1 SUMMARY

This subroutine **assembles a set of element matrices**, that is, it forms the summation

$$\mathbf{A} = \sum_l \mathbf{A}^{[l]},$$

where each element matrix $\mathbf{A}^{[l]}$ has entries only in the principal submatrix corresponding to the variables in element l. Each $\mathbf{A}^{[l]}$ must be held in packed form as a small full square matrix, together with a list of the variables associated with element l. The assembled matrix $\mathbf{A}$ has a symmetric sparsity pattern but may be unsymmetric. An option exists for assembling only the sparsity pattern of $\mathbf{A}$. If the variables are not indexed contiguously, absent rows and columns may optionally be removed.

ATTRIBUTES — Version: 1.1.0. **Types:** Real (single, double). **Language:** Fortran 77. **Date:** December 1999. **Origin:** J. A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Argument lists

There is only one user-callable entry to this package.
The single precision version

```
CALL MC57A(LCNTL,NMAX,NELT,ELTVAR,ELTPTR,AELT,N,LIRN,IRN,IP,LA,A,IW,LP,INFO)
```

The double precision version

```
CALL MC57AD(LCNTL,NMAX,NELT,ELTVAR,ELTPTR,AELT,N,LIRN,IRN,IP,LA,A,IW,LP,INFO)
```

`LCNTL` is a `LOGICAL` array of length `10`. Components `LCNTL(I)`, $\texttt{I} = 1,2,3,4$ must be set by the user as follows.

`LCNTL(1)` must be set to `.TRUE.` if the matrix is symmetric. In this case, only the lower triangular part of each element matrix is needed and only the lower triangular part of $\mathbf{A}$ is assembled. Otherwise, the whole matrix is assembled.

`LCNTL(2)` must be set to `.TRUE.` if the user wants the entries within each column to be ordered by rows. Otherwise, the entries within each column are returned unordered.

`LCNTL(3)` must be set to `.TRUE.` if the user requires absent rows and columns to be removed. Otherwise, if the variables are not indexed contiguously from `1`, the assembled matrix will contain null rows and columns corresponding to unused indices.

`LCNTL(4)` must be set to `.TRUE.` if the entries of the matrix are to be assembled. Otherwise, only the sparsity pattern of the matrix is assembled.

`LCNTL(5)` to `LCNTL(10)` are not currently accessed by `MC57A/AD`.

`NMAX` is an `INTEGER` variable that must be set by the user to be at least as large as the largest integer used to index a variable in the element matrices. **Restriction:** $\texttt{NMAX} \geq 1$.

`NELT` is an `INTEGER` variable that must be set by the user to the total number of element matrices. **Restriction:** $\texttt{NELT} \geq 1$.

ELTVAR is an INTEGER array. On entry, the leading entries must contain lists of the variable indices associated with each of the element matrices, with those for element 1 preceding those for element 2, and so on. Variable indices outside the range [1,NMAX] are not allowed. Also, the same variable must not appear more than once for each element.

ELTPTR is an INTEGER array of length NELT+1. On entry, ELTPTR(I) must contain the position in ELTVAR of the first variable in element I (I = 1,2,...,NELT), and ELTPTR(NELT+1) must be set to the position after the last variable in the last element.

AELT is a REAL (DOUBLE PRECISION in the D version) array. If only the sparsity pattern is required (LCNTL(4) = .FALSE.), AELT is not accessed. Otherwise, AELT must be set by the user to hold the values of the entries in the element matrices, with the entries for element 1 preceding those for element 2, and so on. Each element matrix must be input by columns. If LCNTL(1) = .TRUE., only the entries in the lower triangular part of the element should be input (in packed triangular form).

N is an INTEGER variable that need not be set on entry. On exit, N is the order of the assembled matrix. Note that if absent rows and columns are removed (LCNTL(3) = .TRUE.) and the variables are not indexed contiguously from 1, N will be smaller than NMAX.

LIRN is an INTEGER variable that must be set by the user to the length of the array IRN. The minimum sufficient value for LIRN is $(\sum_l n^{[l]} * (n^{[l]}+1))/2$ if **A** is symmetric (LCNTL(1) = .TRUE.) and $\sum l n^{[l]} * n^{[l]}$ otherwise, where $n^{[l]}$ is the number of variables in element l. If LIRN is too small, the minimum sufficient value for LIRN is returned in INFO(2).

IRN is an INTEGER array of length LIRN. On exit, IRN holds the row indices of the entries of the assembled matrix, ordered by columns. If LCNTL(2) = .TRUE., the entries within each column are ordered. If **A** is symmetric (LCNTL(1) = .TRUE.), only the row indices of the lower triangular part are held.

IP is an INTEGER array of length NMAX+1. On exit, IP(J) holds the position in IRN and, if LCNTL(4) = TRUE., the position in A of the first entry in column J, J = 1,...,N, and IP(N+1) is set to the position immediately following the last entry in column N.

LA is an INTEGER variable that must be set by the user to the length of the array A. If only the sparsity pattern is required (LCNTL(4) = .FALSE.), LA may be set to 1. Otherwise, the minimum sufficient value for LA is equal to the minimum sufficient value for LIRN. If LA is too small, the minimum sufficient value for LA is returned in INFO(2).

A is a REAL (DOUBLE PRECISION in the D version) array of length LA. If only the sparsity pattern is required (LCNTL(4) = .FALSE.), A is not accessed. Otherwise, on exit A holds the entries of the assembled matrix, ordered by columns, so that entry A(K) is in row IRN(K), K = 1,...,IP(N+1)-1.

IW is an INTEGER array of length 2*NMAX that need not be set on entry. On exit, IW(J) is the original variable index for row (or column) J in the assembled matrix (1 ≤ J ≤ N). The order of the variables is retained. If the order of the assembled matrix is equal to the largest integer used to index a variable (that is, INFO(3) = N), IW(J) = J, 1 ≤ J ≤ N.

LP is an INTEGER variable that must be set by the user to the output stream for error messages. Printing of error messages is suppressed if LP is negative.

INFO is an INTEGER array of dimension 10 that need not be set on entry.

INFO(1) is used as an error flag. On successful exit, it is set to 0. If an error has been detected, INFO(1) is set to a negative value:

-1 - NMAX ≤ 0 or NELT ≤ 0. Immediate return with input parameters unchanged.

-2 – One or more variable index lies outside the range [1,NMAX].

-3 – Failure due to insufficient space allocated to the array IRN. INFO(2) is set to the minimum value that will suffice for LIRN.

-4 – Failure due to insufficient space allocated to the array A. INFO(2) is set to the minimum value that will suffice for LA.

-5 – One or more variable indices are duplicated within the list of variables for an element. INFO(5) holds the number of the first element found to have a duplicated index.

INFO(2) holds the minimum sufficient value for LIRN (if LCNTL(4) = .TRUE., this is also the minimum value for LA).

INFO(3) holds the largest integer used in the element variable lists to index a variable.

INFO(4) holds the number of entries in the assembled matrix (this is equal to IP(N+1)-1).

INFO(5) holds the index of the first element found to have duplicated variable indices (INFO(1) = -5 only).

INFO(6) holds the number of null rows (and columns) removed from the assembled matrix (LCNTL(3) = .TRUE.). If LCNTL(3) = .FALSE., this is equal to zero.

INFO(7) holds the smallest integer used in the element variable lists to index a variable.

INFO(8) to INFO(10) are not currently used.

3 GENERAL INFORMATION

Use of common: None.

Input/output: Error messages on unit LP (LP $<$ 0 suppresses them).

Other routines called directly: Subroutine MC57B/BD is internal to the package.

Restrictions: NMAX $\geq$ 1, NELT $\geq$ 1.

4 METHOD

After checking the scalar input data, the number of entries in each column is obtained by a counting pass. At this stage, no account is taken of the overlap between elements. During the counting pass, a check is made for out of range indices and for duplicate indices within an element. If any such indices are found, the computation terminates with an error message. Each element in turn is made the current element. The columns of the current element are taken in turn and the row indices set in IRN and, if LCNTL(4) = .TRUE., the values in A. Once each element has been considered, if the user wants null rows removed from the assembled matrix (LCNTL(3) = .TRUE.), a check is made for such rows. If there are null rows, they are removed by shunting row indices. A final pass through each column is made to sum the duplicates that result from overlapping elements.

MC57B/BD is called if the user wants the entries within each column to be ordered. MC57B/BD uses a pairwise interchange algorithm of maximum order $r(r-1)/2$, for each column, where r is the number of entries in the column.

5 EXAMPLE OF USE

We wish to assemble the following element matrices:

$$\begin{array}{c} 4 \\ 8 \end{array}\begin{pmatrix} 2. & 1. \\ 1. & 7. \end{pmatrix} \qquad \begin{array}{c} 8 \\ 10 \end{array}\begin{pmatrix} 3. & 2. \\ 2. & 8. \end{pmatrix} \qquad \begin{array}{c} 4 \\ 8 \\ 1 \\ 2 \end{array}\begin{pmatrix} 4. & 3. & 2. & 3. \\ 3. & 1. & 3. & 2. \\ 2. & 3. & 6. & 1. \\ 3. & 2. & 1. & 5. \end{pmatrix} \qquad \begin{array}{c} 8 \\ 10 \\ 2 \\ 3 \end{array}\begin{pmatrix} 2. & 1. & 8. & 3. \\ 1. & 3. & 2. & 2. \\ 8. & 2. & 2. & 5. \\ 3. & 2. & 5. & 4. \end{pmatrix}$$

where the variable indices are indicated by the integers before each matrix. The following program may be used.

```
C Example to illustrate the use of MC57.
C
C     .. Parameters ..
      INTEGER LAELT,NMAX,NZMAX,MELT
      PARAMETER (LAELT=30,NMAX=10,NZMAX=30,MELT=4)
C     ..
C     .. Local Scalars ..
      INTEGER I,J,K,K1,K2,LA,LP,LIRN,N,NELT,NENTRY,NVAR,NZ
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION AELT(LAELT),A(LAELT)
      INTEGER ELTPTR(MELT+1),ELTVAR(NZMAX),IRN(LAELT),
     +        INFO(10),IP(NMAX+1),IW(2*NMAX)
      LOGICAL LCNTL(10),
C     ..
C     .. External Subroutines ..
      EXTERNAL MC57AD
C     ..
C Read in element data.
      READ (5,FMT=*) NELT
      READ (5,FMT=*) (ELTPTR(I),I=1,NELT+1)
      NZ = ELTPTR(NELT+1) - 1
      IF (NZ.GT.NZMAX) GO TO 110
      READ (5,FMT=*) (ELTVAR(I),I=1,NZ)
C Add up number of entries in element matrices
C (since symmetric, only the lower triangular part is needed.
      NENTRY = 0
      DO 10 I = 1,NELT
         NVAR = ELTPTR(I+1) - ELTPTR(I)
         NENTRY = NENTRY + (NVAR*(NVAR+1))/2
  10  CONTINUE
      IF (NENTRY.GT.LAELT) GO TO 110
      READ (5,FMT=*) (AELT(I),I=1,NENTRY)
C Set unit number for printing
      LP = 6
C Symmetric matrix ... only want lower triangle
      LCNTL(1) = .TRUE.
C Order by rows within each column
      LCNTL(2) = .TRUE.
```

```
C Remove null rows
      LCNTL(3) = .TRUE.
C Matrix values to be assembled
      LIRN = NENTRY
      LA = LIRN
      LCNTL(4) = .TRUE.

      CALL MC57AD(LCNTL,NMAX,NELT,ELTVAR,ELTPTR,AELT,N,LIRN,IRN,IP,
     +            LA,A,IW,INFO)
      IF (INFO(1).LT.0) GO TO 110

      WRITE(6,'(/A,I6)')  'Largest index         = ', INFO(3)
      WRITE(6,'(A,I6)')   'Order of matrix       = ', N
      WRITE(6,'(A,I6)')   'Minimum value of LIRN = ', INFO(2)
      WRITE(6,'(A,I6)')   'No. entries in assembled matrix = ', INFO(4)
C Write out assembled matrix (lower triangle only), column by column
      DO 30 J = 1,N
         K1 = IP(J)
         K2 = IP(J+1)-1
         WRITE(6,'(A,I2)') ' Column ',J
         DO 20 K = K1,K2
            WRITE(6,'(A,I2,A,F4.1)') '   Row ',IRN(K),' Value ',A(K)
   20    CONTINUE
   30 CONTINUE
  110 STOP
      END
```

The input data used for this problem is:

```
    4
    1    3    5    9   13
    4    8    8   10    4    8    1    2    8   10    2    3
    2.   1.   7.   3.   2.   8.   4.   3.   2.   3.   1.   3.
    2.   6.   1.   5.   2.   1.   8.   3.   3.   2.   2.   2.
    5.   4.
```

This produces the following output:

```
Largest index         =     10
Order of matrix       =      6
Minimum value of LIRN =     26
No. entries in assembled matrix =     17
 Column  1
   Row  1 Value  6.0
   Row  2 Value  1.0
   Row  4 Value  2.0
   Row  5 Value  3.0
 Column  2
   Row  2 Value  7.0
   Row  3 Value  5.0
```

```
   Row  4 Value  3.0
   Row  5 Value 10.0
   Row  6 Value  2.0
 Column  3
   Row  3 Value  4.0
   Row  5 Value  3.0
   Row  6 Value  2.0
 Column  4
   Row  4 Value  6.0
   Row  5 Value  4.0
 Column  5
   Row  5 Value 13.0
   Row  6 Value  3.0
 Column  6
   Row  6 Value 11.0
```