1 SUMMARY

Let A be an $n \times n$ matrix with a symmetric sparsity pattern. HSL_MC70 computes a **nested dissection ordering** of A that is suitable for use with a sparse direct solver. The algorithm allows for some dense or nearly dense rows and columns in A.

The algorithm partitions the rows/columns of A into 3 sets such that reordering the rows/columns to respect to their partitions yields a symmetric matrix of the form

$$\left(\begin{array}{c|c|c} A_1 & 0 & S_1^T \\ \hline 0 & A_2 & S_2^T \\ \hline S_1 & S_2 & S \end{array}\right). \tag{1.1}$$

The rows of S are given an arbitrary order. If the dimension of A_1 (A_2) is smaller than some predefined value, the rows of A_1 (A_2) will be ordered using an approximate minimum degree algorithm; otherwise, the rows/columns of A_1 (A_2) will be partitioned to form another matrix with the above structure and the algorithm will continue to be applied in a recursive manner.

ATTRIBUTES — Version: 1.0.0. **Types:** Real (single, double). **Uses:** HSL_MC78 and HSL_MC79. **Language:** Fortran 2003 subset (F95+TR155581) **Date:** January 2014. **Origin:** I.S. Duff, J.A. Scott and H.S. Thorne, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Calling sequences

Access to the package requires a USE statement of the form

```
USE HSL_MC70_integer
```

2.2 The derived data types

For each problem, the user must employ the derived types defined by the module to declare scalars of the types MC70_control and MC70_info. The following pseudocode illustrates this.

```
use HSL_MC70_integer
...
type (MC70_control) :: control
type (MC70_info) :: info
...
```

The components of MC70_control and MC70_info are explained in Sections 3.3 and 3.4.

3 THE ARGUMENT LISTS

3.1 Input of the matrix A

The user must supply the pattern of either the **lower triangular part** of the matrix A or the **lower and upper triangular** part of A in a compressed sparse column format. 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_MC70 to fail in an unpredictable way. Before using HSL_MC70, 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 the required format. The input of A is illustrated in Section 6.

3.2 To compute a nested dissection ordering

If the user has the **lower triangular part of** A held in compressed sparse columns format, a call of the following form should be made:

```
CALL mc70_order(n,ptr,row,perm,control,info)
```

If the user has the **lower and upper triangular parts of** A held in compressed sparse columns format, a call of the following form should be made:

```
CALL mc70_order_full(n,ptr,row,perm,control,info)
```

n is an INTEGER scalar with INTENT(IN). On entry it must hold the order n of A. **Restriction:** n $\geq$ 1.

ptr is an INTEGER array of rank one with INTENT(IN) and size n+1. It must be set by the user so that ptr(j) is the position in row of the first entry in column j (j=1,2,...,n) and ptr(n+1) must be set to one more than the total number of entries.

row is an INTEGER array of rank one with INTENT(IN) and size ptr(n+1)-1. On a call to mc70_order, it must be set by the user so that row(1:ptr(n+1)-1) holds the row indices of the entries in the **lower triangular part** of A; on a call to mc70_order_full, it must be set by the user so that row(1:ptr(n+1)-1) holds the row indices of the entries in the **lower and upper triangular parts** of A. The entries in a single column must be contiguous. The entries of column j must precede those of column j+1 (j=1,2,...,n-1), and there must be no wasted space between the columns. Row indices within a column may be in any order. Diagonal entries are ignored.

perm is an INTEGER array with INTENT(OUT) and size n. On exit, perm holds the nested dissection ordering. The position of variable i in the nested dissection ordering is perm(i), i=1,2,...,n.

control is a scalar of type MC70_control with INTENT(IN). Its components control the action, as explained in Section 3.3.

info is a scalar of type MC70_info with INTENT(OUT). Its components hold information, as explained in Section 3.4.

3.3 The control derived data type

The derived data type MC70_control is used to control the action. The user must declare a structure of type MC70_control. The components, which are automatically given default values in the definition of the type, are:

Printing controls

print_level is a scalar of type INTEGER that is used to controls the level of printing. The different levels are:

- $<$ 0 No printing.
- $=$ 0 Error messages only.

$= 1$ As 0, plus basic diagnostic printing.

> 1 As 1, plus some additional diagnostic printing.

The default is print_level=0.

unit_diagnostics is a scalar of type INTEGER that holds the unit number for diagnostic printing. Printing is suppressed if unit_diagnostics< 0. The default is unit_diagnostics=6.

unit_error is a scalar of type INTEGER that holds the unit number for error messages. Printing of error messages is suppressed if unit_error$<$0. The default is unit_error=6.

Other controls

max_levels is an INTEGER scalar that holds the maximum number of nested dissection levels. If the maximum number of nested dissection levels is reached, the corresponding matrix in the nested dissection hierarchy will be ordered using an approximate minimum degree algorithm. If max_levels = 0, the ordering of the input matrix is computed using an approximate minimum degree algorithm. Values less than 0 are treated as 0. The default is max_levels=20.

nd_switch is an INTEGER scalar that is used in the criteria for determining what happens a matrix within the nested dissection hierarchy. If the matrix order is greater than nd_switch and the maximum number of nested dissection levels has not been reached, the matrix will be partitioned; otherwise, it will be ordered using an approximated minimum degree algorithm. Values less than 2 are treated as 2. The default is nd_switch=50.

ratio is a DOUBLE PRECISION scalar. The partitioning and refinement methods aim to find partitions (1.1) such that $\max(n_1,n_2) < \min(n_1,n_2) \times$ ratio, where A_1 has order n_1 and A_2 has order n_2. If several candidate partitions satisfy this requirement, then the partition with the smallest value of $\frac{n_s}{n_1 n_2}$ is chosen, where S has order n_s; if none of the candidate partitions satisfy this requirement, the partition that has the smallest value of $\frac{n_s}{n_1 n_2}$ is chosen. Decreasing ratio will, in general, result in a nested dissection ordering that is more amenable to parallel direct solvers; increasing ratio will, in general, reduce the number of non-zeros in the Cholesky factorization of the reordered matrix. The default is ratio=1.5.

remove_dense is a LOGICAL scalar. If remove_dense = .true., then the input matrix is searched for dense (or nearly dense) rows and columns, and the nested dissection algorithm is applied to the matrix that results when these rows and columns are removed. Dense rows/columns are placed at the end of the ordering. If remove_dense = .false., then the input matrix is not searched for dense rows. The default is remove_dense=.true..

3.4 The derived data type for holding information

The derived data type MC70_info is used to hold parameters that give information about the algorithm. The components of MC70_info (in alphabetical order) are:

flag is a scalar of type INTEGER that gives the exit status of the algorithm (details in Section 3.5).

ndense is a scalar of type INTEGER that holds the number of rows/columns in the input matrix that were determined to be dense. If control%remove_dense = .false., the input matrix is not checked for dense rows.

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

3.5 Error diagnostics

On successful completion, MC70_nested will exit with info%flag set to 0. Other values for info%flag are associated with a fatal error. Possible values are:

-1 memory allocation failed.

-2 memory deallocation failed.

-3 n ≤ 0.

4 GENERAL INFORMATION

Workspace: Provided automatically by the module.

Other modules used directly: HSL_MC78 and HSL_MC79.

Input/output: Error and diagnostic messages. Error messages on unit control%unit_err and diagnostic messages on unit control%unit_diagnostics. These have default value 6; printing of these messages is suppressed if the relevant unit number is negative or if control%print_level is negative.

Restrictions: n $\geq$ 1.

Portability: Fortran 2003 subset (F95+TR155581)

5 METHOD

Given a symmetric matrix A of order n, HSL_MC70 preprocesses the matrix to (optionally) remove dense or almost dense rows/columns, returning $\overline{A}$. The matrix $\overline{A}$ is then compressed using HSL_MC78 to give a symmetric matrix $\widetilde{A}$ with order $\tilde{n}$. This matrix is passed to the recursive nested dissection algorithm, Algorithm 1, and the resulting permutation matrix is converted into the corresponding elimination ordering. Having formed a nested dissection ordering for $\widetilde{A}$, the ordering is mapped to give an ordering for $\overline{A}$. The dense rows are appended to the end of the ordering.

Algorithm 1 Nested dissection algorithm

recursive subroutine nested_dissection(A, P)
Input: symmetric matrix A of order n
Output: permutation matrix P
if $n <$ nd_switch **then**
Form the AMD elimination ordering, p, for A and return its equivalent permutation matrix
else
Partition the matrix: compute P, a permutation matrix, such that $P^T AP$ has the form (1.1)
Call **nested_dissection**(A_1, P_1)
Call **nested_dissection**(A_2, P_2)
Perform the update $P = PQ$, where

$$Q = \begin{pmatrix} P_1 & & \\ & P_2 & \\ & & I \end{pmatrix}$$

end if

5.1 Partitioning a matrix

5.2 Refinement

6 EXAMPLE OF USE