

NAG Library Routine Document

G05KGF

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

G05KGF initializes the selected base generator to generate a non-repeatable sequence of variates. The base generator can then be 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 G05KGF (GENID, SUBID, STATE, LSTATE, IFAIL)
  INTEGER GENID, SUBID, STATE(LSTATE), LSTATE, IFAIL
```

3 Description

G05KGF selects a base generator through the input value of the arguments GENID and SUBID, and then initializes it based on the values taken from the real-time clock, resulting in the same base generator yielding different sequences of random numbers each time the calling program is run. 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 $((\text{SUBID} + 272) \bmod 273) + 1$ 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: STATE(LSTATE) – INTEGER array *Communication Array*

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

4: LSTATE – INTEGER *Input/Output*

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

On exit: if LSTATE < 1 on entry, then the required length of the STATE array for the chosen base generator, otherwise LSTATE is unchanged. When 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 GENID = 1, LSTATE ≥ 17;
if GENID = 2, LSTATE ≥ 21;
if GENID = 3, LSTATE ≥ 633;
if GENID = 4, LSTATE ≥ 29;
if GENID = 5, LSTATE ≥ $\max((k + 1) \times p + 9, 14) + 3$, where k and p are defined by SUBID;
if GENID = 6, LSTATE ≥ 61;
otherwise LSTATE < 1.

5: 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: 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 $IFAIL = 0$ 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, $GENID = \langle value \rangle$.

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

$IFAIL = 4$

On entry, $LSTATE = \langle value \rangle$.

Constraint: $LSTATE \leq 0$ or $LSTATE \geq \langle value \rangle$.

$IFAIL = -1$

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

G05KGF is not threaded in any implementation.

9 Further Comments

None.

10 Example

In order to preserve the statistical properties of the base generators, G05KGF should only be called once. If multiple streams of values are required then one of the methods described in Section 2.1.1 in the G05 Chapter Introduction should be used.

However, for illustrative purposes only, this example calls G05KGF twice. At each call a sample of 500 values from a discrete uniform distribution are generated and then the two samples are compared.

10.1 Program Text

```

Program g05kgfe

!      G05KGF Example Program Text

!      Mark 26 Release. NAG Copyright 2016.

!      .. Use Statements ..
      Use nag_library, Only: g05kgf, g05tlf
!      .. Implicit None Statement ..
      Implicit None
!      .. Parameters ..
      Integer, Parameter          :: nout = 6
!      .. Local Scalars ..
      Integer                     :: genid, ifail, lstate, n, subid
!      .. Local Arrays ..
      Integer, Allocatable        :: state(:), x1(:), x2(:)
!      .. Intrinsic Procedures ..
      Intrinsic                   :: any
!      .. Executable Statements ..
      Write (nout,*) 'G05KGF Example Program Results'
      Write (nout,*)

!      Choose the base generator
      genid = 3
      subid = 1

!      Initial call to initializer to get size of STATE array
      lstate = 0
      Allocate (state(lstate))
      ifail = 0
      Call g05kgf(genid,subid,state,lstate,ifail)

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

!      Initialize the generator to a non-repeatable sequence
      ifail = 0
      Call g05kgf(genid,subid,state,lstate,ifail)

!      Using samples of size 500
      n = 500
      Allocate (x1(n),x2(n))

!      Generate a sample of values from a discrete uniform distribution
      Call g05tlf(n,-100,100,state,x1,ifail)

!      Re-initialize the generator to another non-repeatable sequence
!      NB: In practice, in order to preserve its statistical properties,
!      you should only initialize the RNG generators once
      ifail = 0
      Call g05kgf(genid,subid,state,lstate,ifail)

!      Generate a second sample of values from the same distribution
      Call g05tlf(n,-100,100,state,x2,ifail)

!      Check that the two samples are different
      If (any(x1/=x2)) Then
        Write (nout,*) 'The two samples differ, as expected'
      Else
        Write (nout,*) 'The two samples are the same'
        Write (nout,*) 'whilst this is possible, it is unlikely'
      End If

End Program g05kgfe

```

10.2 Program Data

None.

10.3 Program Results

G05KGF Example Program Results

The two samples differ, as expected
