NAG Fortran Library Routine Document

M01DZF

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

M01DZF ranks arbitrary data according to a user-supplied comparison routine.

2 Specification

```
SUBROUTINE M01DZF(COMPAR, M1, M2, IRANK, IFAIL)
INTEGER           M1, M2, IRANK(M2), IFAIL
LOGICAL           COMPAR
EXTERNAL          COMPAR
```

3 Description

M01DZF is a general-purpose routine for ranking arbitrary data. M01DZF does not access the data directly; instead it calls a user-supplied routine COMPAR to determine the relative ordering of any two data items. The data items are identified simply by an integer in the range M1 to M2.

M01DZF uses a variant of list-merging, as described by Knuth (1973), pp 165-166. The routine takes advantage of natural ordering in the data, and uses a simple list insertion in a preparatory pass to generate ordered lists of length at least 10.

4 References

Knuth D E (1973) *The Art of Computer Programming (Volume 3)* (2nd Edition) Addison-Wesley

5 Parameters

1: COMPAR – LOGICAL FUNCTION, supplied by the user. *External Procedure*

COMPAR must specify the relative ordering of any two data items; it must return .TRUE. if item I must come strictly **after** item J in the rank ordering.

Its specification is:

```
LOGICAL FUNCTION COMPAR(I, J)
INTEGER                 I, J
```

1: I – INTEGER *Input*
2: J – INTEGER *Input*

On entry: I and J identify the data items to be compared.

Constraint: $M1 \leq I, J \leq M2$.

COMPAR must be declared as EXTERNAL in the (sub)program from which M01DZF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: M1 – INTEGER *Input*
3: M2 – INTEGER *Input*

On entry: M1 and M2 must specify the range of data items to be ranked, and the range of ranks to be assigned. Specifically, M01DZF ranks the data items identified by integers in the range M1 to M2, and assigns ranks in the range M1 to M2 which are stored in elements M1 to M2 of IRANK.

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

4: IRANK(M2) – INTEGER array *Output*

On exit: elements M1 to M2 of IRANK contain the ranks of the data items M1 to M2. Note that the ranks are in the range M1 to M2: thus, if item i is first in the rank ordering, IRANK(i) contains M1.

5: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

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

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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 or warnings detected by the routine:

IFAIL $= 1$

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

7 Accuracy

Not applicable.

8 Further Comments

The average time taken by the routine is approximately proportional to $n \times \log n$, where $n = \text{M2} - \text{M1} + 1$; it will usually be dominated by the time taken in COMPAR.

9 Example

The example program reads records, each of which contains an integer key and a ***real*** number. The program ranks the records first of all in ascending order of the integer key; records with equal keys are ranked in descending order of the ***real*** number if the key is negative, in ascending order of the ***real*** number if the key is positive, and in their original order if the key is zero. After calling M01DZF, the program calls M01ZAF to convert the ranks to indices, and prints the records in rank order. Note the use of COMMON to communicate the data between the main program and the function COMPAR.

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.

```
*     M01DZF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NMAX
      PARAMETER        (NMAX=100)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Arrays in Common ..
      real             RV(NMAX)
      INTEGER          IV(NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, N
*     .. Local Arrays ..
      INTEGER          IRANK(NMAX)
*     .. External Functions ..
      LOGICAL          COMPAR
      EXTERNAL         COMPAR
*     .. External Subroutines ..
      EXTERNAL         M01DZF, M01ZAF
*     .. Common blocks ..
      COMMON           RV, IV
*     .. Executable Statements ..
      WRITE (NOUT,*) 'M01DZF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.GE.1 .AND. N.LE.NMAX) THEN
         READ (NIN,*) (IV(I),RV(I),I=1,N)
         IFAIL = 0
*
         CALL M01DZF(COMPAR,1,N,IRANK,IFAIL)
         CALL M01ZAF(IRANK,1,N,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) '   Data in sorted order'
         WRITE (NOUT,*)
         DO 20 I = 1, N
            WRITE (NOUT,99999) IV(IRANK(I)), RV(IRANK(I))
   20    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,I7,F7.1)
      END
*
      LOGICAL FUNCTION COMPAR(I,J)
*     .. Parameters ..
      INTEGER                 NMAX
      PARAMETER               (NMAX=100)
*     .. Scalar Arguments ..
      INTEGER                 I, J
*     .. Arrays in Common ..
      real                    RV(NMAX)
      INTEGER                 IV(NMAX)
*     .. Common blocks ..
      COMMON                  RV, IV
*     .. Executable Statements ..
      IF (IV(I).NE.IV(J)) THEN
         COMPAR = IV(I) .GT. IV(J)
      ELSE
         IF (IV(I).LT.0) THEN
            COMPAR = RV(I) .LT. RV(J)
         ELSE IF (IV(I).GT.0) THEN
            COMPAR = RV(I) .GT. RV(J)
         ELSE
```

```
            COMPAR = I .LT. J
          END IF
        END IF
        RETURN
        END
```

9.2 Program Data

```
M01DZF Example Program Data
12
 2   3.0
 1   4.0
-1   6.0
 0   5.0
 2   2.0
-2   7.0
 0   4.0
 1   3.0
 1   5.0
-1   2.0
 1   0.0
 2   1.0
```

9.3 Program Results

```
 M01DZF Example Program Results

    Data in sorted order

      -2    7.0
      -1    6.0
      -1    2.0
       0    4.0
       0    5.0
       1    0.0
       1    3.0
       1    4.0
       1    5.0
       2    1.0
       2    2.0
       2    3.0
```