COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING
Commun. Numer. Meth. Engng 2005; **21**:233–245
Published online 14 March 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cnm.740

Multilevel hybrid spectral element ordering algorithms

Jennifer A. Scott*,†

Computational Science and Engineering Department, Atlas Centre, Rutherford Appleton Laboratory, Oxon OX11 0QX, U.K.

SUMMARY

For frontal solvers to perform well on finite-element problems it is essential that the elements are ordered for a small wavefront. Multilevel element ordering algorithms have their origins in the profile reduction algorithm of Sloan but for large problems often give significantly smaller wavefronts. We examine a number of multilevel variants with the aim of finding the best methods to include within a new state-of-the-art frontal solver for finite-element applications that we are currently developing. Numerical experiments are performed using a range of problems arising from real applications and comparisons are made with existing element ordering algorithms. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: large sparse linear systems; finite elements; element ordering; frontal method

1. INTRODUCTION

The frontal method is frequently the method of choice for solving the large sparse systems of linear equations that arise during the solution of finite-element problems. These systems are of the form

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

where the $n \times n$ matrix A is a sum of n_{elt} finite-element matrices

$$A = \sum_{l=1}^{n_{\text{elt}}} A^{(l)} \tag{2}$$

and B is an $n \times n_{\text{rhs}}$ matrix ($n_{\text{rhs}} \geqslant 1$) of known right-hand sides. Each matrix $A^{(l)}$ has non-zeros only in a few rows and columns; $A^{(l)}$ corresponds to the contribution from element l and is

*Correspondence to: J. A. Scott, Computational Science and Engineering Department, Atlas Centre, Rutherford Appleton Laboratory, Oxon OX11 0QX, U.K.
†E-mail: j.a.scott@rl.ac.uk

Contract/grant sponsor: EPSRC; contract/grant number: GR/S42170

Received 28 May 2004
Accepted 9 November 2004

normally held as a small dense matrix. One reason for choosing a frontal method is that only a small amount of main memory is required to solve the problem. This allows very much larger problems to be solved than is possible using a direct solver that works entirely in-core. However, this is only true if it is possible to preorder the elements to ensure small fronts throughout the computation. If a_{ij} and $a_{ij}^{(l)}$ denote the (i,j)th entry of A and $A^{(l)}$, respectively, the basic assembly operation for constructing A is of the form

$$a_{ij} \Leftarrow a_{ij} + a_{ij}^{(l)} \tag{3}$$

The main feature of the frontal method is that the Gaussian elimination operation

$$a_{ij} \Leftarrow a_{ij} - a_{il}[a_{ll}]^{-1}a_{lj} \tag{4}$$

may be performed once all the terms in the triple product in (4) are fully summed. A variable is *fully summed* if it is involved in no further sums of form (3) and is *partially summed* if it has appeared in at least one of the elements assembled so far but is not yet fully summed. Thus by assembling the contributions $A^{(l)}$ from the finite-elements one at a time and (provided numerical stability conditions are satisfied) performing eliminations as variables become fully summed, the construction of the assembled coefficient matrix A is avoided.

At each stage of the assembly and elimination processes, the fully and partially summed variables are held in an in-core *frontal matrix*. In the innermost loop of the numerical factorization, 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. In other words, the elements need to be ordered so that partially summed variables become fully summed as soon as possible. If we denote by f_i the number of variables in the front before the ith elimination, of interest is:

- the maximum wavefront, since this affects the in-core storage needed,
- the sum of the wavefronts

$$\sum_{i=1}^{n} f_i$$

  known as the *profile*, since this determines the total storage needed for the matrix factors, and
- the *root-mean-square wavefront* defined by

$$\left(\sum_{i=1}^{n} f_i^2/n \right)^{1/2}$$

  since the work performed when eliminating a variable is proportional to the square of the current wavefront.

Reflecting the popularity of the frontal method, a number of algorithms for automatically ordering finite elements for small wavefront and profile have been reported on in the literature (see, for example, Scott [1] for references to element ordering algorithms). Duff *et al.* [2] divide element ordering algorithms into direct and indirect algorithms. As the name suggests, direct algorithms order the elements directly while indirect algorithms use a two-step approach in which the variables (or, more usually, the supervariables) are first relabelled and then

used to resequence the elements; the new variable indices are subsequently discarded. Results presented by Duff *et al* [2] suggest that both approaches can be used effectively and in their experiments neither was found to be consistently superior to the other.

Some of the most well-known element ordering algorithms are based on the profile and wavefront reduction algorithm of Sloan [3]. The Sloan algorithm exploits the close relationship between a matrix A of order n with a symmetric sparsity pattern and its undirected graph with n nodes, that is, the adjacency graph $\mathscr{G}(A)$. In particular, it uses the level set structure of $\mathscr{G}(A)$. In the late 1990s, Scott [1] developed an element ordering package `MC63` for inclusion in the HSL mathematical software library [4]. This package provides efficient implementations of a number of variants of Sloan's algorithm. In particular, it offers a hybrid variant in which Sloan's algorithm is used to refine an ordering provided by the user. Numerical results reported by Scott [1] showed that if the user inputs a spectral ordering then, for large problems, the hybrid method is a significant improvement on Sloan's algorithm (that is, it generally produces smaller maximum and root-mean-squared wavefronts). The disadvantage of the hybrid spectral–Sloan algorithm is the need to compute a spectral ordering, which can add significantly to the overall cost of ordering the elements.

Spectral orderings are expensive because they are dependent upon the computation of the eigenvector corresponding to the smallest non-trivial eigenvalue of the Laplacian matrix associated with the graph of the problem, the so-called Fiedler vector [5]. Recently, a new flexible software package that implements both an efficient multilevel algorithm for computing the Fiedler vector and a number of multilevel profile reduction algorithms for sparse matrices with symmetric sparsity patterns has been designed and developed by Hu and Scott [6]. The new Fortran 95 code is called `HSL_MC73` and is included in HSL 2004 [4]. The aim of this article is to report on using `HSL_MC73` to obtain high-quality multilevel element orderings efficiently for use with a frontal solver.

The outline of this article is as follows. In Section 2, the new code `HSL_MC73` is briefly described then, in Section 3, we look at how we can use `HSL_MC73` to obtain a number of multilevel element ordering algorithms. Both direct and indirect variants are proposed and the use of supervariables, which provide an effective initial coarsening of the adjacency graph, is discussed. Numerical results are presented for the multilevel algorithms in Section 4. Finally, some concluding remarks are made in Section 5.

2. A NEW MULTILEVEL PROFILE REDUCTION CODE

We start by briefly describing the new multilevel Fiedler and profile reduction code `HSL_MC73`.

Following the success of spectral orderings for graph partitioning, Barnard *et al.* [7] first proposed using the Fiedler vector to obtain profile reducing orderings for matrices A with symmetric sparsity patterns. Their algorithm is motivated as an attempt to minimize the two-sum

$$\min_{\mathbf{x}\in\mathscr{P}} \left\{ \sum_{\{i<j:\ a_{ij}\neq 0\}} (x_i - x_j)^2 \right\} \tag{5}$$

where $\mathscr{P}$ denotes the set of vectors whose components are permutations of

$$i - (n+1)/2, \quad i = 1, 2, \ldots, n$$

That is,

$$\min_{\mathbf{x}\in\mathscr{P}} \mathbf{x}^{\mathrm{T}} L\mathbf{x} \tag{6}$$

where L is the Laplacian of A given by

$$L=\{l_{ij}\}=\begin{cases}-1 & \text{if } i\neq j \text{ and } a_{ij}\neq 0\\ 0 & \text{if } i\neq j \text{ and } a_{ij}=0\\ \sum_{i\neq j}|l_{ij}| & \text{if } i=j\end{cases} \tag{7}$$

To make this problem tractable, albeit at the expense of not computing a guaranteed optimal solution, a heuristic is introduced. Instead of minimizing over the discrete set $\mathscr{P}$, problem (6) is relaxed to $\mathbf{x}\in\mathscr{R}^n$ with $\mathbf{x}^{\mathrm{T}}\mathbf{e}=0$ ($\mathbf{e}=[1,1,\ldots,1]^{\mathrm{T}}$) and $\|\mathbf{x}\|_2=\|\mathbf{p}\|_2$ for any $\mathbf{p}\in\mathscr{P}$. The solution is then the eigenvector corresponding to the second smallest eigenvalue of L, that is, the Fiedler vector. Applying the permutation induced by ordering the components of this vector into monotonic order to the matrix A gives the so-called spectral ordering. In general, it not only reduces the two-sum but also the profile and wavefront of A.

The main problem with implementing the spectral method is that computing eigenvectors of large matrices is expensive. This led Barnard and Simon [8] to propose a multilevel algorithm for computing the Fiedler vector. The basic multilevel Fiedler algorithm proceeds as follows:

- Starting with the adjacency graph $\mathscr{G}(A)$, a series of graphs of successively coarser (smaller) sizes is generated.
- At some point the graph has so few nodes that it is very cheap to compute the Fiedler vector of the associated Laplacian.
- The coarse graph Fiedler vector is projected from one level to another. At each level some refinement is performed until, finally, an (approximate) Fiedler vector for the original Laplacian is obtained.

In broad terms, this is the algorithm that is implemented within the new software package `HSL_MC73`. A key observation is that, for both graph partitioning and for profile reduction algorithms, it is not necessary to obtain the Fiedler vector to high accuracy; instead an approximate Fiedler vector is sufficient. Thus `HSL_MC73` is designed to compute an approximate Fiedler vector and a number of parameters under the user's control are used in determining how accurate the requested eigenvector is. Full details of the algorithm, its implementation and user interface are given by Hu and Scott [6].

As well as offering a multilevel spectral ordering algorithm for profile reduction, `HSL_MC73` includes implementations of the hybrid Sloan algorithm of Kumfert and Pothen [9] and the multilevel Sloan algorithm of Hu and Scott [10]. The Sloan algorithm for profile and wavefront reduction employs the adjacency graph $\mathscr{G}(A)$ of A and has two distinct phases:

1. Selection of a start node s and a target end node e.
2. Node reordering.

The first phase computes a pseudo-diameter of $\mathscr{G}(A)$ and uses it to provide s and e. In the second phase, the chosen start node is numbered first and a list of nodes that are eligible to be

numbered next is formed. At each stage of the numbering, the list of eligible nodes comprises the neighbours of the nodes that have already been renumbered together with their neighbours. The next node to be numbered is selected from the list of eligible nodes to maximize the priority function

$$P(i) = -W_1 \,\mathrm{inc}(i) + W_2 \,\mathrm{dist}(i, e) \tag{8}$$

where (W_1, W_2) are fixed positive weights. The first term, $\mathrm{inc}(i)$, is the amount by which the wavefront will increase if node i is ordered next. The second term, $\mathrm{dist}(i, e)$, is the distance between nodes i and the end node e. Thus, a balance is maintained between the aim of keeping the wavefront small and bringing in nodes that have been left behind (that is, those far away from the target end node e). A node has a high priority if it causes either no increase or only a small increase to the current wavefront size and is at a large distance from the end node e.

Kumfert and Pothen [9] observed that there are problems for which the spectral algorithm can perform poorly and this motivated them to propose a hybrid method that combines use of the spectral ordering with a modified version of the second phase of Sloan's algorithm. The first term in (8) affects the priority function in a local way, by giving higher priority to nodes that will result in a small (or negative) increase to the current wavefront. This is done in a greedy fashion, without consideration of the long-term effect. The second term acts in a more global manner, ensuring nodes lying far away from the end node are not left behind. The second phase of the Sloan algorithm can therefore be viewed as an algorithm that refines the ordering implied by the distance function $\mathrm{dist}(i, e)$. Thus Kumfert and Pothen modified the second phase of Sloan's algorithm so that, in place of the distance function, it refined the spectral ordering. Their numerical experiments showed that, for large problems, the resulting hybrid method generally gives significantly smaller profiles than those obtained using the standard Sloan algorithm. This led us to design the package `HSL_MC73` to include an option to compute a multilevel spectral ordering which is then refined to obtain the so-called hybrid spectral–Sloan ordering (for further details, see Hu and Scott [6] and Reid and Scott [11]).

The main disadvantage of the hybrid profile reduction method is that it requires significantly more CPU time than Sloan's algorithm because it is more expensive to compute the Fiedler vector than it is to find a pseudo-diameter for A using the (modified) Gibbs–Poole–Stockmeyer algorithm of Reid and Scott [11]. Even if the Fiedler vector is computed as in `HSL_MC73` using a multilevel approach, the hybrid algorithm can be relatively expensive. In an attempt to avoid computation of the Fiedler vector while still maintaining the quality of the hybrid algorithm, Hu and Scott [10] proposed a multilevel version of Sloan's algorithm. Mirroring the multilevel Fiedler algorithm, the multilevel Sloan profile reduction algorithm comprises three separate steps:

- A series of graphs of successively smaller sizes is generated.
- The coarsest graph is reordered using the Sloan algorithm.
- The coarse graph ordering is projected from one level to another by first mapping the ordering for the previous (coarser) level onto the current level and then performing refinement using the second phase of Sloan's algorithm.

Numerical results presented by Hu and Scott confirm that this approach is faster than the hybrid method and, with appropriate coarsening and refinement, produces orderings that are of comparable quality. Thus, in addition to the multilevel spectral and hybrid methods,

HSL_MC73 includes an efficient implementation of the multilevel Sloan algorithm for profile reduction.

3. MULTILEVEL ELEMENT ORDERINGS

The input required by the package HSL_MC73 is the sparsity pattern of the matrix A or, equivalently, the adjacency graph $\mathcal{G}(A)$. In fact, any undirected (unweighted) graph can be input and we use this facility to obtain multilevel element ordering algorithms.

There are a number of possible graphs associated with a finite-element problem that have been used for element resequencing. We consider two that our previous experiments [1, 2] found to be efficient with the Sloan algorithm: the supervariable connectivity graph and the element communication graph. We will input these graphs to HSL_MC73 and use them to obtain multilevel indirect and direct element orderings, respectively.

In the variable connectivity graph the nodes are the variables defined on the finite-element mesh, and the edges are constructed by making the variables of each element pairwise adjacent. This graph is the adjacency graph $\mathcal{G}(A)$ of the assembled finite-element matrix A. However, because in many finite-element problems there are a number of degrees of freedom at each node of the finite-element mesh, a more compact representation of the finite-element problem is generally possible 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 $\mathcal{G}_S$, whose nodes are the supervariables and whose edges are formed by making the supervariables of each finite element pairwise adjacent.

For finite-element problems, element connectivity graphs may be defined in which the nodes are the finite elements. There is more than one way in which the element connectivity may then be defined. We restrict our attention to the *element communication* graph $\mathcal{G}_{EC}$ in which two elements are defined as being adjacent whenever they share at least one variable in common.

3.1. Indirect multilevel element orderings

Our indirect multilevel element ordering algorithms proceed as follows:

- Generate the supervariable graph $\mathcal{G}_S$ of A.
- Apply HSL_MC73 to $\mathcal{G}_S$ to obtain an ordering of the supervariables.
- Order the elements in ascending sequence of their lowest numbered supervariable and then discard the supervariable ordering.

When HSL_MC73 is applied to $\mathcal{G}_S$ we have two options. We can either use the multilevel Sloan algorithm or the hybrid spectral–Sloan algorithm (which employs the multilevel Fiedler algorithm). We have performed experiments with both approaches and include results in Section 4. We can also try and improve the HSL_MC73 supervariable ordering prior to resequencing the elements using the Hager exchange algorithms. Hager [12] suggested two methods for improving any given profile reducing permutation of a symmetric matrix A; a down exchange algorithm and an up exchange algorithm, which he proposed using in an iterative fashion (further details and results illustrating the effectiveness of the exchanges are given

by Reid and Scott [13]). HSL_MC73 includes an option to perform a user-chosen number of down/up exchanges; we include results in Section 4 for using this option within our indirect multilevel element ordering algorithm.

3.2. Direct multilevel element orderings

Using an analogous approach, our direct multilevel element ordering algorithms comprise the following steps:

- Generate the element communication graph $\mathcal{G}_{\mathrm{EC}}$ of A.
- Apply HSL_MC73 to $\mathcal{G}_{\mathrm{EC}}$ to obtain either a multilevel spectral ordering or a multilevel Sloan ordering for $\mathcal{G}_{\mathrm{EC}}$.
- Refine the element ordering using a modified version of the second step of Sloan's algorithm.

The modified version of Sloan's algorithm that we use to refine the element ordering computed by $\mathcal{G}_{\mathrm{EC}}$ is described in detail by Scott [1] and is implemented in the HSL package MC63. Again, Hager exchanges may be used when ordering $\mathcal{G}_{\mathrm{EC}}$ using HSL_MC73.

4. NUMERICAL RESULTS

The numerical experiments reported on in this section were performed on a single Xeon 3.06 GHz processor of a Dell Precision Workstation 650 with 4 GBytes of RAM under the Fedora Core 1 Linux operating system. The NAG Fortran 95 compiler was used with the compiler optimization flag -O. All timings are CPU times, measured using the Fortran 95 routine cpu_time and are given in seconds. Unless otherwise stated, the default values are used for all MC63 and HSL_MC73 control parameters.

The test problems used in our numerical experiments are listed in Table I. The problems range in size from fewer than 1000 elements to more than 70 000 elements with almost 225 000 degrees of freedom. For each problem the order n of A together with the number nsup of supervariables is given. For cham and tubu, only lists of supervariables belonging to each element were available so for these problems $n = \mathtt{nsup}$. For the remaining problems, we note that the number of supervariables is significantly less than the number of variables.

4.1. Use of supervariables

To illustrate the importance of using supervariables, in Table II we present results for the indirect multilevel Sloan algorithm using the variable connectivity graph $\mathcal{G}(A)$ and the supervariable graph $\mathcal{G}_{\mathrm{S}}$. We see that using supervariables leads to large savings in the time required to reorder the elements. For most of our examples, the time is reduced by a factor of more than 10. Furthermore, for many of the problems, the root-mean-square wavefront is significantly smaller if supervariables are used. It appears that the initial coarsening of the variable connectivity graph by using supervariables is generally more effective than the coarsening used within the multilevel code HSL_MC73. Note that, although not given here, the root-mean-square wavefronts for the direct multilevel element ordering algorithms are not affected by using supervariables in place of variables but there is a small time saving if supervariables are used. All results in the remainder of this report are computed using supervariables.

Table I. The test problems.

Identifier	*n*	*nsup*	Elements	Description/discipline
cham	12 834	12 834	11 070	Part of an engine cylinder
crplat2	18 010	3004	3152	Corrugated plate field
fcondp2	201 822	33 913	35 836	Oil production platform
fullb	199 187	33 442	59 738	Full-breadth barge
halfb	224 617	38 556	70 211	Half-breadth barge
inv-ext-2	78 142	19 734	7193	Fluid flow
mt1	97 578	17 044	5328	Tubular joint
opt1	15 449	3802	977	Part of condeep cylinder
ship_001	34 920	5843	3431	Ship structure—predesign
ship_003	121 728	20 287	45 464	Ship structure—production
shipsec1	140 874	23 479	41 037	Section of a ship
shipsec5	179 860	17 260	52 272	Section of a ship
shipsec8	114 919	19 532	32 580	Section of a 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
troll	213 453	48 435	41 084	Structural analysis
tsyl201	20 685	2881	960	Part of condeep cylinder
tubu	26 573	26 573	23 446	Engine cylinder model
x104	108 384	17 260	26 019	Beam joint

n and *nsup* denote the number of variables and supervariables, respectively.

Table II. The root-mean-square wavefronts and times required by the indirect multilevel Sloan algorithm using the variable and supervariable connectivity graphs.

	r.m.s		Time	
Identifier	$\mathscr{G}(A)$	$\mathscr{G}_S$	$\mathscr{G}(A)$	$\mathscr{G}_S$
crplat2	332	260	0.24	0.01
fcondp2	2631	1862	3.25	0.23
fullb	1943	3110	3.32	0.33
halfb	1638	1462	3.49	0.35
inv-ext-2	3378	2272	9.18	0.37
mt1	1339	1626	2.37	0.15
opt1	530	526	0.44	0.04
ship_001	460	450	1.57	0.05
ship_003	1400	1544	3.28	0.22
shipsec1	2398	1686	2.51	0.22
shipsec5	1496	1370	2.80	0.28
shipsec8	2377	1701	1.86	0.19
srb1	318	327	0.71	0.05
thread	2294	1706	1.06	0.13
trdheim	145	161	0.42	0.02
troll	4265	2377	3.71	0.46
tsyl201	505	513	0.57	0.02
x104	1020	1106	3.56	0.11

Table III. The root-mean-square wavefronts computed by the Sloan and multilevel algorithms.

	Sloan (MC63)		Hybrid spectral–Sloan		Multilevel Sloan	
Identifier	Direct	Indirect	Direct	Indirect	Direct	Indirect
cham	**332**	**332**	**334**	**332**	368	691
crplat2	334	327	**234**	**244**	271	257
fcondp2	3024	2667	1827	**1700**	2164	1863
fullb	2172	2021	**1879**	**1833**	2152	3110
halfb	1776	1608	**1411**	**1365**	1500	1462
inv-ext-2	8429*	3379*	8632*	**2272**	8699*	**2272**
mt1	1546	1335	**1018**	1176	1149	1626
opt1	619	**528**	539	557	592	**526**
ship_001	693	**451**	500	**461**	510	**450**
ship_003	1739	1558	**1427**	**1371**	1565	1544
shipsec1	2629	2494	1895	**1444**	1524	1686
shipsec5	1803	1499	**1345**	**1317**	1425	**1370**
shipsec8	2080	2302	1746	**1644**	1818	1701
srb1	**321**	**318**	**334**	**326**	338	**327**
thread	2215	1962	1442	**1122**	1948	1701
trdheim	172	146	147	155	**135**	161
troll	4334*	4229*	3912	3593	2669	**2377**
tsyl201	**511**	**504**	**512**	**512**	**510**	**513**
tubu	**408**	451	**414**	**403**	452	454
x104	1064	1268	**1007**	1061	**989**	1106

*The wavefront is larger than for the original ordering.

4.2. *Sloan vs multilevel algorithms*

In Table III we compare the performance of the Sloan algorithm (as implemented within MC63) with the multilevel algorithms. We include both direct and indirect variants. For the multilevel algorithms, we report results for the hybrid spectral–Sloan algorithm (with the spectral ordering computed using HSL_MC73) and for the multilevel Sloan algorithm (again computed using HSL_MC73 and, for the direct algorithm, refined using MC63). For each problem, the smallest root-mean-square wavefront (and any within 5 per cent of the smallest) are highlighted in bold. Note that when assessing the relative performance of the algorithms we make no distinction between the smallest wavefront and those that are close to the smallest since tie-breaking strategies within the implementation of each algorithm can lead to small variations in the computed wavefronts. Looking first at the results for the direct ordering algorithms (that is, the numbers in columns 2, 4 and 6), we see that both the hybrid spectral–Sloan and the multilevel Sloan algorithms are generally an improvement on the Sloan algorithm. For many of the larger problems, including fcondp2 and shipsec1, the improvements are substantial. Comparing the two direct multilevel variants, the direct multilevel Sloan algorithm does not perform as well as the direct hybrid spectral–Sloan. Similar conclusions can be drawn when comparing the different indirect variants. Overall, the best method appears to be the indirect hybrid spectral–Sloan algorithm, with the advantage over the MC63 Sloan algorithm being greatest for the large test problems (those with more than about 10 000 elements). The indirect hybrid algorithm produces the best (or close to the best) results for the majority of our test problems.

Table IV. The time (in seconds) for reordering the elements using the Sloan and multilevel algorithms.

	Sloan (MC63)		Hybrid spectral–Sloan		Multilevel Sloan	
Identifier	Direct	Indirect	Direct	Indirect	Direct	Indirect
cham	0.01	0.01	0.15	0.20	0.15	0.15
fcondp2	0.09	0.09	0.31	0.33	0.29	0.23
halfb	0.20	0.13	0.79	0.44	0.73	0.35
mt1	0.02	0.05	0.06	0.22	0.05	0.15
ship_003	0.17	0.09	0.49	0.27	0.64	0.22
shipsec1	0.13	0.09	0.64	0.27	0.45	0.22
shipsec8	0.13	0.08	0.44	0.25	0.44	0.19
thread	0.01	0.04	0.03	0.23	0.03	0.13
x104	0.02	0.05	0.05	0.20	0.05	0.10

If only a single or small number of matrix factorizations and solves are to be performed following the reordering of the elements, the cost of reordering the elements may be a concern. Timings for a subset of our test problems are given in Table IV. The Sloan algorithm (MC63) is clearly significantly faster than the multilevel and hybrid variants, particularly for the problems with a large number of supervariables and elements. As already noted, Hu and Scott [10] introduced the multilevel Sloan profile reduction algorithm to save on the time required to compute a spectral ordering and we do achieve some savings in the element ordering times by using the multilevel Sloan algorithm rather than the spectral approach. For our examples, the indirect multilevel algorithm is between 25 and 50 per cent faster than the indirect hybrid algorithm.

4.3. Effect of Hager exchanges

The results reported so far did not use Hager exchanges. The results in Table V illustrate the reductions in the root-mean-squared wavefront that are achieved by using Hager exchanges within the call to HSL_MC73. In these experiments, up to a maximum of five down/up exchanges were allowed (the number of exchanges performed is fewer than five if the reductions in the profile are less than a prescribed amount; see Reid and Scott [13] for details). Results are given for both the indirect spectral–Sloan and multilevel Sloan algorithms. Problems for which the Hager exchanges do not reduce the root-mean-square wavefront are omitted while those for which the reduction exceeds 5 per cent are highlighted in bold. We see that there are only two problems for which the Hager exchanges applied to the hybrid spectral–Sloan ordering leads to a significant reduction in the wavefront. For the multilevel algorithm, Hager exchanges improve the ordering for a few more test examples and for some, including halfb and shipsec8, the multilevel plus Hager ordering is competitive with the hybrid spectral–Sloan ordering. Using Hager exchanges can add a large overhead to the cost of the element ordering. For example, for problem halfb, the multilevel reordering time increases from 0.35 to 0.58 s and for shipsec8 from 0.19 to 0.33 s. But these increases are small compared with the time required for the subsequent factorization (see Table VI below).

Table V. The root-mean-square wavefronts computed using the indirect spectral–Sloan and multilevel Sloan algorithms with and without Hager exchanges.

	Hybrid spectral–Sloan		Multilevel Sloan	
Identifier	Without	With	Without	With
cham	332	330	691	680
crplat2	244	232	257	239
fcondp2	1700	1621	1863	1805
fullb	1833	1804	3110	**2832**
halfb	1365	1362	1462	**1371**
ship_001	461	460	450	447
ship_003	1371	1322	1544	1544
shipsec1	1444	1418	1686	**1495**
shipsec5	1317	1276	1370	**1300**
shipsec8	1644	**1545**	1701	**1562**
srb1	326	326	327	324
troll	3593	**3349**	2377	**2249**
tubu	403	383	454	444
x104	1061	1052	1106	1104

*Numbers in bold indicate a reduction of at least 5 per cent.

Table VI. The results of using MC63 and the hybrid spectral–Sloan orderings with the frontal solver MA42.

	MA42 time (s)		Number flops ($\times 10^8$)		Factor entries ($\times 10^5$)	
Identifier	MC63	Hybrid	MC63	Hybrid	MC63	Hybrid
cham	**3.1**	**3.2**	29	29	85	85
crplat2	4.0	**2.3**	40	20	118	84
fcondp2	1657	**785**	27 393	11 560	9139	6100
fullb	1052	**882**	16 117	13 309	7735	7125
halfb	712	**575**	10 463	8259	6477	5927
inv-ext-2	962	**478**	16 387	7631	4877	3249
mt1	230	**145**	3462	2013	2397	1879
opt1	**7.2**	**7.3**	85	86	149	148
ship_001	**13**	**13**	140	146	312	318
ship_003	404	**309**	5663	4543	3557	3225
shipsec1	1075	**388**	17 138	5683	6316	3778
shipsec5	544	**431**	7964	6139	5183	4563
shipsec8	641	**413**	9011	6217	4187	3640
srb1	**12**	**12**	114	120	342	394
thread	145	**52**	2271	727	1086	627
trdheim	**1.1**	**1.1**	6	6	60	62
troll	**3468**	**3612**	54 610	55 210	13 428	13 433
tsyl201	**9.3**	**9.3**	104	108	205	210
tubu	**8.6**	**8.4**	91	89	208	208
x104	**149**	**145**	2054	2036	1882	1861

4.4. Element ordering with a frontal solver

As already discussed, the main motivation behind the work in this report is the need to compute element orderings that are efficient when used with a frontal solver. In this section, we present results for using the best element ordering method that we have, that is the indirect hybrid spectral–Sloan algorithm, with the well-known HSL frontal solver MA42 of Duff and Scott [14]. Comparisons are made with ordering the elements for MA42 using MC63 (both the direct and indirect Sloan algorithms are run and for each problem the better of the two is selected). Numerical values in the range $(0, 1)$ are generated for the entries of the matrices using the HSL pseudo-random number generator routine FA14. Default settings are used for the MA42 control parameters (with a minimum pivot block size of 16) and direct access files are used to store the matrix factors. In Table VI timings for factorizing and solving for a single right-hand side are given, together with flop counts (the number of floating point operations required to factorize the matrix) and the number of entries in the matrix factors. For each problem, the fastest time is highlighted in bold (no distinction is made between the two times if the difference between them is less than 5 per cent). We see that the reductions in the wavefronts reported on in Table III lead to sparser factors, smaller flop counts, and substantial savings in the time required by the frontal solver MA42. For problems fcondp2 and shipsec1 the time is reduced by more than half. It is clear that, in general, the faster factorization times more than compensate for the extra time required to reorder the elements using the hybrid algorithm (see Table IV).

5. CONCLUDING REMARKS

In this article, we have looked at using multilevel 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 and the Hager exchange algorithm. Numerical experimentation illustrated the benefits of using supervariables and showed that, in general, the best orderings are obtained using the indirect hybrid spectral–Sloan algorithm. We are currently developing a new Fortran 95 frontal solver for inclusion in the software library HSL 2004 [4]. Previous frontal solvers within HSL have required the user to preorder the elements but because using a good ordering is essential for the efficiency of the method, the new package will automatically reorder the elements for the user. Based on the results presented in this paper, the default setting will be to reorder the elements using the indirect hybrid spectral–Sloan algorithm, with the multilevel Fiedler code HSL_MC73 called internally to compute the necessary spectral ordering of the supervariable connectivity graph. Because the MC63 Sloan orderings are fast and generally produce orderings of a similar quality for relatively small problems, the new package will include an option to reorder using MC63. Hager exchanges generally did not result in significant reductions in the wavefront, so we do not plan to include their use within the new frontal code.

ACKNOWLEDGEMENTS

I would like to thank Ron Fowler of the Rutherford Appleton Laboratory and Christian Damhaug of Det Norske Veritas, Norway for supplying test problems and Yifan Hu for his work on developing HSL_MC73. This work was supported by the EPSRC Grant GR/S42170.

REFERENCES

1. Scott JA. On ordering elements for a frontal solver. *Communications in Numerical Methods in Engineering* 1999; **15**:309–323.
2. Duff IS, Reid JK, Scott JA. The use of profile reduction algorithms with a frontal code. *International Journal for Numerical Methods in Engineering* 1989; **28**:2555–2568.
3. Sloan SW. An algorithm for profile and wavefront reduction of sparse matrices. *International Journal for Numerical Methods in Engineering* 1986; **23**:1315–1324.
4. HSL. A collection of Fortran codes for large-scale scientific computation, 2004. See http://hsl.rl.ac.uk/
5. Fiedler M. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal* 1975; **25**:619–633.
6. Hu YF, Scott JA. HSL_MC73: A fast multilevel Fiedler and profile reduction code. *Technical Report RAL-TR-2003-036*, Rutherford Appleton Laboratory, 2003.
7. Barnard ST, Pothen A, Simon HD. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications* 1995; **2**:317–334.
8. Barnard ST, Simon HD. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience* 1994; **6**:101–117.
9. Kumfert G, Pothen A. Two improved algorithms for envelope and wavefront reduction. *BIT* 1997; **37**(3): 559–590.
10. Hu YF, Scott JA. A multilevel algorithm for wavefront reduction. *SIAM Journal on Scientific Computing* 2001; **23**:1352–1375.
11. Reid JK, Scott JA. Ordering symmetric sparse matrices for small profile and wavefront. *International Journal for Numerical Methods in Engineering* 1999; **45**:1737–1755.
12. Hager WW. Minimizing the profile of a matrix. *SIAM Journal on Scientific Computing* 2002; **23**(5): 1799–1816.
13. Reid JK, Scott JA. Implementing Hager's exchange methods for matrix profile reduction. *ACM Transactions on Mathematical Software* 2002; **28**:1–15.
14. Duff IS, Scott JA. The design of a new frontal code for solving sparse unsymmetric systems. *ACM Transactions on Mathematical Software* 1996; **22**(1):30–45.

