Science and Technology Facilities Council

HSL_MC68

C INTERFACE HSL

1 SUMMARY

Given a symmetric sparse matrix $A = \{a_{ij}\}_{n\times n}$, HSL_MC68 **computes elimination orderings** that are suitable for use with a sparse direct solver. Currently the following choices are available:

- Approximate minimum degree ordering (with provision for some dense rows and columns)
- Minimum degree ordering using the methodology of MA27
- Nested bisection ordering using MeTiS
- MA47 ordering for indefinite matrices which may generate a combination of both 1×1 and 2×2 pivots

The lower triangular part of A must be supplied in compressed sparse column format. The HSL package HSL_MC69 may be used to convert data held in other sparse matrix formats and also to check the user's matrix data for errors.

ATTRIBUTES — Version: 3.3.3 (30 March 2023). **Types:** Integer. **Calls:** HSL_ZB01 and (optionally using MeTiS version 4.x) METIS_NODEND. **Language:** Fortran 2003 subset (F95 + TR 15581 + C interoperability). **Interfaces:** Fortran, C. **Original date:** September 2008. **Origin:** H. S. Dollar and J. A. Scott, Rutherford Appleton Laboratory. **Remark:** The development of this package was supported by EPSRC grants GR/S42170 and EP/E053351/1.

2 HOW TO USE THE PACKAGE

2.1 C interface to Fortran code

This package is written in Fortran and a wrapper is provided for C programmers. This wrapper implements a (large) subset of the full functionality available via the Fortran interface.

The wrapper will automatically convert between 0-based (C) and 1-based (Fortran) indexing, so may be used transparently from C. This conversion involves both time and memory overheads that may be avoided by supplying data that is already stored using 1-based indexing. The conversion may be disabled by setting the control parameters control.f_array_in and control.f_array_out to true and supplying all data using 1-based indexing. Except where stated, this document describes all arrays in 0-based (C) indexing only.

The wrapper uses the Fortran 2003 interoperability features. **Matching C and Fortran compilers must be used**, for example gcc and gfortran, or icc and ifort. If you do not use the Fortran compiler to link your program, then you may need to link additional Fortran compiler libraries explicitly.

2.2 Calling sequences

Access to the package requires a USE statement of the form

```
#include "hsl_mc68i.h"
```

To initialise an mc68_control structure with default values, mc68_default_control may be called. This should be done before calling mc68_order.

To compute an elimination order, mc68_order should be called. This accepts a sparse symmetric matrix that is stored using compressed sparse column format: this may be setup and checked using the HSL_MC69 package, see Section 2.4.1.

2.3 The derived data types

The user must employ the structures defined in the header file to declare scalars of type mc68_control and mc68_info. The following pseudocode illustrates this.

```
#include "hsl_mc68i.h"
...
struct mc68_control control;
struct mc68_info info;
...
```

The members of struct mc68_control and struct mc68_info are described in Section 2.5.2 and Section 2.5.3, respectively.

2.4 MeTiS

The HSL_MC68 package uses the MeTiS graph partitioning library available from the University of Minnesota website. If MeTiS is not available, then the user must compile with the supplied replacement subroutine METIS_NodeND. In this case, the MeTiS ordering option will not be available to the user and, if selected, mc68_order will return with an error.

Important: At present, HSL_MC68 only supports MeTiS version 4, not the latest version 5 releases.

2.4.1 Input of the matrix *A*

The user must input the **lower** triangular part of the matrix *A* using the arguments ptr and row as described in Section 2.5.1.

We recommend that standard HSL format is used to ensure compatibility with other HSL routines. This is a compressed sparse column format with the entries within each column ordered by increasing row index. There is no requirement that zero entries on the diagonal are explicitly included. **No checks** are made on the user's data. It is important to note that any out-of-range entries or duplicates may cause HSL_mc68 to fail in an unpredictable way. Before using HSL_mc68, the HSL package HSL_MC69 may be used to check for errors and to handle duplicates (HSL_MC69 sums them) and out-of-range entries (HSL_MC69 removes them).

If the user's data is held using another standard sparse matrix format (such as coordinate format or sparse compressed row format), we recommend using a conversion routine from HSL_MC69 to put the data into standard HSL format. The input of *A* is illustrated in Section 5.

2.5 To set default values

Default values for the components of the mc68_control structure may be set by a call to ma68_default_control.

```
void mc68_default_control(struct mc68_control *control)
```

control has its components set to their default values, as described in Section 2.5.2.

2.5.1 To compute a symmetric elimination order

The method constructs an elimination order for a sparse symmetric matrix, *A*, using a chosen ordering method.

```
void mc68_order(const int ord, const int n, const int ptr[], const int row[],
   int perm[], const struct mc68_control *control, struct mc68_info *info)
```

ord must be set by the user to declare which ordering is to be used.

1 An approximate minimum degree ordering is used.

2 A minimum degree ordering is used (as in MA27).

3 MeTiS ordering with default settings is used. Note that the user needs to supply the MeTiS library. If it is not supplied and this option is requested, the routine will return immediately with info.flag set to -5.

4 MA47 ordering for indefinite matrices is used.

n must hold the order of A.

ptr is a rank-one array of size n+1. ptr[j] must be set so that ptr[j] is the position in row of the first entry in column j and ptr[n] must be set to the total number of entries in the lower triangular part of A.

row is a rank-one array of size ptr[n]. It must hold the row indices of the entries of A (only the **lower** triangular part), with the row indices for the entries in column 0 preceding those for column 1, and so on.

perm is a rank-one array of size at least n. On exit, it specifies the elimination order. If i is used to index a variable, then abs(perm[i]) holds its position in the pivot sequence. If i, $1 \leq$ i $\leq$ n, is not used to index a variable (that is, column i is empty), then perm(i) is equal -1 (0 if Fortran indexing is used). 1×1 and 2×2 pivots are differentiated only if Fortran array indexing is used for output (control.f_array_out=1). If a 1×1 pivot i is obtained, then perm[i]$>$0. If a 2×2 pivot involving variables i and j is obtained, then perm[i]$<$0, perm[j]$<$0 and |perm[j]|=|perm[i]|+1. Note that if ord$\neq$4, then only 1×1 pivots will be obtained.

control see Section 2.5.2.

info provides information about the execution of the subroutine, as explained in Section 2.5.3. In particular, info.flag is used as an error/warning flag. Negative values indicate an error. Possible negative values for info.flag are:

-1 memory allocation failed. If available, the stat parameter is returned in info.stat.

-2 memory deallocation failed. If available, the stat parameter is returned in info.stat.

-3 n$<$1.

-4 ord is not associated with an ordering.

-5 MeTiS ordering was requested but MeTiS not linked.

-6 error during call to HSL_ZB01. The error flag from HSL_ZB01 is returned in info.zb01 and, if available, the iostat parameter is returned in info.iostat. The user may attempt to avoid the internal call to HSL_ZB01 by rerunning mc68_order with the optional argument min_l_workspace present and set to be larger than the value that has been returned in info.l_workspace (we recommend at least 10% larger).

Positive values for info.flag are associated with a warning. Possible positive values for info.flag are:

+1 ord=4 and A has no non-zero diagonal entries.

+2 ord=4 and some zero eigenvalues were detected in the structure of A.

+3 ord=4, A has no non-zero diagonal entries and some zero eigenvalues were detected in its structure.

2.5.2 The control structure for holding control parameters

The structure mc68_control is used to control the action. The user must declare a structure of type mc68_control. The function mc68_default_control may be called to set the components of the structure to the stated default values. The following components are employed:

int f_array_in indicates whether to use C or Fortran array indexing for input. If f_array_in evaluates to true then 1-based indexing is used for ptr and row. This means that the first row of a matrix is row 1, not row 0, and that the entries row are numbered (for the purposes of ptr only) from 1, not from 0; row[0] should still be used for the first entry, even though ptr[0]=1. Otherwise, if f_array_in evaluates to false, then these arrays are copied and converted to 1-based indexing in the wrapper function. The default is f_array_in=0 (false).

int f_array_out indicates whether to use C or Fortran array indexing for output. If f_array_out evaluates to true then order contains a permutation (in absolute value) of the numbers $1,\ldots n$. Otherwise, if f_array_out evaluates to false, then it contains a permutation of the numbers $0,\ldots n-1$ (converted from $1,\ldots n$ in the wrapper function, which discards any sign information as ± 0 cannot be distinguished). The default is f_array_out=0 (false).

int min_l_workspace specifies the minimum length of int workspace that is allocated within mc68_order. The length of workspace used may be greater than min_l_workspace and will be returned in info.l_workspace. If min_l_workspace$\leq$0 then the package will decide. If info.n_compressions is greater than 10, then rerunning mc68_order on the same problem with min_l_workspace>info.l_workspace may make the method more efficient. The default is min_l_workspace=0.

int lp is the Fortran unit on which error messages are printed. If it is negative, these messages will be suppressed. The default value is 6 (stdout).

int wp is the Fortran unit on which warning messages are printed. If it is negative, these messages will be suppressed. The default value is 6 (stdout).

int mp is the Fortran unit on which diagnostic messages are printed. If it is negative, these messages will be suppressed. The default value is 6 (stdout).

int print_level indicates the level of diagnostic printing desired. The levels are:

<0 no printing.

0 error and warning messages only.

1 as 0 plus basic diagnostic messages.

2 as 1 plus some more detailed diagnostic messages.

The default value is 0. Values greater than 2 are treated as 2.

int row_full_thresh is used by mc68_order to declare a threshold on the number of entries in a row to determine whether a row is dense when ord=2 (if ord=1, then a different strategy is used to detect dense rows). This threshold is given as a percentage. The default is 100.

int row_search is used by mc68_order when ord=4. If row_search is less than or equal to 1, then the pivot order is obtained using the Markowitz strategy. If row_search is greater than 1, then each search for a structured pivot is limited to this number of rows.

2.5.3 The structure for holding information

The structure mc68_info is used to hold information from the execution of mc68_setup and mc68_order. The components are:

int flag is used as an error/warning flag. Negative values indicate a fatal error and positive values are associated with a warning.

int iostat holds the Fortran iostat parameter.

int l_workspace holds the length of int workspace used by mc68_order.

int n_compressions holds the number of compresses of the workspace that mc68_order performed. If n_compressions is greater than 10, then rerunning mc68_order on the same problem with control.min_l_workspace>info.l_workspace may make the method more efficient.

int n_dense_rows holds the number of dense rows detected during mc68_order when ord = 1 or ord = 2. The value -1 will be returned for ord > 2.

int n_zero_eigs holds the number of zero eigenvalues detected in the structure of A. A negative value will be returned for orderings that do not detect zero eigenvalues.

int stat holds the Fortran stat parameter.

int zb01_info holds the error/warning flag returned by the last call to HSL_ZB01 during the routine.

3 GENERAL INFORMATION

Input/output: Error, warning and diagnostic messages, and I/O to sequential-access files whose unit numbers are chosen by HSL_ZB01. Error messages on unit control.lp and warning and diagnostic messages on units control.wp and control.mp, respectively. These have default value 6; printing of these messages is suppressed if the relevant unit number is negative or if print_level is negative.

Restrictions: n≥0, 1≤ord≤4.

4 METHOD

mc68_order constructs an elimination ordering for a symmetric matrix using a chosen algorithm. This elimination ordering may then be used by a sparse direct solver, such as HSL_MA77. If ord=1, an approximate minimum degree ordering is formed using the AMDD method described in [1]: this allows for the efficient detection and handling of dense or almost dense rows; if the matrix has neither dense nor almost dense rows, then this will not cause any additional overhead. If ord=2, a minimum degree ordering is obtained using the same methodology as that with default settings in MA27 [2]. This option also detects and handles dense or almost dense rows but a simpler detection method is used. A nested bisection ordering is formed by calling MeTiS [4] with its default settings when ord=3: this ordering can only be formed if MeTiS is linked. If ord=4, an ordering using the same methodology as that in MA47 is used [3]. This method chooses diagonal pivots of orders 1 and 2 using the Markowitz criterion. Because of the facility for handling matrices with zeros on the diagonal, the 2×2 pivots can be of the form

$$\begin{pmatrix} 0 & x \\ x & 0 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 0 & x \\ x & x \end{pmatrix}$$

called oxo and tile pivots respectively.

References

[1] H. S. Dollar and J. A. Scott. A note on fast approximate minimum degree orderings for symmetric matrices with some dense rows. Numerical Linear Algebra with Applications, 17(2009), pp. 43–55.

[2] I. S. Duff and J. K. Reid. MA27 - A set of Fortran subroutines for solving sparse symmetric sets of linear equations. Technical Report AERE R-10533, HMSO, London, 1982.

[3] I. S. Duff and J. K Reid. MA47, a Fortran code for direct solution of indefinite sparse symmetric linear systems. Technical Report RAL-95-001, Didcot, Oxon, UK, 1995.

[4] G. Karypis and V. Kumar. *MeTis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, Version 4.0*, 1998.

5 EXAMPLES OF USE

5.1 Example 1

In our first example, we give the code required to generate elimination orderings using `HSL_MC68` when input is by sparse column format and the values of the entries are given. Suppose we wish to find the elimination ordering generated by the approximate minimum degree method and `MA47` elimination ordering for the following indefinite matrix:

$$\begin{pmatrix} x & x & x & x \\ x & x & & \\ x & & x & x \\ x & & x & \end{pmatrix}$$

The following code may be used

with the following data supplied in stdin:

This produces the following output:

5.2 Example 2

In our second example, we set-up the same matrix data as that in Example 1 by using `HSL_MC69` and then use `mc68_order` to compute the approximate minimum degree and `MA47` elimination orderings. We initially store the lower and upper triangular parts of the matrices using sparse coordinate format. The following code may be used.

with the following data supplied on stdin:

This produces the following output: