JOURNAL OF COMPUTATIONAL PHYSICS **44**, 227–242 (1981)

Monte Carlo Simulation of Ising Systems Using a Data Structure

M. P. HARDING

CSIRO Division of Materials Science, University of Melbourne, Parkville, Victoria 3052, Australia

Received May 27, 1981

A versatile and efficient algorithm for the Monte Carlo simulation of Ising spin systems is described. A data structure is devised which allows the same program segment to be used for spin reversal trials for lattices of various types. The simulation of two and three-dimensional lattices is equally practical in terms of programming effort and computer time. At $T/T_c = 1.0$ the number of machine instructions executed per configuration change for the *sq*, *sc*, *bcc* and *fcc* lattices are approximately 125, 70, 75 and 90, respectively.

1. INTRODUCTION

The Monte Carlo method has been used extensively to study spin one-half Ising systems with nearest neighbour interactions [1, 2, 3]. The Hamiltonian of the spin-$\frac{1}{2}$ Ising model [4, 5] in a field H is,

$$\mathscr{H}_m = -J \sum_{i,j} s_i s_j - mH \sum_i s_i, \tag{1}$$

where the spin variables s_i take the values ± 1 and the summation indices refer to sites on a d-dimensional lattice. The first summation is restricted to nearest neighbour spins and the magnetic moment is denoted by m.

The Monte Carlo method may be described as a computer experiment in which a sequence of configuration states is generated. While the system exists in one state a tentative transition to another state is selected by a random process which involves comparing a computer generated random number and a transition probability. The transition probabilities are chosen to make the distribution of states tend toward a Boltzmann distribution as the number of states of the sequence increases. The taking of ensemble averages of physically interesting quantities begins when it is judged that the influence of the starting distribution is negligible. The sequence is terminated when the desired precision is reached or the costs outweight the need for further precision. It has been shown [6, 7] that the avearages in the limit of infinitely long sequences converge to values representative of the true distribution, provided the random process is capable of leading to all possible configurations.

0021-9991/81/120227–16$02.00/0

TABLE I

Classification of Lattice Types

Lattice type	Dimensions	Number of nearest neighbours
Square	2	4
Hexagonal	2	6
Simple Cubic	3	6
Body centred cubic	3	8
Face centred cubic	3	12

Usually a Monte Carlo algorithm is written for a particular lattice type such as one of the lattice types given in Table I. The number of nearest neighbours surrounding a lattice site is important as the identity of each nearest neighbour atom must be considered when determining the transition probability. A basic computer storage technique is to allocate one word of memory for each lattice site. The word is set to one or zero depending on whether an up spin or a down spin, respectively, occupies the corresponding lattice site. Owing to the finite size of the computer's memory it is necessary to define a lattice boundary [3]. A convenient boundary is the periodic boundary where the lattice is surrunded on all sides with copies of itself.

An important factor influencing the algorithm design is the physical difference between successive configurations in the sequence. In spin reversal models the next configuration is generated by reversing a single spin, a model where the concentration of up spins (down spins) may vary. In spin exchange simulations a constant concentration of spins is maintained by exchanging unlike nearest neighbour spins.

The errors which arise in Monte Carlo simulation are well investigated and include, finite lattice size [8–12], long relaxation times [13, 14] and finite time averaging [15, 16]. Here a fast and versatile algorithm for Monte Carlo simulation of Ising models will be reported. The algorithm features a data structure which facilitates systematic program writing for a variety of applications and leads to improved execution efficiency. This research was undertaken to evaluate ensemble probabilities on the surface of certain bimetallic catalysts [17].

2. The Standard Monte Carlo Algorithm

In this section the standard Monte Carlo algorithm will be reviewed to establish terminology and identify certain sources of inefficiency. For the sake of simplicity the major part of the discussion will concern the Ising model square lattice with periodic boundary conditions and single spin reversal. The standard algorithm may be divided into three modules as set out below.

BEGIN:—

(1) Generate and store the transition probabilities
(2) Generate the sequence of configurations from the initial configuration
(3) Sample the current configuration

END:—

2.1. *Module* 1

The transition probabilities are calculated from the input parameters, such as temperature, magnetic field and coupling energy and stored in an array for fast referencing. It is convenient to derive expressions for the transition probabilities by relating group probabilities. The 12 groups of the two-dimensional square lattice are shown in Fig. 1 and are referred to as A or B depending on the spin of the central site. The probability of forming a group having a positive central spin and n positive nearest-neighbour spins is denoted as A_n and the corresponding probability of having a negative spin surrounded by n positive spins is B_n. Including the possible degeneracies of each of the spin groups there are 32 different configurations, however, symmetry arguments reduce the number of different group probabilities to 12, there being six A groups and six B groups. It is, however, possible to relate the A_n and B_n probabilities. Consider two spin systems, identical except that in one system a particular site (having n nearest neighbour positive spins) has positive spin (system A), while for the other system the same site has negative spin (system B). The ratio of the probabilities of occurrence of these two systems is given by the ratio of their respective Boltzmann factors:

$$\begin{aligned}\frac{P(\text{system } A)}{p(\text{system } B)} &= \frac{\exp[-(-J\sum_{NN} s_j - mH)/kT]}{\exp[-(-J\sum_{NN} s_j + mH)/kT]}\\ &= \exp([2mH - 8J/kT])[\exp(4J/kT)]^n\\ &= \kappa\lambda^n, \qquad (2)\end{aligned}$$

where

$$\kappa = \exp([2mH - 8J]/kT), \qquad (3)$$

$$\lambda = \exp(4J/kT), \qquad (4)$$

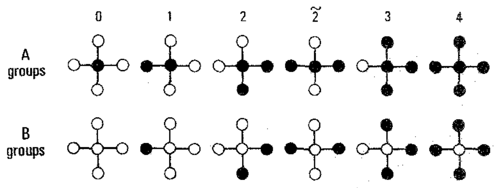


FIG. 1. The 12 groups of the Ising square lattice.

and

$\sum_{NN}$ is the sum over the four nearest neighbours.

As (2) is true for all states of the system we obtain

$$\frac{A_n}{B_n} = \kappa\lambda^n. \tag{5}$$

Therefore from (5) the transition probabilities for the square lattice are given by,

$$B_n \to A_n = \kappa\lambda^n/(1+\kappa\lambda^n), \tag{6}$$

$$A_n \to B_n = 1/(1+\kappa\lambda^n). \tag{7}$$

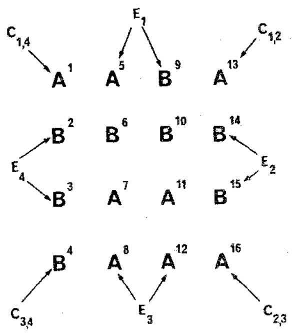
2.2. *Module* 2

As shown below module 2 consists of an inner *repeat until*, where spin reversal trials are executed and an outer *repeat until*, where, in the event of a successful trial, the spin is reversed.

```
BEGIN:—
    REPEAT
            REPEAT spin reversal trials
            UNTIL   spin $I$ is to be reversed
            reverse spin $I$
            SEQCOUNT equals SEQCOUNT + 1
      UNTIL SEQCOUNT equals SEQLENGTH
END:—
```

A spin reversal trial involves choosing spin I, $1 \leqslant I \leqslant N$ at random from the lattice of N spins. Spin I and its four nearest neighbours constitute a group for which a transition probability is read referenced. A computer generated random number in the interval (0, 1) is subtracted from the transition probability. On a negative result a new spin reversal trial begins otherwise the spin is reversed and the spin reversal count is increased by one. When the required number of spin reversals have taken place the execution of module 2 is completed.

The nearest neighbour address problem. To obtain a transition probability for spin I it is necessary to read reference memory at the four addresses corresponding to the nearest neighbours of spin I. To apppreciate the problem of determining the nearest neighbour addresses consider the square lattice of $N = 16$ sites as shown in Fig. 2. It is convenient to distinguish between the edge sites and the interior or bulk sites.



$C_{1,4}$
E_1
$C_{1,2}$
A^1 A^5 B^9 A^{13}
B^2 B^6 B^{10} B^{14}
E_4
E_2
B^3 A^7 A^{11} B^{15}
B^4 A^8 A^{12} A^{16}
$C_{3,4}$
E_3
$C_{2,3}$

FIG. 2. The boundary sites of an $N = 16$ Ising square lattice.

Let the bulk site routine reference the nearest neighbours $I-1$, $I+S$, $I+1$ and $I-S$ of bulk sites I, where I can be sites 6, 7, 10 and 11 and $S = N^{1/2}$. The nearest neighbours of lattice sites other than 6, 7, 10 and 11 are not referenced by the bulk site routine. These boundary sites may be classified into either edge (E_i) or corner (C_{ij}) sites. Using periodic boundary conditions, nine routines (see Table II) are required to reference the nearest neighbours of every lattice site.

The choice of which routine is appropriate for spin I must be made each time a spin reversal trial is executed. High level languages, such as ANSI FORTRAN, arrange the choices in a list and proceed stepwise through the list. This technique is efficient as long as the alternative is near the top of the list. For the square lattice the first item on the list can be the bulk site routine, because a site chosen at random from $N = 40{,}000$ sites has a probability of 0.98 of being a bulk site. Unfortunately

TABLE II

Nearest Neighbours of Site I

Routine	Sites	Nearest neighbours $(S = N^{1/2})$			
Bulk	6, 7, 10, 11	$I-1$	$I+S$	$I+1$	$I-S$
C_{14}	1	$I+S-1$,	$I+S$,	$I+1$,	$I+(S-1)S$
E_4	2, 3	$I-1$,	$I+S$,	$I+1$,	$I+(S-1)S$
C_{12}	4	$I-1$,	$I+S$,	$I-S+1$,	$I+(S-1)S$
E_1	5, 9	$I+S-1$,	$I+S$,	$I+1$,	$I-S$
C_{12}	13	$I+S-1$,	$I-(S-1)S$,	$I+1$,	$I-S$
E_3	8, 12	$I-1$,	$I+S$,	$I-S+1$,	$I-S$
E_2	14, 15	$I-1$,	$I-(S-1)S$,	$I-1$,	$I-S$
C_{23}	16	$I-1$,	$I-(S-1)S$,	$I-S+1$,	$I-S$

for other lattices the list of alternatives is much longer and the alternative decided upon has a greater probability of being further down the list.

Some of the neighbour referencing problems can be avoided by choosing spin I sequentially from the N spins rather than at random. The neighbour referencing routines can then be programmed to keep pace with a counter moving through the lattice. Sequential site simulations, however, have the inherent disadvantage of less degrees of freedom when compared to random site simulations, and there is evidence that sequential site simulations have inefficiencies of their own in dealing with edge sites. In a sequential site simulation [10] for the square lattice the free edge boundary condition algorithm was reported to be four times faster than the periodic boundary condition algorithm.

At present there is a need to develop efficient software for Monte Carlo simulation, particularly for the three-dimensional lattices where there are more nearest neighbours per site and proportionally more edge and corner sites than on the square lattice.

2.3. *Module* 3

The taking of ensemble averages is straight forward as one has an actual lattice of spins with which to work. Module 3 is mentioned because the data structure leads to a simple and efficient sampling technique for the group probabilities.

2.4. *High Cost Areas*

The high costs of Monte Carlo simulation arise from Module 2, that is generating the large number of configurations in the sequence. Therefore the average computing time, t, per configuration change will be used as a measure of efficiency of the simulation algorithm. t is given by,

$$t = S \cdot F + U, \tag{8}$$

where F is the average number of spin reversal trials per configuration change, S is the average execution time of a spin reversal trial and U is the average execution time of actually reversing spin I.

For the standard Monte Carlo algorithm $U < S$ and the major contribution to t is from continued execution of spin reversal trials. For the infinite square lattice with zero field, F may be evaluated by summing the product of the group probabilities [18] and the transition probabilities. At the critical temperature of the square lattice $F = 6.9128$ and at infinite temperature $F = 2.0$. In order to counteract the larger values of F encountered below the critical temperature the n-fold algorithm [19] was introduced. With this algorithm F effectively becomes unity and U and S are considerably increased. For the square lattice at $T/T_c = 0.558$ the efficiency reported is 10 times that of the standard algorithm. The present approach is to accept the value of F and endeavour to reduce the execution time of a spin reversal trial.

3. Monte Carlo Simulation Using a Data Structure

In this section an algorithm using a data structure designed for Monte Carlo simulation of Ising Models will be described. The algorithm has the same form as module 2 of the standard Monte Carlo algorithm, however, the data structure enables spin reversal trials to be executed without referencing the nearest neighbour addresses.

For a lattice of N sites, the data structure consists of N nodes of one computer word per node, where node I contains information regarding spin I (site I). Each node is divided into a "group field" and a "location field" whose contents are defined as GROUP (I) and LOCATION (I), respectively. The node format for the square lattice is given in Fig. 3. The group field, to be discussed first, provides a linkage to the transition probability for spin I, whereas the location field provides a linkage to a program module which reverses spin I.

3.1. *Group Field—Spin Reversal Trial*

The identities of the spins occupying site I and the four nearest neighbour sites of site I are stored in the group field of note I. To illustrate the group field storage convention consider a group labelled "0" chosen at random from a square lattice as shown in Fig. 3. The group made up of the central site 0 and the four nearest neighbour sites labelled 1, 2, 3, and 4 is stored in the group field as the binary number 01011. The 0, 1, 2, 3 and 4 bits of the group field are set if and up spin occupies the 0, 1, 2, 3 and 4 sites, respectively, of the group, and cleared if a down spin occupies the 0, 1, 2, 3 and 4 sites, respectively, of the group. The $N = 16$ lattice given in Fig. 3 is rewritten in group field format and shown in Fig. 4.

For a lattice stored as an array of nodes with base address L, GROUP (I) may be obtained with one reference to address $L + I - 1$. GROUP (I) can be used to directly address the transition probability for spin I if the precalculated probabilities are stored appropriately. GROUP (I) may be thought of as a binary number which may take one of 32 values, 0 octal $\leqslant$ GROUP (I) $\leqslant$ 37 octal. The transition probabilities calculated in program module 1, are stored in an array of 32 words with base address P so that $P +$ GROUP (I) is the address of the transition probability for spin I. With

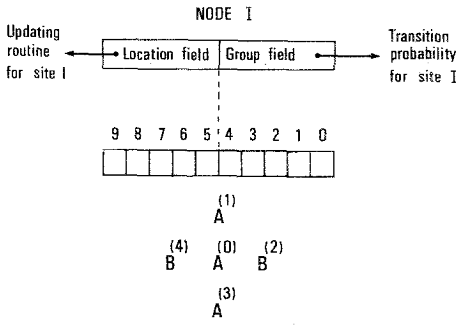


FIG. 3. Data structure node format for the Ising square lattice.

10101^{1}	10011^{5}	10110^{9}	00111^{13}
00010^{2}	01010^{6}	01000^{10}	00010^{14}
00100^{3}	01101^{7}	11001^{11}	11000^{15}
11100^{4}	01111^{8}	10111^{12}	11001^{16}

FIG. 4. The $N = 16$ Ising square lattice in group field format.

this arrangement a spin reversal trial can be executed with only two memory references as outlined below.

```
BEGIN:—
    REPEAT
            generate a random Index $I$
            read reference node $I$
            read reference the transition probability at
               address $P$ + GROUP ($I$)
            generate a random number
            FLAG = trans. prob.—random number
      UNTIL FLAG is negative
END:—
```

3.2. *Location Field—Spin Reversal*

In order to reverse spin I and keep the data structure up to date it is necessary to update the group fields of five nodes, that is node I and the four nearest neighbour nodes. This task will be performed by a program segment referred to as an updating routine. The updating routine must contain instructions to read reference the nearest neighbour nodes before the group fields can be updated. It follows from Table II that nine updating routines are sufficient. The choice of which updating routine is suitable for spin I is facilitated by storing the classification of site I, i.e., bulk, edge (E_i) or corner (C_{ij}) in the location field of node I. In particular the location field of node I contains the starting address of the updating routine for spin I. The program branch can then be completed by executing a machine instruction which allows a parameter, in this case the location field, to specify a program branch destination address. This branching technique facilitates efficient simulation of more complex lattices where there are considerably more than nine updating routines.

3.3. *The Updating Routine*

Updating a node amounts to complementing one of the group field bits. To reverse a spin on site I, where site I is a bulk site the following bits must be complemented.

zero	bit of GROUP (I)
one	bit of GROUP $(I+1)$
two	bit of GROUP $(I-S)$
three	bit of GROUP $(I-1)$
four	bit of GROUP $(I+S)$

With the information provided in Table II it is straightforward to write the list of operations required to reverse a spin on any lattice site. The operations may be executed systematically by using an exclusive *OR* (logical difference) instruction and an updating operand. For example, if the spin on site 7 of Fig. 3 is to be reversed then the group fields of nodes 7, 8, 3, 6 and 11 may be updated as shown below.

```
01101   7
00001   XOR – updating operand
-----
01100   7 updated
01111   8                       00100     3
00010   XOR                     00100    XOR
                                -----
01101   8 updated               00000   3 updated
01010   6                       11001     11
01000   XOR                     10000    XOR
                                -----
00010   6 updated               01001   11 updated
```

The updating operand is left shifted one bit position as the algorithm updates nodes 7, 8, 3, 6 and 11 in turn. The exclusive *OR* instruction is used as it is not then necessary to determine the status of the bit to be complemented.

Updating routines within any lattice type are identical except for the instructions which read reference the nearest neighbour sites. This similarity allows each updating routine to be duplicated from the previous one, which is particularly important for the face centred cubic lattice which has 32 updating routines, each with approximately 50 instructions.

The decision whether or not to halt the simulation involves executing instructions to increment SEQCOUNT and a program branch depending on the outcome of a comparison between SEQCOUNT and SEQLENGTH. To economise on execution time those instructions are not executed after every configuration change. The preferred technique is to count only those configuration changes which take place on a particular set of edge sites. By counting, for example, only E_2 configuration changes the instructions are executed only a few times and have a negligible effect on the overall computer time.

3.4. *The Initial Configuration*

The spin reversal module may also be used to set up the initial configuration. For each node the location field is assigned to its respective value and the group field is set to zero, giving a lattice of down spins. To place an up spin on the lattice on site I, the spin reversal module is set up as a subroutine and called with node I as an argument.

3.5. *Varying and Fixed Concentration Mode*

The data structure algorithm can also be easily modified to simulate in fixed concentration mode. The technique used is to alternate $A_n \rightarrow B_n$ and $B_n \rightarrow A_n$ configuration changes. To achieve this two data structure algorithms are placed in series. The first algorithm executes spin reversal trials only on sites occupied by a down spin. The second algorithm executes spin reversal trials only on sites occupied by an up spin. This technique requires that extra instructions be inserted to isolate and test the lowest order bit of the group field.

3.6. *Sampling the Configuration*

The lattice of nodes can be sampled to obtain the group probabilities. By definition GROUP (I) is an image of a group expressed as a binary number. Therefore using GROUP (I) as an index the frequency of occurrence of a group can be counted directly into an array, say C, by incrementing C (GROUP (I)). The advantages of the data structure here is that the group probabilities may be obtained for the entire lattice with only one memory reference per node.

3.7. *Algorithm Preparation*

Program segments for the square lattice spin reversed trial and bulk site updating routine were written first. These segments written in COMPASS for a Control Data Cyber 7600 [20] computer are given in Appendix A. The remaining eight updating routines for the square lattice were duplicated from the bulk site updating routine with appropriate address changes as designated in Table II.

3.8. *Data Structure Algorithms for Other Lattices*

The structure of each algorithm is identical to the square lattice algorithm, however, the amount of information stored in each node and the number and size of the updating routines varies from lattice to lattice. In Table III data structure details of the *hcp*, *sc*, *bcc* and *fcc* lattices are compared with details of the square lattice. Spin reversal trials for each lattice are made up of the same machine instructions executed in the same order. The values of operands such as the number of sites, masks and shift counts are sufficient to account for the differences between lattices.

The square lattice updating routine establishes a pattern of flow of nodes into the arithmetic registers and back into memory, which keeps pace with the shifting of the

TABLE III

Data Structure Details for Various Lattice Types

Lattice type	Number of updating routines	Group field (bits)
sq	9	5
hcp	12	7
sc	27	7
bcc	16	9
fcc	32	13

updating operand. With this systematic approach it is straight forward to extend the updating routine to update the larger number of nearest neighbour nodes found in the more complex lattices. The task of writing data structure algorithms for other lattices reduces mainly to determining the nearest neighbour addresses of site I.

4. Results and Discussion

Data structure algorithms were prepared in fixed and varying concentration mode for the *sq*, *hcp*, *sc*, *bcc* and *fcc* lattices. Preparation of code, amounting to 35 machine instructions, for the square lattice spin reversal trial and bulk site updating routine, accounted for the major programming effort. With these segments complete algorithms were prepared for each lattice with the aid of an editing language.

The results reported in this section unless otherwise stated are for single spin reversal simulations, (varying X_A mode) with periodic boundary conditions and random site selections. The lattices sizes are $N = 40{,}000$ for the *sq* and *hcp*, $N = 39{,}304$ for *sc* and *bcc* and $N = 32{,}000$ for the *fcc* lattice. Prior to measuring execution times random and ordered initial configurations were allowed to relax until the values of F, from (8), for configurations from each sequence were indistinguishable from each other.

Recalling the nomenclature of (8) execution time details for the *sq*, *hcp*, *bcc* and *fcc* lattices are compared in Table IV. The results show that the average execution time of a spin reversal trial, S, is 1.15 μsec and this time is unaffected by lattice type. The average execution time of reversing spin I, U, increases approximately linearly with the number of nearest neighbours, from 0.96 μsec, for four nearest neighbours on the *sq* lattice, to 2.69 μsec, for 12 nearest neighbours on the *fcc* lattice. U is unaffected by the number of updating routines as the correct updating routine is chosen by using the computed branch point destination instruction. Simulating in fixed X_A mode for $X_A = 0.5$, the value of S increases by approximately 50%, and U is unaffected.

The average execution times, t, per configuration change measured in zero field at $T/T_c = \infty$ and $T/T_c = 1$ are given in Table V. The results provide a measure of the effects of the lattice type on the efficiency of the data structure algorithm. It is evident

TABLE IV

Values of S and U for Data Structure Algorithm

Lattice type	$S(\mu\text{sec})$	$U(\mu\text{sec})$
sq	1.15	0.96
hcp	1.15	1.6
sc	1.15	1.4
bcc	1.15	1.9
fcc	1.15	2.7

from these results that the algorithms for the two and three dimensional lattices are of comparable efficiency. It is surprising to find, however, that at $T/T_c = 1.0$ the *sc*, *bcc* and *fcc* lattice algorithms are more efficient than the square lattice algorithm. The increase in efficiency is due to a decrease in the number of spin reversal trials per configuration change, as indicated by the values of F, from Table V, measured at $T/T_c = 1.0$.

Any absolute comparison of algorithm efficiency is limited as other workers may use different simulation models and computers and execution times are seldom reported. Some execution times, however, are given in Table VI for square lattice algorithms with single spin reversal. The n-fold algorithm was executed on a CDC 6600 series computer and featured random site selections and periodic boundary conditions. Landau's algorithm was executed on an IBM 370/168 computer and featured free edge boundary conditions and sequential site selections. The data for Landau's algorithm was calculated from (8) using $S = 35\ \mu$sec and $U = 0$ whereas the n-fold data are quoted directly.

The present algorithm when compared to the n-fold algorithm, at $T/T_c = 1.0$, is approximately 45 times more efficient for $2H/kT = 0$ and 40 times more efficient for $2H/kT = 0.1763$. At $T/T_c = 0.588$, $2H/kT = 0$ the advantage in efficiency is reduced to a factor of approximately 8. When compared with Landau's algorithm the present algorithm is approximately 20 and 25 times more efficient at $T/T_c = \infty$ and

TABLE V

Effect of Lattice Type of Algorithm Efficiency

Lattice type	$t\ \mu$sec $(T/T_c = \infty)$	$t\ \mu$sec $(T/T_c = 1)$	F $(T/T_c = 1)$
sq	3.3	8.7	6.9124
hcp	3.9	T_c is unknown	—
sc	3.7	4.8	2.99
bcc	4.2	5.0	2.72
fcc	5.2	6.1	2.63

TABLE VI

Comparison of Algorithm Efficiency (time μsec)

	Present	n-fold	Landau [10]
$T/T_c = \infty$ $H = 0$	3.3	—	70
$T/T_c = 1.0$ $H = 0$	8.7	410	240
$T/T_c = 1.0$ $2H/kT = 0.1763$	24	1000	—
$T/T_c = 0.588$ $H = 0$	55	400	—

$T/T_c = 1.0$, respectively. If Landau's algorithm contained periodic boundary conditions and random site selections its efficiency would be reduced [10].

Unlike the present algorithm the efficiency of other algorithms will be affected by lattice type. If any data could be found for the *hcp*, *sc*, *bcc* and *fcc* lattices then an efficiency comparison would be even more in favour of the present algorithm. For further comparions it is worthwhile reporting that, at $T/T_c = 1.0$, using the data structure algorithm for *sc*, *bcc* and *fcc* lattices, approximately 70, 75 and 90 machine instructions respectively, are executed per configuration change. At these temperatures the number of machine instructions executed per configuration change is quite small, and any further significant decreases seem unlikely.

The data structure algorithm has been demonstrated to be an efficient and versatile tool for the Monte Carlo simulation of Ising models. The algorithm has the advantage of having one program segment which can be used for spin reversal trials for various lattice types and boundary conditions. The simulation of two and three dimensional lattices are now equally practical in terms of both computer time and programming effort.

APPENDIX A

In this appendix a sample algorithm for module 2, written in COMPASS for a CDC 7600 computer is described. The algorithm is for a square lattice with periodic boundary conditions, random site selections and single spin reversal. The algorithm contains an operand initialisation segment, a spin reversal trial segment and a bulk site updating routine.

The random numbers are generated by a multiplication method requiring two operands, a constant multiplier and a current random number. Normally on the 7600 computer a double precision floating multiplication requires both *FX* and *DX* multiplication instructions, to retrieve the full 96 bit coefficient and 24 exponent result. If the *DX* instruction alone is used, the lower 48 bits of the 96 bit result and the larger exponent of the two operands, minus 60 octal, is returned to the designated *X* register. The period of the random number is 2^{48} if the constant multiplier is 2000 1207 2642 7173 0565 octal. The current random number is always between 0 and 1, having an exponent of 2000–60 octal = 1717 octal.

The code is specifically for the CDC 7600 compter. However, some features of the algorithm may be implemented on other machines. The current random number, the random number constant multiplier and the number of lattice sites are three operands given special priority. During operand initialization these operands are permanently stored in the arithmetic registers for the duration of execution of module 2. Execution time and program instructions are saved as these operands are not read-referenced after every spin reversal trial or after every spin reversal. Other operands are either stored in the indexing (*B*) registers or generated at execution time.

```
* Operand Initilisation
     SA5          RANDOM         fetch the current random number
     BX7          X5             transmit the random number to X7
     SA1          RANDOM + 1     fetch the random number multiplier
     SA2          N              fetch the number of lattice sites
     SA5          SEQLENGTH      fetch the halt flag
     SB7          X5             transmit SEQLENGTH to B7
     SB1          1              shift operand
     SB7          6              location field shift operand
     SB6          0              initialise SEQCOUNT

* Spin Reversal Trial
     SPINTRIAL    BSS            0B
     DX7          X1*X7          generate the first RN
     FX0          X2*X4          multiply by N
     DX7          X4*X1          generate the second RN
     UX3          B4, X0         unpack the product
     LX6          B4, X3         left shift to form Index I
     SA5          X6 + L         fetch node I
     SB4          A5             save the address of node I
     MX4          54             prepare a mask
     BX4          –X4*X5         mask the location field
     SA3          X4 + P         fetch the transition probability
     AX4          B2, X5         right shift the location field
     SX0          B1             generate the updating operand
     FX6          X3–X7          transition prob. – RN
     SB5          X4             transmit the location field
     NG           X6, SPINTRIAL  on a negative result branch to SPINTRIAL
     JP           *LOC + B5      branch to updating routine
```

* *Bulk Site Updating Routine*

```
    SA3        B4 + 1          fetch node I + 1
    SA4        B4 - S          fetch node I - S
    BX6        X0-X5           update node I
    LX0        X0, B1          left shift the updating operand
    SA5        B4 - 1          fetch node I - 1
    SA6        B4              store node I
    BX6        X0-X3           update node I + 1
    LX0        X0, B1
    SA6        A3              store node I + 1
    SA3        B4 + S          fetch node I + S
    BX6        X0-X4           update node I - S
    LX0        X0, B1
    SA6        A4              store node I + S
    BX6        X0-X5           update node I - 1
    LX0        X0, B1
    SA6        A5              store node I - 1
    BX6        X0-X3           update node I + S
    SA6        A3              store node I + S
    EQ         SPINTRIAL       branch to SPINTRIAL
```

ACKNOWLEDGMENTS

I am indebted to the late Dr. A. M. W. Verhagen for valuable guidance in the development of the mathematical aspect of this work.

REFERENCES

1. L. D. FOSDICK, *Phys. Rev.* **116** (1959), 565.
2. J. R. EHRMAN, L. D. FOSDICK, AND D. C. HANDSCOMB, *J. Math. Phys.* **1** (1960), 547.
3. K. BINDER, *In* "Topics in Current Physics," Vol. 7, "Monte Carlo Methods in Statistical Physics" (K. Binder, Ed.), Springer-Verlag Berlin, Heidelberg, 1979.
4. W. Z. LENZ, *Physik.* **21** (1920), 613.
5. E. ISING, *Z. Physik* **31** (1925), 253.
6. YU. A. SHREIDER, "The Monte Carlo Method," Oxford Univ. Press, London/New York, 1966.
7. J. M. HAMMERSLEY AND D. C. HANDSCOMB, "Monte Carlo Methods," Methuen, London, 1964.
8. A. E. FERDINAND AND M. E. FISHER, *Phys. Rev.* **185** (1969), 832.
9. K. BINDER, *Physica* **62** (1972), 508.
10. D. P. LANDAU, *Phys. Rev. B* **13** (1976), 2997.
11. D. P. LANDAU, *Phys. Rev. B* **14** (1976), 255.
12. K. BINDER, *Phys. Stat. Sol. B* **46** (1971), 567.
13. E. STOLL, K. BINDER AND T. SCHNEIDER, *Phys. Rev. B* **8** (1973), 3266.
14. M. SUZUKI, *Prog. Theor. Phys.* **43**(1970), 43.
15. W. W. WOOD, "Physics of Simple Liquids," Chap. 5, Wiley, New York, 1968.
16. R. FRIEDBERG AND J. E. CAMERON, *J. Chem. Phys.* **52** (1970), 6049.

17. J. R. Anderson, K. Foger, and R. J. Breakspere, *J. Catal.* **57** (1979), 458.
18. M. P. Harding and P. J. Bunyan, *J. Phys. A* **13** (1980), 3243.
19. A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, *J. Comp. Phys.* **17** (1975), 10.
20. Control Data 7600 Series Cyber 70/Model 76, "Hardware Reference Manual" Publication 60367200.