NAG Library Routine Document

G05SHF

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

G05SHF generates a vector of pseudorandom numbers taken from an F (or Fisher's variance ratio) distribution with μ and ν degrees of freedom.

2 Specification

```
SUBROUTINE G05SHF (N, DF1, DF2, STATE, X, IFAIL)
INTEGER            N, DF1, DF2, STATE(*), IFAIL
REAL (KIND=nag_wp) X(N)
```

3 Description

The distribution has PDF (probability density function)

$$f(x) = \frac{\left(\frac{\mu+\nu-2}{2}\right)! x^{\frac{1}{2}\mu-1}}{\left(\frac{1}{2}\mu-1\right)!\left(\frac{1}{2}\nu-1\right)!\left(1+\frac{\mu}{\nu}x\right)^{\frac{1}{2}(\mu+\nu)}} \times \left(\frac{\mu}{\nu}\right)^{\frac{1}{2}\mu} \quad \text{if } x > 0,$$

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

G05SHF calculates the values

$$\frac{\nu y_i}{\mu z_i}, \quad i = 1, 2, \ldots, n,$$

where y_i and z_i are generated by G05SJF from gamma distributions with parameters $\left(\frac{1}{2}\mu, 2\right)$ and $\left(\frac{1}{2}\nu, 2\right)$ respectively (i.e., from χ^2-distributions with μ and ν degrees of freedom).

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 G05SHF.

4 References

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

5 Arguments

1: N – INTEGER *Input*

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

Constraint: $\text{N} \geq 0$.

2: DF1 – INTEGER *Input*

On entry: μ, the number of degrees of freedom of the distribution.

Constraint: $\text{DF1} \geq 1$.

3: DF2 – INTEGER *Input*

On entry: ν, the number of degrees of freedom of the distribution.

Constraint: DF2 ≥ 1.

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 F-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: 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, DF1 $= \langle$*value*$\rangle$.
Constraint: DF1 ≥ 1.

IFAIL $= 3$

On entry, DF2 $= \langle$*value*$\rangle$.
Constraint: DF2 ≥ 1.

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

G05SHF 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

The time taken by G05SHF increases with μ and ν.

10 Example

This example prints five pseudorandom numbers from an F-distribution with two and three degrees of freedom, generated by a single call to G05SHF, after initialization by G05KFF.

10.1 Program Text

```
    Program g05shfe

!     G05SHF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05shf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: lseed = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: df1, df2, genid, ifail, lstate, n,   &
                                          subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: x(:)
      Integer                          :: seed(lseed)
      Integer, Allocatable             :: state(:)
!     .. Executable Statements ..
      Write (nout,*) 'G05SHF 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,*) df1, df2

!     Generate the variates
      ifail = 0
      Call g05shf(n,df1,df2,state,x,ifail)

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

99999 Format (1X,F10.4)
    End Program g05shfe
```

10.2 Program Data

```
G05SHF Example Program Data
1  1  1762543     :: GENID,SUBID,SEED(1)
5 3               :: N,NMIX
2 3               :: DF1,DF2
```

10.3 Program Results

```
 G05SHF Example Program Results

     1.4401
     1.8083
     0.3638
     0.5464
     4.0895
```