# Monte Carlo Studies of Critical Phase Behaviors of Compressible Ising Models

by

Xiaoliang Zhu

(Under the direction of David P. Landau)

## Abstract

Monte Carlo simulations of two different compressible Ising models are presented. One is an elastic, antiferromagnetic Ising model on a distortable diamond net at constant pressure. Spins interact via a Stillinger-Weber-like potential, and data were obtained over a wide range of temperature and magnetic field. The phase boundary is a line of 2nd order transitions between ordered and disordered states. Our analysis shows that this model shares the same critical exponents with the rigid 3-D simple-cubic Ising model, despite the difference in structure and interactions. This implies that they both belong to the same universality class. We also present a thorough examination of the model's elastic degrees of freedom. The other model is a stacked triangular lattice where spins interact via Lennard-Jones potential. This model features adjustable elasticity. However, analysis shows that the phase transition also belongs to the rigid 3-D simple-cubic Ising universality. Both results contradict theoretical predictions.

Index words: Ising, antiferromagnet, ferromagnet, critical point, phase transition, compressible Ising, elastic Ising, Monte Carlo, finite size scaling, histogram reweighting, visualization

# Monte Carlo Studies of Critical Phase Behaviors of Compressible Ising Models

by

Xiaoliang Zhu

B.S., The University of Science and Technology of China, China, 1998

M.S., The University of Georgia, Athens, GA, 2002

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2005

# Monte Carlo Studies of Critical Phase Behaviors of Compressible Ising Models

by

Xiaoliang Zhu

Approved:

Major Professor: David P. Landau

Committee: Steven P. Lewis
Robin L. Shelton
Shan-Ho Tsai

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2005

# Dedication

To Mom and Dad

## Acknowledgments

I am extremely thankful to my major professor, David P. Landau for his invaluable guidance and encouragement throughout my entire research and academic years. I have learned to analyze problems thoroughly, pay attention to details, and communicate efficiently.

I would also like to thank Steven P. Lewis, Robin L. Shelton and Shan-Ho Tsai for serving on my advisory committee, and for their advice and guidance. Steven Lewis has given me tremendous advice and help in my academic endeavor, especially in my job seeking. Robin Shelton has constantly encouraged and advised me. I owe Shan-Ho Tsai a great deal for her great patience in helping me understand the simulation code and maintaining extraordinary computation facility.

I thank Steven Mitchell for his generous help in developing the visualization tools, educating me with American culture, and many more. J. A. Plascak gave me hands-on guidance in reweighting histogram distribution. I should also thank B. Dünweg, N. Branco, F. Tavazza and L. Cannavacciuolo for helpful discussions.

Some of the calculations were performed at TACC (Texas Advanced Computing Center). This research was supported in part by NSF Grant No. DMR-0341874.

# Table of Contents

# Chapter 1

## Introduction

The Ising model is among the most important and most intensely studied models of statistical physics. The original Ising model is defined on a lattice on which each lattice site has a spin that can be +1 or -1 and that interacts with its nearest neighbors with constant $J$. The Hamiltonian typically looks like

$$\mathcal{H} = -J \sum_{<i,j>} \sigma_i \sigma_j - h \sum_i \sigma_i, \tag{1.1}$$

where $\sigma_i$ is the spin value at site $i$, the summation $< i, j >$ is over all nearest neighboring sites, and $h$ is the external magnetic field. If $J > 0$, the system is termed ferromagnetic, meaning neighboring spins tend to have the same values to minimize the total energy. If $J < 0$, the system is called antiferromagnetic, and neighboring spins have a tendency to have opposite values to lower the total energy. The language of the Ising model – spin, ferromagnetic, antiferromagnetic, etc. – reflects its original use to describe magnetic systems. However, this model has far broader applicability for describing systems with binary degrees of freedom. For example, in binary alloys the "spin" variable represents atomic species, not quantum mechanical spin.

Ising models have served as fertile testing grounds for enormous projects in statistical mechanics, in particular phase transitions and critical phenomena. Phase transitions have been a topic of great interest in many fields of physics, but most realistic models of phase transitions are beyond analytical solutions. Typical Ising-like transitions include, for instance, the gas-liquid transition and mixing-unmixing in liquids. One- and two-dimensional ( only in the absence of external field ) Ising models have been solved analytically [1]: the 1d Ising

model has no phase transition, and the 2d Ising model has a second order phase transition. No analytical solution has been found for the 3d Ising model. The simplest theoretical treatment of Ising models is the mean field theory, but the mean field idea was found to be quantitatively incorrect in the neighborhood of thermodynamic critical points[2, 3, 4]. The source of this failure is clear. Fluctuations are at the heart of critical phenomena, whereas mean field theory is based upon the assumption of small fluctuations. Various approaches, including renormalization group, $\epsilon$-expansion, series expansion [5, 6, 7, 8], and Monte Carlo simulations have been used to study the phase transition of the 3d Ising model [9, 10].

Traditional Ising models are rigid: Each lattice site is fixed at some position, and spins interact with a fixed number of neighbors with fixed interaction constants. While it provides the advantages of simplicity and high computational speed, this also imposes limitations on its modeling capabilities for realistic systems, especially when it is used to model binary alloys. A binary alloy is equivalent to an Ising model in that the up and down spins in an Ising model are equivalent to the two different atomic species in a binary alloy, and the magnetic field is proportional to the difference between the two chemical potentials in a binary alloy. Throughout our discussion we will use the term Ising model interchangeably with binary alloy. Atoms in alloys move around their equilibrium positions at finite temperature $T$. The motion of the atoms can be correlated spatially and contribute significantly to the thermodynamic properties around critical points, possibly even changing the nature of the transitions. As a result, compressible Ising models have gained more and more popularity. In a compressible Ising model, the lattice is elastic and can be deformed. The interaction $J$ is no longer constant, but depends on the bond lengths and angles between bonds. A compressible three-dimensional Ising model is even more difficult to deal with than its rigid counterpart. Many numerical studies have been done on compressible Ising models [11, 12, 13, 14, 15], mainly driven by growing interest on semiconductor alloys in 1990's.

The theoretical approaches based on perturbation expansions have been a great success, but they were found to be unreliable in many cases, especially in frustrated systems with

complicated types of magnetic order parameters[16]. Numerical methods such as Molecular Dynamics and Monte Carlo simulations, on the other hand, suffer from the limitations of the computer hardware and can only deal with limited system sizes. Therefore, it might be difficult to see the asymptotic behavior from the numerical results. The choice of numerical methods rely on the ability to deal with any complicated models. Thanks to the availability of high-performance computer facilities with ever-growing speed and developments of computational algorithms such as finite size scaling[17, 18], histogram reweighting[19] and Wang-Landau sampling [20, 21], Monte Carlo simulations have become one of the most powerful tools in providing accurate information about the nature of the phase transitions of Ising models.

This dissertation also attempts to test the theoretical work by B. Dünweg. Inspired by Monte Carlo studies[13, 14], Dünweg [22] conducted a systematic theoretical investigation on phase transitions of elastic Ising models with generic Landau-Ginzburg-Wilson (LGW) Hamiltonians. He identified 4 distinct cases:

1. A ferromagnetic system at constant pressure: a mean-field phase transition

2. A ferromagnetic system at constant volume: two first-order phase transitions ending in critical points

3. An antiferromagnetic system at constant pressure: a first-order phase transition from an ordered to disordered phase

4. An antiferromagnetic system at constant volume: a second-order phase transition with Fisher renormalized exponents, from an ordered to disordered phase

The first prediction agrees with the simulational results reported in Ref. [14]. However, the second one doesn't even agree qualitatively with more recent simulational results[23], which raised questions about the validity of the last two predictions. On the other hand, very little work has been done in the area of elastic antiferromagnets. The first part of this dissertation will concentrate our simulational results for the third case, i.e., elastic antiferromagnet at

constant pressure. To the best of our knowledge, nobody has done any simulational research on this model – antiferromagnet on a distortable diamond net with Stillinger-Weber (SW) potential. This diamond model is carefully chosen so that it is only slightly different from its ferromagnetic counterpart – the SiGe alloy, in the hope that it can be compared with earlier numerical results [14] and theoretical predictions.

The diamond model has a drawback: its elasticity is hard to assess, and the coupling strength between the elasticity and the Hamiltonian is unclear. The second model, a ferromagnet on a stacked triangular lattice with Lennard-Jones (LJ) potential, has been studied by a French group [24, 25]. We study this model because we find their results questionable, and mainly because it features adjustable elasticity.

The rest of this dissertation is organized as follows. The simulational and theoretical backgrounds will be presented in Chapter 2. The compressible diamond model, including interactions, structures and simulational details will be described in Chapter 3. In Chapter 4 the simulation results for the diamond models will be reported, and the effect of the elasticity in the distortable diamond net will be discussed in Chapter 5. In chapter 6 the stacked triangular lattice model and simulation results will be presented. Finally, the dissertation will conclude in Chapter 7.

# Chapter 2

# Background

## 2.1 Phase Transitions

Let's start with a general discussion of phase transitions in Ising models [26]. For a typical rigid Ising model, the energy of the system is:

$$\mathcal{H}\{\sigma\} = -J \sum_{<r,r'>} \sigma(r)\sigma(r') - h \sum_{r} \sigma(r), \tag{2.1}$$

where $J$ is the interaction constant, $\sigma(r) = \pm 1$ is the spin at lattice site $r$, $\{\sigma\}$ is a configuration of all spins. the summation in the first term is over all distinct pairs of nearest-neighbor spins, and $h$ is the magnetic field. All thermodynamic properties are defined by the partition function $Z$,

$$Z(T) = \sum_{\{\sigma\}} \exp[-\mathcal{H}\{\sigma\}/k_B T] = \sum_{\{\sigma\}} \exp\left[-\beta \mathcal{H}\{\sigma\}\right] \tag{2.2}$$

where $k_B$ is the Boltzmann constant, $T$ is the absolute temperature, and $\beta = 1/k_B T$. If we define two parameters

$$K_1 = h/kT = \beta h \tag{2.3}$$

$$K_2 = J/kT = \beta J. \tag{2.4}$$

and two extensive operators

$$S_1 = \sum_{r} \sigma(r) \tag{2.5}$$

$$S_2 = \sum_{<r,r'>} \sigma(r)\sigma(r') \tag{2.6}$$

Then we have

$$-\beta\mathcal{H}\{\sigma\} = K_1 S_1 + K_2 S_2. \tag{2.7}$$

In general, we utilize the notation $S_\alpha$ for the $\alpha^{\text{th}}$ extensive operator in the theory, and $\mathbf{K}$ for the parameter set $(K_1, K_2)$. In terms of these quantities, we can define a free energy density $f(\mathbf{K})$ as

$$f(\mathbf{K}) = \frac{1}{N} \ln Z(\mathbf{K}) = \frac{1}{N} \ln \sum_{\sigma} e^{-\beta\mathcal{H}\{\sigma\}} \tag{2.8}$$

The average of any function of the spins $\theta\{\sigma\}$ is given by

$$\theta\{\sigma\} = \frac{1}{Z} \sum_{\{\sigma\}} \theta\{\sigma\} \exp(-\beta\mathcal{H}\{\sigma\}) \tag{2.9}$$

The magnetization is defined as the average (over all lattice sites) value of $\sigma(r)$

$$m(\mathbf{K}) =< \sigma(r) >= \frac{\partial f(\mathbf{K})}{\partial \mathbf{K}_1} \tag{2.10}$$

In two or higher dimensions, the Ising system may undergo phase transitions by changing temperature $T$ or magnetic field $h$. Fig. 2.1 shows the phase diagram in $h - T$ space. When $T < T_c$, most spins point up ($\sigma = +1$) for $h > 0$ and point down ($\sigma = -1$) for $h < 0$. Along the path $A \to B \to C$, there is a first order phase transition at $B$. There is no transition along $A \to D \to C$. The first order line (or phase boundary) ends at $T_c$, where it becomes 2nd order. The most natural parameters to use in the description of the critical behavior of the Ising model are the magnetic field variable $h$ and the reduced temperature

$$t = \frac{T - T_c}{T_c},$$

where $T_c$ is the infinite lattice critical temperature. Note that the direction of increase of $h$ is perpendicular to the phase boundary, while that of increase of $t$ is parallel to it. Generally speaking, phase transition problems can be defined in terms of these two fields:

(a) A field $h$, which drives the system across the phase boundary and vanishes at criticality;

(b) A field $t$, which moves the system along the phase boundary and also vanishes at the critical point. We will come back to this topic again when we try to reweight a histogram distribution by field-mixing in the next chapter.

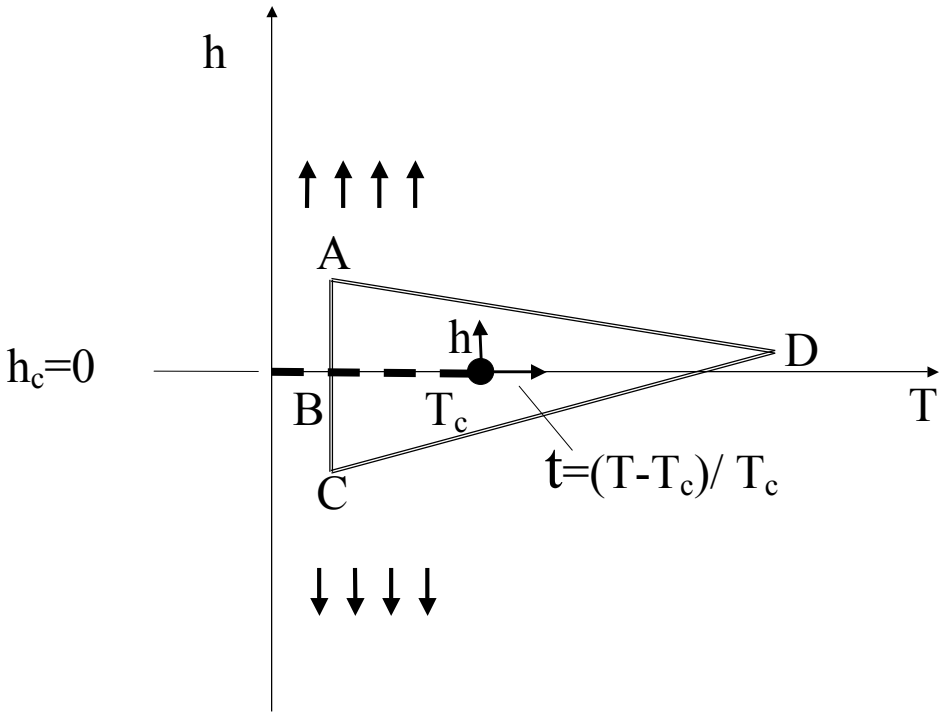


Figure 2.1: The phase diagram of a typical Ising model described by Eq. 2.1. The dashed line indicates a first order transition line. The dot at the end of the dashed line indicates the critical point where the transition becomes second order. The arrows originating from the critical points are the directions of increases of $h$ and $t$, respectively. The two groups of parallel arrows indicate the spin orientations.

Table 2.1: Definition of critical indices

| Critical index | Definition | Condition | |
|---|---|---|---|
| $\beta$ | $m \simeq \pm(-t)^{\beta}$ | $t < 0$ | $h = 0$ |
| $\gamma$ | $\chi \simeq t^{-\gamma}$ | $t > 0$ | $h = 0$ |
| $\gamma'$ | $\chi \simeq (-t)^{-\gamma'}$ | $t < 0$ | $h = 0$ |
| $\delta$ | $m \simeq h^{1/\delta}$ | $t = 0$ | |
| $\alpha$ | $C_h \simeq t^{-\alpha}$ | $t > 0$ | $h = 0$ |
| $\nu$ | $\xi \simeq t^{-\nu}$ | $t > 0$ | $h = 0$ |

### 2.1.1 Critical Singularities

At the critical point, many thermodynamic quantities exhibit power-law singularities. The fundamental source of these singularities is the divergence in the correlation length $\xi$ at criticality. Near criticality, $\xi(t)$ scales as $t^{-\nu}$, $\xi \to \infty$ as $t \to 0$. The definition of the critical indices ( exponents ) are given in Table 2.1, where $m$ denotes magnetization, $\chi$ for susceptibility, and$C_h$ for specific heat capacity at external magnetic field $h$. These exponents are not independent. The following relationships hold.

$$2 - \alpha = \gamma + 2\beta = \gamma' + 2\beta = \beta(\delta + 1). \tag{2.11}$$

For further discussion of these relationships and of critical singularities, in general, see Ref. [26].

### 2.1.2 Universality

Since critical phenomena arise from long-ranged correlations, it is reasonable to expect that some of the details of the interatomic potential might be quite irrelevant to the behavior in the critical region. Thus, for example, it is usually asserted that the values of the critical indices are independent of interaction details, as in the statement of universality hypothesis. In its simplest terms, the universality hypothesis is the statement that all critical problems

may be divided into classes differentiated in part by:

1. The dimensionality of the system;
2. The symmetry group of the system; and
3. Spin dimensionality

Within each class, the critical exponents are supposed to be identical or, at worst, to be a continuous function of a very few parameters.

## 2.2 Monte Carlo Simulation

The Monte Carlo simulation method is a stochastic sampling technique, where random numbers are generated to mimic the fluctuations that occur in nature, in order to simulate a model of interest. It was named after Monte Carlo, Monaco, where the primary attractions are casinos containing games of chance. Monte Carlo simulation has been applied in a broad range of areas from economics to nuclear physics to regulating the flow of traffic. It has the advantages of simplicity and power.

In the simulation, the Metropolis importance sampling method is used to generate configurations from a previous state with a transition probability which depends on the energy difference between the initial and final states. The transition probability has to satisfy detailed balance

$$P_n(t)W_{n\to m} = P_m(t)W_{m\to n}, \tag{2.12}$$

where $P_n(t)$ is the probability of the system being in state $n$, and $W_{n\to m}$ is the transition rate for $n \to m$. In a classical system that follows the Boltzmann distribution, $P_n(t)$ is given by

$$P_n(t) = \exp(-E_n/k_BT)/Z, \tag{2.13}$$

where $Z$ is the partition function. So we have

$$\frac{W_{n\to m}}{W_{m\to n}} = \frac{P_m(t)}{P_n(t)} = \exp(-\Delta E/k_B T), \tag{2.14}$$

where $\Delta E = E_m - E_n$. Any transition rate which satisfies detailed balance is acceptable. The first choice of rate that was used in statistical physics is the Metropolis form [27]

$$W_{n\to m} = \exp\left(-\Delta E/k_B T\right), \quad \text{if } \Delta E > 0 \tag{2.15}$$

$$= 1, \quad \text{if } \Delta E < 0 \tag{2.16}$$

where time unit is set to unity and suppressed in the equations. The recipe for the Metropolis algorithm follows.

1. Choose an initial state
2. Choose a site i
3. Calculate the energy change $\Delta E$ which results if the spin at site $i$ is overturned
4. Generate a uniform random number $r$ in the interval $[0, 1]$.
5. If $r < \exp\left(-\Delta E/k_B T\right)$, flip the spin
6. Go to the next site and go to step 3

The "standard measure" of Monte Carlo time is the Monte Carlo step/site (MCS/site) which corresponds to the consideration of every spin in the system once. After a sufficiently long run, this algorithm generates states that follow the Boltzmann distribution, i.e., the occurrences of a state are proportional to Eq. 2.13. Then, the desired average $< A >= \sum_n P_n A_n$ of a variable $A$ simply becomes the average over the entire sample of states which is kept.

The Metropolis flipping method is not the unique solution. An alternative method is known as 'Glauber dynamics' [28], uses the single spin-flip transition rate

$$W_{n\to m} = 1 + \tanh\left(\sigma_i E_i/k_B T\right), \tag{2.17}$$

where $\sigma_i E_i$ is the energy of the $i^{th}$ spin in state $n$. The transition rate is anti-symmetric about 0.5 for $E_i \to -E_i$. In most situations the choice between Glauber and Metropolis dynamics is arbitrary; but in at least one instance they are different. At very high temperature the Metropolis algorithm will flip a spin on every attempt because the transition probability approaches 1 for $\Delta E > 0$. Thus, in one sweep through the lattice every spin overturns, and in the next sweep every spin overturns again. The system just oscillates between two states and the process becomes non-ergodic. With the Glauber algorithm, however, the transition probability approaches 1/2 in this instance and the process remains ergodic. Refer to the book by Landau and Binder[29] more information about these two methods.

## 2.3 Finite-Size Scaling Analysis

Computer simulations can only handle finite size systems, whereas we are interested in the critical behavior of nearly infinite systems. According to Fisher's finite-size scaling theory[17, 18], the critical behavior of an infinite system may be extracted from that of finite systems by examining the size dependence of the singular part of the free energy density. The free energy of a system of linear dimension $L$ is described by the scaling ansatz

$$F(L, T, h) = L^{-(2-\alpha)/\nu} \mathcal{F}^0(tL^{1/\nu}, hL^{(\gamma+\beta)/\nu}). \tag{2.18}$$

As a reminder, $t = (T - T_c)/T_c$ ($T_c$ is the infinite-lattice critical temperature) and $h$ is the magnetic field. The critical exponents $\alpha$, $\beta$, $\gamma$, and $\nu$ are all the appropriate values for the infinite system. Based on this scaling ansatz, at zero field, i.e., $h = 0$, we may obtain the following scaling form for magnetization per spin

$$m = L^{-\beta/\nu} \tilde{m}(x_t), \tag{2.19}$$

where $x_t = tL^{1/\nu}$ is the temperature scaling variable, and $m = \frac{1}{N} M = \frac{1}{N} \sum_j \sigma_j$ is the magnetization per spin. $N$ is the total number of spins in the system.

The specific heat capacity $C$ can be calculated from the fluctuations of the internal energy $E$

$$C = \frac{1}{N}\frac{1}{T^2}(< E^2 > - < E >^2), \tag{2.20}$$

the finite-lattice susceptibility from the fluctuations of the magnetization $m$

$$\chi = \frac{N}{T}(< |m^2| > - < |m| >^2), \tag{2.21}$$

and the Binder cumulant[30]

$$U_4 = 1 - \frac{< m^4 >}{3 < m^2 >^2}. \tag{2.22}$$

We also have the following scaling forms for these three quantities:

$$C = L^{\alpha/\nu}\tilde{C}(x_t), \tag{2.23}$$

$$\chi(T) = L^{\gamma/\nu}\tilde{\chi}(x_t), \tag{2.24}$$

$$U_4(T) = \tilde{U}(x_t). \tag{2.25}$$

As in Ref. [9], the finite-lattice (or effective) critical temperature $T_c(L)$ is defined to be where the scaling function reaches maximum. The reciprocal of the effective critical temperature, or the effective critical coupling, $K_c(L) = 1/T_c(L)$, has the following scaling form

$$K_c(L) = K_c + \lambda L^{-1/\nu}(1 + bL^{-\omega}) \tag{2.26}$$

where $K_c$ is the critical coupling of infinite lattice, and $bL^{-\omega}$ is an approximation for the series of higher order power-law correction terms.

Binder[30] showed that the maximum slope of the cumulant $U_4$ at $K_c$ varies with system size like $L^{1/\nu}$. Taking into account a correction term, the size dependence becomes

$$\left.\frac{dU_4}{dK}\right|_{max} = aL^{1/\nu}(1 + bL^{-\omega}) \tag{2.27}$$

The logarithmic derivative of any power of the staggered magnetization

$$\begin{aligned} \frac{\partial}{\partial K}\ln < m^n > &= \frac{1}{< m^n >}\frac{\partial}{\partial K} < m^n > \\ &= \left[\frac{< m^n E >}{< m^n >} - < E >\right], \end{aligned} \tag{2.28}$$

has the same scaling properties as the cumulant slope. This provides us with additional estimates for $\nu$ and $K_c(L)$.

## 2.4 Histogram Reweighting Method

The Monte Carlo method suffered from the huge amount of computer resources required for thorough and accurate results until the introduction of the histogram reweighting method by Ferrenberg and Swendsen [19]. This method increases the amount of information obtained from a single simulation, rather than just taking the averages and standard deviations of thermodynamic quantities. It has proven to be very effective and yields excellent results in the neighborhood of the point where a sufficiently long MC simulation is performed.

The Hamiltonian for an Ising system is given in Eq. 2.7. The probability distribution of ($S_1$, $S_2$) ( see Eqns. 2.5 and 2.6) at a point ($K_1$, $K_2$) in the parameter space is given by

$$P_{(K_1,K_2)}(S_1,S_2) = \frac{1}{Z(K_1,K_2)} N(S_1,S_2)\exp(K_1S_1 + K_2S_2), \tag{2.29}$$

where $N(S_1,S_2)$ is the number of configurations at the point $(S_1,S_2)$ in the phase space, and $Z(K_1,K_2)$ is the canonical partition function given by

$$Z(K_1,K_2) = \sum_{S_1,S_2} N(S_1,S_2)\exp(K_1S_1 + K_2S_2). \tag{2.30}$$

From Eq. 2.29, we have

$$N(S_1,S_2) = P_{(K_1,K_2)}(S_1,S_2)\exp(-K_1S_1 - K_2S_2)Z(K_1,K_2). \tag{2.31}$$

We apply Eq. 2.29 at a new point in parameter space $(K_1', K_2')$, then we have

$$\begin{aligned} P_{(K_1',K_2')}(S_1,S_2) &= \frac{1}{Z(K_1',K_2')} N(S_1,S_2)\exp(K_1'S_1 + K_2'S_2), \\ &\propto P_{(K_1,K_2)}(S_1,S_2)\exp[(K_1'-K_1)S_1 + (K_2'-K_2)S_2]. \end{aligned} \tag{2.32}$$

The histogram $H(S_1,S_2)$ at a point $(S_1,S_2)$ in phase space generated by the MC simulation is proportional to $P_{(K_1,K_2)}(S_1,S_2)$. If we normalize the histogram distribution, then we

should have $H(S_1, S_2) = P_{(K_1,K_2)}(S_1, S_2)$. For normalized probability distribution, we have

$$P_{(K_1',K_2')}(S_1, S_2) = \frac{P_{(K_1,K_2)}(S_1, S_2) \exp[(K_1' - K_1)S_1 + (K_2' - K_2)S_2]}{\sum_{(S_1,S_2)} P_{(K_1,K_2)}(S_1, S_2) \exp[(K_1' - K_1)S_1 + (K_2' - K_2)S_2]} \tag{2.33}$$

The denominator in Eq. 2.33 serves as an estimate for the partition function. We can then use $P_{(K_1',K_2')}(S_1, S_2)$ to calculate the quantities of interest, such as the average internal energy, without having to do very long (time-consuming) Monte Carlo simulations at $(K_1', K_2')$. We will talk more about histogram reweighting in combination with the Hamiltonian of the model in next chapter.

## 2.5 Non-linear curve fitting

The curve fittings are done by the Levenberg-Marquardt method[31] which works very well in practice and has become the standard of nonlinear least-square routines. It can fit any differentiable function with any number of parameters. The drawback of this method is that it is sensitive to initial estimate values, i.e., can be trapped in local minima. If the initial values are chosen to be in the neighborhood of the global minimum, this method can yield excellent and robust fitting results.

# Chapter 3

# The Antiferromagnetic Diamond Model

Compressible Ising systems have been studied as models for SiGe binary alloys on the elastic diamond net under various conditions [13, 14, 23]. Existing codes are readily available as a result of these works. These codes are efficient and easy to be tailored for new models. In order to compare with these results and save coding time, we will once again study the antiferromagnetic compressible Ising model on the diamond net, or to be precise, ordering binary alloys having the ZnS (zinc sulfur) structure.

## 3.1 Lattice Topology

To take into account the contribution of bond elasticity and retain the computational efficiency, we assume that spins are always located on the nodes of a diamond network with fluctuating bonds. This is an intermediate approach between a totally free-moving one and a lattice-gas-like one. We also neglect vacancies and interstitials because of their vanishing concentrations in real systems. Although nodes can move stochastically, the topology of the lattice is fixed. For each node, the 4 nearest neighbors and the 12 next nearest neighbors are known at the very beginning and are used throughout the simulation.

The diamond network consists of two FCC sublattices. Spins in the two FCC sublattices are opposite in a totally ordered antiferromagnetic phase. In the simulation, the diamond network is further decomposed into eight simple cubic(SC) sublattices. No two spins within the same SC sublattice interact with each other in this model.

Fig. 3.1 shows the structure of a unit cell. Each unit cell consists of 8 spins. If the number of unit cells is $L$ in each of the $x$, $y$ and $z$ directions, then the total number of spins $N = 8L^3$.

Table 3.1: The coordinates of spins in a unit cell, relative to the spin in sublattice 1

| | | | |
|---|---|---|---|
| spin 1 | $(0,0,0)$ | spin 5 | $(1,1,1)$ |
| spin 2 | $(2,2,0)$ | spin 6 | $(3,3,1)$ |
| spin 3 | $(2,0,2)$ | spin 7 | $(3,1,3)$ |
| spin 4 | $(0,2,2)$ | spin 8 | $(1,3,3)$ |

Table 3.2: The nearest neighbor lists for spins in unit cell $< i,j,k >$.

| spin | nearest neighbors |
|---|---|
| $< i,j,k,1 >$ | $< i,j,k,5 >$ , $< i,j-1,k-1,8 >$ , $< i-1,j-1,k,6 >$ , $< i-1,j,k-1,7 >$ |
| $< i,j,k,2 >$ | $< i,j,k,5 >$ , $< i,j,k,6 >$ , $< i,j,k-1,7 >$ , $< i,j,k-1,8 >$ |
| $< i,j,k,3 >$ | $< i,j,k,5 >$ , $< i,j,k,7 >$ , $< i,j-1,k,6 >$ , $< i,j-1,k,8 >$ |
| $< i,j,k,4 >$ | $< i,j,k,5 >$ , $< i,j,k,8 >$ , $< i-1,j,k,6 >$ , $< i-1,j,k,7 >$ |
| $< i,j,k,5 >$ | $< i,j,k,1 >$ , $< i,j,k,2 >$ , $< i,j,k,3 >$ , $< i,j,k,4 >$ |
| $< i,j,k,6 >$ | $< i,j,k,2 >$ , $< i+1,j+1,k,1 >$ , $< i+1,j,k,4 >$ , $< i,j+1,k,3 >$ |
| $< i,j,k,7 >$ | $< i,j,k,3 >$ , $< i+1,j,k+1,1 >$ , $< i+1,j,k,4 >$ , $< i,j,k+1,2 >$ |
| $< i,j,k,8 >$ | $< i,j,k,4 >$ , $< i,j+1,k+1,1 >$ , $< i,j,k+1,2 >$ , $< i,j+1,k,3 >$ |

If we set the lower left corner spin as the origin, then the coordinates $(x, y, z)$ of the 8 spins in the order of sublattice index are listed in Table 3.1, which also shows the ordering scheme of the SC sublattices. The unit is a quarter of the unit cell side length. The topological nearest neighbor lists for each of the 8 spins in unit cell $< i,j,k >$ are given in Table 3.2, where $i$, $j$, $k$ run from 1 to $L$. Spin $< i,j,k,s >$ refers to the $s^{\text{th}}$ spin in unit cell $< i,j,k >$. Table 3.2 is particularly important for understanding the simulation code and developing visualization tools.

Each spin in the system is described by four degrees of freedom: The first one is spin value $S_i$ which is either +1 or -1. The other three are the three coordinates of the spin $\vec{r_i}$.

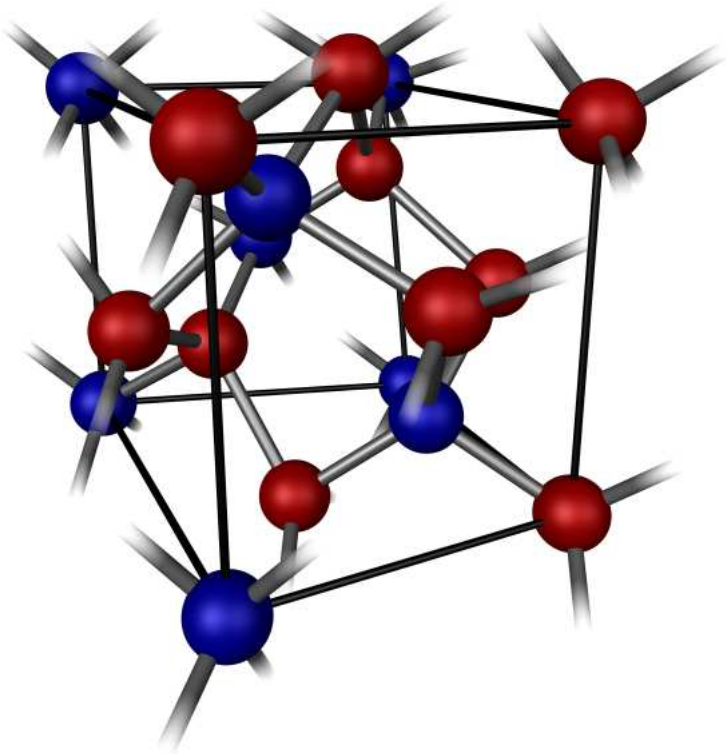

Figure 3.1: The structure of a unit cell in the diamond (or ZnS if totally ordered) structure. The red spheres represent the up spin (+1), and the blue ones represent down spins (-1). The metallic lines connecting spins are nearest neighbor bonds. The black lines connecting the corners are not bonds. They are used to form the frame of the unit cell.

## 3.2 The Hamiltonian

Various empirical interaction models have been proposed[32, 33, 34, 35]. We choose the Stillinger-Weber (SW) potential for the purpose of comparison with Laradji and Landau's work[14] which agrees with the theoretical prediction [22] about the mean-field critical behavior of the compressible ferromagnet at constant pressure. Keep in mind that the theoretical prediction does not depend on the specific potential. The Hamiltonian consists of four parts.

$$\mathcal{H} = \mathcal{H}_1 + \mathcal{H}_1^+ + \mathcal{H}_2 + \mathcal{H}_3 \tag{3.1}$$

where $\mathcal{H}_1$ and $\mathcal{H}_1^+$ are the uniform magnetic field energy and staggered magnetic field energy, respectively.

$$\mathcal{H}_1 = -h \sum_j S_j \tag{3.2}$$

$$\mathcal{H}_1^+ = -h^+ \sum_j S_j^+ \tag{3.3}$$

The staggered spin $S_j^+$ is defined as

$$S_j^+ = \begin{cases} S_j & \text{if } S_j \text{ is in FCC sublattice 1} \\ -S_j & \text{if } S_j \text{ is in FCC sublattice 2} \end{cases}$$

The **staggered magnetization** $M^+$, also called the **order parameter** in the antiferromagnetic case, is the summation of all $S_j^+$,

$$M^+ = \sum_j S_j^+. \tag{3.4}$$

The **concentration** is the ratio of the number ($N_+$) of spin +1's to the total number ($N$) of spins in the system.

$$\text{concentration} = \frac{N_+}{N} \tag{3.5}$$

The two-body part $\mathcal{H}_2$ and three-body part $\mathcal{H}_3$ together are the SW potential energy (see Ref. [14] for details). $\mathcal{H}_2$ can be written as follows:

$$\mathcal{H}_2 = \sum_{<i,j>} \epsilon(S_i, S_j) F_2 \left[ \frac{r_{ij}}{\sigma(S_i, S_j)} \right], \tag{3.6}$$

where the sum is performed over all nearest-neighbor bonds $< i, j >$. $\epsilon(S_i, S_j)$ corresponds to the binding energies: $\epsilon(+1, -1) = \epsilon(-1, +1) = 2.3427eV$, $\epsilon(+1, +1) = 2.17eV$, $\epsilon(-1, -1) = 1.93eV$. $\sigma(S_i, S_j)$ corresponds to ideal bond lengths: $\sigma(+1, +1) = 2.34779$Å, $\sigma(-1, -1) = 2.44598$Å, $\sigma(+1, -1) = \sigma(+1, -1) = 2.396885$Å. These parameters come from the SW potential for the SiGe binary alloy, and are the same as those in Ref. [14], except that $\epsilon(+1, -1)$ is increased from the original $2.0427eV$ to $2.3427eV$ to make the system antiferromagnetic. The choice of this new value is arbitrary as long as it is sufficiently larger than the binding energy between two spin +1's ($2.17eV$) and that between two spin -1's ( $1.93eV$). We will talk more about the choice of $\epsilon(+1, -1)$ a bit later. The function $F_2$ depends on the rescaled bond length $y = r_{ij}/\sigma(S_i, S_j)$:

$$F_2(y) = \begin{cases} A \left[ \frac{B}{y^p} - \frac{1}{y^q} \right] e^{\delta/(y-b)} & \text{for } y < b \\ 0 & \text{for } y \geq b. \end{cases} \tag{3.7}$$

The parameters of the function $F_2$ are: $A = 7.049556277$, $B = 0.6022245584$, $p = 4$, $q = 0$, $\delta = 1$, and $b = 1.80$. $F_2$ reaches a minimum value $-1$ at $y = 2^{1/6}$.

The three-body interaction is

$$\begin{aligned} \mathcal{H}_3 = & \sum_{<i,j,k>} \left[ \epsilon(S_i, S_j) \epsilon(S_j, S_k) \right]^{1/2} \mathcal{L}(S_i, S_j, S_k) \\ & \times F_3 \left[ \frac{r_{ij}}{\sigma(S_i, S_j)}, \frac{r_{jk}}{\sigma(S_j, S_k)} \right] \\ & \times (cos\theta_{ijk} + \frac{1}{3})^2, \end{aligned} \tag{3.8}$$

where the sum is over all triplets $< i, j, k >$ with the vertex at site $j$ ($i$ and $k$ are nearest neighbors of $j$). The cosine of the angle between $\vec{r}_{ji}$ and $\vec{r}_{jk}$ is given by

$$\cos\theta_{ijk} = \frac{\vec{r}_{ji} \cdot \vec{r}_{jk}}{r_{ji} r_{jk}}.$$

If $\cos\theta_{ijk} = 109.47°$, which is the characteristic bond angle of diamond, then $\cos\theta_{ijk} = -\frac{1}{3}$; therefore, the energy contribution from the triplet $< i, j, k >$ becomes zero. This helps the system stabilize towards a diamond structure. $F_3$ is a non-negative function of the rescaled bond lengths:

$$F_3(y_1, y_2) = \begin{cases} e^{\gamma/(y_1-b)+\gamma/(y_2-b)} & \text{for } y_1 \leq b \text{ and } y_2 \leq b \\ 0 & \text{otherwise,} \end{cases} \tag{3.9}$$

where the constant $\gamma = 1.20$.

The function $\mathcal{L}$ is written as follows:

$$\mathcal{L}(S_i, S_j, S_k) = \left[\lambda(S_i)\lambda(S_j)^2\lambda(S_k)\right]^{1/4}, \tag{3.10}$$

where $\lambda(+1) = 21.0$, $\lambda(-1) = 31.0$.

Now we talk more about the choice of $\epsilon(+1, -1)$. For simplicity, we can ignore $\mathcal{H}_3$ here (although it is included in the Monte Carlo simulation) since it is about two orders of magnitude smaller than $\mathcal{H}_2$. Fig. 3.2 shows the comparison of two-spin interactions when $\epsilon(+1, -1)$ assumes different values. When $\epsilon(+1, -1)$ is only slightly larger than $\epsilon(+1, +1)$ which is 2.17, say, $\epsilon(+1, -1) = 2.18$, there is too much overlap between the two energy curves such that, even around the equilibrium, the system prefers one bond type over the other at one bond length, and has the opposite preference at a different bond length. The system would not necessarily prefer to be antiferromagnetic.

Fig. 3.3 shows the energy curves when we further change another parameter $\sigma(+1, -1)$ to be equal to $\sigma(+1, +1)$. In this case, the mixing bonds (+1,-1) is always more preferable than any pure bond (+1,+1) or (-1,-1) around equilibrium. The system is expected to be more stable in this case, but I did not try this parameter set because the analysis of parameter choice was performed after the simulation was done.

## 3.3 The Code Implementation

In the FORTRAN code implementation, the initial state is the ground state of a rigid Ising system unless a configuration file already exists (from previous runs). The program outputs

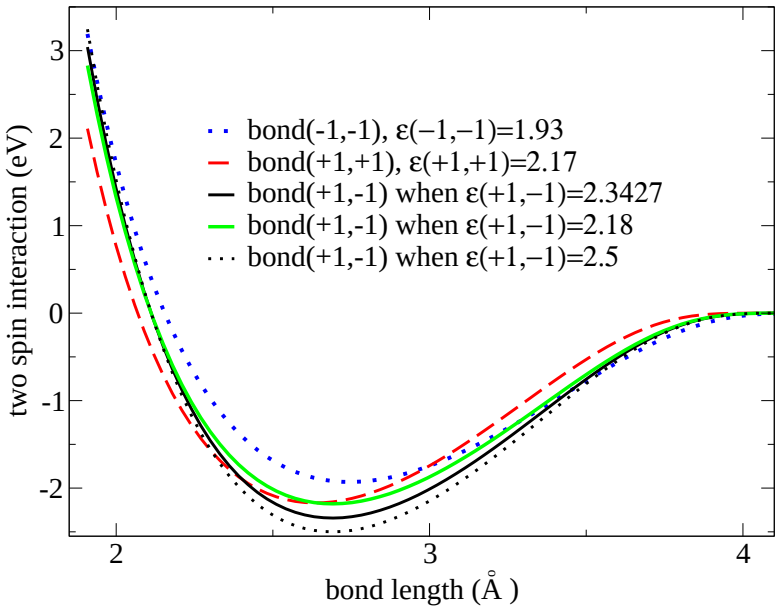


Figure 3.2: The energy curves of the mixing bond (+1,-1) under different values of $\epsilon(+1,-1)$. For comparison purpose, the energy curves of bond (+1,+1) and bond(-1,-1) are also shown. The notation "bond(+1,-1)" represent the bond between spin +1 and spin -1. Similar notations are used for other bond types.

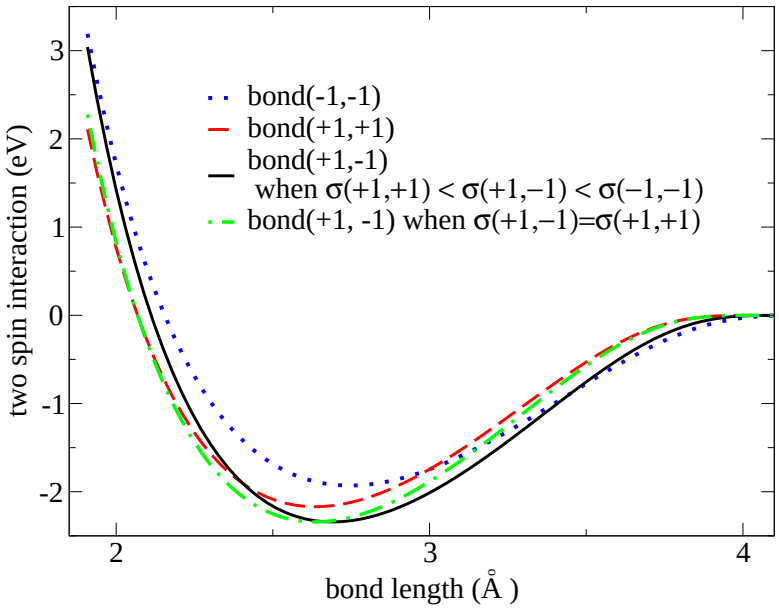


Figure 3.3: The energy curves of the mixing bond (+1,-1) under different ideal bond length $\sigma(+1,-1)$. The energy curves of bond (+1,+1) and bond(-1,-1) are also shown for comparison.

an instantaneous configuration every once a while. The output interval can be specified in the input file. Spins are numbered from 1 to $N = 8L^3$, where $L$ is the lattice size, or the number of unit cells in each of the x-, y- and z-directions. Spins in FCC sublattice 1 precedes any of those in FCC sublattice 2. In other words, spins in FCC sublattice 1 are numbered from 1 to $N/2$, and those in FCC sublattice 2 are numbered from $N/2+1$ to $N$. One of the advantages of this arrangement is that it simplifies the calculation of staggered magnetization. Since no two spins in the same SC sublattice are nearest neighbors, no dependency exists between their states. This makes it possible to update one SC sublattice after another, an ideal candidate for vectorization. Periodic boundary conditions are assumed, so there are no dangling bonds.

The MC simulation is performed as follows. For spin $S_j$ at position $\vec{r_j}$, we randomly generate a new spin $S'_j$ at a slightly altered random position $\vec{r_j}'$, and then use the Metropolis rejection method to accept or reject this attempt. $S'_j$ may or may not be the same as $S_j$. If $S'_j \neq S_j$, then $S'_j = -S_j$, and we call it a spin flip. After sweeping over the entire system, we allow volume fluctuation by attempting to rescale the system to slightly different linear sizes $L'_x$, $L'_y$, $L'_z$ from current ones: $x' = xL'_x/L_x$, $y' = yL'_y/L_y$, $z' = zL'_z/L_z$. The acceptance or rejection of this attempt is determined by Metropolis rejection method using the effective Hamiltonian $\mathcal{H}_{eff} = \mathcal{H} - Nk_BT\ln(L_xL_yL_z)$. Allowing volume fluctuation keeps the pressure constant. Due to spin flips, the number of spin +1's is not constant during simulations, although the total number of spins are constant. Such a system is called a semi-grand-canonical ensemble.

A Tausworthe (shift-register) generator [36] is used to generate random numbers, and the magic numbers are p=1279, q=1063. All floating point quantities are double-precision. The code is parallelized so that it runs on multiple processors with different random number sequences simultaneously. The multiple random number sequences diversify the data and improve the data quality used for histogram reweighting. The system sizes are up to $L = 24$, or $N = 110,592$, and all simulation runs were over $10^7$ MCS. For the diamond lattice, $L = 24$ translates to $L = 48$ for simple cubic lattice. Since this is an elastic Ising model, i.e., spin

positions are continuous variables, we cannot utilize the same ultrafast multispin coding algorithm as in Ref. [9], therefore we cannot handle very large systems such as $L = 96$ for the simple cubic Ising lattice.

As in standard MC studies, to estimate the errors in our results, we divided data into several equal-size blocks (between 5 to 10 blocks used), then calculated all quantities of interest for each block, finally calculated the mean and standard deviation of these quantities. Additional analysis were done using histogram reweighting and finite size scaling techniques.

## 3.4 More on Histogram Reweighting

We rewrite the Hamiltonian of the system as follows.

$$\mathcal{H} = -hM - h^+M^+ + W \tag{3.11}$$

where $W$ is the SW potential energy, $W = \mathcal{H}_2 + \mathcal{H}_3$. An MC simulation of length $n$ performed at temperature $T_0$, uniform magnetic field $h_0$, and staggered magnetic field $h_0^+$ generates $n$ configurations with a distribution frequency proportional to the Boltzmann weight, $\exp[-K_0\mathcal{H}]$, where $K_0 = 1/T_0$. To do the reweighting, we need to know the three-dimensional histogram distribution of ($M$, $M^+$, $W$). The probability distribution of the system is then

$$\begin{aligned} P_{h_0,h_0^+,K_0}(M,M^+,W) &= \frac{1}{Z(h_0,h_0^+,K_0)}\Omega(M,M^+,W) \\ & \quad exp[K_0(h_0M + h_0^+M^+ - W)] \end{aligned} \tag{3.12}$$

where $\Omega(M,M^+,W)$ is the number of configuration(density of states) with uniform magnetization $M$, staggered magnetization $M^+$, and SW potential $W$, and $Z(h_0,h_0^+,K_0)$ is the partition function of the system. Since we already performed $n$ MC steps and have $n$ configurations, we can calculate the histogram $H(M,M^+,W)$ for configuration $(M,M^+,W)$. The we have the probability distribution

$$P_{h_0,h_0^+,K_0}(M,M^+,W) = H(M,M^+,W)/n. \tag{3.13}$$

One problem associated with multi-dimensional reweighting is the huge storage required for the intermediate histogram data. For a $L = 20$ system, it would need 8 terabytes to store the histogram, which might be possible nowadays but certainly inefficient and undesirable. Fortunately, all we need is to find the average values for various properties, and we can obtain them without explicitly calculating the histogram at all. Through simple derivation, the expectation value of a quantity $A$ at a slightly different parameter set $(K, h, h^+)$ is given by

$$\begin{aligned} < A >_{K,h,h^+} &= \frac{1}{Z}\sum_j A(M_j, M_j^+, W_j) \\ &\quad \times \exp[(Kh - K_0h_0)M_j \\ &\qquad + (Kh^+ - K_0h_0^+)M_j^+ \\ &\qquad - (K - K_0)W_j] \end{aligned}$$

where

$$\begin{aligned} Z &= \sum_j \exp[(Kh - K_0h_0)M_j \\ &\qquad + (Kh^+ - K_0h_0^+)M_j^+ \\ &\qquad - (K - K_0)W_j] \end{aligned}$$

and $j$ runs over all the instantaneous configurations generated by the simulations. This form of histogram reweighting eliminates the storage needs of histogram files that can be huge in multi-dimensions. It also avoids dividing the continuous energy space into bins and losing precision due to numerical discretization. The histogram reweighting can, therefore, be done in one scan of the configuration files. If we fix $h = 0$ and $h^+ = 0$, and reweight over temperature, then the expectation value of a quantity $A$ at $K = 1/T$ is given by

$$< A >_K = \frac{1}{Z}\sum_j^N A(W_j)\exp[-(K - K_0)W_j],$$

where

$$Z = \sum_j^N \exp[-(K - K_0)W_j]$$

In histogram reweighting, it is necessary to check the histogram distribution. The reweighted mean internal energy should not be too faraway from the center (or the location where the histogram distribution reaches the maximum value $H_{\max}$). Otherwise, systematic errors will prevail. In practice, we require that the histogram value (normalized probability) $H$ at the reweighted mean internal energy satisfy

$$H(\text{reweighted mean internal energy}) \geq 0.22 H_{\max}$$

This guarantees that the reweighted mean internal energy is within two standard deviations from the center of the histogram.

Besides obtaining mean values, histogram reweighting can also provide a whole reweighted distribution at a different condition. Algorithm 1 reweights the magnetization distribution at a different temperature.

The distribution obtained above can be further rescaled to unit variance so it might be compared to the universal distribution of its universality class.

Now, one can adjust the temperature until the difference between the reweighted distribution and the universal one is minimal. This "magic" temperature is the critical temperature that we are looking for. This method is implemented in Algorithm 2.

One can also plot $H(M)\sigma_M/\delta M$ vs. $M/\sigma_M$ and compare it with the universal distribution. As a matter of fact, the mean field universal distribution is a Gaussian function whose unit-variance form is:

$$P(m) = \frac{1}{\sqrt{2\pi}} \exp^{-\frac{m^2}{2}},$$

and for the 3d Ising universality class, the universal distribution is given in Ref. [37]. Algorithm 3 is an easy-to-use version. It originates from the $32 \times 32 \times 32$ lattice in Ref. [37] and is included in the implementation of the class Fit_Universal_Histogram.

The simple Ising model has time reversal symmetry, and its phase diagram is symmetric about the temperature axis, as shown in Fig. 2.1. In more realistic models, such as the distortable antiferromagnetic diamond net in this dissertation, this symmetry is lost. Fig.

**Algorithm 1** Reweight the distribution of magnetization at a different temperature

1: //Notations:
2: //$W$: potential energy due to spin-spin interactions
3: //$M$: magnetization
4: //$\delta M$: the resolution of M, the size of a bin when discretizing magnetization
5: //$H(M)$: the histogram value at M
6: //$Z$: partition function
7: //$\beta$: $1/T$
8: //initialization:
9: $Z = 0$
10: **for** all indexM **do**
11: $H(\text{indexM}) = 0$
12: **end for**
13: //loop
14: **for** All MC data output $(M, E)$ in data files **do**
15: //do the reweighting
16: $e = \exp(-(\beta - \beta_0)W)$
17: indexM $= M/\delta M$
18: $H(\text{indexM}) = H(\text{indexM}) + e$
19: $Z = Z + e$
20: **end for**
21: //normalize the histogram
22: **for** all M from the minimum M to the maximum M **do**
23: $H(M) = H(M)/Z$
24: **end for**

**Algorithm 2** Scale a distribution to unit variance and compare with its universal distribution

```
1: //Univ: the function used to calculate the universal distribution
2: //find the standard deviation of M
3: avgM1= 0
4: avgM2= 0
5: //loop over all M
6: for all M from the minimum M to the maximum M do
7:    avgM1= M * H(M)
8:    avgM2= M^2 * H(M)
9: end for
10: σ_M = sqrt(avgM2 − avgM1 * avgM1)
11:
12: //calculate the difference weighted by probability
13: diff= 0
14: for all M from the minimum M to the maximum M do
15:    prob=Univ(M/σ_M) //calculate the universal value
16:    h = H(M)σ_M/δM //rescale the histogram to unit variance
17:    print M/σ_M, h //print out results
18:    diff=diff+prob*(h − prob)^2 //weighted sum
19: end for
```

**Algorithm 3** Calculate the 3d Ising distribution value

```
1: //parameters from PRE 62, 73 (2000) for L=32x32x32 3d Ising system
2: const int N32=32*32*32;
3: const double a=0.1553;
4: const double c=0.7776;
5: const double M0=0.18180;
6: const double M02=M0*M0;
7: const double ML=1.0965*0.3914688;
8: const double dev32 =5255.354508087;
9:
10: double Ising3d( double M)
11:
12: double tmp=M*dev32/N32;
13: double M2=tmp*tmp;
14: double root=M2/M02-1.0;
15: return ML*exp(-root*root*(a*M2/M02+c));
16:
```

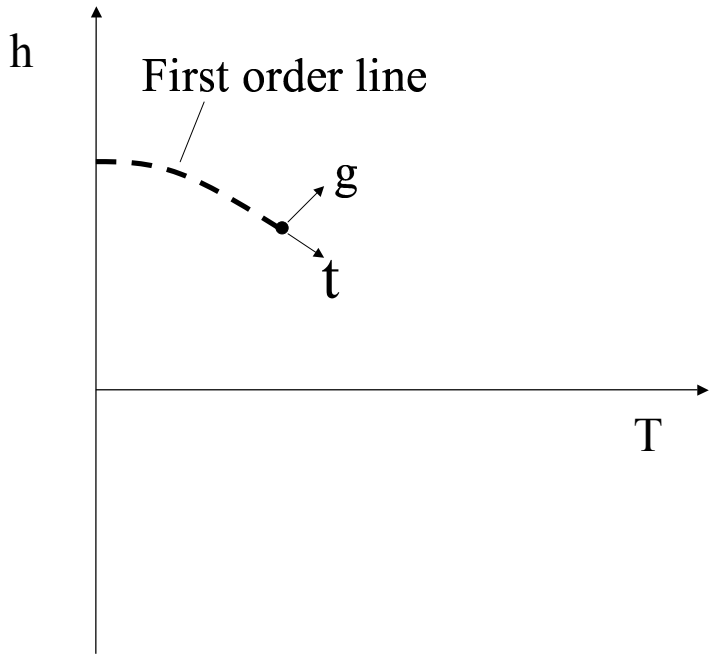


Figure 3.4: An illustrative example of a phase diagram where the boundary is not parallel to either of the two axes. This is not the phase diagram for the present diamond model. The phase boundary consists of a first order transition line ending with a critical point. The direction of one scaling field $t$ is along the tangent line at the end of the phase boundary. The direction of the other scaling field $g$ does not have to be perpendicular to $t$.

3.4 shows a phase diagram that forms a non-zero angle with the field axes. The critical point of the antiferromagnetic diamond model is described by two non-trivial parameter values, the critical coupling $K_c = 1/T_c$ and the critical magnetic field $h_c$. The scaling fields which are appropriate for describing the critical behavior of the system contain linear combinations of the deviations from these critical values:

$$t = K_c - K + s(h - h_c), \tag{3.14}$$

$$g = h - h_c + r(K_c - K), \tag{3.15}$$

where $r$ and $s$ depend upon the system [38]. For the simple Ising model, $r = s = 0$. The two quantities that are conjugate to these scaling fields are also linear combinations of the SW potential energy $W$ and magnetization $M$.

$$\mathcal{E} = \frac{W - rM}{1 - sr}, \tag{3.16}$$

$$\mathcal{M} = \frac{M - sW}{1 - rs}. \tag{3.17}$$

Then the deviations from the average values of these two quantities

$$\delta\mathcal{M} = \mathcal{M} - < \mathcal{M} >_c, \tag{3.18}$$

$$\delta\mathcal{E} = \mathcal{E} - < \mathcal{E} >_c \tag{3.19}$$

will follow the universal distributions of their universality classes, respectively.

If we only care about the distribution of $\mathcal{M}$ which is known for both mean-field and 3d Ising universality classes, we reduce the unknown parameters to a single $s$. The parameter $r$ will be absorbed in the normalization factor. Algorithm 4 is slightly different from Algorithm 1 and illustrates how to implement field-mixing for a new parameter set $(T, h)$ based on data taken at $(T_0, h_0)$.

## 3.5 VISUALIZATION

Visualization does provide some valuable insights that would otherwise take much more effort. For example, at high temperature, the staggered magnetization would oscillate around

**Algorithm 4** Field-Mixing

1: //Notations:
2: //$W$: potential energy due to spin-spin interactions
3: //$M$: magnetization
4: //$\delta M$: the resolution of M
5: //$H(M)$: the histogram value at M
6: //$Z$: partition function
7: //$\beta$: $1/T$
8: //$s$: the linear field mixing coefficient
9: //initialization:
10: $Z = 0$
11: **for** all indexM **do**
12: $H(\text{indexM}) = 0$
13: **end for**
14: //loop
15: **for** All MC data output $< M, E >$ in data files **do**
16: //do the reweighting
17: $e = \exp(-(\beta - \beta_0)W + (h - h_0)M)$
18: indexM $= (M - sW)/\delta M$
19: $H(\text{indexM}) = H(\text{indexM}) + e$
20: $Z = Z + e$
21: **end for**
22: //normalize
23: **for** all M from the minimum M to the maximum M **do**
24: $H(M) = H(M)/Z$
25: **end for**

zero, which is what we expect. We would not notice any difference even if the model breaks down. A 3d chaotic image reveals that the model already breaks down at such a high temperature, because the system topology is violated.

There are a number of tools for visualization. Aviz, developed by the group of J. Adler, is powerful and easy to use. It allows observation from continuous-varying viewpoints. The only drawback of Aviz is that it does NOT support topological information input, although it does draw bonds between physically nearest neighbors.

Another visualization kit is Povray. For a tutorial of Povray, visit

`http://www.physast.uga.edu/~smitchell/` .

Fig. 3.5, constructed with Povray, shows the 3d image of a $6 \times 6 \times 6$ system. The distortion is obvious. The drawing toolkit, including the Povray file, C program, and shell script are attached in Appendix C. Both Aviz and Povray have been used in the dissertation projects.

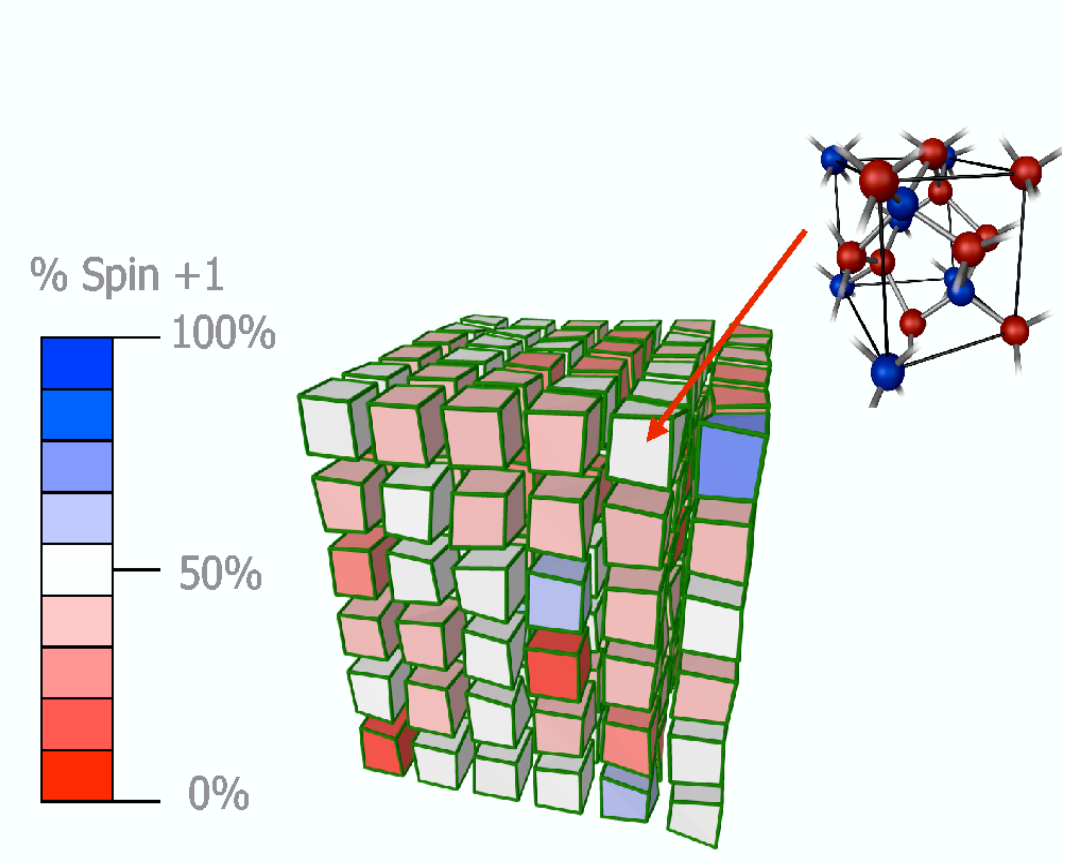


Figure 3.5: A 3d image of the model. Each box represents a unit cell, and each cell contains 8 spins (there are $1,728$ spins in the system). Gaps have been added between cells to show lattice distortion due to elasticity and thermal fluctuations. The colors represent the local concentration of spin +1 ( number of spin +1's in a unit cell divided by 8 ) in each cell, as indicated by the legend. The image at the upper right corner shows the interior structure of a unit cell.

# Chapter 4

# Results for the Diamond Antiferromagnet

## 4.1 Time Evolution and Correlation

Fig. 4.1 shows the time evolution before the system reaches equilibrium near critical temperature. It takes about 3000 MCS for the system to reach equilibrium. Note that the sudden change of the staggered magnetization indicates a lattice flip (or inversion, spins change directions simultaneously), which is quite normal for small lattices. The energy does NOT change much during lattice flip. After reaching equilibrium, the system evolves without any dramatic energy fluctuation, as in Fig. 4.2. The normalized autocorrelations in equilibrium state are shown in Fig. 4.3. The autocorrelation is calculated using

$$\phi_A(t) = \frac{< A(0)A(t) > - < A >^2}{< A^2 > - < A >^2},$$

where $A$ can be any quantity of interest, such as the internal energy and the staggered magnetization. If the time integral of $\phi_A(t)$ exists, i.e.

$$\tau_A \equiv \int_0^\infty \phi_A(t)dt,$$

and $\tau_A$ can be interpreted as the "relaxation time" of quantity $A$. In Fig. 4.3, the relaxation times are 55 MCS for the uniform magnetization, 136 MCS for the staggered magnetization, and 512 MCS for the internal energy.

## 4.2 Phase Diagram

The field dependence and temperature dependence of specific heat and staggered susceptibility are shown in Fig.4.4. These properties reach maxima at slightly different points. The

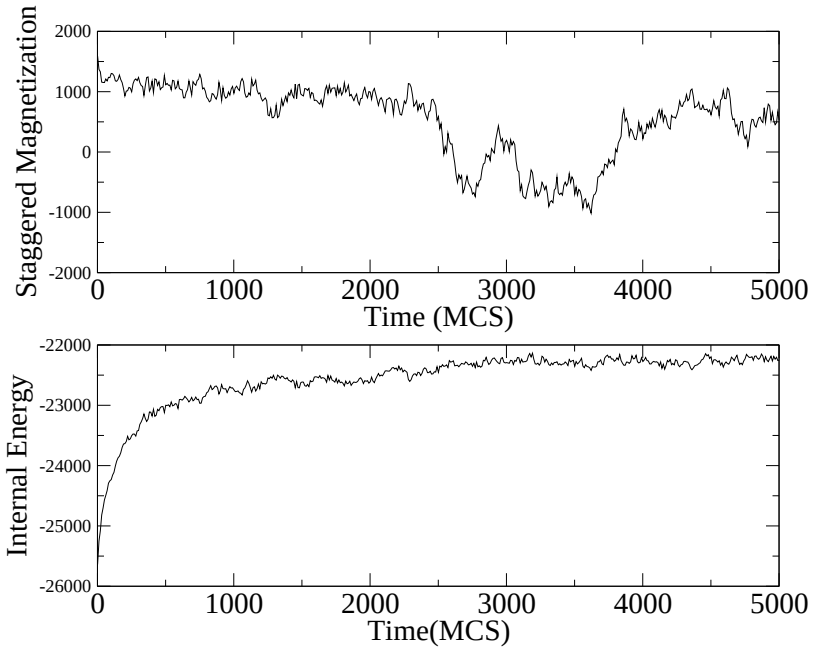


Figure 4.1: The time evolutions of the staggered magnetization and the internal energy before reaching equilibrium. The system starts from the totally ordered antiferromagnetic state. System size $L = 6$, $T = 0.312$, $h = 0$. The energy unit is eV.

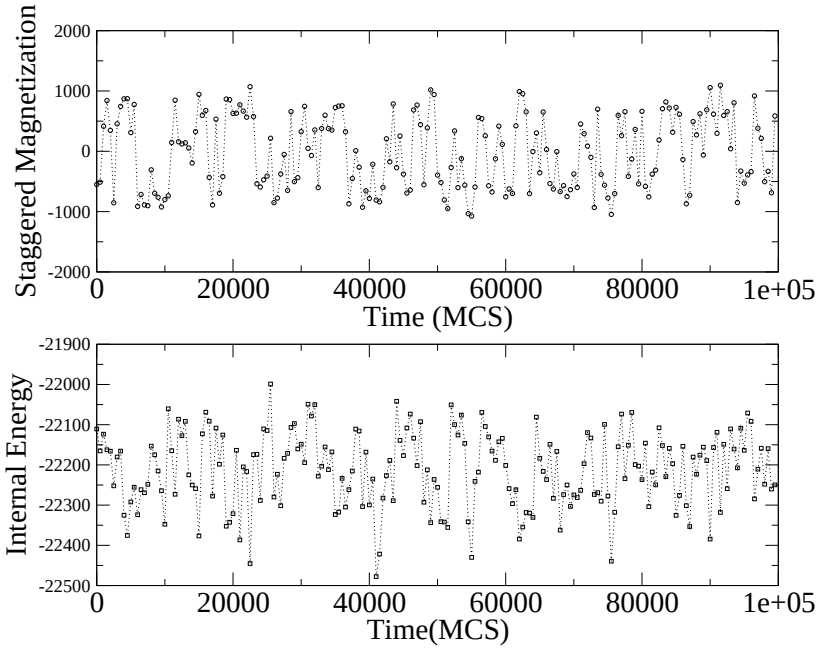


Figure 4.2: The time evolutions of the staggered magnetization and the internal energy in equilibrium. Data are taken every 500 MCS. System size $L = 6$, $T = 0.312$, $h = 0$. The energy unit is eV.

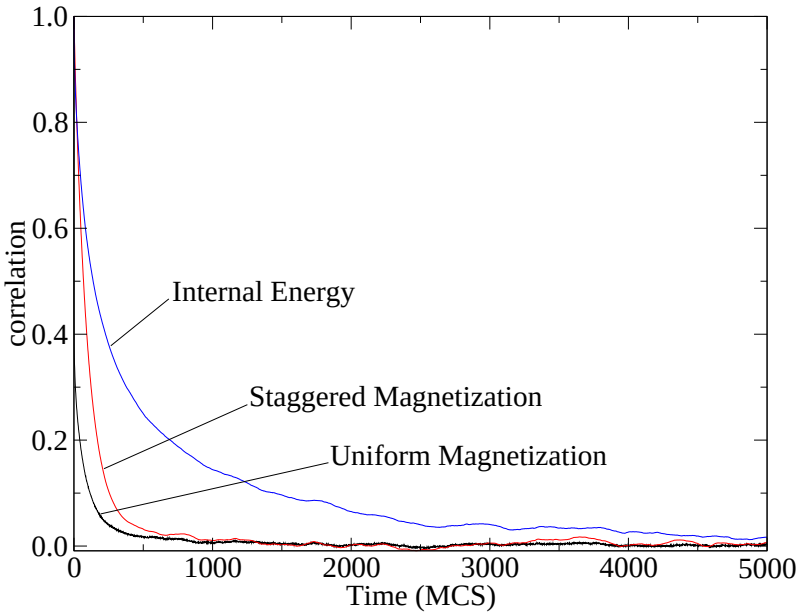


Figure 4.3: The time-displaced correlations of various quantities near criticality. System size $L = 6$, $T = 0.312$, $h = 0$.

specific heat exhibits large fluctuations, but the staggered susceptibility has a much smoother curve which makes it an ideal indicator for critical points.

We determine the phase boundary by locating the points where the staggered susceptibility reaches a maximum. We find a single phase boundary separating a disordered state from an ordered antiferromagnetic state as shown in Fig.4.5. The phase diagrams are rather symmetric because their mirror images (not shown) about their center lines collapse into themselves within error bars, respectively. The concentration is defined as the ratio of the total number of spin +1's in the system vs. the total number of all spins. Note that the temperature-concentration curve turns slightly inward at low temperature. We believe this low-temperature behavior is real, because it occurs consistently in different runs, and the difference (0.0039) to the concentration at T=0.1 exceeds the standard deviation (0.0014). However, we did not investigate this interesting phenomenon systematically.

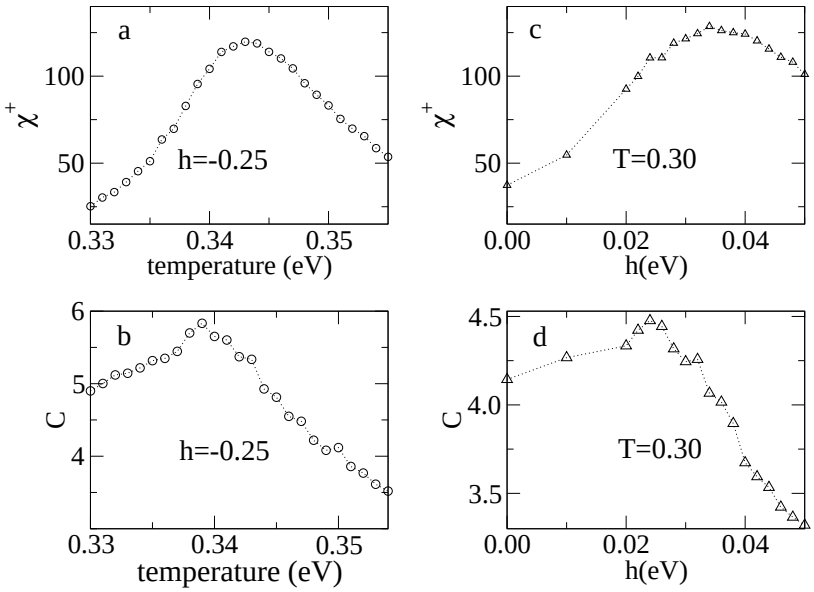


Figure 4.4: The field and temperature dependence of specific heat and susceptibility. The left two plots show the temperature dependence at fixed field. The right two, the field dependence at fixed temperature. The error bars are no larger than twice the symbol sizes.

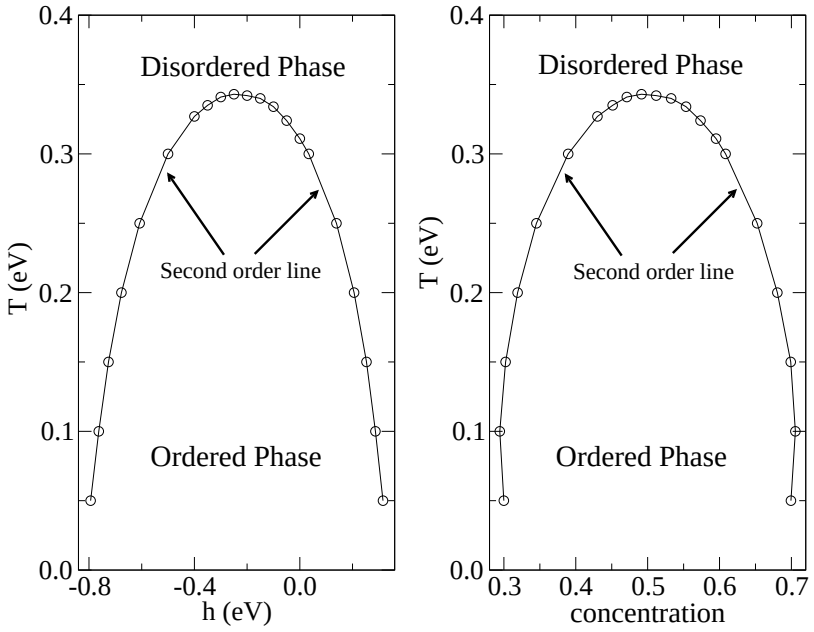


Figure 4.5: Fig.(a) shows the phase diagram in magnetic field - temperature space, and (b) in concentration-temperature space. The system size is 6x6x6. Each simulation length is $10^6$ MCS. The error bars are less than the sizes of data symbols.

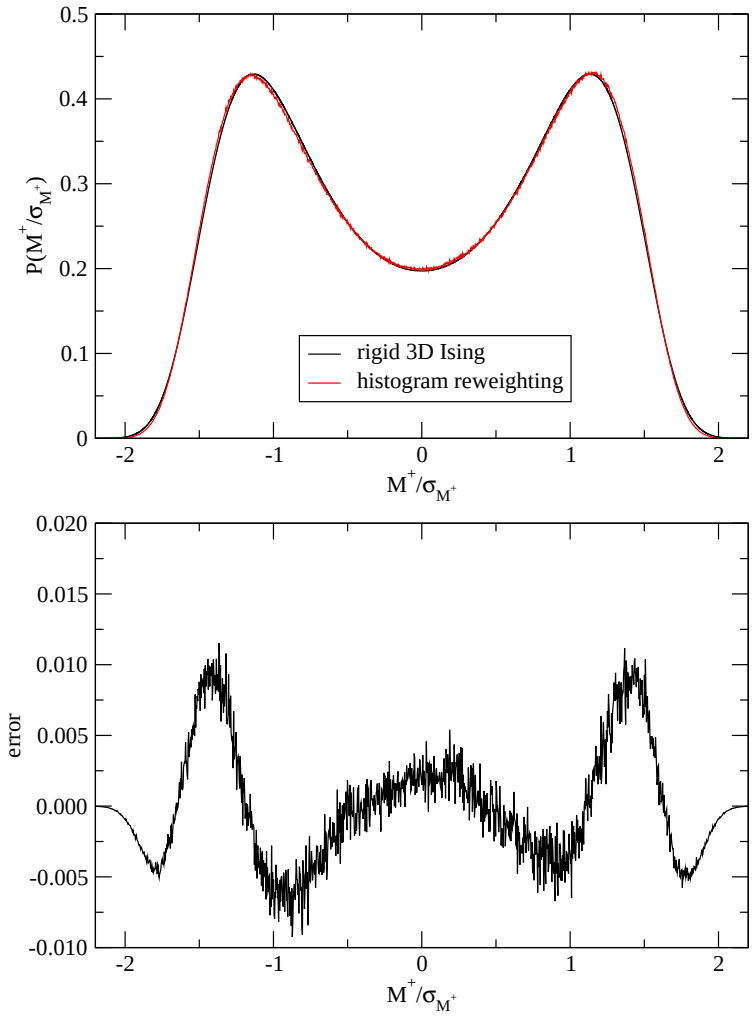


Figure 4.6: The order parameter (staggered magnetization) distribution for a $6\times6\times6$ system at $T = T_c$ and $h = 0$. $\sigma_{M^+}$ is the standard deviation of order parameter. The upper figure shows both the reweighted histogram and the rigid 3D Ising distribution. The lower figures shows the difference between the rigid 3d Ising distribution and the reweighted histogram. The error bars of the reweighted histogram are less than 0.013, or 3% of the maximum histogram value.

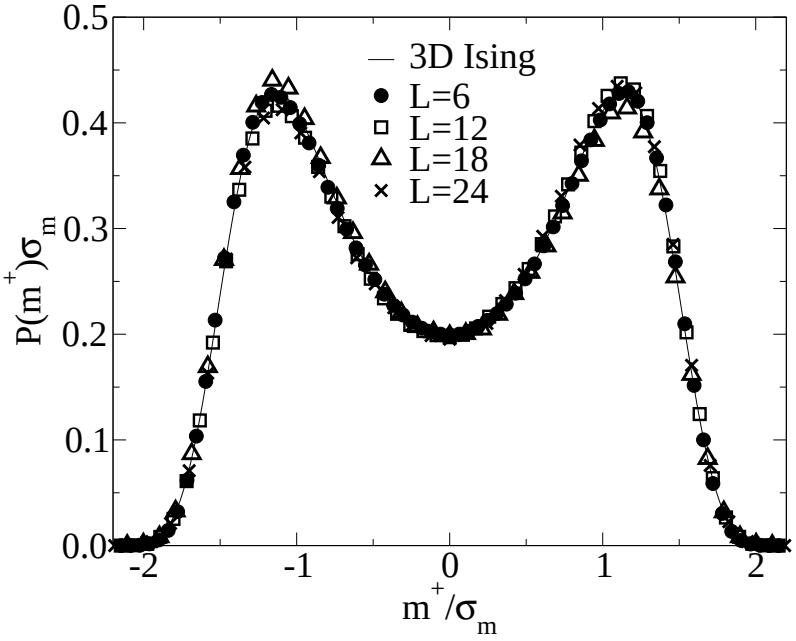


Figure 4.7: The order parameter distributions at the critical temperatures, obtained by fitting the histograms to the rigid 3d Ising universal distribution that is calculated from Ref. [10]. The distributions have been scaled to unit variance. The $\sigma_m$ is the standard deviation of the staggered magnetization $m^+$.

To understand the nature of the transition, we plot the normalized unit-variance probability distribution of the order parameter (staggered magnetization). Fig. 4.6 shows the distribution for the $6 \times 6 \times 6$ system and the comparison with that of the rigid 3D Ising model. As we can see, they agree very well. Fig.4.7 shows that the distributions for different system sizes collapse to the rigid 3d Ising distribution function. This is a strong indication that the phase transition belongs to the rigid 3D Ising universality class which is second order.

## 4.3 Critical Behavior

We extracted $\nu$ by considering the scaling behavior of certain thermodynamic derivatives, including the derivative of the cumulant $U_4$, and the logarithmic derivatives of $< |m^+| >$,

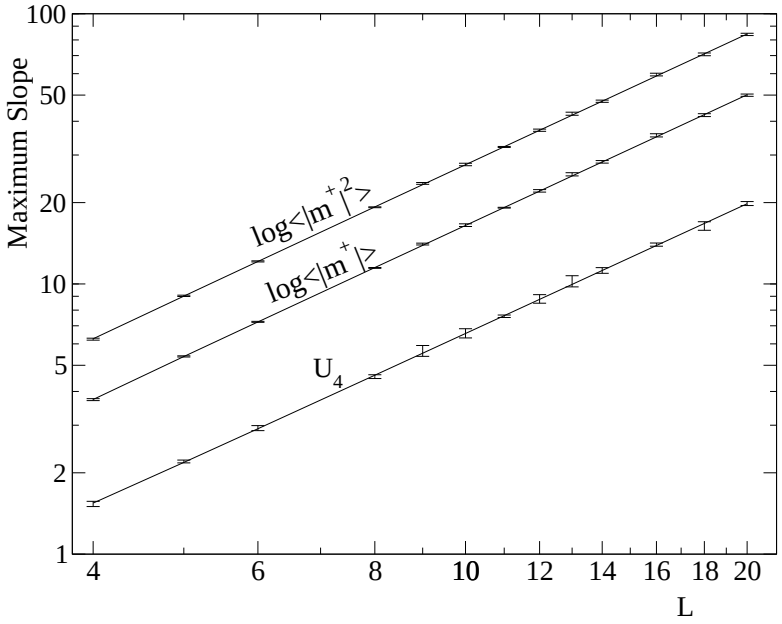


Figure 4.8: Log-log plot of the maximum slopes of various thermodynamic quantities used to determine $\nu$. The straight lines show the nonlinear least-square fit of Eq.2.27. All data points agree within one standard deviation.

$< |m^+|^2 >$, as in Ref. [9]. We plot these properties as a function of lattice size on a log-log scale in Fig.4.8.

The estimates for $1/\nu$ from the nonlinear least square fits are given in Table 4.1. Combining these three estimates we get $1/\nu = 1.60 \pm 0.01$. This agrees with the value $(1.594 \pm 0.004)$ reported in [9] within one standard deviation. Therefore, our estimate for $\nu$ is $0.625 \pm 0.004$. The size of the error bars comes primarily from the statistical errors in our simulation. With relatively small lattice sizes, we expect a noticeable correction term denoted by $\omega$. However, we find estimates for $\omega$ are extremely volatile, ranging from 0.6 to 4.5. This volatility also comes from the statistical errors in our data which submerges the correction terms.

Table 4.1: Estimates for $1/\nu$ obtained by finite size scaling of the maximum slopes of the cumulant and the logarithmic derivatives of $|m^+|^2$ and $|m^+|$.

| | $1/\nu$ |
|---|---|
| $U_4$ | $1.597 \pm 0.016$ |
| $\log < \|m^+\| >$ | $1.607 \pm 0.006$ |
| $\log < \|m^+\|^2 >$ | $1.603 \pm 0.015$ |

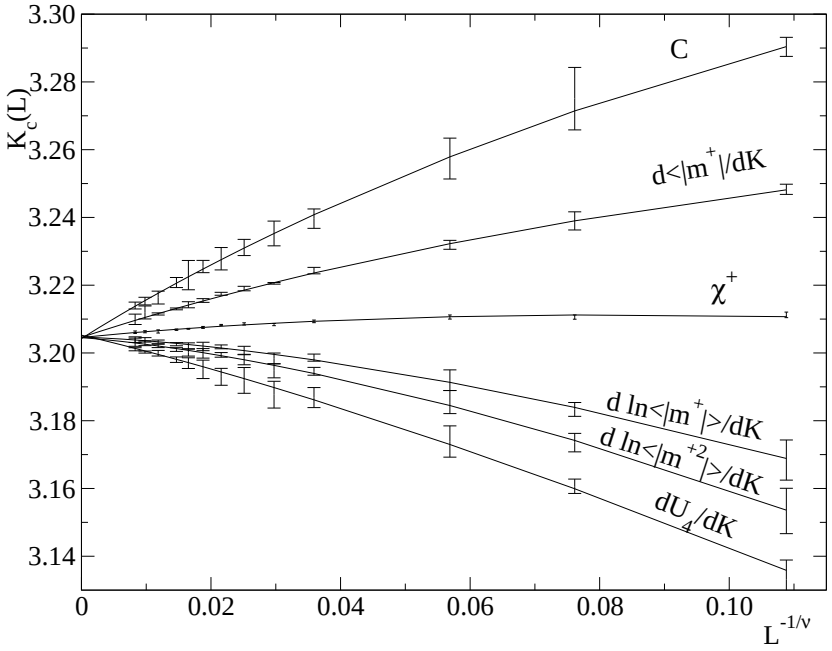


Figure 4.9: Size dependence of the finite-lattice critical temperature estimated from various properties. The solid lines are nonlinear least square fits to Eq.2.26.

Table 4.2: Estimates for $K_c$ obtained by finite size scaling of locations of the maximum slopes of various thermodynamic derivatives.

| | $K_c$ |
|---|---|
| $U_4$ | $3.204\ 50 \pm 0.000\ 64$ |
| $\log < \lvert m^+ \rvert^2 >$ | $3.204\ 11 \pm 0.000\ 36$ |
| $\log < \lvert m^+ \rvert >$ | $3.204\ 54 \pm 0.000\ 30$ |
| $\chi^+$ | $3.204\ 53 \pm 0.000\ 32$ |
| $\lvert m^+ \rvert$ | $3.204\ 52 \pm 0.000\ 50$ |

## 4.4 Determine $K_c$

We find that the elasticity has a strong effect on the critical transition temperature. In the absence of elasticity, the model becomes a rigid Ising model on a diamond lattice, whose transition temperature is known to be $k_B T_c^{diamond} = 2.70404|J|$.[39] With $|J| = |2\epsilon(+1,-1) - \epsilon(+1,+1) - \epsilon(-1,-1)|/4$, the transition temperature would be $k_B T_c = 0.14635eV$, less than half of the transition temperature found in our simulation. As in Ref. [9], we fitted the simulation data to Eq.2.26. We fixed $1/\nu = 1.60$, $\omega = 1.0$, and varied $K_c$, $\lambda$, and $b$ in the fitting. The choice $\omega = 1.0$ is not necessarily optimal, but it works very well. In fact, previous works[14] have suggested $\omega = 1.0$. The results are shown in Fig.4.9 and Table 4.2. Almost all data agree with fitted data within one standard deviation, and all agree within two standard deviations. The average of these values is $K_c = 3.20444 \pm 0.00019$. This corresponds to the critical temperature $k_B T_c = 0.312067 \pm 0.000018eV$. Our error bars are bigger than those reported in Ref. [9] due to the smaller lattice sizes.

## 4.5 $U_4$ Crossing

The Binder cumulant $U_4$ scales with the linear system size $L$ as Eq.2.25. At the critical temperature $T_c$, the $U_4(T)$ curves of all lattice sizes should have the same value $U_4^* = U_4(T_c)$, which would be a crossing point of all curves in Fig.4.10. The crossing value is one of the

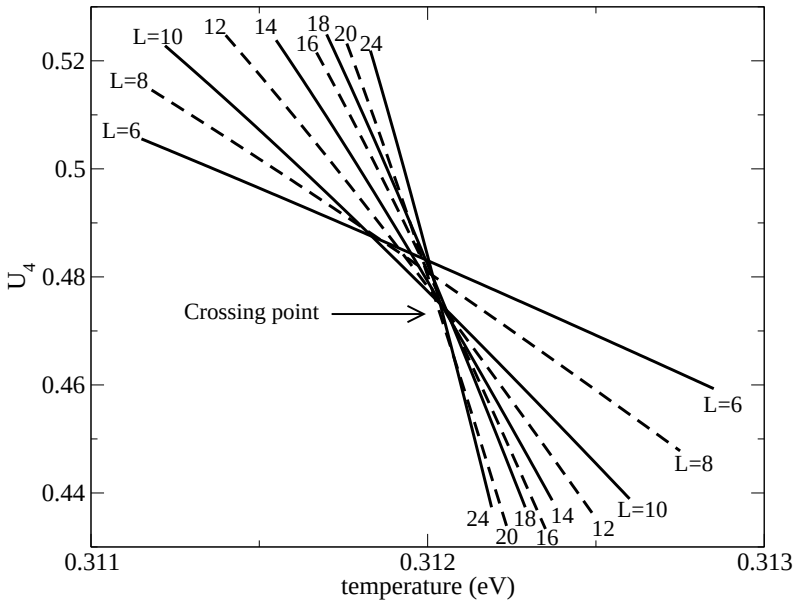


Figure 4.10: The Binder cumulant crossing. The curves alternate in solid and dashed lines for clarity. They are smooth because the data points are reweighted from histogram, and can reach any resolution. Lattice sizes are shown on both ends of each curve.

universal properties, which determines the university class of the model. Due to finite lattice size effect, the curves do not cross exactly at the same point, but have their crossing points spread out in a small neighborhood. By averaging the crossing points for $L \geq 10$, we find that this crossing value is $U_4^* = 0.472 \pm 0.002$. This is the same as that in the universality class of the rigid three-dimensional Ising model[9] $U_4^* \simeq 0.47$.

## 4.6 Other Exponents

We also fitted $m^+$ vs. lattice size $L$ to extract $\beta$. Fig. 4.11 shows the fitting results at a series of temperature values. At $T = 0.31208$, we get the best fitting result: $\beta/\nu = 0.5221 \pm 0.0036$, or $\beta = 0.3213 \pm 0.0092$. This agrees with the rigid 3D Ising exponent $\beta = 0.3270 \pm 0.0015$. The critical temperature is therefore $T_c = 0.31208$, agrees with the one obtained in sect. 4.4

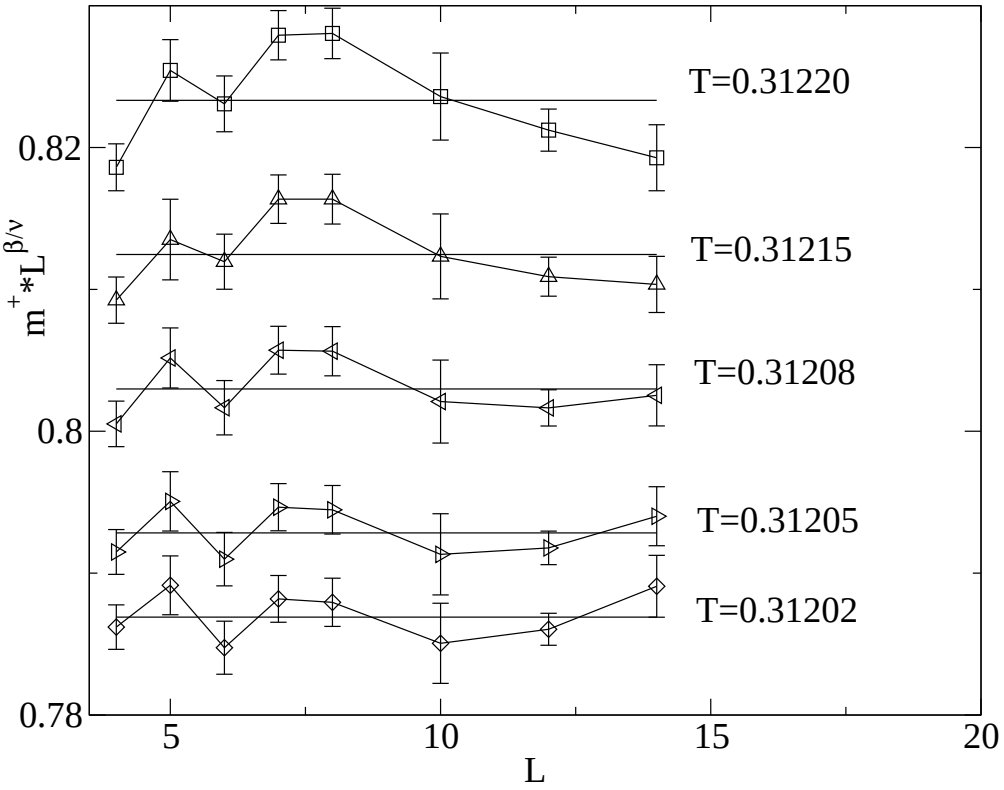


Figure 4.11: The nonlinear least square fitting of $\beta/\nu$ near critical temperature, according to Eq. 2.19. The horizontal solid lines are the fitted values of $\tilde{m}(x_t) = m^+ * L^{\beta/\nu}$. At $T = T_c$, $\tilde{m}(x_t) = \tilde{m}(0)$ becomes independent of lattice size $L$, therefore, remains constant for all lattice sizes.

within one standard deviation. The fitting results at different temperature values are shown in Table 4.3.

The exponent $\gamma/\nu$ is determined by the scaling behavior of the finite-lattice susceptibility defined in Eq.2.21. Fig. 4.12 shows that the fitting is rather rough. The estimate is $\gamma/\nu = 2.027 \pm 0.0045$ at $T = 0.31210$, and the estimate for $\gamma$ is $\gamma = 1.27 \pm 0.01$, which is also close to the $\epsilon$-expansion result $\gamma = 1.2390 \pm 0.0025$.

Table 4.3: Estimates for $\beta/\nu$ near critical temperature

| $K$ | $\beta/\nu$ |
|---|---|
| 0.31220 | $0.5396 \pm 0.0020$ |
| 0.31215 | $0.5354 \pm 0.0037$ |
| 0.31208 | $0.5221 \pm 0.0036$ |
| 0.31205 | $0.5088 \pm 0.0036$ |
| 0.31202 | $0.5008 \pm 0.0036$ |

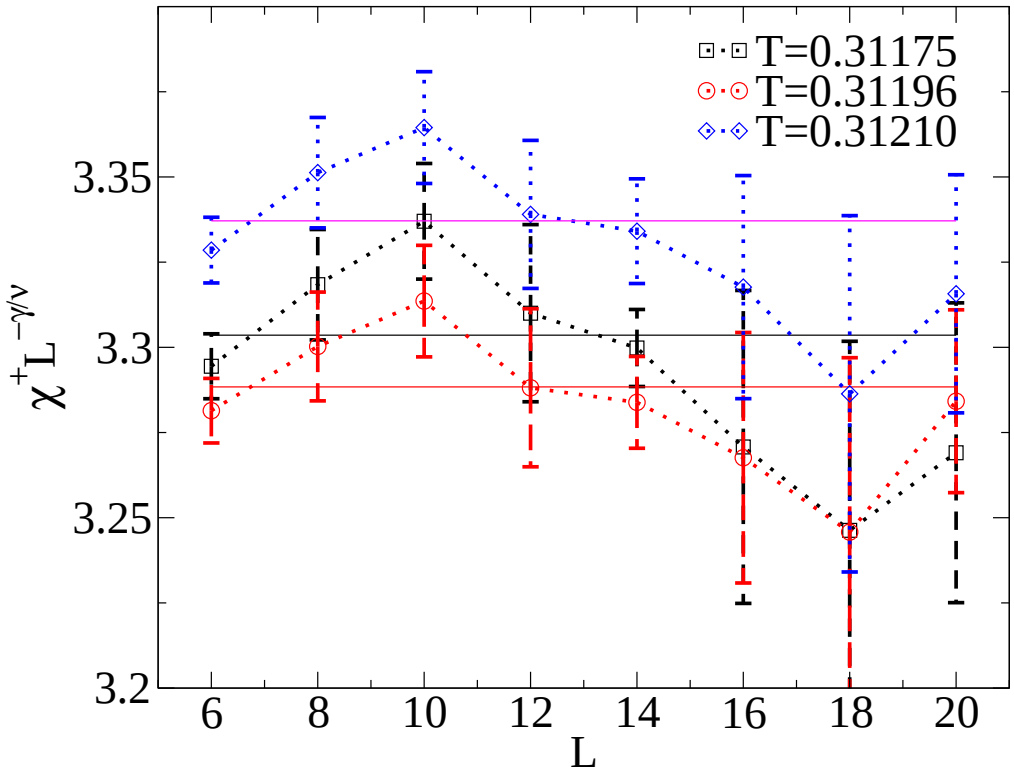


Figure 4.12: The nonlinear least square fitting of $\gamma/\nu$ near critical temperature, according to Eq. 2.21. The horizontal solid lines are the fitted values of $\tilde{\chi}(x_t) = \chi^+ * L^{-\gamma/\nu}$. At $T = T_c$, $\tilde{\chi}(x_t) = \tilde{\chi}(0)$ becomes independent of lattice size $L$, therefore, remains constant for all lattice sizes.

## 4.7 Summary

We have seen that the order parameter distribution is the same as that of the rigid 3D Ising model. So are all the critical exponents and the Binder cumulant crossing. We can conclude with confidence that the phase transition belongs to the rigid 3D Ising universality class, despite the added elasticity. This result disagrees with theoretical predictions [22].

# Chapter 5

## Elasticity in the Antiferromagnetic Diamond Model

We have seen that the critical transition temperature for the antiferromagnetic diamond model is quite different from that of the rigid model, but the phase transition still belongs to the universality class of rigid Ising model. Is this because the model is too rigid? To answer this question, we will assess the elasticity in this model. However, this is a rather vague issue. The theory doesn't tell us how much elasticity is sufficient to see the deviation from Ising behavior, neither does it point out how to measure the elasticity. We will examine the elasticity in five approaches: the bond length distribution, the energy distribution, the coefficient distributions of prefactors (defined later), the coefficient distributions of the Ising-like Hamiltonian, and the field mixing effect.

### 5.1 Bond Length Distribution

As shown in Fig.5.1, the nearest-neighbor bond length distributions are quite broad, with the half-height-width being about 20% of the mean value, which means our model is indeed very fluffy. Note that not all maxima occur at the same bond length value, because different bonds have different equilibrium lengths . Fig.5.1 also shows the uniformity of elasticity in the system, because the bond length distribution of all sites and that of a single site agree very well and almost overlap with each other.

### 5.2 Energy Distribution

Bond length variation leads to energy changes. In this section, we will separate the total energy into two parts: the first part is independent of the nearest-neighbor bond lengths,

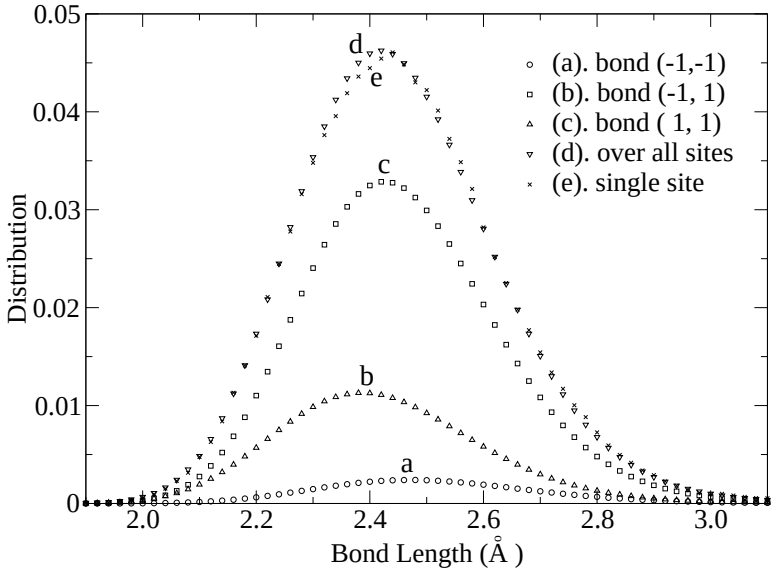


Figure 5.1: The first three bond length distribution (a), (b) and (c) are normalized together. That is, their covered areas reflect their relative concentrations. Plot (d) is the distribution of all bonds, which is the sum of (a), (b) and (c). Plot (e) shows the bond length distribution of a single site over time.

and is called the chemical energy; the second part depends on the nearest-neighbor bond lengths, therefore, is related to the elasticity, and is called the elastic energy. The two-body SW potential can be Taylor-expanded in terms of bond length $r_{ij}$'s as

$$\mathcal{H}_2 = -\sum_{i<j} \epsilon(S_i, S_j) + \sum_{i<j} O(r_{ij}).$$

The first term is independent of bond lengths and is the chemical energy.

$$E_{chem} = -\sum_{i<j} \epsilon(S_i, S_j). \tag{5.1}$$

The second part consists of higher order terms of $\mathcal{H}_2$ that depend on bond lengths. The elastic energy, $E_{Elastic}$, consisting of $\mathcal{H}_3$ and the high-order terms of $\mathcal{H}_2$, depends on the bond lengths and angles. Even without elasticity, i.e., in a rigid Ising model, the chemical energy fluctuates as spins flip. In Fig.5.2 we show the distributions of the chemical energy and the

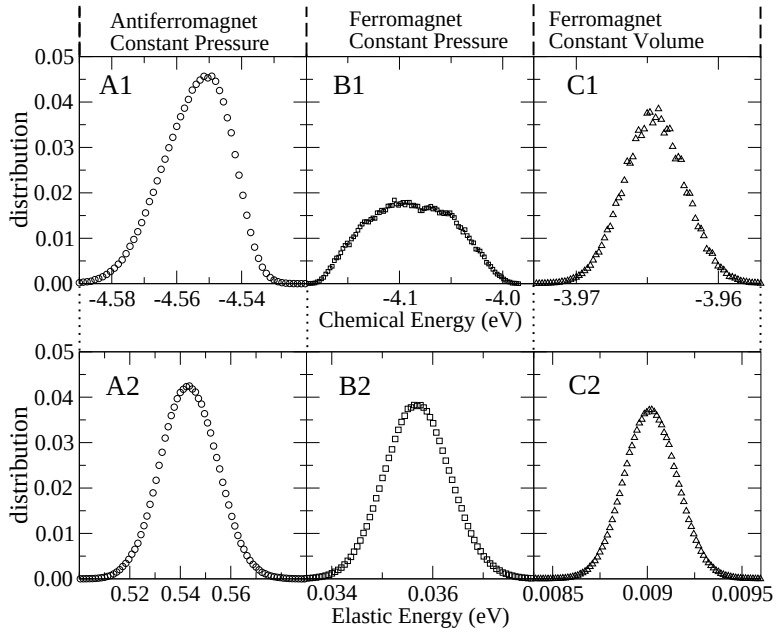


Figure 5.2: The chemical energy and elastic energy distributions near critical temperatures in various models: A1,A2) Antiferromagnet at constant pressure.B1,B2) Ferromagnet at constant pressure.C1,C2) Ferromagnet at constant volume.

elastic energy in antiferromagnetic and ferromagnetic models. The two ferromagnetic models are identical to those reported in Ref. [14] and Ref. [23], and they are used for comparison purpose here. We see that the distributions of elastic energy are symmetric, while those of chemical energy are asymmetric. In the antiferromagnet, the half-height-width of elastic energy distribution ($0.027eV$) is slightly larger than that of chemical energy ($0.025eV$). In the ferromagnets, the elastic energy distributions are far narrower than their chemical energy counterparts. From this point of view, there is much more elasticity in the antiferromagnetic model than in the ferromagnetic models.

## 5.3 LGW Expansion Coefficient Distributions

While the above two measurements indicate there is sufficient elasticity in the model, they mingle the translational and pseudospin degrees of freedom. Dünweg [40] suggested a clean way as follows to separate the two degrees of freedom. First, we expand the two-body interaction as below.

$$\mathcal{H}_2(S_i, S_j, r_{ij}) = A(r_{ij})S_iS_j + B(r_{ij})S_i + C(r_{ij})S_j + D(r_{ij}) \tag{5.2}$$

There are four combinations of $(S_i, S_j)$ for a $r_{ij}$, which gives the four equations in equation array 5.3.

$$\begin{pmatrix} \mathcal{H}_2(+1,+1,r_{ij}) \\ \mathcal{H}_2(+1,-1,r_{ij}) \\ \mathcal{H}_2(-1,+1,r_{ij}) \\ \mathcal{H}_2(-1,-1,r_{ij}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} A(r_{ij}) \\ B(r_{ij}) \\ C(r_{ij}) \\ D(r_{ij}) \end{pmatrix} \tag{5.3}$$

Then calculate the 4 coefficients $A$, $B$, $C$, and $D$ as functions of $r_{ij}$ in terms of $\mathcal{H}_2(1,1,r_{ij})$, $\mathcal{H}_2(1,-1,r_{ij})$, $\mathcal{H}_2(-1,1,r_{ij})$, and $\mathcal{H}_2(-1,-1,r_{ij})$. The solution is straightforward – the transpose of the coefficient matrix.

$$\begin{pmatrix} A(r_{ij}) \\ B(r_{ij}) \\ C(r_{ij}) \\ D(r_{ij}) \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \mathcal{H}_2(+1,+1,r_{ij}) \\ \mathcal{H}_2(+1,-1,r_{ij}) \\ \mathcal{H}_2(-1,+1,r_{ij}) \\ \mathcal{H}_2(-1,-1,r_{ij}) \end{pmatrix} \tag{5.4}$$

Similarly, the three-body interaction is expanded as

$$\begin{aligned} \mathcal{H}_3(S_i, S_j, S_k, r_{ij}, r_{jk}, \theta_{ijk}) &= ES_iS_jS_k + FS_iS_j + GS_jS_k \\ &\quad + PS_kS_i + QS_i + RS_j \\ &\quad + SS_k + T \end{aligned} \tag{5.5}$$

where $E$, $F$, $G$, $P$, $Q$, $R$, $S$, and $T$ are coefficients dependent on $r_{ij}$, $r_{jk}$ and $\theta_{ijk}$. These eight coefficients can also be calculated in terms of $\mathcal{H}_3$'s at the eight combinations of $S_i$, $S_j$, and $S_k$. The equation array is

$$\begin{pmatrix} \mathcal{H}_3(+1,+1,+1,r_{ij},r_{jk},\theta_{ijk}) \\ \mathcal{H}_3(+1,+1,-1,r_{ij},r_{jk},\theta_{ijk}) \\ \mathcal{H}_3(+1,-1,+1,r_{ij},r_{jk},\theta_{ijk}) \\ \mathcal{H}_3(+1,-1,-1,r_{ij},r_{jk},\theta_{ijk}) \\ \mathcal{H}_3(-1,+1,+1,r_{ij},r_{jk},\theta_{ijk}) \\ \mathcal{H}_3(-1,+1,-1,r_{ij},r_{jk},\theta_{ijk}) \\ \mathcal{H}_3(-1,-1,+1,r_{ij},r_{jk},\theta_{ijk}) \\ \mathcal{H}_3(-1,-1,-1,r_{ij},r_{jk},\theta_{ijk}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} E(r_{ij},r_{jk},\theta_{ijk}) \\ F(r_{ij},r_{jk},\theta_{ijk}) \\ G(r_{ij},r_{jk},\theta_{ijk}) \\ P(r_{ij},r_{jk},\theta_{ijk}) \\ Q(r_{ij},r_{jk},\theta_{ijk}) \\ R(r_{ij},r_{jk},\theta_{ijk}) \\ S(r_{ij},r_{jk},\theta_{ijk}) \\ T(r_{ij},r_{jk},\theta_{ijk}) \end{pmatrix} \tag{5.6}$$

And the solution is

$$\begin{pmatrix} E(r_{ij}, r_{jk}, \theta_{ijk}) \\ F(r_{ij}, r_{jk}, \theta_{ijk}) \\ G(r_{ij}, r_{jk}, \theta_{ijk}) \\ P(r_{ij}, r_{jk}, \theta_{ijk}) \\ Q(r_{ij}, r_{jk}, \theta_{ijk}) \\ R(r_{ij}, r_{jk}, \theta_{ijk}) \\ S(r_{ij}, r_{jk}, \theta_{ijk}) \\ T(r_{ij}, r_{jk}, \theta_{ijk}) \end{pmatrix} =$$

$$\frac{1}{8} \begin{pmatrix} 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \mathcal{H}_3(+1, +1, +1, r_{ij}, r_{jk}, \theta_{ijk}) \\ \mathcal{H}_3(+1, +1, -1, r_{ij}, r_{jk}, \theta_{ijk}) \\ \mathcal{H}_3(+1, -1, +1, r_{ij}, r_{jk}, \theta_{ijk}) \\ \mathcal{H}_3(+1, -1, -1, r_{ij}, r_{jk}, \theta_{ijk}) \\ \mathcal{H}_3(-1, +1, +1, r_{ij}, r_{jk}, \theta_{ijk}) \\ \mathcal{H}_3(-1, +1, -1, r_{ij}, r_{jk}, \theta_{ijk}) \\ \mathcal{H}_3(-1, -1, +1, r_{ij}, r_{jk}, \theta_{ijk}) \\ \mathcal{H}_3(-1, -1, -1, r_{ij}, r_{jk}, \theta_{ijk}) \end{pmatrix} \tag{5.7}$$

Now we define the **total prefactor** as the sum of the coefficients of all the nearest-neighbor term $S_i S_j$'s. As suggested by B. Dünweg (but I am not convinced), the width of the total prefactor distribution indicates the elasticity of the model. Fig.5.3 shows the total prefactor distributions in the ferromagnetic and antiferromagnetic cases. The half-height distribution width of the ferromagnet at constant pressure is about 11% of the mean value, while that of the ferromagnet at constant volume is only 0.1% of its mean value. The half-height width of the antiferromagnet at constant pressure is 0.5% of its mean value. Since an apparent deviation from the rigid Ising behavior is observed in the ferromagnet at constant volume [23], it seems that even 0.1% of variation is good enough to incur non-Ising behavior.

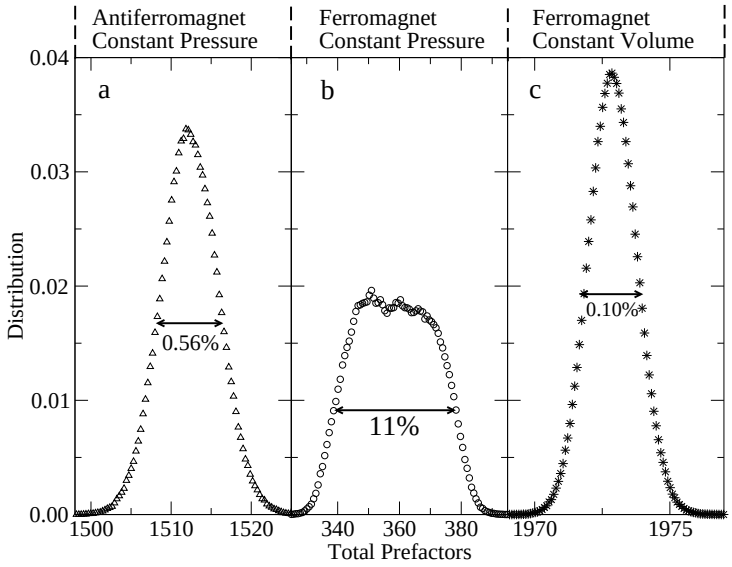


Figure 5.3: The prefactor distributions near critical temperatures in various models: A)Antiferromagnet at constant pressure.B)Ferromagnet at constant pressure.C)Ferromagnet at constant volume.

From this measurement, we cannot say that our antiferromagnetic model is not sufficiently elastic.

## 5.4 Ising-like Hamiltonian Equivalence

The physical meaning of the so-called **total prefactor** is still unclear since it is just a sum of coefficients. It never appears in the Hamiltonian. Instead, we rewrite the Hamiltonian in an Ising-like fashion

$$\mathcal{H} = -J_0 - \sum_j J_1(j)S_j - \sum_{ij} J_2(i,j)S_iS_j - \sum_{ijk} J_3(i,j,k)S_iS_jS_k, \tag{5.8}$$

where $J_0$ is a constant term, $J_1(j)$ is the equivalent coefficient of single-spin term $S_j$, $J_2(i,j)$ is the equivalent coefficient of two-spin term $S_iS_j$, and $J_3(i,j,k)$ is the equivalent coefficient of three-spin term $S_iS_jS_k$. Using Eq. 5.2 and Eq. 5.5, we can see the relationship between

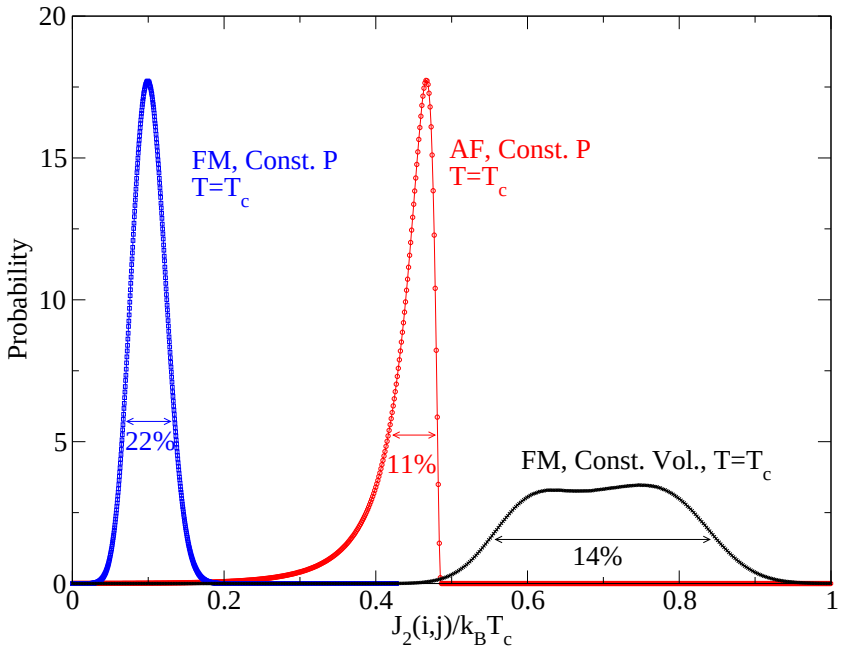


Figure 5.4: Comparing $J_2$ distributions in three systems. The $J_2(i,j)$'s have been normalized by their critical temperatures, respectively.

these $J$'s and those decomposing coefficients. For $J_0$, we have

$$J_0 = -\sum_{\mathrm{NN}<ij>} D(r_{ij}) - \sum_{<ijk>} T(r_{ji}, r_{jk}, \theta_{ijk})$$

And $J_3(i,j,k) = E(r_{ji}, r_{jk}, \theta_{ijk})$. For other $J$'s, the relationship is not so explicit. $J_1(j)$ is the sum of $C_j$ and the twelve $R_j$'s that involves spin $S_j$. $J_2(i,j)$ is the sum of $A(r_{ij})$ and the six $F_{ij}$'s that involves the two-spin term $S_iS_j$. Nonetheless, the calculations of $J$'s are easy to implement in the code. Fig. 5.4 shows the distributions of $J_2$'s in different systems, because we expect $J_2$ to determine the order of spins (ferromagnet or antiferromagnet). Again, we use the relative width, i.e. the ratio of the standard deviation to the mean value, to indicate the elasticity. We can see the $J_2$ distribution for the antiferromagnet is the narrowest among the three. However, $J_2$ turns out to be positive for all three cases. And it should be negative for the antiferromagnet. A possible explanation is that the $J_2$ is insignificant comparing to $J_1$ or

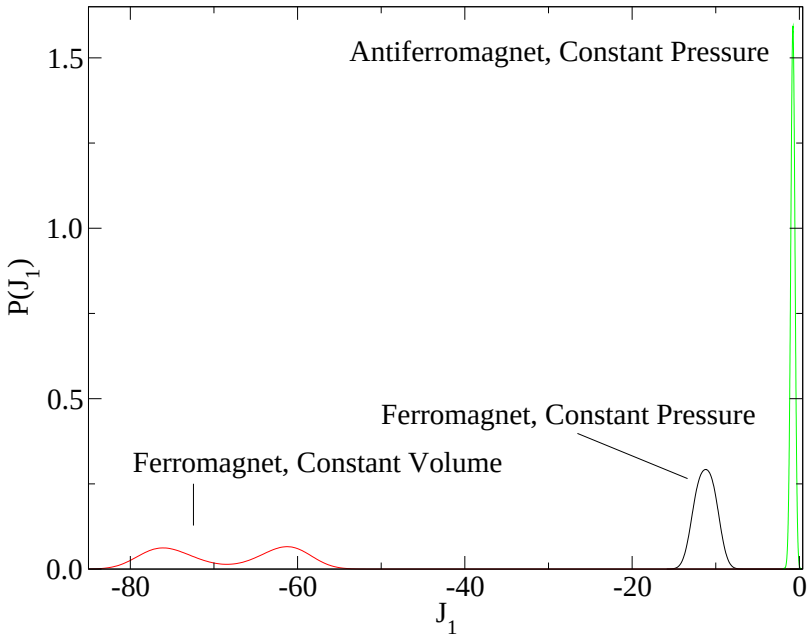


Figure 5.5: Comparing $J_1$ distributions in three systems. The $J_1$'s have been normalized by their critical temperatures.

$J_0$. Maybe $J_1$ should be a staggered field, which means that it should have a distribution with symmetric double peaks around zero. Fig. 5.5 shows the $J_1$ distributions. The $J_1$ distribution of the antiferromagnet is narrowest, has the least absolute values, and is negative. Still, it does NOT explain why the system is antiferromagnetic. We come to a point where we really cannot assess the elasticity this way. For this reason, we use an elasticity-adjustable model in next chapter.

## 5.5 Field Mixing Effect

We also checked the field mixing effect in these models. We find no field mixing effect in these transitions in the antiferromagnetic model, because the order parameter distribution fits to universal 3d Ising distribution without any field-mixing calculation. However, there

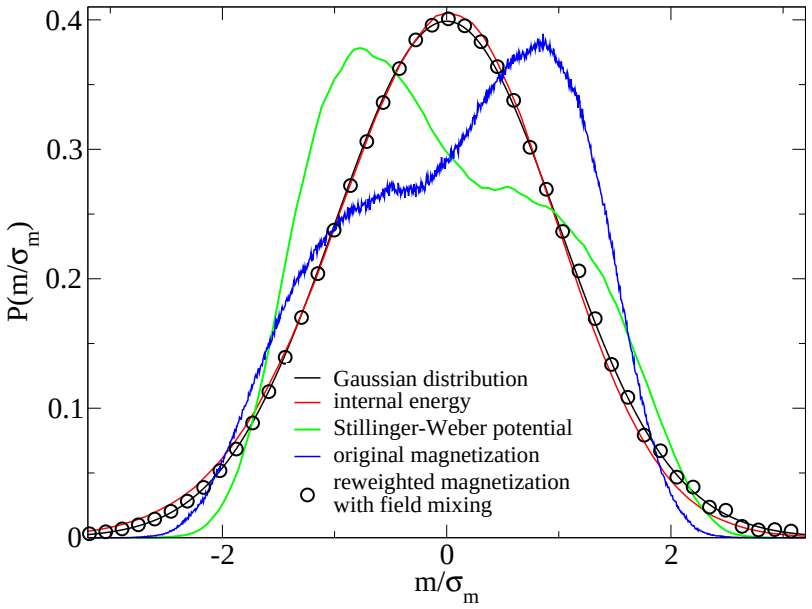


Figure 5.6: The order parameter distribution of the ferromagnet at constant pressure. The original data is taken at $h = -0.239755$, and $T = 0.0213$eV, near the critical temperature. The lattice size is $L = 12$, and data amount $8.4 \times 10^6$ MCS. The reweighted critical point is ($T_c = 0.02161$eV, $h = -0.239777$eV). The field mixing coefficient is $s = 4.17$.

is a strong field-mixing effect in the ferromagnetic model at constant pressure, as shown in Fig. 5.6. In fact, the magnetization $m$ and the SW potential $W$ distributions are complementary to each other. Their linear combination, $m - sW$, is the order parameter. Note that the internal energy $E = -hm + W = -h(m - \frac{1}{h}W)$. So the new field-mixing parameter happened to be the internal energy. This is reasonable because the internal energy follows the Gaussian distribution for the mean field universality class. The field-mixing coefficient, therefore, should be $s = \frac{1}{h}$. This is verified by the result $s = 4.17 = 1/0.239777 = 1/h$.

## 5.6 Summary

These elasticity analyses give contradicting and even unphysical results, which shows the difficulty of separating the elasticity and pseudospin coupling in the SW potential. It would be much easier if we can adjust the elasticity by explicitly modifying a single parameter.

# Chapter 6

# The Stacked Triangular Lattice

As pointed out in the previous chapter, it is very difficult to separate the elasticity and pseudospin couplings in the SW potential in the antiferromagnetic diamond model. Therefore, we now turn our attention to an elasticity-tunable Ising model on a deformable stacked triangular lattice. Another reason that drives us to investigate this model is that the results of the Boubcheur group [24] on this model are quite unusual. Fig. 6.1 shows the time evolution of the model they reported. They find a very strong autocorrelation around 500,000 Monte Carlo steps, while the autocorrelation at the beginning appears to be weak, which is somewhat unphysical in our opinion.

## 6.1 Model

The system consists of a stacked triangular lattice. Let the $z$-direction be the direction of stacking, then the $xy$ plane consists of equilateral triangles. The XY planes (layers) directly stack over adjacent layers without any horizontal displacement. The Hamiltonian is described by

$$\mathcal{H} = U_0 \sum_{<i,j>} J(r_{ij}) + U_m \sum_{<i,j>} J(r_{ij})\sigma_i\sigma_j, \tag{6.1}$$

where the first and second terms are the cohesive and magnetic interactions, respectively. Both interactions are given by the Lennard-Jones potential

$$J(r_{ij}) = (r_0/r_{ij})^{12} - 2(r_0/r_{ij})^6, \tag{6.2}$$

where $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ is the distance between spins at the $i$th and $j$th sites, $r_0$ is the equilibrium distance between the nearest-neighbor(NN) spins. The ratio $Q = U_0/U_m$ measures the

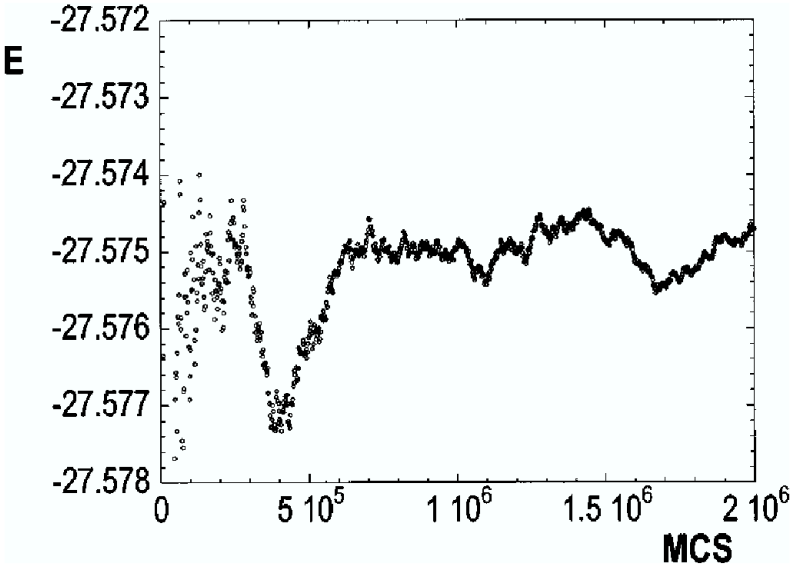


Figure 6.1: The time evolution of the stacked triangular Ising model reported by Boubcheur and Diep. Lattice size $L = 20$, $T = 5.128$, $Q = 8$.

rigidity of the model in a very direct way. By increasing ( or decreasing ) $Q$, we can decrease ( or increase ) the elasticity of the model. When $Q$ goes to infinity, the model becomes completely rigid.

It should be pointed out that such a system is unstable since the system would prefer FCC structure when there are only two-body interactions. When the rigidity ratio $Q$ is too small, the whole system might become invalid.

We take $U_m = 1$ , which makes the system ferromagnetic because $J(r_{ij})$ is negative around equilibrium. The simulation is done in a semi-grand-canonical ensemble at constant volume. According to Boubcheur [24], for $Q > 4$, the critical behavior is rigid 3d Ising. This agrees with our simulational results. For $Q = 3$, Boubcheur reported 3d $XY$ behavior, our

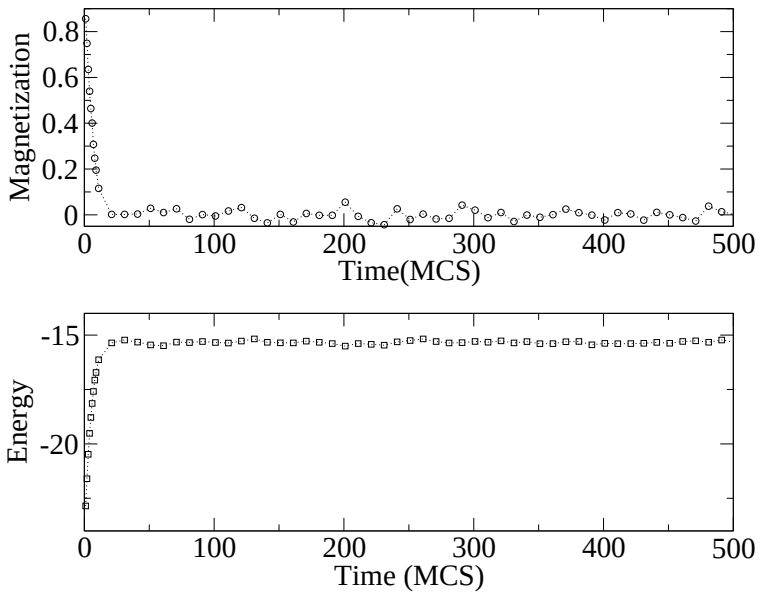


Figure 6.2: The time evolution of the first 500 MCS. The dotted line is for visualization only. Data obtained at $L = 20$, $Q = 8$, $T = 5.128$.

simulational results shows that it is still rigid 3d Ising. We will go through the results quickly since the data analysis techniques have been introduced in previous chapters.

## 6.2 Time Evolution

We attempted to repeat the simulation in Ref. [24] by setting $Q = 8$, $T = 5.128$ and the system size $L = 20$. The time evolution is shown in Fig. 6.2 and Fig. 6.3.

We can see that the system quickly (takes less than 100 MCS) approaches equilibrium at the above condition, and there is no strong correlation after reaching equilibrium. This is already above the critical temperature for $Q = 8$, and the magnetization only fluctuates around zero.

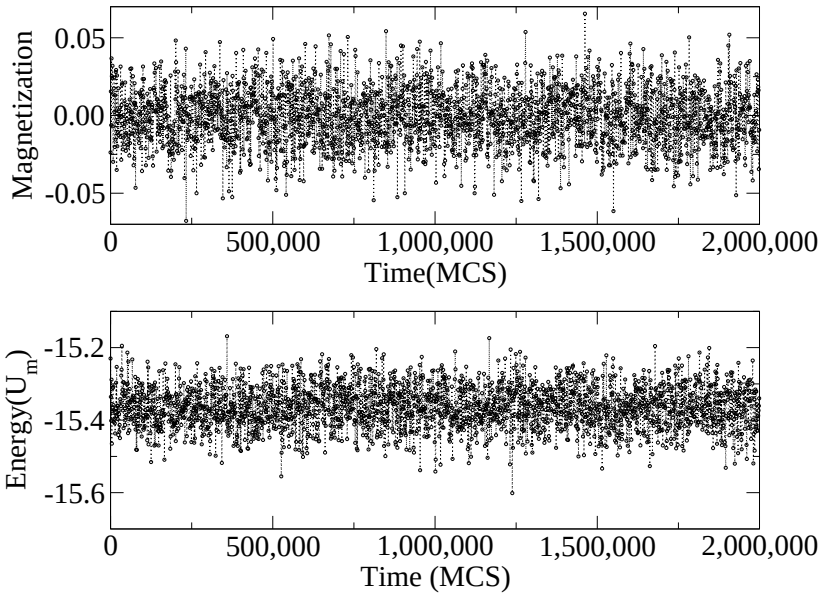


Figure 6.3: The time evolution of the first $2,000,000$ MCS. The dotted line is for visualization only. Data obtained at $L = 20$, $Q = 8$, $T = 5.128$.

## 6.3 Order Parameter Distribution

The distribution of magnetization is shown in Fig.6.4. The lattice sizes run from $L = 16$ to 48. Again, this is exactly rigid 3d Ising-like.

## 6.4 extract $1/\nu$

As we did for the diamond lattice, the exponent $\nu$ can be extracted from the scaling behaviors of various thermodynamic derivatives in Fig.6.5. The results are shown in Table 6.1. The estimate is $1/\nu = 1.5986 \pm 0.0071$, or $\nu = 0.6255 \pm 0.0028$. This value is very close to that of the rigid 3D Ising universality class, $\nu = 0.627 \pm 0.002$ [9].

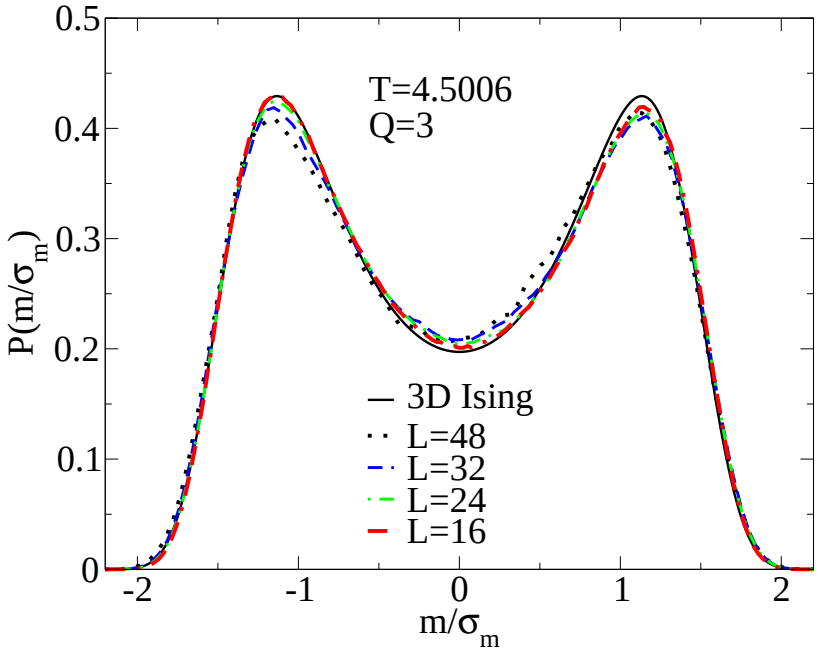


Figure 6.4: The magnetization distributions of different size systems near the critical temperature.

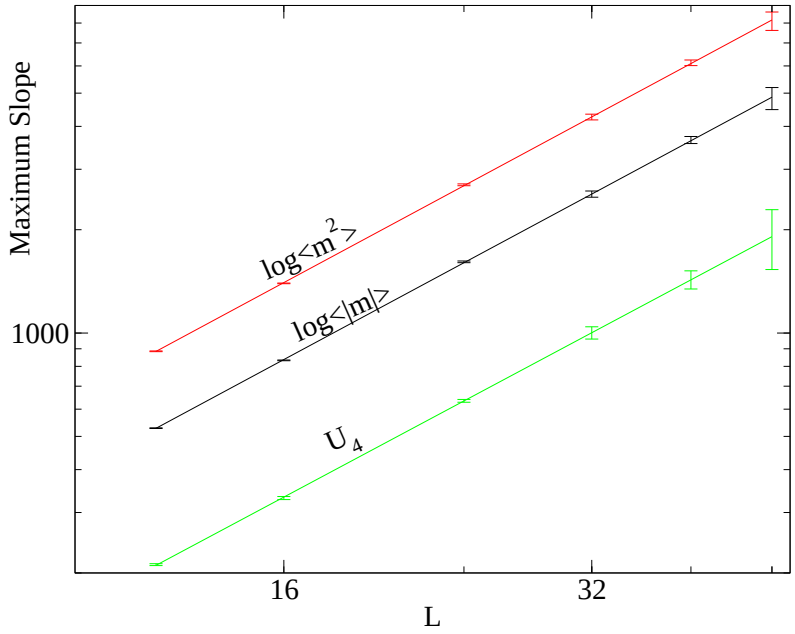


Figure 6.5: Log-log plot of the maximum slopes of various thermodynamic quantities used to determine $\nu$. The straight lines show the nonlinear least-square fit of Eq.2.27. All data points agree within one standard deviation.

Table 6.1: Estimates for $1/\nu$ obtained by finite size scaling of the maximum slopes of the cumulant and the logarithmic derivatives of $m^2$ and $|m|$.

| | $1/\nu$ |
|---|---|
| $U_4$ | $1.587 \pm 0.015$ |
| $\log < \|m\| >$ | $1.600 \pm 0.007$ |
| $\log < m^2 >$ | $1.603 \pm 0.007$ |

Table 6.2: Estimates for $K_c$ obtained by finite size scaling of the locations of the maximium slopes of various thermodynamic derivatives.

| | $K_c$ |
|---|---|
| $C$ | $0.222\ 191 \pm 0.000\ 006$ |
| $U_4$ | $0.222\ 182 \pm 0.000\ 005$ |
| $\log < m^2 >$ | $0.222\ 191 \pm 0.000\ 003$ |
| $\log < \|m\| >$ | $0.222\ 189 \pm 0.000\ 002$ |
| $\chi$ | $0.222\ 193 \pm 0.000\ 002$ |
| $\|m\|$ | $0.222\ 181 \pm 0.000\ 006$ |

## 6.5 EXTRACT $K_c$

The finite size scaling results of $K_c$ from various quantities are given in Fig.6.6 and Table 6.2. The $K_c$ is very close to that of the rigid simple cubic 3D Ising reported in Ref.[9]. The estimate is $K_c = 0.222\ 19 \pm 0.000\ 05$ (which is accidentally close to the rigid 3d Ising critical temperature $0.221\ 67 \pm 0.000\ 02$ [9]).

## 6.6 Binder Cumulant Crossing

The finite size effect of the Binder cumulant is shown in Fig.6.7. A closer look is given in Fig.6.8. The estimated value is $0.465 \pm 0.005$. The crossing value is very close to that of the rigid 3D Ising universality $U_4^* = 0.47$ [9].

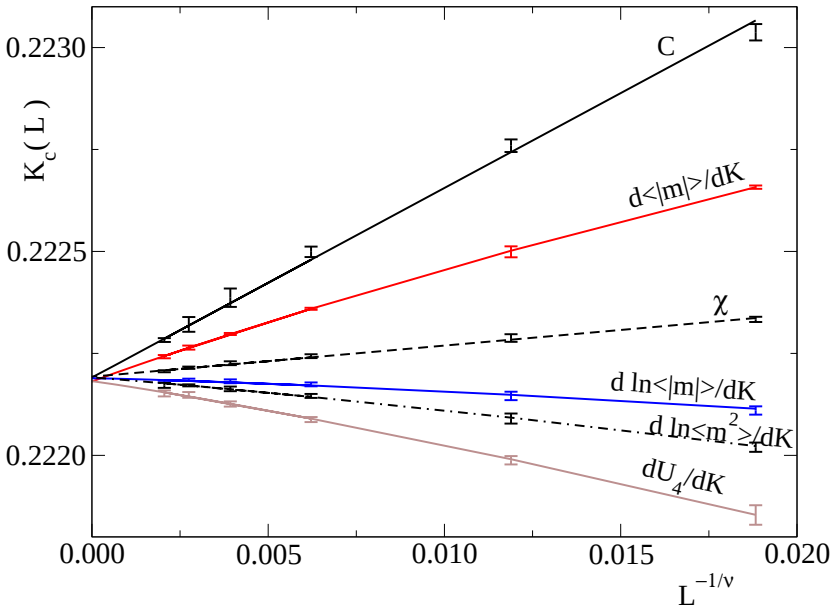


Figure 6.6: Size dependence of the finite-lattice critical temperature estimated from various properties. Data are shown with standard deviations. The solid lines are nonlinear least square fits to Eq.2.26.

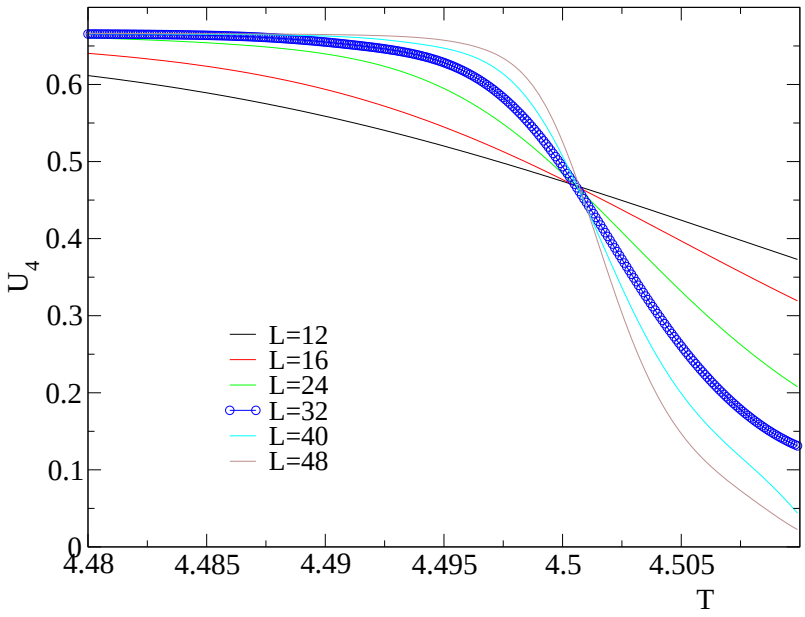


Figure 6.7: The finite size effect of the Binder cumulant. The curves are smooth because the data points are reweighted from histograms.

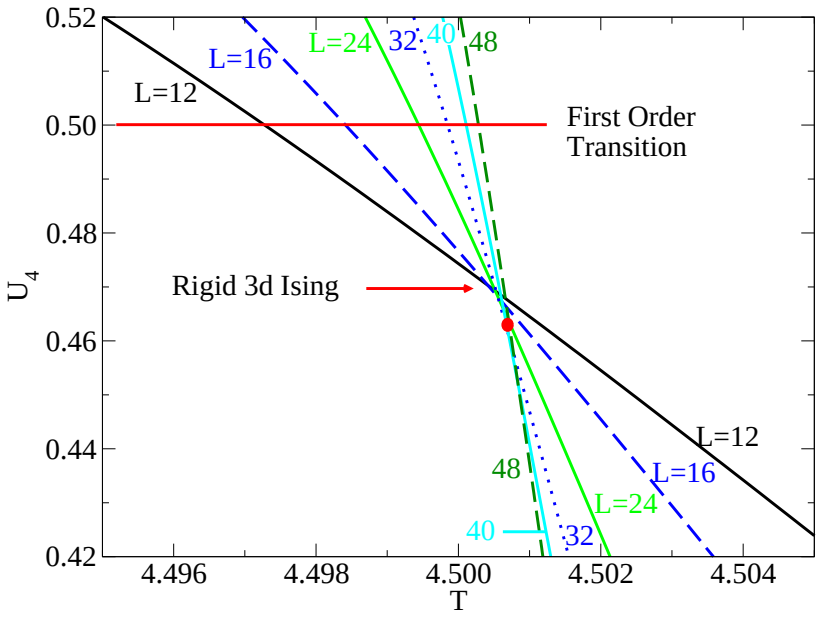


Figure 6.8: The Binder cumulant crossing. They are smooth because the data points are reweighted from histograms.

## 6.7 EXTRACT $\beta$

Fig.6.9 shows the finite size effect of magnetization. The nonlinear least square fit is shown in Fig.6.10. The estimated value is $\beta = 0.317 \pm 0.006$, slightly smaller than the Ising value $\beta = 0.3258 \pm 0.0044$ [9].

## 6.8 EXTRACT $\gamma$

We can extract the exponent $\gamma$ by examining the finite size scaling behaviors of the susceptibility $\chi$, as shown in Fig.6.11 and Fig.6.12. The best fitting is at temperature $T = 4.5005$. The estimated value is $\gamma = 1.252 \pm 0.005$ which is close to the Ising exponent $\gamma = 1.2470 \pm 0.0039$.

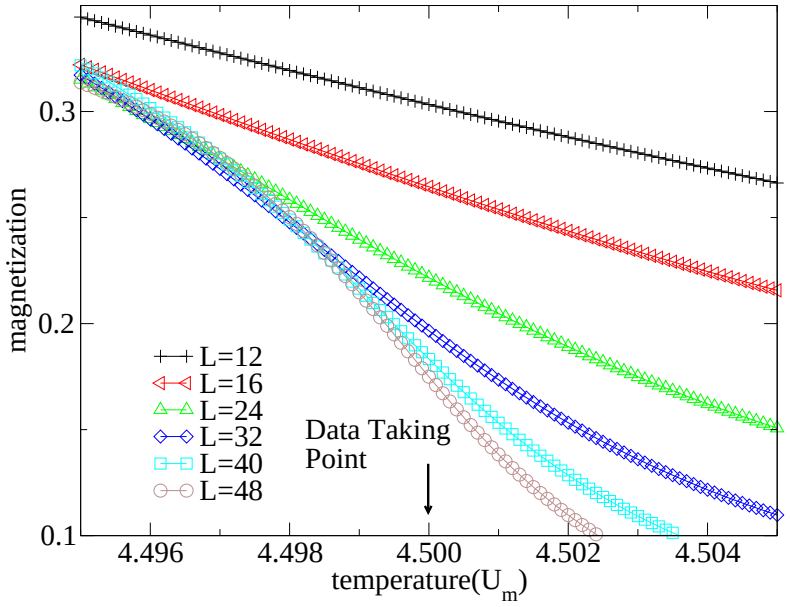


Figure 6.9: The finite size effect of magnetization.

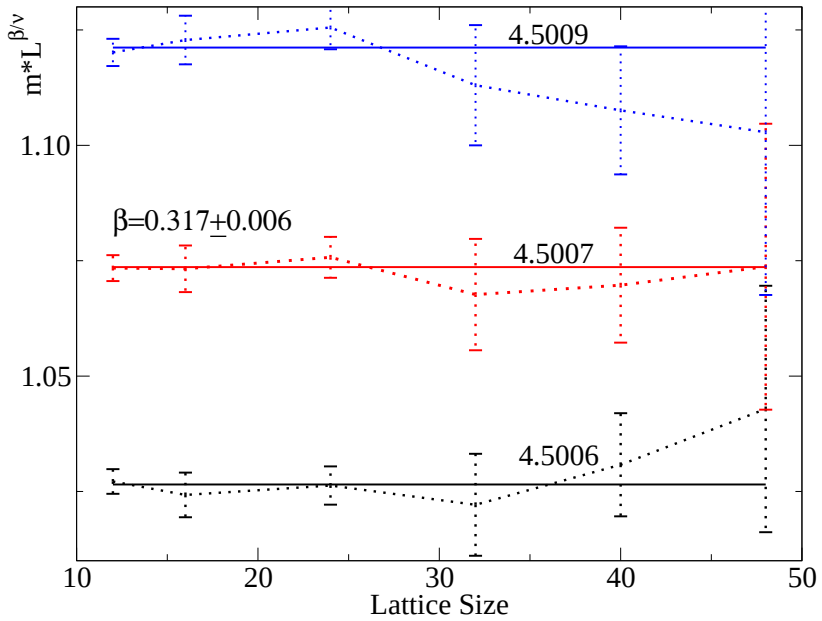


Figure 6.10: $mL^{\beta/\nu}$ vs. $L$ . At the critical temperature, the plots are supposed to be horizontal lines.

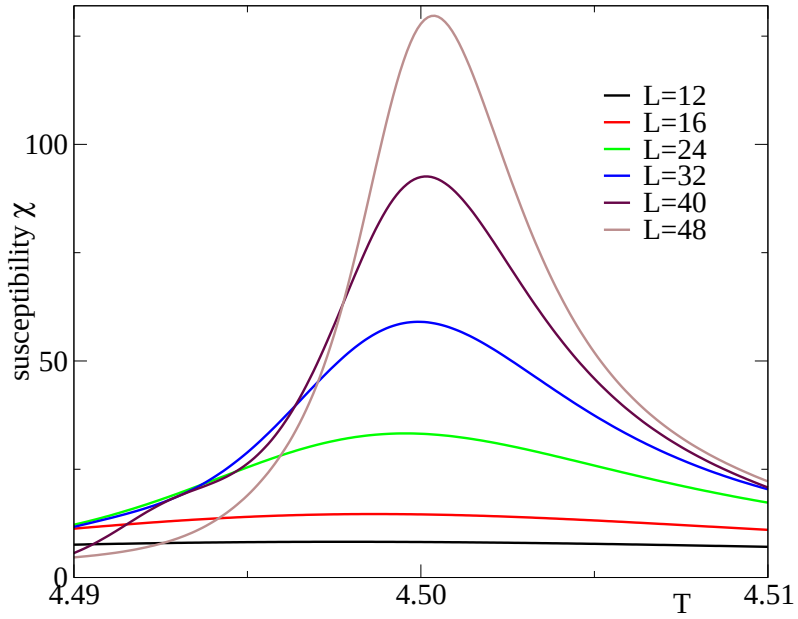


Figure 6.11: The finite size effect of the finite lattice susceptibility near criticality.

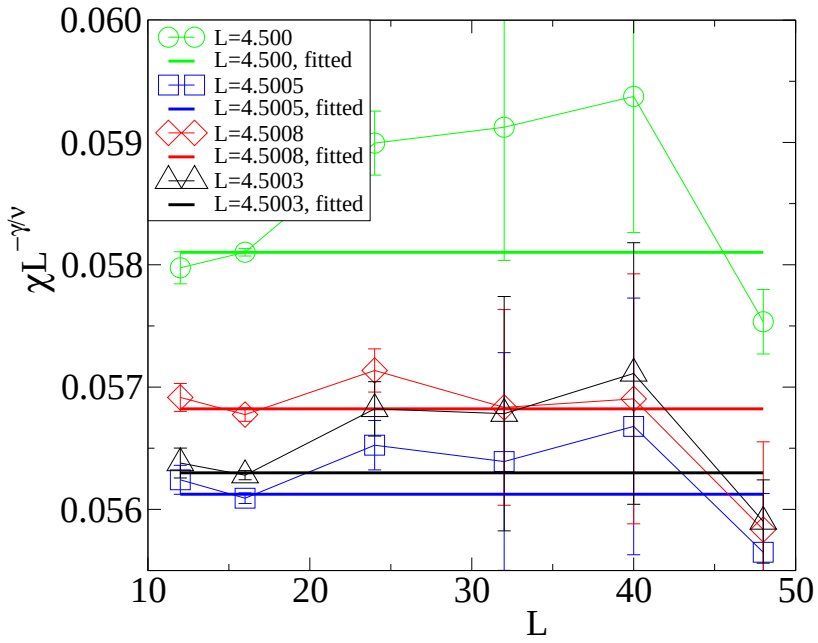


Figure 6.12: $\chi L^{-\gamma/\nu}$ vs. $L$. At the critical temperature, the plots are supposed to be horizontal lines.

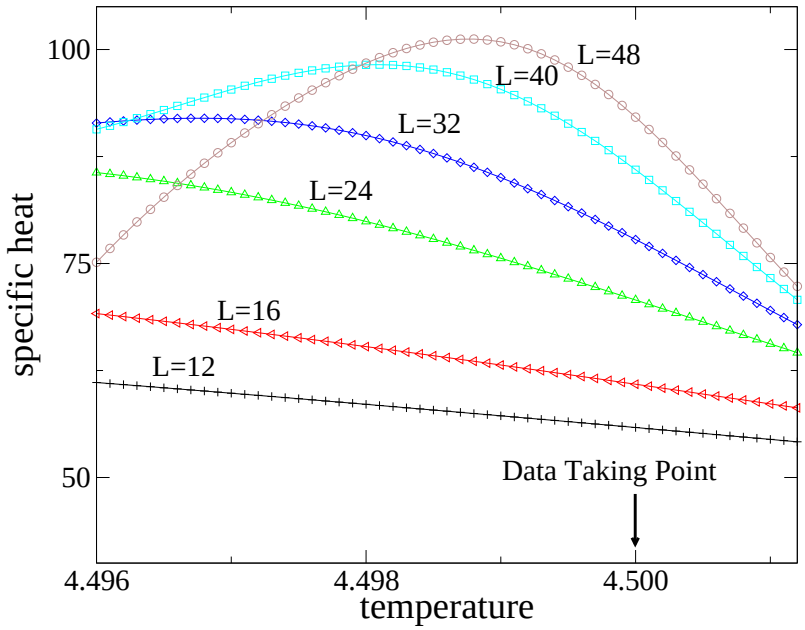


Figure 6.13: The finite size effect of the specific heat near criticality.

## 6.9 EXTRACT $\alpha$

The exponent $\alpha$ can be extracted from the scaling behavior of specific heat $C$, as shown in Fig.6.13 and Fig.6.14. The fitting result is $\alpha = 0.144 \pm 0.005$ which is larger than the rigid 3D Ising value 0.1070 [41].

## 6.10 SUMMARY

Despite the high elasticity in the stacked triangular lattice, the phase transition still belongs to rigid 3d Ising universality class within our lattice size range. We did not find any hint of crossover towards a first order transition, as predicted by the theory.

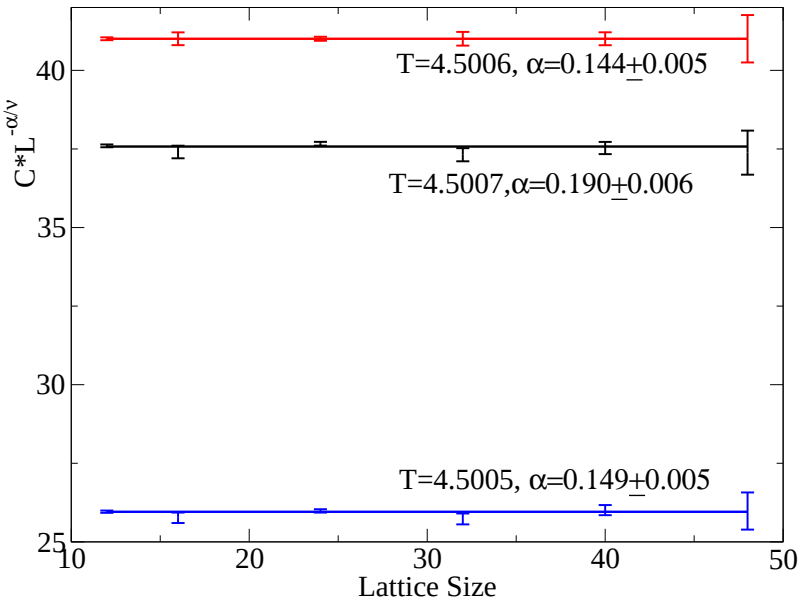


Figure 6.14: $CL^{-\alpha/\nu}$ vs. $L$. At the critical temperature, the plots are supposed to be horizontal lines.

# Chapter 7

## Conclusion and Future Work

We have investigated the critical phase behavior of an elastic antiferromagnetic Ising model with SW potential and thoroughly assessed its elastic degree of freedom. The simulations were performed at constant pressure on a semi-grand-canonical ensemble. The phase transition is found second order everywhere, which disagrees with the theory. The reason might be that the theory is overly simplified, or our lattice sizes are not large enough. Note that Dünweg [22] also points out that the deviation from Ising transition is intrinsically harder to detect in antiferromagnet case due to the quadratic coupling of the order parameter with the strain tensor in antiferromagnet. By examining the order parameter distribution and critical exponents, especially the crossing point of the Binder cumulant, we believe that the transition belongs to the universality class of the rigid three dimensional Ising model.

We also studied the critical phase behavior of a stacked triangular lattice at constant volume on a semi-grand-canonical ensemble that was reported to behave differently from the rigid 3d Ising model. However, the order parameter distribution, the Binder cumulant crossing and critical exponents unanimously show that it is still rigid 3d Ising-like. Future improvements may be made by re-implementing the system on a FCC lattice which is stable even if only two-body interactions exist. The system may also be made antiferromagnet with volume fluctuation, which involves some nontrivial changes of the code. Promising results might be possible after such changes.

# Appendix A

## Code for the Compressible Diamond Model

### A.1 The FORTRAN Code

```
c@PROCESS DIRECTIVE ('*VDIR:')
      PROGRAM DIAM8
C
C VERSION JAN 12, 1992
C FOR IBM ES / 9000
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C CORRECTED THE STATISTICAL TREATMENT OF THE VACANCIES
C
C MONTE CARLO SIMULATION OF A 3D SEMICONDUCTOR ALLOY
C INCLUDING VACANCIES ON A 3D DIAMOND LATTICE
C AT CONSTANT PRESSURE  Z E R O
C BOTH BULK SIMULATION AND THIN FILM GEOMETRY
C ELASTIC CONTRIBUTIONS VIA STILLINGER-WEBER POTENTIAL
C                           *****************
C THE LATTICE IS SET UP AS 8 INTERPENETRATING
C SIMPLE CUBIC LATTICES EACH OF WHICH HAS UNIT LATTICE CONSTANT
C SQUARE OF NEAREST NEIGHBOR DISTANCE IS 3 * 0.25 ** 2 = 0.1875
C THE FIRST 4 AND THE SECOND 4 SUBLATTICES FORM AN FCC, RESPECTIVELY
C SEE ASHCROFT / MERMIN P. 76
C THE 8 SUBLATTICES ARE INDEPENDENT; VECTORIZATION
C BY STANDARD CHECKERBOARD METHOD
C FOR NEIGHBOR TABLE SETUP IN THE BEGINNING,
C THE SC SUBLATTICE HAS LATTICE CONSTANT 4
C NUMBER OF PARTICLES IS NDIAM
C
C STATUS OF EACH SITE:
C 0 - VACANCY
C 1 A-ATOM
C -1 B-ATOM
C
```

```
C EACH SITE NEEDS 5 RANDOM NUMBERS:
C 3 FOR NEW COORDINATES
C 1 FOR NEW STATUS
C 1 FOR ACCEPTANCE
C
C MOREOVER, 4 RANDOM NUMBERS ARE NEEDED FOR BOX FLUCTUATIONS
C 3 FOR THE SPATIAL DIRECTIONS
C 1 FOR ACCEPTANCE
C
C THE SIMULATION USES THE TAUSWORTHE GENERATOR (1279,1063)
C
C
C BEGINNING OF DECLARATIONS
C
C
      PARAMETER(LSIMPX=24)
      PARAMETER(LSIMPY=24)
      PARAMETER(LSIMPZ=24)
      PARAMETER(NCAN=1000)
C
      PARAMETER(PSIMPX=LSIMPX)
      PARAMETER(PSIMPY=LSIMPY)
      PARAMETER(PSIMPZ=LSIMPZ)
      PARAMETER(FSIMPX=1.D0/PSIMPX)
      PARAMETER(FSIMPY=1.D0/PSIMPY)
      PARAMETER(FSIMPZ=1.D0/PSIMPZ)
C
      PARAMETER(NSIMP=LSIMPX*LSIMPY*LSIMPZ)
      PARAMETER(NDIAM=8*NSIMP)
C
      PARAMETER(FACSYS=1.D0/NDIAM)
C
      PARAMETER(NHALF=NDIAM/2)
      PARAMETER(NDIAM2=2*NDIAM)
      PARAMETER(NDIAM3=3*NDIAM)
      PARAMETER(NDIAM4=4*NDIAM)
      PARAMETER(NDIAM5=5*NDIAM)
C
      PARAMETER(NRAND=NDIAM5+4)
C
      PARAMETER(LS4X=4*LSIMPX)
      PARAMETER(LS4Y=4*LSIMPY)
      PARAMETER(LS4Z=4*LSIMPZ)
C
      PARAMETER(NBIT=32)
```

```
      PARAMETER(MERS=1279)
C
C DECLARATIONS FOR THE RANDOM GENERATOR
C
      INTEGER IRWRK(MERS)
      INTEGER IRAN(NRAND)
      INTEGER IRTOT(MERS + NRAND)
      EQUIVALENCE(IRTOT(1),IRWRK(1))
      EQUIVALENCE(IRTOT(1 + MERS),IRAN(1))
C
      COMMON/RANCOM/IRTOT
C
C STATUS AND COORDINATES OF THE ATOMS
C PARTICLE NO. NDIAM + 1 AS A DUMMY PARTICLE
C IN CASE THE PROGRAM RUNS A FREE SURFACE
C
      INTEGER ISTAT(NDIAM+1,2)
      DOUBLE PRECISION XCOORD(NDIAM+1,2),a0(3)
      DOUBLE PRECISION YCOORD(NDIAM+1,2)
      DOUBLE PRECISION ZCOORD(NDIAM+1,2)
C
C BIG SIMPLE CUBIC LATTICE FOR NEIGHBOR LIST SETUP
C
      INTEGER LABEL(LS4X,LS4Y,LS4Z)
C
C NEIGHBOR TABLES
C
      INTEGER NN(NDIAM,4)
      INTEGER NNN(NDIAM,12)
C
C BONDS
C
      DOUBLE PRECISION BONDX(NDIAM+1,4,2)
      DOUBLE PRECISION BONDY(NDIAM+1,4,2)
      DOUBLE PRECISION BONDZ(NDIAM+1,4,2)
      DOUBLE PRECISION bondsq(NDIAM+1,4,2)
C
      INTEGER INDBND(NDIAM+1,4,2)
      DOUBLE PRECISION BOND2(NDIAM+1,4,2)
C
C TABLE OF ANGLES MADE UP BY TWO BONDS
C
      INTEGER IANTBL(4,4)
C
C ANGLE TERMS
```

```
C
      INTEGER INDANG(NDIAM+1,6,2)
      DOUBLE PRECISION ANGTRM(NDIAM+1,6,2)
C
C TEMPORARY ANGLE TERMS
C
      INTEGER IANTMP(NDIAM,12)
      DOUBLE PRECISION ANGTMP(NDIAM,12)
C
C ENERGY DIFFERENCE
C
      DOUBLE PRECISION DELTAE(NDIAM)
      DOUBLE PRECISION delta1(NDIAM),delta2(NDIAM),delta3(NDIAM)
      DOUBLE PRECISION delta4(NDIAM)
C
C ACCEPTANCE POINTER
C
      INTEGER IACC(NDIAM)
C
C HAMILTONIAN PARAMETERS
C
CCHEM      DOUBLE PRECISION CHEM(-1:1)
CCHEM Define uniformed chemical potential and staggered potential
      double precision UCHEM
      double precision SCHEM
      DOUBLE PRECISION EPS(-1:1,-1:1)
      DOUBLE PRECISION ELAS(-1:1,-1:1)
      DOUBLE PRECISION RIDEAL(-1:1,-1:1)
      DOUBLE PRECISION RID2(-1:1,-1:1)
      DOUBLE PRECISION RID3(-1:1,-1:1,-1:1)
      DOUBLE PRECISION ANGLE(-1:1,-1:1,-1:1)
      DOUBLE PRECISION lambda(-1:1),ffat
C
      DOUBLE PRECISION EPS1(9)
      DOUBLE PRECISION ELAS1(9)
      DOUBLE PRECISION RIDE1(9)
      DOUBLE PRECISION RID21(9)
      DOUBLE PRECISION RID31(27)
      DOUBLE PRECISION ANGLE1(27)
      EQUIVALENCE (EPS1(1),EPS(-1,-1))
      EQUIVALENCE (ELAS1(1),ELAS(-1,-1))
      EQUIVALENCE (RIDE1(1),RIDEAL(-1,-1))
      EQUIVALENCE (RID21(1),RID2(-1,-1))
      EQUIVALENCE (RID31(1),RID3(-1,-1,-1))
      EQUIVALENCE (ANGLE1(1),ANGLE(-1,-1,-1))
```

```
C
C TRANSLATION VECTORS TO THE SUBLATTICES
C
      INTEGER IOFFX(8)
      INTEGER IOFFY(8)
      INTEGER IOFFZ(8)
C
C JUMP VECTORS TO THE NEAREST NEIGHBORS, STARTING
C FROM SITES OF THE FIRST FCC SUBLATTICE
C
      INTEGER JUMPX(4), JUMPY(4), JUMPZ(4)
C
C HELP ARRAYS FOR NEXT NEAREST NEIGHBORS
C
      INTEGER IEXCL(3,4)
C
C FUNCTIONS
C
       DOUBLE PRECISION func2,func3,ry,ry1,ry2
C
C
C STATISTICAL AVERAGES
C
      DOUBLE PRECISION UAV(4)
      DOUBLE PRECISION FAV(-1:1,2,4)
      DOUBLE PRECISION UMXAV(4)
      DOUBLE PRECISION ORDAV(4)
      DOUBLE PRECISION VAV(4)
      DOUBLE PRECISION ACCAV
      DOUBLE PRECISION ACVAV
      DOUBLE PRECISION a0V,a02
      DOUBLE PRECISION DISTMED(4),DISTMED2(4),ODISTMED(4),ODISTMED2(4)
      DOUBLE PRECISION SIGMAD(4),SIGMADNUM(4),RNUMDIST(4)
      DOUBLE PRECISION RNUMDIST2(4)
      DOUBLE PRECISION ANGMED(6),ANGMED2(6),RNUMMED(6),RNUMMED2(6)
      DOUBLE PRECISION SIGMAANG(6),SIGMANUM(6),DANGMED(6),DANGMED2(6)
      DOUBLE PRECISION FAC(6),FACTO(4),FACT3
      DOUBLE PRECISION concsi,concge,inv_csi,inv_cge
      INTEGER NUMDIST(4),NUMDIST2(4),NUMDIST1(4),NUMB(4)
      INTEGER NUMMED(6),NUMMED2(6),NUM1(6)
C
      DOUBLE PRECISION UAVOUT(4)
      DOUBLE PRECISION FAVOUT(-1:1,2,4)
      DOUBLE PRECISION UMXOUT(4)
      DOUBLE PRECISION ORDOUT(4)
```

```
      DOUBLE PRECISION VAVOUT(4)
      DOUBLE PRECISION ACCOUT
      DOUBLE PRECISION ACVOUT
C
C C(n)
C
      DOUBLE PRECISION cng1(0:9),cns1(0:9)
      DOUBLE PRECISION c2s1(0:12),c2g1(0:12),ctots1(0:16),ctotg1(0:16)
      INTEGER cng(0:9),cns(0:9),c2s(0:12),c2g(0:12),ctots(0:16)
      INTEGER ctotg(0:16)
CZHU  variables related to parallelization
      integer myID, ierror,numprocs,f21,f22,f23,f27,f28,f24,ioffset
C
C HISTOGRAM (G(r), P(teta))
C
      DOUBLE PRECISION RINT(4),DMAX(4),DMIN(4)
      DOUBLE PRECISION RHISTO(NCAN,4),d0(NCAN,4),DNN(4)
      DOUBLE PRECISION RHISTANG(NCAN,6),teta(NCAN),ANG(6),DEG
      INTEGER NUM(6),HISTO(NCAN,4),HISTANG(NCAN,6)
      pigreek=dacos(-1.d0)/180.d0
CZHU
CZHU  parallelizing the code
CZHU  initialization
CZHU
      CALL MPI_INIT(ierror)
      CALL MPI_COMM_RANK(MPI_COMM_WORLD,myID,ierror)
      CALL MPI_COMM_SIZE(MPI_COMM_WORLD,numprocs, ierror)
      ioffset=(myID-2)*10
      f21=21+ioffset
      f22=22+ioffset
      f23=23+ioffset
      f27=27+ioffset
      f28=28+ioffset
      f24=24+ioffset
      WRITE(f24,*) 'After MPI call. Next, read input'
C
C DECLARATION PART FINISHED
C
C READING OF SIMULATION PARAMETERS FROM INPUT FILE
C
C

      IFREES = 0
      READ(5,*) IFREES
      READ(5,*) MCSINI
```

```
      READ(5,*) MCSMAX
      READ(5,*) NOUTCF
      READ(5,*) NOUTOB
      READ(5,*) MSTART
      READ(5,*) NOUTAV
      READ(5,*) ISEED
      READ(5,*) PMOV
      READ(5,*) PBOX
      READ(5,*) TEMP
      do l=1,4
         READ(5,*) dmin(l)
         READ(5,*) dmax(l)
      enddo
c
      WRITE(f24,*) 'My process ID is ',myID
      WRITE(f24,*) 'Original ISEED = ', ISEED
      ISEED=ISEED+947582*myID
      WRITE(f24,*) 'ISEED = ',ISEED
      WRITE(f24,*) 'TOTAL NUMBER OF SIMULATED SITES = ',NDIAM
      WRITE(f24,*) 'SIMULATION GEOMETRY (0=BULK, 1=THIN FILM) = ',IFREES
      WRITE(f24,*) 'INITIAL VALUE FOR TIME MCS (USUALLY 1) = ',MCSINI
      WRITE(f24,*) 'FINAL VALUE FOR TIME MCS = ',MCSMAX
      WRITE(f24,*) 'TIME INTERVAL FOR DUMPING CONFIGURATIONS = ',NOUTCF
      WRITE(f24,*) 'TIME INTERVAL FOR OUTPUT OBSERVABLES = ',NOUTOB
      WRITE(f24,*) 'START AVERAGING AT MCS = ',MSTART
      WRITE(f24,*) 'TIME INTERVAL FOR OUTPUT STATISTICAL AVERAGES = ',
     *            NOUTAV
      WRITE(f24,*) 'MAXIMUM TRIAL MOVE = ',PMOV
      WRITE(f24,*) 'MAXIMUM RELATIVE MOVE IN BOX SIZE = ',PBOX
      WRITE(f24,*) ' '
      WRITE(f24,*) 'TEMPERATURE = ',TEMP
      WRITE(f24,*) ' '
      WRITE(f24,*) 'Parameters for g(r)'
c
      KOUTCF = NOUTCF + MCSINI - 1
      KOUTOB = NOUTOB + MCSINI - 1
      KOUTAV = NOUTAV + MSTART - 1
      BETA = 1.D0 / TEMP
CZHU
CZHU  dealing with the case TEMP=0.0
CZHU
      if(TEMP.EQ.0.0)then
        BETA=1.0e30
        WRITE(f24,*) 'temp = 0, reset beta '
        WRITE(f24,*) 'BETA = ', BETA
```

```
      endif
c
C
C HAMILTONIAN PARAMETERS
C
CCHEM      DO 2 I = -1,1
CCHEM         CHEM(I) = 0.D0
CCHEM 2    CONTINUE
      UCHEM=0.D0
      SCHEM=0.D0
      DO 3 I = 1,9
         EPS1(I)  = 0.D0
         RIDE1(I) = 0.D0
         RID21(I) = 0.D0
 3    CONTINUE
      DO 4 I = 1,27
         RID31(I)  = 0.D0
         ANGLE1(I) = 0.D0
 4    CONTINUE
C
CCHEM      READ(5,*) CHEM(1)
CCHEM      READ(5,*) CHEM(-1)
      read(5,*) UCHEM
      read(5,*) SCHEM

      READ(5,*) EPS(1,1)
      READ(5,*) EPS(-1,-1)
      READ(5,*) EPS(1,-1)
      READ(5,*) RID2(1,1)
      READ(5,*) RID2(-1,-1)
      READ(5,*) RID2(1,-1)
c
      READ(5,*) lambda(1)
      READ(5,*) lambda(-1)
c
      WRITE(f24,*) ' '
CCHEM      WRITE(f24,*) 'CHEMICAL POTENTIAL SPECIES A = ',CHEM(1)
CCHEM      WRITE(f24,*) 'CHEMICAL POTENTIAL SPECIES B = ',CHEM(-1)
      WRITE(f24,*) 'UNIFORM CHEMICAL POTENTIAL = ',UCHEM
      WRITE(f24,*) 'STAGGERED CHEMICAL POTENTIAL = ',SCHEM
      WRITE(f24,*) ' '
      WRITE(f24,*) 'BOND ENERGY A - A = ',EPS(1,1)
      WRITE(f24,*) 'BOND ENERGY B - B = ',EPS(-1,-1)
      WRITE(f24,*) 'BOND ENERGY A - B = ',EPS(1,-1)
      WRITE(f24,*) ' '
```

```
      WRITE(f24,*) 'BOND LENGTH A - A = ',RID2(1,1)
      WRITE(f24,*) 'BOND LENGTH B - B = ',RID2(-1,-1)
      WRITE(f24,*) 'BOND LENGTH A - B = ',RID2(1,-1)
      WRITE(f24,*) ' '
c
      EPS(-1,1) = EPS(1,-1)
      RID2(-1,1) = RID2(1,-1)
c
c SCALE ALL DISTANCE WITH RESPECT TO SILICON LATTICE CONSTANT
c
      fatt1=1.d0
      a0(1)=5.6487488
      a0(2)=0.0
      a0(3)=5.4219888*fatt1
      a02=a0(3)**2
      a0V=a0(3)/LSIMPX
c
      do i=1,9
         RID21(I)=RID21(I)/a0(3)
      enddo
c
C
C SCALE ALL ENERGIES WITH TEMPERATURE
C DEFINE MORE BOND LENGTH CONSTANTS
C
CCHEM      DO 5 I = -1,1
CCHEM         CHEM(I) = BETA * CHEM(I)
CCHEM 5    CONTINUE
      UCHEM=BETA*UCHEM
      SCHEM=BETA*SCHEM

      ffat=2.d0**(-0.1666666666)
      DO 6 I = 1,9
         EPS1(I)  = BETA * EPS1(I)
         RIDE1(I) = RID21(I)*ffat
 6    CONTINUE
      WRITE(f24,*)
      WRITE(f24,*)'ANGLE CONSTANTS'
      DO 8 K = -1,1
         DO 8 J = -1,1
            DO 8 I = -1,1
               RID3(I,J,K) = (lambda(I)*(lambda(J)**2)*lambda(K))
     1                       **(0.25) * sqrt(EPS(I,J) * EPS(J,K))
            WRITE(f24,*)I,J,K,RID3(I,J,K)*TEMP
 8    CONTINUE
```

```
c From John's input file:
c      RID3(1,1,1)=45.47
c      RID3(-1,-1,-1)=59.83
c      RID3(1,-1,1)=52.1093
c      RID3(-1,1,-1)=52.1093
c      RID3(1,-1,-1)=55.8337
c      RID3(-1,1,1)=48.4239
c
      WRITE(f24,*) ' '
      WRITE(f24,*) 'TEMP = ',TEMP
C
C INITIAL CONFIGURATION: EACH FCC SUBLATTICE UNIFORMLY
C FILLED WITH ONE SPECIES, AND PERFECTLY ORDERED
C DIAMOND LATTICE. FIND OUT LOWEST ENERGY, TAKING
C INTO ACCOUNT THESE CONSTRAINTS.
C START WITH ALL B
C
      ISTAT1 = 1
      ISTAT2 = -1
CCHEM      UINT = - CHEM(1) - 2.D0 * EPS(1,1)
      UINT =  - 2.D0 * EPS(1,1)
      DO 15 J = -1,1
         DO 15 I = -1,1
            INDOLD = 5 + 3 * I + J
            if (ABS(RIDE1(INDOLD)).gt.0.000001) THEN
               ry=.433012702/RIDE1(INDOLD)
            else
               ry=1000000
            endif
CCHEM            TST = - 0.5D0 * CHEM(J) - 0.5D0 * CHEM(I)
    TST = -UCHEM*(I+J)-SCHEM*(I-J)
     &         + 2.D0 * EPS(I,J) * func2(ry)
            WRITE(f24,*)'UINT all''inizio del run:',TST * TEMP,I,J
            IF(TST.LT.UINT) THEN
               ISTAT1 = I
               ISTAT2 = J
               UINT   = TST
            END IF
 15   CONTINUE
      UOUT = UINT * TEMP
      WRITE(f24,*) ' '
      WRITE(f24,*) 'INITIAL SETTING:'
      WRITE(f24,*) 'SIMULATION WILL START IN STATE ',ISTAT1,' ',ISTAT2
      WRITE(f24,*) 'AT INTERNAL ENERGY PER SITE ',UOUT
      WRITE(f24,*) ' '
```

```
C
C DEFINE SHIFT VECTORS TO THE ORIGINS OF THE
C VARIOUS SUBLATTICES
C
      IOFFX(1) = 0
      IOFFY(1) = 0
      IOFFZ(1) = 0
C
      IOFFX(2) = 2
      IOFFY(2) = 2
      IOFFZ(2) = 0
C
      IOFFX(3) = 2
      IOFFY(3) = 0
      IOFFZ(3) = 2
C
      IOFFX(4) = 0
      IOFFY(4) = 2
      IOFFZ(4) = 2
C
      IOFFX(5) = 1
      IOFFY(5) = 1
      IOFFZ(5) = 1
C
      IOFFX(6) = 3
      IOFFY(6) = 3
      IOFFZ(6) = 1
C
      IOFFX(7) = 3
      IOFFY(7) = 1
      IOFFZ(7) = 3
C
      IOFFX(8) = 1
      IOFFY(8) = 3
      IOFFZ(8) = 3
C
C FILL THE LATTICE WITH PARTICLE INDICES IN THE FOLLOWING ORDER:
C SUBLATTICE 1
C SUBLATTICE 2
C ETC.
C
      IPART = 0
      DO 20 ISUBL = 1,8
         DO 20 IZ = 1 + IOFFZ(ISUBL),LS4Z + IOFFZ(ISUBL),4
            DO 20 IY = 1 + IOFFY(ISUBL),LS4Y + IOFFY(ISUBL),4
```

```
               DO 20 IX = 1 + IOFFX(ISUBL),LS4X + IOFFX(ISUBL),4
                  IPART           = IPART + 1
                  LABEL(IX,IY,IZ) = IPART
 20   CONTINUE
C
C DEFINE THE JUMP VECTORS TO THE NEAREST NEIGHBORS
C
      JUMPX(1) = 1
      JUMPY(1) = 1
      JUMPZ(1) = 1
C
      JUMPX(2) = 1
      JUMPY(2) = -1
      JUMPZ(2) = -1
C
      JUMPX(3) = -1
      JUMPY(3) = 1
      JUMPZ(3) = -1
C
      JUMPX(4) = -1
      JUMPY(4) = -1
      JUMPZ(4) = 1
C
C SET UP TABLE OF NEAREST NEIGHBORS
C IN CASE OF A FREE SURFACE, THE SURFACE SITES ARE
C NEIGHBORED BY THE DUMMY PARTICLE
C
      DO 120 INN = 1,4
         MOVX = JUMPX(INN)
         MOVY = JUMPY(INN)
         MOVZ = JUMPZ(INN)
         DO 120 ISUBL = 1,8
            IF(ISUBL.EQ.5) THEN
               MOVX = - MOVX
               MOVY = - MOVY
               MOVZ = - MOVZ
            END IF
            DO 120 IZ = 1 + IOFFZ(ISUBL),LS4Z + IOFFZ(ISUBL),4
               IZNEW = IZ + MOVZ
               IZFLAG = 0
               IF(IZNEW.LT.1) THEN
                  IZNEW = IZNEW + LS4Z
                  IF(IFREES.EQ.1) IZFLAG = 1
               END IF
               IF(IZNEW.GT.LS4Z) THEN
```

```
                  IZNEW = IZNEW - LS4Z
                  IF(IFREES.EQ.1) IZFLAG = 1
               END IF
               DO 120 IY = 1 + IOFFY(ISUBL),LS4Y + IOFFY(ISUBL),4
                  IYNEW = IY + MOVY
                  IF(IYNEW.LT.1) IYNEW = IYNEW + LS4Y
                  IF(IYNEW.GT.LS4Y) IYNEW = IYNEW - LS4Y
                  DO 120 IX = 1 + IOFFX(ISUBL),LS4X + IOFFX(ISUBL),4
                     IXNEW = IX + MOVX
                     IF(IXNEW.LT.1) IXNEW = IXNEW + LS4X
                     IF(IXNEW.GT.LS4X) IXNEW = IXNEW - LS4X
                     IPART = LABEL(IX,IY,IZ)
                     IF(IZFLAG.EQ.1) THEN
                        NN(IPART,INN) = NDIAM + 1
                     ELSE
                        NN(IPART,INN) = LABEL(IXNEW,IYNEW,IZNEW)
                     END IF
 120  CONTINUE
C
C FIND NEXT NEAREST NEIGHBORS
C REACH NNN NO. 1 - 3   VIA BOND NO. 1
C  "     "   "  4 - 6    "   "    "  2
C  "     "   "  7 - 9    "   "    "  3
C  "     "   "  10 - 12  "   "    "  4
c
C
      DO 130 INN = 1,4
         KOUNT = 0
         DO 130 KNN = 1,4
            IF(KNN.NE.INN) THEN
               KOUNT            = KOUNT + 1
               IEXCL(KOUNT,INN) = KNN
            END IF
 130  CONTINUE
C
      DO 150 INN1 = 1,4
         DO 150 INEXT = 1,3
            INNN = (INN1 - 1) * 3 + INEXT
            INN2 = IEXCL(INEXT,INN1)
            MOVX = JUMPX(INN1) - JUMPX(INN2)
            MOVY = JUMPY(INN1) - JUMPY(INN2)
            MOVZ = JUMPZ(INN1) - JUMPZ(INN2)
            DO 150 ISUBL = 1,8
               IF(ISUBL.EQ.5) THEN
                  MOVX = - MOVX
```

```
                  MOVY = - MOVY
                  MOVZ = - MOVZ
               END IF
               DO 150 IZ = 1 + IOFFZ(ISUBL),LS4Z + IOFFZ(ISUBL),4
                  IZNEW = IZ + MOVZ
                  IZFLAG = 0
                  IF(IZNEW.LT.1) THEN
                     IZNEW = IZNEW + LS4Z
                     IF(IFREES.EQ.1) IZFLAG = 1
                  END IF
                  IF(IZNEW.GT.LS4Z) THEN
                     IZNEW = IZNEW - LS4Z
                     IF(IFREES.EQ.1) IZFLAG = 1
                  END IF
                  DO 150 IY = 1 + IOFFY(ISUBL),LS4Y + IOFFY(ISUBL),4
                     IYNEW = IY + MOVY
                     IF(IYNEW.LT.1) IYNEW = IYNEW + LS4Y
                     IF(IYNEW.GT.LS4Y) IYNEW = IYNEW - LS4Y
                     DO 150 IX = 1 + IOFFX(ISUBL),LS4X + IOFFX(ISUBL),4
                        IXNEW = IX + MOVX
                        IF(IXNEW.LT.1) IXNEW = IXNEW + LS4X
                        IF(IXNEW.GT.LS4X) IXNEW = IXNEW - LS4X
                        IPART = LABEL(IX,IY,IZ)
                        IF(IZFLAG.EQ.1) THEN
                           NNN(IPART,INNN) = NDIAM + 1
                        ELSE
                           NNN(IPART,INNN) = LABEL(IXNEW,IYNEW,IZNEW)
                        END IF
 150  CONTINUE
992    format(2(2x,I6),10x,I6)
C
C ASSIGN COORDINATES AND STATUS TO THE BULK PARTICLES
C
      IOLDCF = 1
      INEWCF = 2
C
      IPART = 0
      IASS  = ISTAT1
      DO 200 ISUBL = 1,8
         IF(ISUBL.EQ.5) IASS = ISTAT2
         DO 200 IZ = 1 + IOFFZ(ISUBL),LS4Z + IOFFZ(ISUBL),4
            ZZZ = 0.25D0 * IZ
            DO 200 IY = 1 + IOFFY(ISUBL),LS4Y + IOFFY(ISUBL),4
               YYY = 0.25D0 * IY
               DO 200 IX = 1 + IOFFX(ISUBL),LS4X + IOFFX(ISUBL),4
```

```
                  IPART                = IPART + 1
                  XCOORD(IPART,IOLDCF) = 0.25D0 * IX
                  YCOORD(IPART,IOLDCF) = YYY
                  ZCOORD(IPART,IOLDCF) = ZZZ
                  ISTAT(IPART,IOLDCF)  = IASS
 200  CONTINUE
C
C ASSIGN STATUS AND COORDINATES FOR THE DUMMY PARTICLE
C
      XCOORD(NDIAM + 1,1) = 0.D0
      XCOORD(NDIAM + 1,2) = 0.D0
      YCOORD(NDIAM + 1,1) = 0.D0
      YCOORD(NDIAM + 1,2) = 0.D0
      ZCOORD(NDIAM + 1,1) = 0.D0
      ZCOORD(NDIAM + 1,2) = 0.D0
      ISTAT(NDIAM + 1,1) = 0
      ISTAT(NDIAM + 1,2) = 0
C
C INITIALIZE RANDOM GENERATOR
C
      CALL INIRAN(ISEED)
C
C NORMALIZATION FACTORS FOR THE RANDOM NUMBERS
C IBIG IS LARGEST INTEGER ON A NBIT MACHINE
C
      IBIG   = 2 ** (NBIT - 2)
      IHLP   = IBIG - 1
      IBIG   = IBIG + IHLP
      FNORM  = 1.D0 / IBIG
      FNORM2 = FNORM * 2.D0
      FNSTAT = FNORM * 3.D0
C
      XBOX   = PBOX * PSIMPX
      YBOX   = PBOX * PSIMPY
      ZBOX   = PBOX * PSIMPZ
      FXBOX  = XBOX * FNORM2
      FYBOX  = YBOX * FNORM2
      FZBOX  = ZBOX * FNORM2
C
      PMOVX  = PMOV * FSIMPX
      PMOVY  = PMOV * FSIMPY
      PMOVZ  = PMOV * FSIMPZ
C
C BOX SIZE PARAMETERS
C
```

```
      PERIX = PSIMPX
      PERIY = PSIMPY
      PERIZ = PSIMPZ
C
C THIS IS THE SETTING FOR SETUP FROM SCRATCH
C IF POSSIBLE, OVERWRITE THE CONFIGURATION BY
C DATA FROM INPUT FILE, UNIT 21
C
c       do  I=1,NDIAM
c           read(28,*)XCOORD(I,IOLDCF),YCOORD(I,IOLDCF),
c     &                ZCOORD(I,IOLDCF),ISTAT(I,IOLDCF)
c       enddo
c       read(28,*)PERIX,PERIY,PERIZ
c
      READ(f28,END=300) (IRWRK(I),I=1,MERS),
     &                 (XCOORD(I,IOLDCF),I=1,NDIAM),
     &                 (YCOORD(I,IOLDCF),I=1,NDIAM),
     &                 (ZCOORD(I,IOLDCF),I=1,NDIAM),
     &                 (ISTAT(I,IOLDCF),I=1,NDIAM),
     &                 PERIX,PERIY,PERIZ
      WRITE(f24,*) 'INITIAL SETTING OVERRIDDEN BY INPUT FILE'
 300  CONTINUE
C
C
      FPERIX = 2.D0 / PERIX
      FPERIY = 2.D0 / PERIY
      FPERIZ = 2.D0 / PERIZ
C
C DETERMINE SITE PART OF INTERNAL ENERGY
C
      UINT = 0.D0
CCHEM       DO 500 I = 1,NDIAM
      DO 500 I = 1,NHALF
CCHEM         UINT = UINT - CHEM(ISTAT(I,IOLDCF))
        UINT=UINT-UCHEM*ISTAT(I,IOLDCF)-SCHEM*ISTAT(I,IOLDCF)
 500  CONTINUE
      DO 501 I = 1+NHALF,NDIAM
CCHEM         UINT = UINT + CHEM(ISTAT(I,IOLDCF))
        UINT=UINT-UCHEM*ISTAT(I,IOLDCF)+SCHEM*ISTAT(I,IOLDCF)
 501  CONTINUE
C
C INITIALIZE BONDS
C
C
      DO 510 INN = 1,4
```

```
         DO 510 I = 1,NDIAM
            INEI = NN(I,INN)
            INDOLD = 5 + 3 * ISTAT(INEI,IOLDCF) + ISTAT(I,IOLDCF)
            DDXX = XCOORD(INEI,IOLDCF) - XCOORD(I,IOLDCF)
            DDYY = YCOORD(INEI,IOLDCF) - YCOORD(I,IOLDCF)
            DDZZ = ZCOORD(INEI,IOLDCF) - ZCOORD(I,IOLDCF)
            DDXX = DDXX - PERIX * INT(FPERIX * DDXX)
            DDYY = DDYY - PERIY * INT(FPERIY * DDYY)
            DDZZ = DDZZ - PERIZ * INT(FPERIZ * DDZZ)
            ddsq=sqrt(DDXX ** 2 + DDYY ** 2 + DDZZ ** 2)
            ry=ddsq/ride1(INDOLD)
            DD22 = EPS1(INDOLD) * func2(ry)
            INDBND(I,INN,IOLDCF) = INDOLD
            BONDX(I,INN,IOLDCF)  = DDXX
            BONDY(I,INN,IOLDCF)  = DDYY
            BONDZ(I,INN,IOLDCF)  = DDZZ
            BOND2(I,INN,IOLDCF)  = DD22
            bondsq(I,INN,IOLDCF) = ddsq
 510  CONTINUE
C
C DETERMINE BOND PART OF INTERNAL ENERGY
C
      WRITE(f24,*) 'H1 = CHEM =',UINT* TEMP * FACSYS
      RH1=UINT* TEMP * FACSYS
      DO 560 INN = 1,4
         DO 560 I = 1,NHALF
            uu = uu + BOND2(I,INN,IOLDCF)
            UINT = UINT + BOND2(I,INN,IOLDCF)
 560  CONTINUE
      WRITE(f24,*)  'H2 = UINT1 =',UU * TEMP * FACSYS
      RH2=UU * TEMP * FACSYS
      WRITE(f24,*)  'UINT now (H1+H2) =',UINT* TEMP * FACSYS
C
C TABLE OF ANGLES AS A FUNCTION OF BONDS
C
      IANTBL(1,2) = 1
      IANTBL(1,3) = 2
      IANTBL(1,4) = 3
      IANTBL(2,3) = 4
      IANTBL(2,4) = 5
      IANTBL(3,4) = 6
C
      IANTBL(2,1) = 1
      IANTBL(3,1) = 2
      IANTBL(4,1) = 3
```

```
      IANTBL(3,2) = 4
      IANTBL(4,2) = 5
      IANTBL(4,3) = 6
C
C ANGLE CONTRIBUTIONS WHOSE VERTEX IS THE CENTRAL SITE
C
      DO 600 INN1 = 1,3
         DO 600 INN2 = INN1 + 1,4
            IANGLE = IANTBL(INN2,INN1)
            DO 600 I = 1,NDIAM
               INEI1 = NN(I,INN1)
               INEI2 = NN(I,INN2)
               INDOLD2b1 = 5 + 3 * ISTAT(INEI1,IOLDCF) +
     1                     ISTAT(I,IOLDCF)
               INDOLD2b2 = 5 + 3 * ISTAT(INEI2,IOLDCF) +
     1                     ISTAT(I,IOLDCF)
               INDOLD = 14 + ISTAT(INEI1,IOLDCF)
     &              +  9 * ISTAT(INEI2,IOLDCF)
     &              +  3 * ISTAT(I,IOLDCF)
               INDANG(I,IANGLE,IOLDCF) = INDOLD
               cos=( BONDX(I,INN1,IOLDCF) * BONDX(I,INN2,IOLDCF)
     1           + BONDY(I,INN1,IOLDCF) * BONDY(I,INN2,IOLDCF)
     2           + BONDZ(I,INN1,IOLDCF) * BONDZ(I,INN2,IOLDCF))
     3           / (bondsq(I,INN1,IOLDCF)*bondsq(I,INN2,IOLDCF) )
               ry1=bondsq(I,INN1,IOLDCF)/ride1(INDOLD2b1)
               ry2=bondsq(I,INN2,IOLDCF)/ride1(INDOLD2b2)
               SCALP = RID31(INDOLD) * func3(ry1,ry2) *
     2                 (cos+0.3333333333)**2
               ANGTRM(I,IANGLE,IOLDCF) = SCALP
               uu1=uu1+scalp
               UINT = UINT + SCALP
 600  CONTINUE
C
      WRITE(f24,*) 'H3 = Somma scalp=',uu1 * TEMP * FACSYS
      RH3=uu1 * TEMP * FACSYS
      WRITE(f24,*) 'UINT now (H1+H2+H3) =',UINT * TEMP * FACSYS
      UOUT = UINT * TEMP * FACSYS
      WRITE(f24,*) 'INTERNAL ENERGY PER SITE AT START OF THE RUN: ',UOUT
      WRITE(f24,*) ' '
C
C SET STATISTICAL AVERAGES TO ZERO
C
      DO 980 I = 1,4
         UAV(I)   = 0.D0
         UMXAV(I) = 0.D0
```

```
         ORDAV(I) = 0.D0
         VAV(I)   = 0.D0
 980  CONTINUE
      DO 990 K = 1,4
         DO 990 J = 1,2
            DO 990 I = -1,1
               FAV(I,J,K) = 0.D0
 990  CONTINUE
      ACCAV = 0.D0
      ACVAV = 0.D0
C
C
C INITIALIZATION PART FINISHED
C
CZHU write parameters to the ist file
      write(f22,*)'Lx= ', LSIMPX
      write(f22,*)'Ly= ', LSIMPY
      write(f22,*)'Lz= ', LSIMPZ
      write(f22,*)'T= ', TEMP
      write(f22,*)'mu= ', UCHEM
      write(f22,*)'mus= ', SCHEM
      write(f22,*)'mStart= ', MSTART

C WRITE HEADLINE
C
c      WRITE(24,9100)
       WRITE(f22,9101)
C
C BEGIN MONTE CARLO PROCEDURE
C
      DO 8000 MCS = MCSINI,MCSMAX
C
         CALL RANDOM(NRAND)
CZHU     recording random numbers
CZHU         do 2003 I=1,NRAND
CZHU    write(f29,2002) (MCS-1)*NRAND+I, IRAN(I)
CZHU2003  enddo
CZHU2002     format(I15,5x, I15)
C
C FOR EACH PARTICLE, GENERATE A NEW POINT IN CONFIGURATION SPACE
C AND CALCULATE SITE CONTRIBUTION TO ENERGY DIFFERENCE
C
         DXMAX = PMOVX * PERIX
         DYMAX = PMOVY * PERIY
         DZMAX = PMOVZ * PERIZ
```

```
          FNX   = DXMAX * FNORM2
          FNY   = DYMAX * FNORM2
          FNZ   = DZMAX * FNORM2
C
          DO I = 1,NDIAM
             delta1(i)=0.0
             delta2(i)=0.0
             delta3(i)=0.0
             delta4(i)=0.0
          ENDDO
          DO 1000 I = 1,NDIAM
c              XCOORD(I,INEWCF) = XCOORD(I,IOLDCF)
c              YCOORD(I,INEWCF) = YCOORD(I,IOLDCF)
c              ZCOORD(I,INEWCF) = ZCOORD(I,IOLDCF)
c              ISTNEW=ISTAT(I,IOLDCF)
             XCOORD(I,INEWCF) = XCOORD(I,IOLDCF)
     &                        + FNX * IRAN(I) - DXMAX
             YCOORD(I,INEWCF) = YCOORD(I,IOLDCF)
     &                        + FNY * IRAN(I + NDIAM) - DYMAX
             ZCOORD(I,INEWCF) = ZCOORD(I,IOLDCF)
     &                        + FNZ * IRAN(I + NDIAM2) - DZMAX
             ISTNEW           = INT(FNSTAT * IRAN(I + NDIAM3)) - 1

             ioi=0
454          if (ISTNEW.eq.0)then
         if(FNSTAT*IRAN(I+NDIAM3)>1.5D0)then
  ISTNEW=1
else
  ISTNEW=-1
endif
             endif
             ISTAT(I,INEWCF)  = ISTNEW
CZHU
CCHEM     //using staggered Mu and uniform Mu
    IF(I<=NHALF)THEN
      DELTAE(I)= - (UCHEM+SCHEM)*(ISTNEW-ISTAT(I,IOLDCF))
    ELSE
               DELTAE(I)= - (UCHEM-SCHEM)*(ISTNEW-ISTAT(I,IOLDCF))
CZHU       DELTAE(I)        = CHEM(ISTNEW) - CHEM(ISTAT(I,IOLDCF))
    ENDIF
CCHEM             delta1(i)= - CHEM(ISTNEW) + CHEM(ISTAT(I,IOLDCF))
 1000    CONTINUE
C
          DO 3000 ISUBL = 1,8
C
```

```
            ILOW = (ISUBL - 1) * NSIMP + 1
            IHGH = ISUBL * NSIMP
C
C FIRST NEIGHBOR CONTRIBUTIONS
C
            DO 1250 INN = 1,4
               DO 1250 I = ILOW,IHGH
                  INEI   = NN(I,INN)
                  INDNEW = 5 + 3 * ISTAT(INEI,IOLDCF) + ISTAT(I,INEWCF)
                  INDBND(I,INN,INEWCF) = INDNEW
                  DDXX = XCOORD(INEI,IOLDCF) - XCOORD(I,INEWCF)
                  DDYY = YCOORD(INEI,IOLDCF) - YCOORD(I,INEWCF)
                  DDZZ = ZCOORD(INEI,IOLDCF) - ZCOORD(I,INEWCF)
                  DDXX = DDXX - PERIX * INT(FPERIX * DDXX)
                  DDYY = DDYY - PERIY * INT(FPERIY * DDYY)
                  DDZZ = DDZZ - PERIZ * INT(FPERIZ * DDZZ)
                  ddsq=sqrt(DDXX ** 2 + DDYY ** 2 + DDZZ ** 2)
                  ry=ddsq/ride1(INDNEW)
                  DD22 = EPS1(INDNEW) * func2(ry)
                  BONDX(I,INN,INEWCF) = DDXX
                  BONDY(I,INN,INEWCF) = DDYY
                  BONDZ(I,INN,INEWCF) = DDZZ
                  BOND2(I,INN,INEWCF) = DD22
                  bondsq(I,INN,INEWCF) = ddsq
                  DELTAE(I) = DELTAE(I) + DD22 - BOND2(I,INN,IOLDCF)
                  delta2(i)= DD22 - BOND2(I,INN,IOLDCF)
 1250       CONTINUE
C
C NOW, ANGLE CONTRIBUTIONS WHOSE VERTEX IS THE CENTRAL SITE
C
            DO 1350 INN1 = 1,3
               DO 1350 INN2 = INN1 + 1,4
                  IANGLE = IANTBL(INN2,INN1)
                  DO 1350 I = ILOW,IHGH
                     INEI1  = NN(I,INN1)
                     INEI2  = NN(I,INN2)
                     INDNEW = 14 + ISTAT(INEI1,IOLDCF)
     &                         +  9 * ISTAT(INEI2,IOLDCF)
     &                         +  3 * ISTAT(I,INEWCF)
                     INDNEW2b1 = 5 + 3 * ISTAT(INEI1,IOLDCF) +
     1                               ISTAT(I,INEWCF)
                     INDNEW2b2 = 5 + 3 * ISTAT(INEI2,IOLDCF) +
     1                               ISTAT(I,INEWCF)
                     INDANG(I,IANGLE,INEWCF) = INDNEW
                     cos=( BONDX(I,INN1,INEWCF) * BONDX(I,INN2,INEWCF)
```

```
     1                   + BONDY(I,INN1,INEWCF) * BONDY(I,INN2,INEWCF)
     2                   + BONDZ(I,INN1,INEWCF) * BONDZ(I,INN2,INEWCF))
     3                 / (bondsq(I,INN1,INEWCF)*bondsq(I,INN2,INEWCF) )
                    ry1=bondsq(I,INN1,INEWCF)/ride1(INDNEW2b1)
                    ry2=bondsq(I,INN2,INEwCF)/ride1(INDNEW2b2)
                    SCALP = RID31(INDNEW) * func3(ry1,ry2) *
     1                      (cos+0.3333333333)**2
                    ANGTRM(I,IANGLE,INEWCF) = SCALP
                    DELTAE(I) = DELTAE(I) + SCALP
     &                        - ANGTRM(I,IANGLE,IOLDCF)
                    delta3(i)= SCALP - ANGTRM(I,IANGLE,IOLDCF)
 1350       CONTINUE
c
C
C FINALLY, ANGLE CONTRIBUTIONS WHOSE VERTEX IS A NEIGHBOR SITE
C
            DO 1450 INN1 = 1,4
               DO 1450 INEXT = 1,3
                  INNN = (INN1 - 1) * 3 + INEXT
                  INN2 = IEXCL(INEXT,INN1)
                  IANGLE = IANTBL(INN2,INN1)
                  DO 1450 I = ILOW,IHGH
                     NEINN  = NN(I,INN1)
                     NEINNN = NNN(I,INNN)
                     INDNEW = 14 + 3 * ISTAT(NEINN,IOLDCF)
     &                         + 9 * ISTAT(NEINNN,IOLDCF)
     &                         +     ISTAT(I,INEWCF)
                     INDNEW2b1 = 5 + 3 * ISTAT(NEINN,IOLDCF) +
     1                         ISTAT(I,INEWCF)
                     INDOLD2b2 = 5 + 3 * ISTAT(NEINNN,IOLDCF) +
     1                         ISTAT(NEINN,IOLDCF)
                     IANTMP(I,INNN) = INDNEW
                     cos=-( BONDX(I,INN1,INEWCF)
     &                    * BONDX(NEINN,INN2,IOLDCF)
     &                    + BONDY(I,INN1,INEWCF)
     &                    * BONDY(NEINN,INN2,IOLDCF)
     &                    + BONDZ(I,INN1,INEWCF)
     &                    * BONDZ(NEINN,INN2,IOLDCF))
     1                    /(bondsq(I,INN1,INEWCF)*
     2                      bondsq(NEINN,INN2,IOLDCF))
                     ry1=bondsq(I,INN1,INEWCF)/ride1(INDNEW2b1)
                     ry2=bondsq(NEINN,INN2,IOLDCF)/ride1(INDOLD2b2)
                     SCALP = RID31(INDNEW) * func3(ry1,ry2) *
     2                 (cos+0.3333333333)**2
                     ANGTMP(I,INNN) = SCALP
```

```
                     DELTAE(I) = DELTAE(I) + SCALP
     &                           - ANGTRM(NEINN,IANGLE,IOLDCF)
                     delta4(i)= SCALP-ANGTRM(NEINN,IANGLE,IOLDCF)
 1450       CONTINUE
C
C ENERGY CALCULATION FINISHED
C
C FIND OUT WHICH OF THE MOVES IS ACCEPTED
C
            DO 1500 I = ILOW,IHGH
               PACC = EXP(- DELTAE(I) )
     &              - FNORM * IRAN(I + NDIAM4)
               IACC(I) = 0
               IF(PACC.GE.0.D0) IACC(I) = 1
 1500       CONTINUE
C
C UPDATE STATUS AND COORDINATES AT THE CENTRAL SITE
C
            DO 1510 I = ILOW,IHGH
               IF(IACC(I).EQ.1) THEN
                  ISTAT(I,IOLDCF)  = ISTAT(I,INEWCF)
                  XCOORD(I,IOLDCF) = XCOORD(I,INEWCF)
                  YCOORD(I,IOLDCF) = YCOORD(I,INEWCF)
                  ZCOORD(I,IOLDCF) = ZCOORD(I,INEWCF)
                  UINT             = UINT + DELTAE(I)
               END IF
 1510       CONTINUE
C
C UPDATE BONDS AT THE CENTRAL SITE
C
            DO 1520 INN = 1,4
               DO 1520 I = ILOW,IHGH
                  IF(IACC(I).EQ.1) THEN
                     INDBND(I,INN,IOLDCF) = INDBND(I,INN,INEWCF)
                     bondsq(I,INN,IOLDCF) = bondsq(I,INN,INEWCF)
                     BONDX(I,INN,IOLDCF)  = BONDX(I,INN,INEWCF)
                     BONDY(I,INN,IOLDCF)  = BONDY(I,INN,INEWCF)
                     BONDZ(I,INN,IOLDCF)  = BONDZ(I,INN,INEWCF)
                     BOND2(I,INN,IOLDCF)  = BOND2(I,INN,INEWCF)
                  END IF
 1520       CONTINUE
C
C UPDATE ANGLES AT THE CENTRAL SITE
C
            DO 1530 IANGLE = 1,6
```

```
               DO 1530 I = ILOW,IHGH
                  IF(IACC(I).EQ.1) THEN
                     INDANG(I,IANGLE,IOLDCF) = INDANG(I,IANGLE,INEWCF)
                     ANGTRM(I,IANGLE,IOLDCF) = ANGTRM(I,IANGLE,INEWCF)
                  END IF
 1530       CONTINUE
C
C UPDATE THE BONDS OF THE NEIGHBORING SITES
C
            DO 2500 INN = 1,4
C*VDIR: IGNORE RECRDEPS
CDIR$ IVDEP
               DO 2500 I = ILOW,IHGH
                  INEI = NN(I,INN)
                  INDBND(INEI,INN,IOLDCF) = INDBND(I,INN,IOLDCF)
                  BONDX(INEI,INN,IOLDCF)  = - BONDX(I,INN,IOLDCF)
                  BONDY(INEI,INN,IOLDCF)  = - BONDY(I,INN,IOLDCF)
                  BONDZ(INEI,INN,IOLDCF)  = - BONDZ(I,INN,IOLDCF)
                  BOND2(INEI,INN,IOLDCF)  = BOND2(I,INN,IOLDCF)
                  bondsq(INEI,INN,IOLDCF) = bondsq(I,INN,IOLDCF)
 2500       CONTINUE
C
C UPDATE THE ANGLES OF THE NEIGHBORING SITES
C
            DO 2600 INN1 = 1,4
               DO 2600 INEXT = 1,3
                  INNN = (INN1 - 1) * 3 + INEXT
                  INN2 = IEXCL(INEXT,INN1)
                  IANGLE = IANTBL(INN2,INN1)
C*VDIR: IGNORE RECRDEPS
CDIR$ IVDEP
                  DO 2600 I = ILOW,IHGH
                     INEI = NN(I,INN1)
                     PACC = IACC(I)
                     INDANG(INEI,IANGLE,IOLDCF) =
     &                  IACC(I) * IANTMP(I,INNN)
     &                + (1 - IACC(I)) * INDANG(INEI,IANGLE,IOLDCF)
                     ANGTRM(INEI,IANGLE,IOLDCF) =
     &                  PACC * ANGTMP(I,INNN)
     &                + (1.D0 - PACC) * ANGTRM(INEI,IANGLE,IOLDCF)
 2600       CONTINUE
C
 3000    CONTINUE
c
C
```

```
C LOOP OVER SUBLATTICES FINISHED
C
C NOW, PERFORM HOMOGENEOUS VOLUME FLUCTUATION
C
         PSXNEW = PERIX + FXBOX * IRAN(NDIAM5 + 1) - XBOX
         PSYNEW = PERIY + FYBOX * IRAN(NDIAM5 + 2) - YBOX
         PSZNEW = PERIZ + FZBOX * IRAN(NDIAM5 + 3) - ZBOX
         IF(PSXNEW.LE.0.D0.OR.PSYNEW.LE.0.D0.OR.PSZNEW.LE.0.D0) THEN
            WRITE(f24,*) 'BOX FLUCTUATIONS TOO LARGE !'
            WRITE(f24,*) 'CRASH AT MCS = ', MCS
            STOP
         END IF
         FACX = PSXNEW / PERIX
         FACY = PSYNEW / PERIY
         FACZ = PSZNEW / PERIZ
C
C ENERGY CALCULATION FOR VOLUME FLUCTUATION
C
         UBR0   = - NDIAM * LOG(FACX * FACY * FACZ)
         UBREAT = 0.D0
C
C NEAREST NEIGHBOR CONTRIBUTIONS. LOOP OVER SUBLATTICE 1 - 4
C
         DO 4500 INN = 1,4
            DO 4500 I = 1,NHALF
               INDOLD = INDBND(I,INN,IOLDCF)
               DDXX = FACX * BONDX(I,INN,IOLDCF)
               DDYY = FACY * BONDY(I,INN,IOLDCF)
               DDZZ = FACZ * BONDZ(I,INN,IOLDCF)
               BONDX(I,INN,INEWCF) = DDXX
               BONDY(I,INN,INEWCF) = DDYY
               BONDZ(I,INN,INEWCF) = DDZZ
               ddsq=sqrt(DDXX ** 2 + DDYY ** 2 + DDZZ ** 2)
               ry=ddsq/ride1(INDOLD)
               DD22 = EPS1(INDOLD) * func2(ry)
               BOND2(I,INN,INEWCF) = DD22
               bondsq(I,INN,INEWCF) = ddsq
               UBREAT = UBREAT + DD22 - BOND2(I,INN,IOLDCF)
 4500    CONTINUE
C
C UPDATE THE BONDS AT THE NEIGHBORING SITES OF SUBLATTICE 1 - 4
C
            DO 4600 INN = 1,4
C*VDIR: IGNORE RECRDEPS
CDIR$ IVDEP
```

```
               DO 4600 I = 1,NHALF
                  INEI = NN(I,INN)
                  BONDX(INEI,INN,INEWCF)  = - BONDX(I,INN,INEWCF)
                  BONDY(INEI,INN,INEWCF)  = - BONDY(I,INN,INEWCF)
                  BONDZ(INEI,INN,INEWCF)  = - BONDZ(I,INN,INEWCF)
                  BOND2(INEI,INN,INEWCF)  = BOND2(I,INN,INEWCF)
                  bondsq(INEI,INN,INEWCF) = bondsq(I,INN,INEWCF)
 4600       CONTINUE
C
C ANGLE CONTRIBUTIONS WHOSE VERTEX IS THE CENTRAL SITE
C
         DO 4800 INN1 = 1,3
            DO 4800 INN2 = INN1 + 1,4
               IANGLE = IANTBL(INN2,INN1)
               DO 4800 I = 1,NDIAM
                  INEI1 = NN(I,INN1)
                  INEI2 = NN(I,INN2)
                  INDOLD = INDANG(I,IANGLE,IOLDCF)
                  INDOLD2b1 = 5 + 3 * ISTAT(INEI1,IOLDCF) +
     1                        ISTAT(I,IOLDCF)
                  INDOLD2b2 = 5 + 3 * ISTAT(INEI2,IOLDCF) +
     1                        ISTAT(I,IOLDCF)
                  INDOLD1 = 14 + ISTAT(INEI1,IOLDCF)
     &                       +  9 * ISTAT(INEI2,IOLDCF)
     &                       +  3 * ISTAT(I,IOLDCF)
                  cos=( BONDX(I,INN1,INEWCF) * BONDX(I,INN2,INEWCF)
     1                + BONDY(I,INN1,INEWCF) * BONDY(I,INN2,INEWCF)
     2                + BONDZ(I,INN1,INEWCF) * BONDZ(I,INN2,INEWCF))
     3               / (bondsq(I,INN1,INEWCF)*bondsq(I,INN2,INEWCF) )
                  ry1=bondsq(I,INN1,INEWCF)/ride1(INDOLD2b1)
                  ry2=bondsq(I,INN2,INEWCF)/ride1(INDOLD2b2)
                  SCALP = RID31(INDOLD) * func3(ry1,ry2) *
     2                    (cos+0.3333333333)**2
                  ANGTRM(I,IANGLE,INEWCF) = SCALP
                  UBREAT = UBREAT + SCALP - ANGTRM(I,IANGLE,IOLDCF)
 4800    CONTINUE
c
C
C UPDATE
C
         ACV  = 0.D0
         PACC = EXP(-UBREAT-UBR0) - FNORM * IRAN(NRAND)
         IF(PACC.GT.0.D0) THEN
C
C ACCEPT. FIRST, ASSIGN NEW VALUES FOR THOSE ARRAYS
```

```
C FOR WHICH IT HAS NOT BEEN DONE YET
C
            DO 5000 I = 1,NDIAM
               XCOORD(I,INEWCF) = FACX * XCOORD(I,IOLDCF)
               YCOORD(I,INEWCF) = FACY * YCOORD(I,IOLDCF)
               ZCOORD(I,INEWCF) = FACZ * ZCOORD(I,IOLDCF)
               ISTAT(I,INEWCF)  = ISTAT(I,IOLDCF)
 5000       CONTINUE
            DO 5010 INN = 1,4
               DO 5010 I = 1,NDIAM
                  INDBND(I,INN,INEWCF) = INDBND(I,INN,IOLDCF)
 5010       CONTINUE
            DO 5020 IANGLE = 1,6
               DO 5020 I = 1,NDIAM
                  INDANG(I,IANGLE,INEWCF) = INDANG(I,IANGLE,IOLDCF)
 5020       CONTINUE
C
            ACV  = 1.D0
            UINT = UINT + UBREAT
C
C EXCHANGE POINTER TO OLD AND NEW CONFIGURATION
C
            ITMP   = IOLDCF
            IOLDCF = INEWCF
            INEWCF = ITMP
C
            PERIX = PSXNEW
            PERIY = PSYNEW
            PERIZ = PSZNEW
C
            FPERIX = 2.D0 / PERIX
            FPERIY = 2.D0 / PERIY
            FPERIZ = 2.D0 / PERIZ
C
         END IF
c
         UOUT = UINT * TEMP * FACSYS
C
C MEASURE SOME OBSERVABLES
C
         IF(MCS.GE.MSTART.OR.MCS.EQ.KOUTOB) THEN
C
C DETERMINE OCCUPATION FRACTIONS IN THE FCC SUBLATTICES
C AND ACCEPTANCE RATE
C
```

```
            IS11 = 0
            IS12 = 0
            IS21 = 0
            IS22 = 0
            JACC = 0
            DO 6010 I = 1,NHALF
               IS11 = IS11 + ISTAT(I,IOLDCF)
               IS12 = IS12 + ISTAT(I,IOLDCF) ** 2
               JACC = JACC + IACC(I)
 6010       CONTINUE
            DO 6020 I = NHALF + 1,NDIAM
               IS21 = IS21 + ISTAT(I,IOLDCF)
               IS22 = IS22 + ISTAT(I,IOLDCF) ** 2
               JACC = JACC + IACC(I)
 6020       CONTINUE
CZHU Calculate the extensive magnitization MEXT
    MEXT=IS11+IS21
    MEXTSD=IS11-IS21
CZHU   Calculate the Stillinger-Webber potential contribution
CCHEM
    HSW=UINT+MEXT*UCHEM+MEXTSD*SCHEM
            F1A = FACSYS * (IS12 + IS11)
            F1B = FACSYS * (IS12 - IS11)
            F1V = 1.D0 - F1A - F1B
            F2A = FACSYS * (IS22 + IS21)
            F2B = FACSYS * (IS22 - IS21)
            F2V = 1.D0 - F2A - F2B
            ACC = JACC * FACSYS
C
C INTERNAL ENERGY
C
            UOUT = UINT * TEMP * FACSYS
C
C ORDER PARAMETER FOR UNMIXING
C
            UMX = 0.5D0 * ( (F1A - F1B) + (F2A - F2B) )
            UMX = ABS(UMX)
C
C ORDER PARAMETER FOR SUPERLATTICE FORMATION
C
            ORD = 0.5D0 * ( (F1A - F1B) - (F2A - F2B) )
            ORD = ABS(ORD)
C
C SPECIFIC VOLUME
C
```

```
            VOLUME = PERIX * PERIY * PERIZ * FACSYS
C
         END IF
C
C CUMULATE AVERAGES
C
         IF(MCS.GE.MSTART) THEN
            DO 7000 MOM = 1,4
               UAV(MOM)      = UAV(MOM)      + UOUT ** MOM
               FAV(-1,1,MOM) = FAV(-1,1,MOM) + F1B ** MOM
               FAV(0,1,MOM)  = FAV(0,1,MOM)  + F1V ** MOM
               FAV(1,1,MOM)  = FAV(1,1,MOM)  + F1A ** MOM
               FAV(-1,2,MOM) = FAV(-1,2,MOM) + F2B ** MOM
               FAV(0,2,MOM)  = FAV(0,2,MOM)  + F2V ** MOM
               FAV(1,2,MOM)  = FAV(1,2,MOM)  + F2A ** MOM
               UMXAV(MOM)    = UMXAV(MOM)    + UMX ** MOM
               ORDAV(MOM)    = ORDAV(MOM)    + ORD ** MOM
               VAV(MOM)      = VAV(MOM)      + VOLUME ** MOM

 7000       CONTINUE
            ACCAV = ACCAV + ACC
            ACVAV = ACVAV + ACV
         END IF
C   END OF CUMULATE AVERAGE
CZHUC
C     OUTPUT OBSERVABLES

         IF(MCS.EQ.KOUTOB) THEN
            KOUTOB = KOUTOB + NOUTOB
CZHU            WRITE(24,9000) MCS,F1A,F1B,F1V,F2A,F2B,F2V
            ffa=F1A+F2A
            ffb=F1B+F2B
            ffv=F1V+F2V
CZHU            WRITE(22,9001) MCS,ACC,UOUT,FFA,FFB,FFV
CZHU
CZHU  recording the max and min values of HSW
CZHU  It is inevitable to discard the beginning part of data
CZHU  So start comparing from step #MSTART
CZHU  But instantaneous recording still starts from #1
CZHU
   WRITE(f22,770)MCS,MEXT,MEXTSD,HSW
         END IF
C
C DUMP CONFIGURATION
C
```

```
          IF(MCS.EQ.KOUTCF) THEN
             KOUTCF = KOUTCF + NOUTCF
             REWIND(f21)
             WRITE(f21) (IRWRK(I),I=1,MERS),
     &                  (XCOORD(I,IOLDCF),I=1,NDIAM),
     &                  (YCOORD(I,IOLDCF),I=1,NDIAM),
     &                  (ZCOORD(I,IOLDCF),I=1,NDIAM),
     &                  (ISTAT(I,IOLDCF),I=1,NDIAM),
     &                  PERIX,PERIY,PERIZ
             REWIND(f27)
             do  I=1,NDIAM
             write(f27,732)XCOORD(I,IOLDCF),YCOORD(I,IOLDCF),
     &                  ZCOORD(I,IOLDCF),ISTAT(I,IOLDCF)
             enddo
             write(f27,*)PERIX,PERIY,PERIZ
             WRITE(f24,*) 'CONFIGURATION DUMPED AFTER MCS = ',MCS
          END IF
732       format(3(2x,f12.6),2x,I4)
C
C OUTPUT STATISTICAL AVERAGES
C
          IF(MCS.EQ.KOUTAV) THEN
             KOUTAV = KOUTAV + NOUTAV
             REWIND(f23)
CZHU         REWIND(33)
CZHU         REWIND(34)
C
             FACTOR = 1.D0 / (MCS - MSTART + 1)
             FACTOR1 = FACTOR * FACSYS
c
             DO 7100 MOM = 1,4
                UAVOUT(MOM) = UAV(MOM) * FACTOR
                UMXOUT(MOM) = UMXAV(MOM) * FACTOR
                ORDOUT(MOM) = ORDAV(MOM) * FACTOR
                VAVOUT(MOM) = VAV(MOM) * FACTOR
 7100        CONTINUE
             DO 7200 MOM = 1,4
                DO 7200 ISUBL = 1,2
                   DO 7200 I = -1,1
                      FAVOUT(I,ISUBL,MOM) = FAV(I,ISUBL,MOM) * FACTOR
 7200        CONTINUE
             ACCOUT = ACCAV * FACTOR
             ACVOUT = ACVAV * FACTOR
C
             SPEC   = NDIAM * (BETA ** 2) * (UAVOUT(2) - UAVOUT(1) ** 2)
```

```
            COMPRS = NDIAM * BETA * (VAVOUT(2) - VAVOUT(1) ** 2)
            SUSUMX = NDIAM * BETA * (UMXOUT(2) - UMXOUT(1) ** 2)
            SUSORD = NDIAM * BETA * (ORDOUT(2) - ORDOUT(1) ** 2)
            CUMUMX = 0.D0
            CUMORD = 0.D0
    CUMUMXF= 0.D0
            CUMORDF= 0.D0
    IF(UMXOUT(2).NE.0.D0)
     &        CUMUMX = 1.D0 - UMXOUT(4) / (3.D0 * UMXOUT(2) ** 2)
    IF(ORDOUT(2).NE.0.D0)
     &        CUMORD = 1.D0 - ORDOUT(4) / (3.D0 * ORDOUT(2) ** 2)

CZHU
CZHU     Added to use the full definition of cumulant
CZHU     DENU means denumerator
CZHU
    UMXDENU  = (UMXOUT(2)-UMXOUT(1)**2)**2
      ORDDENU  = (ORDOUT(2)-ORDOUT(1)**2)**2
    IF(UMXDENU.NE.0D0)THEN
            CUMUMXF=UMXOUT(4)+6.D0*UMXOUT(2)*UMXOUT(1)**2
     &               -4.D0*UMXOUT(3)*UMXOUT(1)-3.D0*UMXOUT(1)**4
            CUMUMXF=1.D0-CUMUMXF/3.D0/UMXDENU
    ENDIF
            IF(ORDDENU.NE.0D0)THEN
      CUMORDF=ORDOUT(4)+6.D0*ORDOUT(2)*ORDOUT(1)**2
     &               -4.D0*ORDOUT(3)*ORDOUT(1)-3.D0*ORDOUT(1)**4
      CUMORDF=1.D0-CUMORDF/3.D0/ORDDENU
    ENDIF
C
            FACTOR = 1.D0 / (MCS - MSTART + 1)
            FACTOR1 = FACTOR * FACSYS
            concge=(FAVOUT(-1,1,1)+FAVOUT(-1,2,1))*0.5
            concsi=(FAVOUT(1,1,1)+FAVOUT(1,2,1))*0.5
    WRITE(f23,134) MCS
            WRITE(f23,234) ACCOUT
            WRITE(f23,334) ACVOUT
            WRITE(f23,*) ' '
CCHEM            WRITE(23,335) CHEM(1)*TEMP,CHEM(-1)*TEMP
            WRITE(f23,335) UCHEM*TEMP,SCHEM*TEMP
            WRITE(f23,*) 'TEMP = ',TEMP
            WRITE(f23,*) ' '
c
            WRITE(f23,*) ' '
            DO 7500 MOM = 1,4
               WRITE(f23,434) UAVOUT(MOM),MOM
```

```
 7500       CONTINUE
            WRITE(f23,*) ' '
            WRITE(f23,534) SPEC
            WRITE(f23,*) ' '
            DO 7550 MOM = 1,4
               WRITE(f23,634) VAVOUT(MOM),MOM
 7550       CONTINUE
            WRITE(f23,*) ' '
            WRITE(f23,734) COMPRS
            DO 7600 MOM = 1,4
            WRITE(f23,*) ' '
               DO 7600 ISUBL = 1,2
                  DO 7600 I = -1,1
                     WRITE(f23,834) I,ISUBL,MOM,FAVOUT(I,ISUBL,MOM)
 7600       CONTINUE
            WRITE(f23,*) ' '
            conctot1=(FAVOUT(1,1,1)+FAVOUT(1,2,1))/2.0
            conctot2=(FAVOUT(-1,1,1)+FAVOUT(-1,2,1))/2.0
            WRITE(f23,835)conctot1
            WRITE(f23,836)conctot2
            WRITE(f23,*) ' '

            DO 7700 MOM = 1,4
               WRITE(f23,934) MOM,UMXOUT(MOM)
 7700       CONTINUE
            WRITE(f23,1134) SUSUMX
            WRITE(f23,1234) CUMUMX
    WRITE(f23,1235) CUMUMXF
            WRITE(f23,*) ' '
            DO 7750 MOM = 1,4
               WRITE(f23,1334) MOM,ORDOUT(MOM)
 7750       CONTINUE
            WRITE(f23,1434) SUSORD
            WRITE(f23,1534) CUMORD
    WRITE(f23,1535) CUMORDF
         END IF
C
 8000 CONTINUE
call MPI_FINALIZE(ierror)
C
C LOOP OVER MONTE CARLO STEPS FINISHED
C
C
134      FORMAT('STATISTICAL AVERAGES AFTER MCS = ',I6)
234      FORMAT('AVERAGE ACCEPTACE RATE = ',F15.11)
```

```
334      FORMAT('AVERAGE VOLUME FLUCTUATION ACC. R. = ',F15.9)
CCHEM335      FORMAT('CHEM(1) = ',F15.6,'   CHEM(-1) =',F15.6)
335      FORMAT('Uniform CHEM = ',F15.6,'   Staggered CHEM =',F15.6)
561      FORMAT('DISTANCE Si-Si  = ',F15.6)
562      FORMAT('DISTANCE Si-Ge  = ',F15.6)
563      FORMAT('DISTANCE Ge-Ge  = ',F15.6)
564      FORMAT('LATTICE CONST.  = ',F15.6)
544      FORMAT('SIDES = ',F15.8,1x,F15.8,1x,F15.8,1x)
565      FORMAT('NUMBER OF Si-Si = ',F15.6)
566      FORMAT('NUMBER OF Si-Ge = ',F15.6)
567      FORMAT('NUMBER OF Ge-Ge = ',F15.6)
461      FORMAT(6(1x,F8.6))
462      FORMAT(7(1x,F8.6))
568      FORMAT('Si-Si-Si = ',F15.6,2x,F15.6)
569      FORMAT('Ge-Si-Ge = ',F15.6,2x,F15.6)
570      FORMAT('Si-Si-Ge = ',F15.6,2x,F15.6)
571      FORMAT('Si-Ge-Si = ',F15.6,2x,F15.6)
572      FORMAT('Ge-Ge-Ge = ',F15.6,2x,F15.6)
573      FORMAT('Si-Ge-Ge = ',F15.6,2x,F15.6)
574      FORMAT('N. Si-Si-Si = ',F15.6,2x,E15.6)
575      FORMAT('N. Ge-Si-Ge = ',F15.6,2x,E15.6)
576      FORMAT('N. Si-Si-Ge = ',F15.6,2x,E15.6)
577      FORMAT('N. Si-Ge-Si = ',F15.6,2x,E15.6)
578      FORMAT('N. Ge-Ge-Ge = ',F15.6,2x,E15.6)
579      FORMAT('N. Si-Ge-Ge = ',F15.6,2x,E15.6)

434      FORMAT('INTERNAL ENERGY = ',F15.6,'   MOMENT =',I2)
534      FORMAT('SPECIFIC HEAT = ',F15.9)
634      FORMAT('SPECIFIC VOLUME = ',F15.9,'   MOMENT =',I2)
734      FORMAT('COMPRESSIBILITY = ',F15.9)
834      FORMAT('FRACTION OF ATOMS',I3,' IN SUBLATTICE ',I2,
     *        ' ,MOMENT',I2,' = ',F15.6)
835      FORMAT('CONC. OF ATOMS 1, MOMENT 1, = ',F15.9)
836      FORMAT('CONC. OF ATOMS -1,MOMENT 1, =',F15.9)
837  FORMAT('ORDER PARAMETER =',F15.9)
838      FORMAT('ORDER PARAMETER POWER2=',F15.9)
839      FORMAT('ORDER PARAMETER POWER4=',F15.9)
840      FORMAT('ORDER PARAMETER U4=',F15.9)

934      FORMAT('ORDER PARAMETER UNMIXING, MOMENT',I2,' = ',F15.6)
1134     FORMAT('SUSCEPTIBILITY UNMIXING = ',F15.9)
1234     FORMAT('CUMULANT UNMIXING = ',F15.9)
1235     FORMAT('CUMULANT UNMIXING FULLY DEFINED = ',F15.9)
1334     FORMAT('ORDER PARAMETER SUPERLATTICE, MOMENT',I2,' = ',F15.6)
1434     FORMAT('SUSCEPTIBILITY SUPERLATTICE = ',F15.9)
```

```
1534     FORMAT('CUMULANT SUPERLATTICE = ',F15.9)
1535     FORMAT('CUMULANT SUPERLATTICE FULLY DEFINED = ',F15.9)
444   format(I6,1x,3(F10.6,1x,I6),1x,I6)
445   format(I6,1x,4(F13.7,1x))
761   format(4(f9.5,1x,f8.6,1x))
762   format(f8.4,1x,6(f10.4,1x))
770   format(I10, 1x,I8,2x,I8,2x,e24.18)


 9000 FORMAT(I6,1x,6(f10.6,1x))
 9001 FORMAT(I8,6(E13.6,1x))
 9101 FORMAT(4X,'MCS',2X,
     &       3X,'MEXT',2X,
     &       2X,'MEXTSD',1X,
     &       9X,'HSW',5X)
 9100 FORMAT(2X,'MCS',3X,
     &       2X,'FRC. A SL 1',2X,
     &       2X,'FRC. B SL 1',2X,
     &       2X,'FRC. V SL 1',2X,
     &       2X,'FRC. A SL 2',2X,
     &       2X,'FRC. B SL 2',2X,
     &       2X,'FRC. V SL 2',2X)
C
      STOP
      END
c
c
      Function func2(ry)
c
      DOUBLE PRECISION func2,ry
      DOUBLE PRECISION costA,costB,costp,costbb
c
      costA=7.049556277
      costB=0.6022245584
      costp=4.0
      costbb=1.80
c
      if (ry.lt.costbb) then
         func2=costA*((costB/ry**costp)-1.0)* exp(1.0/(ry-costbb))
      else
         func2=0.0
      endif
c
      return
      end
```

```
c
c
      Function func3(ry1,ry2)
c
      DOUBLE PRECISION func3,ry1,ry2
      DOUBLE PRECISION costbb,costg
c
      costbb=1.80
      costg=1.20
c
      if ((ry1.lt.costbb).and.(ry2.lt.costbb)) then
         func3=exp((costg/(ry1-costbb))+(costg/(ry2-costbb)))
      else
         func3=0.0
      endif
c
      return
      end
c
c
c@PROCESS DIRECTIVE ('*VDIR:')
      SUBROUTINE INIRAN(ISEED)
C
C STARTS THE RANDOM NUMBER GENERATOR
C
      PARAMETER(LSIMPX=24)
      PARAMETER(LSIMPY=24)
      PARAMETER(LSIMPZ=24)
      PARAMETER(NRAND=40*LSIMPX*LSIMPY*LSIMPZ+4)
C
C SPECIFICATIONS FOR THE TAUSWORTHE GENERATOR
C
      PARAMETER(MERS=1279)
      PARAMETER(MERS1=1063)
      PARAMETER(NBIT=32)
C
      INTEGER ISEED
      INTEGER IRTOT(MERS + NRAND)
      COMMON/RANCOM/IRTOT
      DOUBLE PRECISION RMOD, PMOD, PMULT, FACTOR
C
C SPECIFICATIONS FOR THE MODULO GENERATOR
C
      PMOD  = 2147483647.D0
      PMULT = 16807.D0
```

```
C
      RMOD = ISEED
C
C IBIG IS LARGEST INTEGER ON A NBIT MACHINE
C
      IBIG = 2 ** (NBIT - 2)
      IHLP = IBIG - 1
      IBIG = IBIG + IHLP
C
C WARMING UP THE MODULO GENERATOR
C
      DO 10 I = 1,NRAND
         RMOD = PMULT * RMOD
         RMOD = RMOD - INT(RMOD / PMOD) * PMOD
         RMOD = INT(RMOD + 0.1D0)
 10   CONTINUE
C
C PUT RANDOM NUMBERS ON THE WORKING ARRAY
C
      DO 20 I = 1,MERS
         RMOD     = PMULT * RMOD
         RMOD     = RMOD - INT(RMOD / PMOD) * PMOD
         IRTOT(I) = INT(RMOD + 0.1D0)
         RMOD     = IRTOT(I)
 20   CONTINUE
C
C MAYBE, THERE ARE MORE THAN 32 BITS AVAILABLE
C AND WE WANT TO USE THEM
C
      IF(NBIT.NE.32) THEN
         FACTOR = IBIG / PMOD
         DO 30 I = 1,MERS
            IRTOT(I) = IRTOT(I) * FACTOR
 30      CONTINUE
      END IF
C
C LINEAR INDEPENDENCE
C PUT 1'S ON THE MAIN DIAGONAL
C AND 0'S ABOVE IT
C
      IMASK1 = 1
      IMASK2 = IBIG
      DO 40 I = NBIT - 1,2,-1
C--------IBM
         IRTOT(I) = IAND(IOR(IRTOT(I),IMASK1),IMASK2)
```

```
         IMASK2   = IEOR(IMASK2,IMASK1)
C--------CRAY
CC       IRTOT(I) = AND(OR(IRTOT(I),IMASK1),IMASK2)
CC       IMASK2   = XOR(IMASK2,IMASK1)
         IMASK1   = IMASK1 * 2
 40   CONTINUE
      IRTOT(1) = IMASK1
C
C WARM UP THE TAUSWORTHE GENERATOR
C
      CALL RANDOM(NRAND)
C
      RETURN
      END
C@PROCESS DIRECTIVE ('*VDIR:')
      SUBROUTINE RANDOM(N)
C
C STORES N (AT MOST NRAND) UNNORMALIZED INTEGER RANDOM NUMBERS
C
      PARAMETER(LSIMPX=24)
      PARAMETER(LSIMPY=24)
      PARAMETER(LSIMPZ=24)
      PARAMETER(NRAND=40*LSIMPX*LSIMPY*LSIMPZ+4)
C
      PARAMETER(MERS=1279)
      PARAMETER(MERS1=1063)
C
      INTEGER N
      INTEGER IRTOT(MERS + NRAND)
      COMMON/RANCOM/IRTOT
C
      NCYC  = N / MERS1
      NREST = N - MERS1 * NCYC
      IBAS1 = 0
      IBAS2 = MERS - MERS1
      IBAS3 = MERS
C
      DO 100 ICYC = 1,NCYC
C--------IBM
C*VDIR: IGNORE RECRDEPS
         DO 10 I = 1,MERS1
            IRTOT(IBAS3 + I) = IEOR(IRTOT(IBAS1 + I),IRTOT(IBAS2 + I))
 10      CONTINUE
C--------CRAY
CDIR$ IVDEP
```

```
CC       DO 10 I = 1,MERS1
CC          IRTOT(IBAS3 + I) = XOR(IRTOT(IBAS1 + I),IRTOT(IBAS2 + I))
C10      CONTINUE
         IBAS1 = IBAS1 + MERS1
         IBAS2 = IBAS2 + MERS1
         IBAS3 = IBAS3 + MERS1
 100  CONTINUE
C
      IF(NREST.GT.0) THEN
C--------IBM
C*VDIR: IGNORE RECRDEPS
         DO 110 I = 1,NREST
            IRTOT(IBAS3 + I) = IEOR(IRTOT(IBAS1 + I),IRTOT(IBAS2 + I))
 110     CONTINUE
C--------CRAY
CDIR$ IVDEP
CC       DO 110 I = 1,NREST
CC          IRTOT(IBAS3 + I) = XOR(IRTOT(IBAS1 + I),IRTOT(IBAS2 + I))
C110     CONTINUE
      END IF
C
C PUT LAST ELEMENTS TO THE BEGINNING
C
C-----IBM AND CRAY
C*VDIR: IGNORE RECRDEPS
CDIR$ IVDEP
      DO 200 I = 1,MERS
         IRTOT(I) = IRTOT(N + I)
 200  CONTINUE
C
      RETURN
      END
```

## A.2 The Input File input.dat

```
0               0 MEANS BULK SIMULATION, 1 FREE SURFACE
1               INITIAL VALUE FOR TIME MCS (USUALLY 1)
55000           FINAL VALUE FOR TIME MCS
5000            TIME INTERVAL FOR DUMPING CONFIGURATIONS
1               TIME INTERVAL FOR OUTPUT OBSERVABLES
1               FIRST MCS FOR AVERAGING
5000            TIME INTERVAL FOR OUTPUT STATISTICAL AVERAGES
730151709       ISEED
0.005           MAXIMUM TRIAL MOVE
0.001           MAXIMUM RELATIVE TRIAL MOVE IN BOX SIZE
 0.312          TEMPERATURE (ALL UNITS IN EV)
 2.200000048
 2.599999905
 2.200000048
 2.599999905
 2.200000048
 2.599999905
 5.429999828
 5.519999981
0               Uniformed CHEMICAL POTENTIAL
0               Staggered CHEMICAL POTENTIAL
2.17            BOND ENERGY A - A
1.93            BOND ENERGY B - B
2.3427          BOND ENERGY A - B
2.34779         BOND LENGTH A - A
2.44598         BOND LENGTH B - B
2.396885        BOND LENGTH A - B
21.0            lambda(1)
31.0            lambda(-1)
```

## A.3 Shell Script

Below is the simple shell script used to run the job. An advanced version is available in the CDROM.

```
#!/bin/sh
   ln -s    posout_prev     fort.28
   ln -s   posout1          fort.27
   ln -s    posout          fort.21
   ln -s       ist          fort.22
   ln -s     medie          fort.23
   ln -s    output          fort.24

# assume the executable is SW_retNFL.x,
# and the input file is inp_SW.dat.
   /usr/bin/time -p ./SW_retNFL.x <inp_SW.dat >> output_file
```

# Appendix B

## Programs for the Compressible Stacked Triangular Ising System

### B.1 The FORTRAN Code

```
      PROGRAM ISING_LENNARD_JONES
C AUTHOR: N. S. Branco
C VERSION OCTOBER 04, 2000 - 14:00
C FOR IBM ES / 9000
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C MONTE CARLO SIMULATION OF COMPRESSIBLE ISING MODEL
C WITH THE LENNARD-JONES POTENCIAL
C AT CONSTANT VOLUME
C THE STACKED TRIANGULAR LATTICE IS SET UP AS 8 INTERPENETRATING
C TILTED SIMPLE CUBIC LATTICES
C THE 8 SUBLATTICES ARE INDEPENDENT; VECTORIZATION
C BY STANDARD CHECKERBOARD METHOD
C FOR NEIGHBOR TABLE SETUP IN THE BEGINNING,
C THE SC SUBLATTICE HAS LATTICE CONSTANT 2
C NUMBER OF PARTICLES IS NDIAM
c NESTA VERSAO, A PARTICULA SO PODE SE MOVER DENTRO DE
C UMA ESFERA DE RAIO AO QUADRADO DELTA2 EM TORNO DE SUA
C POSICAO EM T=0 E SEUS MOVIMENTOS TEM AMPLITUDE MAXIMA
C PMOV EM CADA UMA DAS 3 DIRECOES
C NESTA VERSAO USAMOS O GNA DA SHAN-HO
C
C EACH SITE NEEDS 5 RANDOM NUMBERS:
C 3 FOR NEW COORDINATES
C 1 FOR NEW SPIN
C 1 FOR ACCEPTANCE
C
C
C BEGINNING OF DECLARATIONS
C
C   IMPORTANT: LSIMP? MUST BE GREATER THAN 2 AND EVEN
```

```
C
      PARAMETER(LSIMPX=64)
      PARAMETER(LSIMPY=64)
      PARAMETER(LSIMPZ=64)
C
      PARAMETER(NDIAM=LSIMPX*LSIMPY*LSIMPZ)
C
      PARAMETER(FACSYS=1.D0/NDIAM)
C
      PARAMETER(NDIAM2=2*NDIAM)
      PARAMETER(NDIAM3=3*NDIAM)
      PARAMETER(NDIAM4=4*NDIAM)
      PARAMETER(NDIAM5=5*NDIAM)
C
      PARAMETER(NRAND=NDIAM5)
C
C DECLARATION FOR THE RANDOM GENERATOR
C
      DOUBLE PRECISION IRAN(NRAND)
C
C SPINS AND COORDINATES OF THE ATOMS
C
C
      INTEGER ISTAT(NDIAM,2)
      DOUBLE PRECISION XCOORD(NDIAM,2),XCOORD0(NDIAM)
      DOUBLE PRECISION YCOORD(NDIAM,2),YCOORD0(NDIAM)
      DOUBLE PRECISION ZCOORD(NDIAM,2),ZCOORD0(NDIAM)
      DOUBLE PRECISION RO2(NDIAM)
C
C BIG SIMPLE CUBIC LATTICE FOR NEIGHBOR LIST SETUP
C
      INTEGER LABEL(LSIMPX,LSIMPY,LSIMPZ)
C
C NEIGHBOR TABLE
C
      INTEGER NN(NDIAM,8)
C
C IN THIS ARRAY WE STORE THE ENERGY OF EACH BOND
C
      DOUBLE PRECISION BOND2(NDIAM,8,2)
C
C ENERGY DIFFERENCE
C
      DOUBLE PRECISION DELTAE(NDIAM)
C
```

```
C ACCEPTANCE POINTER
C
      INTEGER IACC(NDIAM)
C
C HAMILTONIAN PARAMETERS
C
      DOUBLE PRECISION Um,Q,TEMP,R02,DELTA2,SIN60,PMOV
C
C TRANSLATION VECTORS TO THE SUBLATTICES
C
      INTEGER IOFFX(8)
      INTEGER IOFFY(8)
      INTEGER IOFFZ(8)
C
C JUMP VECTORS TO THE NEAREST NEIGHBORS, STARTING
C FROM SITES OF THE FIRST CUBIC SUBLATTICE
C
      INTEGER JUMPX(8), JUMPY(8), JUMPZ(8)
C
C STATISTICAL AVERAGES
C
      DOUBLE PRECISION UAV(4),MAV(4),MUAV(4),ACCAV
C
      DOUBLE PRECISION UAVOUT(4),MAVOUT(4),MUAVOUT(4),ACCOUT

CZHU
CZHU  parallelizing the code
CZHU  initialization
CZHU
      integer myID, ierror,numprocs,f21,f22,f23,f27,f28,f24,ioffset

      myID=0
      CALL MPI_INIT(ierror)
      CALL MPI_COMM_RANK(MPI_COMM_WORLD,myID,ierror)
      CALL MPI_COMM_SIZE(MPI_COMM_WORLD,numprocs, ierror)
      ioffset=(myID-2)*10
      f21=21+ioffset
      f22=22+ioffset
      f23=23+ioffset
      f27=27+ioffset
      f28=28+ioffset
      f24=24+ioffset

      WRITE(f24,*) 'My process ID is ',myID
      WRITE(f24,*) 'After MPI call. Next, read input'
```

```
C
CZHU      OPEN(3,FILE='tricons6.in')
C
      SIN60=DSIN(1.0471975511965977D0)
      DELTA2 = 0.183D0**2
C
C DECLARATION PART FINISHED
C
C
      WRITE(f24,*) 'TOTAL NUMBER OF SIMULATED SITES = ',NDIAM
C
      WRITE(f24,*) 'INITIAL VALUE FOR TIME MCS (USUALLY 1) = ?'
      READ(5,*) MCSINI
      WRITE(f24,*) MCSINI
C
      WRITE(f24,*) 'FINAL VALUE FOR TIME MCS = ?'
      READ(5,*) MCSMAX
      WRITE(f24,*) MCSMAX
C
      WRITE(f24,*) 'TIME INTERVAL FOR DUMPING CONFIGURATIONS = ?'
      READ(5,*) NOUTCF
      WRITE(f24,*) NOUTCF
      KOUTCF = NOUTCF + MCSINI - 1
C
      WRITE(f24,*) 'TIME INTERVAL FOR OUTPUT OBSERVABLES = ?'
      READ(5,*) NOUTOB
      WRITE(f24,*) NOUTOB
      KOUTOB = NOUTOB + MCSINI - 1
C
      WRITE(f24,*) 'START AVERAGING AT MCS = ?'
      READ(5,*) MSTART
      WRITE(f24,*) MSTART
C
      WRITE(f24,*) 'TIME INTERVAL FOR OUTPUT STATISTICAL AVERAGES = ?'
      READ(5,*) NOUTAV
      WRITE(f24,*) NOUTAV
      KOUTAV = NOUTAV + MSTART - 1
C
      WRITE(f24,*) 'ISEED = ?'
      READ(5,*) ISEED
      WRITE(f24,*) 'Original ISEED = ', ISEED
      ISEED=ISEED+947582*myID
      WRITE(f24,*) 'ISEED = ',ISEED
C
```

```
      WRITE(f24,*) 'MAXIMUM DISTANCE FOR ATTEMPTED MOVE = ?'
      READ(5,*) PMOV
      WRITE(f24,*) PMOV
C
      WRITE(f24,*) 'TEMPERATURE (IN UNITS OF Um) = ?'
      READ(5,*) TEMP
      WRITE(f24,*) TEMP
      Um = 1.D0/TEMP
C
C HAMILTONIAN PARAMETERS
C
C
      WRITE(f24,*) 'ELASTIC OVER MAGNETIC ENERGY = ?'
      READ(5,*) Q
      WRITE(f24,*) Q
C
      WRITE(f24,*) 'SQUARE OF NEAREST NEIGHBOR DISTANCE= ?'
      READ(5,*) R02
      WRITE(f24,*) R02
C
C INITIAL CONFIGURATION: EACH FCC SUBLATTICE IN STATE +1
C AND PERFECTLY ORDERED
C DIAMOND LATTICE.
C
      UOUT =  - 4.D0 * (Q+1.D0)
      WRITE(f24,*) 'INITIAL SETTING:'
      WRITE(f24,*) 'SIMULATION WILL START IN THE ORDERED STATE '
      WRITE(f24,*) 'AT INTERNAL ENERGY PER SITE (IN UNITS OF Um):',UOUT
      WRITE(f24,*) 'AT MAGNETIZATION: 1.'
C
C DEFINE SHIFT VECTORS TO THE ORIGINS OF THE
C VARIOUS SUBLATTICES
C
      IOFFX(1) = 0
      IOFFY(1) = 0
      IOFFZ(1) = 0
C
      IOFFX(2) = 1
      IOFFY(2) = 0
      IOFFZ(2) = 0
C
      IOFFX(3) = 0
      IOFFY(3) = 0
      IOFFZ(3) = 1
C
```

```
      IOFFX(4) = 1
      IOFFY(4) = 0
      IOFFZ(4) = 1
C
      IOFFX(5) = 0
      IOFFY(5) = 1
      IOFFZ(5) = 0
C
      IOFFX(6) = 1
      IOFFY(6) = 1
      IOFFZ(6) = 0
C
      IOFFX(7) = 0
      IOFFY(7) = 1
      IOFFZ(7) = 1
C
      IOFFX(8) = 1
      IOFFY(8) = 1
      IOFFZ(8) = 1
C
C SET LABEL TO ZERO
C
      DO 17 IZ = 1,LSIMPZ
         DO 17 IY = 1,LSIMPY
            DO 17 IX = 1,LSIMPX
               LABEL(IX,IY,IZ) = 0
 17   CONTINUE
C
C FILL THE LATTICE WITH PARTICLE INDICES IN THE FOLLOWING ORDER:
C SUBLATTICE 1
C SUBLATTICE 2
C ETC.
C
      IPART = 0
      DO 20 ISUBL = 1,8
         DO 20 IZ = 1 + IOFFZ(ISUBL),LSIMPZ + IOFFZ(ISUBL),2
            DO 20 IY = 1 + IOFFY(ISUBL),LSIMPY + IOFFY(ISUBL),2
               DO 20 IX = 1 + IOFFX(ISUBL),LSIMPX + IOFFX(ISUBL),2
                  IPART           = IPART + 1
                  LABEL(IX,IY,IZ) = IPART
 20   CONTINUE
C
C DEFINE THE JUMP VECTORS TO THE NEAREST NEIGHBORS
C
      JUMPX(1) = 1
```

```
      JUMPY(1) = 0
      JUMPZ(1) = 0
C
      JUMPX(2) = 0
      JUMPY(2) = 1
      JUMPZ(2) = 0
C
      JUMPX(3) = -1
      JUMPY(3) = 1
      JUMPZ(3) = 0
C
      JUMPX(4) = 0
      JUMPY(4) = 0
      JUMPZ(4) = 1
C
      JUMPX(5) = 0
      JUMPY(5) = 0
      JUMPZ(5) = -1
C
      JUMPX(6) = 0
      JUMPY(6) = -1
      JUMPZ(6) = 0
C
      JUMPX(7) = -1
      JUMPY(7) = -1
      JUMPZ(7) = 0
C
      JUMPX(8) = -1
      JUMPY(8) = 0
      JUMPZ(8) = 0
C
C SET UP TABLE OF NEAREST NEIGHBORS
C
      DO 120 INN = 1,8
         MOVX = JUMPX(INN)
         MOVY = JUMPY(INN)
         MOVZ = JUMPZ(INN)
         DO 120 ISUBL = 1,8
    IF(ISUBL.EQ.5) THEN
               IF (INN.EQ.2.OR.INN.EQ.6) MOVX = 1
               IF (INN.EQ.3.OR.INN.EQ.7) MOVX = 0
            END IF
            DO 120 IZ = 1 + IOFFZ(ISUBL),LSIMPZ + IOFFZ(ISUBL),2
               IZNEW = IZ + MOVZ
               IF(IZNEW.LT.1) THEN
```

```
                  IZNEW = IZNEW + LSIMPZ
               END IF
               IF(IZNEW.GT.LSIMPZ) THEN
                  IZNEW = IZNEW - LSIMPZ
               END IF
               DO 120 IY = 1 + IOFFY(ISUBL),LSIMPY + IOFFY(ISUBL),2
                  IYNEW = IY + MOVY
                  IF(IYNEW.LT.1) IYNEW = IYNEW + LSIMPY
                  IF(IYNEW.GT.LSIMPY) IYNEW = IYNEW - LSIMPY
                  DO 120 IX = 1 + IOFFX(ISUBL),LSIMPX + IOFFX(ISUBL),2
                     IXNEW = IX + MOVX
                     IF(IXNEW.LT.1) IXNEW = IXNEW + LSIMPX
                     IF(IXNEW.GT.LSIMPX) IXNEW = IXNEW - LSIMPX
                     IPART = LABEL(IX,IY,IZ)
                     NN(IPART,INN) = LABEL(IXNEW,IYNEW,IZNEW)
 120  CONTINUE
C
C ASSIGN COORDINATES AND SPINS TO THE PARTICLES
C
      IOLDCF = 1
      INEWCF = 2
C
      IPART = 0
      DO 200 ISUBL = 1,8
         DO 200 IZ = 1 + IOFFZ(ISUBL),LSIMPZ + IOFFZ(ISUBL),2
            ZZZ = IZ - 1
            DO 200 IY = 1 + IOFFY(ISUBL),LSIMPY + IOFFY(ISUBL),2
               YYY = SIN60 * (IY-1)
               DO 200 IX = 1 + IOFFX(ISUBL),LSIMPX + IOFFX(ISUBL),2
                  IPART                = IPART + 1
                  XCOORD(IPART,IOLDCF) = IX - 1 + 0.5D0 * MOD(IY-1,2)
                  YCOORD(IPART,IOLDCF) = YYY
                  ZCOORD(IPART,IOLDCF) = ZZZ
                  ISTAT(IPART,IOLDCF)  = 1
 200  CONTINUE
      DO 201 I=1,NDIAM
        XCOORD0(I) = XCOORD(I,IOLDCF)
YCOORD0(I) = YCOORD(I,IOLDCF)
ZCOORD0(I) = ZCOORD(I,IOLDCF)
  201 CONTINUE
C
C INITIALIZE RANDOM GENERATOR
C
      CALL RINITIALIZE(ISEED)
C
```

```
C THIS IS THE SETTING FOR SETUP FROM SCRATCH
C IF POSSIBLE, OVERWRITE THE CONFIGURATION BY
C DATA FROM INPUT FILE, UNIT 21
C
      READ(f28,END=300)
     &                 (XCOORD(I,IOLDCF),I=1,NDIAM),
     &                 (YCOORD(I,IOLDCF),I=1,NDIAM),
     &                 (ZCOORD(I,IOLDCF),I=1,NDIAM),
     &                 (ISTAT(I,IOLDCF),I=1,NDIAM)
      WRITE(f24,*) 'INITIAL SETTING OVERRIDDEN BY INPUT FILE'
300   CONTINUE
C
C
      FPERIX = 1.5D0 / LSIMPX
      FPERIY = 1.5D0 / LSIMPY
      FPERIZ = 1.5D0 / LSIMPZ
C
C INITIALIZE BONDS
C
      DO 510 INN = 1,8
         DO 510 I = 1,NDIAM
            INEI = NN(I,INN)
            DDXX = XCOORD(INEI,IOLDCF) - XCOORD(I,IOLDCF)
            DDYY = YCOORD(INEI,IOLDCF) - YCOORD(I,IOLDCF)
            DDZZ = ZCOORD(INEI,IOLDCF) - ZCOORD(I,IOLDCF)
            DDXX = DDXX - LSIMPX * INT(FPERIX * DDXX)
            DDYY = DDYY - LSIMPY * SIN60 * INT(FPERIY * DDYY/SIN60)
            DDZZ = DDZZ - LSIMPZ * INT(FPERIZ * DDZZ)
    DIST2 = DDXX ** 2 + DDYY ** 2 + DDZZ ** 2
            DD22 = ( (R02/DIST2)**6 - 2.D0*(R02/DIST2)**3 )
     &        * (Q+ISTAT(INEI,IOLDCF)*ISTAT(I,IOLDCF))
            BOND2(I,INN,IOLDCF)  = DD22
 510  CONTINUE
C
C DETERMINE BOND PART OF INTERNAL ENERGY
C
      UINT = 0.D0
      DO 560 INN = 1,8
         DO 560 I = 1,NDIAM
            UINT = UINT + BOND2(I,INN,IOLDCF)
 560  CONTINUE
C
      UINT = UINT / 2.D0
      UOUT = UINT * FACSYS
      WRITE(f24,*)
```

```
     &'INTERNAL ENERGY PER SITE AT START OF THE RUN (IN UNITS OF Um):',
     & UOUT
C
C DETERMINE THE MAGNETIZATION
C
      MAG = 0
      DO 561 I = 1,NDIAM
        MAG = MAG + ISTAT(I,IOLDCF)
 561  CONTINUE
C
      XMAGOUT = ABS(MAG) * FACSYS
      WRITE(f24,*) 'MAGNETIZATION AT START OF THE RUN: ',XMAGOUT
C
C SET STATISTICAL AVERAGES TO ZERO
C
      DO 980 I=1,4
        UAV(I)  = 0.D0
MAV(I)  = 0.D0
MUAV(I) = 0.D0
 980  CONTINUE
      ACCAV = 0.D0
C
C
C INITIALIZATION PART FINISHED
C
CZHU write parameters to the ist file
      write(f22,*)'Lx= ', LSIMPX
      write(f22,*)'Ly= ', LSIMPY
      write(f22,*)'Lz= ', LSIMPZ
      write(f22,*)'T= ', TEMP
      write(f22,*)'Q= ', Q
      write(f22,*)'mStart= ', MSTART

C WRITE HEADLINE
C
       WRITE(f22,9101)
9101  FORMAT(4X,'MCS',8X,'M',15X,'IE')


C WRITE HEADLINE
C
C BEGIN MONTE CARLO PROCEDURE
C
      DO 8000 MCS = MCSINI,MCSMAX
C
```

```
         DO I=1,NRAND
           IRAN(I) = RANF()
C    print*,iran(i)
         ENDDO
C
C FOR EACH PARTICLE, GENERATE A NEW POINT IN CONFIGURATION SPACE
C AND CALCULATE BOND CONTRIBUTION TO ENERGY DIFFERENCE
C
         FNX   = PMOV * 2.D0
         FNY   = PMOV * 2.D0
         FNZ   = PMOV * 2.D0
C
         DO 1000 I = 1,NDIAM
            XCOORD(I,INEWCF) = XCOORD(I,IOLDCF)
     &                       + FNX * IRAN(I) - PMOV
            YCOORD(I,INEWCF) = YCOORD(I,IOLDCF)
     &                       + FNY * IRAN(I + NDIAM) - PMOV
            ZCOORD(I,INEWCF) = ZCOORD(I,IOLDCF)
     &                       + FNZ * IRAN(I + NDIAM2) - PMOV
            ISTNEW           = 2*INT(2.D0 * IRAN(I + NDIAM3)) - 1
            ISTAT(I,INEWCF)  = ISTNEW
 1000    CONTINUE
C
         DO 1001 I = 1,NDIAM
            RO2(I) = (XCOORD(I,INEWCF)-XCOORD0(I))**2 +
     &        (YCOORD(I,INEWCF)-YCOORD0(I))**2 +
     &        (ZCOORD(I,INEWCF)-ZCOORD0(I))**2
 1001    CONTINUE
C
         DO 1002 I = 1,NDIAM
            IF (RO2(I).GT.DELTA2) THEN
      XCOORD(I,INEWCF) = XCOORD(I,IOLDCF)
      YCOORD(I,INEWCF) = YCOORD(I,IOLDCF)
      ZCOORD(I,INEWCF) = ZCOORD(I,IOLDCF)
    ENDIF
 1002    CONTINUE
C
         DO 2000 I=1,NDIAM
   DELTAE(I) = 0.D0
 2000    CONTINUE
C
         DO 3000 ISUBL = 1,8
C
            ILOW = (ISUBL - 1) * NDIAM/8 + 1
            IHGH = ISUBL * NDIAM/8
```

```
C
C FIRST NEIGHBOR CONTRIBUTIONS
C
            DO 1250 INN = 1,8
               DO 1250 I = ILOW,IHGH
                  INEI = NN(I,INN)
                  DDXX = XCOORD(INEI,IOLDCF) - XCOORD(I,INEWCF)
                  DDYY = YCOORD(INEI,IOLDCF) - YCOORD(I,INEWCF)
                  DDZZ = ZCOORD(INEI,IOLDCF) - ZCOORD(I,INEWCF)
                  DDXX = DDXX - LSIMPX * INT(FPERIX * DDXX)
                  DDYY = DDYY - LSIMPY * SIN60*INT(FPERIY * DDYY/SIN60)
                  DDZZ = DDZZ - LSIMPZ * INT(FPERIZ * DDZZ)
                  DIST2 = DDXX ** 2 + DDYY ** 2 + DDZZ ** 2
                  DD22 = ( (R02/DIST2)**6 - 2.D0*(R02/DIST2)**3 )
     #              * (Q+ISTAT(INEI,IOLDCF)*ISTAT(I,INEWCF))
                  BOND2(I,INN,INEWCF) = DD22
                  DELTAE(I) = DELTAE(I) + DD22 - BOND2(I,INN,IOLDCF)
 1250       CONTINUE
C
C ENERGY CALCULATION FINISHED
C
C FIND OUT WHICH OF THE MOVES IS ACCEPTED
C
            DO 1500 I = ILOW,IHGH
               PACC = DEXP(- DELTAE(I) * Um )
     &              - IRAN(I + NDIAM4)
               IACC(I) = 0
               IF(PACC.GT.0.D0) IACC(I) = 1
 1500       CONTINUE
C
C UPDATE SPINS AND COORDINATES AT THE CENTRAL SITE
C
            DO 1510 I = ILOW,IHGH
               IF(IACC(I).EQ.1) THEN
                 ISTAT(I,IOLDCF)  = ISTAT(I,INEWCF)
                 XCOORD(I,IOLDCF) = XCOORD(I,INEWCF)
                 YCOORD(I,IOLDCF) = YCOORD(I,INEWCF)
                 ZCOORD(I,IOLDCF) = ZCOORD(I,INEWCF)
                 UINT             = UINT + DELTAE(I)
               END IF
 1510       CONTINUE
C
C UPDATE BONDS AT THE CENTRAL SITE
C
            DO 1520 INN = 1,8
```

```
               DO 1520 I = ILOW,IHGH
                  IF(IACC(I).EQ.1) THEN
                     BOND2(I,INN,IOLDCF)  = BOND2(I,INN,INEWCF)
                  END IF
 1520       CONTINUE
C
C UPDATE THE BONDS OF THE NEIGHBORING SITES
C
            DO 2500 INN = 1,8
C*VDIR: IGNORE RECRDEPS
CDIR$ IVDEP
               DO 2500 I = ILOW,IHGH
                  INEI = NN(I,INN)
                  BOND2(INEI,-INN+9,IOLDCF)  = BOND2(I,INN,IOLDCF)
 2500       CONTINUE
C
 3000    CONTINUE
C
C LOOP OVER SUBLATTICES FINISHED
C
C MEASURE SOME OBSERVABLES
C
         IF(MCS.GE.MSTART.OR.MCS.EQ.KOUTOB) THEN
C
C DETERMINE ACCEPTANCE RATE AND MAGNETIZATION
C
            JACC = 0
    MAG = 0
            DO 6010 I = 1,NDIAM
               JACC = JACC + IACC(I)
       MAG  = MAG  + ISTAT(I,IOLDCF)
 6010       CONTINUE
    ACC = JACC * FACSYS
    XMOUT = MAG * FACSYS
C
C INTERNAL ENERGY
C
            UOUT = UINT * FACSYS
C
         END IF
C
C END OF MEASURING BLOCK
C
C CUMULATE AVERAGES
C
```

```
         IF(MCS.GE.MSTART) THEN
            DO 7000 MOM = 1,4
               UAV(MOM)      = UAV(MOM)   + UOUT ** MOM
               MAV(MOM)      = MAV(MOM)   + XMOUT ** MOM
       MUAV(MOM)     = MUAV(MOM)  + XMOUT ** MOM * UINT
 7000       CONTINUE
            ACCAV = ACCAV + ACC
         END IF
C
C OUTPUT OBSERVABLES
C
         IF(MCS.EQ.KOUTOB) THEN
            KOUTOB = KOUTOB + NOUTOB
C            WRITE(f22,9000) MCS,ACC,UOUT,XMOUT
    WRITE(f22,9001) MCS,XMOUT,UOUT
         END IF

C
C DUMP CONFIGURATION
C
         IF(MCS.EQ.KOUTCF) THEN
            KOUTCF = KOUTCF + NOUTCF
            REWIND(f21)
            WRITE(f21)
     &                (XCOORD(I,IOLDCF),I=1,NDIAM),
     &                (YCOORD(I,IOLDCF),I=1,NDIAM),
     &                (ZCOORD(I,IOLDCF),I=1,NDIAM),
     &                (ISTAT(I,IOLDCF),I=1,NDIAM)
            REWIND(f27)
            do 730 I=1,NDIAM
            write(f27,732)XCOORD(I,IOLDCF),YCOORD(I,IOLDCF),
     &                ZCOORD(I,IOLDCF),ISTAT(I,IOLDCF)
732         format(3(2x,f12.6),2x,I4)
730         enddo
            write(f27,733)LSIMPX,LSIMPY,LSIMPZ
733     format(3I5)

            WRITE(f24,*) 'CONFIGURATION DUMPED AFTER MCS = ',MCS
         END IF
C
C OUTPUT STATISTICAL AVERAGES
C
         IF(MCS.EQ.KOUTAV) THEN
            KOUTAV = KOUTAV + NOUTAV
            REWIND(f23)
```

```
C
            FACTOR = 1.D0 / (MCS - MSTART + 1)
            DO 7100 MOM = 1,4
               UAVOUT(MOM)  = UAV(MOM)  * FACTOR
               MAVOUT(MOM)  = MAV(MOM)  * FACTOR
       MUAVOUT(MOM) = MUAV(MOM) * FACTOR
 7100       CONTINUE
            ACCOUT = ACCAV * FACTOR
C
            SPEC   = NDIAM * (Um ** 2) * (UAVOUT(2) - UAVOUT(1) ** 2)
            SUSC = NDIAM * Um * (MAVOUT(2) - MAVOUT(1) ** 2)
            CUM = 0.D0
            IF(MAVOUT(2).NE.0.D0)
     &        CUM = 1.D0 - MAVOUT(4) / (3.D0 * MAVOUT(2) ** 2)
          V1 = - ( MUAVOUT(1)/MAVOUT(1) ) + UAVOUT(1)*NDIAM
    V2 = - ( MUAVOUT(2)/MAVOUT(2) ) + UAVOUT(1)*NDIAM
    dUdK = - ( 1.D0 / (3.D0 * MAVOUT(2)**2) ) *
     &        ( MAVOUT(4) * ( 2.D0*MUAVOUT(2)/MAVOUT(2) -
     &        UAVOUT(1)*NDIAM ) - MUAVOUT(4) )

C
            WRITE(f23,*)
     &        'LINEAR SIZE = ',LSIMPX, ' # PARTICLES = ',NDIAM
            WRITE(f23,*) 'TEMPERATURE = ',TEMP
            WRITE(f23,*) 'SEED = ',ISEED
            WRITE(f23,*) 'STATISTICAL AVERAGES AFTER MCS = ',MCS
            WRITE(f23,*) '                        '
            WRITE(f23,*) ACCOUT, ' AVERAGE ACCEPTANCE RATE'
    WRITE(f23,*) '                       '
    DO I=1,4
               WRITE(f23,7770) MAVOUT(I),I
7770     format( E24.18,'  ORDER PARAMETER, MOMENT ', I1)
    ENDDO
    WRITE(f23,*) SUSC, ' SUSCEPTIBILITY'
    DO I=1,4
               WRITE(f23,7772) UAVOUT(I),I
7772     format(E24.18, '  INTERNAL ENERGY, MOMENT ', I1)
    ENDDO
    DO I=1,4
               WRITE(f23,7774) MUAVOUT(I),I
7774     format(E24.18, '  m',I1,'E')
    ENDDO
            WRITE(f23,*) SPEC, ' SPECIFIC HEAT'
    WRITE(f23,*) CUM, ' CUMULANT'
    WRITE(f23,*) dUdK, ' DERIVATIVE OF THE CUMULANT'
```

```
    WRITE(f23,*) V1, ' DERIVATIVE OF ln(m1)'
    WRITE(f23,*) V2, ' DERIVATIVE OF ln(m2)'
  ENDIF
C
 8000 CONTINUE
      call MPI_FINALIZE(ierror)
C
C LOOP OVER MONTE CARLO STEPS FINISHED
C
 9000 FORMAT(I8,3E15.7)
 9001 FORMAT(I8,2(3X,E18.12))
 9999 format(2x,i4,2x,3(f8.4,2x),i2)
C
      STOP
      END

*=====================================================================
* This file contains the Cray specific subroutines for random number
* generation and convolution SUM routine
*=====================================================================

      SUBROUTINE RINITIALIZE(ISEED)
      IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
      PARAMETER(NAB3=101280)
      COMMON/CNUM/NUM
      NUM=NAB3-1280
      CALL RANINI(ISEED)
      END


      DOUBLE PRECISION FUNCTION RANF()
      IMPLICIT REAL*8(A-H,O-Z),INTEGER(I-N)
      COMMON/CNUM/NUM
      ranf =  RANDA(NUM)
      NUM=NUM+1
      END

C*********************************************************************

      SUBROUTINE RANINI(ISeed)
      IMPLICIT NONE

      INTEGER NAB3
      PARAMETER(NAB3 = 101280)
```

```
      INTEGER ISeed,IMod,I,J,K
      INTEGER RanVec,IMax
      REAL*8 RMod,PMod,DMaxI,RanVec2
      COMMON/dom/RanVec(NAB3),Ranvec2(NAB3)

      IMax = 2147483647
      DMaxI = 1.0D0/2147483647.0D0
      RMod = DBLE(ISeed)
      PMod = DBLE(IMax)

      DO I = 1,1000
        RMod = RMod*16807.0D0
        IMod = RMod*DMaxI
        RMod = RMod - PMod*IMod
      END DO

      DO I = 1,1279
        RanVec(I) = 0
        DO J = 0,30
          DO K = 1,36
            RMod = RMod*16807.0D0
            IMod = RMod*DMaxi
            RMod = RMod - PMod*IMod
          END DO
          RMod = RMod*16807.0D0
          IMod = RMod*DMaxi
          RMod = RMod - PMod*IMod
          IF (RMod .GT. 0.5D0*PMod) RanVec(I) = IBSET(RanVec(I),J)
        END DO
      END DO

C**  Generate 1000 random numbers to warm up the generator
      CALL RANDOM(1000)

      RETURN
      END

C*********************************************************************

      SUBROUTINE RANDOM(Number)

      IMPLICIT NONE
      INTEGER NAB3
      PARAMETER(NAB3 = 101280)
      real*8 RanVec2
```

```
      INTEGER RanVec,Number,I
      COMMON/dom/RanVec(NAB3),RanVec2(NAB3)


C**   This works because Number will always be a multiple of four for this
C**   program
C**   Unroll this loop for extra speed

      DO I = 1,Number,4
        RanVec(I+1279) = IEOR(RanVec(I),    RanVec(I+216))
        RanVec(I+1280) = IEOR(RanVec(I+1),  RanVec(I+217))
        RanVec(I+1281) = IEOR(RanVec(I+2),  RanVec(I+218))
        RanVec(I+1282) = IEOR(RanVec(I+3),  RanVec(I+219))
      END DO

C**   Copy the final 1279 elements to the beginning for use on the next call
C**   Unroll this loop for extra speed

      DO I = 1,1276,4
        RanVec(I)   = RanVec(I+Number)
        RanVec(I+1)   = RanVec(I+1+Number)
        RanVec(I+2)   = RanVec(I+2+Number)
        RanVec(I+3)   = RanVec(I+3+Number)
      END DO

      RanVec(1277) = RanVec(1277 + Number)
      RanVec(1278) = RanVec(1278 + Number)
      RanVec(1279) = RanVec(1279 + Number)

      do I = 1,number
        RanVec2(I) = dble(RanVec(I))*4.656612875D-10
      end do

      RETURN
      END

      FUNCTION RANDA(NUM)
      INTEGER NUM,ranvec,NAB3
      REAL*8 RANVEC2,RANDA

      PARAMETER(NAB3=101280)
      COMMON/dom/RanVec(NAB3),Ranvec2(NAB3)

        if(NUM.ge.NAB3-1280)then
          CALL RANDOM(NAB3-1280)
```

```
    RANDA = ranvec2(1)
    NUM = 2
  else
    RANDA = ranvec2(NUM)
    NUM = NUM + 1
  endif
end
```

## B.2 The Input File input_tri.dat

```
1             INITIAL VALUE FOR TIME MCS (USUALLY 1)
100000        FINAL VALUE FOR TIME MCS
10000         TIME INTERVAL FOR DUMPING CONFIGURATIONS
1             TIME INTERVAL FOR OUTPUT OBSERVABLES
1             FIRST MCS FOR AVERAGING
10000         TIME INTERVAL FOR OUTPUT STATISTICAL AVERAGES
410060409     ISEED
0.05D0        MAXIMUM DISTANCE FOR ATTEMPTED MOVE
 4.50         TEMPERATURE (ALL IN UNITS OF Um)
3.0           ELASTIC OVER MAGNETIC PARAMETER
1.0D0         SQUARE OF NEAREST NEIGHBOR DISTANCE
```

## B.3 The Shell Script

This is a very simple version. See the CDROM for the advanced version.

```
#!/bin/sh
   ln -s    posout          fort.28
   ln -s   posout1          fort.27
   ln -s    posout          fort.21
   ln -s       ist          fort.22
   ln -s     medie          fort.23
   ln -s    output          fort.24

# assume the executable is tricons.x,
# and the input file is inp_tricons.dat
   /usr/bin/time -p ./tricons.x <inp_tricons.dat >>& output_file
```

# Appendix C

# Povray Tools

In the chapter, three files are provided: a C code for processing raw data, a povray file, and the shell script to run these two.

## C.1 Shell Script

```
#!/bin/csh
gcc -lm drawLattice.c -o drawLattice
set i=$1
set SN=‘echo $i |awk ’{len=length($1);
        printf "%04d\n", substr($1,11,len-13)}’‘
set ext=‘echo $SN |awk ’{printf "%04d\n",$1/10}’‘
set dividable=‘echo $SN|awk ’{print $1%10}’‘
set frame=$i
./drawLattice $i >balls.inc
cp example2.pov $frame.pov
povray +I$frame.pov +H300 +W400 +A0.1
rm $frame.pov
```

## C.2 Povray File example2.pov

```
camera {
  location  1*<1,1,0.75>  //position of camera
  look_at   <0.0, 0.0, 0.0>  //look at position
  right     x*image_width/image_height  //don't change
  up z  //don't change
  direction -1*x //don't change
  sky z //don't change
}


background{rgb 0.8*<1,1,1>}

light_source{<1000,100,500> rgb 1}

//#declare L=7;
//note, this is to shorten the data set, set=to total size to view all atoms
#declare crad=0.04;

#declare tex1=
texture{
pigment{rgb <0.5,0,0>}
finish{ambient 0.25 diffuse 0.75}// specular 0.3}
}
#declare tex2=
texture{
pigment{rgb <0.5,0.4,0>}
finish{ambient 0.25 diffuse 0.75}// specular 0.3 reflection 0.3}
}
#declare tex3=
texture{
pigment{rgb <0,0.5,0>}
finish{ambient 0.5 diffuse 0.5}// specular 0.3}
}
#declare tex4=
texture{
pigment{rgb <0,0.5,0>}
finish{ambient 0.25 diffuse 0.75}// specular 0.3}
}


#include "balls.inc" //this contains your own cylinders, etc...
```

```
union{
object{
balls
}

translate -<1,1,0>/2
scale <1,1,1>/L
translate -0.5*<1,1,1>

rotate clock*360*z
//no_shadow
}
```

## C.3 File drawLattice.c

This program reads data and output the povray components.

```
#include <stdio.h>
#include <stdlib.h>
#define L 6
#define CELL_SCALE 0.7

//global variables
double X[L+1][L+1][L+1],Y[L+1][L+1][L+1],Z[L+1][L+1][L+1];
double color[L+1][L+1][L+1];
int nr[L+1];
double lx,ly,lz;

//subroutines
void drawBox(int,int,int);
void drawCylinder(int,int,int,int,int,int);
void drawTriangle(int,int,int,int,int,int,
                  int,int,int,double,double,double);

int main(int argc, char** argv)
{
  int i,j,k,m,sub;
  double spin;
  int tmp;
  char str[256];

  FILE* fin=NULL;

  fin=fopen(argv[1],"r");
  printf("//inpufile: %s, fin=%g\n",argv[1],fin);

  for(i=0;i<L+1;i++)
    nr[i]=(i+1)%(L+1);

  //initialization
  for(k=0;k<L;k++)
    for(j=0;j<L;j++)
      for(i=0;i<L;i++)
color[i][j][k]=0.0;

  //read data
```

```
  for(sub=0;sub<8;sub++)
    for(k=0;k<L;k++)
      for(j=0;j<L;j++)
        for(i=0;i<L;i++)
{
  fgets(str,256,fin);
     sscanf(str,"%lg\t%lg\t%lg\t%lg\n",
                 &(X[i][j][k]),&(Y[i][j][k]),&(Z[i][j][k]),&spin);
  color[i][j][k]+=spin/8.0;
}
  fgets(str,256,fin);
  sscanf(str,"%lg\t%lg\t%lg\n",&lx,&ly,&lz);
  //extend to the periodic outlier
  for(i=0;i<L+1;i++)
    for(j=0;j<L+1;j++)
    {
      X[i][j][L]=X[i][j][0];
      Y[i][j][L]=Y[i][j][0];
      Z[i][j][L]=Z[i][j][0]+lz;
      color[i][j][L]=color[i][j][0];
    }
  for(i=0;i<L+1;i++)
    for(k=0;k<L+1;k++)
    {
      X[i][L][k]=X[i][0][k];
      Y[i][L][k]=Y[i][0][k]+ly;
      Z[i][L][k]=Z[i][0][k];
      color[i][L][k]=color[i][0][k];
    }
  for(j=0;j<L+1;j++)
    for(k=0;k<L+1;k++)
    {
      X[L][j][k]=X[0][j][k]+lx;
      Y[L][j][k]=Y[0][j][k];
      Z[L][j][k]=Z[0][j][k];
      color[L][j][k]=color[0][j][k];
    }


  //output surface
  fprintf(stdout,"#declare L=%g;\n",(lx*(L+1))/L);
  fprintf(stdout,"#declare balls=\n union{\n");

  for(i=0;i<L;i++)
    for(j=0;j<L;j++)
```

```
      for(k=0;k<L;k++)
drawBox(i,j,k);

  fprintf(stdout,"}\n");
}

void drawBox(int i,int j, int k)
{
  double spin;
  double red, blue, green;
  spin=color[i][j][k];
  fprintf(stdout,"\t union{
         // begin of box <%d, %d, %d>\n", i,j,k);
  if(spin>=0)
  {
    red=1.0;
    green=1.0-spin;
    blue=1.0-spin;
  }else
  {
    red=1.0+spin;
    green=1.0+spin;
    blue=1.0;
  }
  //spheres
  fprintf(stdout,
    "\t\t sphere{<%lg,%lg,%lg>,crad texture{tex3}}\n",
    X[i][j][k],Y[i][j][k],Z[i][j][k],red,green,blue);

  //cylinders
  // in i-direction
  drawCylinder(i,j,k,nr[i],j,k);
  drawCylinder(i,nr[j],k,nr[i],nr[j],k);
  drawCylinder(i,j,nr[k],nr[i],j,nr[k]);
  drawCylinder(i,nr[j],nr[k],nr[i],nr[j],nr[k]);
  // in j-direction
  drawCylinder(i,j,k,i,nr[j],k);
  drawCylinder(nr[i],j,k,nr[i],nr[j],k);
  drawCylinder(i,j,nr[k],i,nr[j],nr[k]);
  drawCylinder(nr[i],j,nr[k],nr[i],nr[j],nr[k]);

  // in k-direction
  drawCylinder(i,j,k,i,j,nr[k]);
  drawCylinder(i,nr[j],k,i,nr[j],nr[k]);
  drawCylinder(nr[i],j,k,nr[i],j,nr[k]);
```

```
  drawCylinder(nr[i],nr[j],k,nr[i],nr[j],nr[k]);


  //triangles
  //<i,j,k>,<nr[i],j,k>,<i,nr[j],k>, k fixed
  drawTriangle(i,j,k,nr[i],j,k,
               i,nr[j],k,red,green,blue);
  drawTriangle(nr[i],nr[j],k,nr[i],j,k,
       i,nr[j],k,red,green,blue);
  drawTriangle(i,j,nr[k],nr[i],j,nr[k],
       i,nr[j],nr[k],red,green,blue);
  drawTriangle(nr[i],nr[j],nr[k],nr[i],j,nr[k],
               i,nr[j],nr[k],red,green,blue);
   //j fixed
  drawTriangle(i,j,k,nr[i],j,k,
               i,j,nr[k],red,green,blue);
  drawTriangle(nr[i],j,nr[k],nr[i],j,k,
               i,j,nr[k],red,green,blue);
  drawTriangle(i,nr[j],k,nr[i],nr[j],k,
       i,nr[j],nr[k],red,green,blue);
  drawTriangle(nr[i],nr[j],nr[k],nr[i],nr[j],k,
       i,nr[j],nr[k],red,green,blue);

  //i fixed
  drawTriangle(i,j,k,i,nr[j],k,
               i,j,nr[k],red,green,blue);
  drawTriangle(i,nr[j],nr[k],i,nr[j],k,
       i,j,nr[k],red,green,blue);
  drawTriangle(nr[i],j,k,nr[i],nr[j],k,
               nr[i],j,nr[k],red,green,blue);
  drawTriangle(nr[i],nr[j],nr[k],nr[i],nr[j],k,
               nr[i],j,nr[k],red,green,blue);

  //translation and scale
  fprintf(stdout,"\t\t  translate -<%g,%g,%g>\n",
         X[i][j][k],Y[i][j][k],Z[i][j][k]);
  fprintf(stdout,"\t\t  scale <1,1,1>*%g",CELL_SCALE);
  fprintf(stdout,"\t\t  translate  <%g,%g,%g>\n",
         X[i][j][k],Y[i][j][k],Z[i][j][k]);
  fprintf(stdout,"\t} // end of box <%d, %d, %d>\n ", i,j,k);

}
void drawCylinder(int i1,int j1,int k1, int i2, int j2, int k2)
{
  fprintf(stdout,"\t\t cylinder{<%lg,%lg,%lg>,<%lg,%lg,%lg>,
```

```
                crad texture{tex3}}\n",
         X[i1][j1][k1],Y[i1][j1][k1],Z[i1][j1][k1],
                X[i2][j2][k2],Y[i2][j2][k2],Z[i2][j2][k2]);
}

void drawTriangle(int i1,int j1,int k1,
                  int i2, int j2, int k2,
                  int i3,int j3,int k3,
                  double red, double green,double blue)
{
  fprintf(stdout,
         "\t\t triangle{<%lg,%lg,%lg>,
         <%lg,%lg,%lg>,<%lg,%lg,%lg>
         texture{pigment{rgb <%g,%g,%g>}
         finish{ambient 0.5 diffuse 0.5}}}\n",
         X[i1][j1][k1],Y[i1][j1][k1],Z[i1][j1][k1],
         X[i2][j2][k2],Y[i2][j2][k2],Z[i2][j2][k2],
         X[i3][j3][k3],Y[i3][j3][k3],Z[i3][j3][k3],
         red,green,blue);
}
```

# Bibliography


[1] L. Onsager. *Phys. Rev.*, 65:117, 1944.

[2] M. E. Fisher. *J. Math. Phys.*, 4:278, 1963.

[3] M. E. Fisher. *J. Math. Phys.*, 5:944, 1964.

[4] L. P. Kadanoff et al. *Rev. Mod. Phys.*, 39:395, 1967.

[5] E. Brezin, J.-C. Le Guillou, and J. Zinn-Justin. *Phys. Lett. A*, 47:285, 1974.

[6] J.-C. Le Guillou and J. Zinn-Justin. *Phys. Rev. B*, 21:3976, 1980.

[7] J-H Chen, M. E. Fisher, and B. G. Nickel. *Phys. Rev. Lett.*, 48:630, 1982.

[8] G. A. Baker, B. G. Nickel, and D. I. Meiron. *Phys. Rev. B.*, 17:1365, 1978.

[9] A. M. Ferrenberg and D. P. Landau. *Phys. Rev. B*, 44:5081, 1991.

[10] H. W. Blöte, E. Luijten, and J. R. Heringa. *J. Phys. A*, 28:6289, 1995.

[11] P. C. Kelires and J. Tersoff. *Phys. Rev. Lett.*, 63:1164, 1989.

[12] P. C. Weakliem and E. A. Carter. *Phys. Rev. B*, 45:13458, 1992.

[13] B. Dünweg and D. P. Landau. *Phys. Rev. B*, 48:14182, 1993.

[14] M. Laradji, D. P. Landau, and B. Dünweg. *Phys. Rev. B*, 51:4894, 1995.

[15] C. Tzoumanekas and P. C. Kelires. *Phys. Rev. B*, 66:195209, 2002.

[16] P. Azaria and B. Delamotte. In H.T. Diep, editor, *Magnetics Systems with Competing Interactions*. World Scientific, Singpore, 1994.

[17] M. E. Fisher. In M. S. Green, editor, *Critical Phenomena*. Academic, New York, 1971.

[18] M. E. Fisher and M. N. Barber. *Phys. Rev. Lett.*, 28:1516, 1972.

[19] A. M. Ferrenberg and R.H. Swendsen. *Phys. Rev. Lett.*, 61:2635, 1988.

[20] F. Wang and D. P. Landau. *Phys. Rev. Lett.*, 86:2050, 2001.

[21] F. Wang and D. P. Landau. *Phys. Rev. E*, 64:056101, 2001.

[22] B. Dünweg. *Computersimulationen zu Phasenübergängen und Kritischen Phänomenen*. Habilitationsschrift, Max-Planck-Institut für Polymerforschung, Mainz, Germany, 2000.

[23] F. Tavazza, D. P. Landau, and J. Adler. *Phys. Rev. B*, 70:184103, 2004.

[24] E. H. Boubcheur and H. T. Diep. *J. Appl. Phys.*, 85:6085, 1999.

[25] E. H. Boubcheur, P. Massimino, and H. T. Diep. *J. Magnetism and Magnetic Materials*, 223:163, 2001.

[26] C. Domb and M. S. Green. *Phase Transitions and Critical Phenomena*. Academic Press, London, 1976.

[27] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. M. Teller, and E. Teller. *J. Chem Phys.*, 21:1087, 1953.

[28] R. J. Glauber. *J. Math. Phys.*, 4:294, 1963.

[29] D. P. Landau and K. Binder. *A Guide to Monte Carlo Simulations in Statistical Physics*. University Press, Cambridge, Cambridge, UK, 2000.

[30] K. Binder. *Z. Phys. B*, 43:119, 1981.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, editors. *Numerical Recipes in C*, page 683. Cambridge University Press, NY, 1988.

[32] J. G. Kirkwood. *J. Chem. Phys.*, 7:506, 1939.

[33] P. N. Keating. *Phys. Rev.*, 145:637, 1966.

[34] F. H. Stillinger and T. A. Weber. *Phys. Rev. B*, 31:5262, 1985.

[35] J. Tersoff. *Phys. Rev. Lett.*, 56:632, 1986.

[36] R. C. Tausworthe. *Math. Comput.*, 19:201, 1965.

[37] M. M. Tsypin and H. W. J. Blöte. *Phys. Rev. E*, 62:73, 2000.

[38] N. B. Wilding and A. D. Bruce. *J. Phys. Condens. Matter*, 4:3087, 1992.

[39] M. E. Fisher. *Rep. Prog. Phys.*, 30:615, 1967.

[40] B. Dünweg. *private communication.*

[41] S. A. Antonenko and A. I. Sokolov. *Phys. Rev. E*, 51:1894, 1995.