HSL_FA04

1 SUMMARY

This package generates **uniformly distributed pseudo-random numbers.** Random reals are generated in the range $0 < \xi < 1$ or the range $-1 < \eta < 1$ and random integers in the range $1 \le k \le N$ where N is specified by the user.

A multiplicative congruent method is used where a 31 bit generator word g is maintained. On each call to a procedure of the package, g_{n+1} is updated to $7^5 g_n \mathrm{mod}(2^{31}-1)$; the initial value of g is $2^{16}-1$. Depending upon the type of random number required the following are computed $\xi = g_{n+1}/(2^{31}-1)$; $\eta = 2\xi - 1$ or $k = \text{int.part}\{\xi N\}+1$.

The package also provides the facility for saving the current value of the generator word and for restarting with any specified value.

ATTRIBUTES — **Version:** 1.0.0. **Types:** HSL_FA04_SINGLE, HSL_FA04_DOUBLE. **Calls:** None. **Original date:** September 1995. **Origin:** N. I. M. Gould and J. K. Reid, Rutherford Appleton Laboratory. **Language:** Fortran 90. **Remark:** This supersedes FA04.

2 HOW TO USE THE PACKAGE

Access to the package requires a USE statement such as

Single precision version

```
USE HSL_FA04_SINGLE
```

Double precision version

```
USE HSL_FA04_DOUBLE
```

If it is required to use both modules at the same time, the subroutines FA04_RANDOM_REAL, FA04_RANDOM_INTEGER, FA04_GET_SEED, and FA04_SET_SEED (Section 2.1) must be renamed on one of the USE statements.

2.1 Argument lists and calling sequences

There are four procedures for user calls.

2.1.1 Subroutine to obtain a random real value

```
CALL FA04_RANDOM_REAL( POSITIVE, RANDOM_REAL )
```

POSITIVE is a scalar INTENT(IN) argument of type default LOGICAL. If POSITIVE is .TRUE., the generated random number is a real value in the range $0 < \xi < 1$, while if POSITIVE is .FALSE., the generated random number is a real value in the range $-1 < \eta < 1$.

RANDOM_REAL is a scalar INTENT(OUT) argument of type REAL (double precision REAL in HSL_FA04_DOUBLE). It is set to the required random number.

2.1.2 Subroutine to obtain a random integer value

```
CALL FA04_RANDOM_INTEGER( N, RANDOM_INTEGER )
```

N is a scalar INTENT(IN) argument of type default INTEGER. It must be set by the user to specify the upper bound for the range $1 \le k \le$ N within which the generated random number is required to lie. **Restriction:** N must be positive.

RANDOM_INTEGER is a scalar INTENT(OUT) argument of type default INTEGER. It is set to the required random integer k.

2.1.3 Subroutine to obtain the current generator word

```
CALL FA04_GET_SEED( SEED )
```

`SEED` is a scalar `INTENT(OUT)` argument of type default `INTEGER`. It is set to the current value of the generator word g.

2.1.4 Subroutine to reset the current value of the generator word

```
CALL FA04_SET_SEED( SEED )
```

`SEED` is a scalar `INTENT(IN)` argument of type default `INTEGER` that must be set by the user to the required value of the generator word. It is recommended that the value should have been obtained by a previous call of `FA04_GET_SEED`. It should have a value in the range $0 <$ `SEED` $\leq$ `P`, where `P` $= 2^{31} - 1 = 2147483647$. If it is outside this range, the value `SEED` $\mathrm{mod}(2^{31} - 1)$ is used.

3 GENERAL INFORMATION

Use of common: None.

Other modules used directly: None.

Input/output: None.

Restrictions: `N` > 0.

4 METHOD

4.1 Method description

The code is based on that of L.Schrage, ‘A More Portable Fortran Random Number Generator’, TOMS, **5**, 2, June 1979. The method employed is a multiplicative congruential method. The generator word g is held as an integer and is updated on each call as follows

$$g_{n+1} = 7^5 g_n \mathrm{mod}(2^{31} - 1)$$

The result returned from `FA04_RANDOM_REAL`, for a non-negative argument, is ξ, where

$$\xi = g_{n+1}/(2^{31} - 1)$$

and for a negative argument is

$$2\xi - 1$$

The value of k returned by `FA04_RANDOM_INTEGER` is

$$\text{int.part}\{\xi N\} + 1$$

4.2 Comparison with FA01A

`FA04_RANDOM_REAL` provides the Fortran user with a random number generator that has a cycle length of $2^{31} - 1$, which is twice as long as the cycle length of `FA01A`.

5 EXAMPLE

Suppose we wish to generate two random real numbers lying between plus and minus one, reset the generator word to its original value, and then find two positive random integers with values no larger than one hundred. Then we might use the following piece of code.

```
      PROGRAM HSL_FA04_SPEC
```

```
        USE HSL_FA04_DOUBLE
        IMPLICIT NONE
        INTEGER :: random_integer, seed
        REAL ( kind = KIND( 1.0D+0 ) ) :: random_real
  !  Get the current generator word
        CALL FA04_GET_SEED( seed )
        WRITE( 6, "( ' generator word = ', I10 )" ) seed
  !  Generate a random real in [-1, 1]
        CALL FA04_RANDOM_REAL( .FALSE., random_real )
        WRITE( 6, "( ' random real = ', F10.2 )" ) random_real
  !  Generate another random real
        CALL FA04_RANDOM_REAL( .FALSE., random_real )
        WRITE( 6, "( ' second random real = ', F10.2 )" ) random_real
  !  Restore the generator word
        CALL FA04_SET_SEED( seed )
  !  Generate a random integer in [1, 100]
        CALL FA04_RANDOM_INTEGER( 100, random_integer )
        WRITE( 6, "( ' random integer = ', I3 )" ) random_integer
  !  Generate another random integer
        CALL FA04_RANDOM_INTEGER( 100, random_integer )
        WRITE( 6, "( ' second random integer = ', I3 )" ) random_integer
        END PROGRAM HSL_FA04_SPEC
```

This produces the following output:

```
 generator word =      65535
 random real =       0.03
 second random real =      -0.34
 random integer =  52
 second random integer =  33
```