Author(s) : G. Marsaglia, A. Zaman
Submitter : F. Carminati, F. James
Language : Fortran

Library: MATHLIB
Submitted: 08.06.1989
Revised:

Fast Uniform Random Number Generator

RANMAR generates a sequence of 32-bit floating-point random numbers uniformly distributed in the interval (0,1), the end points excluded. These numbers are returned in a vector. The period is about 10^{43} and the quality is good but it fails some tests. For better quality use RANLUX (V115), which is slower.

Several independent sequences can be initialized and used in the same run.

Structure:

SUBROUTINE subprograms
User Entry Names: RMMAR, RMMAQ, RANMAR, RMARIN, RMARUT
COMMON Block Names and Length: /RANMA1/ 104, /RANMA2/ 104

Usage:

For a single sequence:

```
CALL RANMAR(VEC,LEN)
```

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 to be generated. Unchanged on exit.

The initialization is made by

```
CALL RMARIN(IJKLIN,NTOTIN,NTO2IN)
```

IJKLIN (INTEGER) Seed from which to start the sequence. Every integer number from 1 to 900 000 000 originates an independent sequence of random numbers with operand of 2^{144} (about 10^{43}).

NTOTIN (INTEGER) Number (mod 10^9) of random number generated.

NTO2IN (INTEGER) Billions (10^9) of random numbers generated.

The arguments NTOTIN and NTO2IN are used to restart the generation from a given point by skipping over already performed extractions. They are returned by RMARUT and should not be touched by the user.

```
CALL RMARUT(IJKLUT,NTOTUT,NTO2UT)
```

IJKLUT (INTEGER) Seed from which the sequence was started.

NTOTUT (INTEGER) Number (mod 10^9) of random number generated so far.

NTO2UT (INTEGER) Billions (10^9) of random numbers generated so far.

For multiple sequences:

```
CALL RMMAR(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 to be generated. Unchanged on exit.

`ISEQ` (`INTEGER`) Number of the independent sequence from which the `LEN` numbers should be extracted. If $\leq$ `0`, the last valid sequence explicitely defined is used. 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 buffer of space to handle only one sequence. If more sequences are needed, then a bigger buffer should be allocated in the main program defining the `COMMON` block `/RANMA2/` to the appropriate size. The space needed is 1 word + 103 words for every random sequence initialized.
The sequences are initialized by

```
CALL RMMAQ(ISEED,ISEQ,CHOPT)
```

`ISEED` (`INTEGER`) Array of length `3` or `103` according to the option specified in `CHOPT`. The first location contains the integer seed from which to start the sequence. Every integer number from `1` to `900 000 000` originates an independent sequence of random numbers, with a period of 2^{144} (about 10^{43}). The second and the third location contain numbers used internally to re-initialize the generator by skipping and should not be touched by the user. The other numbers are a snapshot of the complete status of the generator. If saved, they can be used to restart the generator without skipping over numbers already generated.

`ISEQ` (`INTEGER`) This variable contains, on entry, the number of the independent random number sequence which should be addressed by the present call. If $\leq$ `0`, then the last valid sequence used will be addressed either for a save or a store. If option `'R'` is specified, on exit the variable will contain the sequence actually used.

`CHOPT` (`CHARACTER`) Specifies the action which `RMMAQ` should take. Possible options are:

- `' '` (Blank) The sequence number `1` will be initialized with a default seed. All arguments are ignored.
- `'R'` Get the present status of the generator. If option `'V'` is also present, then the complete status of the generator will be dumped in the array `ISEED`. This options will use 103 words in `ISEED` but has the advantage that the generator can be restarted immediately without skipping numbers. If option `'V'` is not present, then only 3 words will be used but the generator will have to be restarted by skipping the number of events generated so far.
- `'S'` Set the status of the generator to a previously saved state. If option `'V'` is also present, then an array `ISEED` of 103 words is expected, which comes from a previous call to the routine with option `'RV'`. This kind of initialization is very fast. If the option `'V'` is not specified then the generator will be restarted regenerating the same number of random extractions it generated at the time the status was saved. In this case only the first 3 locations of `ISEED` will be used.
- `'V'` Vector option. 103 words will be saved/restored. This allows to restart the generator without skipping over numbers already generated.

For `RMMAR` one seed is needed to initialize the random number, but it is a one-way initialization. The seed cannot be output and used to restart the sequence. In order to restart the generation, the number of random numbers generated is recorded by the generator. The sequence is restarted either generating this many random numbers or saving and restoring a vector of 103 words. The number of generations is stored in the two array elements `ISEED(2),ISEED(3)` as the period is bigger than the maximum number which can be represented by a 32-bit integer.

Timing:

Time in μsec for extractions and skips:

Extractions per call	1	4	16	128	$1000/10^5$ skips
APOLLO 10000	7.4	6.0	5.6	5.5	15/4.6
APOLLO 4000	69	55	51	50	120/73
IBM390E	4.3	2.5	2.0	1.9	7.4/1.2
CRAY X-MP/48	4.1	2.1	1.7	1.5	6.9/1.6
VAX8650	14	7.3	5.9	5.8	4.7/4.6

References:

1. G. Marsaglia and A. Zaman, Toward a Universal Random Number Generator, Florida State University FSU-SCRI-87-50 (1987).
2. F. James, A Review of Pseudorandom Number Generators, Computer Phys. Comm. **60** (1990) 329–344.

•