NAG Library Routine Document

G05TEF

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

G05TEF generates a vector of pseudorandom integers from the discrete hypergeometric distribution of the number of specified items in a sample of size l, taken from a population of size k with m specified items in it.

2 Specification

```
SUBROUTINE G05TEF (MODE, N, NS, NP, M, R, LR, STATE, X, IFAIL)
INTEGER            MODE, N, NS, NP, M, LR, STATE(*), X(N), IFAIL
REAL (KIND=nag_wp) R(LR)
```

3 Description

G05TEF generates n integers x_i from a discrete hypergeometric distribution, where the probability of $x_i = I$ is

$$P(i = I) = \frac{l!m!(k-l)!(k-m)!}{I!(l-I)!(m-I)!(k-m-l+I)!k!} \qquad \text{if } I = \max(0, m+l-k), \ldots, \min(l, m),$$

$$P(i = I) = 0 \qquad \text{otherwise.}$$

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 G05TEF with the same parameter values can then use this reference vector to generate further variates. The reference array is generated by a recurrence relation if $lm(k-l)(k-m) < 50k^3$, otherwise Stirling's approximation is used.

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 G05TEF.

4 References

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 G05TEF.

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: NS – INTEGER *Input*

On entry: l, the sample size of the hypergeometric distribution.

Constraint: $0 \leq \mathrm{NS} \leq \mathrm{NP}$.

4: NP – INTEGER *Input*

On entry: k, the population size of the hypergeometric distribution.

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

5: M – INTEGER *Input*

On entry: m, the number of specified items of the hypergeometric distribution.

Constraint: $0 \leq \mathrm{M} \leq \mathrm{NP}$.

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

On entry: if $\mathrm{MODE} = 1$, the reference vector from the previous call to G05TEF.

If $\mathrm{MODE} = 3$, R is not referenced.

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

7: LR – INTEGER *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05TEF is called.

Suggested value:

if $\mathrm{MODE} \neq 3$, $\mathrm{LR} = 28 + 20 \times \sqrt{(\mathrm{NS} \times \mathrm{M} \times (\mathrm{NP} - \mathrm{M}) \times (\mathrm{NP} - \mathrm{NS}))/\mathrm{NP}^3}$ approximately;
otherwise $\mathrm{LR} = 1$.

Constraints:

if $\mathrm{MODE} = 0$ or 2, LR must not be too small, but the limit is too complicated to specify;
if $\mathrm{MODE} = 1$, LR must remain unchanged from the previous call to G05TEF.

8: 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.

9: X(N) – INTEGER array *Output*

On exit: the pseudorandom numbers from the specified hypergeometric distribution.

10: 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, NS $= \langle value \rangle$ and NP $= \langle value \rangle$.
Constraint: $0 \leq$ NS $\leq$ NP.

IFAIL $= 4$

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

IFAIL $= 5$

On entry, M $= \langle value \rangle$ and NP $= \langle value \rangle$.
Constraint: $0 \leq$ M $\leq$ NP.

IFAIL $= 6$

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

The value of NS, NP or M is not the same as when R was set up in a previous call with MODE $= 0$ or 2.

IFAIL $= 7$

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

IFAIL $= 8$

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

G05TEF 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

The example program prints 20 pseudorandom integers from a hypergeometric distribution with $l = 500$, $m = 900$ and $n = 1000$, generated by a single call to G05TEF, after initialization by G05KFF.

10.1 Program Text

```
    Program g05tefe

!     G05TEF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05tef, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: lseed = 1, maxlr = 5000, nin = 5,    &
                                          nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: anp, tmp
      Integer                          :: genid, ifail, lr, lstate, m, mode,   &
                                          n, np, ns, 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,*) 'G05TEF 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

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

!     Use suggested value for LR
      anp = real(np,kind=nag_wp)
      tmp = (real(np-m,kind=nag_wp)/anp)*(real(np-ns,kind=nag_wp)/anp)*        &
        (real(m*ns,kind=nag_wp)/real(np,kind=nag_wp))
      lr = 28 + 20*int(sqrt(tmp))

!     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 = -1
      Call g05tef(mode,n,ns,np,m,r,lr,state,x,ifail)

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

99999 Format (1X,I12)
    End Program g05tefe
```

10.2 Program Data

```
G05TEF Example Program Data
1  1  1762543     :: GENID,SUBID,SEED(1)
20                :: N
500  900  1000    :: NS,M,NP
10 5 100
100 2500 10000
9780 50 10000
```

10.3 Program Results

```
 G05TEF Example Program Results

          452
          444
          453
          454
          444
          450
          449
          454
          450
          452
          442
```

```
 447
 451
 442
 451
 447
 447
 462
 456
 450
```