数理解析研究所講究録
第 537 巻 1984 年 112-122

Random Number Generation in Large-Scale Monte Carlo Calculations

Yoshio Oyanagi
Institute of Information Sciences,
University of Tsukuba
筑波大学 電子情報工学系 小柳義夫

Various problems in generating large number of pseudorandom numbers on a "supercomputer" is discussed. As a computational physicist I am interested in a Monte Carlo simulation of physical systems with very large degrees of freedom, such as lattice gauge theory. I hope some of you could help us in solving the problems.

1. Monte Carlo Simulation

Historically, computing has played only a background role in the development of theoretical physics. In the last few years, however, a new use of computer has emerged in "theoretical physics". Monte Carlo simulation has become a powerful tool for studying the solution to quantum field theories. These calculation have concentrated on the gauge theory of the strong interactions, called QCD.

The techniques have been borrowed from the solid-state physicists. We can see the ties between the quantum physics and the statistical physics via Feynman´s path-integral formulation of quantum mechanics, in which quantum field theory in four euclidean space-time dimensions is equivalent to the classical statistical mechanics of a hypothetical thermal system with four spatial dimensions.

The solid-state physicists has long used the Monte Carlo method to simulate classical statistical systems. The computer memory contains the numerical values for the degrees of freedom of the system -- for example, the direction of the spins in a model of magnetism. A random number generator, weighted by the Boltzmann factor exp(-E/kT), then induces changes that simulate thermal evolution and fluctuation. By "measuring" various statistical quantities of this system (energy, specific heat, susceptibility, correlation,...) the physicists gains insight into macroscopic phenomena such as phase transitions.

Particle theorists use the Monte Carlo technique as numerical method for evaluating Feynman path integral. In order to make the integral mathmatically well defined, we must discretized the continuous space. K. Wilson[1] provided a scheme, called lattice gauge theory, by formulating gauge theories on a four-dimensional hypercubic lattice. In this scheme, integral over all space-time points are replace by sums over a lattice of discrete space-time points. Thus the field simply becomes arrays in the computer memory.

Computer technology is escalating at a surprising rate. Ten years ago no one imagined the power of Monte Carlo methods for theoretical physics. But now we use 200-1000 hours of CPU time of contemporary multi-purpose high speed computers for only one paper.[2] The total number of random numbers generated in such calculation is 10^9-10^{12}, which often exceeds the period of multiplicative congruential method for modulus 2^{31}.

2. Processor Array PAX

There are two types of high speed supercomputers: pipeline type (vector processor) and processor array. First I will discuss on the processor array.

The ever increasing power of microprocessors and the continuing decrease in the cost enable us to construct an array of microprocessors for large scale scientific computations, where the performance increases in proportion to the number of processors. In many physical problems, the proximity property that the interaction is limited among nearby degrees of freedom leads to the possibility of a very high speed simulation by a parallel array-type computer, where each processing unit (PU) is connected only with the nearest neighbor PU´s.

We are developing a parallel computer system called "PAX" since 1977, and presently installed system has 128 PU´s with the speed of about 4 MFLOPS[3]. The 128 PU´s in the PAX system are connected in a two-dimentional nearest-neighbor mesh(NNM) rectangular array with the periodic boundary condition. The tasks allocated in each PU are executed asynchronously.

In the simulation of spin[4] or gauge systems, the whole physical space is directly projected onto the PU array, so that each PU takes care of the physical process going in each subregion assigned to that PU. When the program refers to nonlocal data stored in the nearby or distant PU´s, the data are either transferred from the nearest-neighbor PU via the communication memory or are broadcasted via the control unit.

3. Random Number Generation on PAX

Since these data move procedures must be preceded by the synchronization of all PU´s to avoid the memory access contention, they increase the overhead of idling. It is therefore required that the random numbers should be generated independently in each PU, i.e. without referring to nonlocal data. How can we provide 128 series of random numbers? Not only should these series be "good" random numbers themselves, but the mutual correlation should be small. Especailly, a large correlation between the random numbers generated in the nearest neighbors would be fatal for the simulation.

In the case of the M-sequence method of the Lewis-Payne type, an algorithm for a correlation-free initial value setting has been proposed by Fushimi.[5] How can we do in the case of congruential methods?

The series of multiplicative congruential random numbers with the same multiplier, say 48828125, starting from "arbitrary" 128 initial values, say 1415, 9265, 8979, 323$\underline{9}$, 462$\underline{7}$, 433$\underline{9}$, 3279, ... (decimal sequence of pi; the underlined digits are changed), was first used. It might cause trouble. For example, I numerically found that the series starting from 555 and 777 are strongly correlated (r=0.03125). The correlati$\overset{o}{\wedge}$n over the whole period was estimated as 0.0286 in terms of the Dedekind sum.[6]

One would argue that we might change the multipliers PU by PU. But how could we find 128 good multipliers which pass various statistical tests and the spectral test?[7] Moreover, it is difficult to estimate the correlation over the whole period between the two series with different multipliers.

In the case of the ILLIAC IV, a pioneering parallel computer of the SIMD type with 64 processing elements (PE), the whole period of the mixed congruential random numbers for modulus 2^{48} was equally divided into 64 subsequences and each subsequence is allocated to one PE. Namely, the starting values for each PE are separated by just 2^{42}. This algorithm is dangerous, since, as is well known, every 2^{24} terms in a multiplicative or mixed congruential random number series with modulus 2^{48} have an equal difference modulo 2^{48}. The random number series in each PE would be mutually strongly correlated unless disordered by a masking option.

I will propose a simple method to choose the multiplier and to set up the initial values for the PU´s. The idea is to simulate the random number generation on an ordinary von Neumann-type computer. Let the sequence of random numbers which are used in a simulation of a system with 128 degrees of freedom be

$$R(i)=c*R(i-1) \pmod{m} \quad i=1,2,......$$

where m is typically 2^{31}. At the first iteration, random numbers R(1)-R(128) are used. At the second iteration, R(129)-R(256) are used, and so on. From the standpoint of the i-th degree of freedom, every 128 numbers, R(i), R(i+128), R(i+256), ... are used as random numbers. This partial series can be calculated directly in terms of the relation,

$$R(i+128)=cc*R(i) \pmod m,$$

where $cc=c^{128}$ (mod m). In the case of a processor array, if we take the initial values for each PU as R(1)-R(128) as above and the multiplier for the PU as cc, we can simulate on PAX the random number generation on a sequential computer. However, the random number sequence in a PU generated as above is **NOT** a good random number itself, since c=3 or 5 (mod 8) implies $c^{128}=1$ (mod 8). I will propose to take $cc=c^{129}$ instead. Thus the period is maximal (2^{29}).

The nearest neighbor PU´s are separated by 1 or 16 (except the boundary PU´s) in the case of PAX-128. How large is the correlation between the random numbers starting from R(i) and R(i+1) or R(i+16)? Are there better ways to scatter the 128 initial values on PU´s in order to make the correlation between the near-by random numbers small, without just putting them one by one from left to right? I hope the number theorists have comments.

4. Random Numbers on a Pipeline Supercomputer

At the present time, commercial supercomputers such as CRAY-1, Cyber-205, FACOM VP-200, HITAC S810/20, ... are all pipeline computers. In these machines operations on vectors are processed on a special hardware (vectorized), e.g.

```
          A(*)=A(*)+c*B(*)   (elementwise scalar mult and add)
          s=s+A(*)*B(*)      (inner product).
```

In HITAC S810, the first order iteration

```
          do 10 i=1,n
       10 IR(i)=c*IR(i-1)+d
```

is also vectorized. Both multiplicative and mixed congruential random numbers are thus generated by a vector operation. Unfortunately, however, this iterative operation is very slow even if it is vectorized, because the calculation for different i cannot be done parallelly.

In order to speed up the generation, the number of such iterative operation should be minimized. If the vector length n is nearly fixed, the method proposed for PAX is also applicable here. At the first stage, n random numbers are prepared by the first order iteration

```
          do 10 i=1,n
       10 IR(i)=c*IR(i-1)
```

This stage is not very fast, but it is excecuted only once. After that the n (or less) random numbers are generated as:

```
          do 20 i=1,n
          IR(i)=cc*IR(i)
          ...=...IR(i)....
       20 continue
```

where cc=c**n or c**(n+1) if n is even. The operations in this loop is really parallel and is processed in a very high speed. We may call this algorithm "linewise vectorization."

In this algorithm, all the random numbers are stored in the memory IR(*). The store operation limits the speed of this program. Since it is not always necessary to keep all the values of IR(*) in the memory, the program is improved by minimizing the store operation. Aoyama et al.[8] proposed an alternative algorithm, where every four, say, random numbers are generated by the first order iteration and then the rest are generated parallelly.

```
*VOPTION NOFVAL
     do 10 i=1,n
     IR(i)=cc*IR(i-1)
       IR1=c*IR(i)
       IR2=c2*IR(i)
       IR3=c3*IR(i)
     .......
  10 continue
```

where $cc=c^4$, c2=c*c and c3=c*c*c. In this loop four random number sequences IR(i), IR1, IR2, IR3 are generated parallelly. Since the variables IR1, IR2 and IR3 are kept only on vector registors (high speed buffer memory), one can save the time for "store". We will call it "columnwise vectorization." This idea can be extended to more than four columns. The limit is the number of the vector registors and the vector pipes.

Actually this technique is also applicable to the linewise vectorization by generating more than one lines at one time. For instance,

```
     do 10 i=1,n
  10 IR(i)=c4*IR(i-1)           <--- initialization

     ..........

*VOPTION NOFVAL
     do 20 i=1,n
     IR(i)=cc*IR(i)
       IR1=c*IR(i)
       IR2=c2*IR(i)
       IR3=c3*IR(i)
     .......
  20 continue
```

where c4=c**4 and cc=c**(4*n+1).

The next problem would be a vectorizable M-sequence method. In 1985, the bit operations (AND, OR, EOR, ...) will be supported on S810. It is time to elaborate on a new algorithm. I hope some of you are interested.

4. Conversion of Random Numbers on a Pipeline Computer

In most cases, uniform random numbers are converted to other distributions. In the Monte Carlo simulations for statistical or quantum systems, random numbers obeying the Boltzmann distribution exp(-E(x,y,...)/kT) is necessary, where x, y, etc. are variables corresponding to each degree of freedom, k is the Boltzmann constant and T is the temperature.

If the density distribution function is analytically integrable and its inverse is expressed analytically, then the random numbers are generated by a direct method. For example, the distribution of the O(3) Heisenberg spin model is expressed as

$$\exp(-a \cos x) \sin x \, dx \, dy$$

where x and y are polar and azymuthal angles of a spin and a is a constant. Putting z=cos x, we can write

$$\exp(-a z) \, dz \, dy,$$

which is simply integrable.

On the other hand, the distribution of the O(4) spin is (apart from O(3) factors)

$$\exp(-a \cos x) \sin^2 x \, dx = \exp(-a z) \, (1-z^2)^{1/2} \, dz.$$

On an ordinary computers, we first generate an exponential random number for $-1<z<1$ obeying $\exp(-a\ z)$, then we use the rejection method for the factor $(1-z^2)^{1/2}$. Unfortunately, the rejection method cannot be vectorized, since we do not know beforehand how many tries are necessary. Possible method would be

1) make a table and interpolate (we have also to interpolate in a).
2) make an approximate polynomial for various a and interpolate.
3) in the case of the simulation in statistical physics, there is a freedom to keep some of the variables unchanged. If a z value generated according to $\exp(-a\ z)$ misses in the rejection method, we may leave the corresponding variable unupdated.

This O(4) distribution is quite important in the lattice gauge theory, since it is related to the adjoint representation of the SU(2) gauge group, and SU(3) simulation is performed by applying SU(2) three (or six) times.

5. Discussion

I have discussed various problems related to the random number generation in large-scale Monte Carlo simulations on supercomputers. They are not only crucial to realistic simulations but also challenging to applied mathematicians.

REFERENCES

1. K. G. Wilson, Phys. Rev. D10, (1974) 2445.
2. M. Fukugita and Y. Oyanagi, Phys. Letters 123B (1983) 71-76.
3. T. Hoshino et al., ACM Trans. on Computer Systems 1 (1983) 195-221. T. Hoshino et al, Proc. Intl. conf. on Parallel Processing, Michigan (1983) 95-103.
4. T. Hoshino et al., to be published in Comp. Phys. Comm.
5. M. Fushimi, 情報処理学会論文誌 24 (1983) 576-579.
6. M. Sugihara, private comuunication.
7. See for example, I. Borosh and H. Niederreiter, BIT 23 (1983) 65-74.
8. T. Aoyama et al., 情報処理学会第28回全国大会 (1984) p 1319。