NAG Library Routine Document

G05SJF

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

G05SJF generates a vector of pseudorandom numbers taken from a gamma distribution with parameters a and b.

2 Specification

```
SUBROUTINE G05SJF (N, A, B, STATE, X, IFAIL)
INTEGER            N, STATE(*), IFAIL
REAL (KIND=nag_wp) A, B, X(N)
```

3 Description

The gamma distribution has PDF (probability density function)

$$f(x) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-x/b} \quad \text{if } x \geq 0; \quad a, b > 0$$

$$f(x) = 0 \qquad \text{otherwise.}$$

One of three algorithms is used to generate the variates depending upon the value of a:

(i) if $a < 1$, a switching algorithm described by Dagpunar (1988) (called G6) is used. The target distributions are $f_1(x) = cax^{a-1}/t^a$ and $f_2(x) = (1-c)e^{-(x-t)}$, where $c = t/(t + ae^{-t})$, and the switching parameter, t, is taken as $1 - a$. This is similar to Ahrens and Dieter's GS algorithm (see Ahrens and Dieter (1974)) in which $t = 1$;

(ii) if $a = 1$, the gamma distribution reduces to the exponential distribution and the method based on the logarithmic transformation of a uniform random variate is used;

(iii) if $a > 1$, the algorithm given by Best (1978) is used. This is based on using a Student's t-distribution with two degrees of freedom as the target distribution in an envelope rejection method.

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 G05SJF.

4 References

Ahrens J H and Dieter U (1974) Computer methods for sampling from gamma, beta, Poisson and binomial distributions *Computing* **12** 223–46

Best D J (1978) Letter to the Editor *Appl. Statist.* **27** 181

Dagpunar J (1988) *Principles of Random Variate Generation* Oxford University Press

Hastings N A J and Peacock J B (1975) *Statistical Distributions* Butterworth

5 Parameters

1: N – INTEGER *Input*

On entry: n, the number of pseudorandom numbers to be generated.

Constraint: N ≥ 0.

2: A – REAL (KIND=nag_wp) *Input*

On entry: a, the parameter of the gamma distribution.

Constraint: A > 0.0.

3: B – REAL (KIND=nag_wp) *Input*

On entry: b, the parameter of the gamma distribution.

Constraint: B > 0.0.

4: 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.

5: X(N) – REAL (KIND=nag_wp) array *Output*

On exit: the n pseudorandom numbers from the specified gamma distribution.

6: 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, N $= \langle value \rangle$.
Constraint: N ≥ 0.

IFAIL $= 2$

On entry, A $= \langle value \rangle$.
Constraint: A > 0.0.

IFAIL $= 3$

On entry, B $= \langle value \rangle$.
Constraint: B > 0.0.

IFAIL $= 4$

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

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

G05SJF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

This example prints a set of five pseudorandom numbers from a gamma distribution with parameters $a = 5.0$ and $b = 1.0$, generated by a single call to G05SJF, after initialization by G05KFF.

10.1 Program Text

```
    Program g05sjfe

!     G05SJF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05sjf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: lseed = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: a, b
      Integer                         :: genid, ifail, lstate, n, subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: x(:)
      Integer                         :: seed(lseed)
      Integer, Allocatable            :: state(:)
!     .. Executable Statements ..
      Write (nout,*) 'G05SJF 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 initialiser 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))

!     Read in the distribution parameters
      Read (nin,*) a, b

!     Generate the variates
      ifail = 0
      Call g05sjf(n,a,b,state,x,ifail)

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

99999 Format (1X,F10.4)
    End Program g05sjfe
```

10.2 Program Data

```
G05SJF Example Program Data
1  1  1762543     :: GENID,SUBID,SEED(1)
5 3               :: N,NMIX
5.0 1.0           :: A,B
```

10.3 Program Results

```
 G05SJF Example Program Results

     5.0702
     6.1337
     3.1018
     3.9863
     4.9648
```