# NAG Library Routine Document

# G05YKF

**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

G05YKF generates a quasi-random sequence from a log-normal distribution. It must be preceded by a call to one of the initialization routines G05YLF or G05YNF.

## 2 Specification

```
SUBROUTINE G05YKF (XMEAN, STD, N, QUAS, IREF, IFAIL)

INTEGER             N, IREF(liref), IFAIL
REAL (KIND=nag_wp) XMEAN(idim), STD(idim), QUAS(N,idim)
```

## 3 Description

G05YKF generates a quasi-random sequence from a log-normal distribution by first generating a uniform quasi-random sequence which is then transformed into a log-normal sequence using the exponential of the inverse of the Normal CDF. The type of uniform sequence used depends on the initialization routine called and can include the low-discrepancy sequences proposed by Sobol, Faure or Niederreiter. If the initialization routine G05YNF was used then the underlying uniform sequence is first scrambled prior to being transformed (see Section 3 in G05YNF for details).

## 4 References

Bratley P and Fox B L (1988) Algorithm 659: implementing Sobol's quasirandom sequence generator *ACM Trans. Math. Software* **14(1)** 88–100

Fox B L (1986) Algorithm 647: implementation and relative efficiency of quasirandom sequence generators *ACM Trans. Math. Software* **12(4)** 362–376

Wichura (1988) Algorithm AS 241: the percentage points of the Normal distribution *Appl. Statist.* **37** 477–484

## 5 Arguments

**Note**: the following variables are used in the parameter descriptions:

$idim = \text{IDIM}$, the number of dimensions required, see G05YLF or G05YNF;

$liref = \text{LIREF}$, the length of IREF as supplied to the initialization routines G05YLF or G05YNF..

1: XMEAN($idim$) – REAL (KIND=nag_wp) array *Input*

*On entry*: specifies, for each dimension, the mean of the underlying Normal distribution.

*Constraint*: $|\text{XMEAN}(i)| \leq |-\log(\text{X02AMF}) - 10.0 \times \text{STD}(i)|$, for $i = 1, 2, \ldots, idim$.

2: STD($idim$) – REAL (KIND=nag_wp) array *Input*

*On entry*: specifies, for each dimension, the standard deviation of the underlying Normal distribution.

*Constraint*: $\text{STD}(i) \geq 0.0$, for $i = 1, 2, \ldots, idim$.

3: N – INTEGER *Input*

*On entry*: the number of quasi-random numbers required.

*Constraint*: $\mathrm{N} \geq 0$ and $\mathrm{N} +$ previous number of generated values $\leq 2^{31} - 1$.

4: QUAS(N, *idim*) – REAL (KIND=nag_wp) array *Output*

*On exit*: contains the N quasi-random numbers of dimension *idim*.

5: IREF(*liref*) – INTEGER array *Communication Array*

*On entry*: contains information on the current state of the sequence.

*On exit*: contains updated information on the state of the sequence.

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

## 6 Error Indicators and Warnings

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

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

On entry, incorrect initialization has been detected.

$\mathrm{IFAIL} = 2$

On entry, $\mathrm{N} < 1$.

$\mathrm{IFAIL} = 3$

On entry,at least one element of XMEAN is too large.

$\mathrm{IFAIL} = 4$

There have been too many calls to the generator.

$\mathrm{IFAIL} = -99$

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.

$\mathrm{IFAIL} = -399$

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

G05YKF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G05YKF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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.

The Sobol, Sobol (A659) and Niederreiter quasi-random number generators in g05ykc have been parallelized, but require quite large problem sizes to see any significant performance gain. The Faure generator is serial.

## 9 Further Comments

None.

## 10 Example

This example calls G05YLF to initialize the generator and then G05YKF to produce a sequence of five four-dimensional quasi-random numbers variates.

### 10.1 Program Text

```
    Program g05ykfe

!     G05YKF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05ykf, g05ylf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: genid, i, idim, ifail, iskip,        &
                                          ldquas, liref, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: quas(:,:), std(:), xmean(:)
      Integer, Allocatable             :: iref(:)
!     .. Executable Statements ..
      Write (nout,*) 'G05YKF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Read in the generator to use
      Read (nin,*) genid
```

```
!     Read in problem size
      Read (nin,*) n, idim, iskip

      If (genid==4) Then
        liref = 407
      Else
        liref = 32*idim + 7
      End If
      ldquas = n
      Allocate (quas(ldquas,idim),iref(liref),xmean(idim),std(idim))

!     Read in the parameters for the distribution
      Read (nin,*) xmean(1:idim)
      Read (nin,*) std(1:idim)

!     Initialize the generator
      ifail = 0
      Call g05ylf(genid,idim,iref,liref,iskip,ifail)

!     Generate N values for the normal distribution
      ifail = 0
      Call g05ykf(xmean,std,n,quas,iref,ifail)

!     Display results
      Write (nout,99999)(quas(i,1:idim),i=1,n)

99999 Format (1X,4F10.4)
    End Program g05ykfe
```

## 10.2 Program Data

```
G05YKF Example Program Data
1               :: GENID
5  4  1000      :: N,IDIM,ISKIP
1.0 2.0 3.0 4.0 :: XMEAN
1.0 1.0 1.0 1.0 :: XSTD
```

## 10.3 Program Results

```
 G05YKF Example Program Results

     4.8648    9.4382    2.4979   21.5895
    17.7572    4.9813   41.8501  233.2386
     2.5195   20.5384   10.8353   45.3933
     1.8229    6.8823    6.9276   32.4808
     7.4938   49.7034   29.0198  127.4745
```