NAG Library Routine Document

M01ZCF

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

M01ZCF decomposes a permutation into cycles, as an aid to reordering ranked data.

2 Specification

```
SUBROUTINE M01ZCF (IPERM, M1, M2, ICYCL, IFAIL)

INTEGER IPERM(M2), M1, M2, ICYCL(M2), IFAIL
```

3 Description

M01ZCF is provided as an aid to reordering arbitrary data structures without using additional storage. However, you should consider carefully whether it is necessary to rearrange yourr data, or whether it would be simpler and more efficient to refer to the data in sorted order using an index vector, or to create a copy of the data in sorted order.

To rearrange data into a different order without using additional storage, the simplest method is to decompose the permutation which specifies the new order into cycles and then to do a cyclic permutation of the data items in each cycle. (This is the method used by the M01E reordering routines.) Given a vector IRANK which specifies the ranks of the data (as generated by the M01D routines), M01ZCF generates a new vector ICYCL, in which the permutation is represented in its component cycles, with the first element of each cycle negated. For example, the permutation

$$5 \quad 7 \quad 4 \quad 2 \quad 1 \quad 6 \quad 3$$

is composed of the cycles

$$(1 \quad 5) \quad (2 \quad 7 \quad 3 \quad 4) \quad (6)$$

and the vector ICYCL generated by M01ZCF contains

$$-1 \quad 5 \quad -2 \quad 7 \quad 3 \quad 4 \quad -6$$

In order to rearrange the data according to the specified ranks:

item 6 must be left in place;

items 1 and 5 must be interchanged;

items 4, 2, 7 and 3 must be moved right one place round the cycle.

The complete rearrangement can be achieved by the following code:

```
DO 10 K = M1, M2
I = ICYCL(K)
IF (I.LT.0) THEN
J = -I
ELSE
[swap items I and J]
ENDIF
10 CONTINUE
```

4 References

None.

5 Parameters

1: IPERM(M2) – INTEGER array *Input/Output*

On entry: elements M1 to M2 of IPERM must contain a permutation of the integers M1 to M2.

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

2: M1 – INTEGER *Input*

3: M2 – INTEGER *Input*

On entry: M1 and M2 must specify the range of elements used in the array IPERM and the range of values in the permutation, as specified under IPERM.

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

4: ICYCL(M2) – INTEGER array *Output*

On exit: elements M1 to M2 of ICYCL contain a representation of the permutation as a list of cycles, with the first integer in each cycle negated. (See Section 3.)

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, $\mathrm{M2} < 1$,
or $\mathrm{M1} < 1$,
or $\mathrm{M1} > \mathrm{M2}$.

IFAIL $= 2$

Elements M1 to M2 of IPERM contain a value outside the range M1 to M2.

IFAIL $= 3$

Elements M1 to M2 of IPERM contain a repeated value.

If IFAIL $= 2$ or 3, elements M1 to M2 of IPERM do not contain a permutation of the integers M1 to M2.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

This example reads a matrix of real numbers and rearranges its columns so that the elements of the lth row are in ascending order. To do this, the program first calls M01DJF to rank the elements of the lth row, and then calls M01ZCF to decompose the rank vector into cycles. It then rearranges the columns using the framework of code suggested in Section 3. The value of l is read from the data file.

9.1 Program Text

```
    Program m01zcfe

!     M01ZCF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: m01djf, m01zcf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: t
      Integer                          :: i, ifail, ii, j, k, l, ldm, m1, m2,  &
                                          n1, n2
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: rm(:,:)
      Integer, Allocatable             :: icycl(:), iperm(:)
!     .. Executable Statements ..
      Write (nout,*) 'M01ZCF Example Program Results'

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

      Read (nin,*) m2, n2, l

      If (l<1 .Or. l>m2) Then
        Go To 100
      End If

      ldm = m2
      Allocate (rm(ldm,n2),icycl(n2),iperm(n2))

      m1 = 1
      n1 = 1

      Do i = m1, m2
        Read (nin,*)(rm(i,j),j=n1,n2)
      End Do

      ifail = 0
      Call m01djf(rm,ldm,l,l,n1,n2,'Ascending',iperm,ifail)

      ifail = 0
      Call m01zcf(iperm,n1,n2,icycl,ifail)

      Do k = n1, n2
        i = icycl(k)

        If (i<0) Then
          j = -i
        Else

!         Swap columns I and J

          Do ii = m1, m2
            t = rm(ii,j)
            rm(ii,j) = rm(ii,i)
            rm(ii,i) = t
```

```
          End Do

        End If

      End Do

      Write (nout,*)
      Write (nout,99999) 'Matrix sorted on row', l
      Write (nout,*)

      Do i = m1, m2
        Write (nout,99998)(rm(i,j),j=n1,n2)
      End Do

100   Continue

99999 Format (1X,A,I3)
99998 Format (1X,12F6.1)
    End Program m01zcfe
```

9.2 Program Data

```
M01ZCF Example Program Data
3 12 3
5.0 4.0 3.0 2.0 2.0 1.0 9.0 4.0 4.0 2.0 2.0 1.0
3.0 8.0 2.0 5.0 5.0 6.0 9.0 8.0 9.0 5.0 4.0 1.0
9.0 1.0 6.0 1.0 2.0 4.0 8.0 1.0 2.0 2.0 6.0 2.0
```

9.3 Program Results

```
 M01ZCF Example Program Results

 Matrix sorted on row  3

    4.0   2.0   4.0   2.0   4.0   2.0   1.0   1.0   3.0   2.0   9.0   5.0
    8.0   5.0   8.0   5.0   9.0   5.0   1.0   6.0   2.0   4.0   9.0   3.0
    1.0   1.0   1.0   2.0   2.0   2.0   2.0   4.0   6.0   6.0   8.0   9.0
```