1 SUMMARY

Given the sparsity pattern of an $n \times n$ symmetric matrix **A** and a symmetric permutation that reduces the profile of **A**, this routine computes a new symmetric permutation with a smaller **profile**. The exchange algorithms of Hager are used to refine the given permutation.

Any zeros on the diagonal of **A** are regarded as nonzero. If m_i is the column index of the first nonzero in row i ($m_i \leq i$), the length of row i is $i - m_i + 1$ and the profile of **A** is the sum of the lengths of the rows.

`MC61` (or `MC60`) may be used to obtain an initial symmetric permutation.

ATTRIBUTES — Version: 1.0.0. (12 July 2004) **Types:** Real (single, double). **Calls:** `KB06`. **Original date:** February 2002. **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:

(a) The subroutine `MC67I/ID` must be called to initialize the parameters that control the execution of the package.

(b) `MC67A/AD` takes a given symmetric permutation of **A** and uses Hager's exchange algorithms to compute a permutation with a smaller profile.

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 MC67I(ICNTL)
```

The double precision version

```
CALL MC67ID(ICNTL)
```

`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 `MC67I/ID`. The default values are as follows:

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

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

`ICNTL(3)` controls the checking of the user's data. If `ICNTL(3)` $= 0$, no checks are made on the arrays `IRN` and `ICPTR`. Otherwise, checks are made. If the matrix is found not to be symmetric, the computation terminates. The computation also terminates if `ICNTL(3)` < 0 and duplicated or out-of-range indices are detected. If `ICNTL(3)` > 0 and such indices are found, they are removed, a warning is issued and (provided the matrix is symmetric with the duplicated and out-of-range indices removed) the computation continues. The default value is `0`. Note that data checking adds an overhead and is not necessary if the user has called `MC61` to generate the input permutation (`MC61` performs its own checking of the data).

`ICNTL(4)` controls the maximum number of times the exchange algorithms are applied. The default value is `5`.

ICNTL(4) should be set to a large positive value if the user wishes the exchange algorithms to be applied repeatedly until no further reductions in the profile are achieved or until the previous reduction is small (see ICNTL(7)). If ICNTL(4) is less than or equal to zero, the value 1 is used.

ICNTL(5) controls the use of the Hager exchange algorithms. Possible values are:

- 0 Down/up exchanges.
- 1 Up/down exchanges.
- 2 Down exchanges only.
- 3 Up exchanges only.

The default value is 0. All values except 1, 2, and 3 are treated as 0.

ICNTL(6) controls whether the profile for the user-supplied permutation is computed. If ICNTL(6) = 0, it is not computed; for all other values, it is computed. The default value is 0.

ICNTL(7) controls termination. If an application of the exchange algorithm yields a profile reduction that is not greater than ICNTL(7)% of the first reduction, no further applications are made. The default value is 0.

ICNTL(8) 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.

2.1.2 The reordering subroutine

The single precision version

```
CALL MC67A(N,LIRN,IRN,ICPTR,PERM,LIW,IW,ICNTL,INFO,RINFO)
```

The double precision version

```
CALL MC67AD(N,LIRN,IRN,ICPTR,PERM,LIW,IW,ICNTL,INFO,RINFO)
```

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 ≥ 1.

LIRN is an INTEGER variable that must be set by the user to the length of the array IRN. LIRN must be at least as large as the number of entries in **A**. This argument is not altered. **Restriction:** LIRN ≥ ICPTR(N+1)-1.

IRN is an INTEGER array of length LIRN which must be set by the user to hold the row indices of the entries in the whole 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, 2, ..., N-1), and there must be no wasted space between the columns. Row indices within a column may be in any order. This argument is not altered unless ICNTL(3) > 0 and duplicates or out-of-range entries have been found and removed.

ICPTR is an INTEGER array of length N+1 that must be set by the user so that ICPTR(J) holds the position in the array IRN of the first entry in column J (J = 1, 2, ..., N), and ICPTR(N+1)–1 holds the position of the last entry. This argument is not altered unless ICNTL(3) > 0 and duplicates or out-of-range entries have been found and removed.

PERM is an INTEGER array of length N. It must be set by the user to hold the initial symmetric permutation for **A** that MC67 is to be used to improve; that is, PERM(I) must hold the position of column I, I = 1, 2, ..., N, in the initial permutation. This could be the output permutation from MC61 or, if no initial permutation is available, the identity PERM(I) = I may be used (although this is only advisable if the matrix is already well ordered for a small profile). On exit, PERM(I) holds the position of column I in the new permutation. Note that if the exchange algorithms do not reduce the profile, PERM is unchanged on exit.

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+7*N. This argument is not altered. **Restriction:** LIW ≥ 2+7*N.

IW is an INTEGER array of length LIW that is used 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.

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 MC67A/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 – N is less than 1.

–2 – LIRN is too small. LIRN must be at least ICPTR(N+1)–1.

–3 – LIW is too small.

–4 – PERM does not hold a permutation of 1, 2, ..., N.

–5 – The sparsity pattern of **A** is not symmetric (the number of entries in at least one row is not equal to the number of entries in the corresponding column).

–6 – Duplicate and/or out-of-range entries (ICNTL(3) < 0 only). Further information is contained in INFO(3) and INFO(4).

A positive value of INFO(1) is associated with a warning message. If ICNTL(2) > 0, a self-explanatory message is output on unit ICNTL(2).

1 – Duplicate and/or out-of-range entries found and removed (ICNTL(3) > 0 only). Further information is contained in INFO(3) and INFO(4).

INFO(2) holds, on successful exit, the number of times the exchange algorithms were applied. This is at most ICNTL(4) but could be less (the computation terminates if no further reduction in the profile can be achieved using the exchange algorithms or if the previous reduction is small, see ICNTL(7)).

INFO(3) and INFO(4) hold the number of out-of-range and duplicate indices in IRN, respectively (ICNTL(3) ≠ 0 only).

INFO(5) to INFO(10) are not 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 10 that need not be set by the user. On successful exit RINFO returns the following information:

RINFO(1) holds the reduction in the profile of the matrix **A** (this reduction is with respect to the profile for the initial permutation held in PERM).

RINFO(2) and RINFO(3) hold the reductions in the profile of the matrix **A** achieved by the down and up exchange algorithms, respectively. Note that RINFO(1) = RINFO(2) + RINFO(3).

RINFO(4) and RINFO(5) hold the profiles for the initial and final permutations, respectively (ICNTL(6) ≠ 0 only). Note that RINFO(5) = RINFO(4) – RINFO(1).

RINFO(6) to RINFO(10) 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: KB06AI. Subroutines internal to the package are MC67B/BD, MC67C/CD, MC67D/DD.

Input/output: In the event of errors, diagnostic messages are printed. Unit `ICNTL(1)` is used for error messages and unit `ICNTL(2)` for warnings. Printing is suppressed by setting these unit numbers to be negative.

Restrictions: `N`$\geq$`1`, `LIRN`$\geq$`ICPTR(N+1)-1`, `LIW`$\geq$`2+7*N`.

4 METHOD

Hager's (2002) down exchange algorithm involves cyclic permutations that correspond to the successive exchange of rows (k, k+1), (k+1, k+2), ..., (l–1, l) of the permuted matrix and interchanging the corresponding columns. For a given k, the value of l that most reduces the profile is found. A pass over the matrix with k taking the values n–1, n–2, ..., 1 is performed; l is calculated for each k and, if this gives a profile reduction, the corresponding permutation is applied. `MC67B/BD` performs one or more such passes, stopping if there is no improvement in the profile or if the improvement is small (`ICNTL(7)>0`).

Hager's (2002) up exchange is similar, with the direction reversed. For a given k, rows and columns (k, k–1), (k–1, k–2), ..., (l+1, l), are exchanged, again with the value of l that most reduces the profile. A pass over the matrix is performed with k taking the values 2, 3, ..., n. `MC67C/CD` performs one or more such passes, again stopping if there is no improvement in the profile or if the improvement is small (`ICNTL(7)>0`).

`MC67A/AD` checks for errors under the control of `ICNTL(3)` and then calls `MC67B/BD` and/or `MC67C/CD`. Hager's recommendation is to perform a sequence of pairs of down then up passes until a given number has been performed or a pair yields no further improvement. The given number is under the user's control in `ICNTL(4)`. We found that it was rare in test cases for there to be much improvement after 5 passes, so we have set the default value to 5. We also offer facilities to perform up then down pairs or passes of one kind only.

Further details of the method are given by Reid and Scott (2001).

References

Hager W.W. (2002). Minimizing the profile of a symmetric matrix. SIAM Journal on Scientific Computing, **23**, 1799-1816.

Reid, J.K. and J.A. Scott. (2001). Implementing Hager's exchange methods for matrix profile reduction. Technical Report RAL-TR-2001-039, Rutherford Appleton Laboratory.

5 EXAMPLE OF USE

The following program provides an example of the use of `MC67`. We wish to reduce the profile of a matrix with the following sparsity pattern.

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

The initial permutation is `PERM(I)` = `N-I+1`, `I` = `1`, `2`, ..., `5`. Using the program:

```
C Code to illustrate use of MC67

      INTEGER LIRN,MAXN
      PARAMETER (LIRN=20,MAXN=10)
      INTEGER I,J,LIW,N,NZ
      INTEGER IRN(LIRN),ICPTR(MAXN+1),PERM(MAXN),IW(7*MAXN+2),
     +        ICNTL(10),INFO(10)
      DOUBLE PRECISION A(1),RINFO(10)

      EXTERNAL MC34AD,MC67ID,MC67AD
```

```
C Read in data for lower triangular part
      READ (5,*) N,NZ
      READ (5,*) (IRN(I),I = 1,NZ)
      READ (5,*) (ICPTR(I),I = 1,N+1)

C Generate pattern for whole matrix
      CALL MC34AD(N,IRN,ICPTR,.FALSE.,A,IW)
      WRITE (6,FMT='(/A)') ' Input matrix:'
      DO 20 I = 1,N
         WRITE (6,FMT='(A,I3,A,10I3)') ' Column',I,':',
     +     (IRN(J),J=ICPTR(I),ICPTR(I+1)-1)
   20 CONTINUE

C Initial ordering
      DO 30 I = 1,N
         PERM(I) = N-I+1
  30  CONTINUE

C Use MC67 to improve the ordering
      CALL MC67ID(ICNTL)
      ICNTL(6) = 1
      LIW = 7*N + 2

      CALL MC67AD(N,LIRN,IRN,ICPTR,PERM,LIW,IW,ICNTL,INFO,RINFO)

C Check for an error return
      IF (INFO(1).LT.0) THEN
         WRITE (6,*) ' Unexpected error return from MC67'
         STOP
      END IF

C Write out the profile
      WRITE (6,'(3(/A,F6.0))')
     + ' The profile for initial PERM is ', RINFO(4),
     + ' Reduction after MC67A is        ', RINFO(1),
     + ' Final profile is                ', RINFO(5)
C Write out the permutation
      WRITE (6,'(/A,5I4)')
     + ' PERM =   ', (PERM(I),I=1,N)

      END
```

on the data

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

produces the output:

```
 Input matrix:
 Column  1:  1  5  4
 Column  2:  3  5  2
 Column  3:  2  3
 Column  4:  1  4
 Column  5:  1  2  5

 The profile for initial PERM is    12.
 Reduction after MC67A is            3.
 Final profile is                    9.

 PERM =      5   2   1   4   3
```