Journal of Statistical Physics, Vol. 56, Nos. 5/6, 1989

A Fast Vectorized Multispin Coding Algorithm for 3D Monte Carlo Simulations Using Kawasaki Spin-Exchange Dynamics

M. Q. Zhang[1,2]

Received January 19, 1989; revision received April 11, 1989

A new Monte Carlo algorithm for 3D Kawasaki spin-exchange simulations and its implementation on a CDC CYBER 205 is presented. This approach is applicable to lattices with sizes between $4\times4\times4$ and $256\times L2\times L3$ $((L2+2)(L3+4)/4\leqslant65535)$ and periodic boundary conditions. It is adjustable to various kinetic models in which the total magnetization is conserved. Maximum speed on 10 million steps per second can be reached for 3-D Ising model with Metropolis rate.

KEY WORDS: Vectorized multispin coding on Cyber 205; 3D Monte Carlo algorithm; Kawasaki dynamics; conservation law.

1. INTRODUCTION

Since supercomputers, like the CDC CYBER 205, are employed for Monte Carlo simulations, physicists are looking for algorithms which fit best to the architecture of those computers. For the ordinary Ising model, the first step was taken in 1984 when a highly optimized multispin coding algorithm for a scalar CDC computer was transferred to a CDC CYBER 205.[1] Although the speedup was a considerable factor of 15, the resulting vectorized program revealed some problems. The access to the lookup table for the transition probabilities was a bottleneck of the program, because it required a GATHER-command, which is more than twice as slow compared to a usual vectorized operation. In addition, the implementation of periodic boundary conditions in combination with vectorization

[1] Department of Physics, Rutgers University, New Brunswick, New Jersey 08903.
[2] Current Address: Courant Institute of Mathematical Sciences, New York, New York 10012.

0022-4715/89/0900-0939$06.00/0

appeared to be nontrivial; therefore, the helical boundary conditions were used. Later, two methods were published which avoid the use of a lookup table.[2,3] The first is restricted to transition probabilities which can be expressed as a power of a basic Boltzmann weight (the Metropolis is an example); the second is strictly restricted to the Metropolis method. The problem of an elegant implementation of periodic boundary conditions was still not solved. Recently there was a major breakthrough: Wansleben[4] invented a new algorithm which is adjustable to various transition probabilities, but does not use a lookup table. His implementation on the CYBER 205 realized periodic boundary conditions in all lattice directions in a very ingenious way, which made him able to advance the speed up to 27 nsec/update (38 million updates/sec) with the Metropolis rate.

The question then became whether it would be possible to extend the ideas to Kawasaki spin-exchange dynamics. Wansleben tried to make some simple modifications in order to incorporate the conservation law, but failed. Sullivan[5] reported the first efficient vectorizable algorithm for 2D spin-exchange models in which they decomposed a 2D lattice into 16 sublattices. In this paper, I generalize Wansleben's idea and decompose a 3D lattice into 16 sublattices (16 is the minimum number of independent sublattices. Using Sullivan's method, one would have to have 64 sublattices). The maximum speed, 10.2 million updates/sec, can be reached by equilibrium simulations with Metropolis rate. This algorithm was applied to a nonequilibrium stochastic lattice gas model.[6,7] The difficulties in applying this algorithm to such nonequilibrium systems will be discussed.

2. THE ALGORITHM

The efficiency of the program is due to the application of the multispin coding technique in combination with vectorization. Multispin coding is a traditional method of high-speed Monte Carlo simulation of Ising models. The idea is that the basic storage unit for one Ising variable is one bit of a machine word as opposed to one whole word which would be required in a straightforward implementation of a Monte Carlo algorithm in a high-level language. By applying word Boolean function like "exclusive-or" or "and," all spins stored in one word can be treated simultaneously. The reader can find a detailed introduction of this technique in refs. 1, 8, and 9. The implementation of a multispin coding algorithm for the Ising model on a vector computer is explained in refs. 1 and 4.

In our case, the particle jump is carried out by a logical "exclusive-or" operation between the pair of multispin-storage words controlled by an exchange word. A 1-bit in the exchange word causes a spin exchange between the corresponding pair of storage words, a 0-bit keeps the corre-

sponding spins in the old status. Finding a fast multispin algorithm means finding an elegant and efficient way to set the 1- or 0-bits in the exchange word according to the transition probability of the given stochastic dynamics.

For equilibrium spin-exchange dynamics, it fills the exchange word as follows; in the beginning all the bits are set to 1. Then 1-bits are switched into 0-bits in a six-step procedure:

1. Select a random nearest neighbor pair.
2. Calculate the number of parallel nearest-neighbor bonds n to the pair.
3. Generate a random number between 0 and 1.
4. Compare the random number with the transition probability, which is a function of n.
5. Switch from 1 to 0 if $n \geqslant 5$.
6. Switch from 1 to 0 if the random number is greater than the transition probability.

There are many possibilities to choose among for the transition probability. Here we take the Metropolis $P(n) = \min\{1, \exp[-4\beta J(n-5)]\}$.[10]

The implementation of this simple algorithm becomes complicated when as many steps as possible are carried out simultaneously for all spin variables within one word. Therefore, the program is explained by an example which shows an equivalent program for a simple integer variable IS which stores the value of one bit only. The modification to a vectorized code is straightforward. Some of the technique can be found also in ref. 4.

To illustrate, I describe each step in the syntax of scalar CYBER-FORTRAN.

Step 1 consists of randomly choosing one site with spin variable IREF and its nearest neighbor IP in one of the three principle positive directions (we call the plus direction). Step 2 uses a binary representation to calculate parallel bonds (the nearest neighbors of IREF in the minus, left, right, up, and down directions are IM, IL, IR, IU, and ID, respectively; the nearest neighbors of IP in the plus, left, right, up, and down directions are IPP, IPL, IPR, IPU, and IPD, respectively). We have

```
C.....COMPARING WITH "IREF" NEIGHBORS
      IE1 = XORN(IM,IREF)
      IE = XORN(IR,IREF)
      IE2 = AND(IE,IE1)
      IE1 = XOR(IE,IE1)
      IE = XORN(IL,IREF)
```

```
      ISCR = AND(IE,IE1)
      IE2 = XOR(ISCR,IE2)
      IE1 = XOR(IE,IE1)
      IE = XORN(IU,IREF)
      ISCR = AND(IE,IE1)
      IE3 = AND(IE,IE1)
      IE2 = XOR(ISCR,IE2)
      IE1 = XOR(IE,IE1)
      IE = XORN(ID,IE1)
      ISCR = AND(IE,IE1)
      ISCR1 = AND(ISCR,IE2)
      IE3 = XOR(ISCR1,IE3)
      IE2 = XOR(ISCR,IE2)
      IE1 = XOR(IE,IE1)
C.....COMPARING WITH "IP" NEIGHBORS
      IE = XOR(IPP,IREF)
      ISCR = AND(IE,IE1)
      ISCR1 = AND(ISCR,IE2)
      IE3 = XOR(ISCR1,IE3)
      IE2 = XOR(ISCR,IE2)
      IE1 = XOR(IE,IE1)
      IE = XOR(IPR,IREF)
      ISCR = AND(IE,IE1)
      ISCR1 = AND(ISCR,IE2)
      IE3 = XOR(ISCR1,IE3)
      IE2 = XOR(ISCR,IE2)
      IE1 = XOR(IE,IE1)
      IE = XOR(IPL,IREF)
      ISCR = AND(IE,IE1)
      ISCR1 = AND(ISCR,IE2)
      IE4 = AND(ISCR1,IE3)
      IE3 = XOR(ISCR1,IE3)
      IE2 = XOR(ISCR,IE2)
      IE1 = XOR(IE,IE1)
      IE = XOR(IPU,IREF)
      ISCR = AND(IE,IE1)
      ISCR1 = AND(ISCR,IE2)
      ISCR2 = AND(ISCR1,IE3)
      IE4 = XOR(ISCR2,IE4)
      IE3 = XOR(ISCR1,IE3)
      IE2 = XOR(ISCR,IE2)
      IE1 = XOR(IE,IE1)
```

```
IE = XOR(IPD,IREF)
ISCR = AND(IE,IE1)
ISCR1 = AND(ISCR,IE2)
ISCR2 = AND(ISCR1,IE3)
IE4 = XOR(ISCR2,IE4)
IE3 = XOR(ISCR1,IE3)
IE2 = XOR(ISCR,IE2)
IE1 = XOR(IE,IE1)
```

where the Boolean functions may be thought of as the binary operations (mod 2):

AND(IX,IY) = IX*IY XORN(IX,IY) = 1 − XOR(IX,IY)
XOR(IX,IY) = IX + IY OR(IX,IY) = XOR(IX,IY) + AND(IX,IY)

Step 3 employs a very fast random number generator, which will be explained later.

Step 4 can be illustrated by the following example with BOLT$n = P(n)$ and RAND being the random number:

```
C.....COMPARING "RAND" WITH "BOLTN"
      IF(RAND .GT. BOLT5) THEN
      IFL6 = 1
      ELSE
      IFL6 = 0
      :
      IF(RAND .GT. BOLT1) THEN
      IFL1 = 1
      ELSE
      IFL1 = 0
      END IF
C.....SET THE EXCHANGE-WORD "ISCR1"
      ISCR = AND(IE4,IE2)
      ISCR = AND(ISCR,IFL5)
      ISCR1 = ANDN(1,ISCR)
      ISCR = AND(IE4,IE1)
      ISCR = AND(ISCR,IFL4)
      ISCR1 = ANDN(ISCR1,ISCR)
      ISCR = NOR(IE1,IE2)
      ISCR = AND(ISCR,IE4)
      ISCR = AND(ISCR,IFL3)
      ISCR1 = ANDN(ISCR1,ISCR)
      ISCR = AND(IE3,IE2)
      ISCR = AND(ISCR,IE1)
```

```
      ISCR = AND(ISCR,IFL2)
      ISCR1 = ANDN(ISCR1,ISCR)
      ISCR = AND(IE3,IE2)
      ISCR = AND(ISCR,IFL1)
      ISCR1 = ANDN(ISCR1,ISCR)
C.....CHECK EXCLUSION
      ISCR1 = XOR(IREF,IP)
C.....PERFORM SPIN-EXCHANGE
      IREF = XOR(IREF,ISCR1)
      IP = XOR(IP,ISCR1)
```

where ANDN(IX,IY) = IX*(1-IY) mod 2.

The important features of the examples with respect to vectorization as well as multispin coding are:

1. Besides the if-blocks (and the random number generator), all the statements can be carried out simultaneously for all 64 bits of a machine word.
2. Each if-block becomes the single vector statement CALL Q8CMPGE on the CDC CYBER 205 with an asymptotic performance speed of 10 nsec/result on the two pipes.
3. The algorithm takes full advantage of the fact that $P(n) = 1$ for $n \leqslant 5$ in the Metropolis rate, which is not possible for other rates.
4. The words IFLn are filled with 1- and 0-bits regardless of the actual configuration. On a multiprocessor machine, this step could be performed simultaneously with the calculation of the nearest neighbor configuration on an independently working processor.

3. IMPLEMENTATION OF THE SHIFT-REGISTER RANDOM NUMBER GENERATOR AND RANDOM INITIAL CONFIGURATIONS

As discovered by Kalle and Wansleben,[11] the random number generator RANF implemented by CDC on the CYBER 205 can yield wrong results when large systems are being simulated on the vector computer. They presented a vectorized version of TRNG (shift-register sequence random number generator introduced by Tausworthe[12] and programmed in ASSEMBLER on an IBM370/168 by Kirkpartrick and Stoll[13]), which has better statistical properties and has a speed of about 100 million pseudorandom numbers/sec. The basic idea is to realize the recursion

$$N_{k+250} = N_k \oplus N_{k+147} \tag{1}$$

where N_k are integers and $\oplus$ denotes the word Boolean operation "exclusive-or." This random number generator has a period of 2^{250}, i.e., 2.6×10^{61} larger than that of RANF. This large period is the main advantage of TRNG; the drawback in using it on a scaler machine is that 250 seed integers have to be kept in the memory. Therefore, it is very difficult to implement algorithm (1) on a scaler computer without loss of time with accesses to the memory.[13] Since the CYBER 205 is specialized in performing memory-to-memory operations with vector instructions, algorithm (1) can be considerably speeded up by vectorization. We quote only the relavent codes and refor for more details to ref. 11:

```
ASSIGN IA,IRAND(1;LENGTH)
ASSIGN IB,IRAND(148;LENGTH)
ASSIGN IRANDD,IRAND(251;LENGTH)
ASSIGN IC,IRAND(LENGTH + 1;250)
ASSIGN ISEED,IRAND(1;250)
CALL Q8XORV(0,,IA,,IB,,IRANDD)
CALL Q8VTOV(0,,IC,,0,,ISEED)
```

By these statements, a vector of pseudorandom numbers and a new seed represented by the descriptor variables IRANDD and ISEED, respectively, are generated with a speed of two numbers/cycle (LENGTH > 250; the first 250 words may be constructed by using RANF).

It is easy to generate a random configuration without specifying the total magnetization.[4] For a given magnetization, we then have to add some bit manipulations.

4. SUBLATTICE STRUCTURE AND PERIODIC BOUNDARY CONDITIONS

The general idea is to try to decompose the whole lattice into the least number of sublattices, on each pair of which may be performed the given dynamics independently, and store the spin variables on each sublattice by a vector. One can easily convince oneself that the least number of sublattices for the spin-exchange process which has the property of translational invariance is 16. The structure is such that if we take any cube of size 5 (in units of lattice spacing), then all the face centers, corners of the cube, and the center of the cube belong to a same sublattice. A primitive cell is $4 \times 2 \times 2$.

In order to realize the algorithm, it is necessary to know where the occupation variables are stored within the memory. For an $L_1 \times L_2 \times L_3$ lattice, if we take the 1 direction to be the multispin coding direction, the

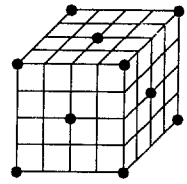

Fig. 1. Center and back sites are not shown.

number of the occupation variables per machine word is MSPC $= L_1/4$ (of course, L_1 must be a multiple of 4; the maximum speed can be reached at MSPC $= 64$). The words used for the storage of one row are indicated by the index $I_1 = 1,..., 16$. The rows are indicated by the second index I_2 using helical boundary conditions (see Fig. 1). Vectors are formed from the array IS with fixed I_1. Thus, I_1 is also the sublattice index, $I_1 = 1, 2,...,$

Table I. Nearest Neighbor Map Table for $L \times L \times L^2$ Lattice

(x, y, z)	(I1,I2),	n	
$(x+1, y, z)$	(I1 + 2,I2),	n	: I1 ≠ 7,8,15,16
	(I1 − 6,I2),	n + (64/MSPC)	: I1 = 7,8,15,16
$(x-1, y, z)$	(I1 − 2,I2),	n	: I1 ≠ 1,2,9,10
	(I1 + 6,I2),	n − (64/MSPC)	: I1 = 1,2,9,10
$(x, y+1, z)$	(I1 + 8,I2),	n	: I1 ⩽ 8
	(I1 − 4,I2 + 1)	n	: I1 = 9,10,11,12;I2 = even
	(I1 − 4,I2 + 1),	n − (64/MSPC)	: I1 = 9,10,11,12;I2 = odd
	(I1 − 12,I2 + 1),	n	: I1 = 13,14,15,16;I2 = odd
	(I1 − 12,I2 + 1),	n + (64/MSPC)	: I1 = 13,14,15,16;I2 = even
$(x, y-1, z)$	(I1 − 8,I2),	n	: I1 > 8
	(I1 + 4,I2 − 1),	n	: I1 = 5,6,7,8;I2 = odd
	(I1 + 4,I2 − 1),	n + (64/MSPC)	: I1 = 5,6,7,8;I2 = even
	(I1 + 12,I2 − 1),	n	: I1 = 1,2,3,4;I2 = even
	(I1 + 12,I2 − 1),	n − (64/MSPC)	: I1 = 1,2,3,4;I2 = odd
$(x, y, z+1)$	(I1 + 1,I2),	n	: I1 = odd
	(I1 + 3,I2 + 1 + L/2),	n	: I1 = 2,4,10,12;I2 = even
	(I1 + 3,I2 + 1 + L/2),	n − (64/MSPC)	: I1 = 2,4,10,12;I2 = odd
	(I1 − 5,I2 + 1 + L/2),	n	: I1 = 6,8,14,16;I2 = odd
	(I1 − 5,I2 + 1 + L/2),	n + (64/MSPC)	: I1 = 6,8,14,16;I2 = even
$(x, y, z-1)$	(I1 − 1,I2),	n	: I1 = even
	(I1 − 3,I2 − 1 − L/2),	n	: I1 = 5,7,13,15;I2 = odd
	(I1 − 3,I2 − 1 − L/2),	n + (64/MSPC)	: I1 = 5,7,13,15;I2 = even
	(I1 + 5,I2 − 1 − L/2),	n	: I1 = 1,3,9,11;I2 = even
	(I1 + 5,I2 − 1 − L/2),	n − (64/MSPC)	: I1 = 1,3,9,11;I2 = odd

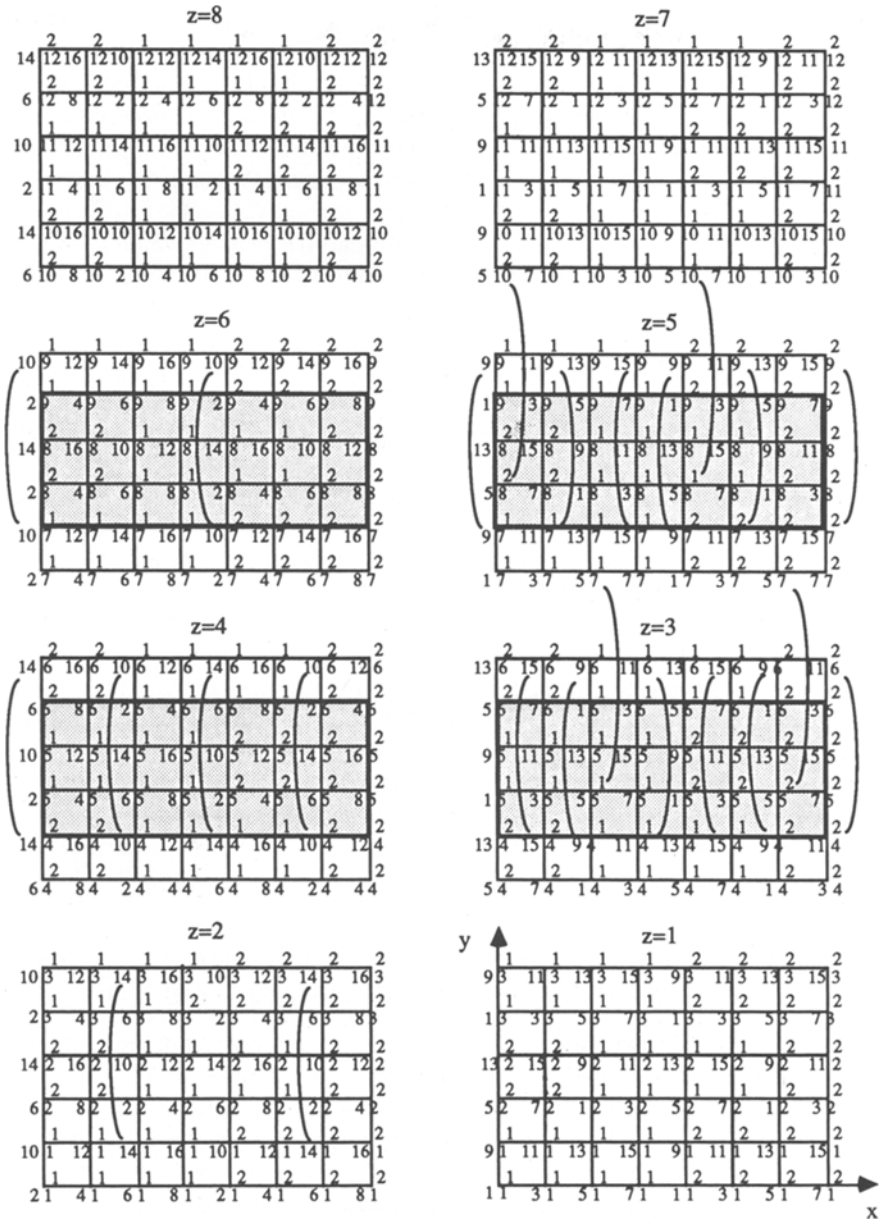

Fig. 2. Example of the vectorized multispin coding structure for an $8 \times 4 \times 4$ lattice system. We set the origin to be $x = y = z = 1$; the physical lattice is contained in $1 \leqslant x \leqslant 8$, $2 \leqslant y \leqslant 5$, $3 \leqslant z \leqslant 6$. The I_1 and I_2 are indicated at the bottom of each site and n at the upper right corner, which means the spin is positioned in $\mathrm{IS}(I_1, I_2)$ at the $[1 + (n-1)L/\mathrm{MSPC}]$th bit. To make the sublattices periodic right before performing spin exchange between sublattices 1 and 9, the physical spins have to be copied onto the "virtual" spins as indicated by curved lines.

$(L_2+2)(L_3+4)/4$ (the maximum I_1 should be less than $2^{16}=65536$). Let (x, y, z) denote a lattice point, and the corresponding variable is stored within the word $\mathrm{IS}(I_1, I_2)$ at bit position n; then the variables on the adjacent lattice points can be located in the array IS using the mapping Table I.

Before checking whether two spins are parallel or not (which is simply performed by a logical operation of two multispin-coding words), a shift operation may have to be performed in order to guarantee their bit positions are right. If this shift operation is periodic ("left end-around,"[14] i.e., bit 1 is shifted to bit position 64 or vice versa), the periodicity is automatically realized. On CYBER 205, the left shift has this convenient property. Thus, all shift operations must be carried out as left shift, a right shift by 64/MSPC bits is carried out as a left shift by (64 − 64/MSPC) bits.

Periodic boundary conditions in other directions than the multispin coding directions are realized by the particular chosen sublattice structure and the use of the "virtual" spins. Figure 2 depicts an example of a vectorized multispin coding structure for an $8\times4\times4$ lattice system. The physical lattice is contained in $1\leqslant x\leqslant 8$, $2\leqslant y\leqslant 5$, $3\leqslant z\leqslant 6$. With one extra layer in both top and bottom of the 2 direction and two extra layers in both top and bottom 3 direction, strict periodic boundary conditions can be realized with the help of these "virtual" spins (i.e., copies of spins of the physical lattice). For instance, if immediately before performing spin exchanges between the sublattices $\mathrm{IS}(1,\cdot)$ and $\mathrm{IS}(9,\cdot)$, the physical spins are copied onto the virtual ones as indicated by the curved lines in Fig. 2, then the periodic boundary conditions for those sublattices are clearly established.

5. COMPARISON WITH NONCONSERVED DYNAMICS AND THE DIFFICULTIES FOR NONEQUILIBRIUM SYSTEMS

Since the conservation law has forced us to choose more sublattices, that slows down the speed as compared with the nonconserved case. If one tries to apply this algorithm to nonequilibrium problem, more "randomization" may be needed. For example, we have applied it to a nonequilibrium stochastic lattice gas model[6,7] in which the jumps in the z direction are biased by a constant external field E. In the strong field limit, the jump probability in the z direction becomes 1 if it is in the field direction or 0 if it is against the field direction. Aside from some obvious modifications on $P(n)$, there are more difficulties arising from the lack of isotropy and detailed balancing. The anisotropic decays of the correlation functions in different directions suggest that one should take the size longer or shorter according to whether the correlation length is longer or shorter in that

direction. The lack of detailed balancing is more serious, as it raises the question of the ergodicity which goes to the heart of the principle of the Monte Carlo method. In the strong field limit mentioned above, the spin exchange in the field direction amounts to the interchange of the sublattices. Therefore the lack of randomness tends to build up correlations within each sublattice. To overcome this without abandoning the vectorized multispin coding, we had to introduce another control vector as a randomizer, in which only a fraction f of the bits were randomly set, then using this control vector to randomly select fraction f of the spins in the sublattices performing the exchanges. The test showed that, in order to keep the error down ($<5\%$), we needed $f \leqslant 1\%$! Clearly, this f factor was the bottleneck of the efficiency in application to the nonequilibrium dynamics. I believe that in any nonequilibrium simulations which do not obey the detailed balancing, one will encounter similar problems if one tries to apply vectorized multispin coding techniques.

ACKNOWLEDGMENTS

I thank S. Wansleben for introducing me to his method. Thanks also go to Prof. J. L. Lebowitz for his constant interest and support and to NSF for the generous supercomputer time grant throughout this research. Innumerable technical help from CCIS (D. Rekant in particular) at Rutgers, JVNC (L. Anacker in particular) at Princeton, and ACMC at Georgia is deeply appreciated. This work was supported by a Rutgers Supercomputer Fellowship and by NSF grant DMR-81-14726.

Upon completion of this work, I received a preprint, "A Monte Carlo study of growth in the two-dimensional spin-exchange kinetic Ising model" by J. G. Amar, F. E. Sullivan, and R. D. Mountain in which they applied a 16-sublattice algorithm to a 2D spin-exchange kinetics with detailed balancing.

REFERENCES

1. S. Wansleben, J. G. Zabolitzky, and C. Kalle, *J. Stat. Phys.* **37**:271 (1984).
2. G. O. Wiliams and M. H. Kalos, *J. Stat. Phys.* **37**:283 (1984).
3. G. Bhanot, D. Duke, and R. Salvador, *J. Stat. Phys.* **44**:985 (1986).
4. S. Wansleben, *Comput. Phys. Comm.* **43**:315 (1987).
5. F. E. Sullivan and R. D. Mountain, *Proceedings of the First Symposium on the Frontiers of Massively Parallel Scientific Computing*, September 1986.
6. M. Q. Zhang, Ph.D. Thesis, Rutgers University, New Brunswick, New Jersey.
7. M. Q. Zhang, J. S. Wang, J. L. Lebowitz, and J. L. Vallés, *J. Stat. Phys.* **52**:1461 (1988).
8. R. Zorn, H. J. Herrmann, and C. Rebbi, *Comput. Phys. Comm.* **23**:337 (1981).

9. C. Kalle and V. Winkelmann, *J. Stat. Phys.* **28**:639 (1982).
10. N. Metropolis, A. Rosenbluth, A. H. Teller, and E. Teller, *J. Chem. Phys.* **21**:21 (1953).
11. C. Kalle and S. Wansleben, *Comput. Phys. Comm.* **33**:343 (1984).
12. R. C. Tausworthe, *Math. Comput.* **19**:201 (1965).
13. S. Kirkpatrick and E. Stoll, *J. Comput. Phys.* **40**:517 (1981).
14. FORTRAN 200 VERSION 1 manual (1984), publ. number 60480200.