COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING
Commun. Numer. Meth. Engng, **15**, 309–323 (1999)

ON ORDERING ELEMENTS FOR A FRONTAL SOLVER

JENNIFER A. SCOTT*
Department for Computation and Information, Atlas Centre, Rutherford Appleton Laboratory, Oxon OX11 0QX, UK

SUMMARY

The efficiency of the frontal method for the solution of finite-element problems depends on the order in which the elements are assembled. This paper looks at using variants of Sloan's algorithm to reorder the elements. Both direct and indirect reordering algorithms are considered and are used in combination with spectral orderings. Numerical experiments are performed on a range of practical problems and, on the basis of the results, a hybrid Sloan element resequencing algorithm is proposed for use with a frontal algorithm. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS ordering finite elements; frontal method; Sloan algorithm; spectral method

1. INTRODUCTION

In this paper, we are interested in the efficient use of the frontal method to solve large sparse systems of linear equations

$$\mathbf{AX} = \mathbf{B} \tag{1}$$

where the $n \times n$ matrix $\mathbf{A}$ is the sum of *nelt* finite-element matrices

$$\mathbf{A} = \sum_{l=1}^{nelt} \mathbf{A}^{(l)} \tag{2}$$

and the $n \times nrhs$ matrix $\mathbf{B}$ of right-hand sides is of the form

$$\mathbf{B} = \sum_{l=1}^{nelt} \mathbf{B}^{(l)} \tag{3}$$

Each matrix $\mathbf{A}^{(l)}$ has non-zeros in a few rows and columns and corresponds to the matrix from element l. The frontal method is a variant of Gaussian elimination, the main feature of the method being that the contributions $\mathbf{A}^{(l)}$ from the finite elements are assembled one at a time and the construction of the assembled coefficient matrix $\mathbf{A}$ is avoided by interleaving assembly and elimination operations. An assembly operation is of the form

$$a_{ij} \Leftarrow a_{ij} + a_{ij}^{(l)} \tag{4}$$

* Correspondence to: J. Scott, Atlas Centre, Rutherford Appleton Laboratory, Chilton, Didcot, Oxon OX11 0QX, U.K.
E-mail: J. Scott@rl.ac.uk

CCC 1069–8299/99/050309–16$17·50
Copyright © 1999 John Wiley & Sons, Ltd.

Received 6 April 1998
Accepted 18 August 1998

where $a_{ij}^{(l)}$ is the (i, j)th non-zero entry of the element matrix $\mathbf{A}^{(l)}$. A variable is *fully summed* if it is involved in no further sums of the form (4) and is *partially summed* if it has appeared in at least one of the elements assembled so far but is not yet fully summed. The Gaussian elimination operation

$$a_{ij} \Leftarrow a_{ij} - a_{il}[a_{ll}]^{-1}a_{lj} \tag{5}$$

may be performed once all the terms in the triple product in (5) are fully summed.

Since variables can only be eliminated after they are fully summed, the assembly order will determine, to a large extent, the order in which the variables are eliminated. At any stage during the assembly and elimination processes, the fully and partially summed variables are held in main memory in a *frontal matrix*. Dense linear algebra operations are performed on the frontal matrix. For efficiency, in terms of both storage and arithmetic operations, the elements must be assembled in an order that keeps the size of the frontal matrix, known as the *wavefront*, as small as possible. Of interest is

- the maximum wavefront, since this affects the in-core storage needed
- the sum of the wavefronts, known as the *profile*, since this determines the total storage needed for the matrix factors
- the root-mean-square wavefront, since the work performed when eliminating a variable is proportional to the square of the current wavefront.

In the past, a number of algorithms for automatically ordering finite elements have been proposed.[1–9] Further references are given in Kaveh.[10] Duff *et al.*[8] divide element ordering algorithms into direct and indirect algorithms. Direct algorithms order the elements directly while indirect algorithms use a two-step approach in which the variables are first relabelled and then used to resequence the elements; the new variable indices are subsequently discarded. Duff *et al.* report that both approaches can be used effectively, and neither has been found to be consistently superior to the other.

The Harwell Subroutine Library[15] (HSL) code MC43[8] implements both a direct and an indirect ordering algorithm, based on the profile reduction algorithm of Sloan.[7] Several authors have considered variants of Sloan's algorithm (see, for example, Medeiros *et al.*[11]). Motivated by the findings of Kumfert and Pothen,[12] we recently looked at a number of ways of improving the performance and efficiency of Sloan's algorithm.[13] These included implementing the priority queue as a binary heap and using a hybrid algorithm that combines a spectral ordering (see, for example, Barnard *et al.*[14]) with the Sloan algorithm. This work led to improved codes for profile reduction (the HSL MC60 package together with a driver MC61) being included in the Harwell Subroutine Library and prompted us to look at revising MC43 in a similar way. The new element ordering code, which supersedes MC43, is called MC63.

The outline of this paper is as follows. In Section 2, we briefly review some of the graphs that can be associated with a finite-element mesh. These graphs are fundamental to our reordering algorithms. In Section 3, we look at Sloan's algorithm. In Section 4, we discuss using spectral orderings to resequence elements and introduce a hybrid method that combines using a spectral ordering with Sloan's algorithm. The design of our new code MC63 is discussed in Section 5. Numerical experiments on a range of practical problems are reported in Section 6. Results illustrating the use of MC63 with the Harwell Subroutine Library frontal solver MA62 are given in Section 7, and some concluding comments are made in Section 8.

2. FINITE ELEMENT GRAPHS

The element resequencing algorithms that we use in this paper are based on the method of Sloan.[7,16] The method, which has been widely used during the last decade for profile reduction, exploits the close relationship between a matrix $\mathbf{A} = \{a_{ij}\}$ of order n with a symmetric sparsity pattern and its undirected graph with n nodes. Two nodes i and j are neighbours (or are adjacent) in the graph if and only if a_{ij} is non-zero. A finite-element mesh is a collection of finite elements in which elements are joined at their common boundaries and vertices. Finite-element nodes may lie at vertices, along the sides, on the faces, or within the element itself. Associated with each finite-element node is a set of variables corresponding to the freedoms at that node. The finite-element mesh with its degrees of freedom can be transformed into the graph of the assembled finite-element matrix and, for convenience, we call this the *variable connectivity* graph. The nodes of the variable connectivity graph are the variables defined on the finite-element mesh, and the edges are constructed by making the variables of each element pairwise adjacent.

In many finite-element problems, there are a number of freedoms at each node of the finite-element mesh. The nodes of the *element clique* graph correspond to the finite-element nodes, and two nodes are adjacent if they belong to the same element (see, for example, Kaveh and Roosta[17]). Provided the list of variables in each node is recorded, the element clique graph provides a more compact representation of the finite-element problem than the variable connectivity graph. Further savings can be achieved by recognizing that some finite-element nodes may belong to the same set of elements. This can be exploited through the use of supervariables. A *supervariable* is a collection of one or more variables, such that each variable belongs to the same set of finite elements. The finite-element mesh can be transformed into a *supervariable connectivity* graph, whose nodes are the supervariables and whose edges are formed by making the supervariables of each finite element pairwise adjacent. For problems in which the number of supervariables is substantially less than the number of variables, Sloan's algorithm is much more efficient if supervariables are used. Reid and Scott[13] report results that illustrate this.

For finite-element problems, Sloan's method may also be applied to the element connectivity graph, in which the nodes are the finite elements. There is more than one way in which the element connectivity may be defined. Bykat[2] generates the element connectivity graph by defining two elements to be adjacent to one another whenever they share a common edge and describes his algorithm in detail for planar triangular elements. This definition was generalized by Fenves and Law[6] to problems in k dimensions ($k = 1, 2, 3$), by defining two elements in k dimensions to be adjacent whenever they possess a common boundary of $k - 1$ dimensions. The resulting graph is termed the *dual graph*[9] or the *natural associate* graph.[17] The main advantage of the dual graph is its economy in terms of data storage because the number of edges is generally substantially fewer than in the variable or supervariable graphs. A disadvantage is that the adjacency of elements cannot always be completely represented by this definition of adjacent elements, since k-dimensional elements are not necessarily connected through $(k - 1)$-dimensional boundaries. In addition, adjacent finite elements do not necessarily have the same dimensionality. In such examples, the dual graph may become disconnected, and each component must be numbered independently. This contributes to the difficulties associated with attempting to implement this algorithm.

A more convenient way of defining element adjacency is to define two elements to be adjacent whenever they have one or more variables in common. The resulting graph is termed the *element communication* graph and has been used by, for example, Duff *et al.*[8] and Paulino *et al.*[9] In the literature, this graph has also been called the *incidence graph*.[17] Throughout the remainder of this paper, the element connectivity graph will refer to the element communication graph.

Other graphs that attempt to embed the connectivity properties of finite element problems and that can be used for deriving profile and wavefront reduction algorithms have been proposed (for example, by Kaveh and Roosta[17] and Kaveh[10]). However, in terms of the quality of element orderings they produce, they do not appear to offer any consistent advantage over the supervariable and element connectivity graphs.

3. SLOAN'S ALGORITHM

In this Section, we give a brief outline of Sloan's algorithm for profile reduction and discuss how the method can be extended for element reordering. Here and elsewhere we assume that the variable connectivity and the element connectivity graphs are connected. If not, it is straightforward to apply the algorithm to each component, and all our software allows for this.

3.1. *The basic algorithm*

Sloan's algorithm for reordering the nodes of a connected undirected graph has two distinct phases:

1. selection of start and end nodes
2. node reordering

In the first phase, the start and end nodes are chosen to be the endpoints of a pseudodiameter. Sloan finds a pseudodiameter using a modified version of the Gibbs-Poole-Stockmeyer algorithm.[18] This has recently been improved further by Reid and Scott[13] (see also the work of Souza and Murray[19]). During the second phase of Sloan's algorithm, the pseudodiameter is used to guide the reordering. One end s of the pseudodiameter is used as the start node and the other e is used as the target end node. Sloan ensures that the position of a node in his ordering is not far from one for which the distance from the target end node is monotonic decreasing. He is able to improve the profile and wavefront by localized reordering. Sloan begins at the start node s and uses for each node i the priority function

$$P_i = -W_1 c_i + W_2 d(i,e) \tag{6}$$

where W_1 and W_2 are integer weights, c_i (the *current degree*) is the amount that the wavefront will increase if node i is numbered next, and $d(i, e)$ is the distance to the target end node. At each stage, the next node in the ordering is chosen from a list of eligible nodes to maximize P_i. The list of eligible nodes comprises the neighbours of nodes that have already been numbered and their neighbours. A node has a high priority if it causes either no increase or only a small increase to the current wavefront and is at a large distance from the target end node. Thus, a balance is kept between the aim of keeping the number of nodes in the front small and including nodes that have been left behind (further away from the target end node than other candidates). At each stage, Reid and Scott[13] give maximum priority to all nodes i with $c_i = 0$.

Following numerical experimentation, Sloan recommends the pair (2, 1) for the weights. However, the results of Kumfert and Pothen[12] and Reid and Scott[13] indicate that, for some problems, there are considerable advantages in using other values. In particular, the choice (16, 1) can yield much smaller profiles. To allow the user to experiment with different choices of the weights, the new profile reduction code MC60 of Reid and Scott has weights that are input parameters.

3.2. *Sloan's algorithm for indirect element ordering*

Sloan's algorithm may be used to reorder elements by applying the method to the variable connectivity graph and then resequencing the elements in ascending order of their earliest variable in the new variable order. In most finite-element problems, the number of supervariables (see Section 2) is significantly less than the number of variables. In such cases it is more efficient to apply a modified version of the Sloan algorithm to the supervariable connectivity graph. The modifications to the Sloan algorithm take into account the number of variables associated with each supervariable (see Duff *et al.*[8]).

3.3. *Sloan's algorithm for direct element ordering*

An alternative approach to element reordering is to apply Sloan's algorithm directly to the element connectivity graph. The main disadvantage of this is that the number of variables in each element is not taken into consideration. To allow for finite-element meshes comprising finite elements with different numbers of freedoms, Duff *et al.*[8] looked at modifying the priority function in the second phase of the algorithm. An element is said to be 'active' if it has been assembled but has one or more unassembled neighbours. In an attempt to reduce both the number of elements that are active at each stage of the frontal method and the number of partially summed variables, Duff *et al.* define the priority of element i to be

$$P_i = -W_1 ngain_{\mathrm{i}} + W_2 d(i, e) - W_3 nadj_{\mathrm{i}} \tag{7}$$

Here $ngain_{\mathrm{i}}$ *is the number of variables element i* will introduce into the front less the number that can be eliminated, and $nadj_{\mathrm{i}}$ is the number of elements adjacent to element i that have not yet been relabelled. The weights used by Duff *et al.* in the Harwell Subroutine Library code MC43 are (10, 5, 1). If assembling element i leads to the elimination of a single variable j, then $ngain_{\mathrm{i}} = c_j$, where c_j is defined as in equation (6). In this case, the priority function (7) is Sloan's function with a third weight to resolve ties. If every element leads to such an elimination, we have another implementation of the Sloan variable ordering algorithm (with a tie-breaking strategy). In general, however, this will not be the case and the algorithm is therefore different but closely related.

4. SPECTRAL AND HYBRID SLOAN REORDERING ALGORITHMS

4.1. *Spectral reordering*

Spectral algorithms have been used in recent years for matrix profile and wavefront reduction. Barnard *et al.*[14] describe a spectral algorithm that associates a Laplacian matrix **L** with the given matrix **A** with a symmetric sparsity pattern.

$$\mathbf{L} = \{l_{ij}\} = \begin{cases} -1 & \text{if } i \neq j \text{ and } a_{ij} \neq 0 \\ 0 & \text{if } i \neq j \text{ and } a_{ij} = 0 \\ \sum_{k \neq i} |l_{ik}| & \text{if } i = j \end{cases} \tag{8}$$

An eigenvector corresponding to the smallest positive eigenvalue of the Laplacian matrix is termed a *Fiedler vector*. The spectral permutation of the variables is computed by sorting the components of a Fiedler vector into monotonically non-increasing or non-decreasing order.

For unassembled finite-element problems, the new variable order can be used to obtain an element ordering. We refer to this as the *indirect spectral* element reordering algorithm. Alternatively, Paulino *et al.*[9] propose constructing the Laplacian matrix associated with the element connectivity graph and reordering the elements by sorting the components of a Fiedler vector of this Laplacian. The results presented by Paulino *et al.* suggest that the method can be effective for finite-element problems but comparisons were only reported with the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms.[20] In our numerical experiments (see Section 6), we call this method the *direct spectral* element reordering method.

4.2. *The hybrid Sloan method*

Kumfert and Pothen[12] observe that spectral orderings do well in a global sense but are often poor locally. They therefore propose using the spectral method to find a global ordering that guides the second phase of Sloan's method. Their results show that this can yield a final ordering with a much smaller profile than using either the spectral method alone or Sloan's method using the Gibbs–Poole–Stockmeyer pseudodiameter. Further experiments by Reid and Scott[13] support this view, particularly for very large problems. The so-called *hybrid Sloan method* uses a priority function in which the distance $d(i, e)$ from the target end node is replaced by p_i, the position of node i in the spectral ordering. Specifically, for a graph with n nodes, Reid and Scott[13] use the priority function

$$P_i = -W_1 c_i - W_2 (h/n) p_i \tag{9}$$

where h is the number of level-sets in the level-set structure rooted at the first node. Reid and Scott recommend computing orderings for the pairs of weights (1, 2) and (16, 1) and choosing the one with the smallest profile. This is the default in the HSL code `MC61`, which provides a driver for `MC60`.

Kumfert and Pothen[12] and Reid and Scott[13] use the hybrid Sloan method to reorder assembled matrices. In the present study, we are concerned with unassembled matrices. We can extend the hybrid method to this class of problems in one of two ways:

1. In the indirect algorithm, use the hybrid method to order the variables. We will refer to this method as the *indirect hybrid* algorithm. In practice, for efficiency, the spectral variable ordering is mapped to a spectral supervariable ordering and the supervariable connectivity graph is used.
2. Apply the spectral method to the element connectivity graph. In the second phase of Sloan's algorithm, replace (7) with the priority function

$$P_i = -W_1 ngain_{\mathrm{i}} - W_2 (h/nelt) pelt_{\mathrm{i}} - W_3 nadj_{\mathrm{i}} \tag{10}$$

where *nelt* is the number of elements, h is the number of level-sets in the level-set structure rooted at the start element, and $pelt_{\mathrm{i}}$ is the position of element i in the direct element spectral ordering. We will call this the *direct hybrid* algorithm.

We remark that, although in our experiments we only use the spectral orderings in the hybrid algorithms, any input ordering can be used. Our software is written to allow this.

5. SOFTWARE DESIGN

In this Section we discuss the design of our new package, MC63, for ordering finite elements. Our new subroutines are named according to the naming convention of the Harwell Subroutine Library.[15]

There are three entries to MC63:

- MC63I sets default values for the control parameters. It should normally be called once prior to calling MC63A. The control parameters include stream numbers for diagnostic printing and parameters that determine whether or not supervariables are to be used and whether the user wishes to supply a global priority function $p = (p_i)$ or $pelt = (pelt_i)$.
- MC63A reorders the elements. The user chooses whether a direct or an indirect algorithm is implemented.
- MC63B computes, for a given element order, the maximum wavefront, the profile, and the root-mean-square wavefront. An option exists for checking the input data.

Full details of the calling sequence and the argument lists are given in the Specification Sheets.[21] Note that we work only with the pattern of the matrix. Thus for matrices that are not positive definite, the actual factorization may be more expensive and require more storage than is indicated by MC63B. We now look in more detail at the reordering routine MC63A.

MC63A accepts lists of variables belonging to the elements and, after performing initial checks on the user's data, calls MC63B to compute statistics for the natural element order, $1, 2, \ldots, nelt$. At this point, MC63B also checks the element variable lists for out-of-range and duplicated indices. If such entries are found, they are either removed and the computation continues after issuing a warning message or terminates, if this has been requested.

If supervariables are wanted, they are constructed using the Harwell Subroutine Library profile reduction package MC60. Otherwise, each variable is treated as a supervariable. The element variable lists are overwritten by element supervariable lists. A map of variable to supervariable indices allows the user to later restore the element variable lists, if desired.

For each supervariable, the number of elements involving it is counted. Lists of the elements associated with the supervariables are then constructed. If the user has selected a direct element reordering algorithm, the element connectivity graph is constructed from the supervariable lists, otherwise the supervariable connectivity graph is constructed from the element lists.

In the indirect element reordering algorithm, MC60C is used to reorder the supervariables. The user may optionally specify a global priority vector whose components p_i are used in the priority function (9). Once the supervariables have been reordered, the elements are resequenced in ascending order of their earliest supervariable in the new supervariable order. The new supervariable indices are not preserved.

In the direct element ordering algorithm, the element connectivity graph is relabelled using either the priority function (7) or (10). If (7) is used, the start and target end nodes (s, e) are computed using the modified Gibbs–Poole–Stockmeyer algorithm (MGPS) of Reid and Scott.[13] Again, MC60 is used for this. To use (10), the user must supply an element global priority vector $pelt$.

Having chosen the start and target end elements, the start element is numbered first and a list of elements that are eligible to be numbered next is formed. When selecting the element with highest priority for renumbering next from the list of eligible elements, a simple sequential search is performed while the list is less than a given threshold and a switch to a binary heap search is made once the list exceeds this threshold. As in MC60, a threshold of 100 is used. Management of the list of eligible elements is discussed in detail by Reid and Scott.[13]

A final call to MC63B (without error checking) computes statistics for the new element order.

We remark that, in MC63, equations (7) and (10) do not define the priority function fully since we give maximum priority to any element that will introduce no new variables into the front, that is, to elements i with $ngain_i = 0$.

6. NUMERICAL RESULTS

In this Section, we first describe the 15 problems that we use for testing the element reordering algorithms discussed in this paper and then present numerical results.

6.1. *Test problems*

Each of the test problems arises from a real engineering or industrial application. A brief description of each problem is given in Table I. The number of unassembled element problems available in the Harwell–Boeing Collection[22] is limited and all are small by today's standards. Consequently, we have only selected two representative problems, cegb3306 and lock3491, from this Collection. Problem ramage02 was provided by Alison Ramage of the University of Strathclyde,[23] aeac5081 came from Andrew Cliffe of AEA Technology, and cham and tubu were from Ron Fowler of the Rutherford Appleton Laboratory. The remaining problems were supplied by Christian Damhaug of Det Norske Veritas, Norway. For cham and tubu, only lists of

Table I. The test problems

Identifier	Degrees of freedom	Number of super-variables	Number of elements	Description/discipline
aea5081	5801	1637	800	Double glazing problem
cegb3306	3222	537	791	2·5D framework problem
cham	12,834	12,834	11,070	Part of an engine cylinder
crplat2	18,010	3004	3152	Corrugated plate field
fullb	199,187	33,442	59,738	Full-breadth barge
lock3491	3416	702	684	Cross-cone vehicle structure
opt1	15,449	3802	977	Part of oil production platform
mt1	97,578	17,044	5328	Tubular joint
ramage02	1476	4939	1400	Navier–Stokes and continuity equations
shipsec1	140,874	23,479	41,037	Full-breadth section of ship
srb1	54,924	9154	9240	Space shuttle rocket booster
thread	29,736	8838	2176	Threaded connector
trdheim	22,098	2868	813	Mesh of the Trondheim fjord
tsyl201	20,685	2881	960	Part of oil production platform
tubu	26,573	26,573	23,446	Engine cylinder model

supervariables belonging to each element were available, so for these problems the number of variables is equal to the number of supervariables.

When testing the element ordering algorithms, the elements were input in random order. Our old code MC43 and the new code MC63 are written in standard Fortran 77, and all the results presented in this Section were obtained using the EPC (Edinburgh Portable Compilers, Ltd) Fortran 90 compiler with optimization -O running on a 143 MHz Sun Ultra 1. In our experiments involving the spectral method, the Fiedler vector was obtained using Chaco 2.0.[24] We used the SymmLQ/RQI option, and the input parameters were chosen to be the same as those used by Kumfert and Pothen.[12] Note that we do not include timings for the hybrid methods because the Chaco package is written in C and the Harwell Subroutine Library does not currently have a Fortran code for computing the Fiedler vector.

6.2. MC43 *vs.* MC63

In Tables II and III we compare the performance of the old code MC43 with that of the new code MC63. Results are given for both the direct and indirect algorithms, using the weights (10, 5, 1) and (2, 1), respectively. In Table II, the maximum and root-mean-square wavefronts are given, and in Table III, timings are presented. As expected, the codes generally yield orderings of comparable quality. The differences are attributable to the differences in the implementations of the algorithms. For example, the two codes handle supervariables in a slightly different manner. MC63 takes the numbers of variables in the supervariables into account when calculating the width of a level-set structure but only MC43 allows for the numbers of variables in the supervariables when calculating the degrees of the supervariables in the list of potential start nodes. The new code also uses a slightly different modification of the Gibbs–Pool–Stockmeyer algorithm when choosing start and end nodes.

Sloan[7] observed that the binary heap search is the method of choice when the root-mean-square wavefront exceeds several hundred nodes, and for smaller problems a simple sequential

Table II. The maximum and root-mean-square wavefronts found by MC43 and MC63

	MC43				MC63			
Identifier	Direct		Indirect		Direct		Indirect	
aea5081	149	180·7	190	119·1	156	180·2	175	116·5
cegb3306	78	60·3	114	73·9	78	60·4	78	60·3
cham	412	333·0	412	313·1	412	331·1	412	332·9
crplat2	538	376·3	560	328·0	392	292·8	470	358·9
fullb	3130	2170·8	3098	2134·0	3142	2172·3	2826	2021·2
lock3491	209	135·5	181	118·1	203	126·5	266	137·5
opt1	1006	619·9	883	544·1	1006	619·3	804	530·1
mt1	2391	1366·6	2067	1259·8	2895	1546·2	2436	1335·4
ramage02	1452	1289·3	1502	1333·3	1452	1289·3	1472	1328·5
shipsec1	3810	2613·4	4308	2702·5	3840	2629·1	3834	2493·6
srb1	546	320·9	546	318·2	546	320·9	546	318·2
thread	3117	2214·3	3630	2413·2	3117	2214·6	2877	1962·0
trdheim	348	172·4	324	139·1	348	172·4	324	146·3
tsyl201	540	511·2	534	511·2	540	511·2	696	505·3
tubu	638	407·2	863	449·4	630	406·6	848	444·3

Table III. A comparison of CPU times for MC43 and MC63 (Sun Ultra)

	MC43		MC63	
Identifier	Direct	Indirect	Direct	Indirect
aea5081	0·50	0·60	0·50	0·60
cegb3306	0·02	0·02	0·02	0·02
cham	1·75	1·34	0·97	0·79
crplat2	0·18	0·18	0·19	0·18
fullb	8·31	5·45	3·88	2·55
lock3491	0·03	0·03	0·03	0·03
mt1	2·05	7·55	1·82	3·61
opt1	0·13	0·32	0·12	0·32
ramage02	0·27	0·75	0·21	0·59
shipsec1	20·6	16·1	7·57	5·72
srb1	1·75	1·91	1·58	1·52
thread	0·45	1·94	0·33	0·83
trdheim	0·08	0·13	0·09	0·13
tsyl201	0·10	0·16	0·11	0·16
tubu	4·41	3·22	2·70	1·62

search is faster. The method we use of commencing with code that performs a simple search, and switches to code that uses a binary heap if the number of eligible nodes exceeds a threshold, ensures MC63 is as efficient as MC43 on small problems, but is substantially faster on large problems (see, for example, fullb, shipsec1 and tubu).

6.3. *Adjusting the weights*

In this Section, we consider the effect of adjusting the weights in the priority function. As already mentioned, Duff *et al.* recommend the weights (10, 5, 1) but Kumfert and Pothen[12] suggest that, for some problems, other values give much better results. In our first test, we compare $W_3 = 1$ in the direct reordering priority functions (7) and (10) with $W_3 = 0$. The weights (W_1, W_2) are given the values of (10, 5) for the Sloan method and, following Reid and Scott,[13] (5, 10) for the hybrid method. Our findings are presented in Table IV. The results suggest that for Sloan there can be a slight advantage in using a third weight to resolve ties but, in general, the difference in the root-mean-square wavefront between using $W_3 = 0$ and $W_3 = 1$ is small (less than 2 per cent). For the hybrid method, the results do not support the use of a third weight.

We have examined the wavefronts for the direct ordering algorithms for $(W_1, W_2) = (5w_1, 5w_2)$ with (w_1, w_2) equal to each of the 13 pairs (1, 64), (1, 32), (1, 16), ..., (1, 1), (2, 1), ..., (64, 1) on all the test matrices. Our findings are shown in Table V. In this Table we show the percentage increases in the root-mean-square wavefront from the best value when the recommended weights of (10, 5, 1) for Sloan and (1, 2, 0) for the hybrid Sloan method are used.

We see that, for many problems, the recommended weights give root-mean-square wavefronts that are within 5 per cent of the minimum value. However, for each method there are a small number of problems for which weights other than the recommended ones give significant improvements. A closer look at the results reveals that the weights that give minimum wavefronts differ with the problem and method. For example, direct Sloan applied to trdheim has the smallest

Table IV. Root-mean-square wavefronts with $W_3 = 0,1$

	Sloan		Hybrid	
Identifier	$W_3 = 0$	$W_3 = 1$	$W_3 = 0$	$W_3 = 1$
aea5081	117·4	108·2	108·3	108·2
cegb3306	59·8	60·3	65·3	65·3
cham	330·6	331·1	329·6	332·6
crplat2	292·1	292·8	238·0	238·0
fullb	2145·6	2172·3	1879·2	1940·8
lock3491	125·5	126·5	208·0	213·2
mt1	1548·8	1546·2	954·8	960·2
opt1	621·6	619·3	537·5	536·4
ramage02	1289·3	1289·3	1321·9	1358·8
shipsec1	2643·3	2629·1	2133·6	2192·5
srb1	320·9	320·9	344·9	350·0
thread	2224·1	2214·6	1621·8	1860·2
trdheim	172·4	172·4	148·1	149·6
tsyl201	511·2	511·2	511·5	511·6
tubu	411·8	406·6	403·6	408·3

Table V. Percentage increases in the root-mean-square wavefront above the minimum value when the recommended weights are used

	Sloan		Hybrid	
Identifier	Direct	Indirect	Direct	Indirect
aea5081	0·4	0·0	0·1	0·1
cegb3306	0·6	0·3	3·9	3·5
cham	0·6	0·0	0·4	3·0
crplat2	0·6	9·5	0·2	0·7
fullb	1·1	0·0	0·0	1·1
lock3491	0·0	10·3	8·5	0·0
mt1	21·1	4·5	0·0	0·0
opt1	11·3	1·4	11·1	0·0
ramage02	2·3	0·1	5·0	16·7
shipsec1	0·3	4·1	0·0	0·7
srb1	0·0	0·0	0·0	0·0
thread	103·7	75·5	50·3	70·1
trdheim	25·3	0·1	2·6	0·0
tsyl201	0·0	0·7	0·0	0·0
tubu	0·0	0·0	4·6	2·4

root-mean-square wavefront when $(w_1, w_2) = (1, 2)$ but for problem thread, the minimum is achieved with the pair (1, 8). For thread, each of the methods performs poorly with the recommended weights and does much better with a larger value for W_2. To allow the user to experiment with different weights, in MC63 the weights are input parameters under the user's control.

6.4. *Sloan vs. hybrid Sloan*

In Table VI, we present root-mean-square wavefronts for the different algorithms discussed in this paper. The weights recommended in Section 6.3 are used. For purposes of comparison, we

Table VI. Root-mean-square wavefronts with different algorithms

Identifer	Original order	Spectral Direct	Spectral Indirect	Sloan Direct	Sloan Indirect	Hybrid Direct	Hybrid Indirect
`aea5081`	142·2	129·4	**108·2**	**108·2**	116·5	**108·2**	**108·4**
`cegb3306`	245·9	100·4	95·0	**60·3**	**60·5**	65·3	64·9
`cham`	769·0	346·1	353·6	**331·1**	**332·9**	**329·5**	**338·9**
`crplat2`	1178·1	257·2	251·7	292·8	358·7	**238·0**	**242·0**
`fullb`	95,181	2009·6	1968·4	2172·3	2021·2	**1879·2**	**1865·7**
`lock3491`	583·0	227·4	128·0	126·5	135·4	208·0	**103·7**
`mt1`	8196·3	1191·1	1191·0	1546·2	1335·4	**954·8**	**1017·6**
`opt1`	2067·7	573·7	604·9	619·3	**530·1**	**536·4**	557·0
`ramage02`	1492·3	1335·2	1403·1	**1289·3**	1328·4	1321·9	1515·6
`shipsec1`	52,642	2292·5	1750·8	2629·1	2493·6	2133·6	**1554·8**
`srb1`	1506·3	380·7	370·8	**320·9**	**318·2**	344·9	344·6
`thread`	4110·2	**1334·7**	**1226·0**	2214·6	1962·0	1621·8	1857·5
`trdheim`	181·9	163·3	156·0	172·4	**146·3**	**148·1**	**144·8**
`tsyl201`	861·6	530·0	513·6	**511·2**	**505·3**	**511·5**	**502·7**
`tubu`	1298·6	470·1	460·9	406·6	447·2	403·6	**393·2**

include results for the original element order. We note that, in most instances, this order was thought, by the originator of the problem, to be a 'good' element order. In Table VI, we highlight in bold the smallest root-mean-square wavefront for each problem, and any within 2% of the smallest. We see that, if the elements are not originally well ordered, both the direct and indirect spectral algorithms can substantially reduce the root-mean-square wavefront. However, comparison of columns 3 and 7 and columns 4 and 8 demonstrate that, in general, it is worthwhile to use Sloan's method to refine the spectral ordering. By looking also at Table V, we observe that the only problems where the hybrid method gives poorer results than the corresponding spectral method are those for which the recommended weights give a root-mean-square wavefront that is far from the minimum. For these problems, the hybrid Sloan method becomes competitive if other weights are used. For example, if we use the weights (1, 16) in the indirect hybrid algorithm in place of the recommended values of (1, 2), the root-mean-square wavefront for `thread` reduces to 1092·1, and this is smaller than the indirect spectral root-mean-square wavefront. Note that, if the recommended weights are used, the Sloan and hybrid Sloan algorithms still provide substantial improvements over the original ordering for this problem.

The hybrid method is primarily intended for very large problems and, on the basis of the results we have obtained, we see that the hybrid Sloan method can significantly out-perform Sloan's method for problems with a large number of elements (for example, `fullb` and `shipsec1`). We conclude from our empirical evidence that the user may wish to reorder the finite elements using either a direct or an indirect algorithm and may wish to use a hybrid Sloan method. `MC63` allows each of these options to be selected.

7. ELEMENT ORDERINGS AND FRONTAL SOLVERS

We have looked at using variants of Sloan's algorithm to reorder finite elements. Both direct and indirect reordering algorithms have been considered and used in combination with spectral orderings. As discussed in the introduction, the main motivation behind this work was the need for element orderings that are efficient when used with a frontal solver. In this Section, we present results of using the MC63 element orderings with a frontal solver. In the Harwell Subroutine Library we have two frontal codes for real matrices: MA42[25] for general unsymmetric problems and MA62[26] for symmetric positive-definite systems. Both codes are designed for unassembled finite-element matrices, although MA42 does include an option for entering the assembled matrix row-by-row. The matrix factors may optionally be held in direct access files. For efficiency, Level 3 BLAS kernels are used in the innermost loop of the matrix factorization.

The element ordering schemes we have considered work only with the pattern of the finite-element matrices. They are therefore most useful for positive definite matrices. For more general matrices, the need to preserve numerical stability may lead to the actual factorization being more expensive and requiring more storage. To illustrate the effectiveness of our element ordering algorithms, in this Section we present results for MA62, using the user-supplied element order and the MC63 element order. As discussed in Section 5, MC63 offers both a direct and indirect version of Sloan's algorithm and a direct and indirect hybrid Sloan algorithm. We saw in Table VI that the method that gives the smallest root-mean-square wavefront is problem dependent so we have chosen the best ordering for each problem (obtained using the recommended weights). Default values are used for all MA62 control parameters.

The experimental results quoted in Table VII were obtained on a single processor of a Cray J932 using 64-bit floating-point arithmetic, the Cray f90 Fortran compiler (with default options), and the vendor-supplied BLAS. For each problem, values for the entries of the matrix were generated using the HSL pseudo-random number generator FA01 and each was made symmetric and positive definite. The times quoted include the i/o overhead for using direct access files for the matrix factors. The storage is the total storage for the factors and includes both real and integer storage. Since, on the Cray, both integers and reals are stored in 64-bit words, this value is just the sum of the number of real and the number of integer words needed. The number of floating-point operations ('ops') counts all operations (+, −, *, /) equally. The 'Solve' times are for a single right-hand side.

Table VII. The results of using MC63 with the frontal solver MA62 (Cray J932)

	Factorization time (seconds)		Solve time (seconds)		Number of ops ($\times 10^7$)		Storage (Kwords)	
Identifier	Before	After	Before	After	Before	After	Before	After
aea5081	1·5	1·1	0·09	0·07	12·4	7·5	786	585
cegb3306	0·64	0·45	0·03	0·03	3·1	1·7	232	219
cham	53·9	13·1	0·95	0·46	766·2	152	9689	4472
crplat2	126	11·3	1·7	0·49	1919	113	17,985	4583
fullb	NS	5028	NS	35·6	NS	79,695	NS	405,607
lock3491	4·0	0·99	0·11	0·06	51·3	7·2	1123	448
mt1	NS	1118	NS	11·1	NS	17,661	NS	125,694
opt1	54·5	31·6	0·9	0·7	857	437	10,031	7596
ramage02	233	175	2·2	1·8	3812	2852	25,547	21,847
shipsec1	NS	5450	NS	28·7	NS	86,304	NS	333,791
srb1	550	48·9	5·9	1·7	8604	595	65,894	18,058
trdheim	5·7	4·6	0·34	0·32	51·8	36·6	2872	2389
thread	1925	680	7·0	4·5	32,996	11,301	92,048	55,207
tsyl201	98·1	38·3	1·5	1·0	1528	535	17,236	10,654
tubu	122	35·2	1·7	1·1	1858·9	440·0	17,277	10,742

Cases marked NS were not solved.

The results demonstrate the importance of reordering the elements and illustrate that, for problems that are not initially well ordered, substantial savings can be achieved by using MC63. We were not able to run the largest problems without reordering because of the memory and CPU times they required. Where there is a significant reduction in the root-mean-square wavefront it is reflected in much lower factorization and solve times, operation counts and factor storage, although we note that the effect of using Level 3 BLAS means that the poorer orderings can have a higher Megaflop rate so that, for some problems, the ratio of times, before and after ordering, is not as high as the operation count ratio. Furthermore, MA62 is partly able to offset the effect of a poor ordering by exploiting zeros within the frontal matrix.[26,27] For example, we note that, for the problem cegb3306, the root-mean-square wavefront is reduced by a factor of 4 (see Table VI) using MC63 but the saving in factor storage is small. This is because the root-mean-square wavefront assumes that the frontal matrix is dense but MA62 is able to take some advantage of zeros in the front.

8. CONCLUSIONS

In this paper, we have looked at using variants of Sloan's algorithm to reorder finite elements for use with a frontal solver. Both direct and indirect versions of the reordering algorithm have been considered and used in combination with spectral orderings. We found that there can be a difference in the quality of the ordering obtained using the direct or the indirect method, but for a given problem we do not know a priori whether the direct or indirect method will give the best reduction in the wavefront. Our results suggest that the hybrid Sloan method is superior to the spectral method and generally out-performs Sloan for large problems but offers no consistent advantage for smaller problems. The choice of weights in the priority function can also influence the quality of the ordering.

A disadvantage of the hybrid method is the need to compute a global priority function. The time taken to compute a spectral ordering is significantly more than that needed to compute start and end nodes for the Sloan algorithm, and depends upon the algorithm used to compute the Fiedler vector. For this reason, if the tradeoff between the quality of the ordering and the time taken for computing the ordering favours fast reordering algorithms, the Sloan algorithm may be preferred, with the direct method selected if the number of elements is less than the number of supervariables and the indirect method used otherwise. However, in our experiments with the frontal method, the time required to compute an element ordering was small compared with that needed by the matrix factorization and solve steps. For large problems, it may therefore be worthwhile experimenting with the options offered by MC63 before using the frontal solver, particularly if a number of factorizations are to use the same element ordering.

The code MC63 is available for use under licence and will be included in the next release of the Harwell Subroutine Library. Anyone interested in using the code should contact the author for details.

ACKNOWLEDGEMENTS

I would like to thank my colleagues John Reid and Iain Duff at the Rutherford Appleton Laboratory for their interest and comments on a draft of this paper.

REFERENCES

1. J. E. Akin and R. M. Pardue, 'Element resequencing for frontal solutions', in *Mathematics of Finite Elements and Applications*, J. R. Whiteman (Ed.), Academic Press, 1975.
2. A. Bykat, 'A note on an element ordering scheme', *Int. j. numer. methods eng.*, **11**, 194–198 (1977).
3. A. Razzaque, 'Automatic reduction of frontwidth for finite element analysis', *Int. j. numer. methods eng.*, **15**, 1315–1324 (1980).
4. H. L. Pina, 'An algorithm for frontwidth reduction', *Int. j. numer. methods eng.*, **17**, 1539–1547 (1981).
5. S. W. Sloan and M. F. Randolph, 'Automatic element reordering for finite-element analysis with frontal schemes', *Int. j. numer. methods eng.*, **19**, 1153–1181 (1983).
6. S. J. Fenves and K. H. Law, 'A two-step approach to finite element ordering', *Int. j. numer. methods eng.*, **19**, 891–911 (1983).
7. S. W. Sloan, 'An algorithm for profile and wavefront reduction of sparse matrices', *Int. j. numer. methods eng.*, **23**, 1315–1324 (1986).
8. 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).
9. G. H. Paulino, I. F. Menezes, M. Gattass 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).
10. A. Kaveh, *Optimal Structural Analysis*, Research Studies Press, John Wiley, England, 1997.
11. 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).
12. G. Kumfert and A. Pothen, 'Two improved algorithms for envelope and wavefront reduction', *BIT*, **18**, 559–590 (1997).
13. J. K. Reid and J. A. Scott, 'Ordering symmetric sparse matrices for small profile and wavefront'. *Technical Report RAL-TR-98-016*, Rutherford Appleton Laboratory, 1998. To appear in *Int. j. numer. methods eng.*
14. S. T. Barnard, A. Pothen and H. Simon, 'A spectral algorithm for envelope reduction of sparse matrices', *Numer. Lin. Algebr. Appl.*, **2**, 317–398 (1995).
15. Harwell Subroutine Library, *A Catalogue of Subroutines (Release 12)*. Advanced Computing Department, AEA Technology, Harwell Laboratory, Oxfordshire, England, 1995.
16. S. W. Sloan, 'A Fortran program for profile and wavefront reduction', *Int. j. numer. methods eng.*, **28**, 2651–2679 (1989).
17. A. Kaveh and G. R. Roosta, 'Comparitive study of finite element nodal ordering methods', *Eng. Struct.*, **20**, 86–96 (1997).
18. N. E. Gibbs, W. G. Poole and P. K. Stockmeyer, 'An algorithm for reducing the bandwidth and profile of a sparse matrix', *SIAM J. Numer. Anal.*, **13**, 236–250 (1976).
19. L. T. Souza and D. W. Murray, 'An alternative pseudopheripheral node finder for resequencing schemes', *Int. j. numer. methods eng.*, **36**, 3351–3379 (1993).
20. J. G. Lewis, 'Implementation of the Gibbs–Poole–Stockmyer and Gibbs–King algorithms', *ACM Trans. Math. Softw.*, **8**, 180–189 (1982).
21. J. A. Scott, 'On ordering elements for a frontal solver'. *Technical Report RAL-TR-98-031*, Rutherford Appleton Laboratory, 1998.
22. I. S. Duff, R. G. Grimes and J. G. Lewis, 'User's guide for the Harwell–Boeing sparse matrix collection (Release 1)'. *Technical Report RAL-TR-92-086*, Rutherford Appleton Laboratory, 1992.
23. A. L. Ramage and A. J. Wathen, 'Iterative solution techniques for the Navier–Stokes equations'. *Technical Report AM-93-01*, School of Mathematics, University of Bristol, 1993.
24. B. Hendrickson and R. Leland, 'The Chaco user's guide: Version 2.0'. *Technical Report SAND94-2692*, Sandia National Laboratories, Albuquerque, NM, 1995.
25. I. S. Duff and J. A. Scott, 'The design of a new frontal code for solving sparse unsymmetric systems', *ACM Trans. Math. Softw.*, **22**(1), 30–45 (1996).
26. I. S. Duff and J. A. Scott, 'MA62 — a new frontal code for sparse positive-definite symmetric systems from finite-element applications'. *Technical Report RAL-TR-97-012*, Rutherland Appleton Laboratory, 1997.
27. K. A. Cliffe, I. S. Duff and J. A. Scott, 'Performance issues for frontal schemes on a cache-based high performance computer'. *Technical Report RAL-TR-97-001*, Rutherford Appleton Laboratory. 1997. *Int. j. numer. methods eng.*, **42**, 127–143 (1998).