1 SUMMARY

Permutes a sparse matrix to a block lower triangular form, i.e. given a sparse matrix $\mathbf{A}=\{a_{ij}\}_{n\times n}$ it attempts to find permutation matrices **P** and **Q** such that $\tilde{\mathbf{A}}=\mathbf{PAQ}$ is block lower triangular and returns $\tilde{\mathbf{A}}$, **P**, and **Q** to the caller.

The code is described by I.S.Duff, Harwell report R.8730 (1977).

ATTRIBUTES — **Version:** 1.0.0. (12 July 2004) **Types:** Real (single, double), Complex (single, double). **Calls:** `MC13D`, `MC21A`. **Original date:** April 2001. **Remark:** The `MC25` entries are threadsafe versions of `MC23` and `ME23`. **Origin:** I. S. Duff, Harwell.

2 HOW TO USE THE PACKAGE

2.1 Initialization

The `MC25I/ID/IC/IZ` entry must be called prior to the first call to the `MC25A/AD/AC/AZ` entry to initialize the control array.

The single precision version

```
CALL MC25I(ICNTL)
```

The double precision version

```
CALL MC25ID(ICNTL)
```

The single precision complex version

```
CALL MC25IC(ICNTL)
```

The double precision complex version

```
CALL MC25IZ(ICNTL)
```

`ICNTL` is an `INTEGER` array of length 10, see Section 2.6.

2.2 Argument list

The single precision version

```
CALL MC25A(N,ICN,A,LICN,LENR,IDISP,IP,IQ,LENOFF,IW,IW1,ICNTL,INFO)
```

The double precision version

```
CALL MC25AD(N,ICN,A,LICN,LENR,IDISP,IP,IQ,LENOFF,IW,IW1,ICNTL,INFO)
```

The single precision complex version

```
CALL MC25AC(N,ICN,A,LICN,LENR,IDISP,IP,IQ,LENOFF,IW,IW1,ICNTL,INFO)
```

The double precision complex version

```
CALL MC25AZ(N,ICN,A,LICN,LENR,IDISP,IP,IQ,LENOFF,IW,IW1,ICNTL,INFO)
```

`N` is an `INTEGER` variable which must be set by the user to the order of the matrix. It is not altered by `MC25`.

`ICN` is an `INTEGER` array of length `LICN` which must be set by the user to contain the column indices of the nonzeros. Those belonging to a single row must be contiguous but the ordering of column indices within each row is unimportant. The nonzeros of row `I` precede those of row `I+1`, `I=1`, `...`, `N-1` and no wasted space is allowed between the rows. On a successful output, information is held in positions 1 to `IDISP(1)-1` and

IDISP(2) to LICN of array ICN. The permuted column indices of the entries of the rows in the diagonal part are held in IDISP(2) to LICN, with rows in permuted order, and no spaces wasted between the rows. If there is more than one diagonal block, the original column indices of the parts of the rows in off-diagonal blocks will be found in original row order in positions 1 to IDISP(1)-1, again with no wasted space between rows.

A is an array of length LICN whose elements must be set by the user to the values of the nonzero elements of the matrix. Its type depends on the entry: for MC25A it is REAL, for MC25AD it is DOUBLE PRECISION, for MC25AC it is COMPLEX and for MC25AZ it is COMPLEX*16. The the nonzero elements must correspond to the positions given in ICN then on output the contents of A will have been reordered to correspond with ICN.

LICN is an INTEGER variable which must be set by the user to the length of arrays ICN and A. It should be at least NZ+N where NZ is the number of entries in the matrix. It is not altered by MC25A.

LENR is an INTEGER array of length N. The user must set LENR(I) equal to the number of nonzeros in row I, I=1,2,..., N. On output from MC25, LENR(I) I=1,2,..., N is equal to the number of nonzeros in the part of row I (of the permuted matrix) in its diagonal block.

IDISP is an INTEGER array of length 2. On output from MC25, IDISP(1) is the position in arrays ICN and A of the first location after the off-diagonal blocks (equal to one if there are no blocks) while IDISP(2) is the position in ICN and A of the first entry in the diagonal blocks.

IP is an INTEGER array of length N. On output from MC25, |IP(I)|, I=1,..., N is the original row index of the row which is I-th in the permuted form. The last row in the permuted form of each diagonal block (except the last) is indicated by a negative value for IP.

IQ is an INTEGER array of length N. On output from MC25, IQ(I), I=1,2,..., N holds the original column index of the column which is I-th in the permuted form.

LENOFF is an INTEGER array of length N. On output from MC25, LENOFF(I), I=1,2,..., N holds the number of nonzeros in the part of row I in the off-diagonal blocks, reference being to the original row number. However, if the matrix has only one diagonal block LENOFF(1) is set to –1, the other entries being unimportant.

IW is an INTEGER array of length 5*N and is used as workspace by MC25.

IW1 is an INTEGER array of length 2*N which is used as workspace by MC25.

ICNTL is an INTEGER array of length 10 whose first two elements can optionally be set by the user, see Section 2.6.

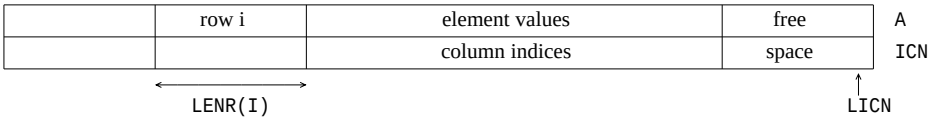
INFO is an INTEGER array of length 10 which is used to return information to the user, see Section 2.6.

2.3 Parameter usage summary

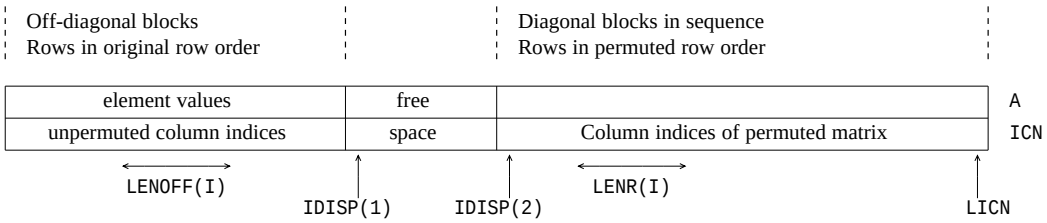
Input: N, ICN, A, LICN, LENR
Unchanged by MC25: N, LICN
Work array: IW(5*N), IW1(2*N)
Output: ICN, A, LENR, IDISP, IP, IQ, LENOFF

2.4 Data structures summary

2.4.1 Input to MC25



2.4.2 Output from MC25



2.5 Error returns

There are two possible error returns indicated by a negative value of INFO(1):

–1 If ICNTL(2) $\neq$ 0 and it is impossible to permute the matrix so that the diagonal of the permuted matrix is free from zeros, then the matrix is structurally singular (and *a fortiori* numerically singular). The subroutine terminates immediately, setting INFO(2) to the value of INFO(3) and printing to the output unit given by ICNTL(1) the message:

```
ERROR RETURN FROM MC25A BECAUSE MATRIX IS STRUCTURALLY SINGULAR,RANK = r
```

where *r* is the calculated value of the structural rank.

–2 There is insufficient space in A/ICN for the subroutine to perform its reordering. The subroutine does not require much elbow room needing a value of LICN equal to at most NZ+N, where NZ is the number of entries in the matrix. If the user has allowed insufficient space the message:

```
ERROR RETURN FROM MC25A BECAUSE LICN NOT BIG ENOUGH INCREASE BY k
```

is output on unit ICNTL(1), where *k* is the number of words by which the arrays ICN and A should be increased (see Section 2.6). INFO(2) is returned set to the value of LICN.

2.6 The control and information arrays

The ICNTL array argument which must be of length 10 can be used to pass optional control values to the routine.

ICNTL(1) specifies the unit number to be used to output messages as described in Section 2.5. It has a default value of 6 which can be reset by the user to another unit or set negative if messages are to be suppressed.

ICNTL(2) is used by the routine to decide if it should stop its reordering when the matrix is structurally singular. The reordering is stopped if ICNTL(2) is nonzero (set to 1 say) otherwise the reordering is continued.

ICNTL(3) to ICNTL(10) should not be altered and at present are not used by MC25.

The INFO array argument which must be of length 10 is used to return information to the user.

INFO(1) is used as an error indicator. It is set to zero when there are no errors otherwise it is set to a negative value, see Section 2.5.

INFO(2) is used to return a supplementary value associated with the error indicated by INFO(1).

INFO(3) is set by MC25 to the number of nonzeros of **A** which can be permuted onto the diagonal of the matrix. If INFO(3) does not equal N, the matrix is structurally singular.

INFO(4) is set by MC25 to the number of diagonal blocks present in the permuted form.

INFO(5) is set by MC25 to the order of the largest diagonal block in the permuted form.

INFO(6) to INFO(10) are not used at present by MC25.

3 GENERAL INFORMATION

Workspace: see Section 2.3

Use of common: none.

Other routines called directly: MC13D, MC13E, MC21A and MC21B.

Input/output: error messages may be printed (see Section 2.5).

Restrictions: none.

4 METHOD

Subroutine MC25 is really a data handling routine. The real work is done by subroutine MC21 which finds a column permutation to make the diagonal zero free and also by MC13 which finds a subsequent symmetric permutation to put the matrix into block lower triangular form.

5 EXAMPLE OF USE

An example to reduce a matrix to block triangular form

In the example code shown below, we read in the entries of a sparse matrix in any order, the input is then sorted into row order. We then call the routine MC25AD to attempt to reduce the input matrix into block triangular form.

```
      INTEGER MAXN, MAXNZ, LICN
      PARAMETER( MAXN = 50 , MAXNZ = MAXN*MAXN ,
     +           LICN = 4*MAXNZ )
      DOUBLE PRECISION A(LICN)
      INTEGER ICN(LICN), JNUM(MAXNZ), JPTR(MAXN+1), N, NZ, I, J,
     +        LENR(MAXN), IDISP(2), IP(MAXN), IQ(MAXN),
     +        LENOFF(MAXN), IW(5*MAXN), IW1(2*MAXN), ICNTL(10),
     +        INFO(10), JDISP, IPOS, ICNT59(10), INFO59(10)
      READ(5,*) N , NZ
      IF ( (N .GT. MAXN) .OR. (NZ .GT. MAXNZ) ) THEN
         WRITE(6,550)
         STOP
      END IF
      READ(5, * ) ( ICN(I) , JNUM(I) , A(I) , I=1,NZ )
      JDISP = 0
C     SORT THE INPUT MATRIX INTO ROW ORDER (MC25 REQUIRES THIS)
      ICNT59(1) = 1
      ICNT59(2) = 0
      ICNT59(3) = 0
      ICNT59(4) = -1
      ICNT59(5) = -1
      ICNT59(6) = 0
      CALL MC59AD( ICNT59, N, N, NZ, JNUM, LICN, ICN, LICN, A, N+1,
     +             JPTR, N+1, IW, INFO59)
C     JPTR(I) POINTS TO THE START OF THE I'TH ROW OF THE MATRIX
C     LENR(I)  HOLDS THE NUMBER OF ENTRIES IN THE I'TH ROW
      DO 100 I=1,NZ
        ICN(I) = JNUM(I)
  100 CONTINUE
      DO 150 I=1,N-1
        LENR(I) = JPTR(I+1) - JPTR(I)
  150 CONTINUE
      LENR(N) = NZ - JPTR(N) + 1
      CALL MC25ID( ICNTL )
      CALL MC25AD( N, ICN, A, LICN, LENR, IDISP, IP,
     +            IQ, LENOFF, IW, IW1, ICNTL, INFO )
      IF ( IDISP(1) .EQ. -1 ) THEN
```

```
         WRITE(6,590)
         STOP
      END IF
      IF ( LENOFF(1) .EQ. -1 ) THEN
C        WE ONLY HAVE ONE DIAGONAL BLOCK
         WRITE(6,610)
         WRITE(6,570)
         IPOS = IDISP(2)
         DO 250 I = 1 , N
           DO 200 J = IPOS , IPOS + LENR(I) - 1
             WRITE(6,600) I , ICN(J) , A(J)
 200       CONTINUE
         IPOS = IPOS + LENR(I)
 250     CONTINUE
      ELSE
         WRITE(6,630)
         WRITE(6,570)
C        OFF-DIAGONAL BLOCKS FIRST
         IPOS = 1
         DO 350 I = 1 , N
           DO 300 J = IPOS , IPOS + LENOFF(I) - 1
             WRITE(6,600) I , ICN(J) , A(J)
 300       CONTINUE
         IPOS = IPOS + LENOFF(I)
 350     CONTINUE
         WRITE(6,640)
         WRITE(6,650)
         WRITE(6,570)
C        NOW DIAGONAL BLOCKS
         IPOS = IDISP(2)
         DO 450 I = 1 , N
           DO 400 J = IPOS , IPOS + LENR(I) - 1
             WRITE(6,600) I , ICN(J) , A(J)
 400       CONTINUE
         IPOS = IPOS + LENR(I)
 450     CONTINUE
      END IF
C     NOW WRITE OUT THE PERMUTATION MATRICES
      WRITE(6,670)
      WRITE(6,690) ( IP(I) , IQ(I) , I = 1 , N )
 550  FORMAT( / ' ERROR IN THE INPUT DATA FOR N AND/OR NZ'//)
 570  FORMAT( / '      ROW   COLUMN          ENTRY' / )
 590  FORMAT( / ' **** ERROR RETURN FROM MC25AD ****' / )
 600  FORMAT( 2I8 , 8X , F13.5 )
 610  FORMAT( / ' THE PERMUTED MATRIX HAS ONLY ONE DIAGONAL BLOCK',
     +          ' WHICH IS:' / )
 630  FORMAT( / ' THE PERMUTED MATRIX IS-' //
     +          ' OFF-DIAGONAL BLOCKS:' / )
 640  FORMAT(/ ' THE ROW AND COLUMN INDICES ABOVE ARE FOR THE POSITIONS'
     +       / ' IN THE ORIGINAL MATRIX, AND NOT THE PERMUTED MATRIX' /)
 650  FORMAT( / ' DIAGONAL BLOCKS:' / )
 670  FORMAT( / ' THE PERMUTATION ARRAYS ARE:' /
     +        / '     IP()    IQ() ' / )
 690  FORMAT( 2I8 )
      STOP
      END
```

For the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 0 & 7 & 8 \\ 9 & 0 & 0 & 12 \\ 0 & 14 & 0 & 16 \end{pmatrix}$$

we could have as input

```
 4        8
 4        4           16.
 1        1            1.
 4        2           14.
 2        3            7.
 3        1            9.
 2        4            8.
 3        4           12.
 1        4            4.
```

and we would get the following output

```
THE PERMUTED MATRIX IS-

OFF-DIAGONAL BLOCKS:


      ROW   COLUMN          ENTRY

       2       4             8.00000
       4       4            16.00000

THE ROW AND COLUMN INDICES ABOVE ARE FOR THE POSITIONS
IN THE ORIGINAL MATRIX, AND NOT THE PERMUTED MATRIX


DIAGONAL BLOCKS:


      ROW   COLUMN          ENTRY

       1       1             4.00000
       1       2             1.00000
       2       1            12.00000
       2       2             9.00000
       3       3            14.00000
       4       4             7.00000

THE PERMUTATION ARRAYS ARE:

     IP()    IQ()

       1       4
      -3       1
      -4       2
       2       3
```

We should point out that the entries for the off-diagonal blocks are given in terms of the original unpermuted matrix. To obtain the positions of these entries in the permuted matrix the user must make use of the information in the permutation arrays `IP()` and `IQ()`. For example the entry `(2,4)` whose value is `4.` in the original matrix is now the entry `(4,1)` in the permuted matrix.

The arrays `IP()` and `IQ()` represent the following permutation matrices

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

and the resulting triangularised matrix $\tilde{\mathbf{A}}$ is

$$\tilde{\mathbf{A}} = \begin{pmatrix} 4 & 1 & 0 & 0 \\ 12 & 9 & 0 & 0 \\ 16 & 0 & 14 & 0 \\ 8 & 0 & 0 & 7 \end{pmatrix}$$