

G05AZFP

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

G05AZFP generates an integer pseudo-random number from a discrete uniform distribution over the closed interval $[m_1, m_2]$. The distribution for the random discrete variable I is given by

$$P(I = i) = \frac{1}{m_2 - m_1 + 1} \quad \text{if } m_1 \leq i \leq m_2,$$

$$P(I = i) = 0 \quad \text{otherwise,}$$

where it is assumed that $m_1 \leq m_2$.

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 economical to use G05BZFP. The routine G05BZFP always generates exactly the same pseudo-random numbers as would n consecutive calls of G05AZFP.

2 Specification

```
INTEGER FUNCTION G05AZFP(M1, M2)
INTEGER          M1, M2
```

3 Usage

3.1 Definitions

None.

3.2 Global and Local Arguments

All arguments are local.

4 Arguments

- 1: M1 — INTEGER *Local Input*
 2: M2 — INTEGER *Local Input*

On entry: the end points m_1 and m_2 of the discrete distribution. It is not necessary to have $M1 < M2$.

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 G05AZFP.

G05AZFP 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 G05AAFP for further details. If $m_1 < m_2$, the routine computes the values i_k from the discrete distribution via $i_k = m_1 + [(m_2 - m_1 + 1)u_k]$ where $[\]$ denotes the integer part; if $m_1 \geq m_2$ then i_k is computed with m_1 and m_2 interchanged.

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

```
*      G05AZFP Example Program Text
*      NAG Parallel Library Release 3. NAG Copyright 1999.
*      .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NX
      PARAMETER        (NX=1)
      INTEGER          MAG
      PARAMETER        (MAG=16909320)
*      .. Local Scalars ..
      INTEGER          ICNTXT, ICOFF, IFAIL, IGEN, J, M1, M2, MP, MYCOL,
+                   MYROW, NP, NPCOL, NPROW
      LOGICAL          ROOT
      CHARACTER        CNUMOP, TITOP
      CHARACTER*20     FORMT
*      .. Local Arrays ..
      INTEGER          I(NX), IS(5), ISEED(4), IWORK(5)
*      .. External Functions ..
      INTEGER          G05AZFP
      LOGICAL          Z01ACFP
      EXTERNAL         G05AZFP, Z01ACFP
*      .. External Subroutines ..
      EXTERNAL         G05BBFP, X04BMFP, Z01AAFP, Z01ABFP, Z01ZAFP
*      .. Intrinsic Functions ..
      INTRINSIC        MOD
*      .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'G05AZFP 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) = 107*(150*MYROW+18*MYCOL) + 2727390
      ISEED(2) = 351*(170*MYROW+30*MYCOL) + 8836384
      ISEED(3) = 812*(139*MYROW+52*MYCOL) + 3646749
      ISEED(4) = 712*(169*MYROW+13*MYCOL) + 3266384
      IGEN = 2*NP*MYROW + MYCOL*3*MP
*
*   Make sure that the seeds are within the maximum value MAG
*
      DO 40 J = 1, 4
20      IF (ISEED(J).GT.MAG) THEN
           ISEED(J) = ISEED(J)/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 the lower and upper limits of the distribution
*
      M1 = 1
      M2 = 49

```

```

*
*   Now generate random integers
*
*   I(1) = G05AZFP(M1,M2)
*
*   Print the numbers on the root processor
*
*   IF (ROOT) THEN
*       WRITE (NOUT,*)
*       WRITE (NOUT,*) 'Random integer on each processor'
*       WRITE (NOUT,*)
*   END IF
*   FORMT = 'I10'
*   TITOP = 'Y'
*   CNUMOP = 'X'
*   ICOFF = 0
*   IFAIL = 0
*   CALL X04BMFP(ICNTXT,NOUT,1,1,I,1,FORMT,TITOP,CNUMOP,ICOFF,IWORK,1,
+               IFAIL)
*
*   IFAIL = 0
*   CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
*   STOP
*
*   END

```

8.2 Example Data

None.

8.3 Example Results

G05AZFP Example Program Results

Seeds and the generator

Array from logical processor	0,	0			
	2727390	8836384	3646749	3266384	0
Array from logical processor	0,	1			
	2729316	8846914	3688973	3275640	6
Array from logical processor	1,	0			
	2743440	8896054	3759617	3386712	4
Array from logical processor	1,	1			
	2745366	8906584	3801841	3395968	10

Random integer on each processor

Array from logical processor	0,	0
7		
Array from logical processor	0,	1
3		
Array from logical processor	1,	0
10		
Array from logical processor	1,	1
33		
