1 SUMMARY

This subroutine **generates an ordering for finite-element matrices** that is efficient when subsequently used with a frontal solver (for example, the HSL subroutine MA42). The storage required by a frontal solver and the time taken to run it are dependent upon the order in which the elements are input; the variation in the performance of different element orderings can be significant. The ordering obtained by MC43 is designed to reduce the profile, the maximum wavefront (frontwidth), and the root mean-squared wavefront, which in turn reduce storage requirements and computation times for the frontal solver.

The subroutine allows the user to choose one of two reordering algorithms: an indirect element reordering algorithm or a direct element reordering algorithm. For a given problem it is difficult to predict which algorithm will produce an element ordering that will subsequently give the smallest profile and wavefronts in the frontal solver. For the reordering algorithm chosen by the user, MC43 returns the maximum wavefront for the original and permuted element orderings. The user may want to run both reordering algorithms; the ordering which gives the smallest maximum wavefront should then be chosen.

ATTRIBUTES — **Version:** 1.0.0. **Types:** MC43A, MC43AD. **Calls:** MC34. **Original date:** September 1988. **Origin:** J.A.Scott, Harwell.

2 HOW TO USE THE PACKAGE

2.1 The argument list

The single precision version

```
CALL MC43A(ICNTL,N,NELT,NZ,ELTVAR,ELTPTR,NORDER,LIW,IW,MXWAVE,IFLAG)
```

The double precision version

```
CALL MC43AD(ICNTL,N,NELT,NZ,ELTVAR,ELTPTR,NORDER,LIW,IW,MXWAVE,IFLAG)
```

ICNTL is an INTEGER variable which must be set by the user to 0 or 1. If ICNTL=0 the direct element reordering algorithm is employed; if ICNTL=1 the indirect element reordering algorithm is employed. This argument is not altered by the routine. **Restriction**: ICNTL = 0 or 1.

N is an INTEGER variable which must be set by the user to the largest integer used to number a variable in the finite-element problem. This argument is not altered by the routine. **Restriction:** $N \geq 1$.

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

NZ is an INTEGER variable which must be set by the user to at least the total number of entries in the element variable lists. This argument is not altered by the routine. **Restriction:** $NZ \geq 1$.

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. This argument is not altered by the routine.

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.

NORDER is an INTEGER array of length NELT. This array need not be set on entry. On exit, the order in which the finite elements should be presented to the frontal solver is given by NORDER(1), NORDER(2),..., NORDER(NELT).

`LIW` is an `INTEGER` variable which defines the length of the work array `IW`. The workspace required depends upon `ICNTL`. If `ICNTL=0` and `NELT>1`, the workspace required will always be at least max(`2*N, 3*NZ+2*NELT+N+3`) but never exceeds max(`2*N, 3*NZ+2*NELT*(MAXEL+3)+N+3`), where `MAXEL` is the maximum number of finite elements to which any one variable in the problem belongs. If `ICNTL=1` and `NELT>1`, the workspace required will always be at least max(`3*NZ+NELT+N+3, NELT+64`) but never exceeds max(`3*(N+NZ+1)+NELT*(NODE*NODE+1), NELT+64`),
where `NODE` is the maximum number of nodes in an element. This argument is not altered by the routine.

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

`MXWAVE` is an `INTEGER` array of length `2` which need not be set by the user. On exit, `MXWAVE(1)` and `MXWAVE(2)` are set to the maximum wavefront for the element order `1`, `2`,..., `NELT`, and for the permuted order `NORDER(1)`, `NORDER(2)`,..., `NORDER(NELT)`, respectively, assuming no stability restrictions. In practice, the size of the maximum wavefront in the frontal solver may be somewhat larger because of stability considerations, but `MXWAVE(1)` and `MXWAVE(2)` provide lower bounds on the size of the frontal matrix for the original and new element orderings.

`IFLAG` is an `INTEGER` variable which need not be set by the user. On exit, `IFLAG` = `0` indicates that the subroutine has performed successfully. Negative values indicate an error (see § 2.3).

2.2 Common blocks

One common block is used. The common block is:

The single precision version

```
COMMON/ MC43T/ LP, IWUSE(2)
```

The double precision version

```
COMMON/ MC43TD/ LP, IWUSE(2)
```

where the parameters are given default values by a block data subprogram `MC43U/UD`.

`LP` is an `INTEGER` variable used as the unit number of the device to which error messages are sent. The default value is `6`. Error messages can be suppressed by setting `LP=0`. This parameter is not altered by the subroutine.

`IWUSE` is an `INTEGER` array of length `2`. On successful exit from `MC43A/AD`, `IWUSE(1)` is set to the amount of workspace used by the subroutine. If the user has provided insufficient workspace, `IWUSE(1)` is set to a value which may suffice for `LIW`, and `IWUSE(2)` is set to a value which will certainly allow enough workspace.

2.3 Errors and diagnostic messages

Successful return from `MC43A/AD` is indicated by a value of `IFLAG` equal to `0`. Possible nonzero values of `IFLAG` are given below. In each case an identifying message is output on unit `LP`.

–1 – `N` ≤ `0`. Immediate return with input parameters unchanged.
–2 – `NELT` ≤ `0`. Immediate return with input parameters unchanged.
–3 – `NZ` ≤ `0`. Immediate return with input parameters unchanged.
–4 – The user has violated the restriction `ICNTL=0` or `1`. Immediate return with input parameters unchanged.
–5 – Failure due to insufficient space allocated to the array `IW`. `IWUSE(1)` in `COMMON` is set to a value which may suffice for `LIW`, and `IWUSE(2)` is set to a value which will certainly allow enough workspace.
–6 – The user has input variable indices `ELTVAR(I)` outside the range `1`≤`ELTVAR(I)`≤`N`. Immediate return with input parameters unchanged.

3 GENERAL INFORMATION

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

Use of common: The subroutine uses the common block `MC43T/TD`; see § 2.2.

Other routines called directly: Subroutines internal to the package are `MC43B/BD`, `MC43C/CD`, `MC43D/DD`, `MC43E/ED`, `MC43F/FD`, `MC43G/GD`, `MC43H/HD`, `MC43I/ID`, `MC43J/JD`, `MC43K/KD`, `MC43L/LD`, `MC43M/MD`, `MC43N/ND`, `MC43O/OD`, `MC43P/PD`, `MC43Q/QD`, `MC43R/RD`, and `MC43S/SD`. In addition, `MC34A/AD` is called.

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

Restrictions:

$N \geq 1$,

$NELT \geq 1$,

$NZ \geq 1$,

`ICNTL` = `0` or `1`.

4 METHOD

The user chooses whether a direct or an indirect algorithm is used to resequence the finite elements. Both algorithms are based upon modifications of Sloan's algorithm (Sloan 1986) for the reduction of the profile of a sparse matrix with a symmetric sparsity pattern (see the HSL subroutine `MC40`). The algorithms use the user-supplied element variable lists to group the variables into supervariables, where a supervariable is a collection of one or more variables that belong to the same set of finite elements.

In the indirect element reordering algorithm the first step is to generate the supervariables and the supervariable connectivity graph. The supervariable connectivity graph is relabelled using a modification of Sloan's algorithm. The modification takes into account the number of variables belonging to each supervariable. The new supervariable labels are used to resequence the elements in ascending order of their lowest numbered supervariable.

In the direct element reordering algorithm the first step is to generate the supervariables and element connectivity graph, with two elements defined to be adjacent whenever they have a supervariable in common. The element connectivity graph is relabelled using another modification of Sloan's algorithm, which considers the growth in the front size as each element is given a new number. The new labelling of the element connectivity graph provides an efficient element ordering for the use with a frontal solver.

For full details of both algorithms the user is referred to Duff, Reid, and Scott (1988).

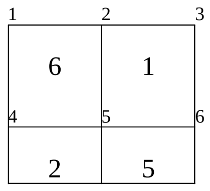
References

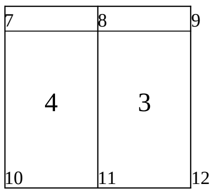
Duff, I.S., Reid, J.K., and Scott, J.A. (1988) The use of profile reduction algorithms with a frontal code. CSS Report 224, Harwell Laboratory.

Sloan, S.W. (1986) An algorithm for profile and wavefront reduction of sparse matrices. Inter. J. Numer. Meth. Engng. **23**, 239-251.

5 EXAMPLE OF USE

The following program provides an example of the use of `MC43`. We wish to reorder the elements in the following simple finite-element mesh comprising six 4-noded quadrilateral elements with one freedom per node. The elements are initially numbered arbitrarily and the freedoms are numbered pagewise parallel to the shortest side.





```
C  EXAMPLE TO ILLUSTRATE THE USE OF MC43A.
C  BOTH THE DIRECT AND THE INDIRECT ELEMENT REORDERING ALGORITHMS
C  ARE EMPLOYED.
C
      PARAMETER( MELT=6, MZ=24, LMAX=200)
      INTEGER ICNTL, IFLAG, N, NZ, NELT, LIW,
     *        NORDER(MELT), ELTPTR(MELT+1), ELTVAR(MZ), IW(LMAX),
     *        MXWAVE(2)
      COMMON /MC43T/ LP, IWUSE(2)
      DATA  LIW/200/
C
C   READ IN THE FINITE-ELEMENT DATA
C
      READ(5,*) N, NELT
      READ(5,*) (ELTPTR(I), I=1,NELT+1)
      NZ = ELTPTR(NELT+1)-1
      READ(5,*) (ELTVAR(I), I=1,NZ)
C
C   *** CALL THE REORDERING ALGORITHM ***
C   FIRST CALL THE DIRECT ELEMENT REORDERING ALGORITHM
C   AND THEN THE INDIRECT REORDERING ALGORITHM
C
      ICNTL = 0
      WRITE(6,980)
      CALL MC43A(ICNTL,N,NELT,NZ,
     *           ELTVAR,ELTPTR,NORDER,LIW,IW,MXWAVE,IFLAG)
      IF(IFLAG.LT.0) THEN
        WRITE(6,990)
        STOP
      END IF
      IF(MXWAVE(1).NE.MXWAVE(2) THEN
         WRITE(6,985) MXWAVE(1)
         WRITE(6,986) MXWAVE(2)
      ELSE
         WRITE(6,987)
         WRITE(6,988) MXWAVE(1)
      END IF
      WRITE(6,991) IWUSE(1)
      WRITE(6,992)
      WRITE(6,993) (NORDER(I), I=1,NELT)
C
      ICNTL = 1
      WRITE(6,981)
      CALL MC43A(ICNTL,N,NELT,NZ,
     *           ELTVAR,ELTPTR,NORDER,LIW,IW,MXWAVE,IFLAG)
      IF(IFLAG.LT.0) THEN
        WRITE(6,990)
        STOP
      END IF
      IF(MXWAVE(1).NE.MXWAVE(2) THEN
         WRITE(6,985) MXWAVE(1)
         WRITE(6,986) MXWAVE(2)
      ELSE
         WRITE(6,987)
         WRITE(6,988) MXWAVE(1)
      END IF
      WRITE(6,991) IWUSE(1)
      WRITE(6,992)
      WRITE(6,993) (NORDER(I), I=1,NELT)
C
 980  FORMAT(/3X,'*** DIRECT ELEMENT REORDERING ***')
 981  FORMAT(/3X,'*** INDIRECT ELEMENT REORDERING ***')
 985  FORMAT(/3X,'MAX. WAVEFRONT FOR ORIGINAL ELEMENT ORDER :',I5)
 986  FORMAT( 3X,'MAX. WAVEFRONT FOR NEW ELEMENT ORDER      :',I5)
 987  FORMAT(/3X,'THE MAX. WAVEFRONT WAS NOT REDUCED')
 988  FORMAT(/3X,'MAX. WAVEFRONT IS :',I5)
 990  FORMAT(/3X,'PROGRAM TERMINATED WITHOUT COMPLETION')
 991  FORMAT(/3X,'WORKSPACE USED BY MC43A WAS ',I5)
 992  FORMAT(/3X,'THE NEW ELEMENT ORDER IS :')
 993  FORMAT(6I5)
      STOP
      END
```

The input data used for this problem is:

```
12   6
 1   5   9  13  17  21  25
 2   5   3   6   4   5   7   8   8  12   9  11
 7   8  10  11   5   8   9   6   1   2   5   4
```

This produces the following output:

```
*** DIRECT ELEMENT REORDERING ***

MAX. WAVEFRONT FOR ORIGINAL ELEMENT ORDER :    8
MAX. WAVEFRONT FOR NEW ELEMENT ORDER      :    5

WORKSPACE USED BY MC43A WAS   123

THE NEW ELEMENT ORDER IS :
 1    6    5    2    4    3

*** INDIRECT ELEMENT REORDERING ***

MAX. WAVEFRONT FOR ORIGINAL ELEMENT ORDER :    8
MAX. WAVEFRONT FOR NEW ELEMENT ORDER      :    5

WORKSPACE USED BY MC43A WAS   165

THE NEW ELEMENT ORDER IS :
 6    1    2    5    3    4
```