1 SUMMARY

To **estimate the rank and find a nonsingular submatrix** of a sparse matrix **A** with *m* rows and *n* columns using Gaussian elimination. The main entry performs a sparse LU factorization of the matrix optionally using rook pivoting. The factors are not returned.

ATTRIBUTES — **Version:** 1.0.0. (21 July 2007) **Types:** Real (single, double). **Calls:** `BLAS` routines: `I_AMAX`,`_AXPY`, `_GEMM`, `_GEMV`, `_SCAL`, `_SWAP`, `_TRSM`, and `_TRSV`. **Origin:** I.S. Duff, Rutherford Appleton Laboratory. **Original date:** November 2007.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequences

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

(a) `MC58I/ID` can be called to set default values for the control parameters in `ICNTL` and `CNTL.` It would normally be called once prior to any calls of `MC58A/AD`. If the user wishes to set any control parameter to a value other than the default, this should be done explicitly after the call to `MC58I/ID`.

(b) `MC58A/AD` factorizes the matrix using a pivotal strategy designed to compromise between maintaining sparsity and controlling loss of accuracy through roundoff. There is an option for using rook pivoting for better numerical robustness. The factors themselves are not returned.

In `MC58I` and `MC58A`, all `REAL`s are default reals. In `MC58ID` and `MC58AD`, all `REAL`s are default double precision.

2.1.1 To set default values of controlling parameters

`MC58I/ID` sets default values for the components of the arrays that hold control parameters for the `MC58` package.

The single precision version

```
CALL MC58I(CNTL,ICNTL)
```

The double precision version

```
CALL MC58ID(CNTL,ICNTL)
```

`CNTL` is a `REAL` array of length 10 that need not be set by the user. On return it contains default values (see Section 2.2 for details).

`ICNTL` is an `INTEGER` array of length 20 that need not be set by the user. On return it contains default values (see Section 2.2 for details).

2.1.2 To estimate the rank and find a nonsingular submatrix.

The single precision version

```
CALL MC58A(M,N,NE,LA,A,LIRN,IRN,LJCN,JCN,CNTL,ICNTL,LIW,IW,
           INFO,RINFO,RANK,ROWS,COLS)
```

The double precision version

```
CALL MC58AD(M,N,NE,LA,A,LIRN,IRN,LJCN,JCN,CNTL,ICNTL,LIW,IW,
            INFO,RINFO,RANK,ROWS,COLS)
```

`M` is an `INTEGER` variable that must be set by the user to the number *m* of rows in the matrix **A**. `M` is not altered by the subroutine. **Restriction:** $M \geq 1$.

N is an INTEGER variable that must be set by the user to the number *n* of columns in the matrix **A**. N is not altered by the subroutine. **Restriction:** N ≥ 1.

NE is an INTEGER variable that must be set by the user to the number of entries in the matrix **A**. NE is not altered by the subroutine. **Restriction:** NE ≥ 1.

LA is an INTEGER variable that must be set by the user to the length of array A. Usually a sufficient value is 3*NE although this value is very problem dependent. If it is too small, a suggested value is returned in INFO(3) (see Section 2.2). After a successful call, the smallest value that would suffice is returned in INFO(3). LA is not altered by the subroutine. **Restriction:** LA ≥ 2*NE.

A is a REAL array of length LA. A(k), k=1, ..., NE must be set by the user to hold the values of the matrix entries. They may be in any order. If there is more than one entry for a particular position, the values are summed. The number of such multiple entries is returned in INFO(7) (see Section 2.2). A is used as workspace by the routine.

LIRN is an INTEGER variable that must be set by the user to the length of array IRN. Usually a sufficient value is 3*NE although this value is very problem dependent. If it is too small, a suggested value is returned in INFO(4) (see Section 2.2). After a successful call, the smallest value that would suffice is returned in INFO(4). LIRN is not altered by the subroutine. **Restriction:** LIRN ≥ 2*NE.

IRN is an INTEGER array of length LIRN. IRN(k) must be set by the user to hold the row index of the entry stored in A(k), k=1, ..., NE. Any entry with an out-of-range index (less than 1 or greater than M) is ignored and the total number detected is returned in INFO(8). IRN is used as workspace by the routine.

LJCN is an INTEGER variable that must be set by the user to the length of array JCN. Usually a sufficient value is 2*NE although this value is very problem dependent. If it is too small, a suggested value is returned in INFO(5) (see Section 2.2). After a successful call, the smallest value that would suffice is returned in INFO(5). LJCN is not altered by the subroutine. **Restriction:** LJCN ≥ NE.

JCN is an INTEGER array of length LJCN. JCN(k) must be set by the user to hold the column index of the entry stored in A(k), k=1, ..., NE. Any entry with an out-of-range index (less than 1 or greater than N) is ignored and the total number detected is returned in INFO(8). JCN is used as workspace by the routine.

CNTL is a REAL array of length 10 that contains control parameters and must be set by the user. Default values for the components may be set by a call to MC58I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

ICNTL is an INTEGER array of length 20 that contains control parameters and must be set by the user. Default values for the components may be set by a call to MC58I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

LIW is an INTEGER variable that must be set by the user to the length of array IW. **Restriction:** LIW ≥ 6*(M+N)+MAX(M,N) unless ICNTL(5)≠0,*when*LIW ≥ 4*M+5*N+MAX(M,N)·

IW is an INTEGER array of length LIW that need not be set by the user and is used as workspace.

INFO is an INTEGER array of length 20 that need not be set by the user. It contains information about the execution of the subroutine. On exit from MC58A/AD, a value for INFO(1) of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3.1. For details of the information output in the other components, see Section 2.2.

RINFO is a REAL array of length 10 that need not be set by the user. On output, it holds information about the execution of the subroutine, see Section 2.2.

RANK is an INTEGER variable that need not be set by the user. On exit from MC58A/AD, RANK will be set to an estimate of the rank of the matrix.

ROWS is an INTEGER array of length M that need not be set by the user. On exit from MC58A/AD, ROWS(I), I=1, ...RANK will hold the indices of the rows of a nonsingular submatrix.

COLS is an INTEGER array of length N that need not be set by the user. On exit from MC58A/AD, COLS(I), I=1,... RANK will hold the indices of the columns of a nonsingular submatrix.

2.2 Arrays for control and information

The elements of the arrays CNTL and ICNTL control the action of MC58A/AD, Default values for the elements are set by MC58I/ID. The elements of the array INFO provide information on the action of MC58A/AD.

CNTL(1) has default value 0.4 and is used to control the switch from sparse to full matrix processing when factorizing the matrix. The switch is made when the ratio of number of entries in the reduced matrix to the number that it would have as a full matrix is greater than CNTL(1). If CNTL(1) is greater than 1.0, the routine will not switch to full matrix processing.

CNTL(2) has default value 0.01 and is used for threshold pivoting. Values near zero emphasize sparsity and values near one emphasize stability. If CNTL(2) < 0.0, it is regarded as having the value 0.0; if CNTL(2) > 1.0, it is regarded as having the value 1.0.

CNTL(3) has default value 10^{-12}. If it is set to a positive value, MC58A/AD will treat any pivot whose modulus is less than CNTL(3)*RINFO(2) as zero. If the matrix is rectangular or rank deficient, it is possible that entries with modulus less than CNTL(3)*RINFO(2) are dropped from the factorization.

Other components of CNTL are not presently used by the code but may be in future releases.

ICNTL(1) has default value 6 and holds the unit number to which the error messages are sent. A non-positive value suppresses all such messages.

ICNTL(2) has default value 6 and holds the unit number to which the warning messages are sent. A non-positive value suppresses all such messages.

ICNTL(3) has default value 6 and holds the unit number to which diagnostic printing is sent. A non-positive value suppresses all such printing.

ICNTL(4) is used by the subroutines to control printing of error, warning, and diagnostic messages. It has default value 2. Possible values are:

<1	No messages output.
1	Only error messages are output.
2	Error and warning messages output.
3	As for 2, plus scalar parameters and a few entries of array parameters on entry and exit from MC58A/AD.
≥4	As for 2, plus all parameter values on entry and exit from MC58A/AD.

ICNTL(5) has default value 0 that is used to control the search for pivots. If ICNTL(5) is left at its default value then the reduced matrix is searched by rows and columns to choose a pivot. If it is is set to any other value, then the pivot search is only conducted by columns.

ICNTL(6) has default value 3 that is used to control the search for pivots. If ICNTL(6) has a positive value, each pivot search is limited to a maximum of ICNTL(6) rows and columns (ICNTL(5)=0) or rows (ICNTL(5)≠0). If ICNTL(6) is set to the value 0, then the reduced matrix will be searched until the best pivot is found.

ICNTL(7) is used to control the full-matrix factorization. It has default value 32. Possible values are:

0	Level 1 BLAS used.
1	Level 2 BLAS used.
≥2	Level 3 BLAS used, with block column size ICNTL(7).

ICNTL(8) has default value 1. If ICNTL(8) is left at its default value, rook pivoting will be performed. Any other value will use threshold pivoting by columns. This should result in the code running faster but at the risk of a less accurate estimate.

Other components of `ICNTL` are not presently used by the code but may be in future releases.

`INFO(1)` has the value zero if the call was successful, positive in the case of a warning, and a negative value in the event of an error (see Section 2.3).

`INFO(2)` is used by `MC58A/AD` to give more information in the case of an error return with `INFO(1) < 0`.

`INFO(3)` is used by `MC58A/AD` to monitor the adequacy of the allocated space in arrays `A`, `IRN`, and `JCN`, by counting the number of garbage collections performed on these arrays. If `INFO(3)` is fairly large (say greater than 10), it may be advantageous to increase the size of the arrays. If `INFO(3)` is small or zero, then one can perhaps save storage by reducing the size of `LA` on a subsequent run of `MC58A/AD` with the same data.

`INFO(4)` indicates, after a successful call to `MC58A/AD`, the minimum length to which `LA` could be reduced while still permitting a successful call for the given matrix. If, however, the user were to decrease the length of array `A` to that size, the number of garbage collections (`INFO(3)`) may be very high. In the event of failure because `LA` is too small (`INFO(1)=–4`), `INFO(4)` gives a minimum size for `LA` that permits the code to proceed further.

`INFO(5)` indicates, after a successful call to `MC58A/AD`, the minimum length to which `IRN` could be reduced while still permitting a successful call for the given matrix. If, however, the user were to decrease the length of array `IRN` to that size, the number of garbage collections (`INFO(3)`) may be very high. In the event of failure because `IRN` is too small (`INFO(1)=–5`), `INFO(5)` gives a minimum size for `IRN` that permits the code to proceed further.

`INFO(6)` indicates, after a successful call to `MC58A/AD`, the minimum length to which `JCN` could be reduced while still permitting a successful call for the given matrix. If, however, the user were to decrease the length of array `JCN` to that size, the number of garbage collections (`INFO(3)`) may be very high. In the event of failure because `JCN` is too small (`INFO(1)=–6`), `INFO(6)` gives a minimum size for `JCN` that permits the code to proceed further.

`INFO(7)` is set to minimum value required for `LIW`.

`INFO(8)` holds, on exit from `MC58A/AD`, the number of multiple entries in the input matrix. Such entries are summed.

`INFO(9)` holds, on exit from `MC58A/AD`, the number of entries with out-of-range indices. Each such entry is ignored.

`INFO(10)` holds, on exit from `MC58A/AD`, the number of rows in the full block.

`INFO(11)` holds, on exit from `MC58A/AD`, the number of columns in the full block.

`INFO(12)` holds, on exit from `MC58A/AD`, the estimated rank of the sparse part of the matrix.

`INFO(13)` holds, on exit from `MC58A/AD`, the estimated rank of the full block.

`INFO(14)` holds, on exit from `MC58A/AD`, the number of rows considered to be zero.

`INFO(15)` holds, on exit from `MC58A/AD`, the number of null columns encountered.

`INFO(16)` holds, on exit from `MC58A/AD`, the number of columns considered to be zero.

Note that entries 9 to 15 may not be of interest to the average user. Other components of `INFO` are not presently used by the code but may be in future releases.

`RINFO(1)` holds, on exit from `MC58A/AD`, the number of floating-point operations required to factorize the matrix.

`RINFO(2)` holds, on exit from `MC58A/AD`, the modulus of the largest entry in the matrix.

Other components of `RINFO` are not presently used by the code but may be in future releases.

2.3 Error diagnostics

2.3.1 Error diagnostics for MC58A/AD

If the subroutine encounters no errors or complications, the value of `INFO(1)` will be zero on exit from `MC58A/AD`. The errors and complications that can arise are as follows:

–1 `M` has a value less than `1`. `INFO(2)` holds the value of `M`.

–2 `N` has a value less than `1`. `INFO(2)` holds the value of `N`.

–3 `NE` has a value less than `1`. `INFO(2)` holds the value of `NE`.

–4 `LA` is too small. `INFO(3)` gives a minimum value that will permit the code to proceed further. `INFO(2)` holds the value of `LA`. Note that, if both `LA` and `LIRN` are too small, the error return will be `INFO(1)` = –4.

–5 `LIRN` is too small. `INFO(4)` gives a minimum value that will permit the code to proceed further. `INFO(2)` holds the value of `LIRN`.

–6 `LJCN` is too small. `INFO(5)` gives a minimum value that will permit the code to proceed further. `INFO(2)` holds the value of `LJCN`.

–7 `LIW` is too small. Must be increased to at least the value returned in `INFO(7)`. `INFO(2)` holds the value of `LIW`.

+1 One or more row or columns indices are out of range. The first 10 are optionally printed on unit `ICNTL(2)`. (See `INFO(8)` in Section 2.2).

+2 One or more entries are for the same position in the matrix. (See `INFO(7)` in Section 2.2).

+3 Both out of range and duplicates are present in the original matrix.

2.4 Badly-scaled systems

If the user's input matrix has entries differing widely in magnitude, then an inaccurate rank may be obtained. In such cases, the user is advised to explicitly scale the matrix using `MC29A/AD`, `MC64A/AD`, or `MC77A/AD` prior to calling `MC58A/AD`.

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly: The internal routines: `MC58B/BD`, `MC58C/CD`, `MC58D/DD`, `MC58E/ED`, `MC58F/FD`, `MC58G/GD` and the BLAS routines: `ISAMAX/IDAMAX`, `SAXPY/DAXPY`, `SGEMM/DGEMM`, `SGEMV/DGEMV`, `SSCAL/DSCAL`, `SSWAP/DSWAP`, `STRSM/DTRSM`, and `STRSV/DTRSV`.

Input/output: Error, warning and diagnostic messages only. Error and warning messages on unit `ICNTL(1)`, diagnostic messages on unit `ICNTL(2)`. Both have default value 6, and output is suppressed if they are not positive.

Restrictions:

`M`$\geq$`1`, `N`$\geq$`1`, `NE`$\geq$`1`,
`LA`$\geq$`2*NE`
`LIRN`$\geq$`2*NE`
`LJCN`$\geq$`NE`
`LIW`$\geq$`6*(M+N)+MAX(M,N)` *or* `LIW`$\geq$`4*M+5*N+MAX(M,N)` *when* `ICNTL(5)`$\neq$0

4 METHOD

MC58A/AD reorders the input data to hold the entries by columns, removes entries with out-of-range indices, sums duplicates, and calls MC58B/BD to factorize the matrix to estimate its rank and a nonsingular submatrix. MC58B/BD uses a sparse variant of Gaussian elimination to compute a pivot ordering for the LU factorization of **A**. It uses pivoting to preserve sparsity in the factors and requires each pivot $a_{pj}^{(k)}$ to satisfy the stability test

$$|a_{pj}^{(k)}| \geq u \max_i |a_{ij}^{(k)}|$$

within the reduced matrix $A^{(k)}$, where u is the threshold held in CNTL(2), with default value 0.01.
If the rook pivoting option (ICNTL(7) = 1) is used then each pivot will also satisfy the test

$$|a_{pj}^{(k)}| \geq u \max_k |a_{pk}^{(k)}|.$$

The code was derived from the HSL packages MA48/MA50 but it does not return a factorized matrix. It also drops small entries during the factorization and will usually be very much faster than using MA48 on the same matrix, sometimes by two or more orders of magnitude.

A discussion of the use of these subroutines in determining the rank of matrices in limit analysis calculations in geotechnical engineering is given by Duff, Lyamin, and Scott (to appear).

5 EXAMPLE OF USE

Estimating the rank and find a nonsingular submatrix.

In the example code shown below, we estimate the rank and determine a nonsingular submatrix of the matrix:

$$\begin{pmatrix} 1. & 1. & 0. \\ 1. & 1. & 0. \\ 0. & 0. & 2. \\ 1. & 1. & 0. \end{pmatrix}$$

```
      INTEGER LA,LIRN,LJCN,LIW
      PARAMETER (LA=21,LIRN=20,LJCN=10,LIW=50)
      INTEGER IRN(LIRN),JCN(LJCN),IW(LIW)
      DOUBLE PRECISION ASPK(LA)
      DOUBLE PRECISION CNTL(10), RINFO(10)
      INTEGER I,M,N,NZ
      INTEGER ICNTL(20), RANK, ROWS(10), COLS(10), INFO(20)

      EXTERNAL MC58ID,MC58AD

C Read matrix
      READ(5,'(3I4)') M,N,NZ
      READ(5,'(20I4)') (IRN(I),I=1,NZ)
      READ(5,'(20I4)') (JCN(I),I=1,NZ)
      READ(5,'(20F4.1)') (ASPK(I),I=1,NZ)

C Set control parameters
      CALL MC58ID(CNTL,ICNTL)
C Turn on full diagnostic printing
      ICNTL(4) = 4

C Call MC58AD
      CALL MC58AD(M,N,NZ,LA,ASPK,LIRN,IRN,LJCN,JCN,
     +            CNTL,ICNTL,LIW,IW,INFO,RINFO,
     +            RANK,ROWS,COLS)

c Output is generated from subroutine
```

```
      STOP
      END
```

we have as input

```
     4   3   7
     1   2   4   1   2   4   3
     1   1   1   2   2   2   3
   1.0 1.0 1.0 1.0 1.0 1.0 2.0
```

and the output would be

```
 Entering MC58AD with ...

 M         Number of rows in matrix           =          4
 N         Number of columns in matrix        =          3
 NE        Number of entries                  =          7
 LA        Length of array A                  =         21
 LIRN      Length of array IRN                =         20
 LJCN      Length of array JCN                =         10
 LIW       Length of array IW                 =         50
 ICNTL(1)  Stream for errors                  =          6
  --- (2)  Stream for warnings                =          6
  --- (3)  Stream for monitoring              =          6
  --- (4)  Level of diagnostic printing       =          4
  --- (5)  Searching by rows/columns (=0 both) =         0
  --- (6)  Number rows/columns searched       =          3
  --- (7)  Blocking for dense matrix working  =         32
  --- (8)  Rook pivoting control (=1 on)      =          1
  CNTL(1) Control for switch to dense code    = 0.40000D+00
  --- (2) Value of threshold parameter        = 0.10000D-01
  --- (3) Threshold for zero entry            = 0.10000D-11

 Matrix entries:
      1     1  1.0000D+00     2     1  1.0000D+00     4     1  1.0000D+00
      1     2  1.0000D+00     2     2  1.0000D+00     4     2  1.0000D+00
      3     3  2.0000D+00

 Leaving MC58AD with ...

 INFO (1)  Error flag                         =          0
  --- (2)  Further information on error       =          0
  --- (3)  Number of compresses               =          0
  --- (4)  Minimum value for LA               =         20
  --- (5)  Minimum value for LIRN             =         16
  --- (6)  Minimum value for LJCN             =          7
  --- (7)  Minimum value for LIW              =         46
  --- (8)  Number of duplicates               =          0
  --- (9)  Number of out-of-range indices     =          0
  --- (10) Number of rows in full block       =          4
  --- (11) Number of columns in full block    =          3
  --- (12) Estimated rank of sparse part      =          0
  --- (13) Estimated rank of full block       =          2
  --- (14) Number of null or "zero" rows      =          0
  --- (15) Number of null columns             =          0
  --- (16) Number of "zero" columns           =          0
 RINFO(1)  Operations during factorization    =   2.500D+01
 -----(2)  Modulus of largest entry in matrix  =   2.000D+00

 Estimated rank of matrix                     =          2

 Rows and columns in nonsingular block
 Rows:
        3       4

 Columns:
        1       3
```