NAG Library Routine Document

G05KFF

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

G05KFF initializes the selected base generator, as used by the group of pseudorandom number routines (see G05KHF–G05KJF, G05NCF, G05NDF, G05PDF–G05PJF, G05PXF–G05PZF, G05RCF, G05RDF, G05RYF, G05RZF and G05SAF–G05TLF) and the quasi-random scrambled sequence initialization routine, G05YNF.

2 Specification

```
SUBROUTINE G05KFF (GENID, SUBID, SEED, LSEED, STATE, LSTATE, IFAIL)
INTEGER GENID, SUBID, SEED(LSEED), LSEED, STATE(LSTATE), LSTATE, IFAIL
```

3 Description

G05KFF selects a base generator through the input value of the arguments GENID and SUBID, and then initializes it based on the values given in the array SEED.

A given base generator will yield different sequences of random numbers if initialized with different values of SEED. Alternatively, the same sequence of random numbers will be generated if the same value of SEED is used. It should be noted that there is no guarantee of statistical properties between sequences, only within sequences.

A definition of some of the terms used in this description, along with details of the various base generators can be found in the G05 Chapter Introduction.

4 References

L'Ecuyer P and Simard R (2002) *TestU01: a software library in ANSI C for empirical testing of random number generators* Departement d'Informatique et de Recherche Operationnelle, Universite de Montreal http://www.iro.umontreal.ca/~lecuyer

Maclaren N M (1989) The generation of multiple independent sequences of pseudorandom numbers *Appl. Statist.* **38** 351–359

Matsumoto M and Nishimura T (1998) Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator *ACM Transactions on Modelling and Computer Simulations*

Wichmann B A and Hill I D (2006) Generating good pseudo-random numbers *Computational Statistics and Data Analysis* **51** 1614–1622

Wikramaratna R S (1989) ACORN - a new method for generating sequences of uniformly distributed pseudo-random numbers *Journal of Computational Physics* **83** 16–31

5 Arguments

1: GENID – INTEGER *Input*

On entry: must contain the type of base generator to use.

GENID = 1
    NAG basic generator.

GENID = 2
    Wichmann Hill I generator.

GENID = 3
: Mersenne Twister.

GENID = 4
: Wichmann Hill II generator.

GENID = 5
: ACORN generator.

GENID = 6
: L'Ecuyer MRG32k3a generator.

See the G05 Chapter Introduction for details of each of the base generators.

Constraint: GENID = 1, 2, 3, 4, 5 or 6.

2: SUBID – INTEGER *Input*

On entry: if GENID = 2, SUBID indicates which of the 273 sub-generators to use. In this case, the $((|\text{SUBID}| + 272)\ mod\ 273) + 1$ sub-generator is used.

If GENID = 5, SUBID indicates the values of k and p to use, where k is the order of the generator, and p controls the size of the modulus, M, with $M = 2^{(p \times 30)}$. If SUBID < 1, the default values of $k = 10$ and $p = 2$ are used, otherwise values for k and p are calculated from the formula, SUBID $= k + 1000(p - 1)$.

If GENID = 6 and SUBID mod 2 = 0 the range of the generator is set to $(0, 1]$, otherwise the range is set to $(0, 1)$; in this case the sequence is identical to the implementation of MRG32k3a in TestU01 (see L'Ecuyer and Simard (2002)) for identical seeds.

For all other values of GENID, SUBID is not referenced.

3: SEED(LSEED) – INTEGER array *Input*

On entry: the initial (seed) values for the selected base generator. The number of initial values required varies with each of the base generators.

If GENID = 1, one seed is required.

If GENID = 2, four seeds are required.

If GENID = 3, 624 seeds are required.

If GENID = 4, four seeds are required.

If GENID = 5, $(k + 1)p$ seeds are required, where k and p are defined by SUBID. For the ACORN generator it is recommended that an odd value is used for SEED(1).

If GENID = 6, six seeds are required.

If insufficient seeds are provided then the first LSEED − 1 values supplied in SEED are used and the remaining values are randomly generated using the NAG basic generator. In such cases the NAG basic generator is initialized using the value supplied in SEED(LSEED).

Constraint: SEED$(i) \geq 1$, for $i = 1, 2, \ldots,$ LSEED.

4: LSEED – INTEGER *Input*

On entry: the size of the SEED array.

Constraint: LSEED ≥ 1.

5: STATE(LSTATE) – INTEGER array *Communication Array*

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

6: LSTATE – INTEGER *Input/Output*

On entry: the dimension of the STATE array, or a value < 1. If the Mersenne Twister ($\mathrm{GENID} = 3$) is being used and the skip ahead routine G05KJF or G05KKF will be called subsequently, then you must ensure that $\mathrm{LSTATE} \geq 1260$.

On exit: if $\mathrm{LSTATE} < 1$ on entry, then the required length of the STATE array for the chosen base generator, otherwise LSTATE is unchanged. When $\mathrm{GENID} = 3$ (Mersenne Twister) a value of 1260 is returned, allowing for the skip ahead routine to be subsequently called. In all other cases the minimum length, as documented in the constraints below, is returned.

Constraints:

if $\mathrm{GENID} = 1$, $\mathrm{LSTATE} \geq 17$;
if $\mathrm{GENID} = 2$, $\mathrm{LSTATE} \geq 21$;
if $\mathrm{GENID} = 3$, $\mathrm{LSTATE} \geq 633$;
if $\mathrm{GENID} = 4$, $\mathrm{LSTATE} \geq 29$;
if $\mathrm{GENID} = 5$, $\mathrm{LSTATE} \geq \max((k+1) \times p + 9, 14) + 3$, where k and p are defined by SUBID;
if $\mathrm{GENID} = 6$, $\mathrm{LSTATE} \geq 61$;
otherwise $\mathrm{LSTATE} < 1$.

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.

On exit: $\mathrm{IFAIL} = 0$ or -1 unless the routine detects an error or a warning has been flagged (see Section 6).

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.**

6 Error Indicators and Warnings

If on entry $\mathrm{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

On entry, $\mathrm{GENID} = \langle value \rangle$.
Constraint: $\mathrm{GENID} = 1$, 2, 3, 4, 5 or 6.

$\mathrm{IFAIL} = 3$

On entry, invalid SEED.

$\mathrm{IFAIL} = 4$

On entry, $\mathrm{LSEED} = \langle value \rangle$.
Constraint: $\mathrm{LSEED} \geq 1$.

$\mathrm{IFAIL} = 6$

On entry, $\mathrm{LSTATE} = \langle value \rangle$.
Constraint: $\mathrm{LSTATE} \leq 0$ or $\mathrm{LSTATE} \geq \langle value \rangle$.

$\mathrm{IFAIL} = -1$

Required length of STATE array returned in LSTATE but STATE array not initialized.

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

Not applicable.

8 Parallelism and Performance

G05KFF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example prints the first five pseudorandom real numbers from a uniform distribution between 0 and 1, generated by G05SAF after initialization by G05KFF.

10.1 Program Text

```
    Program g05kffe

!     G05KFF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05saf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                :: lseed = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                           :: genid, ifail, lstate, n, subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable   :: x(:)
      Integer                           :: seed(lseed)
      Integer, Allocatable              :: state(:)
!     .. Executable Statements ..
      Write (nout,*) 'G05KFF 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)

!     Initial call to initializer to get size of STATE array
```

```
      lstate = 0
      Allocate (state(lstate))
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

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

!     Initialize the generator to a repeatable sequence
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Read in sample size
      Read (nin,*) n

      Allocate (x(n))

!     Generate the variates
      ifail = 0
      Call g05saf(n,state,x,ifail)

!     Display the variates
      Write (nout,99999) x(1:n)

99999 Format (1X,F10.4)
    End Program g05kffe
```

10.2 Program Data

```
G05KFF Example Program Data
1  1  1762543     :: GENID,SUBID,SEED(1)
5                 :: N
```

10.3 Program Results

```
 G05KFF Example Program Results

     0.6364
     0.1065
     0.7460
     0.7983
     0.1046
```