1 SUMMARY

This subroutine **calculates symmetric scaling factors for a sparse matrix A** = $\{a_{ij}\}_{n\times n}$**.** They may be used, for instance, to scale the matrix prior to solving a corresponding set of linear equations, and are chosen so that the scaled matrix has its entries near to unity in the sense that the sum of the squares of the logarithms of the entries is minimized. The natural logarithms of the scaling factors s_i, $i = 1, 2, ..., n$ for the rows and columns are returned so that the scaled matrix has entries

$$b_{ij} = a_{ij} \exp(s_i + s_j).$$

This is an adaptation to the symmetric case of the method described by Curtis and Reid, J. Inst. Maths. Applics. (1972), **10**, pp. 118-124.

We recommend that this package be considered in conjunction with any of the HSL packages for the direct solution of linear sets of equations when symmetric scaling is appropriate (its companion `MC29` is available for unsymmetric scaling) unless the matrix is symmetric and positive definite. This is because the pivoting strategies are based on the aim of finding a solution that is exact for a perturbed system

$$(\mathbf{A} + \delta\mathbf{A})\mathbf{x} = \mathbf{b} + \delta\mathbf{b}$$

where all the entries of $\delta\mathbf{A}$ are small compared with $\max|a_{ij}|$. Pivoting will not be successful if changes that are small compared with $\max|a_{ij}|$ cannot be tolerated, for example, if mixed units are in use so that the entries of some columns are much smaller than those of others. Scaling should really be applied to the errors that can be tolerated in the elements of **A**, but these are probably unknown. If all the entries are known to the same relative accuracy, scaling **A** itself has the same effect and `MC30` is likely to be very helpful. In other cases, it may give undue weight to small entries that are not known accurately.

ATTRIBUTES — Version: 1.0.0. (12 July 2004) **Types:** Real (single, double). **Original date:** March 1993. **Origin:** J. K. Reid, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Argument list

The single precision version

```
CALL MC30A(N,NE,A,IRN,ICN,S,W,LP,IFAIL)
```

The double precision version

```
CALL MC30AD(N,NE,A,IRN,ICN,S,W,LP,IFAIL)
```

`N` is an `INTEGER` variable that must be set by the user to the order n of the matrix **A**. `N` is not altered by `MC30A/AD`. **Restriction:** `N` ≥ 1.

`NE` is an `INTEGER` variable that must be set by the user to the number of entries in **A**. `NE` is not altered by `MC30A/AD`. **Restriction:** `NE` ≥ 1.

`A` is a `REAL` array (`DOUBLE PRECISION` in the D version) of length `NE` that must be set by the user to hold the values of the entries of the matrix **A**. If the matrix is symmetric, only one of a pair of off-diagonal entries should be present. The entries may be in any order and the subroutine does not change their values.

`IRN,ICN` are two `INTEGER` arrays of length `NE` that the user must set to the row and column numbers of the entries. If the matrix entry a_{ij} is held in `A(K)` then `IRN(K)` must contain i and `ICN(K)` must contain j, or vice-versa. If `IRN(k)` or `ICN(k)` is out of range, the entry is ignored. These arrays are not altered by the subroutine.

S is a REAL array (DOUBLE PRECISION in the D version) of length n that need not be set by the user. On return S(i) contains s_i, $i = 1, 2, ..., n$.

W is a REAL array (DOUBLE PRECISION in the D version) of length $4n$ used for workspace.

LP is an INTEGER variable that must be set by the user to specify the unit number to be used for the error messages, or zero to suppress the printing of error messages. LP is not altered by MC30A/AD.

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

- 0 successful entry,
- –1 N < 1,
- –2 NE < 1,

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly: None.

Input/output: in the event of errors, diagnostic messages are output on unit LP.

Restrictions: N ≥ 1, NE ≥ 1.

4 METHOD

The variables s_i are chosen to minimize the function

$$\Phi = \sum_{i,j} (f_{ij} + s_i + s_j)^2$$

where

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

and the summation is over pairs i, j for which $a_{ij} \neq 0$. This done to sufficient accuracy in only a few matrix-by-vector multiplications. For further information, see Curtis and Reid, On the Automatic Scaling of Matrices for Gaussian Elimination, J. Inst. Maths. Applics. (1972), **10**, pp. 118-124.

Use of this method gives far better results on non-definite symmetric sparse matrices than scaling to equilibrate row norms. However, for a positive-definite matrix, scaling to make the diagonal entries equal to unity is equally satisfactory.

5 EXAMPLE OF USE

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

```
      DOUBLE PRECISION A(1000),S(100),W(400)
      INTEGER IRN(1000),ICN(1000),I,IFAIL,K,LP,N,NE
      PARAMETER (LP=6)

C  Read order and number of entries
      READ(5,*) N,NE
C  Check that N and NE are within bounds
      IF(N.LE.0.OR.N.GT.100) GO TO 40
      IF(NE.LE.0.OR.NE.GT.1000) GO TO 40

C  Read matrix entries and call MC30
      READ(5,*) (IRN(I),ICN(I),A(I),I=1,NE)
      CALL MC30AD(N,NE,A,IRN,ICN,S,W,LP,IFAIL)
```

```
C
C  Calculate scaling multipliers
      DO 10 I=1,N
         S(I)=EXP(S(I))
   10 CONTINUE

C  Scale the matrix and print it
      DO 20 K=1,NE
         A(K)=A(K)*S(IRN(K))*S(ICN(K))
   20 CONTINUE
      WRITE(6,30) (IRN(I),ICN(I),A(I),I=1,NE)
   30 FORMAT(2I5,F10.4)
      STOP

C  Deal with error condition
   40 WRITE(6,'(A)')' N OR NE out of permitted range'
      END
```

To scale the following matrix

$$\begin{pmatrix} 100. & 0. & 900. & 0. \\ 0. & 6. & 0. & 14000. \\ 900. & 0. & 110000. & 0. \\ 0. & 14000. & 0 & 16000. \end{pmatrix}$$

we could have as input

```
4       6
4       4          16000.
1       1            100.
4       2          14000.
2       2              6.
3       1            900.
3       3          110000.
```

and we would get the following output

```
4    4    0.1488
1    1    1.9197
4    2    6.7220
2    2    0.1488
3    1    0.5209
3    3    1.9197
```

which corresponds to the scaled matrix

$$\begin{pmatrix} 1.9197 & 0. & 0.5209 & 0. \\ 0. & 0.1488 & 0. & 6.7220 \\ 0.5209 & 0. & 1.9197 & 0. \\ 0. & 6.7220 & 0. & 0.1488 \end{pmatrix}.$$