NAG Library Routine Document

G05SBF

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

G05SBF generates a vector of pseudorandom numbers taken from a beta distribution with parameters a and b.

2 Specification

```
SUBROUTINE G05SBF (N, A, B, STATE, X, IFAIL)
INTEGER            N, STATE(*), IFAIL
REAL (KIND=nag_wp) A, B, X(N)
```

3 Description

The beta distribution has PDF (probability density function)

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

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

One of four algorithms is used to generate the variates depending on the values of a and b. Let α be the maximum and β be the minimum of a and b. Then the algorithms are as follows:

(i) if $\alpha < 0.5$, Johnk's algorithm is used, see for example Dagpunar (1988). This generates the beta variate as $u_1^{1/a} / \left(u_1^{1/a} + u_2^{1/b} \right)$, where u_1 and u_2 are uniformly distributed random variates;

(ii) if $\beta > 1$, the algorithm BB given by Cheng (1978) is used. This involves the generation of an observation from a beta distribution of the second kind by the envelope rejection method using a log-logistic target distribution and then transforming it to a beta variate;

(iii) if $\alpha > 1$ and $\beta < 1$, the switching algorithm given by Atkinson (1979) is used. The two target distributions used are $f_1(x) = \beta x^\beta$ and $f_2(x) = \alpha(1-x)^{\beta-1}$, along with the approximation to the switching argument of $t = (1-\beta)/(\alpha+1-\beta)$;

(iv) in all other cases, Cheng's BC algorithm (see Cheng (1978)) is used with modifications suggested by Dagpunar (1988). This algorithm is similar to BB, used when $\beta > 1$, but is tuned for small values of a and b.

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 G05SBF.

4 References

Atkinson A C (1979) A family of switching algorithms for the computer generation of beta random variates *Biometrika* **66** 141–5

Cheng R C H (1978) Generating beta variates with nonintegral shape parameters *Comm. ACM* **21** 317–322

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

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

5 Arguments

1: N – INTEGER *Input*

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

Constraint: $N \geq 0$.

2: A – REAL (KIND=nag_wp) *Input*

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

Constraint: $A > 0.0$.

3: B – REAL (KIND=nag_wp) *Input*

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

6: 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.

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.**

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

6 Error Indicators and Warnings

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

Errors or warnings detected by the routine:

$\text{IFAIL} = 1$

On entry, $N = \langle value \rangle$.
Constraint: $N \geq 0$.

$\text{IFAIL} = 2$

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

IFAIL = 3

On entry, B = ⟨*value*⟩.
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 3.9 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

G05SBF 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

To generate an observation, y, from the beta distribution of the second kind from an observation, x, generated by G05SBF the transformation, $y = x/(1 - x)$, may be used.

10 Example

This example prints a set of five pseudorandom numbers from a beta distribution with parameters $a = 2.0$ and $b = 2.0$, generated by a single call to G05SBF, after initialization by G05KFF.

10.1 Program Text

```
    Program g05sbfe

!     G05SBF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05sbf, 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,*) 'G05SBF 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))

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

!     Generate the variates
      ifail = 0
      Call g05sbf(n,a,b,state,x,ifail)

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

99999 Format (1X,F10.4)
    End Program g05sbfe
```

10.2 Program Data

```
G05SBF Example Program Data
1  1  1762543     :: GENID,SUBID,SEED(1)
5                 :: N
2.0 2.0           :: A,B
```

10.3 Program Results

```
 G05SBF Example Program Results

     0.5977
     0.6818
     0.1797
     0.4174
     0.4987
```