1 SUMMARY

This subroutine **generates an ordering for finite-element matrices within a subdomain** that is efficient when subsequently used with a multiple front algorithm. In a multiple front algorithm, the finite-element domain is partitioned into a number of subdomains and a frontal decomposition is performed on each subdomain separately. The storage required by a multiple front algorithm and the time taken to run it are dependent upon the order in which the elements in each subdomain are input; the variation in the performance of different element orderings can be significant. The ordering obtained by MC53 is designed to reduce the maximum and root mean-squared frontsizes and to reduce the floating-point operation count for the frontal solver on the subdomain. If *nelt* is the number of elements in the subdomain and $fsize_i$ is the number of variables in the front after the assembly of the ith element, the maximum frontsize *fmax* in the subdomain is defined to be

$$fmax = \max_{1 \le i \le nelt} \{fsize_i\}$$

and the root mean-squared frontsize *frms* in the subdomain is defined to be

$$frms = \sqrt{\frac{1}{nelt} \sum_{i=1}^{nelt} fsize_i^2}.$$

The user is required to use reverse communication to supply a list of the variables belonging to each element in the subdomain one at a time followed by a list of the variables lying on the subdomain interface.

ATTRIBUTES — **Version:** 1.1.0. (9 April 2013) **Types:** Real (single, double). **Calls:** MC44 and KB07. **Original date:** March 1995. **Origin:** J.A.Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequence

There are two entries:

(a) MC53I/ID sets default values for control parameters. It should normally be called once prior to calling MC53A/AD.

(b) MC53A/AD reorders the elements in a subdomain. It must be called *nelt*+1 times, where *nelt* is the number of finite elements in the subdomain. On each of the first *nelt* calls, a list of the variables associated with one of the elements in the subdomain must be passed to the routine (one call for each element and these calls may be made in any order). On the final call, a list of the variables lying on the subdomain interfaces (that is, variables which belong to at least one other subdomain) must be passed to the routine. This list is called the *guard element* for the subdomain and may be generated using the HSL routine MA52A/AD.

2.1.1 To set default values for the control parameters

The single precision version

```
CALL MC53I(ICNTL)
```

The double precision version

```
CALL MC53ID(ICNTL)
```

ICNTL is an INTEGER array of length 5 which need not be set by the user. On return it contains default values (see Section 2.2.1 for details).

2.1.2 To reorder the elements in a subdomain

A call of the following form must be made for each element in the finite-element mesh, followed by a call for the guard element.

The single precision version

```
CALL MC53A(ICALL,NELT,NVAR,IVAR,NORDER,LIW,IW,ICNTL,INFO,RINFO)
```

The double precision version

```
CALL MC53AD(ICALL,NELT,NVAR,IVAR,NORDER,LIW,IW,ICNTL,INFO,RINFO)
```

ICALL is an INTEGER variable which must be set by the user. On the first call ICALL must be set to 1, on the second call it must be set to 2, and so on. On the final call it must be set to NELT+1. This argument is not altered by the routine.

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

NVAR is an INTEGER variable which must be set by the user. On the first NELT calls, NVAR must be set to the number of variables in the current finite element and on the final call it must be set to the number of variables in the guard element. This argument is not altered by the routine. **Restriction:** NVAR ≥ 1.

IVAR is an INTEGER array of length NVAR which must be set by the user. On the first NELT calls, IVAR must contain the indices of the variables associated with the current finite element and on the final call it must contain the indices of the variables in the guard element (the indices of the variables belonging to at least one other subdomain). Variables I for which IVAR(I) < 1 are ignored (see INFO(8)). This argument is not altered by the routine.

NORDER is an INTEGER array of length NELT+1 which need not be set by the user. This argument must be unchanged between calls to MC53A/AD. On exit from the final call, the first NELT entries of NORDER hold the order in which the finite elements should be presented to the frontal solver on the subdomain. If element *ielt* was passed to MC53A/AD on the *i*th call, then the position of element *ielt* in the new ordering is NORDER(*i*). NORDER(NELT+1) is set to NELT+1 on exit.

LIW is an INTEGER variable which must be set by the user to define the length of the array IW. Let $\mathtt{nz} = \sum_{i=1}^{\mathtt{NELT}} \mathtt{nvar}_i$, where $\mathtt{nvar}_i$ is the number of variables in element *i*, and let n be the largest integer used to index a variable in the subdomain (see INFO(3)). The workspace required will always exceed 3*nz+4*NELT+n+7 but never exceeds 3*nz+NELT*(maxnei+10)+maxnei+3*n+14, where maxnei is the largest number of elements which are adjacent to a single element (including the guard) in the subdomain (two elements are adjacent if they have a variable in common). If the user-supplied value of LIW is too small, a value which may suffice is returned in INFO(6). This argument is not altered by the routine.

IW is an INTEGER array of length LIW which is used as workspace. This argument must be unchanged between calls to MC53A/AD.

ICNTL is an INTEGER array of length 5 that holds control parameters. Default values for the components may be set by a call to MC53I/ID. Details of the control parameters are given in Section 2.2.1. This argument is not altered by the routine.

INFO is an INTEGER array of length 9 which need not be set by the user. It is used to hold information about the execution of the subroutine. On exit from MC53A/AD, a value for INFO(1) of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3. For details of the information contained in the other components of INFO, see Section 2.2.2.

RINFO is a REAL (DOUBLE PRECISION in the D version) array of length 4 which need not be set by the user. It is used to hold information about the execution of the subroutine. For details of the information contained in the

components of RINFO, see Section 2.2.2.

2.2 Arrays for control and information

2.2.1 Control parameters

The elements of the array ICNTL control the action of MC53A/AD. Default values may be set by calling MC53I/ID.

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

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

ICNTL(3),ICNTL(4),ICNTL(5) are the three weights used in the element resequencing priority function (see Section 4 for details). The default values are 12, 6, and 1, respectively.

2.2.2 Information arrays

INFO(1) is used as an error and a warning flag. If a call to MC53A/AD is successful, on exit INFO(1) has value 0. A nonzero value of INFO(1) indicates an error has been detected or a warning issued (see Section 2.3).

INFO(2) holds, on exit from the final call to MC53A/AD, the number of components in the subdomain element connectivity graph. If INFO(2) > 1, it may be possible to improve the efficiency of the multiple front algorithm by partitioning the subdomain further so that the element connectivity graph for each subdomain of the original finite-element mesh has only one component. The numbers of elements in each component are held in the first INFO(2) entries of IW.

INFO(3) holds, on each exit from MC53A/AD, the largest integer used so far to index a variable.

INFO(4) holds, on exit from the final call to MC53A/AD, the maximum frontsize *fmax* in the subdomain for the original element order.

INFO(5) holds, on exit from the final call to MC53A/AD, the maximum frontsize *fmax* in the subdomain for the new element order. INFO(5) is not guaranteed to be smaller than INFO(4).

INFO(6) holds, on successful exit from the final call to MC53A/AD, the amount of workspace used (that is, the smallest value for LIW which will ensure success). If INFO(1) = –3, INFO(6) is set to a value which may suffice for LIW.

INFO(7) holds, on successful exit from the final call to MC53A/AD, the number of variables in the subdomain.

INFO(8) holds, on successful exit from the final call to MC53A/AD, the number of variables I which were ignored because IVAR(I) was out-of-range.

INFO(9) holds, on successful exit from the final call to MC53A/AD, the number of variables I which were ignored because they were duplicated.

RINFO(1) holds, on exit from the final call to MC53A/AD, the root mean-squared frontsize *frms* in the subdomain for the original element order.

RINFO(2) holds, on exit from the final call to MC53A/AD, the number of floating-point operations required by a frontal solver when applied to the subdomain using the original element order (assuming no stability restrictions).

RINFO(3) holds, on exit from the final call to MC53A/AD, the root mean-squared frontsize *frms* in the subdomain for the new element order. RINFO(3) may be larger than RINFO(1).

RINFO(4) holds, on exit from the final call to MC53A/AD, the number of floating-point operations required by a frontal solver when applied to the subdomain using the new element order (assuming no stability restrictions). RINFO(4) may be larger than RINFO(2) and, if this is the case, the user is advised to retain the original

ordering.

2.3 Error diagnostics

On successful completion, MC53A/AD will exit with the parameter INFO(1) set to 0. Other values for INFO(1) and the reasons for them are given below.

A negative value for INFO(1) is associated with a fatal error. If ICNTL(1) is greater than zero, a self-explanatory message is, in each case, output on unit ICNTL(1) (see Section 2.2.1). The negative values for INFO(1) are:

–1 NELT $\leq$ 0 on entry to MC53A/AD (this is only checked on the first call to MC53A/AD). Immediate return.

–2 Either NVAR $\leq$ 0 or IVAR(I) $\leq$ 0 for all variables I in the current element. Immediate return.

–3 Failure due to insufficient space allocated to the array IW. INFO(6) is set to a value which may suffice for LIW.

–4 ICALL = NELT+1 on a call to MC53A/AD which was not the NELT+1th call.

Warning messages are associated with INFO(1) $>$ 0. If ICNTL(2) is greater than zero, a self-explanatory message is, in each case, output on unit ICNTL(2) (see Section 2.2.1). The warnings are:

+1 The subdomain element connectivity graph has more than one component. The number of components is in INFO(2).

+2 Out-of-range variable indices or duplcated variable indices were entered. The number of out-of-range variables is in INFO(8) and the number of duplicates is in INFO(9).

+3 Warnings +1 and +2 are operative.

+4 Either the maximum wavefront or the r.m.s. wavefront or the number of floating-point operations required by a frontal solver for the element order generated by MC53A/AD is at least as large as for the original element order. The details are in INFO(4), INFO(5), and RINFO. The user may wish to retain the original order.

+5 Warnings +1 and +4 are operative.

+6 Warnings +2 and +4 are operative.

+7 Warnings +1, +2, and +4 are operative.

3 GENERAL INFORMATION

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

Use of common: None

Other routines called directly: Subroutines internal to the package are MC53B/BD, MC53C/CD, MC53D/DD, MC53E/ED, MC53F/FD, MC53G/GD, MC53H/HD, MC53J/JD, MC53K/KD, MC53L/LD and MC53M/MD. In addition, MC44A/AD, MC44B/BD, and KB07AI are called.

Input/output: Error messages on unit ICNTL(1) (ICNTL(1) = 0 suppresses them) and warnings on unit ICNTL(2) (ICNTL(2) = 0 suppresses them).

Restrictions:

NELT $>$ 1,

NVAR $\geq$ 1.

4 METHOD

MC44 is used to generate the element connectivity graph for the set of NELT+1 elements which comprises the NELT finite-elements in the subdomain and the guard element g supplied by the user. Using this element connectivity graph, the level structure $L(g)$ rooted at g is constructed and the distance $d(g,i)$ of each element i from g is computed. The starting element s for the element reordering is chosen so that $d(g,s)$ is a maximum.

The starting element s is relabelled as element one and a list of elements that are eligible to receive the next label is formed. At each stage in the relabelling process the list of eligible elements comprises those elements which are either adjacent to a element which has been relabelled or are adjacent to a element which is itself adjacent to a relabelled element. The next element to be given a new number is the element among all eligible elements with the highest priority, where the priority P_i of element i is defined to be

$$P_i = -W_1 * ngain(i) + W_2 * d(e,i) - W_3 * nadj(i). \tag{4.1}$$

Here W_1, W_2, and W_3 are weights which are held in the control parameters ICNTL(3), ICNTL(4), and ICNTL(5), respectively. The quantity $ngain(i)$ is the number of variables element i will introduce into the front less the number of variables that can then be eliminated, and $nadj(i)$ is the number of elements adjacent to element i which have not yet been relabelled.

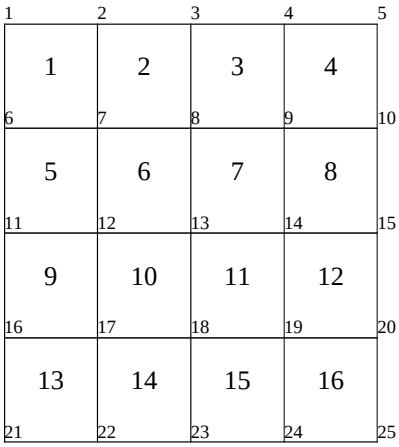
A full description of the algorithm is given in Scott (1995).

References

Scott, J. A. (1995). Element resequencing for use with a multiple front algorithm. Rutherford Appleton Laboratory Report RAL-95-029.

5 EXAMPLE OF USE

Consider the following finite-element mesh comprising sixteen 4-noded quadrilateral elements with one freedom per node.



Suppose the mesh is partitioned into 2 subdomains such that elements 1 to 8 form subdomain 1 and elements 9 to 16 form subdomain 2. The following program provides an example of the use of MC53 to reorder the elements in one of the subdomains.

```
C  Example to illustrate the use of MC53A.
C  The elements in a single subdomain are reordered.
C
C     .. Parameters ..
      INTEGER LVAR,LIW,LGUARD,MAXELT
```

```
      PARAMETER (LVAR=4,LIW=300,LGUARD=6,MAXELT=8)
C     ..
C     .. Local Scalars ..
      INTEGER I,ICALL,IELT,NELT,NGUARD,NVAR
C     ..
C     .. Local Arrays ..
      REAL RINFO(4)
      INTEGER ICNTL(5),IGUARD(LGUARD),INFO(9),IVAR(LVAR),IW(LIW),
     +        NORDER(MAXELT+1)
C     ..
C     .. External Subroutines ..
      EXTERNAL MC53A,MC53I
C     ..
C     .. Intrinsic Functions ..
      INTRINSIC INT
C     ..
C Call the initialisation routine
      CALL MC53I(ICNTL)

C   Read in the number of finite-elements in subdomain 1.
      READ (5,FMT=*) NELT
      IF (NELT.GT.MAXELT) THEN
         WRITE (6,FMT=*) ' NELT is too large.'
         GO TO 20
      END IF

C Call MC53A for each element (elements in any order)
      ICALL = 0
      DO 10 IELT = 1,NELT
         ICALL = ICALL + 1
         READ (5,FMT=*) NVAR
         IF (NVAR.GT.LVAR) THEN
            WRITE (6,FMT=*) ' NVAR is too large.'
            GO TO 20
         END IF
         READ (5,FMT=*) (IVAR(I),I=1,NVAR)

         CALL MC53A(ICALL,NELT,NVAR,IVAR,NORDER,LIW,IW,ICNTL,INFO,RINFO)

         IF (INFO(1).LT.0) THEN
            WRITE (6,FMT=*) ' Unexpected error return.'
            GO TO 20
         END IF
   10 CONTINUE

C   Read in the list of interface variables for subdomain 1
C   (the guard element)
      READ (5,FMT=*) NGUARD
      IF (NGUARD.GT.LGUARD) THEN
         WRITE (6,FMT=*) ' NGUARD is too large.'
         GO TO 20
      END IF
      READ (5,FMT=*) (IGUARD(I),I=1,NGUARD)

C Call MC53A for the guard element
      ICALL = NELT + 1
      CALL MC53A(ICALL,NELT,NGUARD,IGUARD,NORDER,LIW,IW,ICNTL,
     +           INFO,RINFO)

      WRITE (6,FMT='(A,I2)') ' Original max frontsize = ',INFO(4)
      WRITE (6,FMT='(A,I2)') ' New max frontsize      = ',INFO(5)
```

```
      WRITE (6,FMT='(/A,G8.3)') ' Original rms frontsize = ',RINFO(1)
      WRITE (6,FMT='(A,G8.3)') ' New rms frontsize      = ',RINFO(3)
      WRITE (6,FMT='(/A,I4)') ' Original flop count    = ',INT(RINFO(2))
      WRITE (6,FMT='(A,I4)') ' New flop count         = ',INT(RINFO(4))
C
   20 STOP

      END
```

The input data used for this problem is:

```
   8
   4
   6  11   7  12
   4
   1   2   6   7
   4
  14  15   9  10
   4
   8   9  13  14
   4
   7   8  12  13
   4
   4   5   9  10
   4
   2   8   7   3
   4
   3   9   8   4
   5
  11  12  13  14  15
```

This produces the following output:

```
 Original max frontsize = 12
 New max frontsize      =  7

 Original rms frontsize = 9.10
 New rms frontsize      = 6.34

 Original flop count    =  978
 New flop count         =  506
```