NAG 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 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 on pages 165–166 in Knuth (1973). 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.

The specification of COMPAR is:

```
FUNCTION COMPAR (I, J)

LOGICAL COMPAR
INTEGER I, J
```

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

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

COMPAR must either be a module subprogram USEd by, or 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: $0 < \mathrm{M1} \leq \mathrm{M2}$.

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. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

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, if you are 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.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (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 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 $n = \mathrm{M2} - \mathrm{M1} + 1$; it will usually be dominated by the time taken in COMPAR.

9 Example

This example 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 COMPAR.

9.1 Program Text

```
!   M01DZF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module m01dzfe_mod

!     M01DZF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: rv(:)
      Integer, Allocatable             :: iv(:)
    Contains
      Function compar(i,j)

!       .. Function Return Value ..
        Logical                          :: compar
!       .. Scalar Arguments ..
        Integer, Intent (In)             :: i, j
!       .. Executable Statements ..
        If (iv(i)/=iv(j)) Then
          compar = iv(i) > iv(j)
        Else

          If (iv(i)<0) Then
            compar = rv(i) < rv(j)
          Else If (iv(i)>0) Then
            compar = rv(i) > rv(j)
          Else
            compar = i < j
          End If

        End If

        Return

      End Function compar
    End Module m01dzfe_mod
    Program m01dzfe

!     M01DZF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: m01dzf, m01zaf
      Use m01dzfe_mod, Only: compar, iv, nin, nout, rv
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Integer                          :: i, ifail, m1, m2
!     .. Local Arrays ..
      Integer, Allocatable             :: irank(:)
!     .. Executable Statements ..
      Write (nout,*) 'M01DZF Example Program Results'

!     Skip heading in data file
      Read (nin,*)

      Read (nin,*) m2
      Allocate (iv(m2),rv(m2),irank(m2))

      m1 = 1

      Read (nin,*)(iv(i),rv(i),i=m1,m2)
```

```
      ifail = 0
      Call m01dzf(compar,m1,m2,irank,ifail)

      ifail = 0
      Call m01zaf(irank,m1,m2,ifail)

      Write (nout,*)
      Write (nout,*) '   Data in sorted order'
      Write (nout,*)

      Write (nout,99999)(iv(irank(i)),rv(irank(i)),i=m1,m2)

      Deallocate (iv,rv)

99999 Format (1X,I7,F7.1)
    End Program m01dzfe
```

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
```