NAG Library Routine Document

M01DAF

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

M01DAF ranks a vector of real numbers in ascending or descending order.

2 Specification

```
SUBROUTINE M01DAF (RV, M1, M2, ORDER, IRANK, IFAIL)
INTEGER            M1, M2, IRANK(M2), IFAIL
REAL (KIND=nag_wp) RV(M2)
CHARACTER(1)       ORDER
```

3 Description

M01DAF 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. The ranking is stable: equal elements preserve their ordering in the input data.

4 References

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

5 Parameters

1: RV(M2) – REAL (KIND=nag_wp) array *Input*

On entry: elements M1 to M2 of RV must contain real values to be ranked.

2: M1 – INTEGER *Input*

On entry: the index of the first element of RV to be ranked.

Constraint: $\mathrm{M1} > 0$.

3: M2 – INTEGER *Input*

On entry: the index of the last element of RV to be ranked.

Constraint: $\mathrm{M2} \geq \mathrm{M1}$.

4: ORDER – CHARACTER(1) *Input*

On entry: if ORDER = 'A', the values will be ranked in ascending (i.e., nondecreasing) order.

If ORDER = 'D', into descending order.

Constraint: ORDER = 'A' or 'D'.

5: IRANK(M2) – INTEGER array *Output*

On exit: elements M1 to M2 of IRANK contain the ranks of the corresponding elements of RV. Note that the ranks are in the range M1 to M2: thus, if RV(i) is the first element in the rank order, IRANK(i) is set to M1.

6: 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.

IFAIL $= 2$

On entry, ORDER is not 'A' or 'D'.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

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

10 Example

This example reads a list of real numbers and ranks them in ascending order.

10.1 Program Text

```
    Program m01dafe

!     M01DAF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: m01daf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, m1, m2
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: rv(:)
      Integer, Allocatable             :: irank(:)
!     .. Executable Statements ..
      Write (nout,*) 'M01DAF Example Program Results'

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

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

      m1 = 1

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

      ifail = 0
      Call m01daf(rv,m1,m2,'Ascending',irank,ifail)

      Write (nout,*)
      Write (nout,*) '   Data   Ranks'
      Write (nout,*)

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

99999 Format (1X,F7.1,I7)
    End Program m01dafe
```

10.2 Program Data

```
M01DAF Example Program Data
12
5.3 4.6 7.8 1.7 5.3 9.9 3.2 4.3 7.8 4.5 1.2 7.6
```

10.3 Program Results

```
 M01DAF Example Program Results

    Data   Ranks

     5.3      7
     4.6      6
     7.8     10
     1.7      2
     5.3      8
     9.9     12
     3.2      3
     4.3      4
     7.8     11
     4.5      5
     1.2      1
     7.6      9
```