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 $7^5 g_n \bmod (2^{31}-1)$; 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:** 1.1.0. (12 July 2004) **Types:** Real (single, double). **Calls:** None. **Original date:** March 2001. **Origin:** N. I. M. Gould and J. K. Reid, Rutherford Appleton Laboratory. **Language:** Fortran 95. **Remark:** This is a threadsafe version of `HSL_FA04` and supersedes it.

2 HOW TO USE THE PACKAGE

Access to the package requires a `USE` statement such as

Single precision version

```
USE HSL_FA14_single
```

Double precision version

```
USE HSL_FA14_double
```

If it is required to use both modules at the same time, the subroutines `FA14_random_real`, `FA14_random_integer`, `FA14_get_seed`, and `FA14_set_seed` (Section 2.1) must be renamed on one of the `USE` statements. Their seeds will be independent.

2.1 The derived data type

The user must provide a variable of derived type `FA14_seed` to hold the current seed value which must be passed to all calls of to `FA14`. The seed value component is private and can only be set and retrieved through the `FA14_set_seed` and `FA14_get_seed` entries.

2.2 Argument lists and calling sequences

There are five procedures for user calls.

2.2.1 Subroutine to initialize the generator word

This entry may be called to initialize the generator word. This is not necessary with the present version and is included only for compatibility with the Fortran 90 version that was present in HSL 2004.

```
CALL FA14_initialize( seed )
```

`seed` is a scalar `INTENT(OUT)` argument of derived type `FA14_seed` which holds the seed value.

2.2.2 Subroutine to obtain a random real value

```
CALL FA14_random_real( seed, positive, random_real )
```

`seed` is a scalar `INTENT(INOUT)` argument of derived type `FA14_seed` which holds the seed value.

`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_FA14_double). It is set to the required random number.

2.2.3 Subroutine to obtain a random integer value

```
CALL FA14_random_integer( seed, n, random_integer )
```

seed is a scalar INTENT(INOUT) argument of derived type FA14_seed which holds the seed value.

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 $1 \leq k \leq$ 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*.

2.2.4 Subroutine to obtain the current generator word

```
CALL FA14_get_seed( seed, value )
```

seed is a scalar INTENT(IN) argument of derived type FA14_seed which must be provided to hold the seed value.

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

2.2.5 Subroutine to reset the current value of the generator word

```
CALL FA14_set_seed( seed, value )
```

seed is a scalar INTENT(OUT) argument of derived type FA14_seed which holds the seed value.

value 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 FA14_get_seed. It should have a value in the range $0 <$ value $\leq$ P, where P = $2^{31} - 1$ = 2147483647. If it is outside this range, the value value 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

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

The result returned from FA14 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 FA14_random_integer is

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

4.2 Comparison with FA01A

FA14_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_FA14_spec
      USE HSL_FA14_double
      IMPLICIT NONE
      TYPE (FA14_seed) seed
      INTEGER :: random_integer, value
      REAL ( kind = KIND( 1.0D+0 ) ) :: random_real
!  Get the current generator word
      CALL FA14_get_seed( seed, value )
      WRITE( 6, "( ' generator word = ', I10 )" ) value
!  Generate a random real in [-1, 1]
      CALL FA14_random_real( seed, .FALSE., random_real )
      WRITE( 6, "( ' random real = ', F10.2 )" ) random_real
!  Generate another random real
      CALL FA14_random_real( seed, .FALSE., random_real )
      WRITE( 6, "( ' second random real = ', F10.2 )" ) random_real
!  Restore the generator word
      CALL FA14_set_seed( seed, value )
!  Generate a random integer in [1, 100]
      CALL FA14_random_integer( seed, 100, random_integer )
      WRITE( 6, "( ' random integer = ', I3 )" ) random_integer
!  Generate another random integer
      CALL FA14_random_integer( seed, 100, random_integer )
      WRITE( 6, "( ' second random integer = ', I3 )" ) random_integer
      END PROGRAM HSL_FA14_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
```