

G05ABFP

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

G05ABFP selects a Wichmann–Hill [3] pseudo-random number generator g , ($0 \leq g \leq 272$) from the 273 available generators and initializes the seeds s_i , $i = 1,4$ for the generators on the Library Grid.

Either G05ABFP or G05BBFP is usually called before pseudo-random numbers are generated using the routines in this chapter.

2 Specification

```
SUBROUTINE G05ABFP(ICNTXT, ISEED, IGEN, IFAIL)
  INTEGER          ICNTXT, ISEED(4), IGEN, IFAIL
```

3 Usage

3.1 Definitions

None.

3.2 Global and Local Arguments

The global input (output) arguments must (will) have the same value on entry (on exit) to (from) the routine on each processor:

Global input arguments: IFAIL

Global output arguments: IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

Not applicable.

4 Arguments

1: ICNTXT — INTEGER *Local Input*

On entry: the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP.

Note: the value of ICNTXT **must not** be changed.

2: ISEED(4) — INTEGER array *Local Input/Local Output*

On entry: the seeds for the random number generator where

if $1 \leq \text{ISEED}(i) \leq 16909320$ then $s_i = \text{ISEED}(i)$;
 otherwise, $s_i = \text{mod}(1+|\text{ISEED}(i)|, 16909320)$.

On exit: s_i , $i = 1,4$, the actual seed used by the generators.

3: IGEN — INTEGER*Local Input/Local Output*

On entry: the generator number where

if $0 \leq \text{IGEN} \leq 272$ then $g = \text{IGEN}$;
 otherwise, $g = \text{mod}(|\text{IGEN}|, 273)$.

On exit: g , the actual generator used.

4: IFAIL — INTEGER*Global Input/Global Output*

The NAG Parallel Library provides a mechanism, via the routine Z02EAFP, to reduce the amount of parameter validation performed by this routine. For a full description refer to the Z02 Chapter Introduction.

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this argument (described in the Essential Introduction) the recommended values are:

IFAIL = 0, if multigridding is **not** employed;
 IFAIL = -1, if multigridding is employed.

On exit: IFAIL = 0 (or -9999 if reduced error checking is enabled) unless the routine detects an error (see Section 5).

5 Errors and Warnings

None.

6 Further Comments

The values of ISEED are saved internally by the routine. To generate different sequences of pseudo-random numbers on different processors, ensure that ISEED and/or IGEN are initialized to different values on each processor before starting the sequences.

If small seeds are supplied to different generators, the first few (typically 3 or 4) pseudo-random numbers produced by each generator could be similar. If it is important to avoid such behaviour then large (e.g., 6 digit) seeds should be supplied.

7 References

- [1] Maclaren N M (1989) The generation of multiple independent sequences of pseudorandom numbers *Appl. Statist.* **38** 351–359
- [2] Blackford L S, Choi J, Cleary A, D’Azevedo E, Demmel J, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, Stanley K, Walker D and Whaley R C (1997) ScaLAPACK Users’ Guide *SIAM* 3600 University City Science Center, Philadelphia, PA 19104-2688, USA. URL: http://www.netlib.org/scalapack/slug/scalapack_slug.html
- [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 series of random numbers on each processor on a 2 by 2 logical grid of processors. The seeds and the generator numbers are different on each processor so that statistically independent sequences are produced on each processor.

8.1 Example Text

```

*   G05ABFP Example Program Text
*   NAG Parallel Library Release 3 Revised. NAG Copyright 1999.
*   .. Parameters ..
INTEGER          NOUT
PARAMETER       (NOUT=6)
INTEGER          MX
PARAMETER       (MX=5)
INTEGER          MAG
PARAMETER       (MAG=16909320)
*   .. Local Scalars ..
INTEGER          GEN, I, ICNTXT, ICOFF, IFAIL, M, MP, MYCOL,
+               MYROW, NP, NPCOL, NPROW
LOGICAL          ROOT
CHARACTER        CNUMOP, TITOP
CHARACTER*20     FORMT
*   .. Local Arrays ..
DOUBLE PRECISION WORK(1,MX), X(1,MX)
INTEGER          IS(5), ISEED(4), IWORK(5)
*   .. External Functions ..
DOUBLE PRECISION G05AAFP
LOGICAL          Z01ACFP
EXTERNAL         G05AAFP, Z01ACFP
*   .. External Subroutines ..
EXTERNAL         G05ABFP, X04BFFP, X04BMFP, Z01AAFP, Z01ABFP,
+               Z01ZAFP
*   .. Intrinsic Functions ..
INTRINSIC        ABS, MOD
*   .. Executable Statements ..
ROOT = Z01ACFP()
IF (ROOT) THEN
    WRITE (NOUT,*) 'G05ABFP Example Program Results'
    WRITE (NOUT,*)
END IF

*
MP = 2
NP = 2

*
*   Declare the processor grid
*
IFAIL = 0
CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)

*
*   Initialize the seeds and choose a generator number that depends
*   on the processor position in the grid.
*
CALL Z01ZAFP(ICNTXT,NPROW,NPCOL,MYROW,MYCOL)
ISEED(1) = 1207*(4001*MYROW+1579*MYCOL) + 3145267
ISEED(2) = 2043*(3065*MYROW+3073*MYCOL) + 4976354
ISEED(3) = 1811*(3987*MYROW+1201*MYCOL) + 5143469
ISEED(4) = 2813*(2989*MYROW+2205*MYCOL) + 9956574

*
*   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
*
      GEN = NP*MYROW*4 + MYCOL*3
*
*   Make sure that the generator is valid
*
      GEN = MOD(ABS(GEN),273)
*
      IFAIL = 0
      CALL G05ABFP(ICNTXT,ISEED,GEN,IFAIL)
*
*   Print the four 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) = GEN
      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)
*
*   Now fill the vector with random numbers
*
      M = 5
      DO 60 I = 1, M
        X(1,I) = G05AAF()
60 CONTINUE
*
*   Print the random numbers on the root processor
*
      IF (ROOT) THEN
        WRITE (NOUT,*)
        WRITE (NOUT,*) '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,M,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

G05ABFP Example Program Results

Seeds and the generator

Array from logical processor	0,	0			
	3145267	4976354	5143469	9956574	0
Array from logical processor	0,	1			
	5051120	11254493	7318480	16159239	3
Array from logical processor	1,	0			
	7974474	11238149	12363926	9182315	8
Array from logical processor	1,	1			
	9880327	8758144	14538937	12283648	11

Random numbers on each processor

Array from logical processor	0,	0			
	0.91609	0.47716	0.96248	0.16606	0.62057
Array from logical processor	0,	1			
	0.59775	0.30849	0.11756	0.06097	0.79583
Array from logical processor	1,	0			
	0.53164	0.70465	0.85985	0.42821	0.87538
Array from logical processor	1,	1			
	0.29616	0.68851	0.32128	0.05088	0.05142