1 SUMMARY

Calculate **scaling factors** for the rows and columns of an n by n **real or complex matrix.**

If the scaling is applied before Gaussian elimination with pivoting, the choice of pivots will more likely lead to low growth in round-off errors.

The method is described in A.R. Curtis and J.K. Reid, J. Inst. Maths. Applics. (1972), **10**, 118-124.

ATTRIBUTES — Version: 1.0.0. (12 July 2004) **Types:** Real (single, double), Complex (single, double). **Original date:** May 2001. **Remark:** `MC72` is a threadsafe version of `MC42` and `MF42`. **Origin:** A.R.Curtis and J.K.Reid, Harwell.

2 HOW TO USE THE PACKAGE

2.1 Scaling

Scaling factors are given so that pivots should be chosen as if the matrix elements had been

$$b_{ij} = R_i \, a_{ij} \, C_j$$

2.2 The argument list

The single precision version

```
CALL MC72A(A,IA,N,R,C,W,LP,IFAIL)
```

The double precision version

```
CALL MC72AD(A,IA,N,R,C,W,LP,IFAIL)
```

The single precision complex version

```
CALL MC72AC(A,IA,N,R,C,W,LP,IFAIL)
```

The double precision complex version

```
CALL MC72AZ(A,IA,N,R,C,W,LP,IFAIL)
```

`A` is an array whose elements must be set to the elements of the matrix **A** and whose type depends on the entry: `MC72A` requires `REAL`, `MC72AD` requires `DOUBLE PRECISION`, `MC72AC` requires `COMPLEX` and `MC72AZ` requires `COMPLEX*16`. The array is not altered by the subroutine.

`IA` is an `INTEGER` variable which must be set by the user to the first dimension of array **A**. It is not altered by the subroutine.

`N` is an `INTEGER` variable which must be set by the user to n the order of **A**. It is not altered by the subroutine.

`R` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size at least n that need not be set on entry and on return will contain the row scaling factors U_i, $i = 1, 2, , ..., n$.

`C` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size at least n that need not be set on entry and on return will will contain the column scaling factors V_j, $j = 1, 2, , ..., n$.

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) workspace array of dimensions (n,5).

`LP` is an `INTEGER` variable to be used by the subroutine when writing error messages. It must be set by the user to a valid unit number (normally 6), or set to zero to suppress messages. It is not altered by the subroutine.

`IFAIL` is an `INTEGER` variable which need not be set by the user. It is used to indicate success or failure. On exit, `IFAIL` will take one of the following values.

0 successful entry.
1 $n < 1$.

3 GENERAL INFORMATION

Use of common: None.

Workspace: See argument `W`.

Other routines called directly: None.

Input/output: Error messages are written to unit `LP`, see §2.2.

4 METHOD

The variables ρ_i and c_j are chosen to minimize the function

$$\phi = \sum_{i,j} (f_{ij} - \rho_i - c_j)^2$$

where

$$f_{ij} = \log|a_{ij}|$$

and summation is over pairs i,j for which $a_{ij} \neq 0$. This is done to sufficient accuracy in only a few matrix-by-vector multiplications. Then R_i and C_j are obtained as $\exp(-\rho_i)$ and $\exp(-c_j)$. See Curtis and Reid (1972) for further information.

When **A** is a sparse matrix, use of this method is strongly recommended in preference to scaling to equilibrate row and column norms and a subroutine `MC29` using it has been specially written for an economical sparse matrix storage scheme. However, it may well be advantageous even if **A** is full, and `MC72` should definitely be used where there is a possibility that many elements of **A** are zero.

References

Curtis, A.R. and Reid, J.K. 'On the automatic scaling of matrices for Gaussian elimination'. J. Inst. Maths. Applics. (1972), **10**, 118-124.

5 EXAMPLE OF USE

The following program reads a matrix, scales it and prints the result.

```
      INTEGER I,IA,J,IFAIL,LP,N
      PARAMETER (IA=10,LP=6)
      DOUBLE PRECISION A(IA,IA),C(IA),R(IA),W(5*IA)

C  Read size
      READ (5,*) N

C  Check that this is within bounds
      IF (N.LE.0 .OR. N.GT.IA) THEN
         WRITE (6,'(A)') ' N out of permitted range'
         STOP
      END IF

C  Read matrix entries and call MC72
      READ (5,*) ((A(I,J),J=1,N),I=1,N)
      CALL MC72AD(A,IA,N,R,C,W,LP,IFAIL)
```

```
C  Scale the matrix and print it
      DO 20 I = 1,N
        DO 10 J = 1,N
          A(I,J) = A(I,J)*R(I)*C(J)
          WRITE (6,'(10F8.4)') (A(I,J),J=1,N)
   10   CONTINUE
   20 CONTINUE
      END
```

To scale the following matrix

$$\begin{pmatrix} 100.3 & 0.0 & 0.0 & 3.2. \\ 0.0 & 6.0 & 0.0 & 600.7 \\ 900.7 & 0.0 & 110000.6 & 500.8 \\ 0.0 & 14000.2 & 16000.0 & 1.1 \end{pmatrix}$$

we could have as input

```
  4
  100.3     0.0      0.0    3.2
    0.0     6.0      0.0  600.7
  900.7     0.0 110000.6  500.8
    0.0 14000.2  16000.0    1.1
```

and we would get the following output for the scaled matrix

```
    2.0870  0.0000  0.0000  0.5387
    0.0000  0.0247  0.0000 25.9070
    0.4792  0.0000  0.9897  2.1557
    0.0000 40.4315  1.0104  0.0332
```