NAG Library Routine Document

G05SGF

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

G05SGF generates a vector of pseudorandom numbers from an exponential mix distribution composed of m exponential distributions each having a mean a_i and weight w_i.

2   Specification

```
SUBROUTINE G05SGF (N, NMIX, A, WGT, STATE, X, IFAIL)
INTEGER            N, NMIX, STATE(*), IFAIL
REAL (KIND=nag_wp) A(NMIX), WGT(NMIX), X(N)
```

3   Description

The distribution has PDF (probability density function)

$$f(x) = \sum_{i=1}^{m} \tfrac{1}{a_i} w_i e^{-x/a_i} \quad \text{if } x \ge 0,$$
$$f(x) = 0 \qquad\qquad\qquad \text{otherwise,}$$

where $\sum_{i=1}^{m} w_i = 1$ and $a_i > 0$, $w_i \ge 0$.

G05SGF returns the values x_i by selecting, with probability w_j, random variates from an exponential distribution with argument a_j.

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 G05SGF.

4   References

Kendall M G and Stuart A (1969) *The Advanced Theory of Statistics (Volume 1)* (3rd Edition) Griffin

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: N ≥ 0.

2:    NMIX − INTEGER   *Input*

   On entry: m, the number of exponential distributions in the mix.

   Constraint: NMIX ≥ 1.

3:    A(NMIX) − REAL (KIND=nag_wp) array   *Input*

   On entry: the m parameters a_i for the m exponential distributions in the mix.

   Constraint: A(i) > 0.0, for $i = 1, 2, \ldots,$ NMIX.

4:    WGT(NMIX) – REAL (KIND=nag_wp) array   *Input*

On entry: the m weights w_i for the m exponential distributions in the mix.

Constraints:

$$\sum_{i=1}^{m} \text{WGT}(i) = 1.0;$$
$$\text{WGT}(i) \geq 0.0, \text{ for } i = 1, 2, \ldots, m.$$

5:    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.

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

On exit: the n pseudorandom numbers from the specified exponential mix distribution.

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.

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, NMIX $= \langle value \rangle$.
> Constraint: NMIX ≥ 1.

IFAIL $= 3$

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

IFAIL $= 4$

> On entry, sum of WGT $= \langle value \rangle$.
> Constraint: sum of WGT $= 1.0$.

> On entry, WGT($\langle value \rangle$) $= \langle value \rangle$.
> Constraint: WGT(i) ≥ 0.0.

IFAIL = 5

> 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

G05SGF 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 the first five pseudorandom numbers from an exponential mix distribution comprising three exponential distributions with parameters $a_1 = 1.0$, $a_2 = 5.0$ and $a_3 = 2.0$, and with respective weights 0.5, 0.3 and 0.2. The numbers are generated by a single call to G05SGF, after initialization by G05KFF.

10.1 Program Text

```
    Program g05sgfe

!     G05SGF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05sgf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                :: lseed = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                           :: genid, ifail, lstate, n, nmix, subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable   :: a(:), wgt(:), x(:)
      Integer                           :: seed(lseed)
```

```
        Integer, Allocatable                :: state(:)
!       .. Executable Statements ..
        Write (nout,*) 'G05SGF 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 and number of mixtures
        Read (nin,*) n, nmix

        Allocate (x(n),a(nmix),wgt(nmix))

!       Read in the distribution parameters
        Read (nin,*) a(1:nmix)
        Read (nin,*) wgt(1:nmix)

!       Generate the variates
        ifail = 0
        Call g05sgf(n,nmix,a,wgt,state,x,ifail)

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

99999 Format (1X,F10.4)
    End Program g05sgfe
```

10.2 Program Data

```
G05SGF Example Program Data
1  1  1762543      :: GENID,SUBID,SEED(1)
5 3               :: N,NMIX
1.0 5.0 2.0       :: A
0.5 0.3 0.2       :: WGT
```

10.3 Program Results

```
 G05SGF Example Program Results

     0.4520
     2.2398
     1.4649
     0.2253
    11.2884
```