G05AAFP

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

G05AAFP generates a pseudo-random number from a uniform distribution in the semi-open interval [0,1).

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 calling G05ABFP or G05BBFP before G05AAFP.

If G05ABFP or G05BBFP is not used, default values for the generator and seeds are assumed. In that case, generators on different logical processors will produce identical pseudo-random number sequences.

If it is required to produce more than one pseudo-random number at a time then the routine G05BAFP is more economical to use due to low overheads. However, G05BAFP generates pseudo-random numbers in the open interval (0,1) and not in the semi-open interval [0,1).

2 Specification

```
    DOUBLE PRECISION FUNCTION G05AAFP( )
```

3 Usage

3.1 Definitions

None.

3.2 Global and Local Arguments

Not applicable.

4 Arguments

None.

5 Errors and Warnings

None.

6 Further Comments

Repeatable sequences of pseudo-random numbers can be generated by calling G05ABFP to set the seeds and the generator number before starting each sequence.

G05AAFP 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_i:

$$\begin{aligned}
k_{1,i} &= (c_1 \times k_{1,i-1}) \bmod m_1 \\
k_{2,i} &= (c_2 \times k_{2,i-1}) \bmod m_2 \\
k_{3,i} &= (c_3 \times k_{3,i-1}) \bmod m_3 \\
k_{4,i} &= (c_4 \times k_{4,i-1}) \bmod m_4 \\
u_i &= \left(\frac{k_{1,i}}{m_1} + \frac{k_{2,i}}{m_2} + \frac{k_{3,i}}{m_3} + \frac{k_{4,i}}{m_4}\right) \bmod 1.0
\end{aligned}$$

where c_j and m_j, $j = 1,4$ are constant integers for each generator and $k_{j,i}$ on the left and right hand of the equations are newly generated integer seeds and old seeds, respectively. The constants c_j are in the range 112 to 127 and the constants m_j are prime numbers in the range 16718909 to 16776971, which are close to $2^{24} = 16777216$. These constants have been chosen so that they give good results with the spectral test, see Knuth [1] and Maclaren [2].

The period of each generator would be at least 2^{92} if it were not for common factors between $(m_1 - 1)$, $(m_2 - 1)$, $(m_3 - 1)$ and $(m_4 - 1)$. However, each generator should still have a period of at least 2^{80}. Further details of the generators can be obtained from NAG and further discussion of the properties of these generators is given in Maclaren [2] where it was shown that the generated pseudo-random sequences are essentially independent of one another according to the spectral test.

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

The example program generates a single pseudo-random number on each processor on a 2 by 2 logical grid of processors. **Note:** the default values for the seeds and generator are used on each processor (i.e., there is no call to G05ABFP), so the random number is identical on each processor.

8.1 Example Text

```
*     G05AAFP Example Program Text
*     NAG Parallel Library Release 3 Revised. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          ICNTXT, ICOFF, IFAIL, MP, NP
      LOGICAL          ROOT
      CHARACTER        CNUMOP, TITOP
      CHARACTER*20     FORMT
*     .. Local Arrays ..
      DOUBLE PRECISION WORK(1), X(1)
*     .. External Functions ..
```

```
      DOUBLE PRECISION G05AAFP
      LOGICAL          Z01ACFP
      EXTERNAL         G05AAFP, Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         X04BFFP, Z01AAFP, Z01ABFP
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'G05AAFP Example Program Results'
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'The random number on each processor'
         WRITE (NOUT,*)
      END IF
*
      MP = 2
      NP = 2
*
*     Declare the processor grid
*
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     Now generate a random number
*
      X(1) = G05AAFP()
*
*     Print the number on the root processor
*
      FORMT = 'F12.5'
      TITOP = 'Y'
      CNUMOP = 'X'
      ICOFF = 0
      IFAIL = 0
      CALL X04BFFP(ICNTXT,NOUT,1,1,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

```
G05AAFP Example Program Results

The random number on each processor

  Array from logical processor    0,   0

      0.20293

  Array from logical processor    0,   1

      0.20293

  Array from logical processor    1,   0

      0.20293

  Array from logical processor    1,   1

      0.20293
```