1 SUMMARY

This subroutine **sorts the sparsity pattern of a matrix to an ordering by columns.** There is an option for ordering the entries within each column by increasing row indices and an option for checking the user-supplied matrix entries for indices which are out of range or duplicated.

ATTRIBUTES — Version: 1.0.0. **Remark:** This subroutine supersedes `MC20` and `MC39`. **Types:** `MC49A`, `MC49AD`, `MC49AI`. **Original date:** June 1990. **Origin:** I.S. Duff and J.A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 The argument list

The single precision version

```
CALL MC49A(IND,NC,NR,NNZ,IRN,JCN,YESA,LA,A,LIP,IP,LIW,IW,IFLAG)
```

The double precision version

```
CALL MC49AD(IND,NC,NR,NNZ,IRN,JCN,YESA,LA,A,LIP,IP,LIW,IW,IFLAG)
```

The integer version

```
CALL MC49AI(IND,NC,NR,NNZ,IRN,JCN,YESA,LA,A,LIP,IP,LIW,IW,IFLAG)
```

`IND` is an `INTEGER` variable which must be set on entry to specify whether the user-supplied matrix entries are to be checked for errors and whether ordering by increasing row indices within each column is required. If `IND` is set to `1`, error checking is performed and the ordering within each column will be arbitrary. If `IND` is set to `2`, error checking is performed and the ordering within each column will be by increasing row indices. If `IND` is set to `-1`, only the scalar input data are checked for errors and the ordering within each column will be arbitrary. If `IND` is set to `-2`, only the scalar input data is checked for errors and the ordering within each column will be by increasing row indices. This argument is unchanged by the routine. **Restriction:** `IND=-2`, `-1`, `1`, or `2`.

`NC` is an `INTEGER` variable which must be set to the number of columns in the matrix. This argument is unchanged by the routine. **Restriction:** `NC` $\geq$ `1`.

`NR` is an `INTEGER` variable which must be set to the number of rows in the matrix. This argument is unchanged by the routine. **Restriction:** `NR` $\geq$ `1`.

`NNZ` is an `INTEGER` variable which must be set to the number of entries in the matrix. This argument is unchanged by the routine. **Restriction:** `NNZ` $\geq$ `1`.

`IRN` is an `INTEGER` array of length `NNZ` which must be set to contain the row indices of the entries in the matrix. The entries may be in any order. On exit, the row indices are reordered so that the entries of a single column are contiguous with column `J` preceding column `J+1` (`J=1,2,...,N-1`), with no space between columns. If `IND=-1` or `1`, the order within each column is arbitrary; if `IND=-2` or `2`, the order within each column is by increasing row indices.

`JCN` is an `INTEGER` array of length `NNZ`. On entry, `JCN(K)` must be set to the column index of the entry whose row index is held in `IRN(K)` (`K=1,2,...,NNZ`). The contents of this array are altered by the routine.

`YESA` is a `LOGICAL` variable which must be set by the user. If `YESA` = `.FALSE.`, only the sparsity pattern of the matrix will be ordered by columns. If `YESA` = `.TRUE.`, the numerical values of the entries of the matrix will also be ordered by columns. This argument is unchanged by the routine.

`LA` is an `INTEGER` variable which defines the length of the array `A`. This argument is unchanged by the routine.

Restrictions: If YESA = .TRUE., LA ≥ NNZ; if YESA = .FALSE., LA ≥ 1.

A is a REAL (DOUBLE PRECISION in the D version or INTEGER in the I version) array of length LA. If YESA = .TRUE., A(K) must be set by the user to hold the value of the entry with indices IRN(K) and JCN(K) (K=1,2,...,NNZ). On exit, the array will have been permuted in the same way as the array IRN. If YESA = .FALSE., the array is not accessed.

LIP is an INTEGER variable which defines the length of the array IP. This argument is unchanged by the routine. **Restrictions:** If IND=-1 or 1, LIP ≥ NC+1; if IND=-2 or 2, LIP ≥ MAX(NC, NR)+1.

IP is an INTEGER array of length LIP which need not be set on entry. On exit, IP(J) contains the position in the array IRN of the first entry in column J (J=1,2,...,NC), and IP(NC+1) is set to one greater than the number of entries in the matrix.

LIW is an INTEGER variable which defines the length of the array IW. This argument is unchanged by the routine. **Restrictions:** If IND=-1 or 1, LIW ≥ MAX(NC, NR)+1; if IND=-2 or 2, LIW ≥ NR+1.

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

IFLAG is an INTEGER variable which need not be set on entry. On exit, a negative value of IFLAG is used to signal a fatal error in the input data, a positive value of IFLAG is used to indicate that a warning has been issued, and a zero value is used to indicate a successful call to the routine. A positive value of IFLAG can only be returned if IND=1 or 2. Possible nonzero values of IFLAG and their consequences are as follows;

- –1 – The restriction IND=-2, -1, 1, or 2 has been violated. Immediate return with input parameters unchanged.
- –2 – One of the parameters NC, NR, NNZ is too small. Immediate return with input parameters unchanged.
- –3 – The parameter LA is too small. Immediate return with input parameters unchanged.
- –4 – The parameter LIW is too small. Immediate return with input parameters unchanged.
- –5 – The parameter LIP is too small. Immediate return with input parameters unchanged.
- +1 – One or more duplicated entries have been input. One copy of each duplicated entry is kept and, if YESA = .TRUE., the values of these entries are added together. Initially IDUP in COMMON is set to zero. If an entry appears *k* times, IDUP is incremented by *k*–1 and NZOUT in COMMON is set to the revised number of entries in the matrix. Note that this warning will overwrite an IFLAG=2 or an IFLAG=3 warning.
- +2 – One or more of the entries in IRN are out of range. These entries are removed by the routine. IOUT in COMMON is set to the number of entries which were out of range and NZOUT in COMMON is set to the revised number of entries in the matrix.
- +3 – One or more of the entries in JCN are out of range. These entries are removed by the routine. JOUT in COMMON is set to the number of entries which were out of range and NZOUT in COMMON is set to the revised number of entries in the matrix. Note that this warning will overwrite an IFLAG=2 warning.

2.2 Common blocks

One common block is used by MC49A/AD. The common block is:

The single precision version

```
COMMON/ MC49E/ LP, MP, IOUT, JOUT, IDUP, NZOUT
```

The double precision version

```
COMMON/ MC49ED/ LP, MP, IOUT, JOUT, IDUP, NZOUT
```

where the parameters are given default values by a block data subprogram MC49D/DD.

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 unchanged by the routine.

MP is an INTEGER variable used as the unit number of the device to which warning messages are sent. The default

value is 6. Warning messages can be suppressed by setting MP ≤ 0. This parameter is unchanged by the routine.

IOUT, JOUT, IDUP, and NZOUT are INTEGER variables. In case of a warning message, information on the number of entries supplied by the user which were out of range or were duplicated is placed in one or more of these variables. For details, see §2.1.

3 GENERAL INFORMATION

Use of common: MC49A/AD uses the common block MC49E/ED; see §2.2.

Workspace: An integer array IW of length LIW is used by MC49A/AD as workspace; see §2.1.

Other routines called directly: The routine MC49A/AD calls the internal subroutine MC49B/BD and, if IND=-2 or 2, the internal subroutine MC49C/CD. MC49A/AD also uses a block data subprogram MC49D/DD.

Input/output: Error messages on unit LP and warning messages on unit MP. These have default value 6. Printing of error and warning messages is suppressed by setting LP and MP equal to 0.

Restrictions: The routine MC49A/AD has the following restrictions:

IND=-2, -1, 1 or 2,
NC ≥ 1, NR ≥ 1, NNZ ≥ 1,
if YESA = .TRUE., LA ≥ NNZ,
if YESA = .FALSE., LA ≥ 1,
if IND=-1 or 1, LIP ≥ NC+1, and LIW ≥ MAX(NC, NR)+1,
if IND=-2 or 2, LIP ≥ MAX(NC, NR)+1, and LIW ≥ NR+1.

4 METHOD

MC49A/AD first checks the scalar input data for errors. A negative flag is set and control is immediately returned to the calling program if a fatal error is found. If the user chooses to check the matrix input data, a check is made for indices which are out of range. Any such entries are removed and a positive flag is set.

To sort the entries into column order, with arbitrary order within each column, the following procedure is used. The array JCN is scanned to count the number of entries in each column. Pointers are then set in IP to the positions of the leading entries of the columns in the sorted form and are copied to IW. The variable l is initialised to one. At an intermediate stage in the sort, the row indices of the first l–1 columns will be in their new positions and positions IP(l) to IW(l)–1 of IRN will hold row indices for column j (j=l,...,NC). The entry currently stored in position IW(l) of IRN is now taken to be the current entry and examined to see if it is in place. If not, it is put into the first free location in its correct column j, the pointer for IW(j) is increased by one, and the displaced entry is made the current entry. This chain of displacing entries continues until an entry which belongs to column l is found. This entry is stored in the position vacated by the first entry in the chain and IW(l) is increased by one. The next item in column l is then examined, unless column l has now been fully sorted, in which case column l+1 is considered. If the user requires the numerical values of the entries of the matrix to be sorted by columns then, at each step of the sort, the array A, which holds the numerical values, is permuted in the same way as the array IRN.

To sort the entries from arbitrary order to column order, with ordering by increasing row indices within each column, the above procedure is followed using rows in place of columns to obtain an ordering by rows, with arbitrary order within each row. The number of entries in each column is then obtained by a counting pass. The position of each entry is determined by examining each row in turn using a similar algorithm to that discussed above. The entries are then permuted into these positions.

Once the entries have been sorted into column order, if the user has chosen to include matrix error checking, a check is made for duplicates. The array IW is set to zero and each column is then checked in turn. If an entry in column J has row index I, IW(I) is set to J, unless IW(I) is already equal to J, in which case we have a duplicate. If duplicate entries are found, a positive flag is set and one copy of each duplicated entry is retained. If the numerical

values of the entries of the matrix are being sorted, the numerical values of the duplicate entries are summed.

Note that if the user has an ordering by columns, with arbitrary order within each column, and requires an ordering by columns with ordering by increasing row indices within each column, the user should should use the column pointers to set up an array JCN of column indices and call MC49A/AD with IND=−2 or 2.

5 EXAMPLE OF USE

As a simple example, suppose we wish to order the sparse matrix

$$\begin{pmatrix} 1.1 & 1.2 & & 1.4 \\ & 2.2 & & \\ 3.1 & & 3.3 & 3.4 \\ & & & 4.4 \\ & 5.2 & & \end{pmatrix}$$

by columns, with the entries ordered by increasing row indices within each column. Using the following program:

```
      INTEGER          MAXN, MAXNZ
      PARAMETER       (MAXN=10, MAXNZ=30)
      INTEGER          IND,NC,NR,NNZ,LA,LIP,LIW,IFLAG,J,K,K1,K2
      INTEGER          IRN(MAXNZ), JCN(MAXNZ), IW(MAXN+1), IP(MAXN+1)
      DOUBLE PRECISION A(MAXNZ)
      LOGICAL          YESA
      INTEGER          LP,MP,IOUT,JOUT,IDUP,NZOUT
      COMMON/MC49ED/   LP,MP,IOUT,JOUT,IDUP,NZOUT
      EXTERNAL         MC49AD
      IND=2
      YESA=.TRUE.
      READ(5,*) NR,NC,NNZ
      DO 10 K=1,NNZ
         READ(5,*) IRN(K), JCN(K), A(K)
   10 CONTINUE
      LA=NNZ
      LIP=MAX(NC,NR)+1
      LIW=NR+1
      CALL MC49AD(IND,NC,NR,NNZ,IRN,JCN,YESA,LA,A,LIP,IP,LIW,IW,IFLAG)
      WRITE(6,9000) IFLAG
      IF(IFLAG.LT.0) STOP
      DO 30 J=1,NC
         K1=IP(J)
         K2=IP(J+1)-1
         IF(K1 .LE. K2) THEN
            WRITE(6,9010) J
            DO 20 K=K1,K2
               WRITE(6,9020) IRN(K), A(K)
   20       CONTINUE
         END IF
   30 CONTINUE
 9000 FORMAT( '  IFLAG = ',I2, ' ON EXIT FROM MC49AD ')
 9010 FORMAT( '  COLUMN ', I2)
 9020 FORMAT( '     ROW ', I2, ' VALUE ', F4.1)
      STOP
      END
```

on the data

```
 5   4   9
 1   1   1.1
 3   3   3.3
 3   4   3.4
 3   1   3.1
 1   2   1.2
 1   4   1.4
 2   2   2.2
 4   4   4.4
 5   2   5.2
```

produces the output:

```
IFLAG =  0 ON EXIT FROM MC49AD
COLUMN  1
   ROW  1 VALUE  1.1
   ROW  3 VALUE  3.1
COLUMN  2
   ROW  1 VALUE  1.2
   ROW  2 VALUE  2.2
   ROW  5 VALUE  5.2
COLUMN  3
   ROW  3 VALUE  3.3
COLUMN  4
   ROW  1 VALUE  1.4
   ROW  3 VALUE  3.4
   ROW  4 VALUE  4.4
```