are performed by the proposed SFR method (see Figure 2). The direct node resequencing–indirect element resequencing and the direct element resequencing are popular schemes used in conjunction with frontal solution algorithms. Comparing these two approaches, Duff *et al.*[60] have concluded that 'both approaches are practical and neither is consistently superior to the other'. Indirect element resequencing can be performed by numbering the elements in ascending order of their lowest numbered nodes.[43, 44, 60, 61] Analogously, indirect node resequencing can be performed by numbering the nodes in ascending order of their lowest numbered elements.

As shown in Figure 2, the resequencing method is based on spectral properties of the Laplacian matrix (**L**) of an FEG (G, G^* or $G^{\bullet}$) associated with a given finite element mesh. Note that the problem of resequencing the nodes or elements of a mesh is always converted to the problem of resequencing the vertices of the associated FEG. The method is based on the eigenvector of **L** corresponding to the algebraic connectivity of the graph (λ_2), i.e. the second eigenvector ($\mathbf{y}_2$). The components of this eigenvector provide a *characteristic valuation* of the vertices of the finite element graph. Differences in the values of the components of $\mathbf{y}_2$ provide topological distance information about the vertices of the graph. Therefore, the components of $\mathbf{y}_2$ can be taken as vertex-weights which can be used to reorder the vertices of the graph. The SFR algorithm, for connected graphs, is given in Table I. The case of non-connected graphs is considered in the next sub-section.

In the numerical solution of an eigenproblem, the eigenvectors are determined to within an arbitrary scale factor. A change in the sign of this scale factor leads to a reverse vertex ordering for the FEG. Since the SFR algorithm is a global one, the selection of the sign of this scale factor should not significantly affect the results obtained (e.g. profile). Here, the sign of this scale factor is arbitrarily selected.

With the rapid advance of computer technology, the generation of finite element meshes using interactive computer graphics is becoming a common practice.[28, 29] Paulino[28] has successfully developed a separate nodal reordering module in a preprocessor of space frames. Paulino and Gattass[29] have generalized this idea for *generic* finite element meshes. Therefore, the scheme proposed in Figure 2 and Table I could be advantageously implemented as a separated module in a finite element mesh generator or preprocessor.

5.1. Treatment of non-connected graphs

Three techniques for treating non-connected graphs are presented here. The first one is a common technique to treat non-connected graphs in resequencing algorithms. The second one is implicit in the SFR method. The last one is a possible sound alternative.

Table I. Spectral FEG Resequencing Algorithm

1. Construct a *finite element graph* (G, G^* or $G^{\bullet}$) corresponding to a given *finite element mesh*
2. Construct the respective *Laplacian Matrix* ($\mathbf{L}(G)$, $\mathbf{L}(G^*)$ or $\mathbf{L}(G^{\bullet})$) associated with the selected finite element graph
3. Compute the second eigenvalue (λ_2) of $\mathbf{L}(\cdot)$ and its corresponding eigenvector ($\mathbf{y}_2$)
4. Reorder the vertices in ascending order of the vector components in $\mathbf{y}_2$

The first alternative to treat non-connected graphs is given in Table II. It consists of adding an extra step at the end of Table I. Whenever the FEG associated to the problem is disconnected, the algorithm recognizes this case and repeats all the previous steps for each connected component of the FEG. This is a common technique to treat non-connected components in a graph. For instance, this strategy has been adopted by the GPS[14] and GK[21] algorithms. Other related algorithms that also use this strategy are the 'General Refined Quotient Tree' (GENRQT) and the 'General 1-Way Dissection' (GEN1D) in the sparse matrix package SPARSPAK [23].

The second alternative to treat non-connected graphs is given in Table III. It also consists of adding an extra step at the end of Table I. It represents a possible numerical technique for treating non-connected FEGs in the resequencing process. Preliminary numerical experiments, using the subspace iteration method to compute the second eigenpair of the Laplacian matrix, show that in most cases the repeated null eigenvalues (excluding $\lambda_1 = 0$) lead to eigenvector *approximations* which provide a sequential numbering for each connected component in the graph. In other words, each component is numbered separately, there is no mixture for the numbering among components. However, this technique is currently empirical and essentially based on numerical observations. Other numerical experiments also show that, in general, the use of the eigenvector associated with the first non-zero eigenvalue does not give good results for profile and wavefront reductions, and does not separate the components. These observations may have implications on algorithms for domain partitioning such as the ones proposed by Pothen *et al.*[62] and Hendrickson and Leland.[63] Both of these algorithms are also based on spectral graph theory.

Finally, a third alternative to treat non-connected graphs is presented. It consists of monitoring connectivity *a priori*, as explained in Table IV. This strategy has been used by Hendrickson and Leland.[63] However, it does not guarantee a sequential numbering of the components as is the case with the first technique.

Due to its effectiveness and simplicity, the first alternative to treat non-connected graphs (Table II) is recommended.

5.2. Example

Next, a simple example to clearly explain the SFR algorithm (Table I) is given. Consider the finite element mesh of Figure 3(a) with twelve nodes, four T3 elements, four Q4 elements, and no boundary conditions. Initially, the nodes and elements of the mesh in Figure 3(a) are arbitrarily

Table II. First alternative to treat non-connected graphs

Steps 1–4.	Same as Table I.
5.	Repeat the algorithm for each connected component

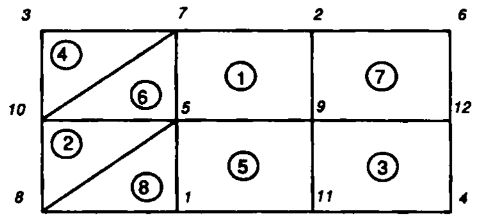
Table III. Second alternative to treat non-connected graphs

Steps 1–4.	Same as Table I.
5.	If there are two or more components in the graph, use the eigenvector associated with $\lambda_2 = 0$ which gives the smallest profile

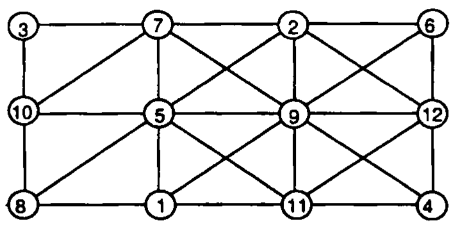
Table IV. Third alternative to treat non-connected graphs

If there are non-connected components in the graph, add a minimum number of *phantom edges* to establish connectivity, execute Steps 1–4 (Table I), and then remove the phantom edges

numbered. The connectivity of the nodes can be topologically represented by the labelled nodal graph G^A of Figure 3(b). Its associated Laplacian matrix $\mathbf{L}(G^A)$ is shown in Figure 3(c), for which $\lambda_2 = 1{\cdot}1071$ and $\mathbf{y}_2 = [-0{\cdot}0608, -0{\cdot}2023, 1{\cdot}0000, -0{\cdot}5303, 0{\cdot}0658, -0{\cdot}4721, 0{\cdot}3099, 0{\cdot}3106, -0{\cdot}2399, 0{\cdot}5829, -0{\cdot}3125, -0{\cdot}4514]^T$. The connectivity of the finite elements can be topologically represented by the labelled dual graph G^{*B} of Figure 3(d). In this figure, the dashed lines represent the original finite element mesh (Figure 3(a)). The associated Laplacin matrix $\mathbf{L}(G^{*B})$ is given in Figure 3(e), for which $\lambda_2 = 0{\cdot}5509$ and $\mathbf{y}_2 = [-0{\cdot}1806, 0{\cdot}2805, -0{\cdot}6151, 1{\cdot}0000, -0{\cdot}3423, 0{\cdot}4491, -0{\cdot}5491, -0{\cdot}0426]^T$. Another convenient way of representing the element connectivity is by means of the labelled communication graph $G^{\bullet C}$, as illustrated by Figure 3(f). In this figure, the dashed lines represent the original finite element mesh and the heavy lines represent its dual graph. Note that the dual graph G^{*B} is contained in the communication graph $G^{\bullet C}$. The associated Laplacian matrix $\mathbf{L}(G^{\bullet C})$ is given in Figure 3(g), for which $\lambda_2 = 1{\cdot}8395$ and $\mathbf{y}_2 = [0{\cdot}0000, 0{\cdot}4807, -1{\cdot}0000, 0{\cdot}8284, -0{\cdot}1605, 0{\cdot}4807, -1{\cdot}000, 0{\cdot}3707]^T$. In this case, the second and sixth components are equal, and the third and seventh components are equal. To order the approximated eigenvector components, ties are naturally broken by using more decimal digits in the numerical solution. This is the numerical tie-breaking strategy adopted for the SFR algorithm. Figure 3(h) shows the node and element numbering using G and G^*, respectively. Figure 3(i) shows the structure of the stiffness matrix assuming one degree of freedom per node, obtained from the relabelled nodal graph G. In Figure 3(i), the symbol '×' denotes a non-null matrix component. Comparing the structure of the matrices shown in Figures 3(c) and



(a) Finite element mesh
(arbitrary node and element numbering).

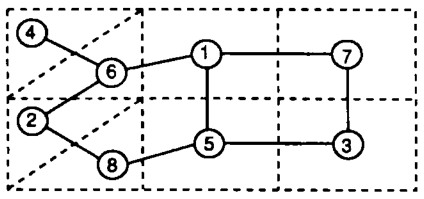


(b) Labeled FEG G^A

$$\begin{bmatrix}
4 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 & 0 & -1 & 0 \\
0 & 5 & 0 & 0 & -1 & -1 & -1 & 0 & -1 & 0 & 0 & -1 \\
0 & 0 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 \\
-1 & -1 & 0 & 0 & 7 & 0 & -1 & -1 & -1 & -1 & -1 & 0 \\
0 & -1 & 0 & 0 & 0 & 3 & 0 & 0 & -1 & 0 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 0 & 5 & 0 & -1 & -1 & 0 & 0 \\
-1 & 0 & 0 & 0 & -1 & 0 & 0 & 3 & 0 & -1 & 0 & 0 \\
-1 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 8 & 0 & -1 & -1 \\
0 & 0 & -1 & 0 & -1 & 0 & -1 & -1 & 0 & 4 & 0 & 0 \\
-1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & 5 & -1 \\
0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 5
\end{bmatrix}$$

(c) $L(G^A)$

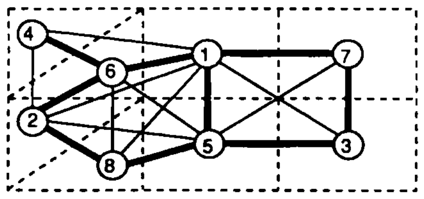
Figure 3. (a–c)



(d) Labeled FEG G^{*B}

$$\begin{bmatrix} 3 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \\ 0 & 2 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 2 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 & 3 & 0 & 0 & -1 \\ -1 & -1 & 0 & -1 & 0 & 3 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 2 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 2 \end{bmatrix}$$

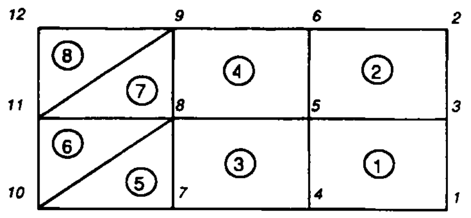
(e) $L(G^{*B})$



(f) Labeled FEG G^{*C}

$$\begin{bmatrix} 7 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 5 & 0 & -1 & -1 & -1 & 0 & -1 \\ -1 & 0 & 3 & 0 & -1 & 0 & -1 & 0 \\ -1 & -1 & 0 & 3 & 0 & -1 & 0 & 0 \\ -1 & -1 & -1 & 0 & 6 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 & 5 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 & 3 & 0 \\ -1 & -1 & 0 & 0 & -1 & -1 & 0 & 4 \end{bmatrix}$$

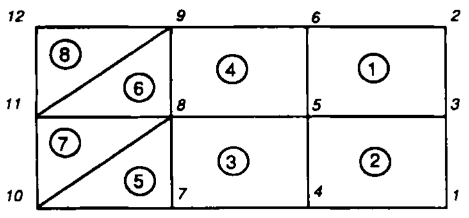
(g) $L(G^{*C})$



(h) Node and element numbering using G and G^*, respectively.

$$\begin{bmatrix} X & & X & X & X & & & & & & & \\ & X & X & & X & X & & & & & & \\ X & X & X & X & X & X & & & & & & \\ X & & X & X & X & & X & X & & & & \\ X & X & X & X & X & X & X & X & X & & & \\ & X & X & & X & X & & X & X & & & \\ & & & X & X & & X & X & & X & & \\ & & & X & X & X & X & X & X & X & X & \\ & & & & X & X & & X & X & & X & X \\ & & & & & & X & X & & X & X & \\ & & & & & & & X & X & X & X & X \\ & & & & & & & & X & & X & X \end{bmatrix}$$

(i) PAP^T



(j) Node and element numbering using G and G^*., respectively.

Figure 3. Example to illustrate the SFR algorithm

3(i), one can easily verify that the non-null matrix components in Figure 3(i) are more concentrated around the diagonal (bandedness property of the stiffiness matrix corresponding to an appropriate node numbering in the mesh) than the ones in Figure 3(c). Finally, Figure 3(j) shows the node and element numbering using G and $G^{\bullet}$, respectively. Note that the node numbering of Figures 3(h) and 3(j) is the same (both use G). However, the element numbering of Figure 3(h) (using G^{*}) is different from the one of Figure 3(j) (using $G^{\bullet}$).

6. NUMERICAL ASPECTS

The goal here is to address the main aspects of a numerical solution for the computational implementation of the procedure shown in Figure (2). The numerical scheme should be able to handle large and *generic* (arbitrary) finite element meshes. The prime interest is the accurate determination of the second eigenvalue at the left end of the spectrum of the Laplacian matrix **L** and its associated eigenvector. The method adopted here for the solution of this particular eigenproblem is the subspace iteration approach,[4, 5] which is widely used for large-scale finite element calculations. However, any other equivalent method can be used for this purpose.

Issues related to the computational efficiency of the SFR algorithm are relevant to this work. The following strategies are suggested for improved performance: (1) adequate convergence criterion and parameters, (2) preconditioning by preordering, and (3) consideration of other eigensolvers. The use of convergence criteria and convergence parameters will be discussed in Section 6.1. The other two issues are discussed below.

Numerically, the efficiency of the SFR is very sensitive to the initial ordering of the vertices in the graph. Preconditioning by preordering can be utilized for improving the running time of the original SFR algorithm. Furthermore, this technique can reduce both storage and number of arithmetic operations. The conventional resequencing strategy can be expressed as (see equation (2))

$$\mathbf{A} \xrightarrow{(t_1)} \mathbf{P}_{\mathrm{SFR}}\mathbf{A}\mathbf{P}_{\mathrm{SFR}}^{\mathrm{T}} = \bar{\mathbf{A}} \tag{16}$$

where t_1 is the Central Processing Unit (CPU) time consumed in the standard resequencing process, and P_{SFR} is the SFR ordering on the system matrix. The idea here is to efficiently pre-order the vertices of the graph before using the SFR algorithm. Note that, in terms of profile and wavefront results, the pre-ordering strategy does not make sense if an algorithm more efficient than the SFR is used as the pre-ordering method. Moreover, the running time of an effective pre-ordering algorithm must not be sensitive to the initial ordering. In this work, the profile ordering provided by the RCM algorithm[23] is used as the pre-ordering method. The pre-ordering strategy can be described as

$$\mathbf{A} \xrightarrow{(t_2)} \mathbf{P}_{\mathrm{RCM}}\mathbf{A}\mathbf{P}_{\mathrm{RCM}}^{\mathrm{T}} = \tilde{\mathbf{A}} \xrightarrow{(t_3)} \mathbf{P}_{\mathrm{SFR}}\tilde{\mathbf{A}}\mathbf{P}_{\mathrm{SFR}}^{\mathrm{T}} = \tilde{\tilde{\mathbf{A}}} \tag{17}$$

where t_2 is the CPU time for RCM algorithm, t_3 is the CPU time for the preconditioned SFR algorithm, and P_{RCM} is the RCM ordering on the system matrix. It is expected that

$$t_2 + t_3 \ll t_1 \tag{18}$$

In this sense, the RCM algorithm is a good choice in equation (17) because its simple and easily accessed data structure makes it a very fast algorithm.[23, 28] The pre-ordering strategy has been used by other researchers[45, 64, 65] for various purposes.

It is worth investigating the advantages and disadvantages of alternative eigensolvers to be used in conjunction with the SFR algorithm. Methods other than subspace iteration can be used to solve large eigenproblems. Examples are the determinant search[4] and Lanczos' method.[66, 5] Pothen *et al.*,[62] Simon[67] and Venkatakrishnan *et al.*[58] have used Lanczos' method to compute the second eigenpair $(\lambda_2, \mathbf{y}_2)$ of the Laplacian matrix associated with large finite element meshes. The application of interest in these papers was domain decomposition for parallel processing. A comparison between Lanczos and the subspace iteration methods has been presented by Nour-Omid *et al.*[68] This is a subject for future investigation.

Finally, it is worth mentioning that the SFR algorithm is of advantage in advanced computing environments with vector and parallel processing capabilities. The numerically intensive part of the SFR algorithm involves standard vector operations using floating point arithmetic. This is opposed to other resequencing algorithms which are based on integer arithmetic and the level structure concept. Moreover, the algebraic nature of the algorithm favours its implementation in a parallel computing environment. The investigation of the SFR algorithm as well as other numerical algorithms in advanced computing environments is also a subject for future investigation.

6.1. A special version of the subspace iteration method

The proposed SFR method (Figure 2 and Table I) requires only the second eigenvalue at the left end of the spectrum of $\mathbf{L}$ and its associated eigenvector. Therefore, a special version of the subspace iteration method is developed here and its main features are discussed below.

Consider the standard form of the eigenproblem

$$(\mathbf{L} - \lambda\mathbf{I})\mathbf{x} = 0 \tag{19}$$

where $\mathbf{x}$ are basis vectors. The numerical solution of equation (19) employs the subspace iteration method and QR iterations in the reduced space.[4] From practical experience, it is recommended that the order of the q-dimensional subspace should be calculated from[4]

$$q = \min(2p, p + 8, N) \tag{20}$$

where p is the number of desired eigenpairs and N is the order of the original space. Here, $N = |V|$, which is, in general, much larger than q. For a connected graph, the interest here is in the second eigenpair. In this case, $p = 2$. Therefore, from equation (20), $q = 4$ is the dimension that can be adopted for the reduced space. In general, the use of $q > 4$ leads to more accurate results but is more computationally expensive.

For the solution of the reduced eigenproblem, various methods can be utilized. In this work, the Householder–QR (H–QR) has been adopted.[4] The householder transformations reduce a matrix into Hessenberg form. In the case of a symmetric matrix, this form is simplified to a tridiagonal one. For the QR method, the orthogonal matrix $\mathbf{Q}$ is obtained here as a product of Jacobi rotation matrices, and $\mathbf{R}$ is an upper triangular matrix. The iterations proceed until diagonalization of the system matrix (see Reference 4 for details of the method). Since the order adopted for the reduced space is in general small (see previous paragraph), the stage corresponding to the Householder transformations has been omitted in the interest of computational efficiency. Therefore, the reduced eigenproblem is solved by QR iterations applied to matrices of order q.

In the present version of the subspace iteration method, the Laplacian matrix has been *shifted* according to,

$$\mathbf{L} \leftarrow \mathbf{L} + \alpha\mathbf{I} \tag{21}$$

where α is the shifting constant. Here $\alpha = 1$ has been chosen, such that all the eigenvalues (see equation (19)) become positive ($\lambda_j \geqslant 1{\cdot}0$; $1 \leqslant j \leqslant |V|$). This procedure does not change the eigenvectors of the Laplacian matrix and solves the problem of singularity in the computation of the eigenvector(s) corresponding to the null eigenvalue(s).

A crucial step of an iterative solution of the eigenproblem is to estimate the accuracy of the results. The solution is obtained once convergence within a prescribed tolerance has been obtained. The *convergence criterion* may be defined in terms of consecutive iterations.[5] Bathe[4] measures convergence by the relative error between successive eigenvalue approximations:

$$\frac{|\lambda_j^{i+1} - \lambda_j^i|}{\lambda_j^{i+1}} \leqslant \mathrm{TOL}, \quad 1 \leqslant j \leqslant q \tag{22}$$

where the subscript denotes the jth eigenvalue, the superscripts denote the iteration numbers, and TOL is a specified tolerance. Equation (22) is denoted here as the *eigenvalue convergence criterion.* The convergence of the eigensolution may also be measured by the absolute difference between successive normalized eigenvectors:

$$|\mathbf{y}_j^{i+1} - \mathbf{y}_j^i| \leqslant \mathrm{TOL}\,\mathbf{e}, \quad 1 \leqslant j \leqslant q \tag{23}$$

where $\mathbf{e} = [1, \ldots, 1]^{\mathrm{T}}$ is a unit vector of dimension $|V|$ and the first q eigenvectors are considered. Equation (23) is denoted here as the *eigenvector convergence criterion.* To ensure the best accuracy, each eigenvector approximation is normalized with respect to the absolute value of its largest component. The componentwise verification in equation (23) is a more severe condition than the one in equation (22). Moreover, the criterion in equation (22) is computationally more efficient (faster) than the one in equation (23). Although both criteria (equations (22) and (23)) are investigated in this work, the eigenvalue convergence criterion is adopted as default because of its efficiency and overall effectiveness.

The execution time and the quality of the results are affected by the value of the tolerance in equations (22) and (23). In the present work, $\mathrm{TOL} = 10^{-6}$ has been bound to be an adequate estimate for practical purposes. However, in many cases, an accurate solution of the eigenproblem may not be necessary and the tolerance could be bigger than this default value. The effect of the prescribed tolerance and maximum number of iterations, on the overall performance of the present version of the subspace iteration method, will be studied in detail in Part II of this work.[1] In general, the SFR is sensitive to the values assigned to the tolerance and the maximum number of iterations, specially for large meshes.

7. CONCLUDING REMARKS AND EXTENSIONS

This paper has presented a noteworthy application in the sense that an algebraic quantity such as an eigenvector of an FEG can be successfully used to reorder nodes and/or elements in a *generic* finite element mesh. The SFR is unlike any previous resequencing method in the literature.* The advantages of the SFR method presented herein are:

1. mathematical basis;
2. complete independence between node and element resequencing;
3. use of global information in the graph;
4. no need of a pseudo-peripheral vertex or the endpoints of a pseudo-diameter;
5. no need of any type of level structure of the FEG;
6. simple implementation.

* We have just become aware, at the proof-reading stage, of similar work by Barnard *et al.*[69] Their report was completed in October 1993.

Mathematical analysis of the SFR method is based on linear algebra, in contrast with purely heuristic resequencing algorithms, which have less theoretical basis. Moreover, the SFR captures essential features of modern resequencing algorithms, for example, the SFR provides a pseudo-peripheral vertex as a natural outcome.

The complete independence between numbering of nodal variables and numbering of finite elements in the SFR algorithm (see Figure 2) permits different resequencing algorithms to be suitably used for resequencing the nodal variables or the finite elements. Whether it is necessary to combine resequencing algorithms, or which combination of algorithms is the best for numbering nodes and elements of a generic finite element mesh, are problems that are being considered by the authors. Moreover, the problem of which algorithm and which element graph (G^* or $G^{\bullet}$) works best for numbering the finite elements also needs further investigation.[28, 60]

The main disadvantage of the SFR algorithm, when compared to other algorithms in a sequential computing environment, is the CPU time required to solve the particular eigenproblem, i.e. to compute the second eigenpair of the Laplacian matrix. Strategies have been presented for improved performance: use of an adequate convergence criterion and related parameters; preconditioning by preordering, and change of the eigensolver. However, the numerically intensive part of the SFR algorithm involves standard vector operations using floating point arithmetic. Therefore, the SFR algorithm is well-suited for computers with vector processors. Moreover, the algebraic nature of the algorithm favours its implementation in parallel computers. To this effect, at Cornell Theory Center, for example, we have available a computing environment that includes vector–scalar supercomputing resources, such as the IBM ES/9000-900 (with 6 vector units), and parallel systems, such as the IBM ES/9000-900 (with 6 CPUs), a 128-processor Kendall Square Research (KSR1) and a cluster of multiple workstations (IBM RS/6000) connected via high-speed networking.

A short-term direct extension of the present work includes vectorization of the SFR algorithm and evaluation of its performance using, for example, the IBM ES/9000-900 vector processing capabilities. On the other hand, a long-term research project involves coarse-grained parallelization of the SFR algorithm; at this stage, this algorithm should be coupled with linear and non-linear finite element analysis.[28, 29, 70, 71]

The use of an FEG such as G^* or $G^{\bullet}$ enables the finite elements to be renumbered even in the early stages of the mesh discretization process, independently of the types of elements to be adopted. An example of a problem that can benefit from the SFR element renumbering strategy is the p-convergence test through a sequence of successively refined meshes.[6] In this test, the number and disposition of finite elements remain unchanged, and the degree (p) of the highest complete piecewise polynomial approximation is progressively increased by adding nodes to elements until some desired level of accuracy is reached. If a frontal solution method is used in conjuction with this type of problem, the finite elements need to be renumbered only once.

The SFR algorithm has potential advantages for applications other than in the FEM, e.g. the Boundary Element Method (BEM). For instance, Kane *et al.*[72] have presented a solution strategy for large-scale multi-zone boundary element analysis. They have shown that the performance of the analysis is significantly affected by the zone numbering. Therefore, the SFR method could be of advantage to reorder nodes and elements in the mesh, and most importantly, the boundary element zones themselves. The representation of the zone connectivity in a boundary element mesh can be expressed in terms of a *zone-graph*.[¶] In this case, the vertices of the zone-graph represent zones in the original mesh. The edges in the zone-graph represent adjacent zones that share a common interface (node or element) in the original mesh. The zone-graph is a Boundary Element Graph (BEG),[¶]—note the parallel with an FEG, defined in Section 4. The advantage of

[¶]This term, to the best of the authors' knowledge, is being defined in this paper for the first time the SFR method becomes more apparent as the number of zones increases, i.e. for large-scale boundary element problems with many zones.

Further numerical exploration of spectral properties of matrices associated with a finite element graph (such as **L**, **A**, or their variants) might open new avenues and ideas in sparse matrix technology. Moreover, as motivated by previous discussions, such a study fits naturally in the emerging field of vector and parallel computing for engineering applications.

ACKNOWLEDGEMENTS

The first author acknowledges the financial support provided by the CNPq Brazilian Agency. Ivan F. M. Menezes acknowledges the financial support provided by the CAPES Brazilian Agency.

The authors are especially grateful to Catherine Mink, CADIF (Computer Aided Design Instructional Facility) Director, for giving credit to the ideas presented to her and for providing the resources necessary to carry out this research.

The authors acknowledge Dr. Bruce Hendrickson and Professor Stephen Vavasis for pointing out Reference 69.

The first author acknowledges Dr. Shang-Hsien Hsieh for very useful discussions about graph theory and finite elements, and Dr. Bruce Hendrickson and Liaqat Khan for their suggestions to this work.

The authors acknowledge Khalid Mosalam for carefully reading the manuscript, for his constructive criticism and valuable suggestions.

REFERENCES

1. G. H. Paulino, I. F. M. Menezes, M. Gattas and S. Mukherjee, 'Node and element resequencing using the Laplacian of a finite element graph—Part II: Implementation and numerical results', *Int. j. numer. methods eng.*, **37**, 1531–1555 (1994).
2. C. S. Desai and J. F. Abel, *Introduction to the Finite Element Method—A Numerical Method for Engineering Analysis*, Van Nostrand Reinhold, New York, 1972.
3. B. M. Irons and S. Ahmad, *Techniques of Finite Elements*, Ellis Horwood, Chichester, U.K., 1980.
4. K. J. Bathe, *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1982.
5. T. J. R. Hughes, *The Finite Element Method—Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1987.
6. O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method*, Vol. 1, 4th. edn, McGraw-Hill, London, 1989.
7. I. S. Duff, 'A survey of sparse matrix research', *Proc. IEEE*, **65**(4), 500–535 (1977), (cites 604 references).
8. G. C. Everstine, 'A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront', *Int. j. numer. methods eng.*, **14**, 837–853 (1979), (cites 65 references).
9. P. Z. Chinn, J. Chvátalová, A. K. Dewdney and N. E. Gibbs, 'The bandwidth problem for graphs and matrices—a survey', *J. Graph Theory*, **6**(3), 223–254 (1982), (cites 93 references).
10. P. L. George, *Automatic Mesh Generation—Application to Finite Element Methods*, Wiley, Masson, Paris, 1991.
11. M. S. Sephard, P. L. Baehmann and K. R. Griece, 'The versatility of automatic mesh generators based on tree structures and advanced geometric constructs', *Commun. Appl. Numer. Methods*, **4**, 379–392 (1988).
12. S. W. Sloan, 'A FORTRAN program for profile and wavefront reduction', *Int. j. numer. methods eng.*, **28**, 2651–2679 (1989).
13. R. K. Livesley and M. A. Sabin, 'Algorithms for numbering the nodes of finite-element meshes', *Comput. Systems Eng.*, **2**, 103–114 (1991).
14. N. E. Gibbs, N. G. Poole Jr. and P. K. Stockmeyer, 'An algorithm for reducing the bandwidth and profile of a sparse matrix', *SIAM J. Numer. Anal.*, **13**, 236–250 (1976).
15. A. Kaveh, 'A connectivity coordinate system for node and element ordering', *Comput. Struct.*, **41**, 1217–1223 (1991).
16. J.-C. Luo, 'Algorithms for reducing the bandwidth and profile of a sparse matrix', *Comput. Struct.*, **44**, 535–548 (1992).
17. B. U. Koo and B. C. Lee, 'An efficient profile reduction algorithm based on the frontal ordering scheme and the graph theory', *Comput. Struct.*, **44**, 1339–1347 (1992).
18. E. Cuthill and J. Mckee, 'Reducing the bandwidth of sparse symmetric matrices', *ACM Proc. 24th National Conference*, 1969, pp. 157–172.
19. S. J. Fenves and K. H. Law, 'A two-step approach to finite element ordering', *Int. j. numer. methods eng.*, **19**, 891–911 (1983).

20. S. R. P. Medeiros, P. M. Pimenta and P. Goldenberg, 'An algorithm for profile and wavefront reduction of sparse matrices with a symmetric structure', *Eng. Comput.*, **10,** 257–266 (1993).
21. N. E. Gibbs, 'A hybrid profile reduction algorithm', *ACM Trans. Math. Software,* **2,** 378–387 (1976).
22. I. P. King, 'An automatic reordering scheme for simultaneous equations derived from network systems', *Int. j. numer. methods eng.*, **2,** 523–533 (1970).
23. A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, N.J., 1981.
24. J. G. Lewis, 'Implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms', *ACM Trans. Math. Software,* **8,** 180–189 (1982).
25. Ch. H. Papadimitriou, 'The NP-completeness of the bandwidth minimization problem', *Comput.*, **16,** 263–270 (1976).
26. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, CA, 1979.
27. M. Yannakakis, 'Computing the minimum fill-in is NP-complete', *SIAM J. Alg. Disc. Meth.*, **2,** 77–79 (1981).
28. G. H. Paulino, 'Preprocessing of three-dimensional space frames, with nodal reordering, using interactive computer graphics,' (in Portuguese), *M.S. Thesis*, Department of Civil Engineering, PUC-Rio, Rio de Janeiro, Brazil, 1988.
29. G. H. Paulino and M. Gattass, 'Design of preprocessors and interactive graphical finite element programs', (in Portuguese), *Internal Report RI 03/89*, Department of Civil Engineering, PUC-Rio, Rio de Janeiro, Brazil, 1989.
30. B. M. Irons, 'A frontal solution program for finite element analysis', *Int. j. numer. methods eng.*, **2,** 5–32 (1970).
31. F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.
32. F. Buckley and F. Harary, *Distances in Graphs*, Addison-Wesley, Redwood City, CA, 1990.
33. D. M. Cvetković, M. Doob and H. Sachs, *Spectra of Graphs—Theory and Application*, Academic Press, New York, 1980.
34. M. Fiedler, 'Algebraic connectivity of graphs', *Czechoslovak Math. J.*, **23,** 298–305 (1973).
35. M. Fiedler, 'Eigenvectors of acyclic matrices', *Czechoslovak Math. J.*, **25,** 607–618 (1975).
36. M. Fiedler, 'A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory', *Czechoslovak Math. J.*, **25,** 619–633 (1975).
37. W. N. Anderson Jr. and T. D. Morley, 'Eigenvalues of the Laplacian of a graph', *Linear and Multilinear Algebra,* **18,** 141–145 (1985), (originally published as University of Maryland *Technical Report TR-71-45* October 1971).
38. G. I. Forsythe and W. R. Wasow, *Finite-Difference Methods for Partial Differential Equations*, Wiley, New York, 1960.
39. L. Lapidus and G. F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering*, Wiley, New York, 1982.
40. J. A. George, 'Computer implementation of the finite element method', *Ph.D. Thesis*, Computer Science, Stanford University, 1971.
41. R. Levy, 'Resequencing of the structural stiffness matrix to improve computational efficiency', *JPL Quart. Tech. Rev.*, **1,** 61–70 (1971).
42. R. A. Snay, 'Reducing the profile of sparse symmetric matrices', *Bull. Géodésique*, **50,** 341–352 (1976).
43. S. W. Sloan and M. F. Randolph, 'Automatic element reordering for finite element analysis with frontal solution schemes', *Int. j. numer. methods eng.*, **19,** 1153–1181 (1983).
44. S. W. Sloan, 'An algorithm for profile and wavefront reduction of sparse matrices', *Int. j. numer. methods eng.*, **23,** 239–251 (1986).
45. B. A. Armstrong, 'A hybrid algorithm for reducing matrix bandwidth', *Int. j. numer. methods eng.*, **20,** 1929–1940 (1984).
46. B. A. Armstrong, 'Near-minimal matrix profiles and wavefronts for testing nodal resequencing algorithms', *Int. j. numer. methods eng.*, **21,** 1785–1790 (1985).
47. W. F. Smith and W. M. L. Benzi, 'An algorithm for finding the diameter of a graph', *IFIP Congress 74*, North-Holland Pub. Co., **3,** 500–503 (1974).
48. K. S. Booth and R. J. Lipton, 'Computing extremal and approximate distances in graphs having unit cost edges', *Acta Inform.*, **15,** 319–328 (1981).
49. J. A. George and J. W.-H. Liu, 'An implementation of a pseudoperipheral node finder', *ACM Trans. Math. Software,* **5,** 284–295 (1979).
50. J. K. Pachl, 'Finding pseudoperipheral nodes in graphs', *J. Comput. System Sci.*, **29,** 48–53 (1984).
51. R. G. Grimes, D. J. Pierce and H. D. Simon, 'A new algorithm for finding a pseudoperipheral node in a graph', *SIAM J. Matrix Anal. Appl.*, **11,** 323–334 (1990).
52. M. Hoit and E. L. Wilson, 'An equation numbering algorithm based on a minimum front criteria', *Comput. Struct.*, **16,** 225–239 (1983).
53. Y. Hu, 'The application of graph theory to the limit analysis for torsion of thin-walled bars', *Int. j. numer. methods eng.*, **26,** 2765–2778.
54. A. Kaveh, 'Graphs and structures', *Comput. Struct.*, **40,** 893–901 (1991).
55. L. T. Souza and M. Gattass, 'A new scheme for mesh generation and mesh refinement using graph theory', *Comput. Struct.*, **46,** 1073–1084 (1993).
56. J. Nievergelt and K. H. Hinrichs, *Algorithms and Data Structures with Applications to Graphics and Geometry*, Prentice-Hall, Englewood Cliffs, N.J., 1993.
57. A. Bykat, 'A note on an element ordering scheme', *Int. j. numer. methods eng.*, **11,** 194–198 (1977).
58. V. Venkatakrishnan, H. D. Simon and T. J. Barth, 'A MIMD implementation of a parallel Euler solver for unstructured grids', *The J. Supercomputing*, **6,** 117–137 (1992).

59. M. Gattass, G. H. Paulino and J. C. Gortaire C., 'Geometrical and topological consistency in interactive graphical preprocessors of three-dimensional framed structures', *Comput. Struct.*, **46**, 99–124 (1993).
60. I. S. Duff, J. K. Reid and J. A. Scott, 'The use of profile reduction algorithms with a frontal code', *Int. j. numer. methods eng.*, **28**, 2555–2568 (1989).
61. A. Razzaque, 'Automatic reduction of frontwidth for finite element analysis', *Int. j. numer. methods eng.*, **15**, 1315–1324 (1980).
62. A. Pothen, H. D. Simon and K. P. Liou, 'Partitioning sparse matrices with eigenvectors of graphs', *SIAM J. Matrix Anal. and Appl.*, **II**, 430–452 (1990).
63 B. Hendrickson and R. Leland, 'Multidimensional spectral load balancing', *SANDIA Report SAND93-0074*, Category UC-405, Sandia National Laboratories, Albuquerque NM 87185, U.S.A., 1993.
64. I. Arany, L. Szoda and W. F. Smyth, 'An improved method for reducing the bandwidth of sparse symmetric matrices', *Proc. IFIP Congr.*, *71*, 1246–1250 (1972).
65. A. George and J. W.-H. Liu, 'The evolution of the minimum degree ordering algorithm', *SIAM Rev.*, **31**, 1–19 (1989).
66. C. Lanczos, 'An iteration method for the solution of the eigenvalue problem of linear differential and integral operators', *J. Res. National Bureau of Standards*, **45**, 255–281 (1950).
67. H. D. Simon, 'Partitioning of unstructured problems for parallel processing', *Comput. Systems Eng.*, **2**, 135–148 (1991).
68. B. Nour-Omid, B. N. Parlett and R. L. Taylor, 'Lanczos versus subspace iteration for solution of eigenvalue problems', *Int. j. numer. methods eng.*, **19**, 859–871 (1993).
69. S. T. Barnard, A. Pothen and H. D. Simon, 'A spectral algorithm for envelope reduction of sparse matrices', *Technical Report CS-93-49*, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, October 1993.
70. A. Conci and M. Gattass, 'Natural approach for geometric non-linear analysis of thin walled frames', *Int. j. numer. methods eng.*, **30**, 207–231 (1990).
71. V. L. Bergmann and S. Mukherjee, 'A hybrid strain finite element for plates and shells', *Int. j. numer. methods eng.*, **30**, 233–257 (1990).
72. J. H. Kane, B. L. K. Kumar and S. Saigal, 'An arbitrary condensing, noncondensing solution strategy for large scale multi-zone boundary element analysis', *Comput. Methods Appl. Mech. Eng.*, **79**, 219–244 (1990).