MC62

PACKAGE SPECIFICATION HSL

1 SUMMARY

Given an $m \times n$ matrix $\mathbf{A}=\{a_{ij}\}$ with an unsymmetric sparsity pattern, this subroutine **generates a row ordering** for a row-by-row frontal solver. If $m=n$ the ordering may be used by the HSL packages `MA42` and `MA43`. The user may optionally specify which variables are candidates for elimination. This option allows the subroutine to be used to generate row orderings for use with a parallel row-by-row frontal solver.

We introduce some definitions for the frontal method. If $frow_i$ denotes the number of rows in the front at the ith elimination and p is the total number of eliminations performed, the maximum row frontsize is defined as

$$\max_{1 \le i \le p} \{frow_i\},$$

and the mean row frontsize is

$$\frac{1}{p}\sum_{i=1}^{p} frow_i .$$

The maximum and mean column frontsizes are defined similarly with $fcol_i$ the number of columns in the front at the ith elimination. Note that if $\mathbf{A}$ is an $n \times n$ structurally nonsingular matrix and all the variables are candidates for elimination then $p=n$.

The maximum frontal matrix size is

$$\max_{1 \le i \le p} (frow_i * fcol_i) ,$$

and the mean frontal matrix size is

$$\frac{1}{p}\sum_{i=1}^{p} (frow_i * fcol_i) .$$

`MC62` generates a row ordering that is designed to reduce the maximum and mean row and column frontsizes, the maximum and mean frontal matrix size, which in turn reduce storage requirements and operation counts for the frontal solver. Only the pattern of the matrix is used. `MC62` is not recommended if $\mathbf{A}$ has one or more rows that are full or have a large number of nonzeros.

`MC62` offers the option of generating the row graph of $\mathbf{A}$. The nodes of the row graph correspond to the rows of $\mathbf{A}$ and two rows i and j $(i \ne j)$ are defined to be adjacent if and only if there is at least one column k of $\mathbf{A}$ for which $a_{ik}.a_{jk} \ne 0$.

Further details are given in *J.A. Scott (1998). A new row ordering strategy for frontal solvers, Technical Report RAL-TR-1998-056, Rutherford Appleton Laboratory.*

ATTRIBUTES — **Version:** 1.0.0. (12 July 2004) **Types:** Real (single, double). **Calls:** `MC38`, `MC60`. **Original date:** June 2000. **Origin:** J. A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequence

There are four entries:

(a) `MC62I/ID` sets default values for control parameters for `MC62A/AD`. It should normally be called once prior to calling `MC62A/AD`.

(b) `MC62A/AD` constructs a representation of **A** by rows from a representation by columns (or a representation of **A** by columns from a representation by rows) and reorders the rows.

(c) `MC62B/BD` computes, for a representation of **A** by rows and a given row order, the maximum and mean row and column frontsizes, the maximum and mean frontal matrix size.

(d) `MC62C/CD` computes the row graph of an $m \times n$ matrix **A**, given a representation by rows and a representation by columns.

Only `MC62A/AD` checks the data. `MC62B/BD` is called by `MC62A/AD` but may also be used in combination with another algorithm for choosing a row order. `MC62C/CD` is also called by `MC62A/AD` but may be used without calling any other routines from the `MC62` package to generate the row graph of a matrix. `MC62C/CD` requires a representation of **A** by rows and a representation by columns. Given a representation by rows, a representation by columns can be constructed using the HSL routine `MC38A/AD`.

2.1.1 To set default values for the control parameters

The single precision version

```
CALL MC62I(ICNTL)
```

The double precision version

```
CALL MC62ID(ICNTL)
```

`ICNTL` is an `INTEGER` array of length `10` that need not be set on entry. This array is used to hold control parameters for `MC62A/AD`. On return from `MC62I/ID`, `ICNTL` contains default values. If the user wishes to use values other than the defaults, the corresponding entries in `ICNTL` should be reset after the call to `MC62I/ID`. The control parameters are:

`ICNTL(1)` is the unit number for error messages and has the default value `6`. Printing of error messages is suppressed if `ICNTL(1)` < `0`.

`ICNTL(2)` is the unit number for printing warnings and computed statistics. It has the default value `6`. Printing of warnings and statistics is suppressed if `ICNTL(2)` < `0`.

`ICNTL(3)` controls the form of the input data. If `ICNTL(3) = 0`, the user must input the matrix using column indices and row pointers in `JCN` and `ROWPTR`, respectively. If `ICNTL(3) = 1`, the user must input the matrix using row indices and column pointers in `IRN` and `COLPTR`, respectively. The default value is `0`. **Restriction:** `ICNTL(3)` = `0` or `1`.

`ICNTL(4)` controls the algorithm used. If `ICNTL(4)` = `0`, the MSRO algorithm is used. If `ICNTL(4)` = `1`, the MSRO and the RMCD algorithms are both used and the best ordering selected. If `ICNTL(4)` = `2`, only the RMCD algorithm is used. Details of the algorithms are given in Section 4. The default value is `0`. **Restriction:** `ICNTL(4)` = `0`, `1`, or `2`.

`ICNTL(5)` is only used if `ICNTL(4)` = `0` or `1`. `ICNTL(5)` controls the priority function and weights as follows:

If `ICNTL(5) = 1`, the user must supply a priority vector in `ORDER` and weights for the priority function in `WT`.

If `ICNTL(5) = 2`, the user must supply a priority vector in `ORDER` (weights for the priority function are generated automatically).

If `ICNTL(5) = 3`, the user must supply weights for the priority function in `WT` (the priority vector is generated automatically).

For all other values of `ICNTL(5)`, the priority vector and weights are generated automatically (see Section 4). The default value is `0`.

`ICNTL(6)` is the minimum number of variables that may be eliminated at a single stage (zero and negative values are treated as `1`). The default value is `1`.

`ICNTL(7)` controls whether or not the user wishes to restrict which variables may be eliminated. If `ICNTL(7) = 1`, the user must specify which variables may be eliminated using LELIM. Otherwise, it is assumed that each of the variables is a candidate for elimination. The default value is 0.

ICNTL(8) controls whether an inexpensive initial calculation is performed to check the length LIW of the workarray IW provided by the user (ICNTL(4) = 0 or 1 only). If ICNTL(8) ≠ 0, the calculation is performed; if LIW is less than the computed length, an error is issued and the computation terminates. The user should then recall MC62A/AD with LIW set to the value returned in INFO(2). If ICNTL(8) = 0, the call to MC62A/AD may be successful using a smaller value of LIW but, if LIW is too small, the computation to compute a necessary and sufficient value for LIW is more expensive. The default value is 0.

ICNTL(9) to ICNTL(10) are currently not used but are given the default value 0 by MC62I/ID.

2.1.2 To reorder the rows

The single precision version

```
      CALL MC62A(M,N,NZ,JCN,ROWPTR,IRN,COLPTR,LELIM,ORDER,WT,LIW,IW,W,
     +           ICNTL,INFO,RINFO)
```

The double precision version

```
      CALL MC62AD(M,N,NZ,JCN,ROWPTR,IRN,COLPTR,LELIM,ORDER,WT,LIW,IW,W,
     +           ICNTL,INFO,RINFO)
```

M is an INTEGER variable that must be set by the user to the number of rows in the matrix **A**. This argument is not altered by the routine. **Restriction:** M ≥ 1.

N is an INTEGER variable that must be set by the user to the number of columns in the matrix **A**. This argument is not altered by the routine. **Restriction:** N ≥ 1.

NZ is an INTEGER variable that must be set by the user to be at least as large as the total number of entries in the matrix. This argument is not altered by the routine. **Restriction:** NZ ≥ 1.

JCN is an INTEGER array of length NZ. JCN is used to hold lists of the column indices of the entries in the matrix, with those in row 1 preceding those in row 2, and so on. The entries within a row may be in any order). If ICNTL(3) = 1, JCN is set by the routine. If ICNTL(3) = 0, JCN must be set by the user and is not altered by the routine. If indices outside the range [1, N] are detected, the computation terminates with an error message.

ROWPTR is an INTEGER array of length M+1. ROWPTR(IROW) holds the position in JCN of the first entry in row IROW (IROW = 1, 2, ..., M), and ROWPTR(M+1) holds the position after the last entry in the last row. If ICNTL(3) = 1, ROWPTR is set by the routine. If ICNTL(3) = 0, ROWPTR must be set by the user and is not altered by the routine.

IRN is an INTEGER array of length NZ. IRN is used to hold lists of the row indices of the entries in the matrix, with those in column 1 preceding those in column 2, and so on. The entries within a column may be in any order). If ICNTL(3) = 0, IRN is set by the routine. If ICNTL(3) = 1, IRN must be set by the user and is not altered by the routine. If indices outside the range [1, M] are detected, the computation terminates with an error message.

COLPTR is an INTEGER array of length N+1. COLPTR(JCOL) holds the position in IRN of the first entry in column JCOL (JCOL = 1, 2, ..., N), and COLPTR(N+1) holds the position after the last entry in the last column. If ICNTL(3) = 0, COLPTR is set by the routine. If ICNTL(3) = 1, COLPTR must be set by the user and is not altered by the routine.

LELIM is a LOGICAL array of length N. It must be set by the user if ICNTL(7) = 1. In this case, LELIM(I) must be set to .TRUE. if variable I is a candidate for elimination and to .FALSE. otherwise (I = 1, 2, ..., N). LELIM is unchanged on exit. If ICNTL(7) ≠ 1, LELIM(I) is set by MC62A/AD to .TRUE. (I = 1, 2, ..., N).

ORDER is an INTEGER array of length M. It must be set by the user if ICNTL(5) = 1 or 2. In this case, ORDER must hold positive global priority values for the rows (see Section 4). ORDER may be a permutation, in which case the row for which ORDER(IROW) = 1 is likely to be chosen first in the new row order and the row for which

ORDER(IROW) = M is likely to be chosen last. On exit, the new row order is ORDER(1), ORDER(2), ..., ORDER(M).

WT is a REAL (DOUBLE PRECISION in the D version) array of length 3. It must be set by the user if ICNTL(5) = 1 or 3. In this case, WT must be set by the user to hold the weights for the priority function (see Section 4). On exit, WT holds the weights used in the priority function. WT(1) = WT(2) = WT(3) = 0 indicates the new row order was obtained using the RMCD algorithm.

LIW is an INTEGER variable which defines the length of the work array IW. If ICNTL(4) = 2, LIW must be at least L1 = 3*N + 2*M. If ICNTL(4) = 0 or 1, the workspace needed is L1 + INFO(4), where L1 = 6*max(M, N) + 4 (7*max(M, N) +4 if ICNTL(5) = 2) and INFO(4) is the number of edges in the row graph. If LIW is too small, a value for LIW is returned in INFO(2). Provided LIW ≥ L1, the value in INFO(2) will be sufficient to successfully reorder the rows. This argument is not altered by the routine. **Restriction:** LIW ≥ L1.

IW is an INTEGER array of length LIW. This array is used by the subroutine as workspace.

W is a REAL (DOUBLE PRECISION in the D version) array of length N. This array is used by the subroutine as workspace.

ICNTL is an INTEGER array of length 10 that must be set by the user to hold control parameters. Default values are set by a call to MC62I/ID. This argument is not altered by the routine.

INFO is an INTEGER array of dimension 10 that need not be set on entry. INFO(1) is used as an error flag. On a successful exit, it is set to 0 and INFO(2) holds the amount of workspace used. If an error has been detected, INFO(1) is set to a negative value and INFO(2) is used to hold more information:

–1 – M ≤ 0 or N ≤ 0. INFO(2) holds min(M, N). Immediate return with remaining input parameters unchanged.

–2 – NZ too small. INFO(2) holds a copy of NZ. Immediate return with remaining input parameters unchanged.

–4 – Out-of-range indices detected in IRN or JCN. The number of such indices is given by INFO(2). Remaining input parameters unchanged.

–5 – Failure due to insufficient space allocated to the array IW. The user should increase LIW to at least INFO(2).

–6 – ICNTL(3) out of range. INFO(2) holds a copy of ICNTL(3). Immediate return with remaining input parameters unchanged.

–7 – ICNTL(4) out of range. INFO(2) holds a copy of ICNTL(4). Immediate return with remaining input parameters unchanged.

A positive value of INFO(1) is associated with a warning. Possible values are:

+1 – The reordering has failed to reduce one or more of the following: the maximum row frontsize, the maximum column frontsize, the mean row frontsize, the mean column frontsize, the maximum frontal matrix size, the mean frontal matrix size.

+2 – The row graph of **A** is not connected (ICNTL(4) = 0 or 1 only).

+3 – Both the above warnings apply.

On exit with INFO(1) ≥ 0, if ICNTL(4) = 0 or 1, the remaining components of INFO contain the following information:

INFO(3) holds the length of the pseudodiameter of the row graph of **A** (ICNTL(4) = 0 or 1 only). If the graph is not connected, INFO(3) holds the longest pseudodiameter. Note that if INFO(3) is equal to 0, the rows of **A** are not connected (for example, if **A** is diagonal).

INFO(4) holds the number of edges in the row graph of **A** (ICNTL(4) = 0 or 1 only).

INFO(5) holds the number of eliminations performed for the initial row order 1, 2, ..., M.

INFO(6) holds the number of eliminations performed for the new row order ORDER(1), ORDER(2), ..., ORDER(M).

INFO(7) to INFO(10) are currently not used.

RINFO is a REAL (DOUBLE PRECISION in the D version) array of dimension 30 that need not be set on entry. On successful exit, RINFO contains the following information:

RINFO(1) and RINFO(2) hold the maximum row and column frontsizes for the initial row order 1, 2, ..., M.

RINFO(3) and RINFO(4) hold the mean row and column frontsizes for the initial row order 1, 2, ..., M.

RINFO(5) holds the maximum frontal matrix size for the initial row order 1, 2, ..., M.

RINFO(6) holds the mean frontal matrix size for the initial row order 1, 2, ..., M.

RINFO(7) and RINFO(8) hold the maximum row and column frontsizes for the new row order ORDER(1), ORDER(2), ..., ORDER(M).

RINFO(9) and RINFO(10) hold the mean row and column frontsizes for the new row order ORDER(1), ORDER(2), ..., ORDER(M).

RINFO(11) holds the maximum frontal matrix size for the new row order ORDER(1), ORDER(2), ..., ORDER(M).

RINFO(12) holds the mean frontal matrix size for the new row order ORDER(1), ORDER(2), ..., ORDER(M).

RINFO(13) to RINFO(30) do not currently hold data of interest to the user.

2.1.3 To compute statistics for a given row order

The single precision version

```
CALL MC62B(M,N,NZ,JCN,ROWPTR,LELIM,ORDER,IW,ICNTLB,INFOB,RINFOB)
```

The double precision version

```
CALL MC62BD(M,N,NZ,JCN,ROWPTR,LELIM,ORDER,IW,ICNTLB,INFOB,RINFOB)
```

M is an INTEGER variable that must be set by the user to the number of rows in the matrix **A**. This argument is not altered by the routine. **Restriction:** $M \geq 1$.

N is an INTEGER variable that must be set by the user to the number of columns in the matrix **A**. This argument is not altered by the routine. **Restriction:** $N \geq 1$.

NZ is an INTEGER variable that must be set by the user to be at least as large as the total number of entries in the matrix. This argument is not altered by the routine.

JCN is an INTEGER array of length NZ. On entry, JCN must contain lists of the column indices of the entries in the matrix, with those in row 1 preceding those in row 2, and so on. The entries within a row may be in any order. This argument is not altered by the routine.

ROWPTR is an INTEGER array of length M+1. On entry, ROWPTR(IROW) must contain the position in JCN of the first entry in row IROW (IROW = 1, 2, ..., M), and ROWPTR(M+1) must be set to the position after the last entry in the last row. This argument is not altered by the routine.

LELIM is a LOGICAL array of length N. It must be set by the user if ICNTLB(3) = 1. In this case, LELIM(I) must be set to .TRUE. if variable I is a candidate for elimination and to .FALSE. otherwise (I = 1, 2, ..., N). LELIM is unchanged on exit.

ORDER is an INTEGER array of length M. On entry, ORDER must be set by the user to hold the row order ORDER(1), ORDER(2), ..., ORDER(M) for which statistics are required. This argument is not altered by the routine.

IW is an INTEGER array of length 2*N. This array is used by the subroutine as workspace.

ICNTLB is an INTEGER array of length 3. On entry, ICNTLB(1) and must be set by the user to the unit number for printing computed statistics. Printing is suppressed if ICNTLB(1) < 0. ICNTLB(2) must be set to the minimum number of variables to be eliminated at a single stage (zero and negative values are treated as 1). ICNTLB(3) must be set to 1 if the user wishes to specify which variables may be eliminated using LELIM. Otherwise, it is assumed that each of the variables is a candidate for elimination. This argument is not altered by the routine.

INFOB is an INTEGER variable. On exit, INFOB holds *p*, the number of eliminations performed.

RINFOB is a REAL (DOUBLE PRECISION in the D version) array of length 15. On exit, RINFOB contains the following information:

RINFOB(1) and RINFOB(2) hold the maximum row and column frontsizes for the given row order ORDER(1), ORDER(2), ..., ORDER(M).

RINFOB(3) and RINFOB(4) hold the mean row and column frontsizes for the given row order ORDER(1), ORDER(2), ..., ORDER(M).

RINFOB(5) holds the maximum frontal matrix size for the given row order ORDER(1), ORDER(2), ..., ORDER(M).

RINFOB(6) holds the mean frontal matrix size for the given row order ORDER(1), ORDER(2), ..., ORDER(M).

RINFOB(7) to RINFOB(15) do not currently hold data of interest to the user.

2.1.4 To compute the row graph of a matrix

The single precision version

```
  CALL MC62C(M,N,NZ,JCN,ROWPTR,IRN,COLPTR,LINDEX,INDEX,IP,IW,ICNTLC,INFOC)
```

The double precision version

```
  CALL MC62CD(M,N,NZ,JCN,ROWPTR,IRN,COLPTR,LINDEX,INDEX,IP,IW,ICNTLC,INFOC)
```

M is an INTEGER variable that must be set by the user to the number of rows in the matrix **A**.

N is an INTEGER variable that must be set by the user to the number of columns in the matrix **A**.

NZ is an INTEGER variable that must be set by the user to be at least as large as the total number of entries in the matrix. This argument is not altered by the routine.

JCN is an INTEGER array of length NZ. On entry, JCN must contain lists of the column indices of the entries in the matrix, with those in row 1 preceding those in row 2, and so on. The entries within a row may be in any order. This argument is not altered by the routine.

ROWPTR is an INTEGER array of length M+1. On entry, ROWPTR(IROW) must contain the position in JCN of the first entry in row IROW (IROW = 1, 2, ..., M), and ROWPTR(M+1) must be set to the position after the last entry in the last row. This argument is not altered by the routine.

IRN is an INTEGER array of length NZ. On entry, IRN must contain lists of the row indices of the entries in the matrix, with those in column 1 preceding those in column 2, and so on. The entries within a column may be in any order. This argument is not altered by the routine.

COLPTR is an INTEGER array of length N+1. On entry, COLPTR(JCOL) must contain the position in IRN of the first entry in column JCOL (JCOL =1, 2, ..., N), and COLPTR(N+1) must be set to the position after the last entry in the last column. This argument is not altered by the routine.

LINDEX is an INTEGER variable which defines the length of the array INDEX. If LINDEX is too small, a value that will suffice is returned in INFOC(2). This argument is not altered by the routine.

INDEX is an INTEGER array of length LINDEX that need not be set on entry. On exit, INDEX contains adjacency lists for the row graph, with the list of rows that are adjacent to row 1 preceding the list for row 2, and so on.

`IP` is an `INTEGER` array of length `M+1` that need not be set on entry. On exit, `IP(IROW)` points to the position in `INDEX` of the first entry in the adjacency list for row `IROW` (`IROW` = `1`, `2`, ..., `M`), and `IP(M+1)` is set to the position after the last entry in `INDEX.` This argument is not altered by the routine.

`IW` is an `INTEGER` array of length `M` that is used by the routine as workspace.

`ICNTLC` is an `INTEGER` array of length `2`. On entry, `ICNTLC(1)` must be set by the user to the unit number for error messages and `ICNTLC(2)` to the unit number for warnings. Printing of error messages (respectively, warnings) is suppressed if `ICNTLC(1)` < `0` (respectively, `ICNTLC(2)` < `0`). This argument is not altered by the routine.

`INFOC` is an `INTEGER` array of length `2`. `INFOC(1)` is used as an error flag. On a successful exit, it is set to `0`. A warning is issued and `INFOC(1)` set to `1` (or `2`) if the matrix is found to have one or more null rows (or columns). `INFOC(1)` = `3` if the matrix has both null rows and null columns. `INFOC(1)` is set to `-1` if `LINDEX` is too small. In this case, `INFOC(2)` is set to a sufficient value for `LINDEX` (which is equal to the number of edges in the row graph).

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly: Subroutines internal to the package are `MC62D/DD`, `MC62E/ED`, and `MC62F/FD`. In addition, `MC38` and `MC60` are called.

Input/output: Error messages on unit `ICNTL(1)` and printing of warnings and statistics on unit `ICNTL(2)`. Printing is suppressed by choosing negative unit numbers.

Restrictions:

`MC62A/AD` : `N` ≥ `1`, `M` ≥ `1`, `NZ` ≥ `1`, `ICNTL(3)` = `0` or `1`, `ICNTL(4)` = `0`, `1`, or `2`, `LIW` ≥ `L1`.

4 METHOD

Having performed initial checks on the user's data, `MC62A/AD` calls `MC38A/AD` to construct a representation of the matrix by rows from a representation by columns (or a representation by columns from a representation by rows). `MC62B/BD` is then called to compute statistics for the initial row order 1,2,...,*m*.

MSRO algorithm

`MC62C/CD` is called to generate the row graph. For each component of the row graph, `MC60L/LD` is used to construct the level structure rooted at the row with the lowest global priority (`ICNTL(5)` = `1` or `2`) or `MC60H/HD` is used to find a pseudoperipheral pair of rows. The row graph is relabelled using the priority function defined for row *i* by

$$\texttt{WT(1)}*rcgain(i) + \texttt{WT(2)}*v*glob(i) - \texttt{WT(3)}*nold(i). \tag{1}$$

Here *rcgain*(*i*) is the sum of the increases in the row and column frontsizes resulting from ordering row *i* next, v is a normalizing factor, *glob*(*i*) is the (positive) global priority value of row *i* (generated automatically or provided in `ORDER` if `ICNTL(5)` = `1` or `2`), and *nold*(*i*) is the number of variables in row *i* that are candidates for elimination and have already been brought into the front. At each stage, from a list of eligible rows, the row that minimizes the priority function is chosen to be next in the new row order. If `ICNTL(5)` = `1` or `3`, the weights supplied by the user are used. Otherwise, the code will use the triplets of weights (2,1,0.2) and (1,32,0.2) (or (1,2,0.2) and (1,32,0.2) if `ICNTL(5)` = `2`) and will return results for whichever triplet yields the best row order (in terms of the mean frontal matrix size). For full details the user is referred to Scott (1998, 1999). `MC62B/BD` computes statistics for the new row order.

RMCD algorithm

The RMCD algorithm stores the degree of each column and, at each stage, chooses the column of minimum degree and orders next all the rows with an entry in the chosen column. The column degrees are then updated before the next column is selected. MC62B/BD computes statistics for the new row order.

MC62B/BD is also used to compute statistics for the rows taken in reverse order. Reversing the order may reduce the maximum and mean row frontsizes and the maximum and mean frontal matrix size but has no effect on the maximum and mean column frontsizes. If the mean frontal matrix size is smaller for the reverse order, the reverse order is returned to the user as the new row order.

The RMCD algorithm has the advantage of being considerably faster than the MSRO algorithm and requires less storage. However, the MSRO algorithm has been found to generally produce much better orderings and is much more consistent in the quality of the orderings produced. The MSRO does not work well if the row graph has a short pseudodiameter. The default is to compute the MSRO ordering but options exist (through the control parameter ICNTL(4)) for computing orderings using both the MSRO and the RMCD algorithms or only the RMCD algorithm.

References

Scott, J. A. (1998). A new row ordering strategy for frontal solvers. Technical Report RAL-TR-1998-056, Rutherford Appleton Laboratory. Also *Numerical Linear Algebra and Applications, 6 (1999,) 1-23.*

Scott, J. A. (1999). Row ordering for frontal solvers in chemical process engineering. Technical Report RAL-TR-1999-035, Rutherford Appleton Laboratory. Also *Computers in Chemical Engineering, 24 (2000), 1865-1880.*

Scott, J. A. (2000). Two-stage ordering for unsymmetric parallel row-by-row frontal solvers. Technical Report RAL-TR-2000-030, Rutherford Appleton Laboratory. Also *Computers in Chemical Engineering, 25 (2001), 323-332.*

5 EXAMPLE OF USE

The following program provides an example of the use of MC62. We wish to reorder the rows of the matrix

$$\mathbf{A} = \begin{pmatrix} \times & & & \times & & \times \\ & \times & & & \times & \\ & & \times & & & \times \\ & \times & & & & \\ & & & & \times & \times \\ \times & & \times & \times & & \times \end{pmatrix}.$$

```
C  Example to illustrate the use of MC62AD.
C     .. Parameters ..
      INTEGER MXN,MXNZ,LIW
      PARAMETER (MXN=20,MXNZ=30,LIW=5000)
C     ..
C     .. Local Scalars ..
      INTEGER I,N,NZ
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION RINFO(30),W(MXN),WT(3)
      INTEGER ROWPTR(MXN+1),JCN(MXNZ),COLPTR(MXN+1),IRN(MXNZ),
     +        ICNTL(10),INFO(10),IW(LIW),ORDER(MXN)
      LOGICAL LELIM(MXN)
C     ..
C     .. External Subroutines ..
      EXTERNAL MC62AD,MC62ID
C     ..
C   Read in the column indices and row pointers
      READ (5,*) N,NZ
      READ (5,*) (ROWPTR(I),I=1,N+1)
      READ (5,*) (JCN(I),I=1,NZ)
      CALL MC62ID(ICNTL)
C Use default settings for all parameters
      CALL MC62AD(N,N,NZ,JCN,ROWPTR,IRN,COLPTR,LELIM,ORDER,WT,LIW,
     +            IW,W,ICNTL,INFO,RINFO)

      IF (INFO(1).LT.0) THEN
         WRITE (6,*) ' Unexpected error return from MC62AD'
          STOP
      END IF

      WRITE (6,'(//A,6I4)') ' New row order :', (ORDER(I),I=1,N)

      STOP
      END
```

The input data used for this problem is:

```
 6   14
 1  4  6  8  9  11  15
 1  4  6  2  5  3  6  2  5  6  1  4  6  3
```

This produces the following output:

```
 For initial row order:

 Max row/col front size             =             4           6
 Mean row/col front size            =  3.00000E+00 3.50000E+00
 Max frontal matrix size            =  2.40000E+01
 Mean frontal matrix size           =  1.23333E+01

 Number of edges in the row graph   =            16
 Length of pseudodiameter           =             4

 For new row order:

 Max row/col front size             =             2           4
 Mean row/col front size            =  1.16667E+00 2.33333E+00
 Max frontal matrix size            =  8.00000E+00
 Mean frontal matrix size           =  3.00000E+00



 New row order :   1   6   3   5   2   4
```

The reordered matrix is

$$\begin{pmatrix} \times & & & \times & & \times \\ \times & & \times & \times & & \times \\ & & \times & & & \times \\ & & & & \times & \times \\ & \times & & & \times & \\ & \times & & & & \end{pmatrix}.$$