RANECU	CERN Program Library	V114

Author(s) : P. l'Ecuyer **Library:** MATHLIB
Submitter : F. Carminati **Submitted:** 27.02.1989
Language : Fortran **Revised:**

Uniform Random Number Generator

RANECU generates a sequence of uniformly distributed random numbers in the interval (0,1). The numbers are returned in a vector. Several independent sequences can be initialized and used in the same run.

Structure:

SUBROUTINE Subprograms
User Entry Names: RANECU, RANECQ
COMMON Block Names and Lengths: /RANEC1/ 402

Usage:

```
CALL RANECU(VEC,LEN,ISEQ)
```

VEC (REAL) Array of length LEN at least. On exit, it will contain the in (0,1) uniformly distributed random numbers.

LEN (INTEGER) Number of random numbers wanted. Unchanged on exit.

ISEQ (INTEGER) Number of the independent sequence from which the LEN numbers should be extracted. If ISEQ $\leq$ 0 then the extraction will be made from the sequence used last. Unchanged on exit.

Several independent sequences can be defined and used. Each sequence MUST be initialized by the user, otherwise the result is unpredictable. By default the routine contains a space buffer to handle only one sequence. If more sequences are needed, then a bigger buffer should be allocated in the calling program defining the COMMON block /RANEC1/ appropriately. *Two* words have to be allocated plus *four* words for every sequence initialized.

Two integer seeds are used to initialize a sequence. Not all pairs of integers define a good random sequence or one which is independent from others. Sections of the same random sequence can be defined as independent sequences. The period of the generator is $2^{60} \approx 10^{18}$. A generation has been performed in order to provide the seeds to start any of the generated sections. There are 100 possible seed pairs and they are all 10^9 numbers apart. Thus a sequence started from one of the seed pairs, after 10^9 numbers will start generating the next one. Each of these sequences is of the same order of magnitude as the basic sequence offered by RNDM (V104). Longer sequences will be generated and the corresponding seeds made available to users. Note that, while the numbers generated by the default sequence will always be the same, the introduction of more sequences may modify some of them. In order to handle the initialization of the package, the following routine is provided:

```
CALL RANECQ(ISEED1,ISEED2,ISEQ,CHOPT)
```

`ISEED1` (`INTEGER`) On entry, it contains the first integer seed from which to start the sequence. Unchanged on exit.

`ISEED2` (`INTEGER`) On entry, it contains the second seed from which to start the sequence. Unchanged on exit.

`ISEQ` (`INTEGER`) On entry, it contains the number of the independent sequence of random numbers to be addressed by this call. If `ISEQ` $\leq$ `0`, then the last valid sequence used will be addressed either for a save or a store. In case the option `'R'` is specified, on output the variable will contain the sequence actually used.

`CHOPT` (`CHARACTER*1`) A character specifying the action which `RANECQ` should take. Possible options are:

- `' '` If `1` $\leq$ `ISEQ` $\leq$ `100`, the sequence number `ISEQ` will be initialized with the default seeds of the pre-computed independent sequence number `ISEQ`. `ISEED1` and `ISEED2` are ignored.

  If `ISEQ` $\leq$ `0` or `ISEQ` $>$ `100`, then sequence number `1` will be initialized with the default seeds. `ISEED1` and `ISEED2` are ignored.
- `'R'` Get the present status of the generator. The two integer seeds `ISEED1` and `ISEED2` will be returned for sequence `ISEQ`.
- `'S'` Set the status of the generator to a previously saved state. The two integer seeds `ISEED1` and `ISEED2` will be used to restart the generator for sequence `ISEQ`.
- `'Q'` Get the pre-generated seeds for `ISEQ` (`1` $\leq$ `ISEQ` $\leq$ `100`). There are 100 pre-generated sequences each one will generate 10^9 numbers before reproducing the following one.

Timing:

Time in μsec for extractions:

Extractions per call	1	4	16	128
Apollo 10000	6.2	4.4	3.9	3.8
Apollo 4000	52	37	34	33
IBM 3090E	4.9	2.9	2.5	2.4
IBM 3090EVF	3.4	2.3	2.0	1.8
Cray X-MP/48	4.2	2.2	1.7	1.5
VAX 8650	19	13	12	11.6

References:

1. P. l'Ecuyer, Efficient and Portable Random Number Generators, Comm. ACM **31** (1988) 742.
2. F. James, A Review of Pseudorandom Number Generators, Computer Phys. Comm. **60** (1990) 329–344.

•