1 SUMMARY

Given the structure of an unassembled finite-element matrix, this subroutine **groups the variables into supervariables and optionally generates either the element connectivity graph or the supervariable connectivity graph.**

A supervariable is a collection of one or more variables, such that each variable belongs to the same set of finite elements. In the supervariable connectivity graph, the nodes are the supervariables and the edges are constructed by making the supervariables of each finite element pairwise adjacent. The supervariable connectivity graph, together with the number of variables in each supervariable, provide a compact representation of the variable connectivity graph. In the element connectivity graph, the nodes are the elements and the edges are constructed by defining two elements to be adjacent whenever they have one or more variables in common.

ATTRIBUTES — **Version:** 1.0.0. (12 July 2004) **Types:** Real (single, double). **Remark:** `MC44` is used by `MC53`. **Language:** Fortran 77. **Original date:** March 1995. **Origin:** J.K. Reid and J.A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequences

Note that generating element connectivity graphs involves only integers so the single and double precision versions, `MC44A` and `MC44AD`, are identical. Both versions are included for ease of use.

There are two subroutines that can be called by the user:

(a) `MC44A/AD` groups the variables into supervariables.

(b) `MC44B/BD` uses the supervariables to generate either the element connectivity graph or the supervariable connectivity graph. `MC44B/BD` must be preceded by a call to `MC44A/AD`.

2.1.1 To group the variables into supervariables

The single precision version

```
CALL MC44A(N,NELT,NZ,ELTVAR,ELTPTR,NSUP,SVAR,LIW,IW,LP,INFO)
```

The double precision version

```
CALL MC44AD(N,NELT,NZ,ELTVAR,ELTPTR,NSUP,SVAR,LIW,IW,LP,INFO)
```

`N` 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 finite-element problem. This argument is not altered by the routine. **Restriction:** `N`$\geq$`1`.

`NELT` is an `INTEGER` variable that must be set by the user to the number of finite elements in the problem. This argument is not altered by the routine. **Restriction:** `NELT`$\geq$`1`.

`NZ` is an `INTEGER` variable that must be set by the user to the length of the array `ELTVAR`. This argument is not altered by the routine. **Restriction:** `NZ`$\geq$`ELTPTR(NELT+1)-1`.

`ELTVAR` is an `INTEGER` array of length `NZ`. On entry, `ELTVAR` must contain lists of the variables belonging to each of the finite elements, with those for element `1` preceding those for element `2`, and so on. If the user inputs a variable index `ELTVAR(I)` outside the range `1`$\leq$`ELTVAR(I)`$\leq$`N`, the index is reset to `0` and ignored (see `INFO(2)`). Similarly, duplicated indices are set to `0` and ignored (see `INFO(3)`).

`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. This argument is not altered by the routine.

NSUP is an INTEGER variable that need not be set by the user. On successful exit, NSUP holds the number of supervariables.

SVAR is an INTEGER array of dimension 0:N that need not be set by the user. On successful exit, SVAR(0) = 0 and, if I was used to index a variable, SVAR(I) contains the supervariable to which variable I belongs; if I is not used to index a variable, SVAR(I) = 0, I = 1, 2,..., N.

LIW is an INTEGER variable that defines the length of the array IW. A sufficient value for LIW is NSUP*3 + 3. If LIW is too small, a value that will suffice is returned in INFO(4). This argument is not altered by the routine.

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

LP is an INTEGER variable that must be set by the user to the stream number for error messages. Printing of error messages is suppressed if LP ≤ 0. This argument is not altered by the routine.

INFO is an INTEGER array of length 6 that need not be set by the user. On exit from MC44A/AD, a value for INFO(1) of zero indicates that the routine has performed successfully. For nonzero values, see Section 2.2. On exit with INFO(1) = 0, INFO(2) contains the number of variable indices that were ignored because they were out of range and INFO(3) contains the number of variable indices that were ignored because they were duplicates. On exit with INFO(1) = 0, INFO(4) contains the minimum sufficient value for LIW, and if INFO(1) = -4, INFO(4) contains a value for LIW which will suffice (but which may be larger than the minimum sufficient value). INFO(5) and INFO(6) are not accessed.

2.1.2 To generate the connectivity graph

The single precision version

```
      CALL MC44B(LGRAPH,N,NELT,NZ,ELTVAR,ELTPTR,NSUP,SVAR,
     +           LENIRN,IRN,LENIP,IP,LIW,IW,LP,INFO)
```

The double precision version

```
      CALL MC44BD(LGRAPH,N,NELT,NZ,ELTVAR,ELTPTR,NSUP,SVAR,
     +            LENIRN,IRN,LENIP,IP,LIW,IW,LP,INFO)
```

LGRAPH is a LOGICAL variable that must be set by the user. If LGRAPH = .TRUE., the element connectivity graph is generated; otherwise, the supervariable connectivity graph is generated. This argument is not altered by the routine.

N, NELT, NZ, ELTVAR, ELTPTR, NSUP, SVAR must all be unchanged since the call to MC44A/AD.

LENIRN is an INTEGER variable that must be set by the user to the length of the array IRN. If LENIRN is too small, a value that will suffice is returned in INFO(6). This argument is not altered by the routine.

IRN is an INTEGER array of length LENIRN that need not be set by the user. IRN is used to hold the element adjacency lists (LGRAPH = .TRUE.) or the supervariable adjacency lists (LGRAPH = .FALSE.). On successful exit, if LGRAPH = .TRUE., the first IP(NELT+1)-1 entries of IRN contain element adjacency lists. For element J, only the adjacent elements I with I > J are held. The elements which are adjacent to element 1 precede those for element 2 and so on. If LGRAPH = .FALSE., the first IP(NSUP+1)-1 entries of IRN contain supervariable adjacency lists. For supervariable J, only the adjacent supervariables I with I > J are held. The supervariables which are adjacent to supervariable 1 precede those for supervariable 2 and so on.

LENIP is an INTEGER variable that defines the length of the array IP. If LGRAPH = .TRUE., LENIP must be at least NELT+1. If LGRAPH = .FALSE., LENIP must be at least NSUP+1. This argument is not altered by the routine.

IP is an INTEGER array of length LENIP that need not be set by the user. On successful exit, if LGRAPH = .TRUE., IP(J) contains the position in IRN of the first element which is adjacent to element J (J = 1, 2,..., NELT) and

IP(NELT+1) contains the position after the last adjacent element for element NELT. If LGRAPH = .FALSE., IP(J) contains the position in IRN of the first supervariable which is adjacent to supervariable J (J = 1, 2,..., NSUP) and IP(NSUP+1) contains the position after the last adjacent supervariable for supervariable NSUP.

LIW is an INTEGER variable that defines the length of the array IW. A sufficient value for LIW is 2*NZ + NELT + NSUP + 3 + max(NSUP, NELT). This argument is not altered by the routine.

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

LP is an INTEGER variable that must be set by the user to the stream number for error messages. Printing of error messages is suppressed if LP ≤ 0. This argument is not altered by the routine.

INFO is an INTEGER array of length 6 that need not be set by the user. On exit from MC44B/BD, a value for INFO(1) of zero indicates that the routine has performed successfully. For nonzero values, see Section 2.2. On exit, INFO(5) contains the minimum sufficient value for LIW and INFO(6) contains the minimum sufficient value for LENIRN. INFO(2), INFO(3), and INFO(4) are not accessed.

2.2 Error diagnostics

On successful completion, MC44A/AD and MC44B/BD will exit with the parameter INFO(1) set to 0. Other values for INFO(1) and the reasons for them are given below. In each case, a self-explanatory message is output on unit LP.

–1 N ≤ 0 on entry to MC44A/AD. Immediate return (MC44A/AD entry only).

–2 NELT ≤ 0 on entry to MC44A/AD. Immediate return (MC44A/AD entry only).

–3 NZ < ELTPTR(NELT+1)-1 on entry to MC44A/AD. Immediate return (MC44A/AD entry only).

–4 Failure due to insufficient space allocated to the array LIW (MC44A/AD and MC44B/BD entries). If this error is returned by MC44A/AD, INFO(4) holds a value that will suffice for LIW, and if it is returned by MC44B/BD, INFO(5) holds the minimum sufficient value for LIW.

–5 Failure due to insufficient space allocated to the array IRN. INFO(6) is set to the minimum sufficient value for LENIRN (MC44B/BD entry only).

–6 Failure due to insufficient space allocated to the array IP. Immediate return (MC44B/BD entry only).

3 GENERAL INFORMATION

Workspace: An integer array IW of length LIW is used by the routine as workspace.

Use of common: None

Other routines called directly: Subroutines internal to the package are MC44C/CD and MC44D/DD.

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

Restrictions:

N ≥ 1,

NELT ≥ 1,

NZ ≥ ELTPTR(NELT+1)-1.

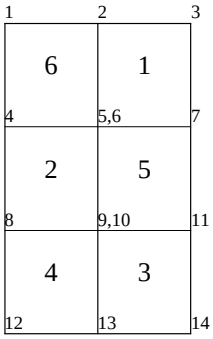
4 METHOD

In MC44A/AD, the supervariables are constructed progressively so that after j steps we have the supervariable structure for the subproblem consisting of the first j elements. We start with all variables in supervariable 0, which corresponds to the problem with no elements. At stage j, if some but not all of the variables of a supervariable are in the element, we create a new supervariable and move the variables involved to it. In a first pass through the list, we decrement the counts of numbers of variables in the supervariables to exclude the variables of element j. In a second pass, we can see which new supervariables are needed and increment the counts in the supervariables (new or old). The overall complexity of this algorithm is $O(n+nz)$, where nz is the total length of the element lists.

In MC44B/BD, supervariable lists for the elements are constructed and for each supervariable, the number of elements involving it is counted. This allows element lists for the supervariables to be constructed. Finally, the element connectivity graph is constructed from the supervariable lists or the supervariable connectivity graph is constructed from the element lists.

5 EXAMPLE OF USE

The following program provides an example of the use of MC44. We wish to generate the element connectivity graph for the following simple finite-element mesh comprising six 4-noded quadrilateral elements. The elements 1, 3, 4, and 6 each have 4 variables and elements 2 and 5 each have 6 variables (the finite element nodes which are on the boundary each have one variable and those not on the boundary have two variables). The elements are numbered arbitrarily and the variables are numbered pagewise parallel to the shortest side.




```
C  Example to illustrate the use of MC44A.
C
C     .. Parameters ..
      INTEGER MAXELT,MAXNZ,LENIRN,MAXN,LIWMX
      PARAMETER (MAXELT=6,MAXNZ=34,LENIRN=30,MAXN=14,LIWMX=100)
C     ..
C     .. Local Scalars ..
      INTEGER I,IELT,ISTOP,ISTRT,LENIP,LIW,LP,N,NELT,NSUP,NZ
      LOGICAL LGRAPH
C     ..
C     .. Local Arrays ..
      INTEGER ELTPTR(MAXELT+1),ELTVAR(MAXNZ),INFO(6),
     +        IP(MAXELT+1),IRN(LENIRN),IW(LIWMX),SVAR(0:MAXN)
C     ..
C     .. External Subroutines ..
      EXTERNAL MC44A,MC44B
C     ..
C   Read in the finite-element data
C
      READ (5,FMT=*) N,NELT
      IF (N.GT.MAXN) THEN
```

```
         WRITE (6,FMT=*) ' N is too large.'
         GO TO 20
      END IF
      IF (NELT.GT.MAXELT) THEN
         WRITE (6,FMT=*) ' NELT is too large.'
         GO TO 20
      END IF

      READ (5,FMT=*) (ELTPTR(I),I=1,NELT+1)
      NZ = ELTPTR(NELT+1) - 1
      IF (NZ.GT.MAXNZ) THEN
         WRITE (6,FMT=*) ' NZ is too large.'
         GO TO 20
      END IF

      READ (5,FMT=*) (ELTVAR(I),I=1,NZ)

      LP = 6
C
C Generate supervariables
      LIW = LIWMX
      CALL MC44A(N,NELT,NZ,ELTVAR,ELTPTR,NSUP,SVAR,LIW,IW,LP,INFO)
      IF (INFO(1).LT.0) THEN
         WRITE (6,FMT=*) ' Unexpected error return from MC44A.'
         GO TO 20
      END IF
      WRITE (6,FMT=9020) NSUP
      WRITE (6,FMT=9010) INFO(4)

C  Generate the element connectivity graph
      LGRAPH = .TRUE.
      LENIP = NELT + 1
      LIW = MAX(NSUP,NELT) + 2*NZ + NELT + NSUP + 3
      IF (LIW.GT.LIWMX) THEN
         WRITE (6,FMT=*) ' LIWMX is too small'
         GO TO 20
      END IF

      CALL MC44B(LGRAPH,N,NELT,NZ,ELTVAR,ELTPTR,NSUP,SVAR,
     +           LENIRN,IRN,LENIP,IP,LIW,IW,LP,INFO)
      IF (INFO(1).LT.0) THEN
         WRITE (6,FMT=*) ' Unexpected error return from MC44B.'
         GO TO 20
      END IF

      WRITE (6,FMT=9000) INFO(6)
      WRITE (6,FMT=9040) INFO(5)
      DO 10 IELT = 1,NELT
         ISTRT = IP(IELT)
         ISTOP = IP(IELT+1) - 1
         WRITE (6,FMT=9030) IELT, (IRN(I),I=ISTRT,ISTOP)
   10 CONTINUE
C
   20 STOP


 9000 FORMAT (/' Minimum value for LENIRN is: ',I3)
 9010 FORMAT (/' Minimum value for LIW (MC44A) is:    ',I3)
 9020 FORMAT (/' The number of supervariables is: ',I3)
 9030 FORMAT (/' The elements which are adjacent to element ',I2,
     +       '  are:',5I4)
```

```
 9040 FORMAT (/' Minimum value for LIW (MC44B) is:    ',I3)
      END
```

The input data used for this problem is:

```
  14   6
   1   6  12  17  22  28  33
   2   5   3   6   7   4   5   6   9   8  10  14   9  11
  10  13  12  13   8   9  10   5   6   7   9  11  10   1
   2   5   4   6
```

This produces the following output:

```
 The number of supervariables is:  12

 Minimum value for LIW (MC44A) is:     39

 Minimum value for LENIRN is:  11

 Minimum value for LIW (MC44B) is:     97

 The elements which are adjacent to element  1  are:   6   5   2

 The elements which are adjacent to element  2  are:   6   5   4   3

 The elements which are adjacent to element  3  are:   5   4

 The elements which are adjacent to element  4  are:   5

 The elements which are adjacent to element  5  are:   6

 The elements which are adjacent to element  6  are:
```