INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 37, 1531–1555 (1994)

NODE AND ELEMENT RESEQUENCING USING THE LAPLACIAN OF A FINITE ELEMENT GRAPH: PART II—IMPLEMENTATION AND NUMERICAL RESULTS

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, Brazil

MARCELO GATTASS†

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

SUBRATA MUKHERJEE‡

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

SUMMARY

In Part I of this work, Paulino *et al.*[1] have presented an algorithm for profile and wavefront reduction of large sparse matrices of symmetric configuration. This algorithm is based on spectral properties of a Finite Element Graph (FEG). An FEG has been defined as a nodal graph G, a dual graph G^* or a communication graph $G^\bullet$ associated with a generic finite element mesh. The novel algorithm has been called Spectral FEG Resequencing (SFR). This algorithm has specific features that distinguish it from previous algorithms. These features include (1) use of global information in the graph, (2) no need of a pseudoperipheral vertex or the endpoints of a pseudodiameter, and (3) no need of any type of level structure of the FEG. To validate this algorithm in a numerical sense, extensive computational testing on a variety of problems is presented here. This includes algorithmic performance evaluation using a library of benchmark test problems which contains both connected and non-connected graphs, study of the algebraic connectivity (λ_2) of an FEG, eigensolver convergence verification, running time performance evaluation and assessment of the algorithm on a set of practical finite element examples. It is shown that the SFR algorithm is effective in reordering nodes and/or elements of generic finite element meshes. Moreover, it computes orderings which compare favourably with the ones obtained by some previous algorithms that have been published in the technical literature.

1. INTRODUCTION

This is the second of two papers concerning node and element resequencing using the Laplacian matrix of graphs associated with finite element meshes. The main purpose of this paper is to validate, through numerical experiments, a novel algorithm for reducing matrix profile and wavefront, which has been presented in the first paper on this work.[1] This algorithm is based on spectral properties of a Finite Element Graph (FEG). An FEG has been defined[1] as a nodal graph (G), a dual graph (G^*) or a communication graph ($G^\bullet$) associated with a consistent[2] and generic

* Ph.D. Student
† Associate Professor
‡ Professor

CCC 0029–5981/94/091531–25$9.00

Received 30 March 1993
Revised 2 June 1993

finite element mesh. This novel algorithm has been named SFR, which stands for Spectral FEG Resequencing. Nodes or elements in the mesh can be reordered depending on the use of an appropriate graph representation associated with the mesh. If G is used, then the nodes in the mesh are reordered for achieving profile and wavefront reduction of the system matrix. If either G^* or $G^\bullet$ is used, then the elements in the mesh are properly reordered for a finite element frontal solver. All the conceptual aspects related to this algorithm have been discussed in detail in Reference 1.

In this paper, the terms *nodes* and *elements* relate to finite element meshes, and the terms *vertices* and *edges* relate to graphs, more specifically, FEGs. This terminology has been motivated in Part I of this work[1] and is adopted here.

Owing to the lack of sound theoretical methods for evaluating resequencing algorithms, empirical tests on a computer are generally used.[3-7] Here, besides evaluating the SFR algorithm on a set of benchmark test problems,[3] other specific numerical experiments are presented for a thorough evaluation of the SFR algorithm.

The remaining sections of this paper are organized as follows. First, some numerical aspects about the computational implementation of the SFR algorithm are presented. Second, the sparse matrix terminology is defined. Next, the numerical examples are presented and discussed. These examples include algorithmic performance evaluation using a library of benchmark problems with both connected and non-connected graphs, study of the algebraic connectivity (λ_2) of an FEG, convergence verification of the eigensolver used (a special version of the subspace iteration method), running time performance evaluation and assessment of the SFR algorithm on a set of five practical finite element meshes (lattice dome, L-shaped building, space station, cracked fuselage panel and gas turbine blade). The last section presents some conclusions.

2. COMPUTATIONAL IMPLEMENTATION

An efficient and robust computer program, using C language, has been developed. The program reads the conventional finite element input data, namely nodal co-ordinates and element connectivity. Based on the chosen FEG representation (G, G^* or $G^\bullet$), the Laplacian matrix $\mathbf{L}$ is assembled. Next, the first q (subspace order) eigenpairs of $\mathbf{L}$ are computed using the subspace iteration method. The vertices are then resequenced in increasing order of the components of the second eigenvector $\mathbf{y}_2$. Finally, the ordering for the vertices of the graph is associated with the nodes or elements of the corresponding finite element mesh.[1]

In general, the program uses *dynamic allocation of memory*, except for the reduced eigenproblem of dimension q, where *static* allocation has been used. The dynamic allocation of memory uses standard functions available in the C language.

Due to sparsity of the Laplacian matrix $\mathbf{L}$, the implementation provides the option of storing it in a *skyline* format. As a result, mathematical operations such as solution of a linear system of equations and standard matrix–vector products take storage in a vector form into account. For medium to large size meshes, the skyline format promotes excellent savings in both storage and execution time of the SFR algorithm. If the original mesh is extremely disordered, the savings obtained are not clear. However, for meshes generated by usual Finite Element Method (FEM) preprocessors (which give node and element numbering that are not completely arbitrary), savings in the Central Processing Unit (CPU) time of the order of 50 per cent have been noticed by using a skyline storage format when compared to a full storage format.

The linear system of equations in the subspace iteration algorithm is solved by a *modified* version of the *Crout method*, as reported in Reference 8. The ordering of the eigenvalues and the $\mathbf{y}_2$ eigenvector components is performed using the *quicksort* algorithm.[9]

Program SFR – Spectral FEG Resequencing	
Hardware	*HP apollo 9000 - Model 720*
Operating System	*HP-UX Release 8.07*
Computer language	*C*
Compiler	*cc for HP-UX Release 8.07*
# routines	*33*
# active lines of code	*974*
Source code size	*56241 Bytes*
Object code size	*37332 Bytes*
Executable code size	*61640 Bytes*

(a) General description of the SFR program.

Default Parameters		
< 1 >	Input file name...[.dat]:	*example*
< 2 >	Input data format:	*FEM*
< 3 >	Eigenproblem matrix:	*Laplacian*
< 4 >	Type of convergence:	*Eigenvalue*
< 5 >	Subspace order:	4
< 6 >	Maximum # Iterations for Subspace:	125
< 7 >	Maximum # Iterations for QR:	125
< 8 >	Tolerance for Subspace:	10^{-6}
< 9 >	Tolerance for QR:	10^{-6}
< 10 >	Shifting constant:	1.0
< 11 >	Number of Endpoints (NE) to be printed....:	1
< 12 >	Number of eigenpairs to be computed:	1
	Eigenpair(s): 2	

(b) Options available.

Figure 1. The SFR program

An overview of the SFR program is presented in Figure 1. A general description of the computational aspects of the SFR algorithm is given in Figure 1(a). In order to give a more objective idea about the implementation, the quantitative results reported in Figure 1(a) already assume the existence of an FEG (G, G^* or $G^\bullet$) together with its data structure representation. If the graph is non-connected, it can be identified *a priori* by the program.

The screen layout of the SFR program is given in Figure 1(b), which shows the options available and corresponding default parameters. The terminology used is in agreement with Part I of this work.[1] The resequencing method reported in Figure 1 can be readily implemented as a separated module in a finite element mesh generator or preprocessor.[1,5,10]

Next, a few practical comments about some of the options in Figure 1(b) are presented. The third option assumes the Laplacian matrix as a default. However, in terms of the eigensolution, the program is fairly general, and can handle matrices other than the Laplacian, e.g. the adjacency matrix.[1] The fourth option assumes the *eigenvalue convergence criterion*[1] as a default. However, the *eigenvector convergence criterion*[1] is also available. The fifth option assumes $q = 4$ as a default for the dimension of the reduced space, which is a reasonable estimate for most connected graphs. The sixth and seventh options allow one to specify the maximum number of iterations for the subspace and QR methods, respectively. For efficiency purposes, the default value for both these options has been set as 125. For accuracy purposes, these values can be replaced by a larger number. The eighth and ninth options allow one to specify the tolerance for both the subspace and QR methods. The default value for both these options is 10^{-6}. However, in many cases, an approximate solution of the eigenproblem is acceptable and the tolerance could be larger than this default value. The tenth option has been thoroughly discussed in Part I of this work.[1] The eleventh option allows one to choose the Number of Endpoints (NE) to be printed. These endpoints correspond to the first and last NE resequenced vertices. This option is specially useful to confirm the fact that the smallest (or largest) component in $\mathbf{y}_2$ corresponds to a pseudoperipheral vertex in the graph. Note that there is no extra computation required to obtain this information! The twelfth option allows one to print the first eigenpairs of the system matrix.

In the case of the Laplacian matrix, the output starts from the second eigenpair because, as mentioned previously,[1] the first eigenpair is (0, e). Here, $\mathbf{e} = [1, \ldots, 1]^T$ is a unit vector of dimension $|V|$, where $|\cdot|$ denotes the cardinality of the set and V is the set of vertices of the associated graph.

Unless otherwise stated, the default parameters of Figure 1(b) are used for the examples in this paper.

3. SPARSE MATRIX TERMINOLOGY

The basic sparse matrix terminology is being defined at this stage of the work because the SFR algorithm does *not* depend on the quantities defined next. These quantities are used for the evaluation of resequencing algorithms and also for comparison among algorithms.

Given a sparse matrix **A** of order N, the matrix *bandwidth* (B) is defined as

$$B = \max b_i, \quad i \leqslant N \tag{1}$$

where b_i is the 'ith row bandwidth', i.e. the number of columns from the first non-zero component in the row to the diagonal, inclusive.*

The matrix *profile* (P) is defined as

$$P = \sum_{i=1}^{N} b_i \tag{2}$$

The matrix *wavefront* (W) or *maximum wavefront* is defined as

$$W = \max c_i, \quad i \leqslant N \tag{3}$$

where c_i is the 'ith row wavefront', i.e. the number of active columns for row i. A column j is active in row i if $j \geqslant i$ and there is a non-zero component in column j with a row index k satisfying $k \leqslant i$.

Following Everstine,[3] the matrix *average wavefront* ($\bar{W}$) is defined as

$$\bar{W} = \frac{1}{N} \sum_{i=1}^{N} c_i = \frac{P}{N} \tag{4}$$

From the symmetric structure of **A**, it follows that $\sum_{i=1}^{N} c_i = \sum_{i=1}^{N} b_i$, although, in general, $c_i \neq b_i$.

The matrix *root-mean-square* (r.m.s.) *wavefront* ($\hat{W}$) is defined as

$$\hat{W} = \left(\frac{1}{N} \sum_{i=1}^{N} c_i^2 \right)^{1/2} \tag{5}$$

From the above definitions, it follows that

$$\bar{W} \leqslant \hat{W} \leqslant W \leqslant B \leqslant N \tag{6}$$

The matrix density (g) is given as

$$g = \frac{2|E| + N}{N^2} 100 \text{ per cent} \tag{7}$$

where E is the set of edges in the graph.

* Note that this definition of bandwidth includes the diagonal components

To illustrate the above definitions, Figure 2(a) shows a gable frame with five nodes, four beam elements (1-D) and no boundary conditions. In this example, where only 1-D finite elements have been used in Figure 2(a), the topology of the finite element model and its nodal graph representation coincide. Another possible *isomorphic* nodal graph representation $G^K(V^K, E^K)$ for the model given in Figure 2(a) is shown in Figure 2(b), where $V^K = \{1, \ldots, 5\}$ and $E^K = \{\{1, 4\}, \{4, 3\}, \{3, 5\}, \{5, 2\}\}$. For the sake of simplicity, assume one degree of freedom (d.o.f.) per node in Figure 2(a). The stiffness matrix representation associated with this mesh is given in Figure 2(c) by the symmetric matrix $\mathbf{K}$ (of order 5) with components k_{ij} and density $g = 52$ per cent. In the matrix of Figure 2(c), the boxes denote active columns. Finally, using equations (1)–(5), one verifies that $B = 4$, $P = 11$, $W = 3$, $\bar{W} = 2{\cdot}2000$ and $\hat{W} = \sqrt{5{\cdot}4} \approx 2{\cdot}3238$, which agree with the inequalities in equation (6).

4. EXAMPLES AND DISCUSSIONS

To assess the effectiveness of the SFR algorithm, several important examples are presented and their results are discussed. These examples include algorithm performance evaluation using a library of benchmark test problems, study of the algebraic connectivity (λ_2) of an FEG, convergence verification, running time performance evaluation and assessment of the algorithm on a set of practical finite element test problems. Whenever feasible, the results obtained by the SFR algorithm are compared with other algorithms.

In most examples that follow, the proposed SFR algorithm is compared with the GPS[11] and GK[12,13] algorithms. These algorithms have been selected for comparison because they are widely used and are well established in the technical literature. For example, the GPS algorithm has been used by Everstine[3] and Araújo Filho;[14] the GK algorithm has been used by Sloan[15,16] and Medeiros *et al.*;[7] and both the GPS and GK algorithms have been used by Armstrong,[17] Paulino[5] and Koo and Lee.[18]

4.1. Benchmark test problems

Everstine[3] has presented a collection of 30 finite element meshes for testing nodal resequencing algorithms. The meshes range in size from 59 to 2680 nodes. A complete description of the test problems, together with plots of the corresponding meshes, can be found in Everstine's[3] paper. Another library of general test problems has been presented by Duff *et al.*[19]

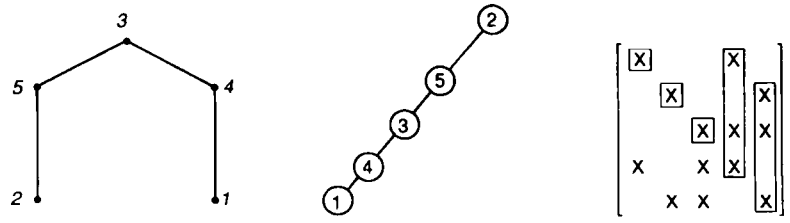


Figure 2. Example to illustrate sparse matrix terminology

Everstine's[3] test problems use the graph adjacency data structure as input data. In the present study, the number of iterations for both the subspace and QR methods is unlimited in a numerical sense, i.e. the maximum number of iterations has been chosen to be a very large number (here, 12 500 has been used).

For the numerical examples, Everstine's[3] test problems are separated into two groups. First, the SFR is tested using the meshes for which the associated nodal graph G is connected ($\lambda_2 \neq 0{\cdot}0$), i.e. 24 examples. Next, the other six examples, associated with non-connected graphs ($\lambda_2 = 0{\cdot}0$), are considered. This sequence of presentation is consistent with Part I of this work.[1]

4.1.1. Connected graphs. Connected graphs represent an important class of graphs because, for many practical cases in finite element analysis, the stiffness matrix is associated with a connected mesh (or graph). Table I lists some initial data and the results obtained from the SFR algorithm on Everstine's[3] benchmark test problems for which the associated graphs are connected. The eigenvalue convergence criterion is used for all these cases. The initial data are the number of nodes (# Nodes), $|E|$, and matrix densities g (per cent). The SFR results are the profiles (P), average ($\bar{W}$), r.m.s. ($\hat{W}$), maximum wavefronts (W), bandwidths (B), algebraic connectivities (λ_2) of the associated graph G, and number of iterations (# ITER) in the subspace iteration method.

In Table I, if the eigenvector convergence criterion is used, results similar to P, $\bar{W}$, $\hat{W}$, W, B and λ_2 are obtained. However, the # ITER is always larger than the ones listed in Table I. For the example with # Nodes = 162, a problem in the numerical solution has been noticed in the sense

Table I. Results produced by the SFR algorithm

Problem		Density	Profile	Wavefront			Bandwidth		
# Nodes = $\|V\|$	$\|E\|$	g(%)	P	$\bar{W}$	$\hat{W}$	W	B	λ_2	# ITER
59	104	7·67	290	4·91	5·06	7	11	0·0666	264
66	127	7·35	194	2·94	2·95	3	4	0·0115	75
72	75	4·28	255	3·54	3·68	6	13	0·0215	737
87	227	7·15	604	6·94	7·31	11	20	0·0969	68
162	510	4·50	1564	9·65	10·07	16	30	0·0577	43
193	1650	9·38	5149	26·68	27·62	39	46	0·8147	72
209	767	3·99	3320	15·88	16·59	27	53	0·1211	127
221	704	3·34	2028	9·18	9·53	14	24	0·0256	57
245	608	2·43	2957	12·07	13·10	23	94	0·0354	171
307	1108	2·68	7742	25·22	25·91	32	65	0·1605	47
310	1069	2·55	3100	10·00	10·14	13	19	0·0164	40
361	1296	2·27	5339	14·79	15·02	17	25	0·0358	109
419	1572	2·03	8908	21·26	22·83	40	66	0·0362	119
503	2762	2·38	14 672	29·17	30·98	46	92	0·1001	108
592	2256	1·46	10 379	17·53	18·69	36	131	0·0201	124
758	2618	1·04	7573	9·99	10·77	22	41	0·0025	253
869	3208	0·96	15 750	18·12	20·03	39	135	0·0078	317
878	3285	0·97	20 529	23·38	23·87	32	95	0·0147	370
918	3233	0·88	18 246	19·88	20·53	33	81	0·0085	209
992	7876	1·70	35 716	36·00	36·33	42	62	0·0590	137
1005	3808	0·85	34 784	34·61	35·93	54	152	0·0304	210
1007	3784	0·85	21 450	21·30	21·67	29	73	0·0104	294
1242	4592	0·68	43 025	34·64	35·94	59	139	0·0159	160
2680	11 173	0·35	92 498	34·51	35·20	51	161	0·0046	204

that the prescribed maximum number of iterations is reached even if it is a very large number (here, 12 500 has been used). If the eigenvalue convergence criterion is used, this problem does not happen, as can be verified in Table I.

Table II lists the profiles (P) produced by the SFR algorithm, together with the profiles produced by Lewis'[4] implementation of the GPS[11] and GK[12,13] algorithms, and the Simulated Annealing-profile and wavefront (SApw)[17] algorithm.

The algorithm SApw[17] uses simulated annealing techniques for reducing the matrix profile and wavefront. According to Armstrong,[17] his algorithm leads to minimal or near-minimal matrix profile and wavefront, but is too slow for general use. There is empirical evidence that the SApw always leads to profiles that are less than or equal to other heuristic algorithms, such as RCM,[20] GPS,[11] GK,[4,12,13] and those by Levy[21] and Sloan.[16] Therefore, the SApw algorithm is useful for evaluating and comparing the performance of faster but more approximate resequencing algorithms.

From Table II, it is clear that the GPS algorithm gives poorer results for profile than the SFR and GK algorithms. This is expected because the GPS algorithm was designed primarily to reduce bandwidth. On average, the SFR, GPS and GK algorithms give profiles which are 14, 27 and 24 per cent, respectively, in excess of the SApw profile. The *worst case* profiles for the SFR, GPS and GK algorithms are 37, 94 and 78 per cent, respectively, in excess of the SApw profile.

Table II. Profiles of SFR, SApw, GPS and GK algorithms

Problem	Actual P					Normalized P (or $\bar{W}$)		
# Nodes	Initial	SFR	SApw	GPS	GK	SFR / SApw	GPS / SApw	GK / SApw
59	464	290	273	342	314	1·062	1·253	1·150
66	640	194	193	193	193	1·005	1·000	1·000
72	244	255*	219	339*	327*	1·164	1·548	1·493
87	2336	604	515	729	789	1·173	1·416	1·532
162	2806	1564	1272	1662	1579	1·230	1·307	1·241
193	7953	5149	4409	5013	4609	1·168	1·137	1·045
209	9712	3320	2693	4749	4434	1·233	1·763	1·646
221	10 131	2028	1848	2266	2223	1·097	1·226	1·203
245	4179	2957	2161	4191	3813	1·368	1·939	1·764
307	8132	7742	6535	8541*	8221*	1·185	1·307	1·258
310	3006	3100*	2940	3036*	3007*	1·054	1·033	1·023
361	5445	5339	4992	5060	5060	1·069	1·014	1·014
419	40 145	8908	6512	8960	8073	1·368	1·376	1·240
503	36 417	14 672	11 958	15 049	15 042	1·227	1·258	1·258
592	29 397	10 379	9417	11 317	10 925	1·102	1·202	1·160
758	23 871	7573	7123	8223	8175	1·063	1·154	1·148
869	20 397	15 750	13 207	16 370	15 728	1·192	1·239	1·191
878	26 933	20 529	17 835	19 955	19 696	1·151	1·119	1·104
918	109 273	18 246	15 949	21 287	20 498	1·144	1·335	1·285
992	263 298	35 716	32 528	34 068	34 068	1·098	1·047	1·047
1005	122 075	34 784	32 513	43 142	40 141	1·070	1·327	1·235
1007	26 793	21 450	19 913	22 708	22 429	1·077	1·140	1·126
1242	111 430	43 025	33 098	55 738	58 864	1·300	1·684	1·778
2680	590 543	92 498	84 900	104 131	99 426	1·089	1·227	1·171

*No improvement on original ordering

Aside from the results by the SApw (which always gives the best profiles), the SFR algorithm gives the lowest profiles on 16 occasions while the GK algorithm gives the lowest profiles on eight occasions. Note that when there are no improvements from the SFR algorithm (# Nodes = 72 and 310), there are also no improvements from either the GPS or GK algorithms. Overall, for simple regular grids, e.g. # Nodes = 66 and 992, the profiles produced by the GPS and GK algorithms are lower than those produced by the SFR algorithm. In contrast, for complicated 3-D meshes, e.g. # Nodes = 209, 245 and 1242, the profiles produced by the SFR algorithm are much lower than those produced by the GPS and GK algorithms.

Since the average wavefront is proportional to the profile (see equation (4)), the relative profiles in the last four columns of Table II are the same as for normalized average wavefronts. For instance,

$$\frac{P_{\mathrm{SFR}}}{P_{\mathrm{SApw}}} = \frac{\bar{W}_{\mathrm{SFR}}}{\bar{W}_{\mathrm{SApw}}} \tag{8}$$

Table III lists the r.m.s. wavefronts ($\hat{W}$) produced by the SFR, SApw,[17] GPS[4] and GK[4] algorithms. One can verify that the r.m.s. wavefront reductions follow similar trends as the profile results shown in Table II. On average, the SFR, GPS and GK algorithms give r.m.s. wavefronts which are 18, 31 and 27 per cent, respectively, in excess of the SApw r.m.s. wavefront. The worst case r.m.s. wavefronts for the SFR, GPS and GK algorithms are 44, 99 and 90 per cent, respectively, in excess of the SApw r.m.s. wavefront. Aside from the results by the SApw (which

Table III. R.m.s. wavefronts of SFR, SApw, GPS and GK algorithms

Problem # Nodes	Initial	SFR	SApw	GPS	GK
59	8·22	5·06	4·74	6·03	5·53
66	11·01	2·95	2·94	2·94	2·94
72	3·46	3·68*	3·12	4·89*	4·71*
87	29·38	7·31	6·16	8·95	9·79
162	18·96	10·07	7·97	10·63	10·09
193	43·84	27·62	23·70	27·06	24·86
209	50·32	16·59	13·33	24·49	22·63
221	50·39	9·53	8·64	10·78	10·53
245	18·48	13·10	9·23	18·38	16·64
307	27·36	25·91	22·33	29·37*	28·03*
310	9·85	10·14*	9·62	9·96*	9·85*
361	15·38	15·02	14·08	14·23	14·23
419	107·07	22·83	15·96	22·19	19·96
503	78·60	30·98	24·93	32·13	32·22
592	55·18	18·69	16·65	20·52	19·70
758	37·95	10·77	10·05	12·07	12·01
869	25·02	20·03	15·63	20·69	19·87
878	31·92	23·87	20·98	22·90	22·60
918	131·14	20·53	18·02	24·31	23·39
992	301·99	36·33	33·51	34·66	34·66
1005	137·66	35·93	33·79	49·34	44·80
1007	26·93	21·67	20·26	22·90	22·60
1242	105·20	35·94	27·30	48·62	51·78
2680	234·42	35·20	32·27	39·91	38·03

*No improvement on original ordering

always gives the best r.m.s. wavefronts), the SFR algorithm gives the lowest $\hat{W}$ on 15 occasions while the GK algorithm gives the lowest $\hat{W}$ on nine occasions. Similar to the results for profile, for the cases where # Nodes = 72 and 310, there are no improvements in the r.m.s. wavefront by any of the algorithms (SFR, GPS or GK).

For completeness, Table IV lists the maximum wavefronts (W) produced by the SFR, SApw,[17] GPS[4] and GK[4] algorithms for Everstine's[3] benchmark problems. On average, the SFR, GPS and GK algorithms give maximum wavefronts which are 26, 41 and 38 per cent, respectively, in excess of the SApw maximum wavefront. The *worst case* maximum wavefronts for the SFR, GPS and GK algorithms are 90, 111 and 123 per cent, respectively, in excess of the SApw maximum wavefront. Note that the results by the SApw are not always the lowest ones in Table IV, e.g. # Nodes = 307, 361, 878 and 992. Comparing the SFR and GK algorithms, one verifies that the SFR algorithm gives the lowest W on 14 occasions, the GK algorithm gives the lowest W on eight occasions, and the SFR and GK algorithms tie on two occasions.

Finally, Table V lists the bandwidths produced from the SFR algorithm, together with the bandwidths produced by the GPS,[4] GK[4] and the Simulated Annealing-bandwidth (SAb).[22] The algorithm SAb[22] uses a simulated annealing technique for reducing matrix bandwidth. The results reported in Table V use the Node-Shuffling Algorithm Long (NSAL) strategy for producing minimal or near-minimal bandwidths.

On most occasions, the SFR algorithm gives better results for profile, r.m.s. wavefront and maximum wavefront than those obtained by the GPS and GK algorithms (see Tables II–IV).

Table IV. Maximum wavefronts of SFR, SApw, GPS and GK algorithms

Problem # Nodes	Initial	SFR	SApw	GPS	GK
59	11	7	6	8	8
66	21	3	3	3	3
72	4	6*	4	7*	7*
87	43	11	9	13	17
162	33	16	11	14	13
193	62	39	31	38	36
209	71	27	20	40	35
221	77	14	12	17	17
245	30	23	13	29	27
307	35	32	33	43*	37*
310	16	13	12	14	13
361	25	17	18	15	15
419	172	40	21	33	30
503	126	46	31	50	50
592	88	36	28	34	33
758	61	22	21	25	25
869	41	39	27	38	37
878	40	32	30	26	25
918	194	33	29	40	39
992	514	42	46	36	36
1005	228	54	46	97	89
1007	32	29	28	33*	33*
1242	193	59	43	85	97
2680	362	51	44	61	60

*No improvement on original ordering

Table V. Bandwidths of SFR, SAb, GPS and GK algorithms

Problem # Nodes	Initial	SFR	SAb	GPS	GK
59	26	11	7	9	12
66	45	4	4	4	4
72	13	13	7	7	9
87	64	20	12	20	21
162	157	30	14	14	18
193	63	46	36	43	46
209	185	53	24	43	49
221	188	24	14	19	21
245	116	94	23	40	49
307	64	65*	37	44	64*
310	29	19	13	15	22
361	51	25	15	15	15
419	357	66	28	34	42
503	453	92	49	57	70
592	260	131	33	37	48
758	201	41	21	26	26
869	587	135	38	39	63
878	520	95	26	28	41
918	840	81	36	50	65
992	514	62	36	36	36
1005	852	152	72	107	136
1007	987	73	30	35	55
1242	937	139	61	100	130
2680	2500	161	58	69	90

*No improvement on original ordering

This situation is the opposite in the case of bandwidth reduction. This is expected because, in general, a numbering scheme which is efficient for reducing the profile is not efficient for reducing the bandwidth.[5,18,23] Table V shows that the GPS algorithm gives the closest results to the SAb algorithm. Clearly, neither the SFR nor the GK algorithm are as effective as the GPS algorithm for bandwidth reduction. However, from Table V, one can observe that for all the cases but one (# Nodes = 307), the SFR algorithm reduces the initial bandwidth of the test problems. Therefore, the results obtained by the SFR algorithm may be acceptable if a bandwidth reduction algorithm is not available.

4.1.2. Non-connected graphs. Everstine's[3] test problems, associated with non-connected graphs ($\lambda_2 = 0{\cdot}0$), are considered here. Two of the three alternatives presented by Paulino *et al.*,[1] for treating non-connected graphs, are investigated. They are denoted by SFR(1) and SFR(2), and refer to Tables II and III, respectively, in Reference 1. Non-connected graphs are important, for example, in applications such as substructuring[5] or domain partitioning for parallel finite element analysis.[24-26]

Table VI lists some initial data and the results obtained with the SFR algorithm on Everstine's[3] benchmark test problems for which the associated graph is non-connected. The initial data are the # Nodes, $|E|$, matrix densities g (per cent), and the number of connected components (# CC). The results are the profiles (P) produced by the SFR, SApw,[17] GPS[4], and GK[4] algorithms. With respect to the SFR algorithm and the second alternative for treating non-connected graphs

Table VI. Profiles of SFR, SApw, GPS and GK algorithms for the meshes associated with non-connected graphs in Everstine's[3] test problems

Problem		Density		Actual P					Normalized P (or $\bar{W}$)		
									SFR(i)*	GPS	GK
# Nodes	$\|E\|$	g(%)	#CC	Initial	SFR(i)*	SApw	GPS	GK	SApw	SApw	SApw
198	597	3·55	6	5817	1438	1287	1336	1313	1·117	1·038	1·020
					1454				1·130		
234	300	1·52	7	1999	1462	1016	1509	1349	1·439	1·485	1·328
					1487				1·464		
346	1440	2·69	4	9054	7722	6136	7996	8442	1·258	1·303	1·376
					10873				1·772		
492	1332	1·30	2	34282	3973	3304	5714	5513	1·202	1·729	1·669
					3984				1·206		
512	1495	1·34	32	6530	4936	4384	5181	4821	1·126	1·182	1·100
					5139				1·172		
607	2262	1·39	4	30615	15088	13065	15704	14760	1·155	1·202	1·130
					15600				1·194		

*i = 1, 2; the first result (i = 1) refers to Table II in Part I of this work.[1] The second result (i = 2) refers to Table III in Part I of this work.[1]

(SFR(2)[1]), the orders adopted for the reduced space (q) in the examples of Table VI are 15, 17, 11, 7, 42 and 11, respectively.

On average, the SFR(1), SFR(2), GPS and GK algorithms give profiles which are 21, 32, 32 and 27 per cent, respectively, in excess of the SApw profile. The *worst case* profiles for the SFR(1), SFR(2), GPS and GK algorithms are 44, 77, 73 and 67 per cent in excess of the SApw profile. Note, however, that Table VI is a very limited set of data and the above results are preliminary ones.

A few comments about the SFR(2) algorithm are in order. For the case where # Nodes = 492, there are two connected components in the graph. All the negative numbers in $\mathbf{y}_2$ are associated with one component (249 vertices), and the positive numbers in $\mathbf{y}_2$ are associated with the other component (243 vertices). For all the examples in Table VI, except for the case with # Nodes = 198, the components in the graph have been numbered sequentially. For the example with # Nodes = 198, there is a little mixture in the numbering between the two components with the smallest number of vertices (six vertices in each). However, if the eigenvector convergence criterion is used, then the components are separated. Moreover, to solve this type of problem, perhaps the two alternatives (SFR(1) and SFR(2)) can be efficiently combined. Note that these observations are essentially based on numerical calculations.

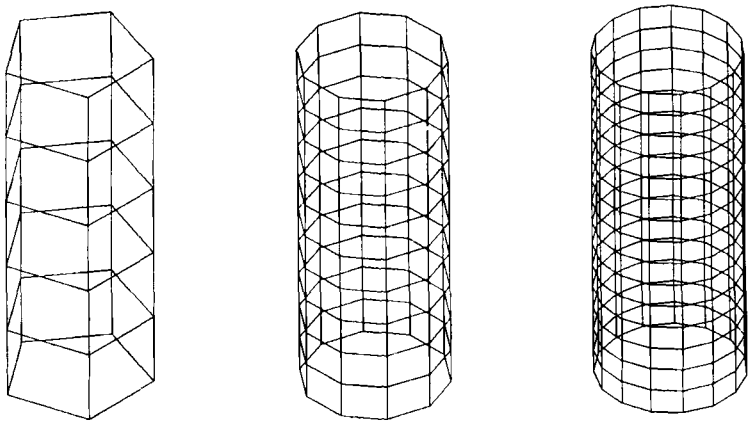
4.2. *Algebraic connectivity of an FEG*

A set of 2-D and 3-D meshes of regular patterns is used here to qualitatively assess the asymptotic behaviour of the algebraic connectivity (λ_2) of the associated FEGs. The objective is to evaluate λ_2 as the mesh size and the connectivity in the FEG change. In this study, the number of iterations for both the subspace and QR methods is unlimited in a numerical sense, i.e. the maximum number of iterations has been chosen to be a very large number (here, as in Everstine's[3] test problems, 12 500 has been used).

Six families of meshes are considered for the present study, as illustrated by Figures 3–8. Each family is associated with FEGs of similar structures but of different sizes. Figure 3 shows cylindrical grids with beam type (1-D) elements. The discretizations adopted ((a) 5×5; (b) 10×10; (c) 15×15) relate to the angular (θ) and vertical (z) number of divisions with respect to cylindrical co-ordinates (r, θ, z). The discretizations adopted for the grids of Figures 4–6 relate to 2-D Cartesian co-ordinates, and those for the grids of Figures 7 and 8 relate to 3-D Cartesian co-ordinates. Figures 4–6 show square grids with beam (1-D), Q4 and Q8 elements, respectively. Figures 7 and 8 show cubic grids with 1-D and BRICK-8 elements, respectively.

Table VII lists the example classes, the finite element types for each family of meshes, the grid discretizations, # Nodes, # Elements, $|V|$, $|E|$, the actual λ_2, and the upper bound for λ_2 obtained from

$$\lambda_2 \leqslant \left(\frac{|V|}{|V|-1}\right) \min_i l_{ii} \quad \text{(no sum)} \tag{9}$$



(*a*) 5×5 (*b*) 10×10 (*c*) 15×15

Figure 3. Cylindrical grids with beam (1-D) elements

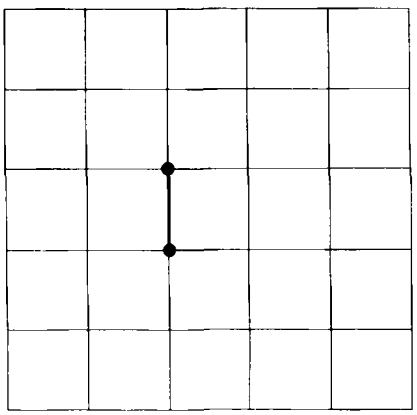


Figure 4. Square grid with beam (1-D) elements

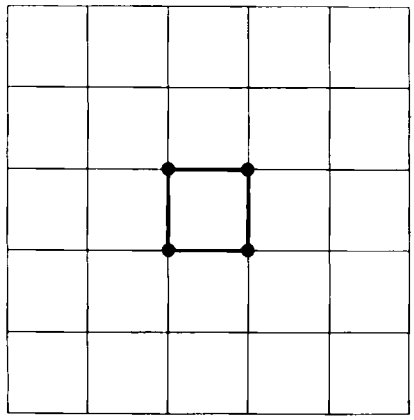


Figure 5. Square grid with Q4 elements

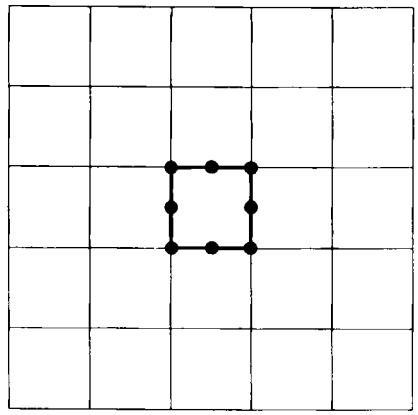


Figure 6. Square grid with Q8 elements

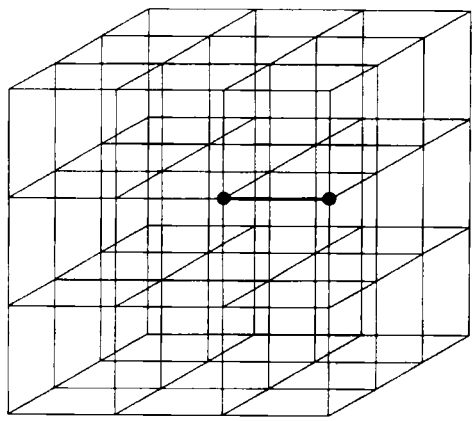


Figure 7. Cubic grid with beam (1-D) elements

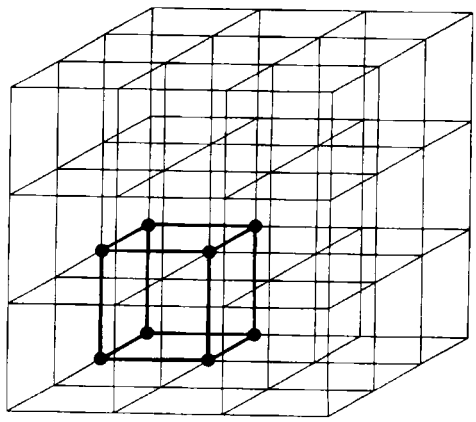


Figure 8. Cubic grid with BRICK-8 elements

as reported in Part I of this work (see equation (12) of Section 2[1]). For each example and each element type (family of meshes), as the grid discretization increases, λ_2 decreases because the *diameter*† of the FEG increases. Comparing families of square (or cubic) grids, one can verify that as the order of the finite element increases (e.g. from linear to quadratic), λ_2 increases. This happens because $|E|$ increases. From Table VII, it is interesting to observe that the cylindrical grids and the square grids with the same discretization (in different co-ordinate systems) have equal λ_2. Comparing the last two columns of Table VII, one verifies that, in general, the upper bound for λ_2 obtained from equation (9) is always much larger than the actual λ_2.

Another parameter that can be used as a measure of connectivity of graphs is the *isoperimetric number*. For explanations about this parameter, see, for example, Reference 27.

† For definition of diameter of a graph, see, for example, the book by George and Liu[20]

Table VII. Study of the algebraic connectivity of an FEG (G)

Example	Element type	Grid	#Nodes $= \|V\|$	#Elements	$\|E\|$	λ_2	λ_2(max) (equation (9))
Cylinder	Beam (1-D) (Figure 3)	5 × 5	30	55	55	0·2679	3·1034
		10 × 10	110	210	210	0·0810	3·0275
		15 × 15	240	465	465	0·0384	3·0125
	Beam (1-D) (Figure 4)	5 × 5	36	60	60	0·2679	2·0571
		10 × 10	121	220	220	0·0810	2·0167
		20 × 20	441	840	840	0·0223	2·0045
Square	Q4 (Figure 5)	5 × 5	36	25	110	0·7078	3·0857
		10 × 10	121	100	420	0·2276	3·0250
		20 × 20	441	400	1640	0·0648	3·0068
	Q8 (Figure 6)	5 × 5	96	25	580	0·9432	7·0736
		10 × 10	341	100	2260	0·2857	7·0206
		20 × 20	1281	400	8920	0·0786	7·0055
Cube	Beam (1-D) (Figure 7)	3 × 3 × 3	64	144	144	0·5858	3·0476
		6 × 6 × 6	343	882	882	0·1981	3·0088
		9 × 9 × 9	1000	2700	2700	0·0979	3·0030
	BRICK-8 (Figure 8)	3 × 3 × 3	64	27	468	3·5153	7·1111
		6 × 6 × 6	343	216	3258	1·4385	7·0205
		9 × 9 × 9	1000	729	10476	0·7620	7·0071

4.3. Convergence study

The convergence of the special version of the subspace iteration method is investigated here. Also, the default values adopted for the tolerance (TOL) and the maximum number of iterations (see Figure 1(b)) are, to some extent, justified. To solve the eigenproblem, the eigenvalue (see equation (22) in Part I of this work[1]) or the eigenvector (see equation (23) in Part I of this work[1]) convergence criterion may be used. Here, the eigenvalue convergence criterion is considered because it has been explicitly used for the examples reported in this paper.

For the present study, the cylindrical grid shown in Figure 3(c) is considered as a representative example. As in the previous examples, the number of iterations for both the subspace and QR methods is unlimited in a numerical sense, i.e. the maximum number of iterations has been chosen to be a very large number (here, 12 500 has been used). Moreover, the tolerances for both the subspace and QR methods are equally prescribed as listed in the first column of Table VIII. In addition to some preliminary information about the cylindrical grid (Figure 3(c)) example, Table VIII lists the tolerances (TOL), the required number of iterations per subspace (#ITER), the CPU time in seconds (Time), the approximate values obtained for the algebraic connectivity (λ_2) of the FEG G, profile (P), r.m.s. wavefront ($\hat{W}$), and bandwidth (B).

Table VIII shows that TOL $= 10^{-3}$ leads to the lowest values for both P and $\hat{W}$. For several examples, it is sufficient to use a tolerance around 10^{-3} to obtain good values for P, W, $\bar{W}$, and $\hat{W}$. However, for most of the examples tested considering the eigenvalue convergence criterion, TOL $= 10^{-6}$ gives, on the average, the best results. Furthermore, in Table VIII, the results for λ_2, P, $\hat{W}$ and B converge for TOL $= 10^{-6}$.

For accuracy purposes, setting the maximum number of iterations to a very large number is adequate. However, for practical purposes, the default value for this number has been set as 125

Table VIII. Convergence results

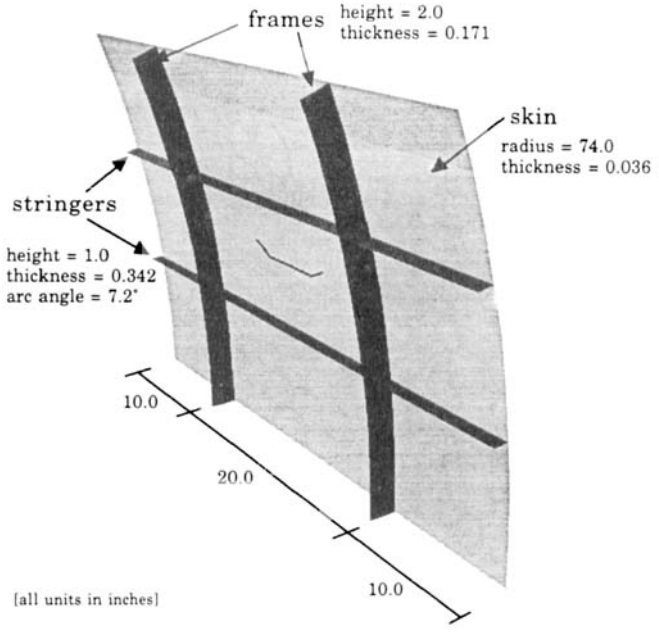
TOL	#ITER	Time (s)	λ_2	P	$\hat{W}$	B
10^{-1}	5	1·98	0·160762	4423	19·5918	44
10^{-2}	12	3·93	0·132499	5251	23·2557	68
10^{-3}	29	8·72	0·042019	3404	14·5662	26
10^{-4}	64	18·69	0·038922	3569	15·1486	24
10^{-5}	92	27·92	0·038429	3642	15·4102	17
10^{-6}	**122**	**37·84**	**0·038429**	**3642**	**15·4102**	**16**
10^{-7}	195	59·87	0·038429	3642	15·4102	16
10^{-8}	226	73·52	0·038429	3642	15·4102	16

Preliminary information: cylindrical grid (see Figure 3(c)); #Nodes = $|V| = 240$; $|E| = 465$; convergence criterion: eigenvalue; initial values: $P = 16258$; $\hat{W} = 76·8730$; $B = 239$

for both the subspace and QR methods. In Table VIII, the results for λ_2, P, W and B converge with 122 iterations.

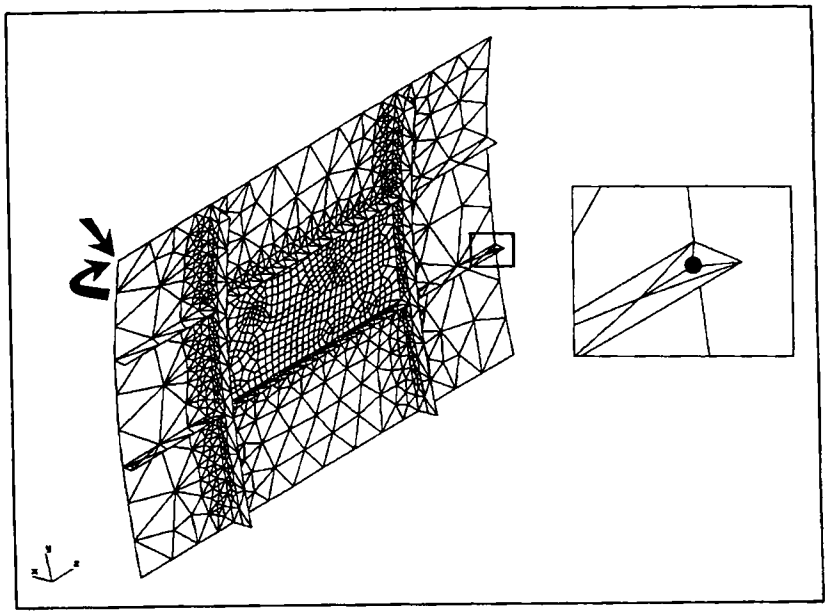
4.4. SFR algorithm running time performance

The cracked fuselage panel of an aeroplane[28] is considered here for studying the SFR algorithm running time performance. Figure 9(a) shows the physical model, Figure 9(b) shows the discretized finite element model and Figure 9(c) shows the corresponding dual graph G^*. Graphical



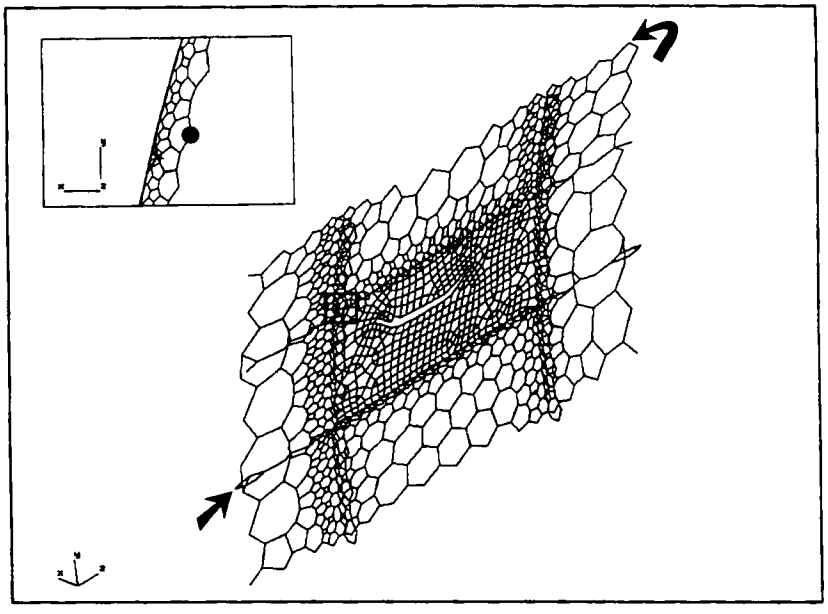
(a) Physical model.

Figure 9a



(b) FEM discretization

(1262 nodes; 2018 elements: 1526 T3s & 492 Q4s)



(c) Dual graph (G^*)

$|V| = 2018$; $|E| = 3266$

Figure 9. Cracked fuselage panel

representations for the FEGs G and $G^\bullet$ are not shown in Figure 9 because they are too dense (4264 and 12 594 edges, respectively), but they are considered in the present analysis.

Table IX shows the computer running time per task (assembling of $\mathbf{L}$, solution of the eigenproblem and renumbering of the FEG), the total time to produce node (using G) and element (using G^* or $G^\bullet$) orderings, and the speed-up (S) rates.

According to Part I of this work,[1] the speed-up (S) is defined as the ratio between the CPU time to run the standard SFR and the preconditioned SFR:

$$S = \frac{t_1}{t_2 + t_3} \tag{10}$$

where t_1 is the CPU time for the standard SFR algorithm, t_2 is the CPU time to preorder the vertices of the FEG by the RCM algorithm and t_3 is the CPU time for the SFR after preordering. The CPU time for the RCM algorithm is negligible compared to the CPU time for the SFR algorithm (nevertheless, the RCM has been considered to compute S). Clearly, the speed-up column in Table IX shows that the preordering strategy provides improved computational efficiency, specially for the dense graphs G and $G^\bullet$.

In Table IX, the FEG G^* is the most efficient in terms of CPU time. Comparing the results for G^* and $G^\bullet$, one verifies that they have the same number of vertices but the number of edges in $G^\bullet$ is much larger than in G^*. Therefore, in this case, $G^\bullet$ demands more CPU time.[1] It should be noted that the efficiency (in terms of CPU time) of G is due to the effective speed-up.

Table IX shows that, for each FEG (G, G^* or $G^\bullet$), almost all the CPU time is spent on the solution of the eigenproblem to compute the second eigenpair $(\lambda_2, \mathbf{y}_2)$ of $\mathbf{L}$. Therefore, the efficiency of the SFR algorithm depends on the efficiency of the algorithm used for the eigensolution. Perhaps the present special version of the subspace iteration method may be further improved in the future. Moreover, another algorithm for solving eigenproblems, such as that of Lanczos,[29,30] could improve the efficiency of the SFR algorithm.

4.5. *Practical finite element examples*

In this section, the performance of the SFR algorithm is evaluated by means of representative FEM application problems. We have tried a selection of five meaningful and practical problems which have the essential features to test effectiveness of the resequencing techniques discussed in this paper. The finite element meshes are illustrated in Figures 9–13. The initial node and element ordering is arbitrary.

The following variables are studied in this section. The speed-up (S) achieved with preconditioning by preordering is evaluated (see equation (10)). Also, the computational efficiency between the eigenvalue and eigenvector convergence criteria[1] is compared by means of the factor

Table IX. Timing statistics for the cracked fuselage panel (HP apollo 9000—Model 720)

FEG	$\lvert V\rvert$	$\lvert E\rvert$	Assemble $\mathbf{L}$ (s)	Eigenproblem (s)	Renumbering (s)	TOTAL* (s)	Speed-up (S)
G	1262	4264	0·03	469·45	0·02	469·50	3·07
G^*	2018	3266	0·02	1037·38	0·02	1037·42	1·38
$G^\bullet$	2018	12 594	0·07	1211·27	0·02	1211·36	1·81

*After preordering by RCM
Cracked fuselage panel[28] (see Figure 9); #Nodes = 1262; #Elements = 2018 (1526 T3s and 492 Q4s)

f, which is defined with respect to the SFR algorithm as

$$f = \frac{\text{CPU time considering the eigenvector convergence criterion}}{\text{CPU time considering the eigenvalue convergence criterion}} \tag{11}$$

where, in this case, the original mesh (and FEG) configurations are assumed (i.e. no preconditioning by preordering). All the CPU time statistics have been obtained on an HP apollo 9000—Model 720 (see Figure 1(a)). The profiles (P), maximum wavefronts (W), r.m.s. wavefronts ($\hat{W}$), and bandwidths (B) obtained by the SFR algorithm are compared with those obtained by the GPS and GK algorithms.[4] The algebraic connectivities of the associated FEGs are also evaluated.

In the first part of this work,[1] we claim that a pseudoperipheral vertex, i.e. a vertex with high *eccentricity*[‡] (e), is obtained as a natural outcome of the SFR algorithm. Moreover, this eccentricity must be as close as possible to the *diameter*[‡] (δ) of the graph. In order to support the above claim, the pseudoperipheral vertices obtained by the SFR (the vertex corresponding to the smallest component in $\mathbf{y}_2$) are directly compared with the ones obtained by George and Liu's[20,31] algorithm. In each of the Figures 9–13, straight and curved arrows indicate the pseudoperipheral nodes obtained by the standard and preconditioned SFR algorithms, respectively, and a bullet indicates the pseudoperipheral node obtained by George and Liu's[20,31] algorithm.

Since the SFR algorithm is based on global properties of the graph, the vertex corresponding to either the smallest or the largest component in $\mathbf{y}_2$ can be used as a starting vertex for resequencing algorithms based on the pseudoperipheral vertex concept. The next example that follows illustrates this point.

4.5.1. Lattice dome. Figure 10 shows a lattice dome. This and other types of framed dome systems have been studied by Paulino[5] and Haber *et al.*[32] In Figure 10, note that the SFR pseudoperipheral node is in a location topologically analogous to George and Liu's[31] pseudoperipheral node. The eccentricity (e) of these nodes is equal to the diameter (δ) of the nodal graph, $e = \delta = 10$.

For meshes made up of 1-D finite elements (two noded elements), the isometric projection of the structure is also one possible isomorphic representation of the associated FEG G—see the main view in Figure 10. Moreover, the *plane view xz* in Figure 10 is another *isomorphic* representation of the FEG G.

Table X shows that the SFR algorithm gives the best results for P and $\hat{W}$, while the GK[4] gives the best result for W. The last two columns of Table X show an application of the Interactive Modified Reverse Cuthill Mckee (IMRCM) algorithm presented by Paulino.[5] Basically, this algorithm allows the user to select a set of nodes which define the first level of the level structure associated with the graph corresponding to the topology of the finite element mesh. In the first and second columns under IMRCM (Table X), the vertices corresponding to the smallest ($\min_i (y_2)_i$, $i = 1, \ldots, |V|$) and largest ($\max_i (y_2)_i$, $i = 1, \ldots, |V|$) components in $\mathbf{y}_2$, respectively, are used as pseudoperipheral vertices for the IMRCM algorithm. Comparing these two columns, one can verify that all the results obtained are very close or equal, as expected. Note that the SFR algorithm (third column in Table X) and the IMRCM algorithm using $\min_i (y_2)_i$ as a pseudoperipheral vertex (sixth column in Table X) have the same starting vertex for the renumbering process. Moreover, this example shows how the rooted level structure reverse numbering (last two columns in Table X) compares to the SFR numbering (third column in Table X), which is based on the $\mathbf{y}_2$ components.

[‡] For definitions of eccentricity (e) of a vertex and diameter of a graph (δ), see, for example, the book by George and Liu[20]

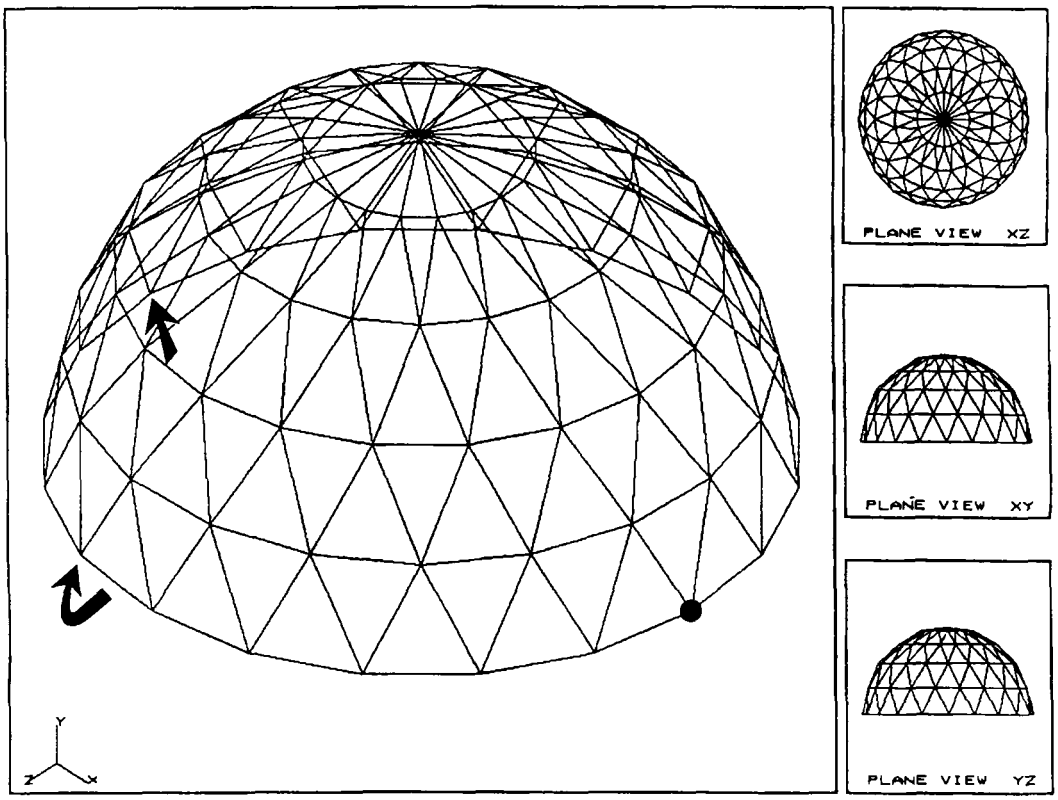


Figure 10. Lattice dome[5]

Table X. Results for the lattice dome (see Figure 10)

					IMRCM	
Parameter	Initial	SFR	GPS	GK	$\min_i (y_2)_i$	$\max_i (y_2)_i$
P	3539	1207	1306	1226	1324	1303
W	63	19	19	18	21	21
$\hat{W}$	38·84	12·49	13·57	12·70	13·86	13·60
B	100	26	19	25	23	23

#Nodes = $|V| = 101$; $|E| = 280$; #Elements = 280 (beams); FEG: G; $\lambda_2 = 0{\cdot}2554$; $S = 1{\cdot}76$; $f = 1{\cdot}66$

4.5.2. L-shaped building. Figure 11 shows a lateral load-resistant L-shaped building. In this example, the SFR and George and Liu's[31] pseudoperipheral nodes coincide. The eccentricity (e) of these nodes is equal to the diameter (δ) of the nodal graph, $e = \delta = 11$.

Table XI shows that the SFR algorithm gives the best results for P and $\hat{W}$, while the GK[4] gives the best result for W.

4.5.3. Space station. Figure 12 shows a space station. This structural system has been studied by Aubert.[33] It is interesting to observe that, again, the SFR and George and Liu's[31]

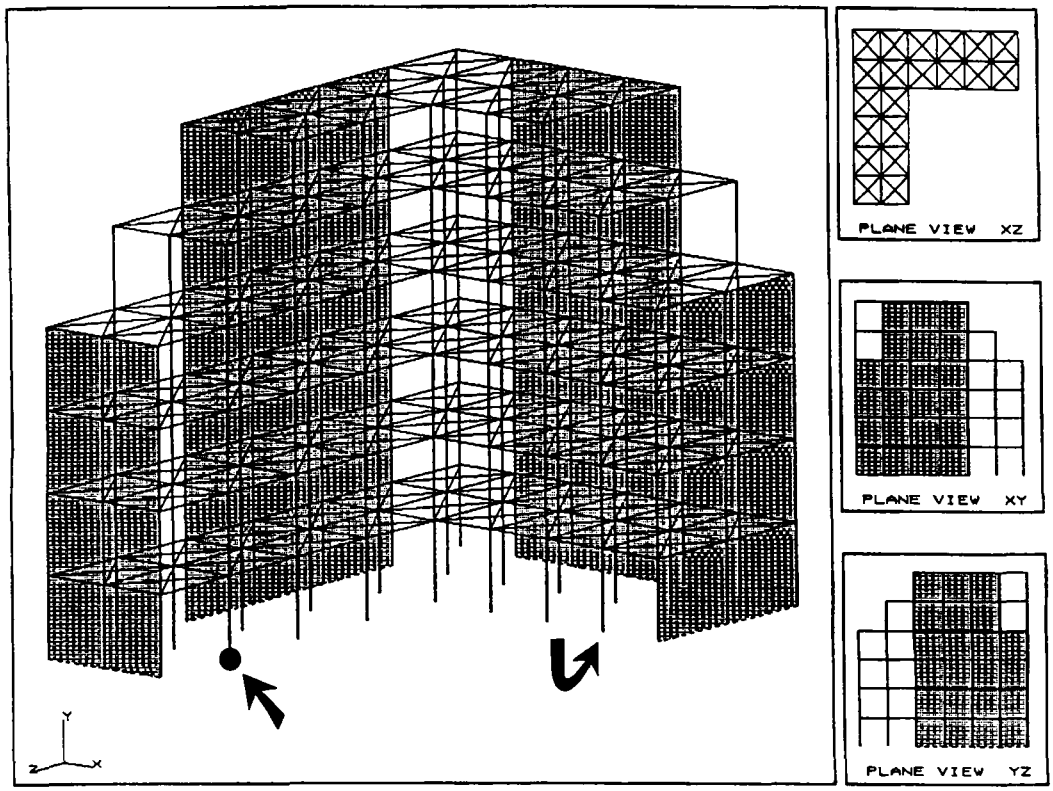


Figure 11. L-shaped building

Table XI. Results for the L-shaped building (see Figure 11)

Parameter	Initial	SFR	GPS	GK
P	14956	4109	4681	4345
W	125	32	34	29
$\hat{W}$	79·41	20·71	23·33	21·56
B	212	66	43	51

#Nodes = $|V| = 213$; $|E| = 792$; #Elements = 730 (678 beams and 52 Q4s); FEG: G; $\lambda_2 = 0{\cdot}2162$; $S = 0{\cdot}97$; $f = 2{\cdot}45$

pseudoperipheral vertex coincide. The eccentricity (e) of these nodes is equal to the diameter (δ) of the graph, $e = \delta = 25$.

Table XII shows that the GK algorithm gives the best results for P, W and $\hat{W}$. However, the results of the SFR algorithm are very close to the ones from the GK[4] algorithm.

4.5.4. Cracked fuselage panel. Figure 9(b) shows the finite element model for the cracked fuselage panel of an aeroplane. For the associated nodal graph (G), the SFR pseudoperipheral vertex has eccentricity $e = 23$. George and Liu's[31] pseudoperipheral vertex has eccentricity

Table XII. Results for the space station (see Figure 12)

Parameter	Initial	SFR	GPS	GK
P	28 816	5405	5815	5390
W	184	34	35	31
$\hat{W}$	108·97	19·73	21·01	19·34
B	303	95	39	53

$\#\text{Nodes} = |V| = 304$; $|E| = 1428$; $\#\text{Elements} = 1428$ (beams); FEG: G; $\lambda_2 = 0{\cdot}0546$; $S = 3{\cdot}34$; $f = 1{\cdot}73$

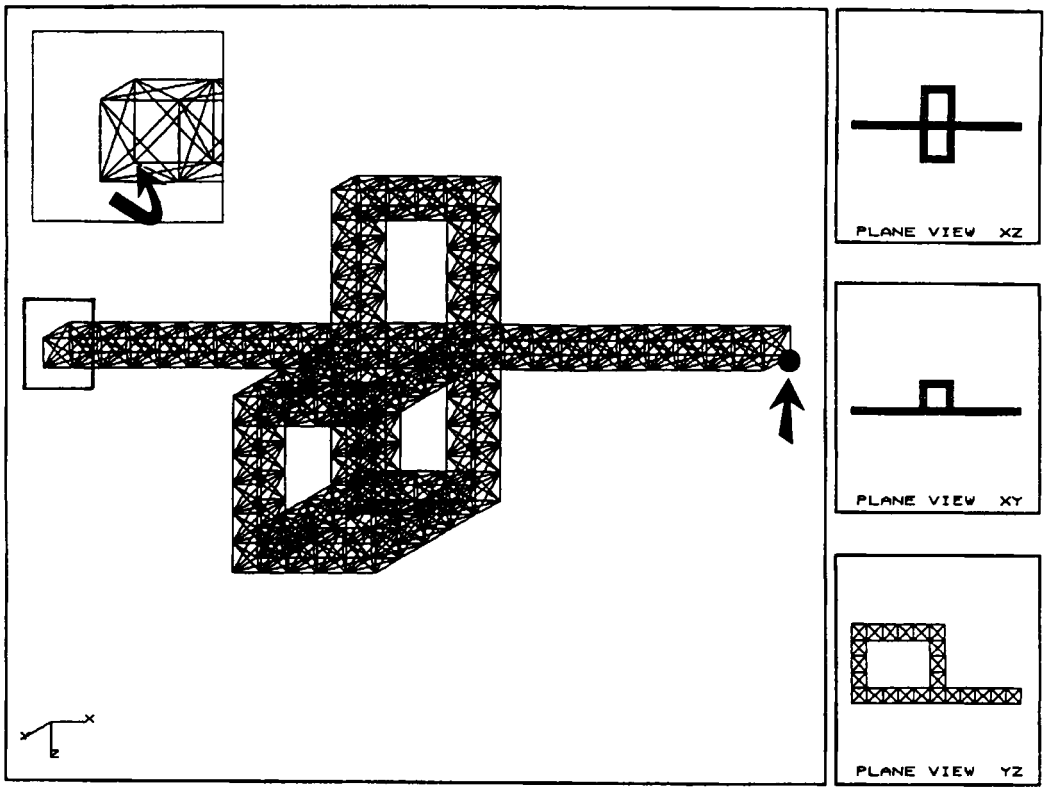


Figure 12. Space station[33]

$e = 24$, which is equal to the diameter (δ) of the nodal graph. Therefore, George and Liu's[31] algorithm gives a slightly better solution than the SFR.

Figure 9(c) shows the dual graph (G^*) representation for the mesh of Figure 9(b). In this example, the eccentricity for the SFR pseudoperipheral vertex is $e = 53$, and the one for George and Liu's[31] algorithm is $e = 52$. The diameter of the dual graph is $\delta = 55$. Therefore, in this case, the SFR gives a slightly better solution than George and Liu's[31] algorithm.

Consider now the communication graph (G^* is not shown in this paper) associated with the mesh of Figure 9(b). The eccentricity for the SFR pseudoperipheral vertex is $e = 24$, which is equal to the diameter (δ) of the associated communication graph. The eccentricity for George and

Liu's[31] algorithm is $e = 23$. Again, the SFR gives a slightly better solution than George and Liu's[31] algorithm.

For illustration purposes, Table XIII lists the results for P, W, $\hat{W}$ and B using the FEGs G, G^* and $G^\bullet$ associated with the mesh of Figure 9(b). Note that if G is used, then the nodes are reordered; if either G^* or $G^\bullet$ is used, then the finite elements are reordered.[1] Therefore, the results for G cannot be directly compared with the ones for G^* and $G^\bullet$. The results for G^* and $G^\bullet$ can be compared because both graphs represent connectivity among finite elements and these graphs relate to matrices of the same order (but with different number of components). For all the FEGs (G, G^* and $G^\bullet$), the SFR algorithm gives better results for P, W and $\hat{W}$ than those from the GPS and GK[4] algorithms.

For each of the graphs G, G^* and $G^\bullet$, the values obtained for speed-up (S) are 3·07, 1·38 and 1·81, respectively. Also, the factors f, obtained using these graphs, are very close to 1, i.e. the efficiencies of the eigenvector and eigenvalue convergence criteria are similar.

4.5.5. Gas turbine blade. Figure 13 shows a gas turbine blade. This type of structure has been studied by Wawrzynek[34] for the simulation of fatigue crack growth. In this example, the eccentricities of both the SFR and George and Liu's[31] pseudoperipheral nodes coincide with the diameter of the nodal graph, $e = \delta = 28$.

Table XIV shows that the SFR algorithm gives the best results for P, W and $\hat{W}$.

4.5.6. Discussion about the practical finite element examples. On most occasions, the SFR algorithm gives the best results for profile, maximum wavefront and r.m.s. wavefront. It is observed that the GPS algorithm gives the smallest bandwidth for all the five examples tested. This issue has been discussed in the last paragraph of Section 4.1.1.

The eigenvalue convergence criterion is, on the average, faster than the eigenvector convergence criterion by a factor of 1·59. Also, with the eigenvalue convergence criterion, the preconditioned SFR is, on the average, faster than the standard SFR algorithm by a factor of 2·61.

Numerically, the qualitative behaviour of the preconditioned SFR is similar to that of the standard SFR. For instance, for the practical finite element examples in Figures 9–13, a curved

Table XIII. Results for the cracked fuselage panel (see Figure 9)

FEG	Parameter	Initial	SFR	GPS	GK
G	P	504 649	47 321	72 017	69 686
$\lambda_2 = 0{\cdot}0175$	W	682	65	84	89
$\|V\| = 1262$	$\hat{W}$	447·61	38·87	58·93	57·34
$\|E\| = 4264$	B	1259	168	99	139
G^*	P	273 788	61 558	87 787	84 343
$\lambda_2 = 0{\cdot}0042$	W	204	59	61	60
$\|V\| = 2018$	$\hat{W}$	143·58	32·69	44·77	43·05
$\|E\| = 3266$	B	1516	429	67	117
$G^\bullet$	P	581 012	126 765	189 240	183 643
$\lambda_2 = 0{\cdot}0243$	W	552	101	128	147
$\|V\| = 2018$	$\hat{W}$	318·12	64·91	96·38	93·74
$\|E\| = 12\,594$	B	1985	292	159	199

#Nodes = 1262; #Elements = 2018 (1526 T3s and 492 Q4s)

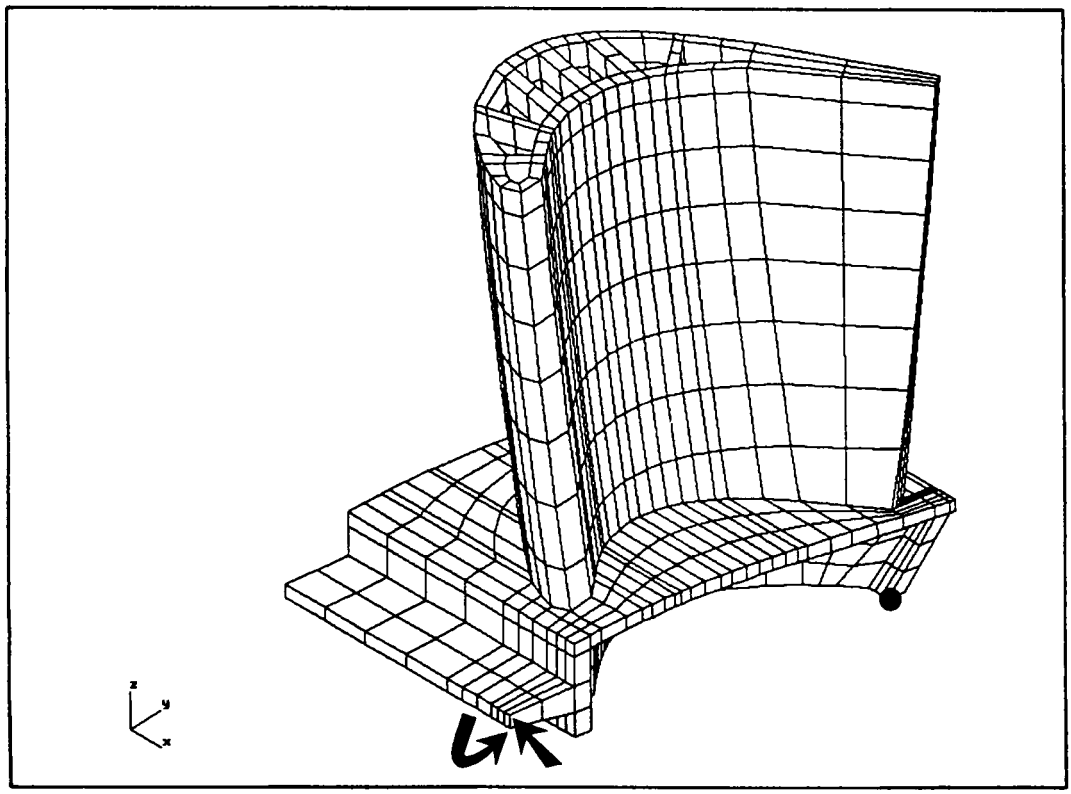


Figure 13. Gas turbine blade[34]

Table XIV. Results for the gas turbine blade (see Figure 13)

Parameter	Initial	SFR	GPS	GK
P	1 313 225	127 133	137 261	129 869
W	1305	103	124	120
$\hat{W}$	817·29	72·36	78·86	74·53
B	1819	145	127	148

#Nodes = $|V|$ = 1820; $|E|$ = 15833; #Elements = 944 (BRICKs-8);
FEG: G; λ_2 = 0·0843; S = 3·90; f = 1·11

arrow indicates the preconditioned SFR pseudoperipheral vertex. As expected, these vertices are in topologically comparable locations (in terms of eccentricity) to the ones provided by the standard SFR algorithm.

5. CONCLUSIONS

The SFR algorithm has been shown to be effective for profile and wavefront reduction of large sparse matrices with symmetric configuration. Moreover, the algorithm is effective for reordering nodes and/or elements of generic finite element meshes.

A general implementation of the SFR algorithm has been presented and evaluated. The main computation in this algorithm is the eigensolution, which has been accomplished by a robust

implementation of the subspace iteration method. This implementation has delivered satisfactory results for all the examples tested. The use of the eigenvalue convergence criterion and preconditioning by preordering have been shown to provide improved computational efficiency. Furthermore, consideration of alternative eigensolvers, such as that of Lanczos,[29,30] could improve the computational efficiency of the resequencing algorithm.

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.

Most of the computations in the present work have been performed in CADIF (Computer Aided Design Instructional Facility) at Cornell University. The authors are especially grateful to Catherine Mink, CADIF Director, for giving credit to the ideas presented to her and for providing all the resources necessary to carry out this research. The excellent computing system support by John M. Wolf, from CADIF, is also acknowledged.

The meshes of Figures 9(b) and 13 were drawn using the finite element postprocessor POS-3D, developed by Waldemar C. Filho, from TeCGraf (Group of Technology in Computer Graphics), Rio de Janeiro, Brazil.

Dr. Gordon C. Everstine, from the David W. Taylor Model Basin (Department of the Navy), has given the authors a computer tape with his collection of benchmark problems for testing nodal resequencing algorithms.

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.

Last, but not least, 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. Gattass and S. Mukherjee, 'Node and element resequencing using the Laplacian of a finite element graph—Part I: General concepts and algorithm', *Int. j. numer. methods eng.*, **37,** 1511–1530 (1994).
2. M. Gattass, G. H. Paulino and J. C. Gortairee, 'Geometrical and topological consistency in interactive graphical preprocessors of three-dimensional framed structures', *Comput. Struct.*, **46,** 99–124 (1993).
3. 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).
4. J. G. Lewis, 'Implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms', *ACM Trans. Math. Software*, **8,** 180–189 (1982).
5. 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.
6. S. W. Sloan and W. S. Ng, 'A direct comparison of three algorithms for reducing profile and wavefront', *Comput. Struct.*, **33,** 411–419 (1989).
7. 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).
8. M. Gattass, M. Ferrari and L. H. Figueiredo, 'Solution of systems of sparse symmetric positive definite matrices—the modified Crout method', (in Portuguese), *Internal Report AT 24/84*, Department of Civil Engineering, PUC-Rio, Rio de Janeiro, Brazil, 1984.
9. R. Sedgewick, 'Implementing quicksort programs', *Commun. ACM*, **21,** 847–857 (1978).
10. G. H. Paulino and M. Gattass, 'A methodology for the development of interactive graphical preprocessors of finite elements', (in Portuguese), *ABCM Proc. 10th Brazilian Congr. of Mech. Eng.*, 1989, pp. 117–120.
11. 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).
12. N. E. Gibbs, 'A hybrid profile reduction algorithm', *ACM Trans. Math. Software*, **2,** 378–387 (1976).

13. I. P. King, 'An automatic reordering scheme for simultaneous equations derived from network systems', *Int. j. numer. methods eng.*, **2**, 523–533 (1970).
14. H. A. Araújo Filho, 'Nodal reordering for the solution of large systems considering sparsity', (in Portuguese), *M.S. Thesis*, Department of Mechanical Engineering, COPPE/UFRJ, Rio de Janeiro, Brazil, 1986.
15. S. W. Sloan, 'An algorithm for profile and wavefront reduction of sparse matrices', *Int. j. numer. methods eng.*, **23**, 239–251 (1986).
16. S. W. Sloan, 'A FORTRAN program for profile and wavefront reduction', *Int. j. numer. methods eng.*, **28**, 2651–2679 (1989).
17. B. A. Armstrong, 'Near-minimal matrix profiles and wavefronts for testing nodal resequencing algorithms', *Int. j. numer. methods eng.*, **21**, 1785–1790 (1985).
18. 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).
19. I. S. Duff, R. G. Grimes and J. G. Lewis, 'Sparse matrix test problems', *ACM Trans. Math. Software*, **15**, 1–14 (1989).
20. J. A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1981.
21. R. Levy, 'Resequencing of the structural stiffness matrix to improve computational efficiency', *JPL Quart. Tech. Rev.*, **1**, 61–70 (1971).
22. B. A. Armstrong, 'A hybrid algorithm for reducing matrix bandwidth', *Int. j. numer. methods eng.*, **20**, 1929–1940 (1984).
23. R. A. Snay, 'Reducing the profile of sparse symmetric matrices', *Bull. Géodésique*, **50**, 341–352 (1976).
24. A. Pothen, H. D. Simon and K. P. Liou, 'Partitioning sparse matrices with eigenvectors of graphs', *SIAM J. Matrix Anal. Applic.*, **II**, 430–452 (1990).
25. H. D. Simon, 'Partitioning of unstructured problems for parallel processing', *Comput. Systems Eng.*, **2**, 135–148 (1991).
26. B. Hendrickson and R. Leland, 'Multidimensional spectral load balancing', *SANDIA Report SAND93-0074*, Category UC-405, Sandia National Laboratories, Albuquerque, NM 87185, 1993.
27. B. Mohar, 'Isoperimetric numbers of graphs', *J. Combin. Theory, Ser. B*, **47**, 274–291 (1989).
28. D. O. Potyondy, 'A software framework for simulating curvilinear crack growth in pressurized thin shells', *Ph.D. Thesis*, School of Civil and Environmental Engineering, Cornell University, Ithaca, New York, 1993.
29. T. J. R. Hughes, *The Finite Element Method—Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1987.
30. B. Nour-Omid, B. N. Parlett and R. L. Taylor, 'Lanczos versus subspace iteration for solution of eigenvalue problem', *Int. j. numer. methods eng.*, **19**, 859–871 (1983).
31. J. A. George and J. W.-H. Liu, 'An implementation of a pseudoperipheral node finder', *ACM Trans. Math. Software*, **5**, 284–295 (1979).
32. R. B. Haber, T. A. Mutryn, J. F. Abel and D. P. Greenberg, 'Computer aided design of framed dome structures with interactive graphics', *Comput. Aided Design*, **9**, 157–164 (1977).
33. B. H. Aubert, 'Numerical simulation of the transient nonlinear dynamics of actively controlled structures', *Ph.D. Thesis*, School of Civil and Environmental Engineering, Cornell University, Ithaca, New York, 1992.
34. P. A. Wawrzynek, 'Discrete modeling of crack propagation: theoretical aspects and implementation issues in two and three dimensions', *Ph.D. Thesis*, School of Civil and Environmental Engineering, Cornell University, Ithaca, New York, 1991.