G05ADFP

NAG Parallel Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check for implementation-dependent details. You are advised to enclose any calls to NAG Parallel Library routines between calls to Z01AAFP and Z01ABFP.

1 Description

The real function G05ADFP generates a pseudo-random number from the Normal (Gaussian) distribution which has the probability density function (PDF)

$$f(x) = \frac{1}{\sqrt{2\pi b^2}} \exp\left(-\frac{(x-a)^2}{2b^2}\right).$$

A total of 273 statistically independent generators are available; it is possible to select a particular generator and initialize the seeds for the generator by a preceding call to G05BBFP. If G05BBFP is not used, default values for the generator and the seeds are assumed.

If more than one pseudo-random number is required then it may be more econmical to use G05BDFP. The routine G05BDFP always generates exactly the same pseudo-random numbers as would n consecutive calls of G05ADFP.

2 Specification

```
DOUBLE PRECISION FUNCTION G05ADFP(A, B)
DOUBLE PRECISION                  A, B
```

3 Usage

3.1 Definitions

None.

3.2 Global and Local Arguments

All arguments are local.

4 Arguments

1: A — DOUBLE PRECISION *Local Input*

2: B — DOUBLE PRECISION *Local Input*

On entry: the mean a and the standard deviation b of the distribution. It is not necessary that B is positive; if B is negative then a different sequence is generated but the PDF is the same. If B is zero then the generator outputs the value A.

5 Errors and Warnings

None.

6 Further Comments

Repeatable sequences of random numbers can be generated by calling G05BBFP to set the seeds and generator number before calling G05ADFP.

G05ADFP may be called without a prior call to Z01AAFP.

6.1 Algorithmic Detail

Each basic generator uses a Wichmann–Hill type generator (Wichmann and Hill [3]), which is a variant of a multiplicative congruential algorithm to produce real pseudo-random numbers u_k in the semi-open interval $[0, 1)$. See G05ACFP for further details.

Two consecutive pseudo-random numbers v_i and v_{i+1} from the normal distribution are computed using the polar Box–Müller method which requires two pseudo-random numbers (u_k and u_{k+1}) from the uniform distribution. However, due to a rejection phase, not all u_k are used in the algorithm for computing v_i.

7 References

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

[2] Maclaren N M (1989) The generation of multiple independent sequences of pseudorandom numbers *Appl. Statist.* **38** 351–359

[3] Wichmann B A and Hill I D (1982) AS183 An efficient and portable pseudo-random number generator *Appl. Statist.* **31** 188–190

8 Example

This example generates a random number on each processor on a 2 by 2 logical grid of processors. The routine G05BBFP is used to initialise the seeds and the generators.

8.1 Example Text

```
*     G05ADFP Example Program Text
*     NAG Parallel Library Release 3. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT, NX
      PARAMETER        (NOUT=6,NX=4)
      INTEGER          MAG
      PARAMETER        (MAG=16909320)
*     .. Local Scalars ..
      DOUBLE PRECISION A, B
      INTEGER          I, ICNTXT, ICOFF, IFAIL, IGEN, MP, MYCOL, MYROW,
     +                 NP, NPCOL, NPROW
      LOGICAL          ROOT
      CHARACTER        CNUMOP, TITOP
      CHARACTER*20     FORMT
*     .. Local Arrays ..
      DOUBLE PRECISION WORK(NX), X(NX)
      INTEGER          IS(5), ISEED(4), IWORK(5)
*     .. External Functions ..
      DOUBLE PRECISION G05ADFP
      LOGICAL          Z01ACFP
      EXTERNAL         G05ADFP, Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         G05BBFP, X04BFFP, X04BMFP, Z01AAFP, Z01ABFP,
     +                 Z01ZAFP
*     .. Intrinsic Functions ..
      INTRINSIC        MOD
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'G05ADFP Example Program Results'
         WRITE (NOUT,*)
      END IF
```

```
*
      MP = 2
      NP = 2
*
*     Declare the processor grid
*
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     Initialise the seeds and the generator
      CALL Z01ZAFP(ICNTXT,NPROW,NPCOL,MYROW,MYCOL)
*
*     Initialize the seeds and choose a generator number that depends
*     on the processor position on the grid.
*
      ISEED(1) = 207*(50*MYROW+19*MYCOL) + 2727269
      ISEED(2) = 451*(70*MYROW+31*MYCOL) + 7256253
      ISEED(3) = 912*(39*MYROW+53*MYCOL) + 5537890
      ISEED(4) = 812*(69*MYROW+14*MYCOL) + 2283939
      IGEN = NP*MYROW*5 + MYCOL*7*MP
*
*     Make sure that the seeds are within the maximum value MAG
*
      DO 40 I = 1, 4
   20    IF (ISEED(I).GT.MAG) THEN
            ISEED(I) = ISEED(I)/2
            GO TO 20
         END IF
   40 CONTINUE
*
*     Make sure that the generator is valid
*
      IGEN = MOD(IGEN,273)
*
*     Print the seeds and the generator on each processor
*
      IS(1) = ISEED(1)
      IS(2) = ISEED(2)
      IS(3) = ISEED(3)
      IS(4) = ISEED(4)
      IS(5) = IGEN
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Seeds and the generator'
         WRITE (NOUT,*)
      END IF
      FORMT = 'I10'
      TITOP = 'Y'
      CNUMOP = 'X'
      ICOFF = 0
      IFAIL = 0
      CALL X04BMFP(ICNTXT,NOUT,1,5,IS,1,FORMT,TITOP,CNUMOP,ICOFF,IWORK,
     +             1,IFAIL)
*
      CALL G05BBFP(ISEED,IGEN)
*
*     Set Average value and standard deviation
*
```

```
      A = 0.0D0
      B = 1.0D0
*
*     Now generate the random numbers
*
      DO 60 I = 1, 4
         X(I) = G05ADFP(A,B)
   60 CONTINUE
*
*     Print the random numbers on each processor
*
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +     'The generated random numbers on each processor'
         WRITE (NOUT,*)
      END IF
      FORMT = 'F12.5'
      TITOP = 'Y'
      CNUMOP = 'X'
      ICOFF = 0
      IFAIL = 0
      CALL X04BFFP(ICNTXT,NOUT,1,NX,X,1,FORMT,TITOP,CNUMOP,ICOFF,WORK,1,
     +             IFAIL)

      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
*
      END
```

8.2 Example Data

None.

8.3 Example Results

```
G05ADFP Example Program Results


Seeds and the generator

  Array from logical processor    0,   0

    2727269   7256253   5537890   2283939         0

  Array from logical processor    0,   1

    2731202   7270234   5586226   2295307        14

  Array from logical processor    1,   0

    2737619   7287823   5573458   2339967        10

  Array from logical processor    1,   1
```

```
   2741552   7301804   5621794   2351335        24

 The generated random numbers on each processor

   Array from logical processor    0,   0

       0.38145     0.28157    -0.47168    -0.72358

   Array from logical processor    0,   1

       0.61588     0.01511    -0.65125     0.73476

   Array from logical processor    1,   0

      -0.29989    -1.65519     0.26875     1.30004

   Array from logical processor    1,   1

      -0.95218    -0.79425     0.47239    -0.09816
```