G05EHF – NAG Fortran Library Routine Document

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

G05EHF performs a pseudo-random permutation of a vector of integers.

2 Specification

```
SUBROUTINE G05EHF(INDEX, N, IFAIL)
INTEGER           INDEX(N), N, IFAIL
```

3 Description

The routine permutes the elements of INDEX without inspecting their values. Each of the $n!$ possible permutations of the n values may be regarded as being equiprobable.

If n is 20 or more, it is theoretically impossible that all $n!$ permutations may occur, as $n!$ exceeds the cycle length of G05CAF. For practical purposes this is irrelevant, as the time necessary to generate all possible permutations is many millenia.

4 References

[1] Knuth D E (1981) *The Art of Computer Programming (Volume 2)* Addison–Wesley (2nd Edition)

[2] Kendall M G and Stuart A (1969) *The Advanced Theory of Statistics (Volume 1)* Griffin (3rd Edition)

5 Parameters

1: INDEX(N) — INTEGER array *Input/Output*

On entry: the n integer values to be permuted.

On exit: the n permuted integer values.

2: N — INTEGER *Input*

On entry: the number of values to be permuted.

Constraint: $\mathrm{N} \geq 1$.

3: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

Errors detected by the routine:

IFAIL $= 1$

On entry, $\mathrm{N} < 1$.

7 Accuracy

Not relevant.

8 Further Comments

The time taken by the routine is of order n.

In order to permute other kinds of vectors, or matrices of higher dimension, the following technique may be used:

(a) Set INDEX(i) = i, for $i = 1, 2, \ldots, n$
(b) Use G05EHF to permute INDEX
(c) Use the contents of INDEX as a set of indices to access the relevant vector or matrix.

In order to divide pseudo-randomly a vector or matrix into subgroups of chosen sizes, a similar procedure may be used. INDEX is first set to the number of 1's, 2's, etc., corresponding to the size of each group, then permuted, and used to index the groups.

9 Example

A vector containing the first 8 positive integers in ascending order is permuted and the permutation is printed. This is repeated a total of 10 times.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EHF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N
      PARAMETER        (N=8)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, K, M
*     .. Local Arrays ..
      INTEGER          INDEX(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EHF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EHF Example Program Results'
      WRITE (NOUT,*)
      M = 10
      CALL G05CBF(0)
      WRITE (NOUT,99998) M, ' Permutations of first ', N, ' integers'
      WRITE (NOUT,*)
      DO 40 J = 1, M
         DO 20 I = 1, N
            INDEX(I) = I
   20    CONTINUE
         IFAIL = 0
*
         CALL G05EHF(INDEX,N,IFAIL)
*
         WRITE (NOUT,99999) (INDEX(K),K=1,N)
```

```
   40 CONTINUE
      STOP
*
99999 FORMAT (1X,8I3)
99998 FORMAT (1X,I2,A,I1,A)
      END
```

9.2 Program Data

None.

9.3 Program Results

```
G05EHF Example Program Results

10 Permutations of first 8 integers

  7  8  1  2  4  6  3  5
  3  1  4  6  7  8  5  2
  7  6  5  1  3  4  8  2
  6  2  7  3  8  5  1  4
  1  6  2  4  7  8  5  3
  4  1  5  8  7  6  3  2
  8  3  1  6  4  2  5  7
  1  2  6  7  8  4  3  5
  2  5  7  6  3  1  4  8
  2  8  6  7  3  5  1  4
```