RAL-TR-2003-036

HSL_MC73 : A fast multilevel Fiedler and profile reduction code

Yifan Hu[1] and Jennifer A. Scott[2,3,4]

ABSTRACT

In recent years, multilevel algorithms have been used for the efficient computation of the eigenvector corresponding to the smallest positive eigenvalue of the Laplacian matrix associated with a graph of a symmetric matrix (the Fiedler vector). Multilevel algorithms have also been proposed for computing profile-reducing orderings for sparse symmetric matrices. In this paper, these multilevel algorithms are described within a unified framework. This is then used in the design of a new Fortran 95 code HSL_MC73 that implements a multilevel algorithm for the computation of an approximate Fiedler vector as well as a number of multilevel profile-reducing algorithms. HSL_MC73 is used to compute spectral orderings for a class of undirected random graphs and its performance is compared with obtaining the Fiedler vector using a state-of-the-art sparse eigensolver.

---

[1] Wolfram Research, Inc., 100 Trade Center Drive, Champaign, IL61820, USA.
Email: yifanhu@wolfram.com

[2] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, UK.
Email: j.a.scott@rl.ac.uk

[3] The work of this author was supported by the EPSRC grant GR/R46641.

[4] Current reports available from "http://www.numerical.rl.ac.uk/reports/reports.html"

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxfordshire OX11 0QX

December 19, 2003.

1 Introduction

During the last decade there has been considerable interest in spectral orderings for use in a number of different application areas, including graph partitioning for parallel computing (for example, Hendrickson and Leland, 1995), image analysis (Shi and Malik, 2000), profile and wavefront reduction algorithms for sparse symmetric matrices (Barnard, Pothen and Simon, 1995) and, most recently, the study of range-dependent random graphs (Higham, 2003*a*, 2003*b*). Spectral orderings 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 (Fiedler, 1975). Early implementations used a Lanczos algorithm to compute the Fiedler vector (Simon, 1991). This can be prohibitively expensive for large problems and led Barnard and Simon (1994) to propose a multilevel approach that they demonstrated to be capable of achieving an order-of-magnitude improvement in run time.

Even with a multilevel implementation, computing a spectral ordering for use as a profile reduction ordering is still significantly more expensive than using a heuristic algorithm such as Reverse Cuthill-McKee (Cuthill and McKee, 1969) or the Sloan method (Sloan, 1986). The hybrid Sloan algorithm of Kumfert and Pothen (1997) produces higher quality orderings than the Sloan method (particularly for very large problems), but it requires the spectral ordering to be computed. This motivated the development by Hu and Scott (2001) of an efficient multilevel algorithm for profile and wavefront reduction that in terms of quality is competitive with hybrid Sloan but is faster because it avoids the need for any spectral information.

The aim of this report is to describe the design and development of a new flexible software package that implements both a multilevel algorithm for computing the Fiedler vector and a number of multilevel profile reduction algorithms. The multilevel Fiedler vector and profile reduction algorithms are included in a single package because, as we will see, they employ similar techniques and the software needed to implement them contains common elements; it is thus efficient for software development and maintenance to incorporate both within one package. The new code, which is called `HSL_MC73`, is written in Fortran 95 and will be included in the next release of the mathematical software library HSL (HSL, 2002).

The report is organised as follows. In Sections 2 and 3, we briefly introduce Fiedler vectors and multilevel algorithms for the computation of approximate Fielder vectors and for obtaining profile reducing orderings. A unified framework for these algorithms is introduced in Section 4. We then describe the design of our new package `HSL_MC73` in Section 5 and in Section 6 we illustrate its use to reorder a class of range-dependent random graphs.

1.1 Notation

We end this section with some nomenclature and notation that is used throughout the remainder of the report. We use plain lowercase letters (such as λ) to refer to scalars. Bold lowercase letters $\mathbf{x}$ are vectors; x_j denotes the jth entry of the vector $\mathbf{x}$. $(A)_{ij}$ refers to the (i,j)th entry of the matrix A. $\mathcal{G}$ denotes a graph.

Let A be an $n \times n$ matrix with a symmetric sparsity pattern. The *profile* of A is the total number of entries in the lower triangle when any zero ahead of the first entry in its row is excluded, that is,

$$P(A) = \sum_{i=1}^{n} \max_{(A)_{ij} \neq 0} \{i + 1 - j\}.$$

An undirected graph $\mathcal{G}$ is defined to be a pair (V, E) where V is a set of n vertices $\{v_1, v_2, ..., v_n\}$ and E is a set of edges defined as unordered pairs (v_i, v_j) of distinct vertices. A *weighted graph* is one for which a (positive) value w_i is associated with each vertex v_i and a (positive) weight w_{ij} is associated with each edge (v_i, v_j). An *ordering* (or *labelling*) of a graph $\mathcal{G}$ with n vertices is a bijection of $\{1, 2, ..., n\}$ onto V.

A *path of length* k in $\mathcal{G}$ is an ordered set of distinct vertices $(v_{i_1}, v_{i_2}, ..., v_{i_{k+1}})$ where $(v_{i_j}, v_{i_{j+1}}) \in E$ for $1 \leq j \leq k$. Two vertices are *connected* if there exists a path between them. The *distance* between two vertices u and v is the length of the shortest path connecting them and is denoted by $dist(u, v)$. The *diameter length* $\delta(\mathcal{G})$ is the greatest distance between any two vertices in $\mathcal{G}$. A *diameter* is a shortest path between two vertices u and v in $\mathcal{G}$ whose distance apart is equal to $\delta(\mathcal{G})$. A *pseudodiameter* is either a diameter or a shortest path between two vertices in $\mathcal{G}$ whose distance apart is slightly more than $\delta(\mathcal{G})$.

The adjacency graph $\mathcal{G}(A)$ of the symmetric matrix A has vertices $V(A) = \{1, 2, \ldots, n\}$ and edges $E(A) = \{(i, j) \mid (A)_{ij} \neq 0,\ i > j\}$. Symmetric permutations of A correspond to relabelling the vertices of the $\mathcal{G}(A)$.

Throughout our discussion, it is assumed that the matrix A of interest is irreducible so that its adjacency graph $\mathcal{G}(A)$ is connected (that is, each pair of distinct vertices is connected). Disconnected graphs can be treated by considering each component separately.

2 Introduction to Fiedler vectors and a multilevel approach

We start by defining the Laplacian matrix. Let $\mathcal{G} = (V, E)$ be an undirected weighted graph with (positive) weights $\{w_{ij}\}$. The weighted *Laplacian* $L(\mathcal{G})$ is defined to be the $n \times n$ symmetric matrix

$$L(\mathcal{G}) = D - W, \tag{2.1}$$

where W has entries

$$(W)_{ij} = \begin{cases} w_{ij} & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise,} \end{cases}$$

and D is the diagonal matrix with entries $(D)_{ii} = \sum_i W_{ij}$. The unweighted Laplacian (often simply referred to as the *Laplacian*) is obtained by setting the weights equal to one.

The Laplacian has a number of important properties; details may be found, for example, in Mohar (1991). First note that the bilinear form associated with the Laplacian can be written as

$$\mathbf{p}^T L(\mathcal{G})\mathbf{p} = \sum_{i<j} w_{ij}(p_i - p_j)^2. \tag{2.2}$$

It follows that $L(\mathcal{G})$ is diagonally dominant and positive-semidefinite. Furthermore, it is easy to see that the smallest eigenvalue is zero, with associated eigenvector $\mathbf{e} = [1, 1, \ldots, 1]^T$. Provided $\mathcal{G}$ is connected, the smallest eigenvalue is simple and the second smallest eigenvalue λ_2 is positive.

The special properties of the eigenvector $\mathbf{x}$ corresponding to λ_2 and its relationship to the connectivity of a graph were first studied by Fiedler (1975). His work provides much of the theoretical justification of the use of the second eigenvector of $L(\mathcal{G})$ for graph partitioning and hence this vector is usually called the *Fiedler* vector. The Fiedler vector is a real-valued vector but by ordering its components into monotonic increasing (or decreasing) order a permutation vector $\mathbf{p}$ can be induced. Thus $\mathbf{p}$ is chosen such that $p_i \leq p_j$ if and only if $x_i \leq x_j$. A permutation induced by the Fiedler vector is what is meant by a *spectral ordering*. A key property of this ordering is that if a component x_j in the ordered Fiedler vector is distinct from its neighbours ($x_{j-1} < x_j < x_{j+1}$), vertex j must be connected both to its set of predecessors and its set of successors. Similarly, a set of vertices whose components have identical values distinct from its neighbours ($x_{j-1} < x_j = x_{j+1} = \ldots = x_{j+s-1} < x_{j+s}$) must be connected to its set of predecessors and its set of successors.

The graph partitioning problem is to find a set of disjoint subsets V_i of the vertex set V with $V = \bigcup_i V_i$ such that the number of edges $(v_i, v_j) \in E$ with v_i and v_j belonging to different subsets is small (that is, the *edge cut* is small) while also satisfying certain load balance conditions. One approach to this problem is the recursive spectral bisection (RSB) method of Pothen, Simon and Liou (1990). RSB is based upon computing the Fiedler vector $\mathbf{x}$. For the initial partition, the median value μ of $\mathbf{x}$ is computed and all vertices j with $x_j < \mu$ are assigned to subdomain 1, all vertices with $x_j > \mu$ to subdomain 2, and an equal number of all the vertices j with $x_j = \mu$ to subdomains 1 and 2. If partitioning to more than two subdomains is required, the algorithm is applied recursively to each subdomain until the desired number of partitions is achieved.

The main problem with implementing the RSB method is that computing eigenvectors of large matrices is expensive. This led Barnard and Simon (1994) to propose a multilevel algorithm for computing the Fiedler vector. The basic multilevel algorithm proceeds as follows:

- A series of graphs of successively coarser (smaller) sizes is generated (we discuss this further in Section 4.1).
- At some point the graph has so few vertices that it is very cheap to compute the Fiedler vector of the associated Laplacian matrix.
- 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 matrix is obtained. The refinement process is discussed in Section 4.3.

This algorithm is implemented within the new software package `HSL_MC73`. A key observation is that, for graph partitioning, 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, as we shall see in our discussion

of the software design (Section 5.1), a number of parameters under the user's control are used in determining how accurate the requested eigenvector is.

3 Multilevel profile reduction algorithms

In this section we briefly describe the three profile reduction algorithms offered by `HSL_MC73`, namely the spectral ordering algorithm, the hybrid algorithm and the multilevel Sloan algorithm.

3.1 Spectral profile ordering algorithm

Following the success of spectral orderings for graph partitioning, Barnard et al. (1995) proposed using the Fiedler vector to obtain profile reducing orderings for matrices A with symmetric sparsity patterns. The algorithm is motivated as an attempt to minimize the two-sum

$$\min_{\mathbf{x}\in\mathcal{P}}\{\sum_{\{i<j:\ (A)_{ij}\neq 0\}}(x_i-x_j)^2\}. \tag{3.1}$$

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

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

That is,

$$\min_{\mathbf{x}\in\mathcal{P}}\mathbf{x}^T L(\mathcal{G})\mathbf{x} \tag{3.2}$$

where $L(\mathcal{G})$ is the unweighted Laplacian. 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 $\mathcal{P}$, the problem (3.2) is relaxed to $\mathbf{x}\in\mathcal{R}^n$ with $\mathbf{x}^T\mathbf{e}=0$ and $\|\mathbf{x}\|_2=\|\mathbf{p}\|_2$ for any $\mathbf{p}\in\mathcal{P}$. The solution is then the eigenvector corresponding to the second smallest eigenvalue of $L(\mathcal{G})$, that is, the Fiedler vector. Applying the permutation induced by ordering the components of the Fiedler vector to the matrix A gives the so-called spectral ordering algorithm. In general, it not only reduces the two-sum but also the profile and wavefront of the matrix. Again, the Fiedler vector does not generally need to be computed to great accuracy to obtain a good ordering.

Barnard et al. (1995) work only with the unweighted Laplacian associated with the adjacency graph $\mathcal{G}(A)$ of A. However, in his recent work on undirected range-dependent random graphs, Higham (2003*a*) computes the Fiedler vector of a weighted Laplacian and uses it to obtain a spectral ordering (we discuss this further in Section 6). To be as general-purpose as possible, `HSL_MC73` offers a multilevel spectral ordering algorithm that optionally uses a weighted Laplacian.

3.2 Sloan algorithm

Before describing the hybrid algorithm, it is helpful to recall the Sloan algorithm for profile and wavefront reduction (Sloan, 1986). The method uses the adjacency graph $\mathcal{G}(A)$ of A and has two distinct phases:

1. Selection of a start vertex s and a target end vertex e.

2. Vertex reordering.

The first phase looks for a pseudodiameter of $\mathcal{G}(A)$ and uses it to provide s and e. A pseudodiameter may be computed using a modification of the Gibbs-Poole-Stockmeyer algorithm (see Reid and Scott (1999) for details of an efficient approach). In the second phase, the chosen start vertex is numbered first and a list of vertices that are eligible to be numbered next is formed. At each stage of the numbering, the list of eligible vertices comprises the neighbours of the vertices that have already been renumbered and their neighbours. The next vertex to be numbered is selected from the list of eligible vertices to maximise the priority function

$$P(i) = -W_1 \text{ inc}(i) + W_2 \text{ dist}(i, e), \tag{3.3}$$

where (W_1, W_2) are positive weights. The first term, $\text{inc}(i)$, is the amount by which the wavefront will increase if vertex i is ordered next. The second term, $\text{dist}(i, e)$, is the distance between vertices i and the end vertex e. Thus, a balance is maintained between the aim of keeping the wavefront small and bringing in vertices that have been left behind (that is, those far away from the target end vertex e). A vertex 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 vertex e. Experiments have shown that the best choice for the weights (W_1, W_2) is problem dependent, but either $(2, 1)$ or $(16, 1)$ generally gives good results (see Reid and Scott, 1999).

Sloan's algorithm has been enhanced by a number of authors, notably Duff, Reid and Scott (1989), Kumfert and Pothen (1997), and Reid and Scott (1999). In the HSL mathematical software library, an efficient Fortran implementation of Sloan's algorithm is included as routine `MC60`, with a straightforward user interface to this package provided by routine `MC61`.

3.3 Hybrid algorithm

The hybrid algorithm is a generalisation of the Sloan algorithm proposed by Kumfert and Pothen (1997). They observed that there are problems on 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 (3.3) affects the priority function in a local way, by giving higher priority to vertices 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 vertices lying far away from the end vertex 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 $\text{dist}(i, e)$. Thus the idea of Kumfert and Pothen was to modify the second phase of Sloan's algorithm so that, in place of the distance function, it refined the spectral ordering.

The hybrid algorithm chooses as the start vertex s the first vertex in the spectral ordering and replaces (3.3) with the priority function

$$P(i) = -W_1 \text{ inc}(i) - W_2 \ \nu \ p_i. \tag{3.4}$$

Here ν is a normalising factor and p_i is the position of vertex i in the spectral ordering, also referred to as its *global priority value*. ν is chosen so that the factor for W_2 varies up to $\mathrm{dist}(s, e)$, as in (3.3). Numerical experiments reported by Kumfert and Pothen (1997) and Reid and Scott (1999) have shown that, for large problems, in terms of the quality of the orderings produced, the hybrid method with an appropriate choice of weights can significantly outperform both the spectral and the Sloan algorithms. Our new package `HSL_MC73` may be used to compute the multilevel spectral ordering and then it optionally employs `MC60` to compute a hybrid ordering.

3.4 Multilevel Sloan algorithm

The main disadvantage of the hybrid 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 pseudodiameter for A using the (modified) Gibbs-Poole-Stockmeyer algorithm of Reid and Scott (1999). Even if the Fiedler vector is computed using a multilevel algorithm, the hybrid algorithm can be too expensive. In an attempt to avoid computation of the Fiedler vector while still maintaining the quality of the hybrid algorithm, Hu and Scott (2001) proposed a multilevel version of Sloan's algorithm. Mirroring the multilevel Fiedler algorithm introduced in Section 2, the multilevel Sloan 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 (this is discussed in Section 4.3).

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.

4 Unified framework for the multilevel approach

We now summarise both the multilevel Fiedler and profile reduction algorithms outlined in Sections 2 and 3 within a single framework. This forms the basis of the design of our software package `HSL_MC73`. It is convenient to introduce some further notation. The subscripts f and c are used to represent fine and coarse graph quantities, respectively. For example, $\mathcal{G}_f$ denotes the fine graph with n_f vertices and $\mathcal{G}_c$ is the graph with n_c vertices obtained after coarsening ($n_c < n_f$). We will associate with $\mathcal{G}_f$ an $n_f \times n_f$ adjacency matrix G_f which has a nonzero entry in position (i, j) if and only if vertices i and j are adjacent in $\mathcal{G}_f$. G_c is defined analogously.

When moving from a coarse graph to a fine graph, the Fiedler vector or the ordering for the coarse graph must be mapped from the coarse graph onto the fine graph. This mapping is represented by a prolongation (or interpolation) matrix. If P denotes an $n_f \times n_c$ prolongation matrix, the coarse graph may be expressed as the Galerkin product

$$G_c \leftarrow P^T G_f P.$$

We denote by $\mathbf{p}$ either the Fiedler vector or the profile reducing ordering that is computed for the graph $\mathcal{G}$. We let $CoarsestAlg(\mathcal{G})$ be an algorithm that returns the Fiedler vector or ordering for the coarsest graph $\mathcal{G}$ and let $Refine(\mathcal{G}, \mathbf{p}^0)$ denote the algorithm that takes the graph $\mathcal{G}$, and its Fiedler vector or ordering $\mathbf{p}^0$ and returns a refined Fiedler vector or ordering $\mathbf{p}$ for $\mathcal{G}$.

With this notation, the multilevel Fiedler vector and profile reduction algorithms can be formally presented as follows. The starting point is the fine graph $\mathcal{G}_f$ and associated adjacency matrix G_f.

FUNCTION MultilevelAlg ($\mathcal{G}_f$)

- *If no further coarsening is needed*
  - $\mathbf{p}_f = CoarsestAlg(\mathcal{G}_f)$
  - *return* $\mathbf{p}_f$
- *The coarsening phase:*
  - *set up the* $n_f \times n_c$ *prolongation matrix* P
  - $G_c \leftarrow P^T G_f P$
  - $\mathbf{p}_c = MultilevelAlg\ (\mathcal{G}_c)$
- *The prolongation and refinement phase:*
  - $\mathbf{p}_f^0 = P\ \mathbf{p_c}$
  - $\mathbf{p}_f = Refine(\mathcal{G}_f, \mathbf{p}_f^0)$
  - *return* p_f

In the remainder of this section, we briefly discuss each of the three key steps (coarsening, the coarse graph algorithm, and prolongation and refinement).

4.1 The coarsening phase

There are a number of ways to coarsen an undirected graph. For graph partitioning the aim is to ensure a small edge-cut. The most popular method is based on edge collapsing (Hendrickson and Leland, 1993, Karypis and Kumar, 1999), in which pairs of adjacent vertices are selected and each pair is coalesced into one new vertex. Each edge of the fine graph has weight one. Each edge of the coarse graph has a weight associated with it which is related to the number of edges in the fine graph that it replaces; heavy-edge

matching preferentially collapses heavier edges. HSL_MC73 uses heavy-edge collapsing in its implementation of the multilevel Fiedler vector and spectral ordering algorithms.

Hu and Scott (2000) report that for their multilevel Sloan algorithm, heavy-edge collapsing, while improving on the profiles obtained using the Sloan algorithm, generally produces profiles that are of a poorer quality than those obtained using the hybrid algorithm. Instead, following Ruge and Stüben (1987) and Barnard and Simon (1994), Hu and Scott use a maximal independent vertex set strategy. An independent set of vertices is a subset of the vertices such that no two vertices in the subset are connected by an edge in the graph. An independent set is maximal if the addition of an extra vertex always destroys the independence. Each vertex of the fine graph has weight one and each vertex of the coarse graph has a weight related to the number of neighbouring fine graph vertices that belong to the maximal independent set. The algorithm for constructing a maximal independent set used by HSL_MC73 is discussed by Hu and Scott (2001).

Having chosen an appropriate coarsening process, it is applied recursively until one of the following is achieved:

- the number of vertices n_c in the coarsest graph is less than a preset number
- the number of levels exceeds a preset limit
- the ratio of the number of vertices in two successive graphs exceeds a preset constant.

The last condition is necessary, particularly if edge collapsing is used, because it is possible that after a number of levels of coarsening the coarsest graph has one supervertex with a very high vertex weight, possibly exceeding 50% of the total. In this case, subsequent coarsening will not reduce the size of the graph significantly. Furthermore, a multilevel algorithm with only a small reduction between fine and coarse graph sizes will have a high algorithmic complexity.

4.2 The coarse graph algorithm

If the aim of the multilevel algorithm is to compute an approximate Fiedler vector, then on the coarsest graph $\mathcal{G}_c$ the Fiedler vector $\mathbf{x} = \mathbf{p}_c$ of the associated Laplacian L_c must be computed. As the number of vertices n_c in the coarsest graph is small compared with the order n of the original matrix A, this can be done cheaply. Clearly, there are several possible approaches; in HSL_MC73 a standard Lanczos algorithm is used.

For the multilevel Sloan algorithm, a profile reducing ordering $\mathbf{p}_c$ must be computed on $\mathcal{G}_c$. HSL_MC73 uses the enhanced Sloan algorithm of Reid and Scott (1999) to compute $\mathbf{p}_c$.

4.3 Prolongation and refinement

The prolongation step depends on the coarsening algorithm used. When computing the Fiedler vector, HSL_MC73 performs the coarsening using heavy-edge collapsing and the position of a vertex j in the coarse graph is injected to give a value to its parent (or parents). A parent of j is defined to be a vertex on the fine graph that either coalesces into j, or remains as j itself. The prolongation matrix P thus has entries $(P)_{ij}$ given by

$$(P)_{ij} = \begin{cases} 1, & \text{if fine graph vertex } i \text{ is a parent of coarse graph vertex } j, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Having prolonged the coarse grid Fiedler vector onto the fine grid ($\mathbf{p}_f^0 = P\ \mathbf{p_c}$), the Rayleigh Quotient Iteration algorithm (see, for example, Parlett, 1980) is used to give a more accurate result.

ALGORITHM RQI $(\mathbf{x}, L, \epsilon)$

$\theta \leftarrow \mathbf{x}^T L\mathbf{x}$

do

Solve $(L - \theta I)\mathbf{z} = \mathbf{x}$

$\mathbf{x} \leftarrow \mathbf{z}/||\mathbf{z}||$

$\theta \leftarrow \mathbf{x}^T L\mathbf{x}$

if $\|L\mathbf{x} - \theta\mathbf{x}\| < \epsilon$ return $\mathbf{x}$

end do

An approximate eigenvector $\mathbf{x} = \mathbf{p}_f^0$ of L_f is input and, after refinement, a more accurate eigenvector $\mathbf{p}_f$ is output. Here ϵ is the convergence tolerance. A limit is normally imposed on the number of iterations performed at each refinement step. A sparse indefinite linear system of the form $(L_f - \theta I)\mathbf{z} = \mathbf{x}$ must be solved at each iteration. The SYMMLQ algorithm (Paige and Saunders, 1974), which is an extension of the conjugate gradient method, is used by `HSL_MC73`.

When coarsening is based on a maximal independent vertex set, the vertices of $\mathcal{G}_c$ comprise the maximal independent set of $\mathcal{G}_f$. For each coarse graph vertex j, let $fine(j)$ denote the corresponding fine graph vertex. For each fine graph vertex i, define $mdeg(i)$ to be the number of neighbouring fine graph vertices that belong to the maximal independent set. The prolongation matrix has entries

$$(P)_{ij} \ = \ \begin{cases} 1, & \text{if } i = fine(j), \\ \frac{1}{mdeg(i)}, & \text{if } i \leftrightarrow fine(j),\ i \neq fine(j), \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where $i \leftrightarrow j$ if i and j are neighbours.

Prolongation of the coarse grid ordering $\mathbf{p}_c$ onto the fine grid gives an ordering $\mathbf{p}_f^0$ that must be refined. The multilevel Sloan algorithm does this using the second phase of Sloan's algorithm with the priority function

$$P(i) = -W_1\ \mathrm{inc}(i) - W_2\ \nu\ \{\mathbf{p}_f^0\}_i. \tag{4.3}$$

Here ν is again a normalising factor, (W_1, W_2) are positive weights, and the global priority value $\{\mathbf{p}_f^0\}_i$ is the position of vertex i in the prolonged coarse grid ordering. The vertex which is first in the ordering $\mathbf{p}_f^0$ is chosen as the start vertex; a list of eligible vertices is then constructed and the ordering of the fine grid proceeds as in the Sloan algorithm with (3.3) replaced by (4.3).

5 Software design

In this section, we discuss the design of our new software package HSL_MC73 for computing either the (approximate) Fiedler vector of the unweighted or weighted Laplacian associated with A or a symmetric permutation that reduces the profile and wavefront of A by using a multilevel algorithm. The new subroutines are named according to the naming convention of the mathematical software library HSL. The package itself is now available; details of obtaining a licence to use the code can be found at www.cse.clrc.ac.uk/nag/index.shtml.

The interface is designed to be straightforward to use but at the same time offers flexibility through the choice of algorithms available and the freedom to set parameters that control the performance of the algorithms. The following procedures are available to the user:

(a) The initialization subroutine MC73_INITIALISE must first be called. This call gives default values to the parameters that control the execution of the package (see Section 5.1).

(b) If the user wishes to compute the (approximate) Fiedler vector of the (weighted) Laplacian, MC73_FIEDLER should be called.

(c) To compute a symmetric permutation that aims to reduce the profile and wavefront, MC73_ORDER should be called.

(d) MC73_PRINT_MESSAGE can be used after a return from MC73_FIEDLER or MC73_ORDER to print the error or warning message associated with a nonzero error flag.

We describe these now in greater detail.

5.1 The control parameters

The derived data type MC73_CONTROL is used to control the action. The user must declare a structure of type MC73_CONTROL. Components of this derived type are initialised to their default values by a call to MC73_INITIALIZE, and the user does not need to reset them unless values other than the defaults are required. The components of MC73_CONTROL include stream numbers to which error and warning messages, as well as diagnostic printing, are sent. A further component controls the level of diagnostic printing the user requires. The remaining components, which we now discuss, are likely to be of most interest to the more experienced user. The default values were all selected following extensive numerical experiments.

COARSEST_SIZE determines the problem size in the multilevel hierarchy below which no further coarsening is performed. The default value is $n_c = 200$.

MGLEVEL holds the maximum number of levels in the multilevel hierarchy and has a default value of 100. If the default is used then, in general, coarsening will terminate long before MGLEVEL is reached (because further coarsening would reduce the number of vertices to less than COARSEST_SIZE). If the user sets MGLEVEL to one, the multilevel Fiedler code MC73_FIEDLER reduces to the Lanczos algorithm and the multilevel

Sloan algorithm is replaced by the Sloan algorithm. In the latter case, MC73_ORDER calls the HSL code MC61 to compute a profile-reducing ordering.

MAX_REDUCTION and MIN_REDUCTION are maximum and minimum grid reduction factors (MAX_REDUCTION > MIN_REDUCTION). They have default values of 0.8 and 0.1, respectively. If two successive grids have n_c and n_f vertices, respectively, coarsening continues while $n_c < n_f$*MAX_REDUCTION and $n_c > n_f$*MIN_REDUCTION. MAX_REDUCTION must lie in the interval $(0.5, 1.0)$; values greater than 1 are treated as 1 and values less than 0.5 are treated as 0.5. This restriction avoids a slow reduction between fine and coarse graph sizes.

MLANCZ is the maximum number of Lanczos vectors used in the computation of the Fiedler vector on the coarsest grid. The default value is 300.

TOL, TOL1, and RTOL hold convergence tolerance parameters used in the computation of the Fiedler vector and the multilevel spectral ordering. On the coarsest graph $\mathcal{G}_c$ the norm of the residual $L_c\mathbf{p}_c - \lambda_2\mathbf{p}_c$ must be less than TOL. Here λ_2 is the smallest positive eigenvalue and $\mathbf{p}_c$ is the approximate Fiedler vector (with $\|\mathbf{p}_c\|_2 = 1$). TOL1 holds the convergence tolerance for the RQI algorithm. TOL and TOL1 have default value 10^{-3}. Smaller values may slow convergence while providing greater accuracy. If the computation of the Fiedler vector is requested, the convergence tolerance used by the SYMMLQ algorithm within the RQI algorithm is min(10^{-3}*RTOL, 10*TOL). If the Fiedler vector is used within MC73_ORDER to obtain a spectral ordering, less accuracy is generally required and in this case the tolerance used by SYMMLQ is min(RTOL, 10*TOL). The default value for RTOL is 10^{-2}.

MAXIT controls the maximum number of Rayleigh Quotient iterations at each refinement step. The default value is 10.

HAGER_EXCHANGE controls the use of the Hager exchange algorithms that aim to refine the profile reducing ordering (see below for further details). By default, no exchanges are performed but if HAGER_EXCHANGE > 0, down/up exchanges are applied a maximum of HAGER_EXCHANGE times.

5.2 Computation of an approximate Fiedler vector

MC73_FIEDLER accepts the pattern of the lower-triangular part of the matrix A. This is checked for errors and the sparsity pattern of the whole of A (with the diagonal removed) is constructed using the HSL package of sparse matrix manipulation routines HSL_MC65. Whether or not the user requires the weighted or unweighted Laplacian is controlled by the optional argument WGT. If this array is present, it must be set by the user to hold the weights corresponding to the nonzero entries of A. In this case, the weighted Laplacian is constructed; otherwise the unweighted Laplacian is used.

The approximate Fiedler vector is then computed using the multilevel algorithm and is returned to the user without sorting. If A is reducible or equivalently, if the adjacency graph $\mathcal{G}(A)$ of A has more than one component, the approximate Fiedler vector for each component is computed and the resulting Fiedler vectors are merged into the array

FVECTOR. The integer array LIST is used to index the component to which each variable belongs. If a component consists of a single vertex, it is assigned the value 0 in LIST and in the returned Fiedler vector.

5.3 Computation of a symmetric permutation

The algorithm implemented by MC73_ORDER is determined by the parameter JOB, which must be set by the user. The possible values are:

1: multilevel Sloan algorithm

2: multilevel spectral ordering algorithm

3: hybrid ordering algorithm.

Again, the user must supply the pattern of the lower-triangular part of A and, for JOB = 2 or 3, the use of the weighted Laplacian is controlled by the presence of the optional argument WGT. The computed ordering starts with the components that consist of single vertices and is followed in turn by the orderings of each of the non-trivial components.

If JOB = 1, the adjacency graph $\mathcal{G}(A)$ is constructed and the multilevel Sloan algorithm applied to each component of this graph. At the coarsest level, the HSL package MC60 that implements the (enhanced) Sloan algorithm is used with the two pairs of weights (2,1) and (16,1) and the ordering with the smallest profile is chosen as the coarse graph ordering. At each level, MC60 is again used to perform the refinement, but in this case the pairs (1,2) and (16,1) are used (this choice is based on the reported experience of Hu and Scott, 2001).

If JOB = 2, the approximate Fiedler vector of the Laplacian of each component is computed. The entries of this vector are sorted in non-decreasing order to give the required multilevel spectral ordering.

If JOB = 3, the algorithm proceeds as in the case JOB = 2. Once the spectral ordering has been computed, MC60 is used (on the finest graph) to compute the hybrid ordering. Two sets of weights (1,2) and (16,1) are used and the ordering with the smallest profile is selected (see Reid and Scott, 1999). If different weights are wanted, the user may call MC60 with his or her own choice of weights after the spectral ordering has been computed.

For each value of JOB, the computed ordering may optionally be refined by the HSL routine MC67, which implements Hager's exchange algorithms. Hager (2000) suggested two methods for improving any given permutation for profile reduction. These methods work only with the sparsity pattern of the matrix. His down exchange algorithm involves a cyclic permutation, that is, the successive exchange of rows $(k, k+1)$, $(k+1, k+2)$, $\ldots$, $(l-1, l)$ of the permuted matrix and interchanging corresponding columns. For a given k, Hager finds the value of l that most reduces the profile. He performs a pass over the matrix with k taking the values $n-1$, $n-2$, $\ldots$, 1; he calculates l for each k and, if this gives a profile reduction, applies the corresponding permutation. Hager's up exchange is similar, with the direction reversed. For a given k, he exchanges rows and columns $(k, k-1), (k-1, k-2), \ldots, (l+1, l)$, finding the value of l that most reduces the profile. He performs a pass over the matrix with k taking the values 2, 3, $\ldots$, n. Hager proposed using the down exchange and up exchange schemes in an iterative fashion: the down exchange

algorithm is first applied, followed by the up exchange algorithm, followed by the down exchange algorithm, and so on. If the unweighted graph of A is used, `HSL_MC73` will perform down/up exchanges provided the user resets the control parameter `HAGER_EXCHANGE` to a nonzero value; if other combinations of down and up exchanges are required, the user may call `MC67` directly after the call to `MC73_ORDER`. Note that if the weighted graph is used, `HAGER_EXCHANGE` is not accessed and exchanges are not performed.

6 An application: spectral reordering of range-dependent random graphs

In this section, we report on using `HSL_MC73` to obtain spectral orderings for range-dependent random graphs. The numerical experiments were performed on a single Xeon 3.06 GHz processor of a Dell Precision Workstation 650 with 4 GBytes of RAM under the Red Hat 9 Linux operating system. The NAG Fortran 95 compiler was used with the compiler optimization flag `-O`. Unless otherwise stated, the default values are used for all the `HSL_MC73` control parameters discussed in Section 5.1.

Higham (2003*a*) defines an undirected range-dependent random graph as follows.

Definition 1 *Given a set of edge density functions* $\{f^{[k]}(x)\}_{k=1}^{n}$, *with* $f^{[k]}(x) = 0$ *for* $x < 0$, *the corresponding undirected range-dependent random graph (URDRG) has vertices ordered* $1, 2, \ldots, n$ *with independent edge weights* w_{ij} *such that* w_{ij} *has density* $f^{[j-i]}(x)$.

Higham considers the question "If the vertices of a URDRG are ordered arbitrarily, can an ordering be found that reveals the underlying short-range connectivity patterns?" Higham shows that a possible approach to this problem is to obtain a reordering for the URDRG using the spectral algorithm applied to the weighted graph. To illustrate the approach, edge weights for the original URDRG are generated from a pseudo-random number generator using an exponential distribution with parameter $(i-j)^2$. An arbitrary symmetric row and column permutation is applied to the URDRG; this results in a "shuffled" graph, which is then reordered using the spectral algorithm. The interest is in how close the final graph is to the original URDRG.

To measure how well the unshuffling works, we need some notation. Let $\mathbf{q}$ denote the initial permutation such that $\mathbf{q}_j$ is the position in the original graph of vertex j in the shuffled graph. Similarly, let $\mathbf{p}$ denote the spectral ordering such that $\mathbf{p}_i$ is the position in the final graph of vertex i in the shuffled graph. Thus $(\mathbf{q}(\mathbf{p}))_i$ is the position in the original graph of vertex i in the final graph. If the unshuffling is exact, plotting $\mathbf{q}(\mathbf{p})$ would give points on a straight line with slope ± 1.

We now define the maximum and mean errors in the unshuffling to be *perr* and *pavg* where

$$perr = \min\left(\max_i |(\mathbf{q}(\mathbf{p}))_i - i|, \max_i |(\mathbf{q}(\mathbf{p}))_i - (n-i+1)|\right)$$

and

$$pavg = \frac{1}{n}\min\left(\sum_i |(\mathbf{q}(\mathbf{p}))_i - i|, \sum_i |(\mathbf{q}(\mathbf{p}))_i - (n-i+1)|\right)$$

To define the two-sum error, let the weights of the original and final graphs be w_{ij} and $\hat{w}_{ij}$, respectively. Then

$$2sumerr = \left| \frac{\sum_{ij}(i-j)^2 \ \hat{w}_{ij} - \sum_{ij}(i-j)^2 \ w_{ij}}{\sum_{ij}(i-j)^2 w_{ij}} \right| \tag{6.1}$$

Thus $2sumerr$ records how close the two-sum of the final unshuffled matrix is to that of the original matrix. Clearly, the three statistics should be small if the unshuffling (that is, the spectral reordering of the permuted graph) has been successful.

If we generate the weights for the URDRG using $f^{[k]}(x) = k^2 exp(-k^2 x)$, the associated matrix A will be dense but with a large number of small entries. To obtain a sparse matrix $\tilde{A}$ we "drop" small entries, that is, we set them to zero. Once the resulting sparse matrix has been shuffled, `HSL_MC73` is employed to compute the spectral ordering of the resulting weighted Laplacian. Note that the errors $perr$ and $pavg$ are for the spectral ordering applied to the sparse matrix $\tilde{A}$ but $2sumerr$ measures the success of the unshuffling relative to the original matrix A.

Higham (2003*a*) derives $\tilde{A}$ by retaining only the largest 20 per cent of the entries of A. We do not follow this method here for a number of reasons. Firstly, the choice of 20 per cent appears to be somewhat arbitrary and the effects of varying this choice on both the quality of the ordering and on the computational time are not considered. Moreover, to retain a fixed percentage of the entries requires n^2 entries to be stored and then sorted. For very large n this will be prohibitively expensive in terms of the memory required as well as the time needed for sorting. Instead, we set all entries that are less than a given value ϵ to zero. We have performed experiments with a range of values of n and have examined how sensitive the results are to the choice of the dropping parameter ϵ. In Table 6.1, results are presented for $n = 200, 2000$ and 20000 using different choices for ϵ. In each case, 100 examples are generated; we report the maximum and mean values of $2sumerr$ and the mean CPU time (in seconds) to compute the spectral ordering. We also report the density of $\tilde{A}$, that is $100 * nz/n^2$, where nz is the number of entries in $\tilde{A}$. In our tests we consider $\epsilon \leq 0.01$; we found that larger values can lead to so many entries being dropped that $\tilde{A}$ may have columns with no nonzero entries. We see that, provided ϵ is sufficiently small, the $2sumerr$ is not very sensitive to the precise choice of ϵ, but as ϵ is reduced, the density of $\tilde{A}$ increases and this leads to an increase in the CPU time. Based on our findings, in the remainder of our experiments we set $\epsilon = 0.001$.

Results obtained using `HSL_MC73` for a range of values of n are given in Table 6.2. Again, in each case, 100 examples are generated. Because we are anxious for `MC73_ORDER` to converge to the correct eigenvector, we set the control parameter `RTOL` used by the SYMMLQ algorithm to 10^{-5} (see Section 5.1). Even with a small value for `RTOL`, we cannot guarantee convergence to the required vector. In many applications, the spectral ordering obtained from a misconverged vector (that is, from an eigenvector that is not the Fiedler vector) will be adequate (for example, it will usually provide a good profile-reducing ordering, see Kumfert and Pothen, 1997) but experiments show that it will not give a good reordering of a shuffled matrix. The entry in the final column is the number of times `HSL_MC73` misconverged; these misses are not included in the computation of the maximum and mean values reported in other columns. We see that, in general, `HSL_MC73` is able to

n	ϵ	Density of $\tilde{A}$	$2sumerr$ max.	$2sumerr$ mean	Time
200	1e-02	8	3.47e-03	4.06e-04	0.02
	1e-03	25	8.85e-04	2.83e-04	0.03
	1e-04	63	9.18e-04	2.72e-04	0.06
	1e-05	93	9.18e-04	2.62e-04	0.09
2000	1e-02	0.8	3.57e-04	1.95e-05	0.13
	1e-03	3	7.00e-05	1.08e-05	0.26
	1e-04	9	8.26e-05	1.31e-05	0.74
	1e-05	25	5.98e-04	4.08e-05	2.24
20000	1e-02	0.08	3.93e-05	5.01e-06	1.31
	1e-03	0.3	7.66e-05	2.77e-06	3.28
	1e-04	0.9	2.60e-04	2.95e-05	8.92
	1e-05	3	4.67e-05	1.94e-05	29.1

Table 6.1: The sensitivity of the reordering algorithm to the dropping parameter ϵ. Times for computing the spectral ordering are in seconds.

n	$perr$		$pavg$		$2sumerr$		Misses
	max.	mean	max.	mean	max.	mean	
500	3	1.66	1.24e-01	8.01e-02	3.20e-04	8.22e-05	0
1000	7	1.84	1.14e-01	8.25e-02	4.05e-04	3.28e-05	0
2000	5	2.09	1.04e-01	8.34e-02	7.00e-05	1.08e-05	0
3000	5	2.18	1.04e-01	8.25e-02	6.05e-05	7.32e-06	0
5000	10	2.59	9.52e-02	8.18e-02	2.08e-04	9.09e-06	0
10000	16	2.67	8.92e-02	8.05e-02	1.59e-04	5.06e-06	0
20000	21	3.55	8.79e-02	8.05e-02	7.66e-05	2.77e-06	2
40000	35	3.83	8.82e-02	8.03e-02	7.67e-05	2.14e-06	2

Table 6.2: Statistics for the spectral reordering algorithm for a range of values of n.

provide an ordering that unshuffles the matrix. However, occasionally misconvergence does occur for larger values of n.

Our final set of results compares using HSL_MC73 with using the sparse eigensolver EA16. The code EA16 is designed to compute selected eigenvalues of large sparse symmetric problems using an implicitly restarted block Lanczos method. The algorithm used is described in Meerbergen and Scott (2000) and the code is available in HSL. In our tests, we use default settings for the EA16 control parameters. We start by using the simple regular mode (MODE = 1), with a block size of 1, and request the two left-most eigenvalues (WHICH = 2 and NWANT = 2). We take the second eigenvector as an approximation to the Fiedler vector and use this to determine the required spectral ordering. The efficiency of EA16 is dependent upon the number NV of Lanczos vectors used. This parameter must be set by the user. Larger values tend to result in the restarted Lanczos process requiring fewer restarts but more work is performed at each iteration. Furthermore, because the code requires a real work array of size at least $n*$NV, for large problems NV may be limited by the amount of memory available. In our experiments, we set NV = 100 (for $n = 40000$

n	*perr*		*pavg*		2*sumerr*		Misses	
	HSL_MC73	EA16	HSL_MC73	EA16	HSL_MC73	EA16	HSL_MC73	EA16
500	1.66	1.63	7.90e-02	8.01e-02	7.44e-05	8.22e-05	0	0
1000	1.88	1.84	8.10e-02	8.25e-02	2.69e-05	3.28e-05	0	0
2000	2.01	2.09	7.99e-02	8.34e-02	2.69e-05	1.08e-05	0	0
3000	2.03	2.18	7.85e-02	8.25e-02	8.63e-06	7.32e-06	0	0
5000	2.59	2.11	8.18e-02	7.90e-02	9.09e-06	5.18e-06	0	0
10000	2.67	2.50	8.05e-02	7.76e-02	5.06e-06	2.44e-06	0	0
20000	3.55	2.10	8.05e-02	7.75e-02	2.77e-06	1.23e-06	2	0
40000	3.83	2.30	7.03e-02	7.68e-02	2.14e-06	6.17e-07	2	0

Table 6.3: A comparison of HSL_MC73 and EA16 for a range of values of n.

we used NV = 200 since this cut the convergence time by almost half).

In Table 6.3, we compare the maximum and mean values of *perr*, *pavg* and 2*sumerr* for HSL_MC73 and EA16; again 100 examples are run for each n (10 examples for EA16 when $n \geq 10000$) and the number of "misses" (misconvergence of the Fiedler vector) is given. In general, the spectral orderings obtained using the two different approaches are of comparable quality but EA16 appears to be slightly more reliable for large n.

The important advantage of the multilevel approach over EA16 in regular mode is its speed. The average CPU times (in seconds) are recorded in Table 6.4. We see that as n doubles, the time for HSL_MC73 increases by a factor a little over 2. However, as n increases, the cost of EA16 using the regular mode quickly becomes prohibitive. To try and reduce the EA16 time we have also run using the shift-invert mode (MODE = 2). To do this, we have used the useful facility offered by EA16 of allowing the user to start the computation using the regular mode and then automatically switches to shift-invert mode once the code has found a suitable shift σ. The disadvantage of working in shift-invert mode is the need to solve linear systems of the form $(L - \sigma I)\mathbf{x} = \mathbf{b}$. We use the HSL direct solver MA57 to do this (Duff, 2002). $(L - \sigma I)$ is factorized once and then the factors used to solve repeatedly for different right-hand sides. We record in Table 6.4 the time

n	MC73	EA16	
		Regular	Shift-invert
500	0.06	0.16	0.11 (0.02)
1000	0.13	0.54	0.22 (0.04)
2000	0.26	2.10	0.44 (0.09)
3000	0.40	4.53	0.68 (0.15)
5000	0.73	10.2	1.12 (0.24)
10000	1.60	41.0	2.48 (0.55)
20000	3.28	233	12.2 (1.48)
40000	8.19	1385	57.2 (2.68)

Table 6.4: A comparison of the CPU times (in seconds) for HSL_MC73 and EA16 for a range of values of n. The figures in parentheses are the times taken to factorize the shifted Laplacian using MA57.

taken by MA57 for the factorization of the shifted Laplacian. We see that, although EA16 remains more expensive than HSL_MC73, use of the shift-invert mode substantially speeds up the computation of the Fiedler vector.

7 Acknowledgements

We would like to thank Des Higham of the University of Strathclyde for helpful discussions on his work on URDRGs. We are also grateful to Iain Duff and John Reid of the Rutherford Appleton Laboratory for many useful comments on a draft of this report.

References

S.T. Barnard and H.D. Simon. A fast multilevel implmentation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, **6**, 101–117, 1994.

S.T. Barnard, A. Pothen, and H.D. Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications*, **2**, 317–198, 1995.

E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. *in* 'Proceedings of the 24th National Conference of the ACM'. Brandon Systems Press, 1969.

I.S. Duff. MA57– a new code for the solution of sparse symmetric definite and indefinite systems. Technical Report RAL-TR-2002-024, Rutherford Appleton Laboratory, 2002.

I.S. Duff, J.K. Reid, and J.A. Scott. The use of profile reduction algorithms with a frontal code. *Inter. Journal on Numerical Methods in Engineering*, **28**, 2555–2568, 1989.

M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.*, **25**, 619–633, 1975.

W.W. Hager. *Minimizing the profile of a matrix*. Department of Mathematics, University of Florida (www.math.ufl.edu/∼hager/), 2000. To appear in *SIAM J. Scientific Computing*.

B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical report sand93-1301, Sandia National Laboratories, Allbuquerque, NM, 1993.

B. Hendrickson and R. Leland. The Chaco user's guide: Version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque, NM, 1995.

D.J. Higham. Spectral reordering of a range-dependent weighted random graph. Mathematics Research Report 14, Univeristy of Strathclyde, 2003*a*.

D.J. Higham. Unravelling small world networks. *J. Comp. Appl. Maths*, 2003*b*. To appear.

HSL. A collection of Fortran codes for large scale scientific computation, 2002. Full details from www.cse.clrc.ac.uk/nag/hsl/.

Y.F. Hu and J.A. Scott. A multilevel algorithm for wavefront reduction. Technical Report RAL-TR-2000-031, Rutherford Appleton Laboratory, 2000.

Y.F. Hu and J.A. Scott. A multilevel algorithm for wavefront reduction. *SIAM J. Scientific Computing*, **23**, 1352–1375, 2001.

G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, **20**, 359–392, 1999.

G. Kumfert and A. Pothen. Two improved algorithms for envelope and wavefront reduction. *BIT*, **37:3**, 559–590, 1997.

K. Meerbergen and J.A. Scott. The design of a block rational lanczos code with partial reorthogonalization and implicit restarting. Technical Report RAL-TR-2000-011, Rutherford Appleton Laboratory, 2000.

B. Mohar. *The Laplacian Spectrum of Graphs*. J. Wiley and Sons, 1991.

C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numerical Analysis*, **12**, 617–629, 1974.

B.N. Parlett. *The Symmetric Eignevalue Problem*. Prentice Hall, Engglewood Cliffs, New Jersey, 1980.

A. Pothen, H.D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Analysis and Applications*, **11**, 430–452, 1990.

J.K. Reid and J.A. Scott. Ordering symmetric sparse matrices for small profile and wavefront. *Inter. Journal on Numerical Methods in Engineering*, **45**, 1737–1755, 1999.

J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). *in* S. F. McCormick, ed., 'Multigrid Methods', Vol. 3 of *Frontiers in Applied Mathematics*, pp. 73–130. SIAM, Philadelphia, PA, 1987.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machnine Intelligence*, **22**, 888–905, 2000.

H.D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, **2**, 135–148, 1991.

S.W. Sloan. An algorithm for profile and wavefront reduction of sparse matrices. *Inter. Journal on Numerical Methods in Engineering*, **23**, 1315–1324, 1986.