1 SUMMARY

Let **A** be an $n \times n$ sparse matrix with a symmetric sparsity pattern. Given the sparsity pattern of **A**, this subroutine uses a variant of Sloan's method to calculate a symmetric permutation that aims to **reduce the profile and wavefront** of **A**. Alternatively, the Reverse Cuthill-McKee (RCM) method may be requested to reduce the bandwidth, or the user may request an ordering for the rows of **A** that is efficient when used with a row-by-row frontal solver (for example, equation entry to MA42).

MC61 provides the user with a straightforward interface to the MC60 package when detailed control of the steps in constructing a symmetric permutation or row ordering is not required.

ATTRIBUTES — Version: 1.1.0 (26 October 2012). **Remark:** MC61 supersedes MC40. **Types:** Real (single, double). **Calls:** MC60. **Original date:** January 1998. **Origin:** J. K. Reid and J. A. Scott (Rutherford Appleton Laboratory).

2 HOW TO USE THE PACKAGE

2.1 Argument lists

There are two entries:

MC61I/ID must be called to initialize the parameters that control the execution of the package.

MC61A/AD either chooses a variable permutation that aims to reduce the profile and wavefront or bandwidth of the matrix or constructs an ordering for the rows that is efficient when used with a row-by-row frontal solver, such as MA42 (equation entry) or MA43.

2.1.1 The initialization subroutine

To initialize control parameters, the user should make a call of the following form:
The single precision version

```
CALL MC61I(ICNTL,CNTL)
```

The double precision version

```
CALL MC61ID(ICNTL,CNTL)
```

ICNTL is an INTEGER array of length 10 that need not be set by the user. This array is used to hold control parameters. On exit, 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 MC61I/ID. The default values are as follows:

ICNTL(1) is the stream number for error messages. It has the default value 6. Printing of error messages is suppressed if ICNTL(1) ≤ 0.

ICNTL(2) is the stream number for warning messages. It has the default value 6. Printing of warning messages is suppressed if ICNTL(2) ≤ 0.

ICNTL(3) controls the action taken if duplicate or out-of-range indices are detected. If ICNTL(3) $= 0$ and such indices are detected, the computation terminates with IRN and ICPTR unchanged. If ICNTL(3) $= 1$, a warning is issued and the computation continues with such indices ignored. The default value is 0.

ICNTL(4) controls whether supervariables are used (a supervariable is a set of variables that correspond to a set of identical columns and the condensed matrix is the representation of **A** by supervariables). If ICNTL(4) = 0, supervariables are used and are not used if ICNTL(4) = 1. If the problem has significantly fewer supervariables than variables, using supervariables will substantially reduce the execution time and amount of integer workspace used. The default value is 0.

ICNTL(5) indicates whether the user wishes to supply a global priority function. If ICNTL(5) = 0, no priority function is supplied; if ICNTL(5) = 1, a priority function is supplied in PERM. The default value is 0.

ICNTL(6) controls whether the user wishes to supply the weights for the priority function (JOB = 1 or 3) (see Section 4). If ICNTL(6) = 0, the code will use the pairs of weights (2,1) and (16,1) and will return results for whichever pair yields the best permutation; if ICNTL(6) = 1, the pairs of weights (1,2) and (16,1) will be used and results for whichever pair yields the best permutation will be returned; if ICNTL(6) = 2, the weights in CNTL(1) and CNTL(2) will be used. The weights that are used to give the final permutation are returned in RINFO(9) and RINFO(10). The default value is 0.

ICNTL(7) to ICNTL(10) are given the default value 0. They are currently not used but may be used in a later release of the code.

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 5 that need not be set by the user. This array is used to hold control parameters. On exit, CNTL contains default values. If the user wishes to use values other than the defaults, the corresponding entries in CNTL should be reset after the call to MC61I/ID. The default values are as follows:

CNTL(1) and CNTL(2) hold the weights W_1 and W_2 that are used in the priority function in the case ICNTL(6) = 2. The default values are 2.0 and 1.0, respectively.

CNTL(3) to CNTL(5) are given the default value zero. They are currently not used but may be used in a later release of the code.

2.1.2 To find a variable permutation or a row ordering

The single precision version

```
CALL MC61A(JOB,N,LIRN,IRN,ICPTR,PERM,LIW,IW,W,ICNTL,CNTL,INFO,RINFO)
```

The double precision version

```
CALL MC61AD(JOB,N,LIRN,IRN,ICPTR,PERM,LIW,IW,W,ICNTL,CNTL,INFO,RINFO)
```

JOB is an INTEGER variable that must be set by the user to 1 if a variable permutation to reduce the profile and wavefront of the matrix is required, to 2 if a variable permutation to reduce the bandwidth is required, and to 3 if a row ordering for a row-by-row frontal solver is required. This argument is not altered. **Restriction:** JOB = 1, 2, or 3.

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

LIRN is an INTEGER variable that must be set by the user to the length of the array IRN, which must be large enough to hold the sparsity pattern of the whole matrix. 2*(ICPTR(N+1)-1) is always sufficient. This argument is not altered.

IRN is an INTEGER array of length LIRN whose leading part must be set by the user to hold the row indices of the entries in the lower triangle of the matrix, including those on the diagonal. The entries of each column must be contiguous. The entries of column J must precede those of column J+1 (J=1,...,N-1), and there must be no wasted space between the columns. Row indices within a column may be in any order. On successful exit, IRN holds the row entries of the condensed matrix (upper and lower triangular parts), using the same format.

ICPTR is an INTEGER array of length N+1 that must be set by the user so that ICPTR(J) points to the position in the array IRN of the first entry in column J (J=1,2,...,N), and ICPTR(N+1)-1 must be the position of the last entry. On successful exit, ICPTR holds corresponding data for the condensed matrix.

PERM is an INTEGER array of length N. This array need be set on entry only if JOB $\neq$ 2 and ICNTL(5) = 1 (the default is ICNTL(5) = 0). In this case, PERM must be set by the user to hold positive global priority values for the variables (see Section 4); this may be a permutation, in which case the variable for which PERM(I) = 1 is likely have a low index in the new ordering and the variable for which PERM(I) = N is likely to appear towards then end of the new ordering. In all cases, on exit, the new ordering is contained in PERM. If a variable permutation is requested (JOB = 1 or 2), the new index for variable I is given by PERM(I) (I = 1,2,...,N). If a row order is requested (JOB = 3), the order in which the rows should be presented to the row-by-row frontal solver is PERM(1),PERM(2),...,PERM(N).

LIW is an INTEGER variable that must be set by the user to the length of the array IW. LIW must be at least 2+5*N and 2+8*N is always sufficient. If JOB $\neq$ 2 and ICNTL(5) = 1 (the default is 0), a sufficient value is 2+7*N and if JOB $\neq$ 2, ICNTL(5) = 1 and ICNTL(6) = 2 (the default is 0), a sufficient value is 2+6*N. Note that if supervariables are used (ICNTL(4) = 0, which is the default), 2+2*N+max(2*N, 6*nsup) is always sufficient, where nsup is the number of supervariables (see INFO(2)). This argument is not altered. **Restriction:** LIW $\geq$ 2+5*N.

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

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

ICNTL is an INTEGER array of length 10 that must be set by the user to hold control parameters. This argument is not altered.

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 5 that must be set by the user to hold control parameters. This argument is not altered.

INFO is an INTEGER array of length 10 that need not be set by the user. On exit, INFO returns the following information:

INFO(1) is used as an error flag. If a call to MC61A/AD is successful, on exit INFO(1) has value 0. A negative value for INFO(1) is associated with a fatal error. If ICNTL(1) $>$ 0, a self-explanatory message is, in each case, output on unit ICNTL(1). The negative values for INFO(1) are:

- -1 JOB is not equal to 1, 2, or 3, or N $<$ 1, or LIRN $<$ ICPTR(N+1)-1. Immediate return with input parameters unchanged.
- -2 LIRN is too small. INFO(6) is set to the minimum value that will suffice for LIRN. If ICNTL(3) = 1, any out-of-range or duplicated variable indices will have been excluded from IRN and ICPTR. Otherwise, the input parameters are unchanged.
- -3 LIW is too small. INFO(3) is set to a value that will suffice for LIW.
- -4 ICNTL(3) = 0 and one or more variable indices either lies outside the lower triangle of the matrix or is duplicated. Further information is contained in INFO(4) and INFO(5).

A positive value of INFO(1) is associated with a warning message. If ICNTL(2) $>$ 0, a self-explanatory message is, in each case, output on unit ICNTL(2) and further information is contained in INFO(3) to INFO(6). Note that if a fatal error is detected during a call to MC61A/AD, the information contained in INFO and RINFO will be incomplete.

INFO(2) holds, on successful exit, the total number of supervariables in the problem. If supervariables are not used (ICNTL(4) = 1), INFO(2) is set to N.

INFO(3) holds the amount of workspace used by the routine. If the user has provided insufficient workspace (INFO(1) = -3), INFO(3) is set to a value that will suffice for LIW.

INFO(4) holds the number of out-of-range indices in IRN.

INFO(5) holds the number of duplicate indices in IRN.

INFO(6) holds the minimum value that will suffice for LIRN.

INFO(7) holds the number of non-trivial components (two or more nodes) in the graph of the matrix.

INFO(8) to INFO(10) are currently not used but may be used in a later release of the code.

RINFO is a REAL (DOUBLE PRECISION in the D version) array of length 15 that need not be set by the user. If JOB $=1$ or 2, on successful exit RINFO returns the following information:

RINFO(1) holds the profile of the matrix **A**.

RINFO(2) holds the maximum wavefront of the matrix **A**.

RINFO(3) holds the semibandwidth of the matrix **A**.

RINFO(4) holds the root mean squared wavefront of the matrix **A**.

RINFO(5) holds the profile of the permuted matrix.

RINFO(6) holds the maximum wavefront of the permuted matrix.

RINFO(7) holds the semibandwidth of the permuted matrix.

RINFO(8) holds the root mean squared wavefront of the permuted matrix.

If JOB $=3$, on successful exit RINFO returns the following information:

RINFO(1) and RINFO(2) hold the maximum row and column front sizes for the original row order $1,2,\ldots,$N.

RINFO(3) holds the root-mean-square row front size for the original row order $1,2,\ldots,$N.

RINFO(4) holds the mean frontal matrix sizes for the original row order $1,2,\ldots,$N.

RINFO(5) and RINFO(6) hold the maximum row and column front sizes for the new row order PERM(1),PERM(2),...,PERM(N).

RINFO(7) holds the root-mean-square row front size for the new row order PERM(1),PERM(2),...,PERM(N).

RINFO(8) holds the mean frontal matrix sizes for the new row order PERM(1),PERM(2),...,PERM(N).

In addition, if JOB $=1$ or 3, on successful exit, RINFO(9) and RINFO(10) hold the pair of weights that are used to give the final permutation (see ICNTL(6)). RINFO(11) to RINFO(15) are currently not used but may be used in a later release of the code.

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly: MC60A/AD, MC60B/BD, MC60C/CD, MC60D/DD, MC60E/ED, MC60F/FD, MC60G/GD.

Input/output: In the event of errors, diagnostic messages are printed. Stream ICNTL(1) is used for error messages and stream ICNTL(2) for warnings.

Restrictions: JOB $=1$, 2, or 3, N ≥ 1, LIW $\geq 2+5*$N.

4 METHOD

MC61 first calls MC60A to construct the pattern of the whole matrix **A** and to perform extensive checks on the data. If JOB = 1 or 2, MC60F computes the profile, the maximum wavefront, the semibandwidth, and the root mean squared wavefront of **A**, and if JOB = 3, MC60G computes the maximum row and column front sizes and the root mean squared row and column front sizes for the original row order.

If supervariables are being used (ICNTL(4) $=0$), MC61 calls MC60B to look for sets of columns with identical patterns. A permutation is constructed that places the variables of each supervariable together. The pattern of **A** is replaced by that of the permuted matrix represented as supervariables (that is, by its condensed equivalent).

MC61 then calls MC60C to construct a supervariable (ICNTL(4) $=0$) or variable (ICNTL(4) $=1$) permutation that aims to reduce either the profile and wavefront of the matrix or the bandwidth. If JOB = 1 or 3, the priority function for supervariable s is

$$W_1 * deg(s) + W_2 * \nu * glob(s),$$

where W_1 and W_2 are weights, $deg(s)$ is the number of supervariables that will enter the front if supervariable s is chosen next, ν is a normalizing factor, and $glob(s)$ is the positive global priority function (generated automatically or provided in PERM). If JOB = 2, the Reverse Cuthill McKee algorithm is used to reduce the bandwidth.

If JOB = 1 or 2, MC60F computes the profile, the maximum wavefront, the semibandwidth, and the root mean squared wavefront for the permuted matrix and, if supervariables are being used, MC60D constructs the permutation for the variables that corresponds to the permutation for the supervariables. If JOB = 3, MC60E uses the supervariable permutation to construct an ordering for the rows, as required by a row-by-row frontal solver, and MC60G computes the maximum row and column front sizes and the maximum and mean frontal matrix sizes for the new row order.

Further details of the method are given in the MC60 specification document and in [1].

References:

[1] J.K Reid and J.A. Scott. (1998). Ordering symmetric sparse matrices for small profile and wavefront. Technical Report RAL-TR-98-016, Rutherford Appleton Laboratory.

5 EXAMPLE OF USE

The following program provides an example of the use of MC61. We wish to reduce the profile of a matrix with the following sparsity pattern:

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

The input data will be the lower triangle of **A** in column-wise format. Using the program:

```
C Code to illustrate use of MC61

      INTEGER LIRN,MAXN
      PARAMETER (LIRN=20, MAXN=5)
      INTEGER I,JOB,LIW,N,NZ
      INTEGER IRN(LIRN),ICPTR(MAXN+1),PERM(MAXN),IW(8*MAXN+2),
     +        ICNTL(10),INFO(10)
      REAL CNTL(5),RINFO(15),W(MAXN)

C Read in data
      READ (5,*) N,NZ
      READ (5,*) (IRN(I),I = 1,NZ)
      READ (5,*) (ICPTR(I),I = 1,N+1)

C Set control parameters
      CALL MC61I(ICNTL,CNTL)
C Prepare to call MC61A
      JOB = 1
      LIW = 8*N + 2
      CALL MC61A(JOB,N,LIRN,IRN,ICPTR,PERM,LIW,IW,W,ICNTL,CNTL,
     +           INFO,RINFO)

C Check for an error return
      IF (INFO(1).LT.0) THEN
         WRITE (6,*) ' Unexpected error return!'
         STOP
      END IF

C Write out the profile
      WRITE (6,'(/A,F6.0)') ' The profile initially  is ', RINFO(1)
      WRITE (6,'(/A,F6.0)') ' The profile after MC61 is ', RINFO(5)
      END
```

on the data

```
    5  9
    1  2  3  4  5  2  3  4  5
    1  6  7  8  9  10
```

produces the output:

```
 The profile initially  is   15.

 The profile after MC61 is    9.
```