JOURNAL OF COMPUTATIONAL PHYSICS 41, 203–210 (1981)

Note

Multi-spin Coding: A Very Efficient Technique for Monte Carlo Simulations of Spin Systems

1. Introduction

Computer simulations have often constituted an invaluable tool for the understanding of thermodynamical properties of many body systems. These simulations are generally done by the Monte Carlo method and the size of the system which can be considered is limited by the availability of memory space and computer time. The purpose of this note is to illustrate a technique which allows one to sensibly reduce the memory space and central processor (CP) time required by a Monte Carlo simulation whenever the states of the system are specified by a collection of spin variables ranging over a limited number of different values. This technique, which we shall refer to as multi-spin coding (MSC), permits one to perform simulations which would be otherwise unfeasible or to substantially lower the burden that the same simulation would place on the computer, if done by conventional methods. MSC has been tested in a variety of investigations performed at the Brookhaven National Laboratory [1–5]. As we have learned recently, moreover, a technique analogous to ours in the general idea, although quite different in the details, was successfully used a few years ago by Friedberg and Cameron [6].

The basic idea of MSC is that only few binary digits are necessary to record the value of a spin variable, which ranges over a finite and rather small set. The memory words of a computer, on the other hand, being designed to represent real numbers with a high degree of accuracy, contain many bits. It is therefore possible to record the values of several spins in the same memory word (MSC). If this packing of information is done suitably, beyond reducing the memory requirements, it also allows arithmetic operations involving many spins simultaneously to be performed and thus lowers the CP time needed for the simulation.

In this note we shall not present any specific program, but will rather illustrate the most salient features of the algorithm. We consider the particularly simple case of the Ising model in Section 2, generalize the method to other spin systems in Section 3 and present a few concluding remarks in Section 4.

2. Multi-spin Coding for the Ising Model

The Ising model is defined by a collection of spins which can take only two values, denoted by 0 and 1. We shall assume that the spins occupy the sites of a two-

0021-9991/81/050203–08$02.00/0

dimensional square lattice and will label them $s_{i,j}$ (i being the row, j the column index). The generalization to other lattices or other dimensionalities is straightforward. The energy of the system is the sum of the energies associated with the bonds between nearest neighbor pairs. The energy of a bond, in suitable units, is defined to be 0 if the neighboring spins have the same value, 1 otherwise.

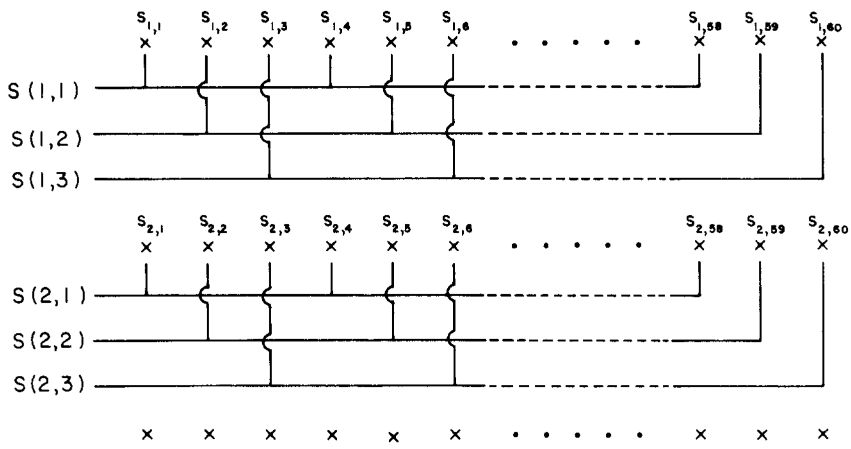
Following a method introduced by Metropolis *et al.* [7] to deal with a different problem, a dynamical simulation of the thermodynamical behavior of the system at temperature T is done as follows. One considers a definite spin s and evaluates the energy E of the four bonds involving s. Then the energy E' obtained with a different value s' of the spin is evaluated (in this model, of course, s' can only be the complement of s). If E' is lower than E, then the spin is set to the new value s'. If $E' > E$, a (pseudo)random number R with uniform distribution between 0 and 1 is generated and the value of the spin is changed only if $R < \exp(-(E' - E)/kT)$ (k is the Boltzmann constant). After s, a new spin of the lattice is thus analyzed and so on, until all spins are probed. This completes a Monte Carlo iteration. Many iterations are performed in succession and it is possible to show that the probability of encountering any definite spin configuration in the system eventually becomes proportional to the Boltzmann factor $\exp(-E_{\text{tot}}/kT)$, where E_{tot} is the total energy of the configuration. Clearly, the computation involves two steps: the evaluation of the energy associated with s, first, and then the stochastic change of the spin. It is the first step which is done in a particularly efficient way by the *MSC* technique.

To be specific, we shall assume that the memory words of the computer consist of 60 bits (as in the case of the CDC computers, on which the algorithm has been implemented). But this is only for sake of exemplification: most of the details of the method may be trivially modified to adapt it to different situations. We will then use groups of three bits within the memory word to record the values of different spins, as follows. The first bit in the group takes value s, the other two are zero (bits are enumerated from the right). Twenty spins are coded in a single memory word. We shall assume that the horizontal size of the lattice is $20N$, with N an integer greater than one. The vertical size is arbitrary. The spins $s_{i,j}$ are then organized in memory words $S(I, J)$, $1 \leqslant I \leqslant N$, $J = j$, in the following way (see Fig. 1):

$$\begin{aligned} S(1, J)&: \quad s_{1,j} s_{N+1,j} s_{2N+1,j} \cdots s_{19N+1,j}, \\ S(2, J)&: \quad s_{2,j} s_{N+2,j} s_{2N+2,j} \cdots s_{19N+2,j}, \\ S(N, J)&: \quad s_{N,j} s_{2N,j} s_{3N,j} \cdots s_{20N,j}. \end{aligned} \tag{1}$$

$N > 1$ is needed to preserve the statistical independence of the readjustments of the spins. If we consider the spins in $S(I, J)$, it is obvious that their neighbors are contained in $S(I - 1, J)$, $S(J + 1, J)$, $S(I, J - 1)$, $S(I, J + 1)$, with neighboring spins occupying the same position inside the memory word (at least for $I \neq 1$ and $I \neq N$; for these values of I, see below). On the other hand the "exclusive or" logic operation,

$$S = \text{XOR}(S1, S2) = S1.\text{AND}.(\text{NOT}.S2).\text{OR}.S2.\text{AND}.(\text{NOT}.\text{S}1), \tag{2}$$



$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,4}$ $s_{1,5}$ $s_{1,6}$ $s_{1,58}$ $s_{1,59}$ $s_{1,60}$
S(1,1)
S(1,2)
S(1,3)
$s_{2,1}$ $s_{2,2}$ $s_{2,3}$ $s_{2,4}$ $s_{2,5}$ $s_{2,6}$ $s_{2,58}$ $s_{2,59}$ $s_{2,60}$
S(2,1)
S(2,2)
S(2,3)

FIG. 1. Assignment of spins to memory levels in MSC.

produces a zero whenever the corresponding bits in $S1$ and $S2$ are equal, a one otherwise. Thus the instruction

$$\begin{aligned} E = {} & \mathrm{XOR}(S(I,J), S(I-1,J)) + \mathrm{XOR}(S(I,J), S(I+1,J)) \\ & + \mathrm{XOR}(S(I,J), S(I,J-1)) + \mathrm{XOR}(S(I,(J), S(I,J+1)) \end{aligned} \tag{3}$$

will generate a memory word, which, in each of the 20 groups of three bits of E, will contain the energy of the four bonds emanating from the corresponding spin in $S(I,J)$ (we see now why it has been convenient to allow for some redundancy in the packing of spins: the possible energy values range from 0 to 4 and require three bits for their coding). Notice that the bond energies of all 20 spins are evaluated simultaneously!

The values $I=1$ or $I=N$ require just a little more attention. If one assumes, for instance, periodic boundary conditions, the left neighbors of the spins in $S(1,J)$ are in $S(N,J)$, but shifted in position:

$$\begin{aligned} S(1,J)&: \quad s_{1,j}s_{N+1,j}s_{2N+1,j} \cdots s_{19N+1,j}, \\ S(N,J)&: \quad s_{N,j}s_{2N,j}s_{3N,j} \cdots s_{20N,j}. \end{aligned} \tag{4}$$

Thus a shift of the content of the word $S(N,J)$ is necessary before it is used in the step producing E. But this is straightforward to include in the program.

Eventually, the decision on whether to keep the current value of the spin or change it requires floating point arithmetic and must be done for each spin separately. The individual values of the bond energies, however, are extracted very efficiently in a loop.

```
 1     DO 10 K = 1, 20
 2     IE = E.AND.7                (5)
       ...
10     E = SHIFT(E, 3).
```

(The SHIFT(A, N) instruction, implemented in the CDC FORTRAN compiler, generates a cyclic permutation of the bits in the word A by N spaces to the left. N must be positive and a cyclic permutation by M spaces to the right is obtained setting $N = 60$ (number of bits in a word) $- M$).

The logic product in instruction 2 between E and 7, an integer which in binary representation consists of 57 zeroes followed by 3 ones, acts as a projection, which extracts the last three bits of E. These provide the value of the energy of the individual bond, At the end of the loop the bits in E are shifted by three places and a new spin is analyzed. If IE, which ranges from 0 to 4, is increased by one, it may be used as the index for an array of Boltzmann factors. The increase can be done most efficiently before the loop with an instruction

$$E = E + 11111111111111111111B. \tag{6}$$

The symbol B after a number indicates an octal constant. The octal constant added to E has a one in each of the 20 three-bit groups and increases all the bond energies by one unit. Once again, we see that an arithmetic operation may be done in parallel for all the 20 spins at the same time.

3. Other Spin Systems

The expression for the energy of the interacting spins may take a variety of forms, according to the model which is being considered. In many cases it is possible to devise an algorithm, which will generate the required values of the energy with logic instructions and will thus allow the simultaneous processing of all the spins stored in the same memory word. No general rule for this algorithm can be given and some ingenuity may be occasionally required. We exemplify in this section a few cases where the MSC technique may be profitably used.

In the simplest generalization of the Ising model, known as the q-state Potts model, the spin can take q integer values between 0 and $q - 1$. The interaction energy between neighboring spins s and s' is still 0 if $s = s'$, 1 if $s \neq s'$. With three bits per spin, models with q up to 8 can be considered and the interaction energy between the spins in the words $S1$ and $S2$ is simply given by the instructions

$$\begin{array}{ll} 1 & S = \mathrm{XOR}(S1, S2) \\ 2 & E = \mathrm{OR}(S, \mathrm{SHIFT}(S, 59), \mathrm{SHIFT}(S, 58)).\mathrm{AND}.11111111111111111111B. \end{array} \tag{7}$$

The "exclusive or" in instruction 1 leaves three zeroes in each of the three-bit groups in S if and only if the corresponding spins in $S1$ and $S2$ are equal. In the second instruction the logical sum of the first, second, and third bit in each group is performed (to move the second (third) bit one (two) places to the right, the entire content of the 60-bit word is rotated cyclically 59 (58) positions to the left; the logical product with the octal constant projects out the first bit of each group). The result will be zero if the two original spins are equal, one otherwise.

In many instances, the spin variables should be considered as group elements and the interaction energy is obtained first combining two or more spins according to the group multiplication and then evaluating a class function of the result. This occurs in particular in lattice gauge models, where spins are defined on the links between neighboring sites, rather than on the sites themselves, and the four spins associated with the links forming an elementary square are combined to define the interaction [8].

In some cases it is straightforward to reproduce the group operation. As an example, consider the cyclic group of four elements, Z_4. The values of the spins, between 0 and 3, are coded in the first two bits of the three-bit groups (recall that we enumerate bits from the right). Then instructions

$$\begin{array}{ll} 1 & S = S1 + S2 \\ 2 & S = S.\mathrm{AND}.33333333333333333333B \end{array} \qquad (8)$$

then combine the spins in $S1$ and $S2$ according to the group multiplication (i.e., addition mod 4) and place the result in S. In instruction 1 the spins in $S1$ and $S2$ are added; the redundancy provided by the zeroes in the third bits avoids carry-overs between different spins in the same word. In the second instruction the logical product with the octal constant, which consists of the binary digits 011 repeated 20 times, deletes the possible carry-over bits in the sums and thus converts ordinary addition into addition modulus 4.

Additions modulus higher powers of 2 may be performed also, but more bits must be allocated for the coding of each spin and fewer spins may be stored in the same memory word. As an example of a different group multiplication, the reader may easily satisfy himself that the sequence

$$\begin{array}{ll} 1 & S = S1 + S2 + 11111111111111111111B \\ 2 & S3 = S.\mathrm{AND}.44444444444444444444B \\ 3 & S = (S.\mathrm{AND}.33333333333333333333B) \\ & \quad + \mathrm{SHIFT}(S3, 58)\text{-}11111111111111111111B \end{array} \qquad (9)$$

generates addition modulus 3. (The SHIFT instruction in the third line moves all bits in S3 two places to the right.)

Eventually, as the complexity of the combination rule of the spins increases, the use of a logical algorithm may become impractical and recourse to a multiplication table may be more convenient. The advantage of MSC then reduces essentially to the saving in memory space. However, the combination of the many spins in the same memory word can be still done quite efficiently, as illustrated by the following example.

We assume now that only 10 spins, denoted by numbers ranging from 1 to 7, are coded in $S1$ and $S2$. The bits in position $4 \div 6$, $10 \div 12$,..., $58 \div 60$ are zero. Moreover, we assume that the result of the group multiplication of two spins s and s'

is recorded in the array $MT(I)$, with $I = 8s + s'$. Notice that multiplication of a binary number by 8 is equivalent to shifting its digits three positions to the left. Then the instructions

```
1    I = SHIFT(S1, 3) + S2
2    S = 0
3    DO 6 K = 1, 10
4    S = S + MT(I.AND.77B)                    (10)
5    I = SHIFT(I, 6)
6    S = SHIFT(S, 6)
```

perform the group multiplications of the spins in $S1$ and $S2$ and place the results in S. In instruction 1 the values of the spins in $S1$ and $S2$ are combined so as to produce in I 10 indices, which will become entries of the multiplication table. The six bits of the individual indices are projected out by the logical product with octal 77 in instruction 4 and the result of the group operation is added to S. Finally the bits in I and S are rotated cyclically to proceed to two new spins. The loop, being very short, generally will fit into the stack of current machine instructions and will be performed very efficiently by most computers.

4. Conclusions

The MSC technique has been used at the Brookhaven National Laboratory for a variety of Monte Carlo simulations. In several of these the quantum mechanical properties of lattice gauge models relevant to particle physics have been studied. The quantum mechanical behavior of the system is reproduced by a sum over all its space–time configurations (continuous time being replaced by a finite number of points) weighted by a measure factor $\exp(-(1/\hbar)S)$, where $\hbar$ is Planck's constant and S is the total action of the configuration, a positive definite quantity after a Wick's rotation to imaginary time. Thus one effectively simulates the behavior of a four-dimensional statistical system and, for any acceptable size, the number of spin variables is extremely large. It is in studies of this kind that we have found the MSC technique most helpful. Figure 2 illustrates one of our results [2]. It displays the behavior of a Z_8 gauge model (the spins, ranging in value between 0 and 7, are added modulus 8) under slow variations of the coupling constant of the system. The parameter β, which is proportional to the inverse squared coupling constant and which plays the same role as $1/kT$ in the analogy with a thermodynamical system, was varied in steps of 10^{-3} after each Monte Carlo iteration and the value of the action, E, was measured. (These values are plotted on the graph every 50 iterations.) As β varies from 3.5 down to 0 and then up again to 3.5, the system undergoes a thermal cycle and the appearance of two hysteresis loops (at $\beta \sim 1$ and $\beta \sim 2.8$)

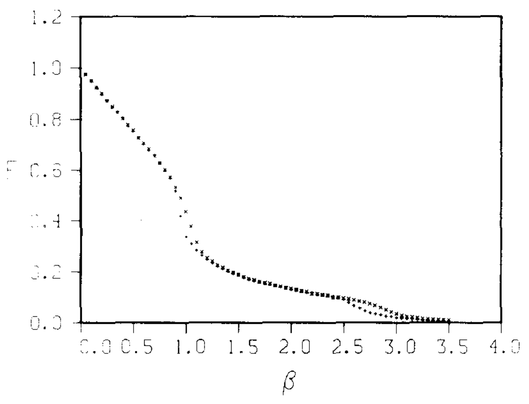


FIG. 2. Behavior of the internal energy (or average action), versus inverse temperature (or inverse squared coupling constant) in the simulation of a four-dimensional lattice gauge model, with gauge group Z_8.

signals the existence of two phase transitions. (That these loops do not correspond to spurious relaxation effects is confirmed through further analysis.) The system extended for 8 sites in each of the three spatial directions, 20 sites in the temporal direction, with periodic boundary conditions, and therefore contained 40,960 spins, the spins being defined on the links of the lattice. Yet the computer time required for a Monte Carlo iteration with its 40,960 individual upgrading steps, each involving the evaluation of the action of six different elementary squares of the lattice, was only 259 msec on a CDC 7600. The gain in time through the use of MSC in the case of Z_N gauge theories can be seen in the following table, where the average times needed to upgrade a single spin variable with a standard program and one using MSC are compared. The programs were run on a CDC 7600 with a clock period of 27.5 nsec.

The basic idea of multi-spin coding is straightforward; nevertheless it is not apparently widely used. We have presented it in some detail here because we feel that it may constitute an invaluable computational tool whenever the availability of memory space and central processor time effectively limit the Monte Carlo analysis of a statistical spin system.

Average Time per Spin (μsec)

N	Standard	MSC
2	26.01	3.05
3	25.51	4.69
4	25.71	3.78
5	25.45	5.85
6	26.06	5.71
7	25.45	****
8	25.39	6.21
9	25.33	****

Acknowledgment

This research was performed under contract DE-AC02-76CH00016 with the U.S. Department of Energy.

References

1. M. Creutz, L. Jacobs, and C. Rebbi, *Phys. Rev.* **42** (1979), 1390.
2. M. Creutz, L. Jacobs, and C. Rebbi, *Phys. Rev. D* **20** (1979), 1915.
3. R. Swendsen and C. Rebbi, *Phys. Rev. B* **21** (1980), 4094.
4. C. Rebbi, *Phys. Rev. D* **21** (1980), 3350.
5. C. Rebbi, Lecture presented at the Winter Institute on Common Trends in Particle and Condensed Matter Physics, Les Houches, 1980, *Phys. Rep.*, in press.
6. R. Friedberg and J. E. Cameron, *J. Chem. Phys.* **52** (1970), 6049. Although we differ in the implementation of MSC and extend it to other systems, the basic feature of storing many spins in the same memory word and combining them by logic operations is present in this work. We thank R. H. Swendsen for bringing this reference to our attention.
7. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *J. Chem. Phys.* **21** (1953), 1087.
8. K. G. Wilson, *Phys. Rev. D* **10** (1974), 2445. F. J. Wegner, *J. Math. Phys.* **12** (1971), 2259. R. Ballan, J. M. Drouffe, and C. Itzykson, *Phys. Rev. D* **10** (1974), 3376; **11** (1975), 2098; **11** (1975), 2104.

Received: September 24, 1980

Laurence Jacobs
Institute of Physics
University of Mexico
Mexico City 20, Mexico

Claudio Rebbi
Physics Department
Brookhaven National Laboratory
Upton, New York 11973