HSL_MC68

PACKAGE SPECIFICATION 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). **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 Calling sequences

Access to the package requires a USE statement of the form

```
USE HSL_MC68_integer
```

The following equivalent USE statements are also provided for backwards compatibility with previous versions. They are **deprecated** and may be removed at a later date.

```
USE HSL_MC68_single
USE HSL_MC68_double
```

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 2.4.3.

2.2 The derived data types

The user must employ the derived types defined by the module to declare scalars of type MC68_control and MC68_info. The following pseudocode illustrates this.

```
use hsl_mc68_double
...
type (mc68_control) :: control
type (mc68_info)    :: info
```

The components of MC68_control and MC68_info are described in Section 2.4.5 and Section 2.4.6, respectively.

2.3 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 Argument lists and calling sequences

2.4.1 Optional arguments

We use square brackets [ ] to indicate OPTIONAL arguments. In each call, optional arguments follow the argument info. Since we reserve the right to add additional optional arguments in future releases of the code, **we strongly recommend that all optional arguments are called by keyword, not by position.**

2.4.2 Integer types

INTEGER denotes default INTEGER and INTEGER(long) denotes INTEGER(kind=selected_int_kind(18)).

2.4.3 Input of the matrix *A*

The user must supply the **lower** triangular part of the matrix *A* in standard HSL format. 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.

2.4.4 To compute a symmetric elimination order

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

```
CALL MC68_order(ord,n,ptr,row,perm,control,info[,min_l_workspace])
```

ord is an INTENT(IN) scalar of type INTEGER. It 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 is an INTENT(IN) scalar of type INTEGER that must hold the order of *A*.

ptr is an INTENT(IN) rank-one array of type INTEGER and 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+1) must be set to one more than the total number of entries in the lower triangular part of *A*.

`row` is an `INTENT(IN)` rank-one array of type `INTEGER`. The first `ptr(n+1)-1` entries must hold the row indices of the entries of A (only the **lower** triangular part), with the row indices for the entries in column 1 preceding those for column 2, and so on.

`perm` is an `INTENT(INOUT)` rank-one array of type `INTEGER` and 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 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`. If `i`, `1`$\leq$`i`$\leq$`n`, is not used to index a variable (that is, column i is null), then `perm(i)` is equal to zero. Note that if `ord`$\neq$`4`, then only 1×1 pivots will be obtained.

`control` is an `INTENT(IN)` scalar of type `mc68_control` (see Section 2.4.5).

`info` is an `INTENT(OUT)` scalar of type `mc68_info`. Its components provide information about the execution of the subroutine, as explained in Section 2.4.6. 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.

`min_l_workspace` is an `OPTIONAL` scalar of type `INTEGER` with `INTENT(IN)` that may be set by the user to the minimum length of `INTEGER` 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 `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.

2.4.5 The control derived data type for holding control parameters

The derived data type `MC68_control` is used to control the action. The user must declare a structure of type `MC68_control`. Components of this derived type are automatically given their default values in the definition of the type: the user does not need to set them unless values other than the defaults are required. The following components are employed:

`lp` is an `INTEGER` scalar that is used as the output stream for error messages. If it is negative, these messages will be suppressed. The default value is `6`.

`wp` is an `INTEGER` scalar that is used as the output stream for warning messages. If it is negative, these messages will be suppressed. The default value is `6`.

mp is an INTEGER scalar that is that is used as the output stream for diagnostic messages. If it is negative, these messages will be suppressed.The default value is 6.

print_level is an INTEGER scalar indicating 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.

row_full_thresh is an INTEGER scalar that 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.

row_search is an INTEGER scalar that 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.

2.4.6 The derived data type for holding information

The derived data type MC68_info is used to hold information from the execution of MC68_setup and MC68_order. The components are:

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

iostat is an INTEGER scalar that holds the Fortran iostat parameter.

l_workspace is an INTEGER(long) scalar that holds the length of INTEGER workspace used by MC68_order.

n_compressions is an INTEGER scalar that 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 the optional parameter min_l_workspace>l_workspace may make the method more efficient.

n_dense_rows is an INTEGER scalar that 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.

n_zero_eigs is an INTEGER scalar that 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.

stat is a scalar of type INTEGER that holds the Fortran stat parameter.

zb01_info is a scalar of type INTEGER that 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: $n \geq 0$, $1 \leq ord \leq 4$.

Changes from Version 1.0.0 The approximate minimum degree method that the packages uses has been changed. In Version 1.0.0, the package called `MC47`. In Version 2.0.0, the package uses the `AMDD` method described in [1]. Accordingly, the `restarts` component has been removed from the data type for holding information and the `n_dense_rows` component has been added.

Changes from Version 2.0.1 In Version 3.0.0, `mc68_setup` has been removed from the package and replaced by a reference to `HSL_MC69`. Accordingly, in the data type for holding information, the possible values of `flag` component have been updated, and the components `duplicate` and `out_range` have been removed. The package no longer requires `HSL_ZD11` and the matrix is now input using a rank-1 arrays.

Changes from Version 3.2.0 In Version 3.3.0 the kind of `info%l_workspace` changed from default integer to `INTEGER(long)`.

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

```
    PROGRAM main
      USE hsl_mc68_double
      IMPLICIT NONE

! Local variables
      INTEGER :: n, ne

      INTEGER, DIMENSION (:), ALLOCATABLE :: row, ptr, perm

      TYPE (mc68_control) :: control
      TYPE (mc68_info) :: info

! Read in the order n of the matrix and the number
! of non-zeros in its lower triangular part.
      READ (5,*) n, ne

! Allocate arrays
      ALLOCATE (row(ne),ptr(n+1),perm(n))

! Read in pointers for lower triangular part of matrix
      READ (5,*) ptr(1:n+1)

! Read in column indices for lower triangular part of matrix
      READ (5,*) row(1:ne)

! Compute elimination order using approximate minimum degree method
      CALL mc68_order(1,n,ptr,row,perm,control,info)
      WRITE (6,'(a)') ' Approximate minimum degree ordering : '
      WRITE (6,'(8i4)') perm
      WRITE (6,'(a)') ' '

! Compute elimination order using MA47 method
      CALL mc68_order(4,n,ptr,row,perm,control,info)
      WRITE (6,'(a)') ' MA47 ordering : '
      WRITE (6,'(8i4)') perm

! Deallocate all arrays
      DEALLOCATE (row,ptr,perm)

    END PROGRAM main
```

with the following data:

```
4 7
1 5 6 8 8
1 2 3 4 2 3 4
```

This produces the following output:

```
 Approximate minimum degree ordering :
   4   1   2   3

 MA47 ordering :
   4   1  -3  -2
```

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.

```
    PROGRAM main
      USE hsl_mc68_integer
      USE hsl_mc69_double
      IMPLICIT NONE

! Local variables
      INTEGER :: n, ne_in, matrix_type, flag

      INTEGER, DIMENSION (:), ALLOCATABLE :: row_in, col_in, row, ptr, perm

      TYPE (mc68_control) :: control
      TYPE (mc68_info) :: info

! Read in the order n of the matrix and the total number
! of non-zeros in the lower and upper triangular parts of A.
      READ (5,*) n, ne_in

! Allocate arrays
      ALLOCATE (row_in(ne_in),col_in(ne_in),ptr(n+1),perm(n))

! Read in row indices for lower and upper triangular parts of A.
      READ (5,*) row_in(1:ne_in)

! Read in column indices for lower and upper triangular parts of A.
      READ (5,*) col_in(1:ne_in)

! Convert matrix into standard HSL format
      matrix_type = 4
      CALL mc69_coord_convert(matrix_type,n,n,ne_in,row_in,col_in,ptr,row, &
        flag)
      WRITE (6,'(a,i4)') 'n:', n
      WRITE (6,'(a)') 'ptr:'
      WRITE (6,'(8i4)') ptr
      WRITE (6,'(a)') 'row:'
      WRITE (6,'(8i4)') row

! Compute elimination order using approximate minimum degree method
      CALL mc68_order(1,n,ptr,row,perm,control,info)
      WRITE (6,'(a)') ' Approximate minimum degree ordering : '
      WRITE (6,'(8i4)') perm
      WRITE (6,'(a)') ' '

! Compute elimination order using MA47 method
      CALL mc68_order(4,n,ptr,row,perm,control,info)
      WRITE (6,'(a)') ' MA47 ordering : '
      WRITE (6,'(8i4)') perm

! Deallocate all arrays
      DEALLOCATE (row_in,row,col_in,ptr,perm)
```

```
    END PROGRAM main
```

with the following data:

```
4 11
1 1 1 1 2 2 3 3 3 4 4
1 2 3 4 1 2 1 3 4 1 3
```

This produces the following output:

```
n:    4
ptr:
   1    5    6    8    8
row:
   1    2    3    4    2    3    4    2
   3    4    4
 Approximate minimum degree ordering :
   4    1    2    3

 MA47 ordering :
   4    1   -3   -2
```