NAG Library Routine Document

M01EAF

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

M01EAF rearranges a vector of real numbers into the order specified by a vector of ranks.

2 Specification

```
SUBROUTINE M01EAF (RV, M1, M2, IRANK, IFAIL)
INTEGER            M1, M2, IRANK(M2), IFAIL
REAL (KIND=nag_wp) RV(M2)
```

3 Description

M01EAF 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, M01EAF can be called to rearrange a vector of real 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 M01EAF is called.

4 References

None.

5 Arguments

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

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

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

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

4: IRANK(M2) – INTEGER array *Input/Output*

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.

On exit: used as internal workspace prior to being restored and hence is unchanged.

5: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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 $= 2$

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.

IFAIL $= -99$

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

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 RV may be corrupted. To check the validity of IRANK without the risk of corrupting RV, use M01ZBF.

7 Accuracy

Not applicable.

8 Parallelism and Performance

M01EAF is not threaded in any implementation.

9 Further Comments

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

10 Example

This example reads a matrix of real numbers and rearranges its rows so that the elements of the kth column are in ascending order. To do this, the program first calls M01DAF to rank the elements of the kth column, and then calls M01EAF to rearrange each column into the order specified by the ranks. The value of k is read from the datafile.

10.1 Program Text

```
    Program m01eafe

!     M01EAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: m01daf, m01eaf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, j, k, m1, m2, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: rm(:,:)
      Integer, Allocatable             :: irank(:)
!     .. Executable Statements ..
      Write (nout,*) 'M01EAF Example Program Results'

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

      Read (nin,*) m2, n, k

      If (k<1 .Or. k>n) Then
        Go To 100
      End If

      Allocate (rm(m2,n),irank(m2))

      m1 = 1

      Do i = m1, m2
        Read (nin,*)(rm(i,j),j=1,n)
      End Do

      ifail = 0
      Call m01daf(rm(1,k),m1,m2,'Ascending',irank,ifail)

      Do j = 1, n

        ifail = 0
        Call m01eaf(rm(m1,j),m1,m2,irank,ifail)

      End Do

      Write (nout,*)
      Write (nout,99999) 'Matrix sorted on column', k
      Write (nout,*)

      Do i = m1, m2
        Write (nout,99998)(rm(i,j),j=1,n)
      End Do

100   Continue

99999 Format (1X,A,I3)
99998 Format (1X,3F7.1)
    End Program m01eafe
```

10.2 Program Data

```
M01EAF Example Program Data
12 3 1
6.0 5.0 4.0
5.0 2.0 1.0
2.0 4.0 9.0
4.0 9.0 6.0
4.0 9.0 5.0
4.0 1.0 2.0
3.0 4.0 1.0
2.0 4.0 6.0
1.0 6.0 4.0
9.0 3.0 2.0
6.0 2.0 5.0
4.0 9.0 6.0
```

10.3 Program Results

```
 M01EAF Example Program Results

 Matrix sorted on column  1

     1.0    6.0    4.0
     2.0    4.0    9.0
     2.0    4.0    6.0
     3.0    4.0    1.0
     4.0    9.0    6.0
     4.0    9.0    5.0
     4.0    1.0    2.0
     4.0    9.0    6.0
     5.0    2.0    1.0
     6.0    5.0    4.0
     6.0    2.0    5.0
     9.0    3.0    2.0
```