CMST 5(1) 81-85 (1999) DOI:10.12921/cmst.1999.05.01.81-85

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY 5, 81-85 (1999)

PSEUDORANDOM NUMBER GENERATORS BASED ON THE WEYL SEQUENCE

K. W. WOJCIECHOWSKI

Institute of Molecular Physics, Polish Academy of Sciences
Smoluchowskiego 17/19, 60-179 Poznań, Poland
kww@man.poznan.pl

Abstract: Some modifications of the pseudorandom number generator proposed recently by Holian et al. [B. L. Holian, 0. E. Percus, T. T. Warnock, P. A. Whitlock, Phys. Rev. E50,1607 (1994)] are presented.

1. INTRODUCTION

Random numbers are useful in all the areas of human activity to which probabilistic methods can be applied. Although truly random numbers of various distributions can be generated in many ways by physical systems exhibiting chaotic properties[1], using them can be inconvenient or expensive when the generated sequences of random numbers have to be long enough and possibility of repetition of these sequences is required[2]. Recent development of fast computers caused an increasing interest in *deterministic algorithms* generating sequences of numbers which reproduce (as close as possible) many statistical properties of random numbers. The numbers produced by such algorithms are known as *pseudorandom numbers* (PRNs). Various generators of pseudorandom numbers have been discussed in the literature, see e. g. [1-4].

It follows from the definition that no generator of PRNs can be perfect, i. e. *no PRN generator can be successfully used for all possible applications*[3]. Hence it is meaningful to introduce the notion of quality of generators with respect to particular classes of applications [5], Better generators have broader spectrum of applications for which they give correct results.

From a point of view of a practitioner the quality of a PRN generator depends on the speed of the generator, its memory usage, ease of its implementation, portability of its implementations, possibility to parallelize (distribute) the calculations using it, period length, structural properties, correlations, etc. [1-3],

The speed, parallelizability, and small memory requirements are crucial for possibility to apply a given PRN generator to large scale computer simulations. Recently, a simple and interesting, in this context, generator which bases on the shuffled nested Weyl sequence (SNWS) has been proposed by Holian and co-workers [6]. The SNWS generator was designed for large scale molecular dynamics simulations. Its modification, designed for Monte Carlo (MC)

---

[1] These are, e.g., radioactive materials, various electronic devices or mechanical systems.

[2] This is, e.g., the case of Monte Carlo computer simulations.

[3] The deterministic algorithm generating the pseudorandom numbers can be always used to construct a "test" in which the generator will produce wrong results.

simulations, has been tested in [5]. The behaviour of both the original SNWS generator [6] and its modification described in [5] depends on the computer implementation of the floating point arithmetic. In the case of the modified SNWS generator this reduces the maximal length of a sequence[4] of PRNs to about 10^{14}. Existence of some very long-range correlations is, however, possible in such a sequence. In the present note we give some hints how to reduce such correlations, and how to increase the maximal length of the sequence.

2. THE SNWS GENERATOR AND ITS MODIFICATIONS

The pseudo-code for the original SNWS generator can be written as follows [6]:

```
MULT=  1234567
SEED  =  MOD(SQRT(2.), 1.)
  DO N = 1,NUMBER                                          (*)
  X(N)  =  MOD(N*MOD(N*SEED, 1.), 1.) *MULT +  0.5
  X(N)  =  MOD(X(N) *MOD(X(N) *SEED, 1.), 1.)
ENDDO
```

For comparison, the pseudo-code of the modified SNWS generator used in [5] writes:

```
LENGTH =  1000003
BIGN =  MOD(EXP(L), 1.) *10^7
DO N =  1,NUMBER
  ISEED  =ISEED  +1
  SEED  =  MOD(SQRT(ISEED), 1.)
  IF(SEED.EQ.O)  THEN
     ISEED=ISEED +1                                        (**)
     SEED  =  MOD(SQRT(ISEED), 1.)
  ENDIF
  DO K =  1, LENGTH
     X=  MOD(K*MOD(K*SEED, 1.), 1.) *BIGN +0.5
     X=  MOD(X*MOD(X*SEED, 1.), 1.)
  ENDDO
ENDDO
```

[4] For arithmetic of infinite precision, the maximal length of a sequence of PRNs produced by the SNWS generator is equal to infinity. However, the round-off errors caused by the finite precision of the computer floating point arithmetic reduce this length to a finite number depending on the precision used. For *SINGLE PRECISION* arithmetic this length is obviously much smaller than for the *DOUBLE PRECISION* arithmetic for which it is, in turn, much smaller than for the *QUADRUPLE PRECISION* arithmetic. Since, however, the latter arithmetic is performed by present computers much slower than the former ones, further discussion in this paper concerns the *DOUBLE PRECISION* case.

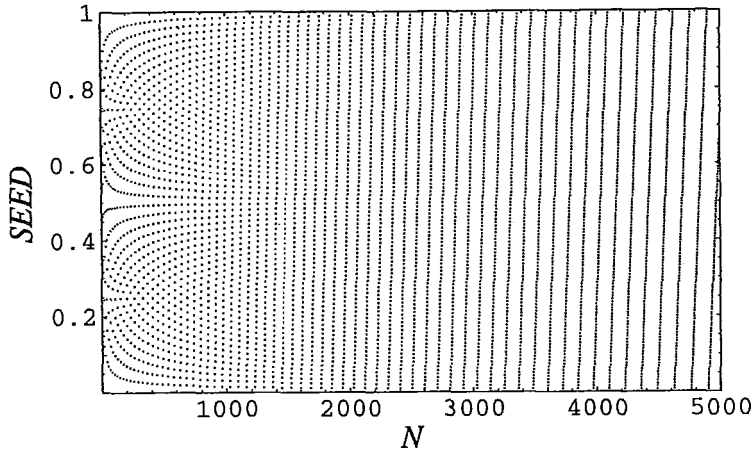

Fig. 1. The dependence of the variable *SEED* on *N* for the generator of the pseudo-code (**)

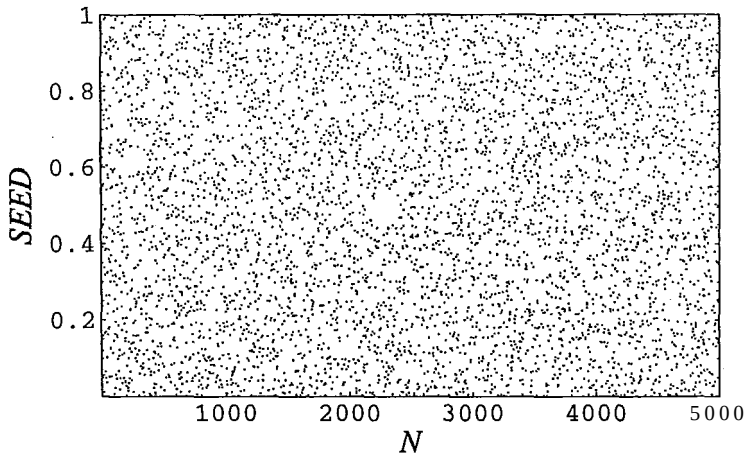

Fig. 2. The same dependence as in Fig. 1 but for the pseudo-code containing the modification (***).

As it is easy to notice, the essence of the difference between the above generators consists in fixing the variable *SEED* in the first example and varying it in the second example. It can be seen, however, in Fig. 1 that in the case (**) the values of this variable exhibit strong correlations producing some clearly visible patterns in the plot of the dependence of *SEED* on *N*. One can check (see Fig. 2) that these patterns disappear when replacing the fifth and the eighth lines of the pseudo-code (**) by the following line:

*SEED=MOD((1. +SIN(ISEED)) *SQRT(ISEED), 1.)* (***)

It is not difficult to check that many other modifications of the SNWS generator in which *SEED* is obtained by using various combinations of nonlinear and oscillatory functions of *ISEED*

do not reveal any simple patterns in the plots analogous to Fig. 1. Some of them can be used to increase the length of the maximal sequence of the generated PRNs. As an example we present the following algorithm:

```
LENGTH = 1000003
BIGN = MOD(EXP(l.), 1.) *10^7
SCALE = 29
SECOND = SQRT(2.).
ISEED1 = 1
SEED2 = 1
DON1 = 1,NUMBER
  ISEED1 = ISEED1+1
  DON2 = 1,NUMBER
     ISEED2 = ISEED2+1
     SEED = SCALE *MOD((l. +SIN(ISEED1)) *SQRT(ISEED1)+
                (l.+SIN(MOD(ISEED2,SECOND))) *SQRT(ISEED2), 1.)
     IF(SEED.EQ.O) THEN
        ISEED2 = ISEED2+1                                          (****)
        SEED = SCALE*MOD((l. +SIN(ISEED1)) *SQRT(1SEED1)+
                (1. +SIN(MOD(ISEED2,SECOND))) *SQRT(ISEED2), 1.)
     ENDIF
     DO K= 1,LENGTH
        X = MOD(K*MOD(K*SEED, 1.), 1.) *BIGN +0.5
        X= MOD(X*MOD(X*SEED, 1.), 1.)
     ENDDO
  ENDDO
ENDDO
```

We estimate that the longest sequences of the PRNs generated by the above algoritm exceed 10^{20} numbers. This is more than enough for the present large scale MC simulations. The maximum length of generated sequences of PRNs can be increased even further, e. g. by introducing higher than *SECOND* incommensurate periods.

We should stress that the presented above modifications of the SNWS generator do not reduce its very high speed [5]. This is because these modifications do not touch its most internal loop which has the length of order 10^6 cycles.

3. FINAL REMARKS

Some "practical" tests, which consist of large scale MC computations of various physical properties of certain many-body systems, were performed. They have shown that the results obtained for the above described generalizations of the SNWS generator are in good agreement with those obtained by using a few other (well known) random number generators.

There exist infinitely many ways to generate pseudorandom values of the variable *SEED* which is used in the internal loop of the described, modified SNWS algorithms. One of the simplest idea, which has not been discussed in this paper would be to exploit other (well known) PRN generators to reach this goal.

Acknowledgments

This work was supported by the Committee for Scientific Research (KBN) within the grant 8T11F 01214.

References

[1] D. E. Knuth, *Seminumerical Algorithms*, 3rd edition, vol. 2 of: *The Art of Computer Programming*, Addison-Wesley, Reading, MA (1998).
[2] P. L'Ecuyer, Ann. Oper. Res. 53, 77 (1994).
[3] P. Hellekalek, Mathematics and Computers in Simulation 46, 485 (1998).
[4] M. Matsumoto and T. Nishimura, *ACM Transactions on Modelling and Computer Simulations*: *Special Issue on Uniform Random Number Generation* (1998).
[5] K. V. Tretiakov and K. W. Wojciechowski, Phys. Rev. E 60, 7626 (1999).
[6] B. L. Holian, O. E. Percus, T. T. Warnock, P. A. Whitlock, Phys. Rev. E 50, 1607 (1994).