NAG Library Routine Document

F01CRF

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

F01CRF transposes a rectangular matrix in-situ.

2 Specification

```
SUBROUTINE F01CRF (A, M, N, MN, MOVE, LMOVE, IFAIL)
INTEGER            M, N, MN, MOVE(LMOVE), LMOVE, IFAIL
REAL (KIND=nag_wp) A(MN)
```

3 Description

F01CRF requires that the elements of an m by n matrix A are stored consecutively by columns in a one-dimensional array. It reorders the elements so that on exit the array holds the transpose of A stored in the same way. For example, if $m = 4$ and $n = 3$, on entry the array must hold:

$$a_{11} \quad a_{21} \quad a_{31} \quad a_{41} \quad a_{12} \quad a_{22} \quad a_{32} \quad a_{42} \quad a_{13} \quad a_{23} \quad a_{33} \quad a_{43}$$

and on exit it holds

$$a_{11} \quad a_{12} \quad a_{13} \quad a_{21} \quad a_{22} \quad a_{23} \quad a_{31} \quad a_{32} \quad a_{33} \quad a_{41} \quad a_{42} \quad a_{43}.$$

4 References

Cate E G and Twigg D W (1977) Algorithm 513: Analysis of in-situ transposition *ACM Trans. Math. Software* **3** 104–110

5 Arguments

1: A(MN) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the elements of the m by n matrix A, stored by columns.

On exit: the elements of the transpose matrix, also stored by columns.

2: M – INTEGER *Input*

On entry: m, the number of rows of the matrix A.

3: N – INTEGER *Input*

On entry: n, the number of columns of the matrix A.

4: MN – INTEGER *Input*

On entry: n, the value $m \times n$.

5: MOVE(LMOVE) – INTEGER array *Workspace*

6: LMOVE – INTEGER *Input*

On entry: the dimension of the array MOVE as declared in the (sub)program from which F01CRF is called.

Suggested value: LMOVE $= (m+n)/2$.

Constraint: LMOVE ≥ 1.

7: 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, MN $\neq$ M $\times$ N.

IFAIL $= 2$

On entry, LMOVE ≤ 0.

IFAIL < 0

A serious error has occurred. Check all subroutine calls and array sizes. Seek expert help.

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.

7 Accuracy

Exact results are produced.

8 Parallelism and Performance

F01CRF is not threaded in any implementation.

9 Further Comments

The time taken by F01CRF is approximately proportional to mn.

10 Example

This example transposes a 7 by 3 matrix and prints out, for convenience, its transpose.

10.1 Program Text

```
    Program f01crfe

!     F01CRF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f01crf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, lmove, m, mn, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:)
      Integer, Allocatable             :: move(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: real
!     .. Executable Statements ..
      Write (nout,*) 'F01CRF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n
      mn = m*n
      lmove = (m+n)/2
      Allocate (a(mn),move(lmove))
      Do i = 1, mn
        a(i) = real(i,kind=nag_wp)
      End Do

!     ifail: behaviour on error exit
!            =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f01crf(a,m,n,mn,move,lmove,ifail)

      Write (nout,*)
      Write (nout,99999) a(1:mn)

99999 Format (1X,7F7.1)
    End Program f01crfe
```

10.2 Program Data

```
F01CRF Example Program Data
  3  7           : m, n
```

10.3 Program Results

```
 F01CRF Example Program Results

     1.0    4.0    7.0   10.0   13.0   16.0   19.0
     2.0    5.0    8.0   11.0   14.0   17.0   20.0
     3.0    6.0    9.0   12.0   15.0   18.0   21.0
```