M01EDF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

M01EDF rearranges a vector of ***complex*** numbers into the order specified by a vector of ranks.

2 Specification

```
SUBROUTINE M01EDF(CV, M1, M2, IRANK, IFAIL)
INTEGER           M1, M2, IRANK(M2), IFAIL
complex           CV(M2)
```

3 Description

M01EDF is designed to be used typically in conjunction with the M01D- ranking routines. After one of the M01D- routines has been called to determine a vector of ranks, M01EDF can be called to rearrange a vector of complex numbers into the rank order. If the vector of ranks has been generated in some other way, then M01ZBF should be called to check its validity before M01EDF is called.

4 References

None.

5 Parameters

1: CV(M2) — ***complex*** array *Input/Output*

On entry: elements M1 to M2 of CV must contain ***complex*** values to be rearranged.

On exit: these values are rearranged into rank order. For example, if IRANK(i) = M1, then the initial value of CV(i) is moved to CV(M1).

2: M1 — INTEGER *Input*

3: M2 — INTEGER *Input*

On entry: M1 and M2 must specify the range of the ranks supplied in IRANK and the elements of CV to be rearranged.

Constraint: $0 < \text{M1} \leq \text{M2}$.

4: IRANK(M2) — INTEGER array *Input*

On entry: elements M1 to M2 of IRANK must contain a permutation of the integers M1 to M2, which are interpreted as a vector of ranks.

5: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, M2 < 1,
or M1 < 1,
or M1 > M2.

IFAIL = 2

Elements M1 to M2 of IRANK contain a value outside the range M1 to M2.

IFAIL = 3

Elements M1 to M2 of IRANK contain a repeated value.

If IFAIL = 2 or 3, elements M1 to M2 of IRANK do not contain a permutation of the integers M1 to M2. On exit, the contents of CV may be corrupted. To check the validity of IRANK without the risk of corrupting CV, use M01ZBF.

7 Accuracy

Not applicable.

8 Further Comments

The average time taken by the routine is approximately proportional to n, where $n = \text{M2} - \text{M1} + 1$.

9 Example

The example program reads a matrix of ***complex*** numbers and rearranges its rows so that the elements in the kth column are in ascending order of modulus. To do this, the program first calls M01DDF to rank the moduli of the elements in the kth column, and then calls M01EDF to rearrange each column into the order specified by the ranks. The value of k is read from the data-file.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     M01EDF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX
      PARAMETER        (MMAX=20,NMAX=20)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, K, M, N
      CHARACTER*30     STRING
*     .. Local Arrays ..
      complex          CM(MMAX,NMAX)
      real             CMOD(MMAX)
      INTEGER          IRANK(MMAX)
*     .. External Subroutines ..
```

```
      EXTERNAL         M01DAF, M01EDF, X04DAF
*     .. Intrinsic Functions ..
      INTRINSIC        ABS
*     .. Executable Statements ..
      WRITE (NOUT,*) 'M01EDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, K
      IF (M.GE.1 .AND. M.LE.MMAX .AND. N.GE.1 .AND. N.LE.NMAX .AND.
     +    K.GE.1 .AND. K.LE.N) THEN
*
*        Read matrix from data file.
*
         DO 20 I = 1, M
            READ (NIN,*) (CM(I,J),J=1,N)
   20    CONTINUE
*
*        Calculate the moduli of the elements in the K-th column.
*
         DO 40 I = 1, M
            CMOD(I) = ABS(CM(I,K))
   40    CONTINUE
*
*        Rearrange the rows so that the elements in the K-th column
*        are in ascending order of modulus.
*
         IFAIL = 0
*
         CALL M01DAF(CMOD,1,M,'Ascending',IRANK,IFAIL)
*
*        Rearrange each column into the order specified by IRANK.
*
         DO 60 J = 1, N
            IFAIL = 0
*
            CALL M01EDF(CM(1,J),1,M,IRANK,IFAIL)
*
   60    CONTINUE
*
*        Print the results.
*
         WRITE (NOUT,*)
         WRITE (STRING,99999) 'Matrix sorted on column', K
         IFAIL = 0
*
         CALL X04DAF('General',' ',M,N,CM,MMAX,STRING,IFAIL)
*
      END IF
      STOP
*
99999 FORMAT (1X,A,I3)
      END
```

9.2 Program Data

```
M01EDF Example Program Data
12 3 2
(6.0, 1.0) (5.0,-2.0) (4.0, 4.0)
(5.0,-3.0) (2.0,-2.0) (1.0, 1.0)
(2.0, 2.0) (4.0, 1.0) (9.0,-3.0)
(4.0, 2.0) (9.0, 6.0) (6.0, 4.0)
(4.0, 0.0) (9.0, 3.0) (5.0, 1.0)
(4.0,-8.0) (1.0, 5.0) (2.0, 1.0)
(3.0,-3.0) (4.0,-5.0) (1.0, 0.0)
(2.0, 4.0) (4.0,-2.0) (6.0,-1.0)
(1.0, 1.0) (6.0, 1.0) (4.0, 0.0)
(9.0, 1.0) (3.0, 3.0) (2.0,-4.0)
(6.0,-1.0) (2.0, 3.0) (5.0,-3.0)
(4.0,-5.0) (9.0, 9.0) (6.0, 7.0)
```

9.3 Program Results

```
 M01EDF Example Program Results

  Matrix sorted on column  2
                1          2          3
  1      5.0000     2.0000     1.0000
        -3.0000    -2.0000     1.0000

  2      6.0000     2.0000     5.0000
        -1.0000     3.0000    -3.0000

  3      2.0000     4.0000     9.0000
         2.0000     1.0000    -3.0000

  4      9.0000     3.0000     2.0000
         1.0000     3.0000    -4.0000

  5      2.0000     4.0000     6.0000
         4.0000    -2.0000    -1.0000

  6      4.0000     1.0000     2.0000
        -8.0000     5.0000     1.0000

  7      6.0000     5.0000     4.0000
         1.0000    -2.0000     4.0000

  8      1.0000     6.0000     4.0000
         1.0000     1.0000     0.0000

  9      3.0000     4.0000     1.0000
        -3.0000    -5.0000     0.0000

 10      4.0000     9.0000     5.0000
         0.0000     3.0000     1.0000

 11      4.0000     9.0000     6.0000
         2.0000     6.0000     4.0000

 12      4.0000     9.0000     6.0000
        -5.0000     9.0000     7.0000
```