NAG Library Routine Document

G05TAF

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

G05TAF generates a vector of pseudorandom integers from the discrete binomial distribution with parameters m and p.

2 Specification

```
SUBROUTINE G05TAF (MODE, N, M, P, R, LR, STATE, X, IFAIL)
INTEGER            MODE, N, M, LR, STATE(*), X(N), IFAIL
REAL (KIND=nag_wp) P, R(LR)
```

3 Description

G05TAF generates n integers x_i from a discrete binomial distribution, where the probability of $x_i = I$ is

$$P(x_i = I) = \frac{m!}{I!(m-I)!} p^I \times (1-p)^{m-I}, \quad I = 0, 1, \ldots, m,$$

where $m \geq 0$ and $0 \leq p \leq 1$. This represents the probability of achieving I successes in m trials when the probability of success at a single trial is p.

The variates can be generated with or without using a search table and index. If a search table is used then it is stored with the index in a reference vector and subsequent calls to G05TAF with the same parameter values can then use this reference vector to generate further variates.

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 G05TAF.

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: MODE – INTEGER *Input*

On entry: a code for selecting the operation to be performed by the routine.

MODE $= 0$
Set up reference vector only.

MODE $= 1$
Generate variates using reference vector set up in a prior call to G05TAF.

MODE $= 2$
Set up reference vector and generate variates.

MODE $= 3$
Generate variates without using the reference vector.

Constraint: MODE $= 0$, 1, 2 or 3.

2: N – INTEGER *Input*

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

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

3: M – INTEGER *Input*

On entry: m, the number of trials of the distribution.

Constraint: $\mathrm{M} \geq 0$.

4: P – REAL (KIND=nag_wp) *Input*

On entry: p, the probability of success of the binomial distribution.

Constraint: $0.0 \leq \mathrm{P} \leq 1.0$.

5: R(LR) – REAL (KIND=nag_wp) array *Communication Array*

On entry: if MODE $= 1$, the reference vector from the previous call to G05TAF.

If MODE $= 3$, R is not referenced.

On exit: if MODE $\neq 3$, the reference vector.

6: LR – INTEGER *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05TAF is called.

Suggested value:

if MODE $\neq 3$, $\mathrm{LR} = 22 + 20 \times \sqrt{\mathrm{M} \times \mathrm{P} \times (1 - \mathrm{P})}$;
otherwise $\mathrm{LR} = 1$.

Constraints:

if MODE $= 0$ or 2,

$$\mathrm{LR} > \min\left(\mathrm{M}, \mathrm{int}\left[\mathrm{M} \times \mathrm{P} + 7.15 \times \sqrt{\mathrm{M} \times \mathrm{P} \times (1 - \mathrm{P})} + 1\right]\right) - \max\left(0, \mathrm{int}\left[\mathrm{M} \times \mathrm{P} - 7.15 \times \sqrt{\mathrm{M} \times \mathrm{P} \times (1 - \mathrm{P})} - 7.15\right]\right) + 8;$$

if MODE $= 1$, LR must remain unchanged from the previous call to G05TAF.

7: 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.

8: X(N) – INTEGER array *Output*

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

9: 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, MODE $= \langle value \rangle$.
Constraint: MODE $= 0$, 1, 2 or 3.

IFAIL $= 2$

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

IFAIL $= 3$

On entry, M $= \langle value \rangle$.
Constraint: M ≥ 0.

IFAIL $= 4$

On entry, P $= \langle value \rangle$.
Constraint: $0.0 \leq$ P ≤ 1.0.

IFAIL $= 5$

On entry, some of the elements of the array R have been corrupted or have not been initialized.

P or M is not the same as when R was set up in a previous call.
Previous value of P $= \langle value \rangle$ and P $= \langle value \rangle$.
Previous value of M $= \langle value \rangle$ and M $= \langle value \rangle$.

IFAIL $= 6$

On entry, LR is too small when MODE $= 0$ or 2: LR $= \langle value \rangle$, minimum length required $= \langle value \rangle$.

IFAIL $= 7$

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

G05TAF 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 20 pseudorandom integers from a binomial distribution with parameters $m = 6000$ and $p = 0.8$, generated by a single call to G05TAF, after initialization by G05KFF.

10.1 Program Text

```
    Program g05tafe

!     G05TAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05taf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: lseed = 1, maxlr = 5000, nin = 5,    &
                                          nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: p
      Integer                          :: genid, ifail, lr, lstate, m, mode,   &
                                          n, subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: r(:)
      Integer                          :: seed(lseed)
      Integer, Allocatable             :: state(:), x(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: int, real, sqrt
!     .. Executable Statements ..
      Write (nout,*) 'G05TAF 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 mode
      Read (nin,*) n

!     Read in the distribution parameters
      Read (nin,*) p, m

!     Use suggested value for LR
      lr = int(2.2E1_nag_wp+2.0E1_nag_wp*sqrt(real(m,                          &
        kind=nag_wp)*p*(1.0E0_nag_wp-p)))

!     If R is a reasonable size use MODE = 2
!     else do not reference R and use MODE = 3
      If (lr<maxlr) Then
        mode = 2
      Else
        mode = 3
        lr = 0
      End If

      Allocate (x(n),r(lr))

!     Generate the variates
      ifail = 0
      Call g05taf(mode,n,m,p,r,lr,state,x,ifail)

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

99999 Format (1X,I12)
    End Program g05tafe
```

10.2 Program Data

```
G05TAF Example Program Data
1  1  1762543     :: GENID,SUBID,SEED(1)
20                :: N
0.8 6000          :: P,M
```

10.3 Program Results

```
 G05TAF Example Program Results

         4811
         4761
         4821
         4826
         4761
         4800
         4791
         4825
         4800
         4814
         4749
         4780
         4810
         4750
         4807
         4782
         4778
         4877
         4840
         4802
```