NAG Library Routine Document

G05KHF

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

G05KHF allows for the generation of multiple, independent, sequences of pseudorandom numbers using the leap-frog method.

2 Specification

```
SUBROUTINE G05KHF (N, K, STATE, IFAIL)
INTEGER N, K, STATE(*), IFAIL
```

3 Description

G05KHF adjusts a base generator to allow multiple, independent, sequences of pseudorandom numbers to be generated via the leap-frog method (see the G05 Chapter Introduction for details).

If, prior to calling G05KHF the base generator defined by STATE would produce random numbers $x_1, x_2, x_3, \ldots$, then after calling G05KHF the generator will produce random numbers $x_k, x_{k+n}, x_{k+2n}, x_{k+3n}, \ldots$.

One of the initialization routines G05KFF (for a repeatable sequence if computed sequentially) or G05KGF (for a non-repeatable sequence) must be called prior to the first call to G05KHF.

The leap-frog algorithm can be used in conjunction with the NAG basic generator, both the Wichmann–Hill I and Wichmann–Hill II generators, the Mersenne Twister and L'Ecuyer.

4 References

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

5 Parameters

1: N – INTEGER *Input*

On entry: n, the total number of sequences required.

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

2: K – INTEGER *Input*

On entry: k, the number of the current sequence.

Constraint: $0 < \mathrm{K} \leq \mathrm{N}$.

3: STATE$(*)$ – INTEGER array *Communication Array*

Note: the actual argument supplied must be the array STATE supplied to the initialization routines G05KFF or G05KGF.

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

4: 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, $\mathrm{N} \leq 0$.

IFAIL = 2

On entry, $\mathrm{K} > \mathrm{N}$.

IFAIL = 3

On entry, STATE vector was not initialized or has been corrupted.

IFAIL = 4

On entry, cannot use the leap-frog method with the base generator defined by STATE.

7 Accuracy

Not applicable.

8 Further Comments

The leap-frog method tends to be less efficient than other methods of producing multiple, independent sequences. See the G05 Chapter Introduction for alternative choices.

9 Example

This example creates three independent sequences using G05KHF, after initialization by G05KFF. Five variates from a uniform distribution are then generated from each sequence using G05SAF.

9.1 Program Text

```
    Program g05khfe

!     G05KHF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05khf, g05saf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
```

```
      Integer, Parameter               :: lseed = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: genid, i, ifail, lstate, n, nv, subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: x(:,:)
      Integer                          :: seed(lseed)
      Integer, Allocatable             :: state(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'G05KHF Example Program Results'
      Write (nout,*)

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

!     Read in the base generator information and seed
      Read (nin,*) genid, subid, seed(1)

!     Read in number of streams and sample size for each stream
      Read (nin,*) n, nv

!     Initial call to initialiser to get size of STATE array
      lstate = 0
      Allocate (state(lstate,1))
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Reallocate STATE
      Deallocate (state)
      Allocate (state(lstate,n))

      Allocate (x(nv,n))

!     Prepare N streams
      Do i = 1, n
!       Initialize each stream to a repeatable sequence
        ifail = 0
        Call g05kff(genid,subid,seed,lseed,state(1,i),lstate,ifail)

!       Prepare the I'th out of N streams
        ifail = 0
        Call g05khf(n,i,state(1,i),ifail)
      End Do

!     Generate a NV variates, from a uniform distribution, from each stream
      Do i = 1, n
        ifail = 0
        Call g05saf(nv,state(1,i),x(1,i),ifail)
      End Do

!     Display results
      Do i = 1, n
        Write (nout,99998) 'Stream ', i
        Write (nout,99999) x(1:nv,i)
        Write (nout,*)
      End Do

99999 Format (1X,F10.4)
99998 Format (1X,A,I16)
    End Program g05khfe
```

9.2 Program Data

```
G05KHF Example Program Data
1  1  1762543     :: GENID,SUBID,SEED(1)
3  5              :: N,NV
```

9.3 Program Results

```
G05KHF Example Program Results

Stream                1
    0.7460
    0.4925
    0.4982
    0.2580
    0.5938

Stream                2
    0.7983
    0.3843
    0.6717
    0.6238
    0.2785

Stream                3
    0.1046
    0.7871
    0.0505
    0.0535
    0.2375
```