# Modified late-time domain growth behavior due to Compressibility

by

Meng Meng

(Under the direction of David P. Landau)

## Abstract

We study the static and dynamic properties of the compressible Ising models under constant pressure. Our system of study is a two-dimensional triangular-net Ising model with continuous particle positions. To investigate the effects of compressibility on the static and dynamic characteristics of the model, we include an elastic energy part in the Hamiltonian to adjust the rigidity. Through investigating the fourth order cumulant and the normalized order parameter distribution by Monte Carlo simulation, we find that the elastic models belong to the same universality class of the rigid Ising model. Besides that, we perform large scale Monte Carlo simulations to study long-time domain growth behavior following a quench under its critical temperature in our compressible Ising model with zero total magnetization. For the rigid (lattice) model we find the late-time domain growth factor $n$ in $R(t) = A + Bt^n$ has Lifshitz-Slyozov value of $\frac{1}{3}$. For flexible models, results clearly indicate that $n$ is reduced by compressibility.

Index words: phase transition, phase separation, compressible Ising model, universality class, late-time domain growth, domain growth exponent

# Modified late-time domain growth behavior due to Compressibility

by

Meng Meng

B.S., Zhengzhou University, Zhengzhou, China, 2002

M.S., Shanghai Jiaotong University, Shanghai, China, 2005

M.S., University of Georgia, United States, 2011

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

2012

Modified late-time domain growth behavior due to Compressibility

by

Meng Meng

Approved:

| | |
|---|---|
| Major Professor: | David P. Landau |
| Committee: | Steven P. Lewis |
| | Michael Bachmann |
| | William Dennis |

## Dedication

This dissertation is dedicated to my parents, my wife, and my son.

## Acknowledgments

I am deeply grateful to my advisor, Prof. David P. Landau, for his invaluable instructions, insightful suggestions, generous help and great patience to my dissertational research. Without his help, this dissertation wouldn't have been possible. Beyond science, I also learned from him the rigorous attitude toward scientific research.

I'd also like to thank Dr. Steven J. Mitchell for helpful discussions. His experience helped me greatly when I started on my research.

I also appreciate Dr. Shan-Ho Tsai for her ready and constant support from GACRC. She always gave me the most prompt and generous help when I had technical problems about computing.

I also want to say thanks to the Systems Administrator Associate of CSP, Mr. Jeff Deroshia, who kindly helped me many times with using CSP's computing resources.

I would say many thanks to Prof. M. Howard Lee and Prof. Loris Magnani for their most generous and kind help to me throughout my study in CSP.

I would like to thank Prof. William Dennis, Prof. Steven Lewis and Prof. Michael Bachmann for serving on my advisory committee, and for their advice and guidance.

Last but not least, I would like to express my great gratitude to my parents and my wife, for their constant support and beliefs.

## Table of Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

The Ising model is amongst the most widely known and studied models of statistical physics, and people have known quite well about its critical behavior due to its simplicity [1]. The original Ising model is defined on a lattice with the following Hamiltonian:

$$\mathcal{H} = -J \sum_{<i,j>} s_i s_j - H \sum_i s_i \tag{1.1}$$

where $\sum_{<i,j>}$ is a sum over nearest-neighbor pairs of particles, $J$ is the exchange energy, $s_i$ is the spin value at site $i$ with possible values of either $+1$ or $-1$, and $H$ is the external magnetic field. $J$ can be either positive or negative. If $J > 0$, the system is called ferromagnetic, in which case the neighboring spins tend to have the same values to lower the total energy; and if $J < 0$, the system is termed antiferromagnetic, and the ground state will have all the neighboring spins with opposite values.

The Ising model serves as a good testbed for binary alloys. The two atom species of a binary alloy can be readily represented by the up and down spin of the Ising model and the difference between the two chemical potentials for the two atomic species is proportional to the magnetic field. The chemical potential is the amount of energy by which the system would change if an additional particle were introduced. However, the limitations of simulate binary alloys by using the rigid Ising model are quite obvious. The rigid Ising model fails to reflect the atom movements of the binary alloys. With varying interactions by their neighbors, atoms in alloys move randomly around their equilibrium positions at finite temperature $T$. And more importantly, the motion of the atoms can be correlated spatially and thus contribute significantly to the thermodynamics of the system around critical points. So compressible Ising models, which remove the rigidity constraints of traditional systems, are appropriate for simulating realistic binary alloys. A compressible Ising model assumes an elastic and deformable net with the interaction $J$ depending on the bond lengths instead of being constant. In some way it is capable of capturing the nature of physical binary alloys.

For decades there has been great interests on the part of physicists in studying the static and dynamic properties of the compressible Ising models [2–7, 11–15, 18–25]. Many theoretical approaches have been developed to study compressible Ising models but unfortunately they were found unreliable in many cases. So Monte Carlo simulations become a very important and useful tool to study the nature of the phase transitions of off-lattice Ising models. But a good understanding of compressible Ising models is still not achieved. Theoretical and numerical studies [11–14, 22, 23, 25] give inconsistent descriptions of its critical behavior. So in the first part of this dissertation, we will focus on the simulational results for the static critical properties of ferromagnetic compressible system at constant pressure.

Besides the static critical behavior, the dynamic properties of the Ising model could also be affected by compressibility [8–10]. So we also investigate the late-time domain growth behavior following a quench under the critical temperature in the compressible Ising model. The theory of phase-ordering kinetics or "domain coarsening" following a temperature quench from a homogeneous phase into a two-phase region has a history going back more than four decades to the pioneering work of Lifshitz, Slyozov and Wagner [2,28]. Since that time, many studies [3–7] have been done on the phase separation phenomenon. Systems quenched from a disordered phase into an ordered phase do not order instantaneously. Instead, domains of equilibrium phases increase as a result of surface tension and diffusion. Lifshitz and Slyozov [2] predicted the late-time domain size, in the limit of a dilute amount of one phase, grows as a power law,

$$R(t) = A + Bt^n \tag{1.2}$$

where $R$ is the domain size, $t$ is the time after the quenching, $A$ and $B$ are constants, and $n$ as the domain growth exponent equals $\frac{1}{3}$. Their theory has been generalized to equal fractions of two phases by Huse [3] and consistent results for lattice models are observed by simulations [3,4,7]. In our study, we only focus on the dynamic behavior following a critical quench where the system has equal fraction of phases. To find out how the compressibility affects the domain growth exponent $n$, we study the late-time domain growth behavior of a compressible Ising model using Kawasaki dynamics in our simulation to conserve the total magnetization. This is because in conventional Monte Carlo dynamics, flipping a single spin in the Ising model corresponds to converting one atom to the other, which is inadmissible in binary alloys. The dynamics must conserve the number of two species separately, which leads to the conservation of the magnetization ( or order parameter ) of the Ising model.

As suggested by [8–10], domain growth may be affected by compressibility. Also, as we know from recent studies [11,13–15], the compressibility can affect the static critical behavior of Ising models and hence may affect the domain growth behavior. Mitchell and Landau [7] have studied the late-time domain growth behavior for one particular kind of compressible 2D Ising model. They found significant deviations from the theoretically expected $n = \frac{1}{3}$ late-time growth factor for the compressible mismatch model, but for a compressible model with no mismatch, they found only a slight deviation from $n = \frac{1}{3}$. The mismatch refers to the different preferred bond lengths for different neighboring spin pairs, we will explain it in Chapter 3.

Their results suggest that the current understanding of the classes of domain growth is incomplete. So, in our research, we investigate the changes in dynamic behavior due to the flexibility of the model by a large length scale (512$\times$ 512) and long time scale ($10^6$ MCS) Monte Carlo simulation. For the study on both static and dynamic properties of compressible Ising models, our system has no mismatch and use a triangular-net to maintain the topology of the lattice. We include an elastic energy term into the Hamiltonian and adjust the elasticity of the model through varying the elastic constant $k$. In this way we can directly change the elasticity of the model and, on the other hand, we can adjust the relative importance of the magnetic energy and the elastic energy terms easily.

The rest of this dissertation is organized as follows. We present the theoretical and computational backgrounds in Chapter 2. The details of the model and methods we use in our simulation are included in Chapter 3. In Chapter 4, we show results of static critical behavior for our compressible Ising model. The results of domain growth behavior for both the rigid model and the elastic models are presented in Chapter 5. We conclude in Chapter 6.

# Chapter 2

# Background

## 2.1 Phase Transition

The study of phase transitions has long been a topic of great interest in a variety of related scientific disciplines and plays a central role in research in many fields of physics. Some simple theoretical approaches, such as mean field theory, tackle phase transitions in simple and intuitive ways. But they fail to provide a general framework to explain different phenomena occurring under different internal and external conditions [46, 47, 49]. For example, the mean field theory doesn't account for the effects from fluctuations, which are the key to critical phenomena. More efforts have been taken in developing more mature theoretical and numerical approaches to understand phase transitions in the last half century.

### 2.1.1 order parameter

A phase transition is closely related to the order parameter of a system. It assumes a non-zero value in ordered phase and a zero value in disordered phase. It may have different expressions for different systems. For instance, in a ferromagnet it is simply the spontaneous magnetization; in a liquid-gas system it is the density difference between the liquid and gas phases. Phase transitions are classified in the following way: if the first derivatives of the free energy are discontinuous at the transition temperature, the transition is of first order; if the first derivatives are continuous whereas the second derivatives are not, the transition is termed second order. We usually term the second order transition as a critical transition and the temperature where we observe the critical transition as the critical temperature $T_C$.

We now discuss the phase transition of Ising model. The Hamiltonian of a typical ferromagnetic Ising model is shown in Eq. 1.1 with $J > 0$. The order parameter is defined as:

$$\langle m \rangle = \frac{1}{N} \sum_i s_i \tag{2.1}$$

The Ising system (except for the one dimensional system) has phase transitions by varying temperature $T$ or magnetic field $H$. As shown in the Fig. 2.1, most spins point upward for $H > 0$ and downward for $H < 0$, when $T$ is below $T_C$. The dashed first order line is also plotted in the figure. It ends at $T_C$, at which point the transition turns into second order with $H_C = 0$. As we see, a first order transition occurs at $T < T_C$ when $H$ is swept through zero. For example, it happens at $B$ along the line $A \to B \to C$. But there is no transition occurring along $A \to D \to C$.

### 2.1.2 Universality classes

We have the following power-law singularities [44, 45]:

$$m = m_0 \epsilon^{\beta} \tag{2.2}$$

$$\chi = \chi_0 \epsilon^{-\gamma} \tag{2.3}$$

$$C = C_0 \epsilon^{-\alpha} \tag{2.4}$$

$$\xi = \xi_0 \epsilon^{-\nu} \tag{2.5}$$

where $\epsilon = |1 - T/T_C| \to 0$ is the reduced distance from the critical temperature and the powers are critical exponents that are unique for systems with the same symmetries. $m$ is the magnetization per spin, $\chi$ is the magnetic susceptibility per spin, $C$ is the specific heat per spin and $\xi$ is correlation length. The fundamental reason that leads to these singularities is the divergence in the correlation length $\xi$ at the critical point.

The universality hypothesis asserts that the interatomic interactions don't contribute to the system's critical behavior. Instead, it argues that relevant properties which determine

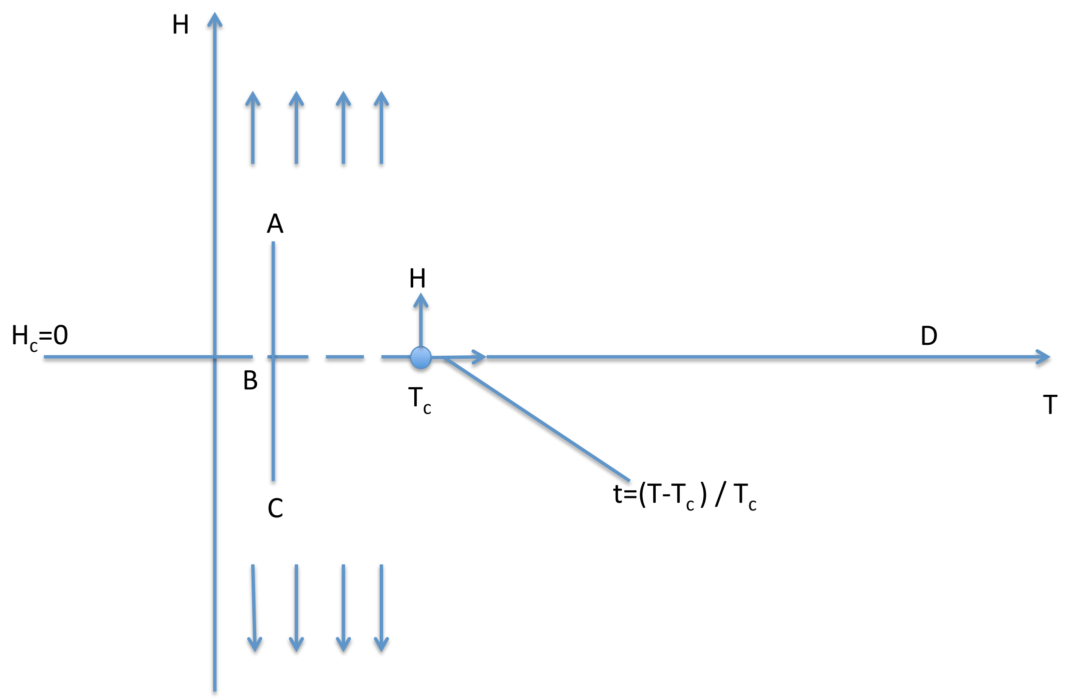

Figure 2.1: Phase diagram of a typical Ising model. The dashed line indicates a first order transition. The second order transition occurs at $T_C$ and $H_C = 0$. $A, B, C, D$ represent different states of the system. The parallel arrows indicate the spin directions.

the universality class include spatial dimensionality, spin dimensionality, symmetry of the ordered state, the presence of symmetry breaking fields, and the range of interaction. The systems of the same class share the same set of critical exponents [45]. We include the elastic interaction in our system, which is a long range interaction. And according to the universality hypothesis, we expect the universality class of our compressible model to be affected by the elastic interactions.

### 2.1.3 Thermodynamic derivatives

Binder [27] showed that the maximum slope of the Binder cumulant $U_4$ at $K_C$ varies with system size as $L^{1/\nu}$, where $L$ is the system size and $K_C(L) = 1/T_C(L)$. We will explain Binder cumulant in Sec. 4.1. With a correction term, the size dependence becomes [42]:

$$\left.\frac{dU_4}{dK}\right|_{max} = aL^{1/\nu}(1 + bL^{-\omega}) \tag{2.6}$$

The logarithmic derivative of any power of the magnetization

$$\frac{\partial}{\partial K} \ln\langle m^n\rangle = \left[\frac{\langle m^n E\rangle}{\langle m^n\rangle} - \langle E\rangle\right] \tag{2.7}$$

has the same scaling properties as the cumulant slope. In our analysis we will consider the logarithmic derivatives of $\langle|m|\rangle$ and $\langle m^2\rangle$, as well as the derivative of the cumulant, to determine $\nu$.

## 2.2 Phase Separation

When a disordered, homogeneous binary alloy is quenched from high temperature to an inhomogeneous phase at a temperature below its critical temperature, the system does not order instantaneously and instead, multiple domains coexist and grow as a result of diffusion and surface tension [5]. We only consider the most familiar system: the ferromagnetic Ising model. As explained in the previous sections, we assume that the total magnetization of a system is conserved. When a system undergoes a first order transition it will divide into different regions in which one phase or the other dominates. In Fig. 2.2, the solid portions of the curve are thermodynamically stable, while the dashed portions are metastable, and the dotted portion is unstable. The end points of the unstable region are termed ‘spinodal points’ and occur at magnetizations $\pm M_{sp}$, when the magnetic field is $\pm H_C$. As the magnetic field is swept, the transition occurs at $H = 0$ and the limits of the corresponding coexistence region are at $M_{\pm}$.

Based on that, if we quench our system from a disordered high temperature state to a metastable state below the critical temperature, the system may respond differently

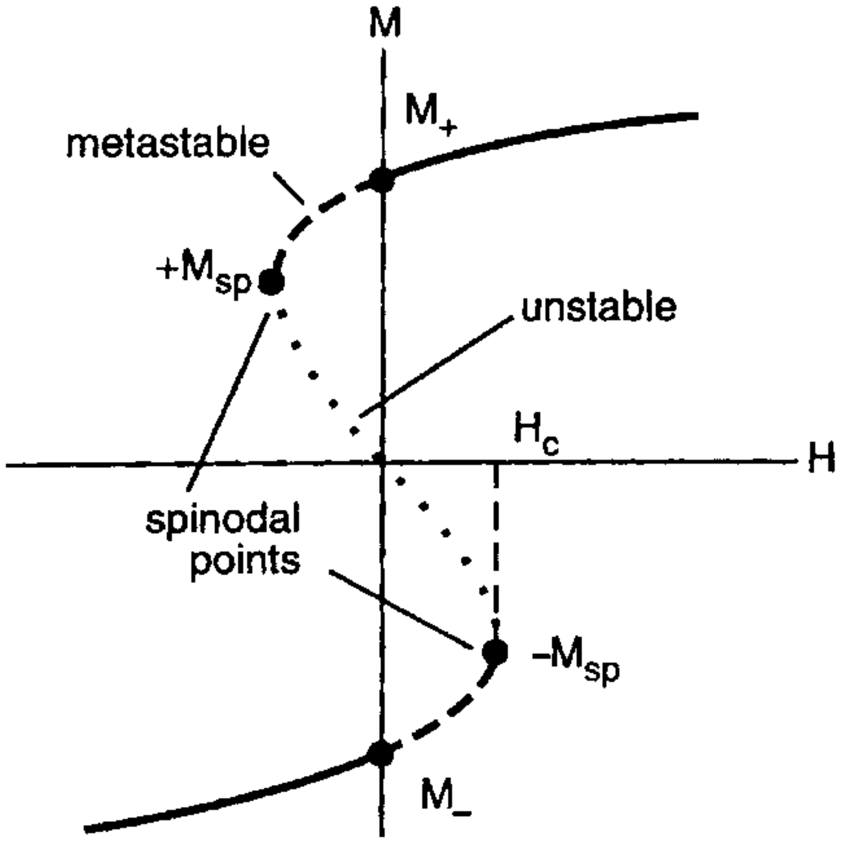

Figure 2.2: Magnetization as a function of magnetic field for $T < T_C$. The solid curves represent stable, equilibrium regions, the dashed lines represent ‘metastable’, and the dotted line ‘unstable’ states. The values of the magnetization at the ‘spinodal’ are $\pm M_{sp}$ and the spinodal fields are $\pm H_C$. $M_+$ and $M_-$ are the magnetizations at the opposite sides of the coexistence curve. Graph taken from [1].

depending on where the system is immediately after the quench. We show in the Fig. 2.3 two different cases.

We show in Fig. 2.4 the spontaneous magnetization as a function of temperature. At time $t = 0$ the system is quenched from the position either A or B at $T_o$ above the critical point $T_C$ to $T$ below $T_C$, as indicated by the arrow. If the initial state is A, then the system is quenched into the nucleation regime; otherwise, if the initial state is B, the system is in

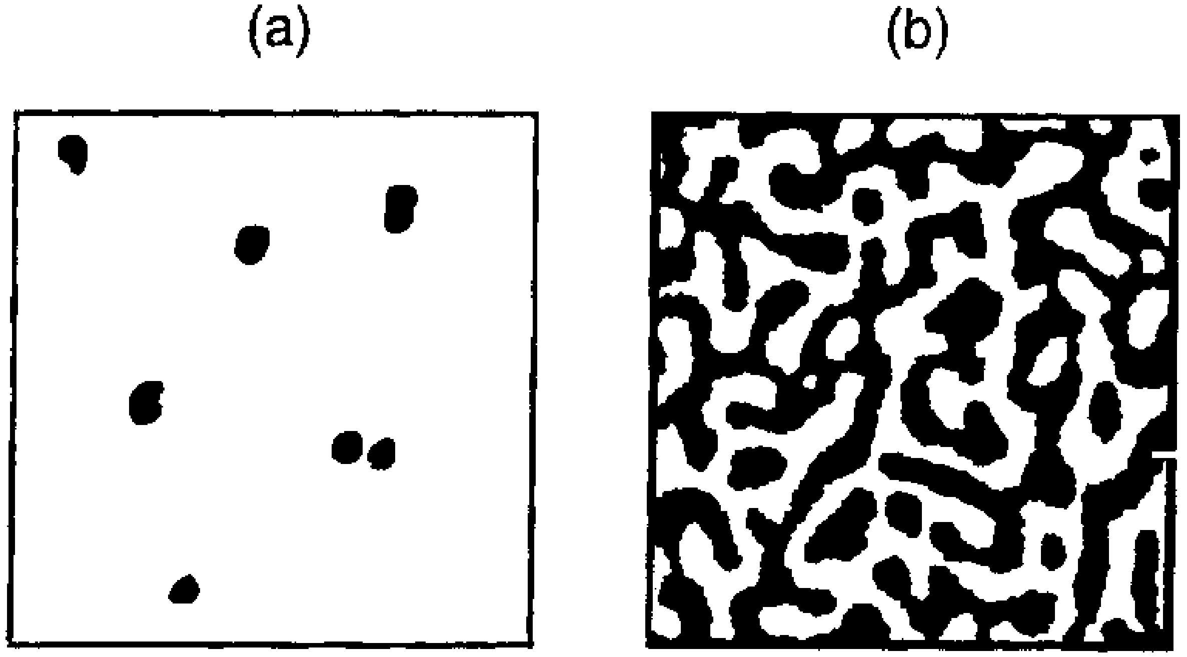

Figure 2.3: Different modes for phase separation: (a) nucleation; (b) spinodal decomposition. The dark regions represent the phase with $M_{-}$, and the white regions represent the phase with $M_{+}$. Graph taken from [1].

the spinodal decomposition regime. If the system is quenched from position A, the latter stages of this growth are thought to be described by the Lifshitz-Slyozov theory. At short times the system overcomes a nucleation barrier before droplets can grow, and at later time the process leads to a power law growth of the characteristic length scale, as shown in Eq. 1.2, where $n = \frac{1}{3}$. Their theory has been generalized qualitatively to the case 2 in which the system starts from position B by Huse [3], after which consistent results for lattice models are observed by simulations [3, 4, 7]. For a good review of phase-ordering kinetics, see Refs. [29–31]. Right after the quench, the system is in an unstable disordered state corresponding to equilibrium at temperature $T_o$ from Fig. 2.4. So the theory of phase ordering kinetics is about the dynamical evolution of the system from the initial disordered state to the final equilibrium state.

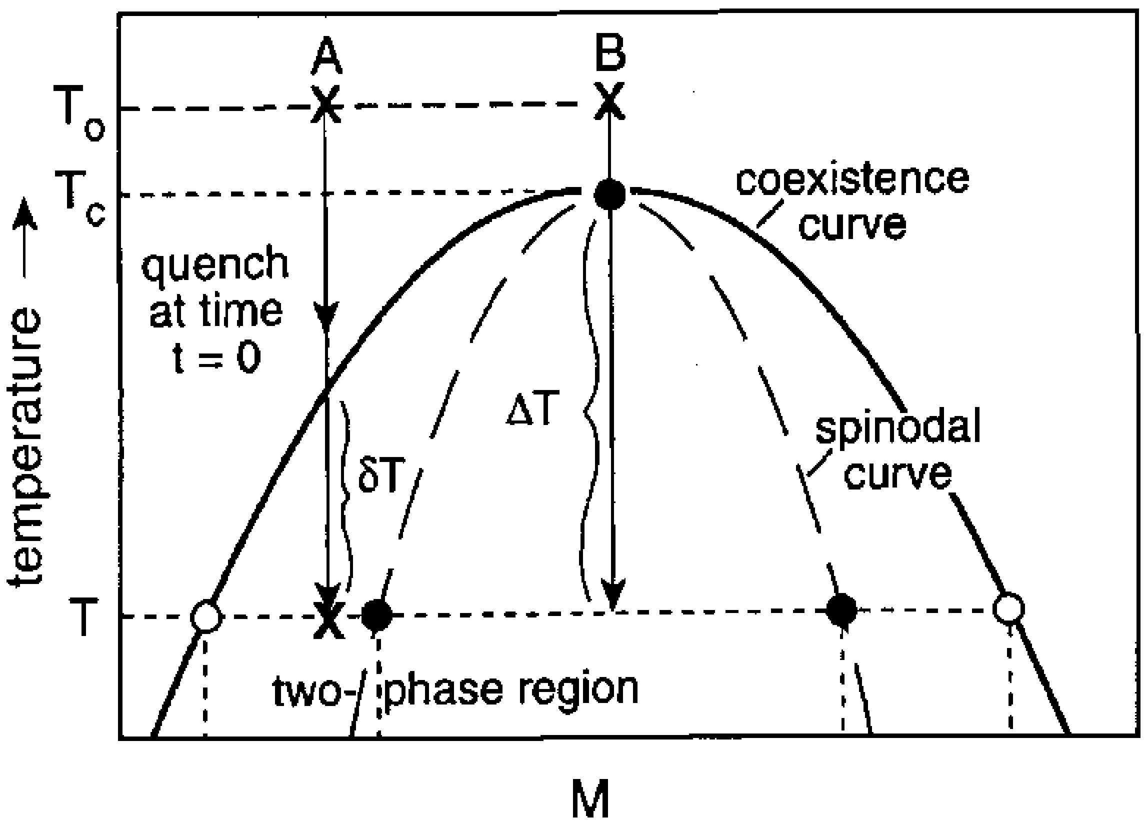

Figure 2.4: Magnetization of the Ising model in zero applied field as a function of temperature. The arrow indicates a temperature quench, at time $t = 0$, from $T_I$ to $T_F$. Graph taken from [1].

The typical length scale associated with the domains that form after a quench increases with time $t$. In the thermodynamic limit, the quenched system can never reach equilibrium due to the divergence of the relaxation time with the system size in the ordered phase [5], and this is one of the reasons that the phase ordering kinetics remains a challenge to physicists about half a century after the first theory [2] was proposed. We illustrate the domain growth behavior in Fig. 2.5. It shows a Monte Carlo simulation of a two-dimensional Ising model, quenched from $T_o = \infty$ to $T = 0$. The scaling phenomenon of domain growth is clear from the figure.

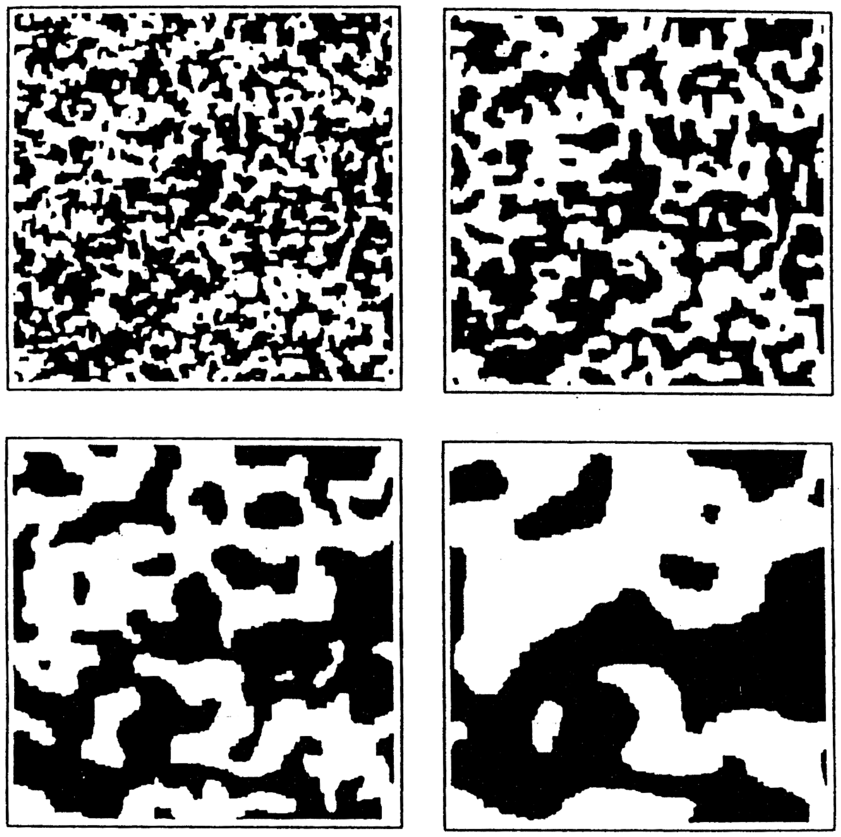

Figure 2.5: Monte Carlo simulation of domain growth in the $d = 2$ Ising model with the periodic condition at $T = 0$ (taken from Ref. [36]). The system size is $256 \times 256$, and the snapshots are taken at 5, 15, 60 and 200 Monte Carlo steps per spin after a quench from $T = \infty$.

Phase separation is commonly observed in condensed matter systems. Examples include magnets, alloys, fluids, and polymers. The late stages of growth are known as Ostwald ripening in binary alloys. However, there is one common complication of all the above systems cannot be captured by the Ising ferromagnetic with spin-flip Monte Carlo dynamics. As explained in the previous sections, we need to conserve the magnetization (or order parameter) of the Ising model as an analog of the binary alloy system. Actually, whether or not to conserve the order parameter can influence the domain growth behavior significantly. In our study, we only focus on the conserved order parameter case.

## 2.3 Domain Growth Behavior and Compressibility

A model with compressibility will be a better reflection of the real world system such as a binary alloy than a lattice model is. However, as suggested by [8–10], domain growth may be affected by compressibility. Also, as we know from recent studies [11, 13–15], the compressibility can affect the static critical behavior of Ising models and hence may affect the domain growth behavior. Mitchell and Landau [7] have studied the late-time domain growth behavior for a particular compressible 2D Ising model (to be introduced in the next chapter). They found significant deviations from the theoretically expected $n = \frac{1}{3}$ late-time growth exponent for their compressible mismatch model, but for a compressible model with no mismatch, they found only a slight deviation from $n = \frac{1}{3}$. The mismatch refers to the different preferred bond lengths for different neighboring spin pairs, we will explain it in the next chapter.

Their results suggest that the current understanding of the classes of domain growth is incomplete. So, in our research, we investigate the changes in dynamic behavior due to the flexibility of the model by a large scale ($512\times$ 512) and long time ($10^6$ MCS) Monte Carlo simulation. Different from their model, our system has no mismatch and uses a triangular-net to maintain the topology of the lattice. We use our model because we can continuously change the elasticity of the model and on the other hand, we can adjust the relative importance of the magnetic energy and the elastic energy terms. We will explain the details of our model in the next chapter.

# Chapter 3

# Model and Methods

## 3.1 The Hamiltonian

Our model is a two-dimensional, compressible, triangular Ising net. The Ising model Hamiltonian is

$$H = -J \sum_{<i,j>} e^{-\gamma r_{ij}} s_i s_j + \frac{1}{2} k \sum_{<i,j>} (r_{ij} - \ell)^2 \tag{3.1}$$

where $\sum_{<i,j>}$ is a sum over nearest-neighbor pairs of particles, $J > 0$ is the exchange energy, $r_{ij}$ is the distance between nearest-neighbor particles, $s_i$ is the spin value of the $i$th particle (+1 or -1), $\gamma$ is the coupling strength between the elastic energy and the magnetic energy, $k$ is the elastic constant, $\ell = 1$ (unit length) is the preferred bond length. Specifically, we use $J = 200$, $\gamma = 4.5$, and $k = \infty$ (rigid), $120000, 20000, 3000, 1000$ and 500.

The system is a triangular net. For the rigid case, which is the limiting case where the particles are forbidden to move, we assume that one side of the lattice lines up with the x-axis of our coordinate system, and the unit length in the x-axis is 1, the unit length in the y-axis is $\frac{\sqrt{3}}{2}$, then the system size $L \times L$ means on each row, $L$ particles evenly scatter on the lattice with total length $L$, and on the other lattice direction, there are $L$ particles evenly distributed on the lattice with total height $L \times \frac{\sqrt{3}}{2}$ in the y-axis direction. As shown in the Fig. 3.1, it represents part of the triangular Ising lattice. If there are elastic interactions between neighboring particles and the particles are allowed to move around the lattice sites, then the system is compressible. A typical triangular net is shown in Fig 3.2, where the nearest neighbor bond lengths vary and we can imagine there is a spring connecting neighboring particles.

We only include the nearest neighbor magnetic and elastic interactions in our Hamiltonian. In our simulation, the nearest neighbors of any particle are fixed, as shown in Fig 3.1, the spin $s$ has six nearest neighbors. However, for the compressible systems, the particles can move away from the lattice sites, which changes the nearest neighbor distances. In Fig 3.2 we show a compressible triangular net. In order to make sure that the six nearest neighbors are always the "nearest", we need to control the compressibility of systems. To choose appropriate elastic constant $k$, we compare the bond length distribution for the nearest and the 2nd nearest neighbors. In Fig 3.3-3.6 we show the bond length distribution for the systems with $k = 500$ and $k = 3000$ before and after the critical quench.

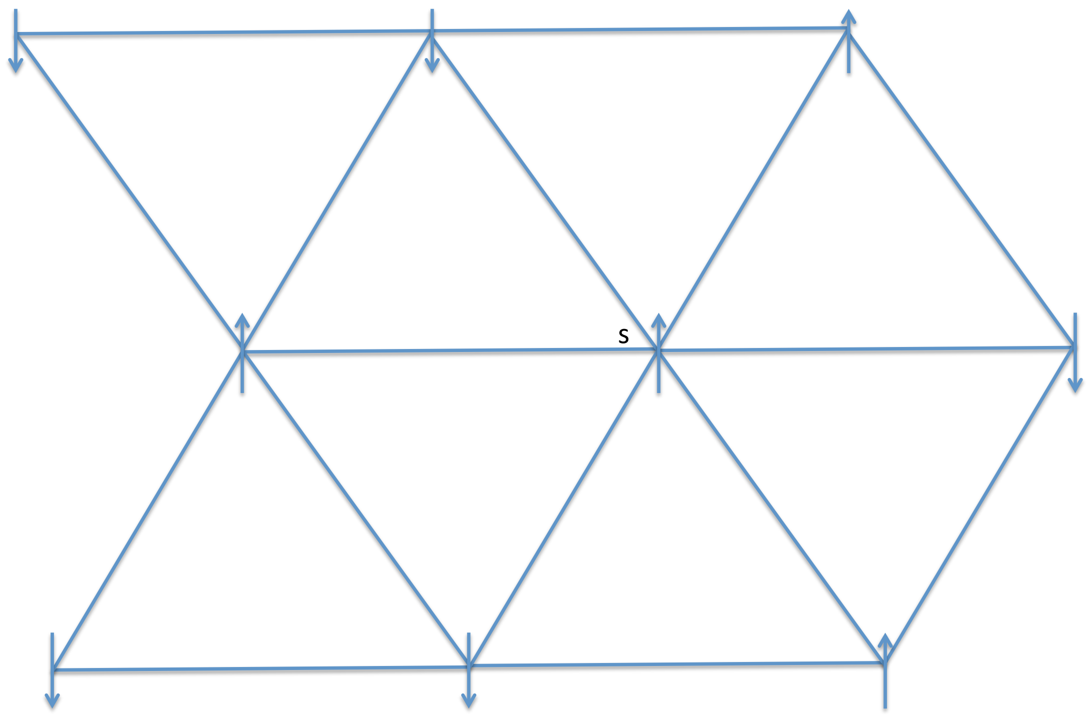

Figure 3.1: Part of a triangular Ising lattice. Spin $S$ as shown in the graph has six nearest neighbors.

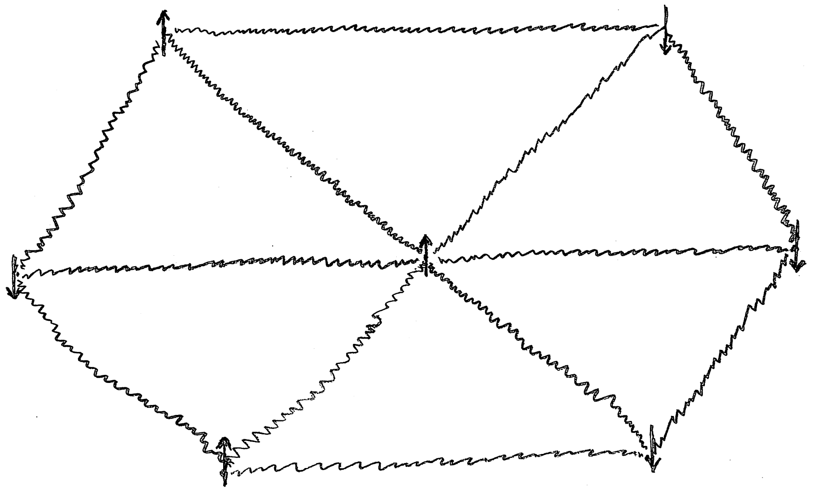

Figure 3.2: Part of a triangular Ising net for the compressible systems.

We observe that the nearest neighbor and the 2nd nearest neighbor clearly follow the normal distributions centered at different points. The overlapped region of them are narrow. For $k = 500$, at the high temperature, the triangular net is distorted more due to its high compressibility, so the overlapped region is wider compared to other cases. But only very a few 2nd nearest neighbor bond lengths are probably smaller than those nearest neighbor bond lengths. We can still stick to our inclusion of only the nearest neighbor interactions. At the low temperature, the nearest and 2nd nearest neighbor bond length distributions are more separated as we expect. After the quench, the bond length distribution doesn't change with time.

Also the system size of the compressible systems are allowed to increase or shrink. We use $\Lambda_x$ and $\Lambda_y$ to represent the system size in the x and y-direction. By using this model we can continuously change the elasticity of the model and on the other hand, we can adjust

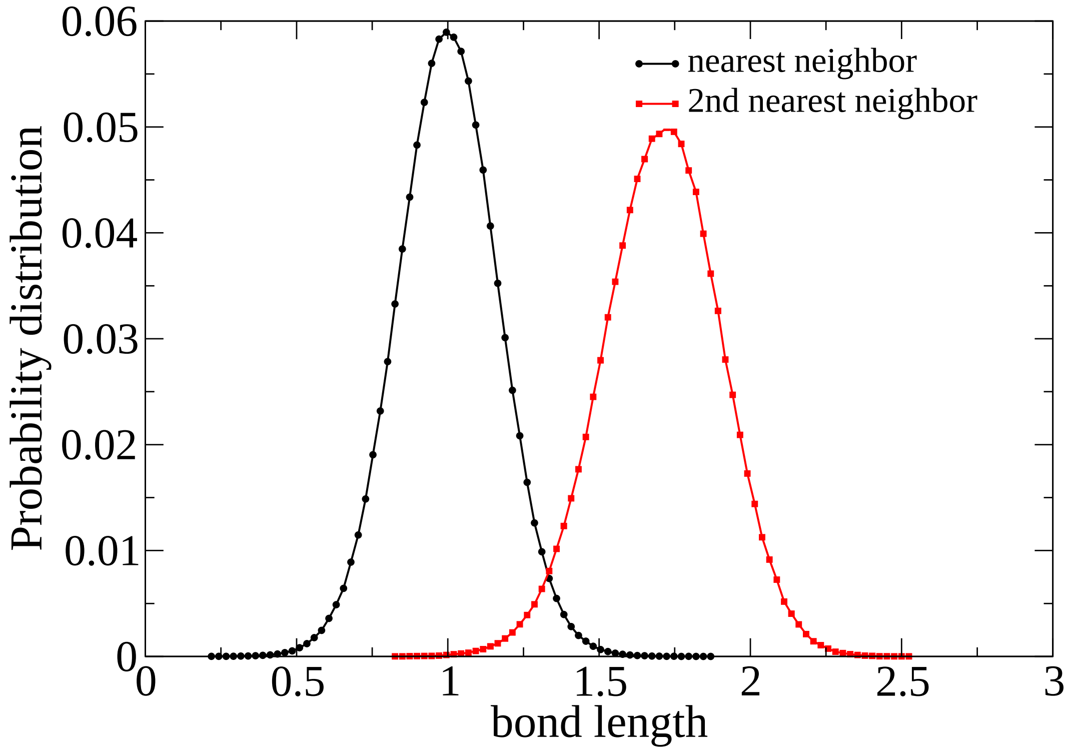

Figure 3.3: Nearest neighbor and 2nd nearest neighbor bond length distribution for the system with $k = 500$ when it is in the equilibrium at $t = 5000$ MCS under a high temperature $T > T_C$.

the relative importance of the magnetic energy and the elastic energy terms. This is different from the model used in Mitchell and Landau's paper [7]. They use a two-dimensional square-net Ising model in which $L^2$ Ising spins have continuous positions within a periodic box of size $L \times L$. The Hamiltonian of their model is:

$$H = \sum_{<i,j>} f(r_{ij}, s_i, s_j) + J_\theta \sum_{<i,j,k>} \cos^2(\theta_{ijk}) \tag{3.2}$$

where $J_\theta$ is the bond angle stiffness in dimensionless units and $\sum_{<i,j,k>}$ is the sum over bond angles $\theta_{ijk}$ (four per particle), where $i$ and $k$ are nearest neighbors of $j$. The second term

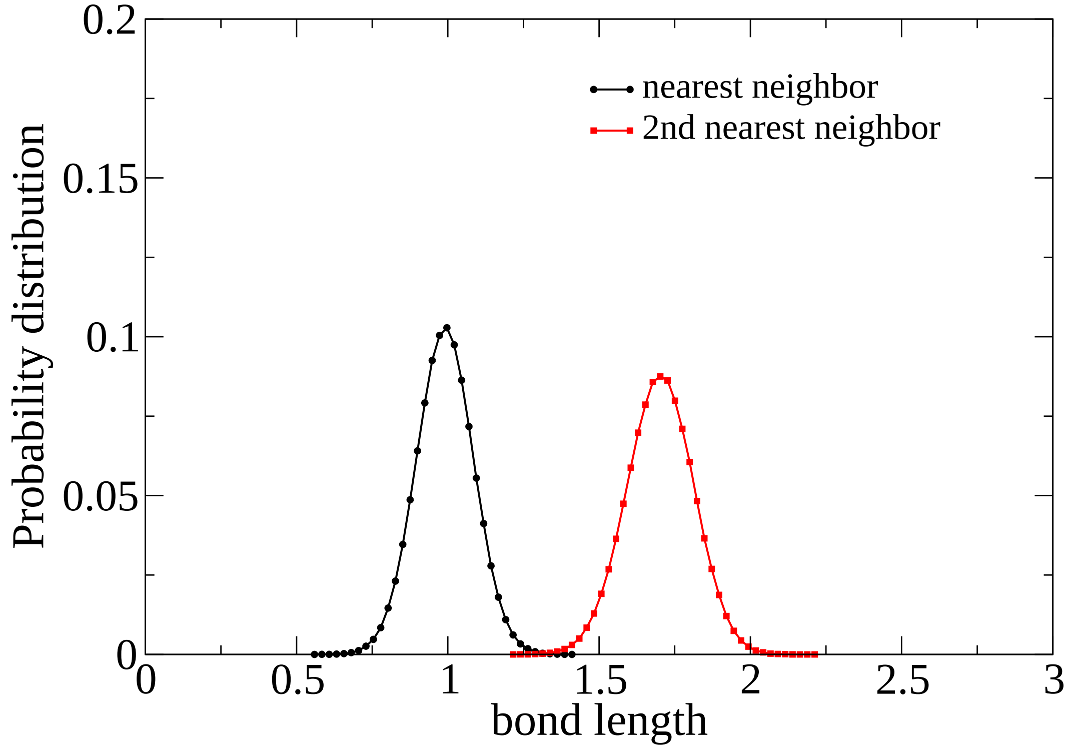

Figure 3.4: Nearest neighbor and 2nd nearest neighbor bond length distribution for the system with $k = 500$ when it is at $t = 100,000$ MCS after the critical quench.

is needed to stabilize an ordered square lattice structure. $f(r, a, b)$ is the nearest neighbor interaction potential which is Lennard-Jones-like:

$$f(r, a, b) = J_{ab}[(\frac{\ell_{ab}}{r})^{12} - 2(\frac{\ell_{ab}}{r})^{6}] \tag{3.3}$$

where $r$ is the displacement between two nearest neighbors with spin values $a$ and $b$, $\ell_{ab}$ is the preferred bond length. For instance, $\ell_{++}$ is the preferred bond length for two neighboring positive spins and $\ell_{+-}$ is the preferred bond length for a neighboring spin pair with different signs. $J_{++}$ is the exchange energy between two positive neighboring spins and $J_{+-}$ is the exchange energy between one positive spin and its negative neighboring spin. $J_{++} = J_{--} =$

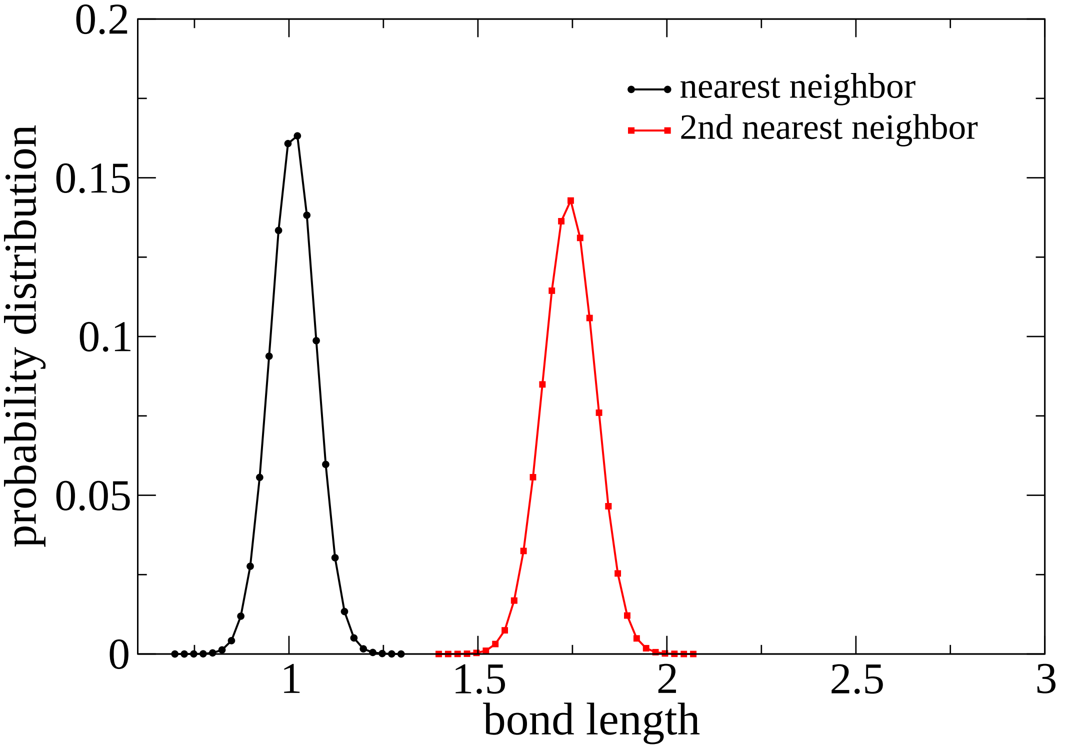

Figure 3.5: Nearest neighbor and 2nd nearest neighbor bond length distribution for the system with $k = 3000$ when it is in the equilibrium at $t = 5000$ MCS under a high temperature $T > T_C$.

30, $J_{+-} = J_{-+} = J_{++} - 2$, and $\ell_{+-} = \ell_{-+} = 1$. With these interactions, when $\ell_{ab} = 1$ and $J_{++}$ and $J_\theta \to \infty$, the Hamiltonian reduces to that of the common ferromagnetic rigid Ising model. They consider three specific cases of the model: the rigid two-dimensional Ising model; a symmetric model, where $\ell_{ab} = 1$ is referred to as the 0% mismatch model; and a 4% lattice mismatch model, where $\ell_{++} = 1.02$ and $\ell_{--} = 0.98$. The 4% mismatch value was chosen to give insight into phase separation in Si-Ge alloys [11, 13, 14].

There are many differences between these two models. We use a triangular net in our model to maintain the structure of the system due to the stability of the triangles, and

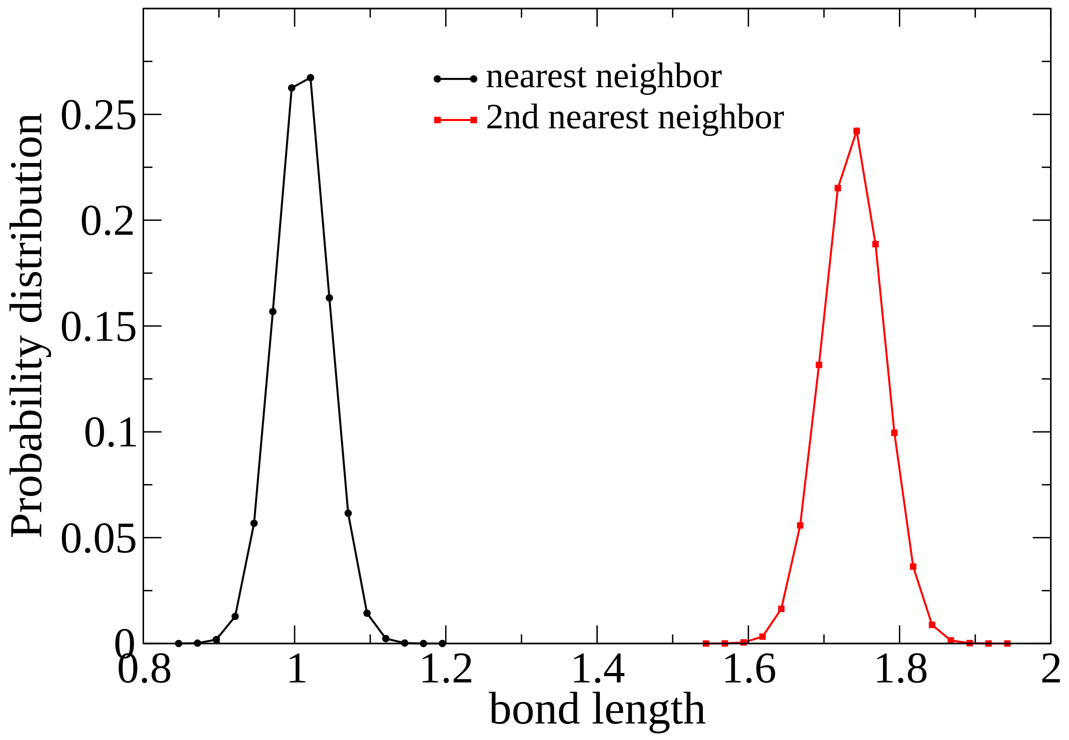

Figure 3.6: Nearest neighbor and 2nd nearest neighbor bond length distribution for the system with $k = 3000$ when it is at $t = 100,000$ MCS after the critical quench.

Mitchell and Landau's model use the angle bond interactions to stabilize the square net structure. Our model explicitly includes the elastic energy and this enables us to directly observe how the domain growth exponent $n$ changes with different elastic constants $k$, which helps us find functional relationship between them if there exists one. However, in our study, we only focus on the rigid system and the simple compressible systems, although our model can be easily generalized to include the lattice mismatch by assigning different $\ell$ value to different neighboring spins pairs.

## 3.2 Monte Carlo Simulation

Monte Carlo methods are often used in computer simulations of physical and mathematical systems. The most obvious feature of a Monte Carlo algorithm is to take random samplings to compute the interesting results. It was invented in the 1940s by John von Neumann, Stanislaw Ulam and Nicholas Metropolis [34], while they were working on nuclear weapon projects (Manhattan Project) in the Los Alamos National Laboratory. In physics, the concept of chance comes into play when the model evolves in a stochastic manner, and in such cases, a sequence of random numbers is needed in the simulation. Monte Carlo methods usually follow the pattern below:

1. Define possible inputs for the system to be studied.
2. Generate inputs randomly from the probability distribution over the range of inputs.
3. Perform a deterministic computation on the inputs.
4. Aggregate and analyze the results.

In physical simulations, the Metropolis importance sampling method is one of the most important and extensively used algorithms, and this will be described in the next section.

## 3.3 The Metropolis Algorithm

The Metropolis method [33] generates a new state from a previous state using a transition probability which depends on the energy difference between the previous and the new state. In such a way a sequence of states can be produced following the master equation below:

$$\frac{\partial P_n(t)}{\partial t} = -\sum_{n \neq m} P_n(t) W_{n \to m} - P_m(t) W_{m \to n} \tag{3.4}$$

where $P_n(t)$ is the probability of the system being in the state $n$ at time $t$, and $W_{n \to m}$ is the transition rate for $n \to m$, where $m$ represents another system state. A sufficient condition for equilibrium is to make the following so called "detailed balance" happen:

$$P_n W_{n \to m} = P_m W_{m \to n} \tag{3.5}$$

This is because in equilibrium we have the left hand side of Eq. 3.4 as 0, and the Eq. 3.5 is a sufficient condition to make the right hand side of Eq 3.4 as 0. For a fixed temperature $T$, a classical system following Boltzmann distribution in its equilibrium states must have:

$$P_n = \exp\left[-\mathcal{H}_n / k_B T\right] / Z \tag{3.6}$$

where $Z$ is the partition function and $k_B$ is the Boltzmann constant. So we get

$$\frac{W_{n \to m}}{W_{m \to n}} = \frac{P_m(t)}{P_n(t)} = \exp\left(-\Delta\mathcal{H} / k_B T\right) \tag{3.7}$$

$\Delta\mathcal{H}$ is the energy change $(\mathcal{H}_m - \mathcal{H}_n)$. Any transition rate which satisfies detailed balance is acceptable. One of the most popular methods used in statistical physics is the Metropolis form described as the following (Again, we take the Ising model as an example):

1. Choose an initial state
2. Choose a site $l$
3. Calculate the energy change $\Delta\mathcal{H}$ which results if the spin update/change proposed at site $l$ is accepted
4. Generate a random number $r$ such that $0 < r < 1$ and $r$ is uniform in the interval.
5. If $r < \exp\left(\frac{-\Delta\mathcal{H}}{k_B T}\right)$, update the spin state in the proposed way
6. Go the next site and go to (3)

After a set number of spins have been considered, the properties of the system are determined and added to the statistical average which is being kept. Note that the random number $r$ is chosen uniformly in the interval [0,1], and successive random numbers should be uncorrelated. In our simulation, spins are randomly chosen $L \times L$ times before we call it a "Monte Carlo Sweep" or "Monte Carlo Step", which is a measure of Monte Carlo time. Note that in one Monte Carlo Step, there could be some spins that have been chosen for multiple times, and there could also be some spins that have never been chosen. After a long enough Monte Carlo Sweeps, the system is in its equilibrium state following the Boltzmannn distribution. Then we can take samples of the physical quantities we are interested in and analyze them. Specifically, the statistical mechanical average of any physical quantity $A$, which we write it as

$$< A >= \sum_{n=1}^{N} P_n A_n \tag{3.8}$$

will be calculated simply as the arithmetic average of all samples generated through the above Metropolis procedure, as shown below.

$$< A >= \frac{1}{N} \sum_{n=1}^{N} A_n \tag{3.9}$$

This is because the states we generated through Metropolis method follows the Boltzmannn distribution, which assigns different weights to different $A_n$ such that the arithmetic average Eq. 3.9 approximates Eq. 3.8 very well.

It is worth mentioning that although the above recipe applies to the traditional Ising model only, we will apply the algorithm to our model with spin-exchanges and spin lateral displacement instead of spin-flips only. We will explain this in more detail in Sec. 5.1.

## 3.4 Spin-Exchange Sampling

The spin-exchange or Kawasaki method [32] conserves the system's magnetization. Instead of considering a single spin which may change its orientation, one chooses a pair of spins and allows them to attempt to exchange positions. One examines the interacting nearest neighbors of both spins in the pair and determines the change in energy if the spins are interchanged. This energy difference is then used in the acceptance procedure described above.

## 3.5 Method for Constant Pressure

B. Dünweg and D. Landau [11] have proposed a method to maintain constant pressure in a Monte Carlo simulation. For a two dimensional system as an example, one randomly chooses new system sizes $\Lambda_x^{'}$ and $\Lambda_y^{'}$. In order to keep this trial configuration homogeneous, correspondingly, the new spin coordinates must be $x_i^{'} = (\Lambda_x^{'}/\Lambda_x)x_i$, etc. The spin values remain unchanged. Then the energy change associated with this global distortion of the system is, however, not the only quantity entering the Metropolis acceptance criterion. Instead, we have to use

$$\Delta\mathcal{H}_{eff} = \Delta\mathcal{H} - Nk_BT\ln\frac{\Lambda_x^{'}\Lambda_y^{'}}{\Lambda_x\Lambda_y} \tag{3.10}$$

The first term includes the energy change before the global rescale, and the latter term denotes the change in translational entropy when the volume changes.

## 3.6 Histogram Reweighting

Monte Carlo study may require huge computational resources sometime, especially when the system is at a temperature close to $T_C$. The divergence of correlation times at around $T_C$ necessitates an extraordinarily long time to obtain enough independent samples for reducing the standard error. So the efficiency of the simulation are of major importance. Various methods appear as tools to increase the efficiency. Histogram reweighting is one of them. It was proposed by Ferrenberg *et al* [42] to increase the amount of information obtained from a simulation. The technique uses restricted-energy MC simulations to generate the partition function for a range of parameter values.

Consider the Ising model in a magnetic field. The Hamiltonian for this system is:

$$-\beta\mathcal{H} = K\sum_{<i,j>} s_i s_j + h\sum_i s_i = KS + hM \tag{3.11}$$

where $K$ is the dimensionless coupling constant $(J/k_BT)$ and $h$ is an applied magnetic field $(H/k_BT)$. The probability distribution of $S$ and $M$ at a point in the parameter space$(K, h)$ is given by:

$$P_{K,h}(S, M) = \frac{1}{Z(K, h)} N(S, M)\exp(KS + hM) \tag{3.12}$$

where $N(S, M)$ is the number of configurations at the point $(S, M)$ in the phase space, and $Z(K, h)$ is the canonical partition function given by:

$$Z(K, h) = \sum_{S,M} N(S, M)\exp(KS + hM). \tag{3.13}$$

The normalized probability distribution with new parameters $(K', h')$ can be described in terms of $(K, h)$ in the following way:

$$P_{K',h'}(S, M) = \frac{P_{K,h}(S, M)\exp[(K' - K)S + (h' - h)M]}{\sum_{S,M} P_{(K,h)}(S, M)\exp[(K' - K)S + (h' - h)M]} \tag{3.14}$$

Using this reweighted probability distribution $P(S', M')$, we can calculate new thermodynamic quantities based on the data given from $(S, M)$.

## 3.7 Simulation

### 3.7.1 The Monte Carlo Moves

In the Monte Carlo simulation of the system's kinetic behavior, we use the same types of Monte Carlo moves as in [7]: a spin-exchange (Kawasaki dynamics), lateral displacement of the particle's position, and a global volume rescale to maintain a constant pressure. In the spin-exchange attempt a nearest-neighbor pair of spins is randomly chosen and spin values are exchanged and in the spin-move attempt a particle is randomly selected and displaced by a small random amount from its current position. In particular, the particle is allowed to move within the circle of the radius $0.3\sqrt{x^2+y^2}$ centered at the particle's current position, where $x$ and $y$ are the average neighboring distance in the $x$ and $y$ directions. The radius of the circle is chosen as such to be not too large to lower the acceptance rates and not too small to be inefficient to move the particles. The energy change due to global rescale is used to reproduce a constant pressure ensemble [11]. And in our simulation the system size is allowed to increase or shrink by up to 0.25% of its current size in both directions. The energy term to maintain constant pressure is not shown in Eq. 3.1. One Monte Carlo Sweep (MCS) consists of one attempted volume change followed by $L^2$ exchange or lateral displacement attempts, where the probability to choose exchange or displacement was 50%. The 50% chance to do displacement or spin-exchange is chosen to be consistent with previous research [7] and for the convenience in comparison of the results. For the rigid model, we only include the spin-exchange moves.

As a critical quench simulation, we initialize each simulation with randomly chosen 50% spin ups and 50% spin downs, and the Kawasaki dynamics used in our algorithm conserves the total magnetization.Then the system is equilibrated at a high temperature at $\sim 1.76T_C$ and then quenched down to $\sim 0.59T_C$. These temperatures are taken to be consistent with previous studies [3, 7] and then for the convenience of comparing the results. Actually $0.59T_C$ at which the domains grow is appropriate because it is high enough to allow the domains grow fast and low enough to avoid big thermal fluctuations. The system is under constant, zero pressure. We use $L = 512$ as system size with the periodic boundary condition and run $10^6$ MCS after quench, the reason why we do a large scale $(512 \times 512)$ simulation to such a late time $(10^6)$ is to be explained in the following subsection. Every $10^3$ MCS we record the configuration for analysis. The sample interval $10^3$ is taken to be the same as the previous study [7] for comparison concern. For each simulation of $10^6$ MCS after quench, about 25 days are needed for compressible models on quad-core Intel Xeon with 2GB RAM/CORE. Due to limited computation resources, we also take samples only up to $3 \times 10^5$ MCS after quench and combine them with those $10^6$ MCS samples for analysis. The total CPU time is about $8 \times 10^5$ CPU hours. We use the R1279 generator, one variant of generalized feedback shift-register generator, for random number generation in our simulation. We also test other random number generators (e.g. the R250 generator, another variant of generalized feedback shift-register generator.) and smaller systems ($L = 256$), and get consistent results to within error bars.

On the other hand, to study the static critical behavior of the systems, instead of using Kawasaki dynamics, we use the spin-flip and lateral displacement with the global volume rescale in Metropolis algorithm.

### 3.7.2 Critical Quench

As illustrated in Fig. 2.4, the system follows the path directed by the arrow starting from B. In other words, each simulation begins with a random spin state, where the value of the spin is randomly chosen to be $+1$ or $-1$ with 50% probability, and our Monte Carlo algorithm conserves the total magnetization at this initial value. The system is then equilibrated at a temperature much higher than the critical temperature ($T_C$). After equilibration, the temperature is quenched down to about $0.59T_C$, and the configurations are recorded and analyzed every $10^3$ MCS.

## 3.8 Correlation Function

To quantify the phase separation behavior, we calculate and analyze the spatial correlation function; however, because the compressible models have continuous particle positions, methods developed for regularly spaced particles could not be used. Therefore, the correlation function was calculated by direct summation over the lattice in only three directions,

$$C(r) = C_{(p,q)}(r) = \langle s_i s_j \rangle \tag{3.15}$$

where $(p, q)$ denote one of the three displacement directions: the lattice directions: $(1, 0)$ and $(0, 1)$, and the another side direction of the triangles in the net: $(1, 1)$. We show these directions in the Fig 3.7. $\langle s_i s_j \rangle$ are the average over all pairs of particles with a distance of

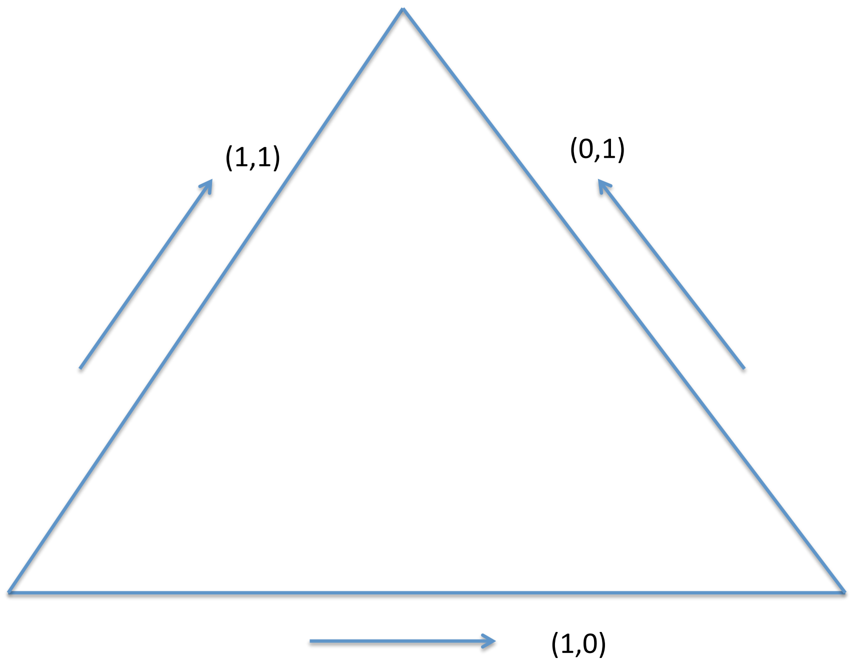

Figure 3.7: Lattice directions of the triangular net.

$r$ between each other. Since the system's configuration evolves with time, at different time points, we have different $C(r)$ curves. The scale that we use to describe the domain size, $\xi$, is defined as the first-zero crossing of $C(r)$ and is found by fitting a second order polynomial to the three points closest to the crossing. The first-zero crossing is the $r$ value at where $C(r)$ drops to 0 the first time ever. Since we have different $C(r)$ curves at different times, the first-zero crossing $\xi$ also varies with time. So the domain size $R(t)$ in Eq. 1.2, in our simulation, is represented by $\xi(t)$.

For the rigid system, $r$ is simply a multiple of 1. Because the lattice spacing between any two neighboring pairs in the three directions are always a fixed number, which is 1. For the compressible systems, the distance between particles changes. We cannot fix the distance $r$ and calculate $C(r)$ directly. So we do the following to calculate the correlation function. For the $(p, q)$ direction, to calculate $C_{(p,q)}(r)$, we first define the scale in $(p, q)$ direction. It is the average neighboring distance in the $(p, q)$ direction, let us call it $d_{(p,q)}$. Then we scan through the net, only consider those particles whose lattice sites are lined up in the $(p, q)$ direction. For any two of them, assuming they have a distance $r_0$, we assign an index $i = \lfloor (r_0 + 0.5)/d_{(p,q)} \rfloor$. The $\lfloor x \rfloor$ operation is to take the largest integer that is not greater than $x$. The added term 0.5 is intendedly given to round up the value. Finally, for each fixed index number $i$, we have different $r_0$'s corresponding to it, and we take the average $r_{avg}$ of them and output $r_{avg}$ and $C(r_{avg})$ as the correlation function values in the $(p, q)$ direction.

$C(r)$ is not self-averaging. That means, we cannot reduce its standard error by increasing the size of the system. So meaningful results are only obtained if one averages the simulated "quenching experiment" over a large number of independent runs. In our study, it is necessary to study the late-time behavior where $R(t)$ is much larger than the lattice spacing, but $R(t)$ must also be much smaller than the lattice size $L$, because otherwise one runs into finite size effects. In a word, we need to do a large-scale simulation with a good number of samples to study the late-time domain growth behavior. In particular, we use the system size of $512 \times 512$ and do simulation up to $10^6$ MCS.

## 3.9 Smoothing Splines

In our local exponent analysis of the domain growth exponents in Sec. 5.4, we observe large fluctuations. In order to better observe the overall trend of the domain growth exponents, we use smoothing splines to fit the data.

The word "spline" has been originated from the ship building industry. It was a thin strip of wood used by draftsmen as a flexible French curve. Metal weights were placed on the drawing surface and the spline was threaded between the ducks. As a natural extension of this concept of spline, a spline curve in industry is referred to as a sequence of curve segments connected to each other to form a single continuous curve. And the Basic-Spline, abbreviated as B-Spline, is spline functions that has minimal support with respect to a give degree, smoothness, and domain partition. A curve $s(u)$ is called a *spline of degree* $n$ with the *knots* $a_0, ...a_m$, where $a_i \leqslant a_{i+1}$ and $a_i < a_{i+n+1}$ for all possible $i$. And $s(u)$ is $n - r$ times differentiable at any $r - fold\ knot$. (We call a knot $a_{i+1}$ the *r-fold knot* if $a_i < a_{i+1} = \cdots = a_{i+r} < a_{i+r+1}$). It is also common to refer to a spline of degree $n$ as a spline of order $n + 1$.

### 3.9.1 B-Spline

In order to define B-splines, let $(a_j)_{j=-1}^{N}$ be a series of knots on interval $[a, b]$, which satisfies

$$a < a_1 < ... < a_j < a_{j+1} ... < a_N < b.$$

Suppose we have $N$ interior knots and the B-spline is of order $d$ (or degree $d - 1$), we can set the following boundary knots

$$a_{-(d-1)} = ... = a_{-1} = a_0 = a < a_1 < ... < a_N < b = a_{N+1} = ... = a_{N+d},$$

thus we have $N + 2d$ knots in total. With the above knots, we define B-spline of order $d$ (or degree $d - 1$), $B_{j,d}$, using the following recursion formula:

$$B_{j,1}(x) = \begin{cases} 1 & \text{if } x \in [a_j, a_{j+1}); \\ 0 & \text{o.w..} \end{cases}$$

and

$$B_{j,d}(x) = \frac{(x - a_j)B_{j,d-1}(x)}{a_{j+d-1} - a_j} + \frac{(a_{j+d} - x)B_{j+1,d-1}(x)}{a_{j+d} - a_{j+1}}.$$

Now suppose we have 5 equally spaced interior knots on the interval $[0, 6]$, then we have the following constant spline basis functions:

$$\begin{aligned}
B_{0,1}(x) &= \begin{cases} 1 & \text{if } 0 \leq x < 1; \\ 0 & \text{o.w..} \end{cases} \\
B_{1,1}(x) &= \begin{cases} 1 & \text{if } 1 \leq x < 2; \\ 0 & \text{o.w..} \end{cases} \\
&\vdots \\
B_{5,1}(x) &= \begin{cases} 1 & \text{if } 5 \leq x < 6; \\ 0 & \text{o.w..} \end{cases}
\end{aligned}$$

Based on those first-order basis functions, we deduce the linear basis functions $B_{j,2}(x)$ of order 2 as:

$$\begin{aligned}
B_{0,2}(x) &= (1-x)B_{1,1}(x) \\
B_{1,2}(x) &= xB_{1,1}(x) + (2-x)B_{2,1}(x) \\
B_{2,2}(x) &= (x-1)B_{2,1}(x) + (3-x)B_{3,1}(x) \\
B_{3,2}(x) &= (x-2)B_{3,1}(x) + (4-x)B_{4,1}(x) \\
B_{4,2}(x) &= (x-3)B_{4,1}(x) + (5-x)B_{5,1}(x) \\
B_{5,2}(x) &= (x-4)B_{5,1}(x)
\end{aligned}$$

We repeat the pattern up to any desired degree. The degree of 3 (cubic spline) is used in most cases, since low degree may cause unsmooth curve while too high degree may result in over-fitting. Degree of 3 is a good compromise between them.

Next we introduce the definition of B-spline space. Let $S_n^{(d)}$ be the space of B-splines on $[a, b]$ of order $d \geq 1$. The space $S_n^{(d)}$ consists of all the functions $s$ satisfying

- $s$ is a polynomial of degree $d-1$ which we fix on each of the subintervals $[a_j, a_{j+1})$, where $k = 0, ..., N-1$.
- $s$ is $d-2$ continuously differentiable on the interval $[a, b]$, for $d \geq 1$.

Fig. 3.6 is a sequence of B-splines up to order four with ten knots evenly spaced from 0 to 1.

$$B_{j,d}(x) = \frac{(x - a_j)B_{j,d-1}(x)}{a_{j+d-1} - a_j} + \frac{(a_{j+d} - x)B_{j+1,d-1}(x)}{a_{j+d} - a_{j+1}}.$$

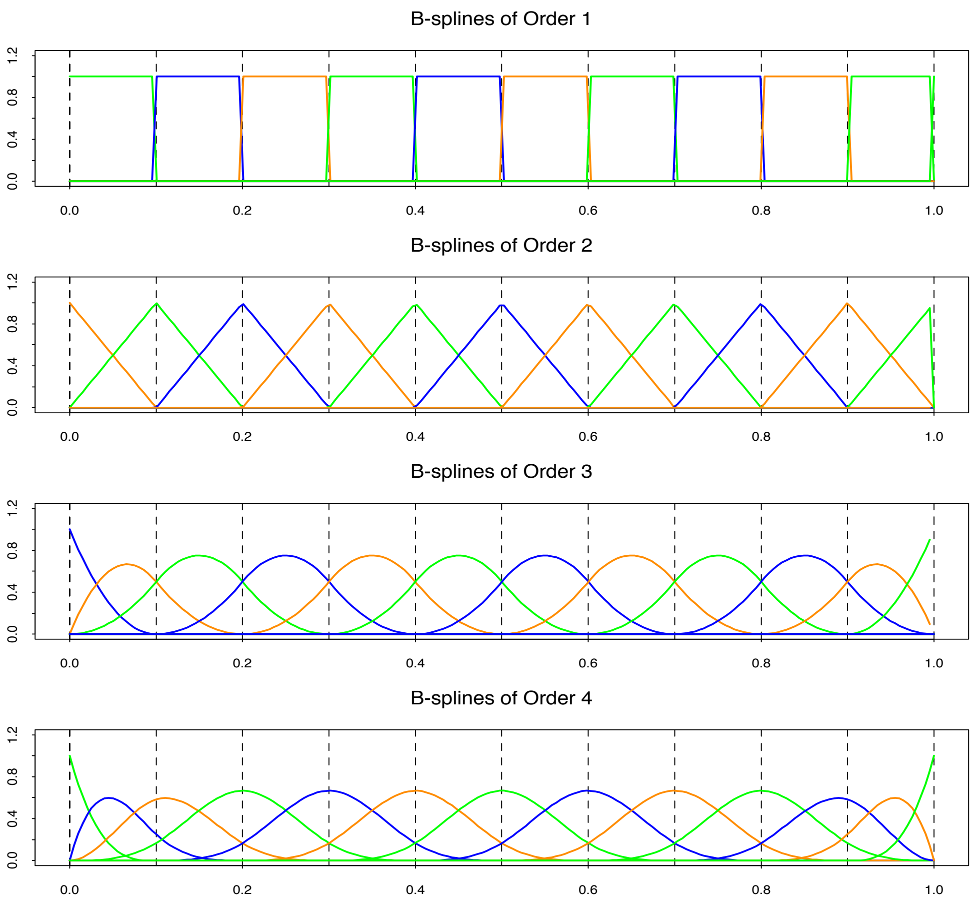

Figure 3.8: The sequence of B-splines up to order four with ten knots evenly spaced from 0 to 1. Graph taken from [37].

### 3.9.2 Smoothing Splines

Let's consider the following problem: among all functions $f(x)$ with two continuous derivatives, find one that minimizes the penalized residual sum of squares:

$$RSS(f, \lambda) = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int f''(t)^2 \mathrm{d}t \tag{3.16}$$

where $\lambda$ is a fixed *smoothing parameter*. The first term is a measure of how close the fitted data to the original data, and the second term controls the complexity of the model by penalizing curvature in the function. The parameter $\lambda$ assigns weight to the second term so as to establish a trade-off between them. As it clearly indicates, we have two special cases:

1. $\lambda = 0$: $f$ can be any function that interpolates the data.
2. $\lambda = \infty$: the least squares linear fit because no second derivative can be tolerated.

So by varying $\lambda \in (0, \infty)$, we end up with fitting functions of different smoothness. Then if we let the solution to be a natural spline, we can write it as:

$$f(x) = \sum_{j=1}^{N} N_j(x)\theta_j \tag{3.17}$$

where the $N_j(x)$ are an $N$-dimensional set of basis functions for representing this family of natural splines. So the above criterion reduces to:

$$RSS(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\mathbf{\Omega}_N\theta \tag{3.18}$$

where $\mathbf{N}_{ij} = N_j(x_i)$ and $\mathbf{\Omega}_{\mathbf{N}ij} = \int N_j^{''}(t)N_k^{''}(t)dt$. The solution is:

$$\hat{\theta} = (\mathbf{N}^T\mathbf{N} + \lambda\Omega_N)^{-1}\mathbf{N}^Ty \tag{3.19}$$

and the fitted smoothing spline is given by

$$f(x) = \sum_{j=1}^{N} N_j(x)\hat{\theta}_j \tag{3.20}$$

In our analysis, we use cubic B-spline basis functions for $N_j(x)$ and so we call it cubic smoothing splines fitting.

# Chapter 4

# Results: Static Critical Behavior

## 4.1 Binder Cumulant Crossing

Binder cumulant, the fourth order cumulant of the order parameter introduced by Binder [27], provides important information on a system's critical behavior. In zero field, for an Ising model, the fourth order cumulant reduces to the following equation.

$$U_4 = 1 - \frac{\langle m^4 \rangle}{3\langle m^2 \rangle^2} \tag{4.1}$$

where $\langle m^2 \rangle$ is the average of the 2nd order moment of magnetization of one single particle, and $\langle m^4 \rangle$ is the average of its 4th order moment. For Ising models, in the thermodynamic limit, the cumulant $U_4$ approaches $\frac{2}{3}$ at $T < T_C$, and when $T > T_C$, $U_4 = 0$. Curves of difference system sizes for $U_4$ cross at the critical point with a fixed point value $U_4^*$, which assumes different values for different universality classes. For example, for the two-dimensional rigid Ising model, according to [38–40], the $U_4^*$ is found to be $\sim 0.61$. For the mean-field universality class, $U_4 \sim 0.27$ [41], and for the three-dimensional Ising model, $U_4 \sim 0.47$ [42].

We investigate systems with different elasticities; in particular, we study the rigid model and the compressible models with $k = 120000$, $k = 20000$, $k = 3000$, $k = 1000$ and $k = 500$ (See Eq. 3.1 for the description of $k$.). Fig. 4.1-4.6 show $U_4$ as a function of $T$ for different size systems for these models. Temperatures are normalized by the ground state interaction $Je^{-\gamma\ell}/k_B$. For the rigid model, the $T_C$ estimated from our simulation is $\sim 8.09$, which is consistent with the critical temperature ($\sim 3.64$) for the 2D rigid triangular-lattice Ising model in Ref. [48] with a conversion of dimension. All of these figures show that the crossing points $U_4^*$ vary very slightly with $L$ and $\sim 0.61$. These results are consistent with the $U_4^*$ of two-dimensional rigid Ising models. The magnetic field $H_C = 0$ because the hysteresis loop forms equivalent area around $H_C = 0$ at $T < T_C$. So we conclude that even with elasticities, the systems used in our simulation belong to the same universality class of the two-dimensional rigid Ising model. We plot in Fig. 4.7 the critical temperatures as a function of $1/k$.

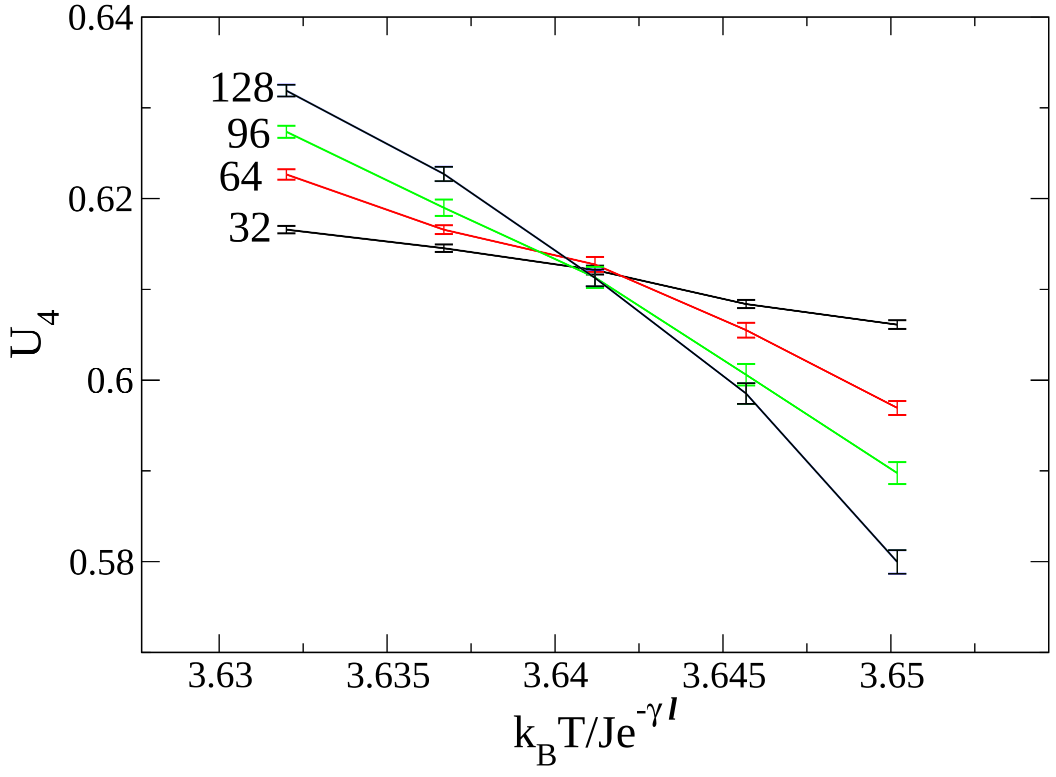

Figure 4.1: Binder cumulant for the rigid model. The magnetic field $H = 0$. Crossing point indicates that it belongs to the same universality class of the two-dimensional rigid Ising model.

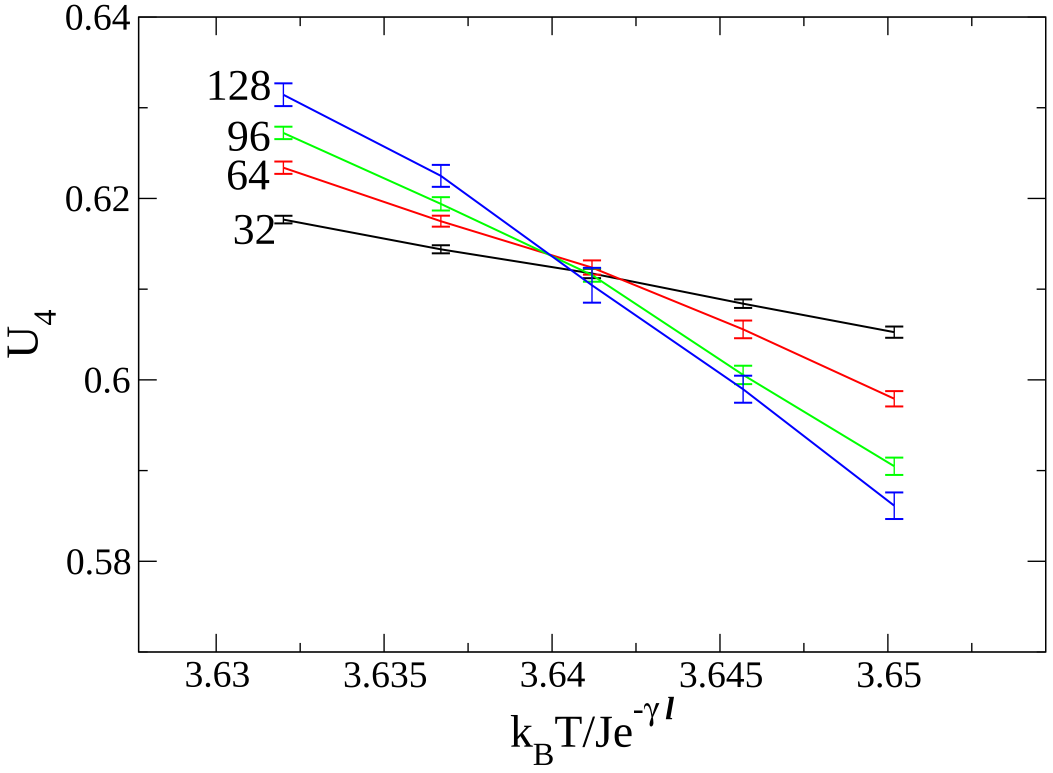

Figure 4.2: Binder cumulant for the model with $k = 120000$. The magnetic field $H = 0$. Crossing point indicates that it belongs to the same universality class of the two-dimensional rigid Ising model.

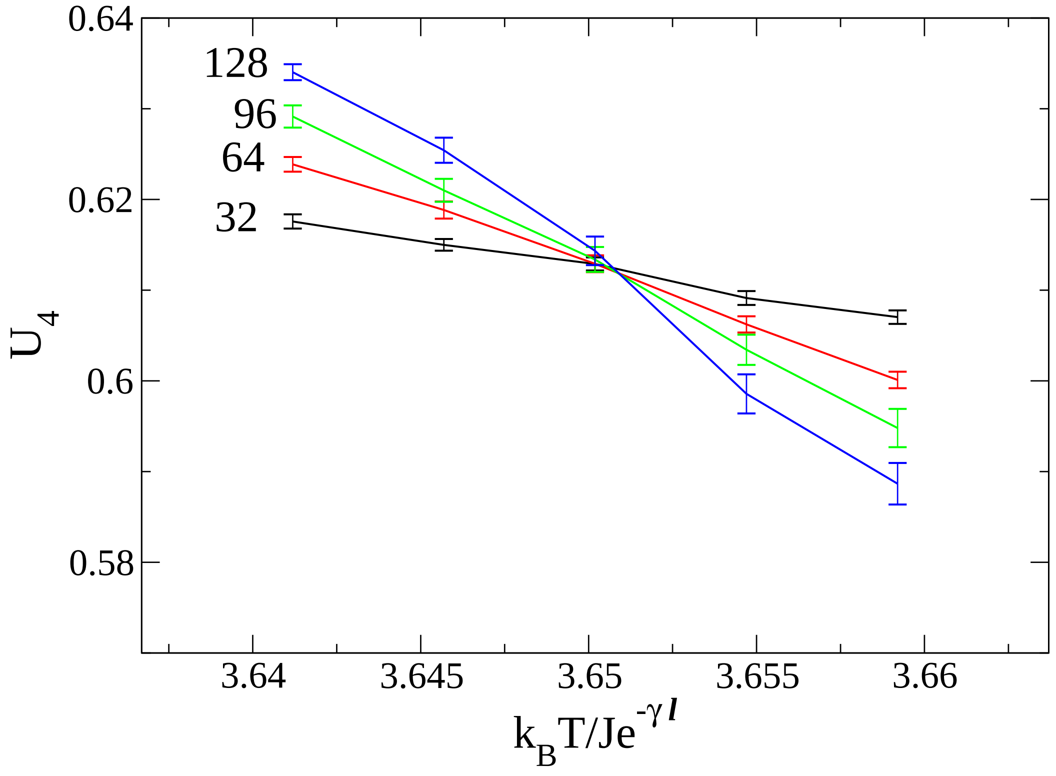

Figure 4.3: Binder cumulant for the model with $k = 20000$. The magnetic field $H = 0$. Crossing point indicates that it belongs to the same universality class of the two-dimensional rigid Ising model.

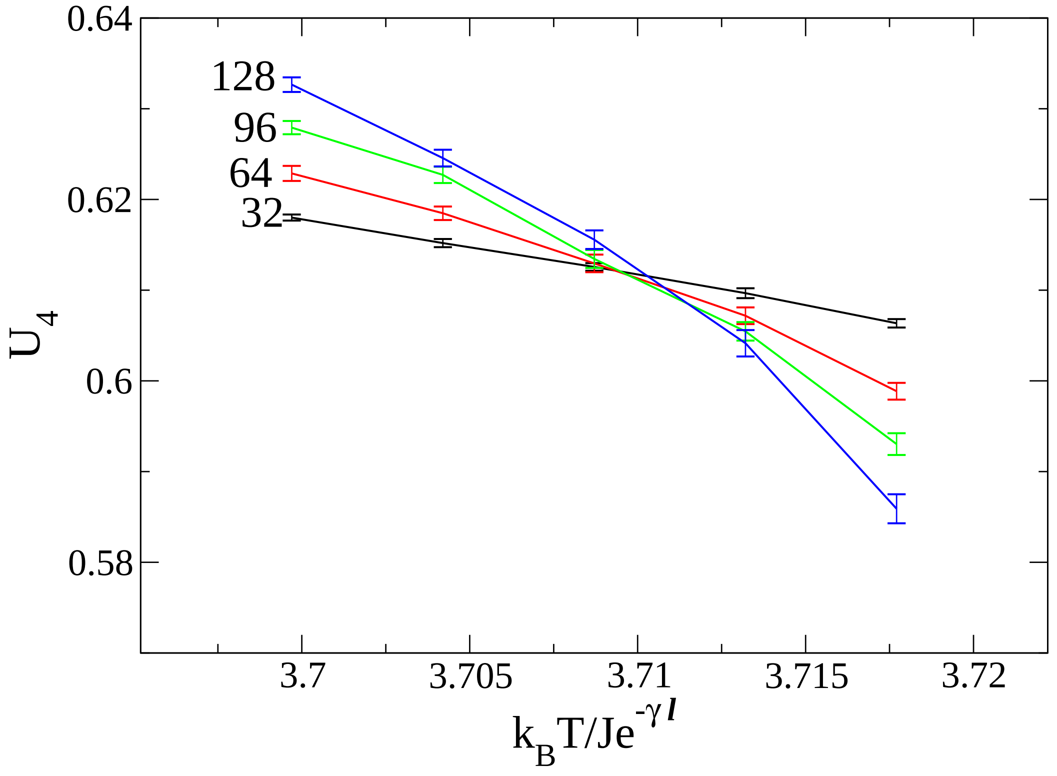

Figure 4.4: Binder cumulant for the model with $k = 3000$. The magnetic field $H = 0$. Crossing point indicates that it belongs to the same universality class of the two-dimensional rigid Ising model.

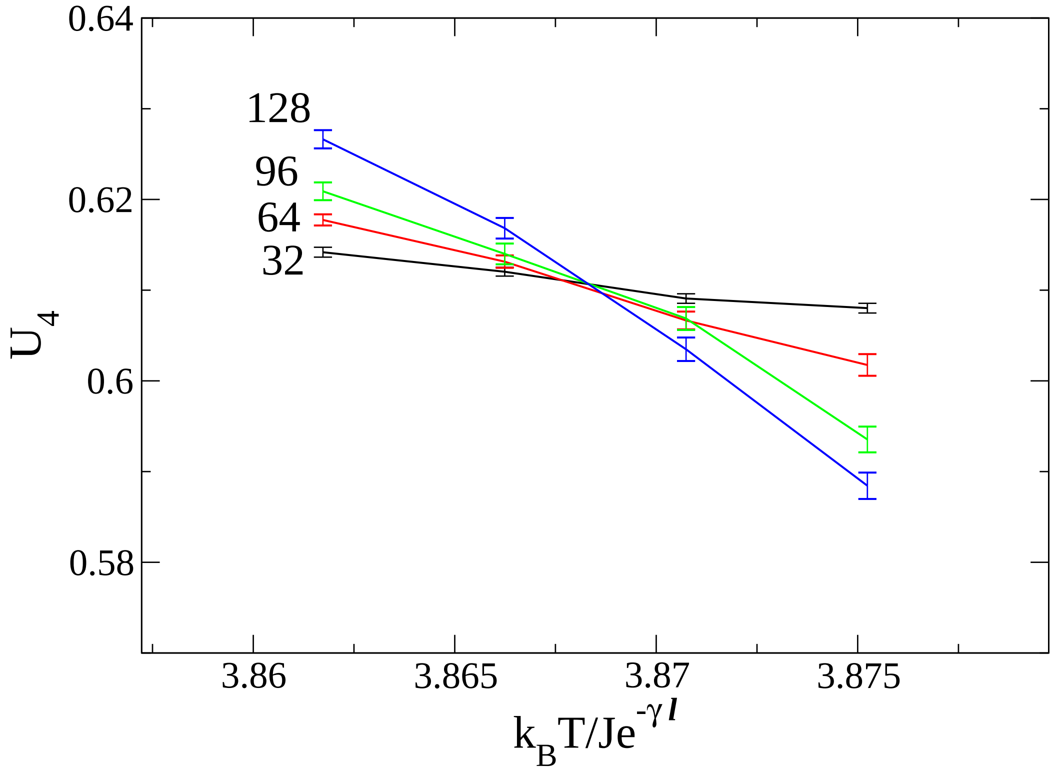

Figure 4.5: Binder cumulant for the model with $k = 1000$. The magnetic field $H = 0$. Crossing point indicates that it belongs to the same universality class of the two-dimensional rigid Ising model.

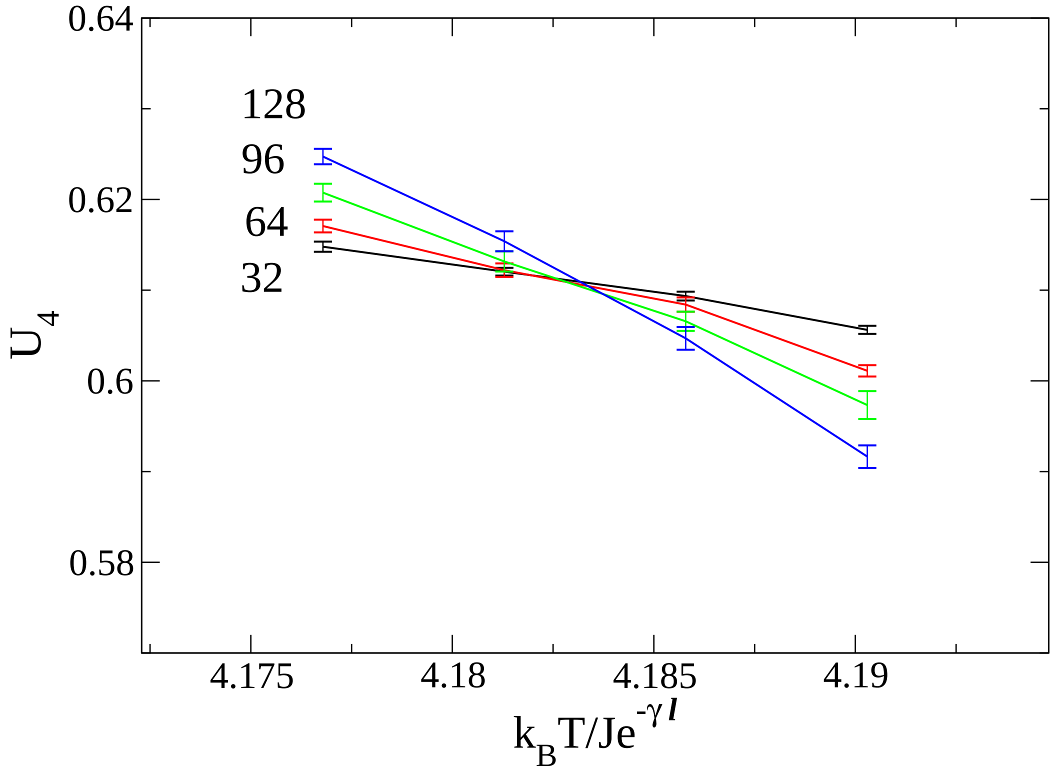

Figure 4.6: Binder cumulant for the model with $k = 500$. The magnetic field $H = 0$. Crossing point indicates that it belongs to the same universality class of the two-dimensional rigid Ising model.

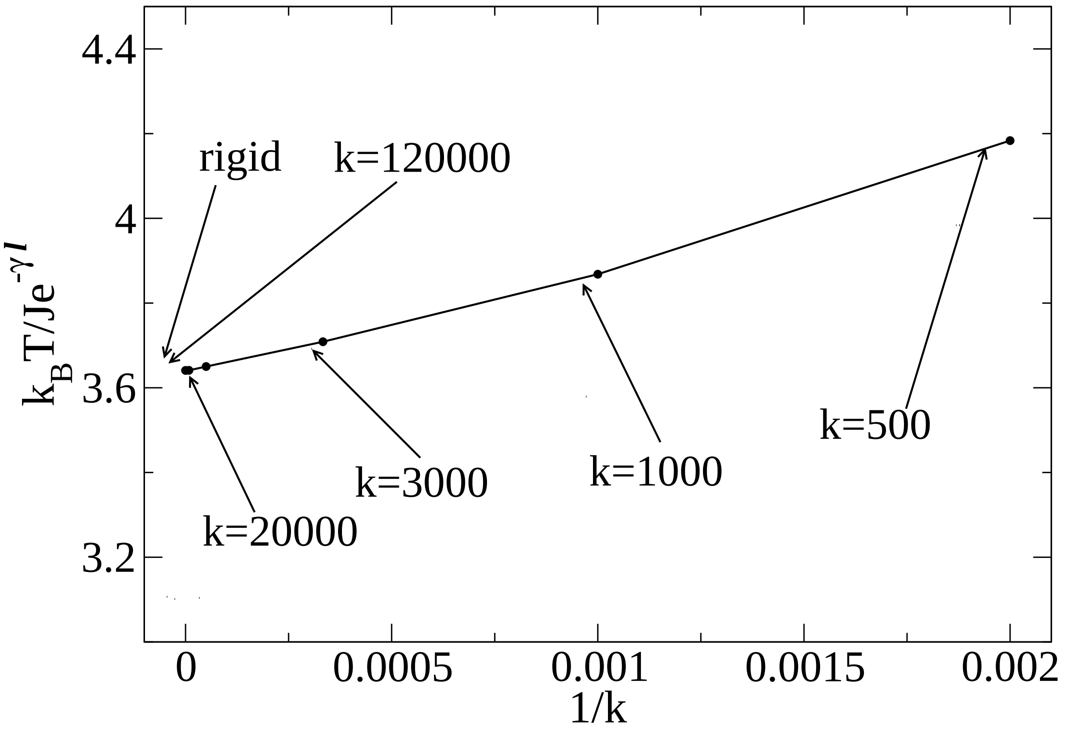

Figure 4.7: Critical temperatures vs. $1/k$.

## 4.2 Normalized Magnetization Distribution

Binder cumulant plots not only help us identify the system's universality class but also estimate the critical point $T_C$ from the locations of the crossing points. By using the estimate of $T_C$ and the zero magnetic field (the estimate of $H_C$), we can also examine the probability distribution of order parameter $m$ (magnetization per spin) under (($T'$, $H'$) by the histogram reweighting method as shown in Eq. 3.14, where ($T'$, $H'$) is very close to our estimate of ($T_C$, $H_C$). We adjust $(T', H')$ until the curves of normalized unit-variance probability distribution of the order parameter of different system sizes collapse. We do this to make sure that our estimate of ($T_C$, $H_C$) is accurate enough and also to understand the nature of the transition. In particular, we calculate the standard variance $\sigma$ of the magnetization per spin $m$ and then plot $P(m)\sigma$ vs. $m/\sigma$, where $P(m)$ is the probability distribution of the magnetization. This is because for a large finite system of liner dimension $L$ at the critical point, $P(m)$ takes the form

$$P(m) = bP^*(\tilde{m}) \tag{4.2}$$

where $b = b_0 L^{\beta/\nu}$, $\beta$ and $\nu$ are critical exponents, $\tilde{m} = bm$, $b_0$ is a constant, and $P^*(\tilde{m})$ is a universal scaling function, normalized to unit norm and unit variance. Scaling functions $P^*(\tilde{m})$ is characteristic of the corresponding universality class. So we can identify universality class by using $P^*(\tilde{m})$. Here $b = \frac{1}{\sigma}$, and then $P^*(\tilde{m}) = P(m)\sigma$. This is why we plot $P(m)\sigma$ vs. $m/\sigma$.

Fig. 4.8-4.13 show the collapse of the scaling functions of different system sizes for each system, and the scaling functions agree with that of the rigid 2D Ising model. This indicates that the phase transition belongs to the rigid 2D Ising universality class.

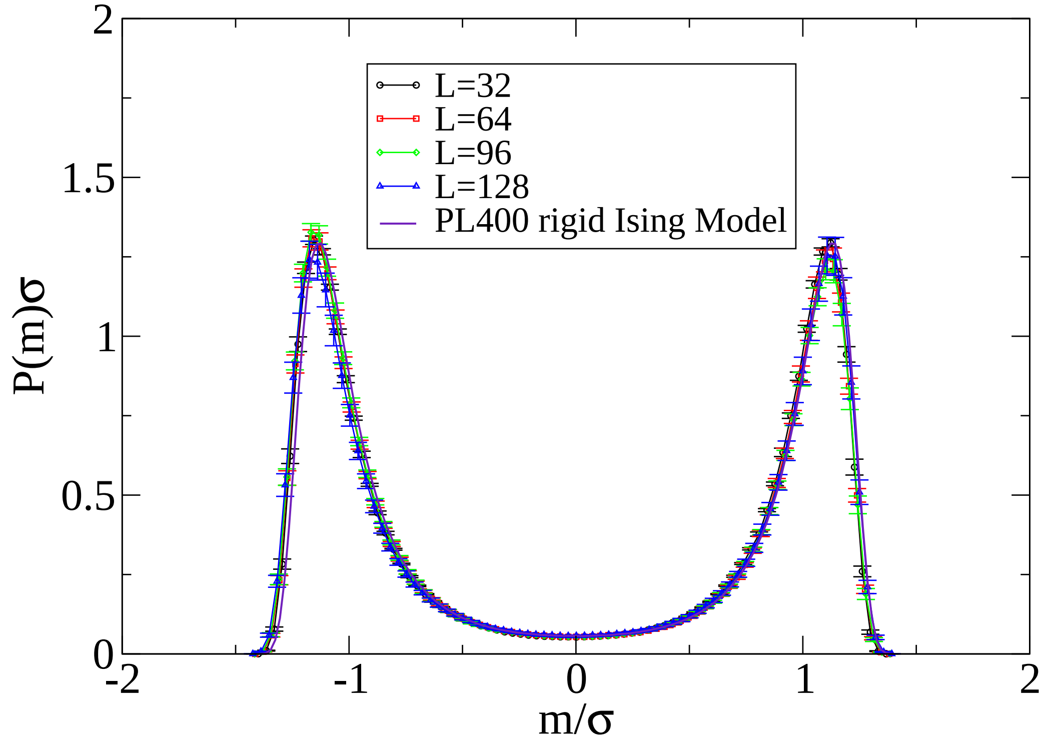

Figure 4.8: The scaled probability distribution of magnetization for the rigid model.

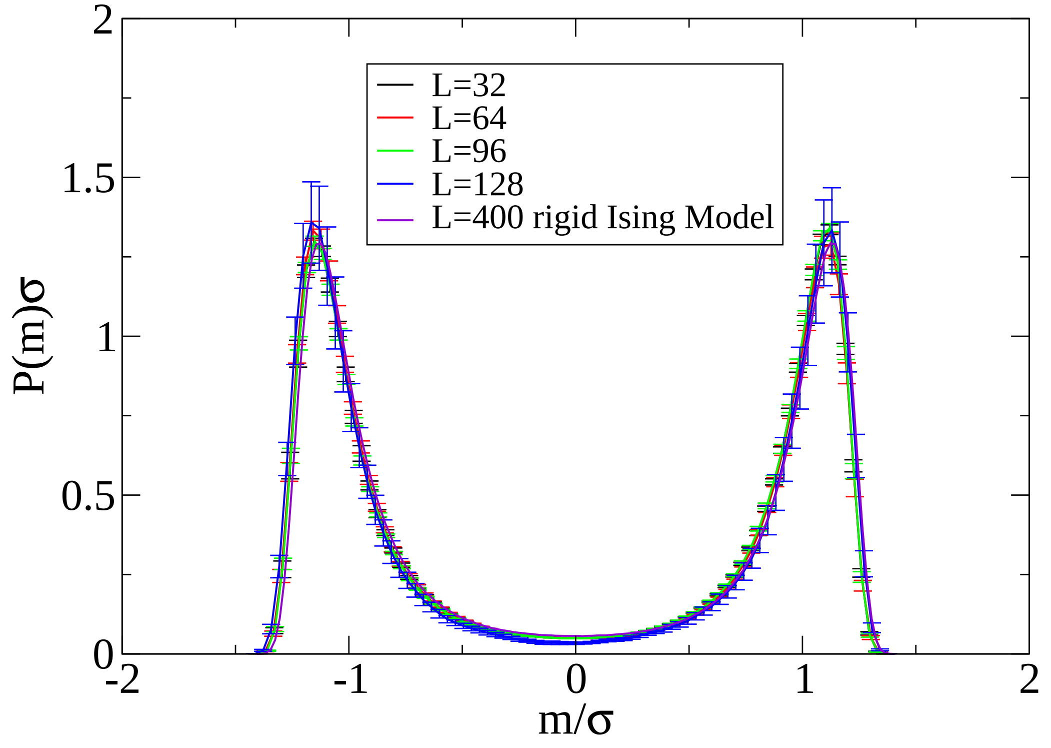

Figure 4.9: The scaled probability distribution of magnetization for the model with $k = 120000$.

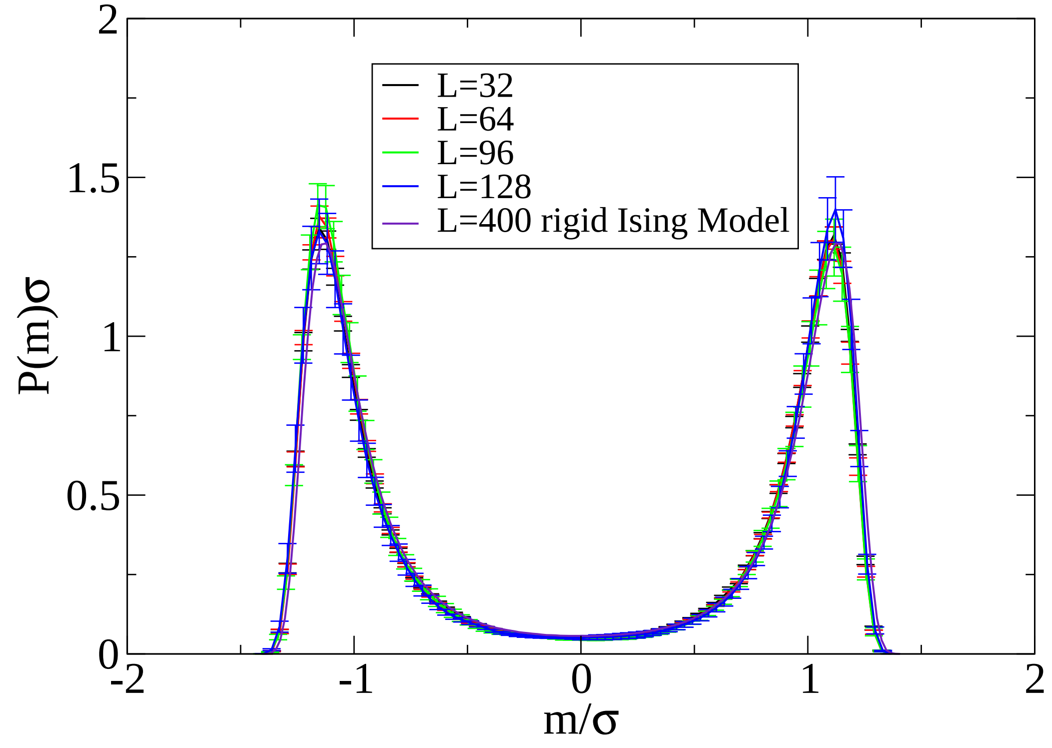

Figure 4.10: The scaled probability distribution of magnetization for the model with $k = 20000$.

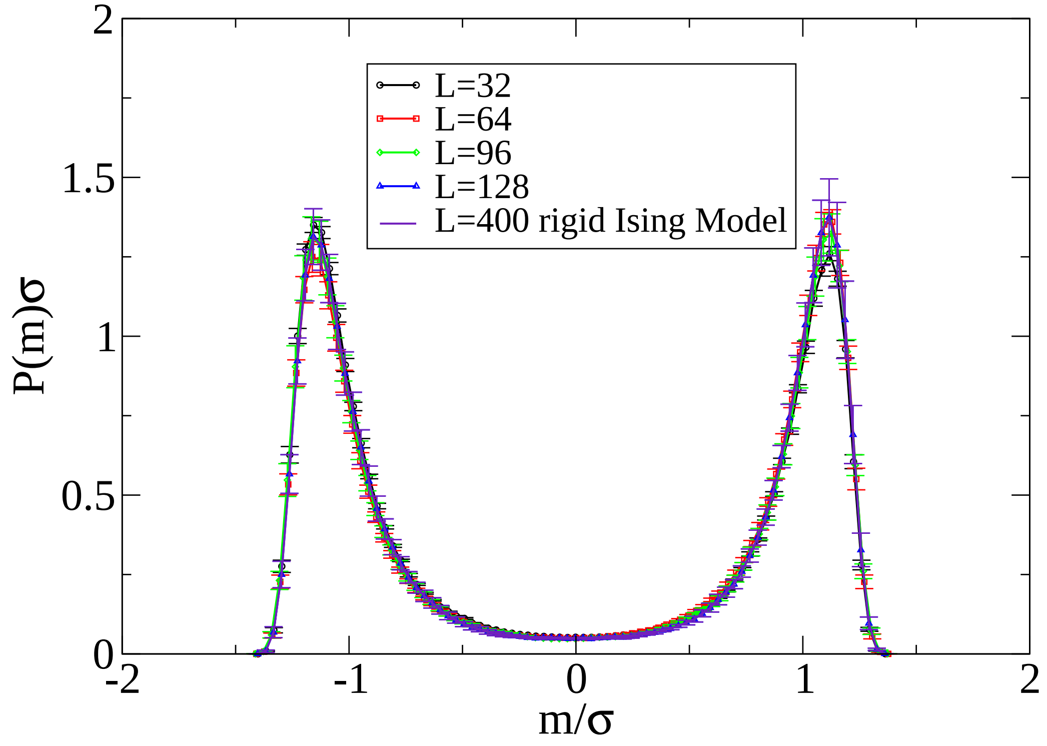

Figure 4.11: The scaled probability distribution of magnetization for the model with $k =$ 3000.

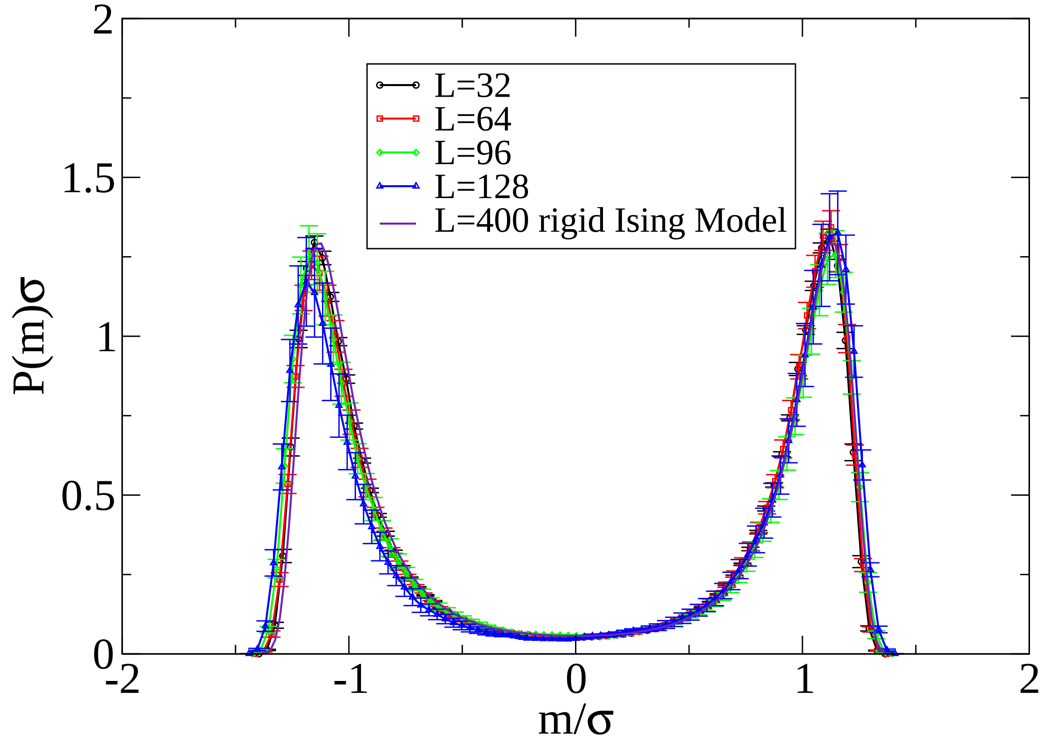

Figure 4.12: The scaled probability distribution of magnetization for the model with $k = 1000$.

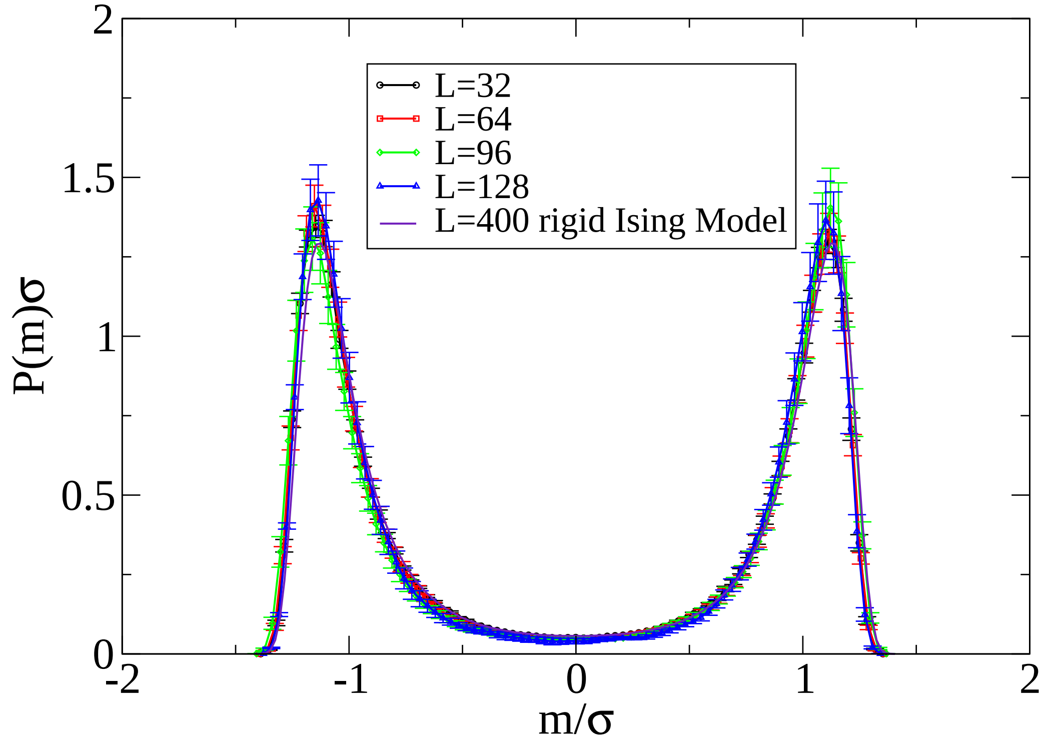

Figure 4.13: The scaled probability distribution of magnetization for the model with $k = 500$.

## 4.3 Thermodynamic derivatives

We extract $\nu$ by considering the scaling behavior of certain thermodynamic derivatives, including the derivative of the logarithmic derivatives of $\langle m^2 \rangle$ and $\langle |m| \rangle$, as in [42]. We plot these properties as a function of lattice size on a log-log scale in Fig. 4.14-4.19.

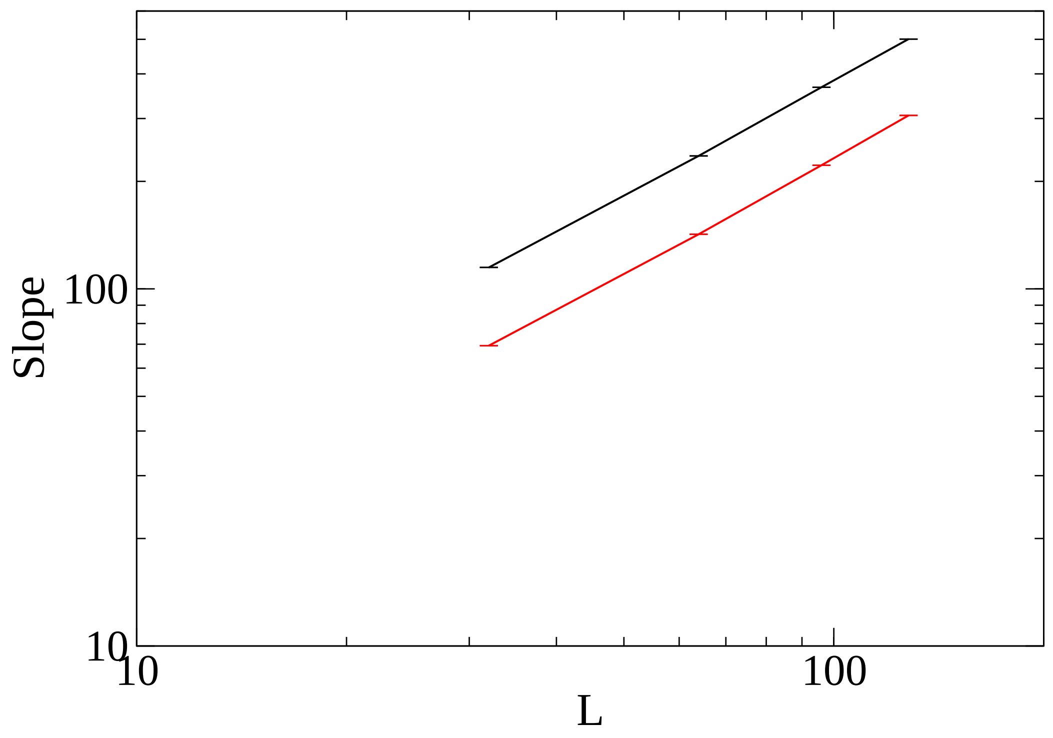

Figure 4.14: For the rigid model: log-log plot of the maximum slopes of various thermodynamic quantities used to determine $\nu$. The black one is for the derivative of $\ln m^2$, and the red one is for the derivative of $\ln |m|$.

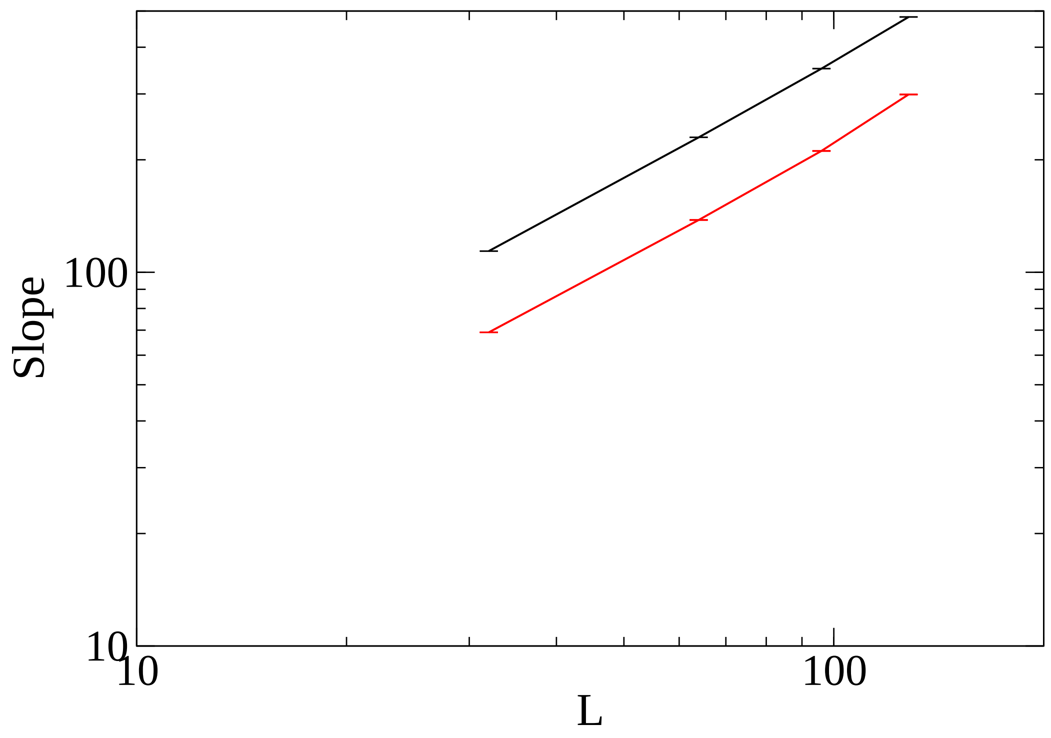

Figure 4.15: For the model with $k = 120000$: log-log plot of the maximum slopes of various thermodynamic quantities used to determine $\nu$. The black one is for the derivative of $\ln m^2$, and the red one is for the derivative of $\ln |m|$.

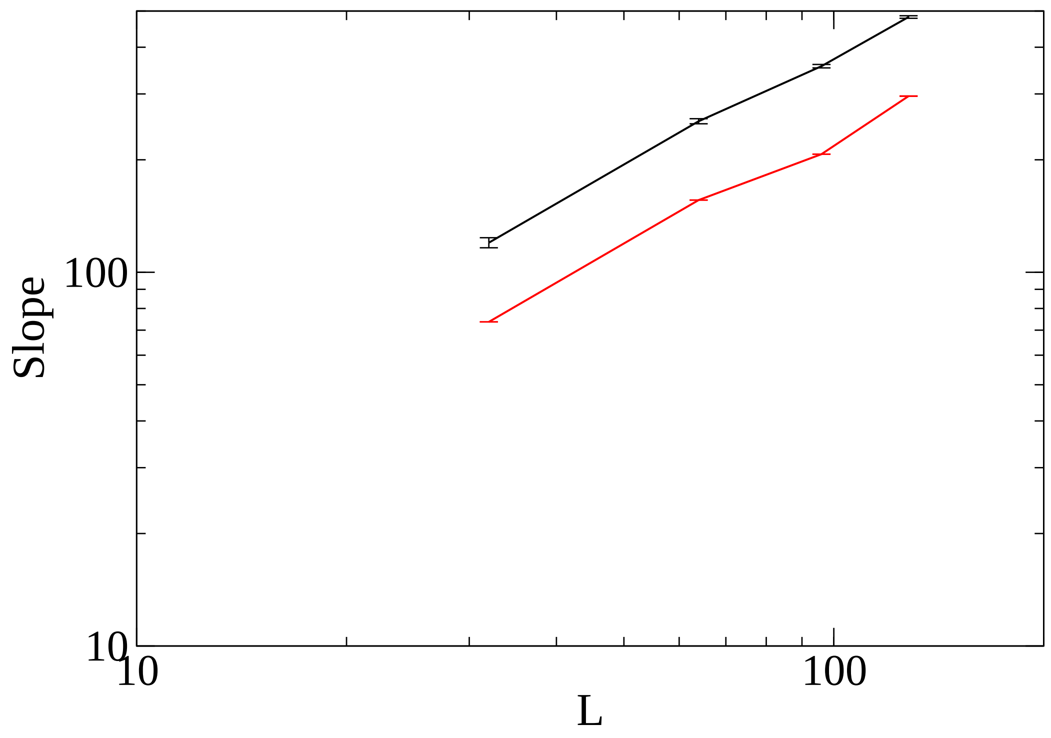

Figure 4.16: For the model with $k = 20000$: log-log plot of the maximum slopes of various thermodynamic quantities used to determine $\nu$. The black one is for the derivative of $\ln m^2$, and the red one is for the derivative of $\ln |m|$.

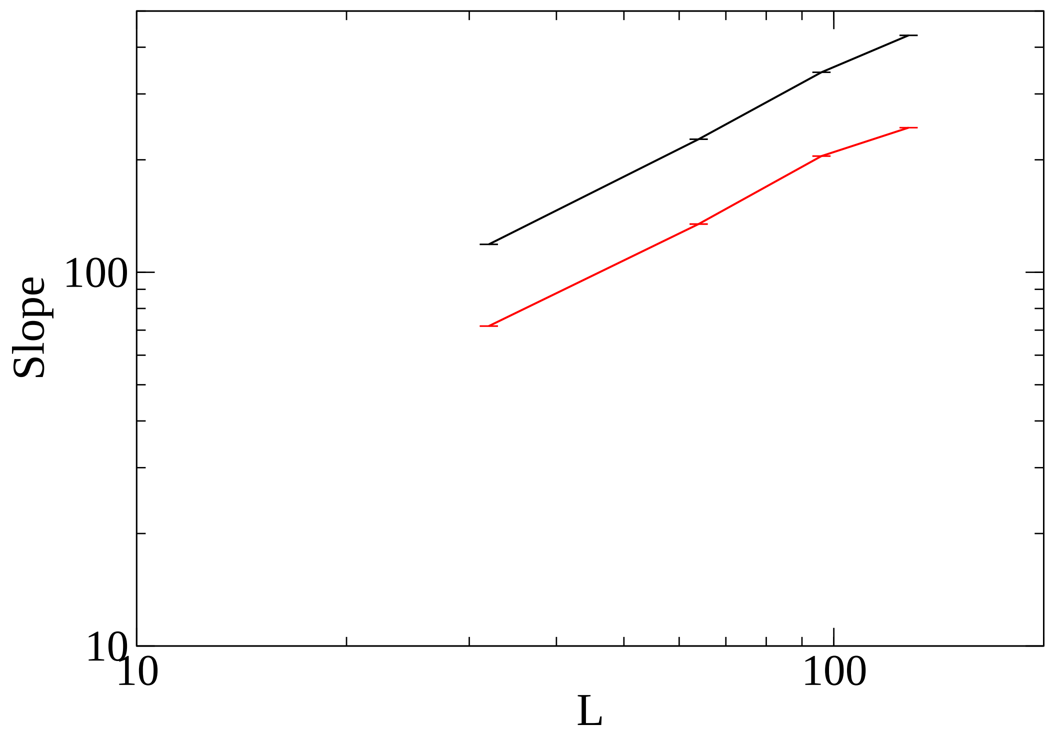

Figure 4.17: For the model with $k = 3000$: log-log plot of the maximum slopes of various thermodynamic quantities used to determine $\nu$. The black one is for the derivative of $\ln m^2$, and the red one is for the derivative of $\ln |m|$.

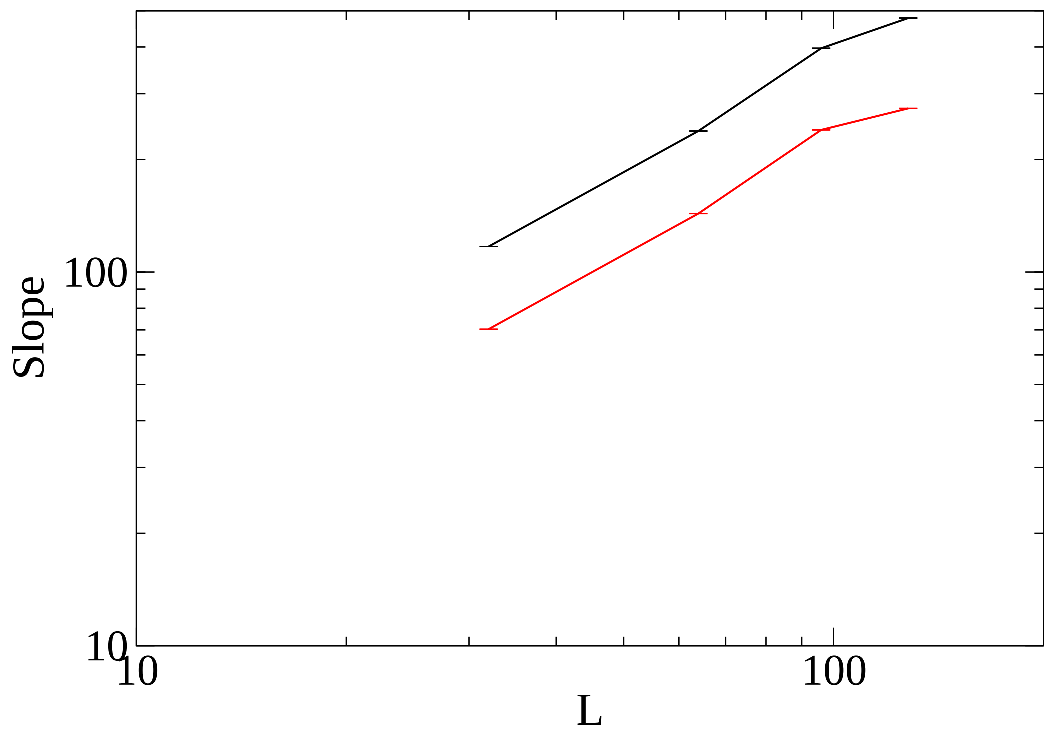

Figure 4.18: For the model with $k = 1000$: log-log plot of the maximum slopes of various thermodynamic quantities used to determine $\nu$. The black one is for the derivative of $\ln m^2$, and the red one is for the derivative of $\ln |m|$.

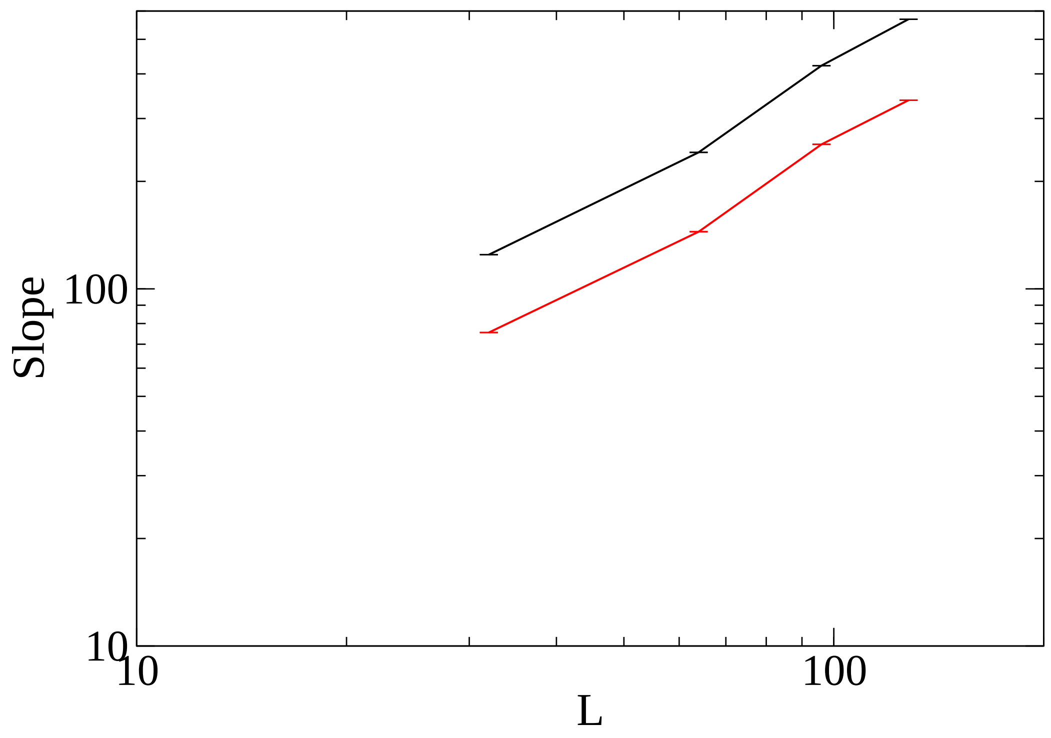

Figure 4.19: For the model with $k = 500$: log-log plot of the maximum slopes of various thermodynamic quantities used to determine $\nu$. The black one is for the derivative of $\ln m^2$, and the red one is for the derivative of $\ln |m|$.

The estimates for $1/\nu$ from nonlinear least square fits to Eq. 2.6 are given in the Table 4.1-4.5. Combing these two results we get the estimates of $\nu$ as shown in Table 4.6. For the 2D rigid Ising model, $\nu = 1$. Our results are close to it but not good enough. This is because the critical temperature $T_C$ we estimate from the Binder cumulant crossing is not accurate enough. To have an accurate estimate of $T_C$, we still need to do a finite size scaling analysis of $U_4$. In our study of the static behavior, our goal is only to have an estimate of $T_C$ to be used in the simulation of the dynamic behavior (phase separation) of these systems. So we go with these estimates of $T_c$'s.

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

| | $1/\nu$ |
|---|---|
| $\log\langle m^2\rangle$ | $1.062 \pm 0.008$ |
| $\log\langle \|m\|\rangle$ | $1.060 \pm 0.009$ |

Table 4.2: For the model with $k = 120000$: estimates for $1/\nu$ obtained by finite size scaling of the maximum slopes of the cumulant and the logarithmic derivatives of $m^2$ and $|m|$.

| | $1/\nu$ |
|---|---|
| $\log\langle m^2\rangle$ | $1.036 \pm 0.013$ |
| $\log\langle \|m\|\rangle$ | $1.046 \pm 0.027$ |

Table 4.3: For the model with $k = 20000$: estimates for $1/\nu$ obtained by finite size scaling of the maximum slopes of the cumulant and the logarithmic derivatives of $m^2$ and $|m|$.

| | $1/\nu$ |
|---|---|
| $\log\langle m^2\rangle$ | $1.016 \pm 0.031$ |
| $\log\langle \|m\|\rangle$ | $0.984 \pm 0.050$ |

Table 4.4: For the model with $k = 3000$: estimates for $1/\nu$ obtained by finite size scaling of the maximum slopes of the cumulant and the logarithmic derivatives of $m^2$ and $|m|$.

| | $1/\nu$ |
|---|---|
| $\log\langle m^2\rangle$ | $0.935 \pm 0.018$ |
| $\log\langle \|m\|\rangle$ | $0.892 \pm 0.035$ |

Table 4.5: For the model with $k = 1000$: estimates for $1/\nu$ obtained by finite size scaling of the maximum slopes of the cumulant and the logarithmic derivatives of $m^2$ and $|m|$.

| | $1/\nu$ |
|---|---|
| $\log\langle m^2\rangle$ | $1.043 \pm 0.066$ |
| $\log\langle \|m\|\rangle$ | $1.055 \pm 0.047$ |

Table 4.6: For the model with $k = 500$: estimates for $1/\nu$ obtained by finite size scaling of the maximum slopes of the cumulant and the logarithmic derivatives of $m^2$ and $|m|$.

| | $1/\nu$ |
|---|---|
| $\log\langle m^2\rangle$ | $1.103 \pm 0.042$ |
| $\log\langle \|m\|\rangle$ | $1.091 \pm 0.042$ |

Table 4.7: Estimates for $\nu$ obtained by combining the results from Table 4.1-4.5.

| k | $\nu$ |
|---|---|
| rigid | $0.943 \pm 0.010$ |
| 120000 | $0.961 \pm 0.020$ |
| 20000 | $1.000 \pm 0.040$ |
| 3000 | $1.090 \pm 0.0031$ |
| 1000 | $0.953 \pm 0.032$ |
| 1000 | $0.953 \pm 0.032$ |
| 500 | $0.911 \pm 0.051$ |

# Chapter 5

# Results: Domain Growth

## 5.1 Correlation Function

We show in Fig. 5.1 - Fig. 5.3 configurations at different times for models with different elasticities (rigid model, $k = 20000$ and $k = 500$). Down-spins(-1) are blue, and up-spins(+1) are yellow.

All simulations were performed in the spinodal decomposition regime, as the above figures indicate, and all three models clearly show phase separation behavior. Few qualitative differences in the overall structure can be seen between the rigid and the compressible models. And for each system, the late-time domain patterns look statistically similar to the ones formed at early times, which indicates that phase separation is a dynamical scaling phenomenon.

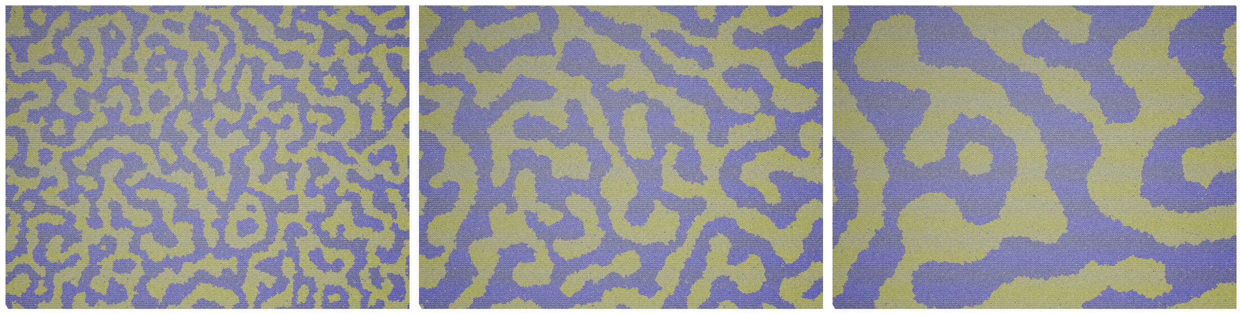

Figure 5.1: The sequence of $512 \times 512$ system's configurations at $t = 10000, t = 100000, t = 1000000 MCS$ after quench down to $0.59T_C$. For the rigid model with the periodic boundary condition.

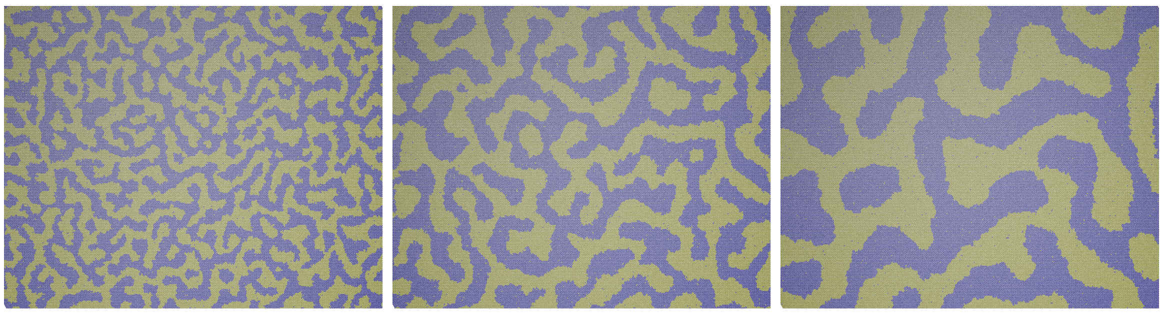

Figure 5.2: The sequence of $512 \times 512$ system's configurations at $t = 10000, t = 100000, t = 1000000MCS$ after quench down to $0.59T_C$. For the model with $k = 20000$ with the periodic boundary condition.

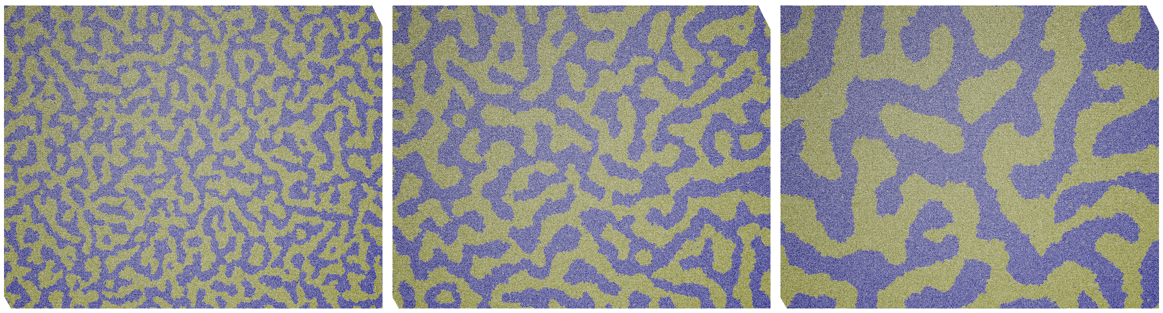

Figure 5.3: The sequence of $512 \times 512$ system's configurations at $t = 10000, t = 100000, t = 1000000MCS$ after quench down to $0.59T_C$. For the model with $k = 500$ with the periodic boundary condition.

In Fig. 5.4-5.7 we show the correlation function for the rigid model and the one with $k = 500$. The correlation functions for both the rigid model and the elastic models are isotropic, and in these graphs each sample of $C(r)$ is averaged over the three equivalent directions of the triangular net and the curves are obtained from averaging over 100 samples.

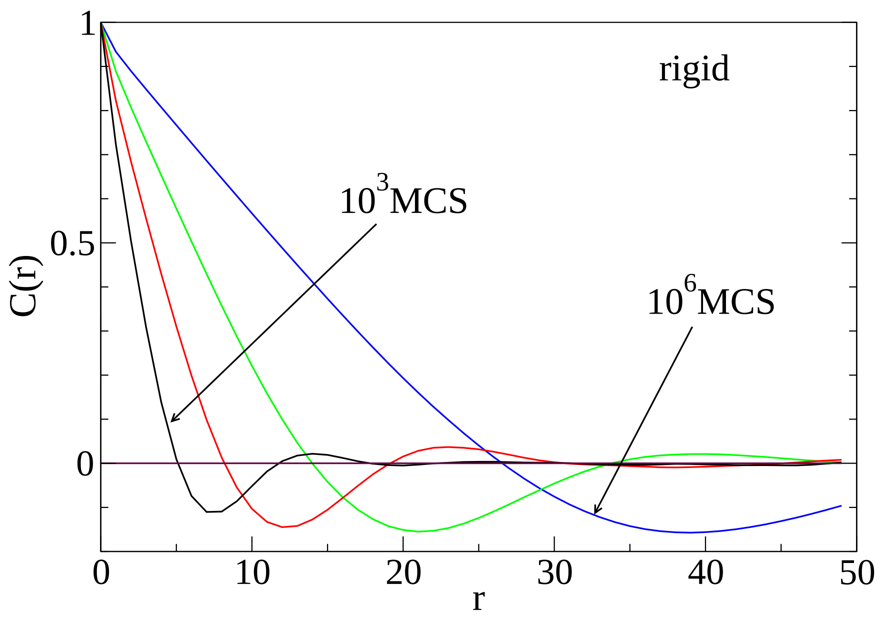

Figure 5.4: $C(r)$ vs. $r$ for the rigid model. The curves are for $t = 10^3, t = 10^4, t = 10^5, t = 10^6$ MCS. $L = 512$, averaged over the three triangle directions for 100 samples. Error bars are less than the line thickness.

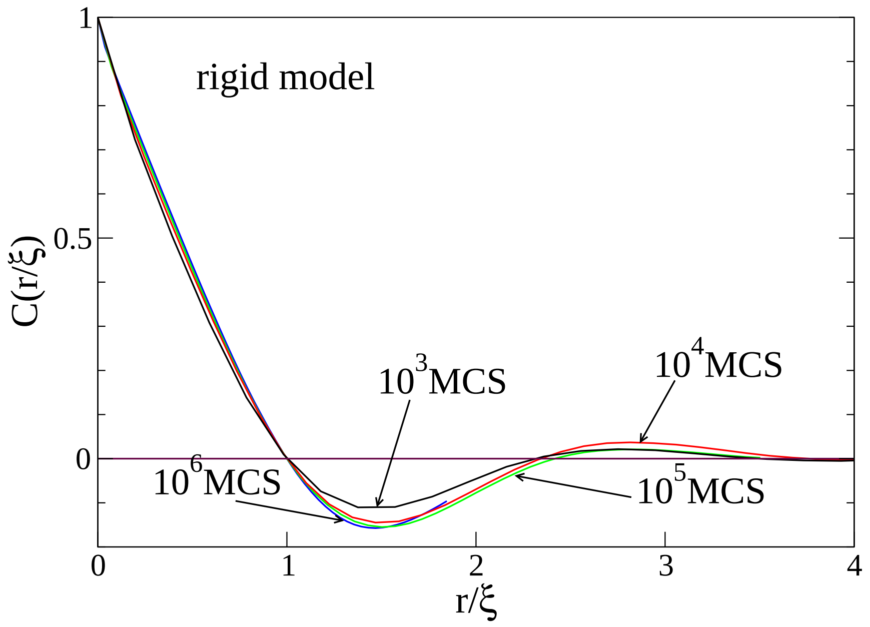

Figure 5.5: The scaled form of $C(r)$ for the rigid model. Curves of different $t$ collapse for large enough $t$.

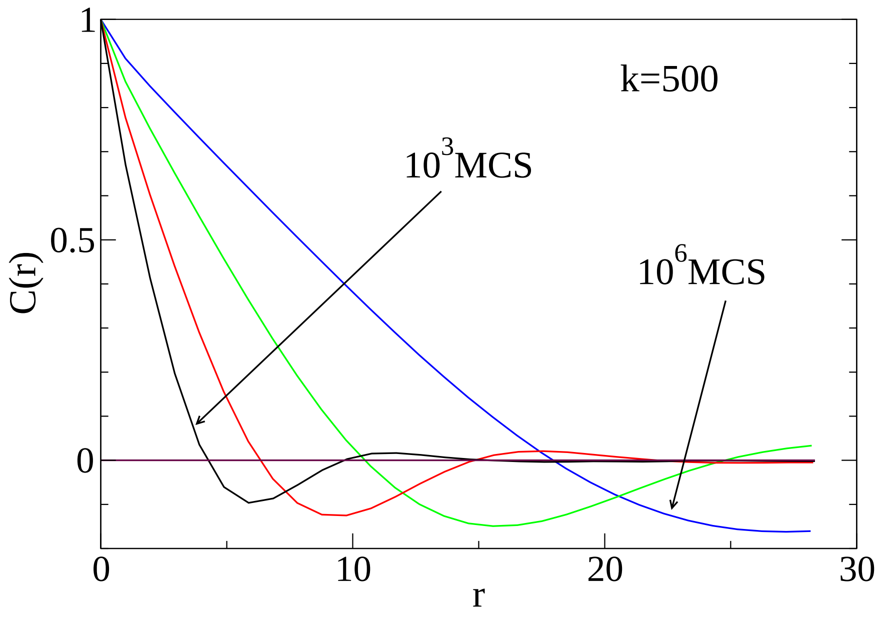

Figure 5.6: $C(r)$ vs. $r$ for the model with $k = 500$. The curves are for $t = 10^3, t = 10^4, t = 10^5, t = 10^6$ MCS. $L = 512$, averaged over the three triangle directions for 100 samples. Error bars are less than the line thickness.

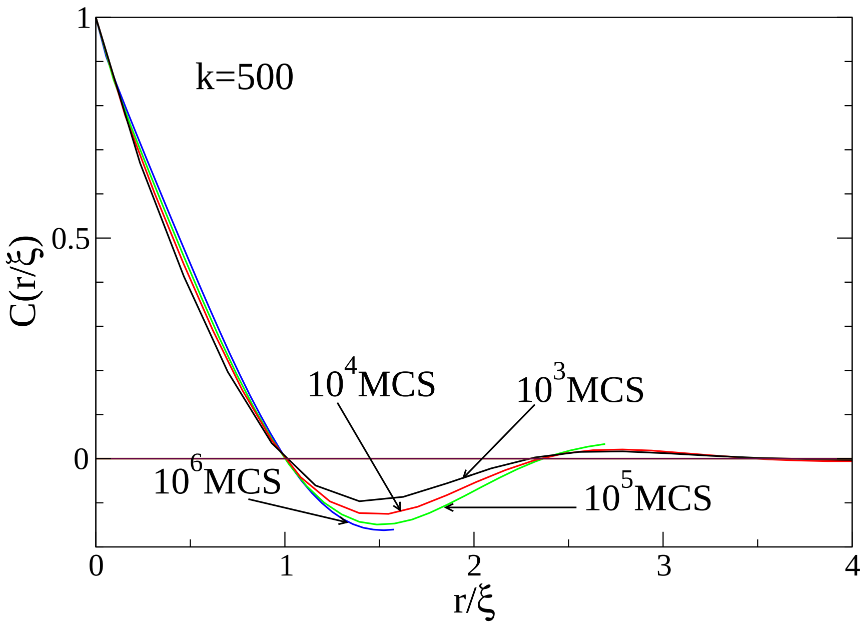

Figure 5.7: The scaled form of $C(r)$ for the model with $k = 500$. Curves of different $t$ have a tendency to collapse with $t$ increasing.

Both the rigid and elastic models have well-defined first-zero crossings at every time points. And from the scaled form of the correlation function we observe the collapse of curves of different time. It clearly indicates that phase separation is a dynamic scaling phenomenon. In the Table 5.1-5.2 we show part of the data plotted in Fig. 5.1-5.2.

$\xi$ is defined as the first zero crossing of the correlation function curves, as illustrated in Sec. 3.8. So from each sample of $C(r)$ vs. $r$, we calculate $\xi$ as a function of $t$. Then we can have an average of them as the expected $\xi$. For example, in the Table 5.3 we show one sample of $C(r)$ vs. $r$.

We expect the first-zero crossing $\xi$ in between of 4.99 and 5.99 for $t = 10^3$ MCS. Let's just take it as an example to show how to get $\xi$ at $t = 10^3$ MCS. First of all we pick 3 points whose $C(r)$ are closest to 0, which are $(4.99993, 0.00638202)$, $(5.99991, -0.0786152)$, $(6.9999, -0.11656)$ from Table 5.3. These numbers are represented as $(r, C(r))$. And then we do a second order polynomial fit of these three points and end up with the equation

Table 5.1: $C(r)$ vs. $r$ for the rigid model averaged over 100 samples at $t = 10^3$ MCS.

| $r$ | $C(r)$ | error |
|---:|---:|---:|
| 0 | 1 | 0 |
| 0.99999 | 0.7242 | 1.01476E-4 |
| 1.99997 | 0.50565 | 1.67422E-4 |
| 2.99995 | 0.30892 | 2.22388E-4 |
| 3.99994 | 0.13873 | 2.50851E-4 |
| 4.99993 | 0.00706 | 2.51303E-4 |
| 5.99991 | -0.07779 | 2.22846E-4 |
| 6.9999 | -0.11617 | 1.99524E-4 |
| 7.99988 | -0.11646 | 2.144E-4 |
| 8.99987 | -0.09195 | 2.50396E-4 |

which fits these three points : $y = 0.02353x^2 - 0.34379x + 1.13717$, where $x$ takes the $r$ value and $y$ takes the corresponding $C(r)$ value. Finally, we solve this equation and get two solutions: $x_1 = 5.06043$, $x_2 = 9.55316$. Obviously we only need to keep the lower one, so $\xi = 5.06043$ at $t = 10^3$ MCS. In the Table 5.4 we show $\xi(t)$ vs. $t$ for the rigid model for this specific sample. We observe that $\xi$ increases with $t$.

## 5.2 Least Square fitting

According to Eq. 1.2, we have:

$$\xi'(t) = Bt^n \tag{5.1}$$

where $\xi'(t) = \xi(t) - A$. Here we use $\xi(t)$ in place of $R(t)$ because the domain size $R(t)$ at time $t$ is represented as the first zero crossing $\xi(t)$ in our simulation. So we have the following equation:

$$\ln \xi'(t) = n \ln t + \ln B \tag{5.2}$$

If we can get a good estimation of $A$ first, then we can do a linear fitting of $\ln \xi'$ vs. $\ln t$ as shown in Eq. 5.2. So $n$ naturally appears as the slope of this linear fitting. We can actually

Table 5.2: $C(r)$ vs. $r$ for the rigid model averaged over 100 samples at $t = 10^4$ MCS

| $r$ | $C(r)$ | error |
|---:|---:|---:|
| 0 | 1 | 0 |
| 0.99999 | 0.82143 | 1.02506E-4 |
| 1.99997 | 0.68238 | 1.77396E-4 |
| 2.99995 | 0.55144 | 2.45757E-4 |
| 3.99994 | 0.42519 | 3.08406E-4 |
| 4.99993 | 0.30463 | 3.62116E-4 |
| 5.99991 | 0.19242 | 3.97314E-4 |
| 6.9999 | 0.09209 | 4.15161E-4 |
| 7.99988 | 0.00726 | 4.09974E-4 |
| 8.99987 | -0.05939 | 3.77499E-4 |

get a good estimation of $A$ and $n$ simultaneously by varying $A$ and comparing $\Re^2$ values for the above fitting. $\Re^2$, the coefficient of determination, is as an index reflecting how much volatility is explained by the linear model [16]. As a result, the one with the best $\Re^2$ score gives both good estimations of $A$ and $n$. We plot $\ln \xi'$ vs. $\ln t$ in Fig 5.5-5.10 for different systems and include $\Re^2$ vs. $A$ as the inset graphs. We use the $A$ value which gives the best $\Re^2$ score as the one used in the fitting, which we call it as $A_0$.

Table 5.3: $C(r)$ vs. $r$ for the rigid model for one sample at $t = 10^3$ MCS

| $r$ | $C(r)$ |
|---:|---:|
| 0 | 1 |
| 0.999985 | 0.723536 |
| 1.99997 | 0.505073 |
| 2.99995 | 0.308135 |
| 3.99994 | 0.137947 |
| 4.99993 | 0.00638202 |
| 5.99991 | -0.0786152 |
| 6.9999 | -0.11656 |
| 7.99988 | -0.116454 |
| 8.99987 | -0.0927422 |

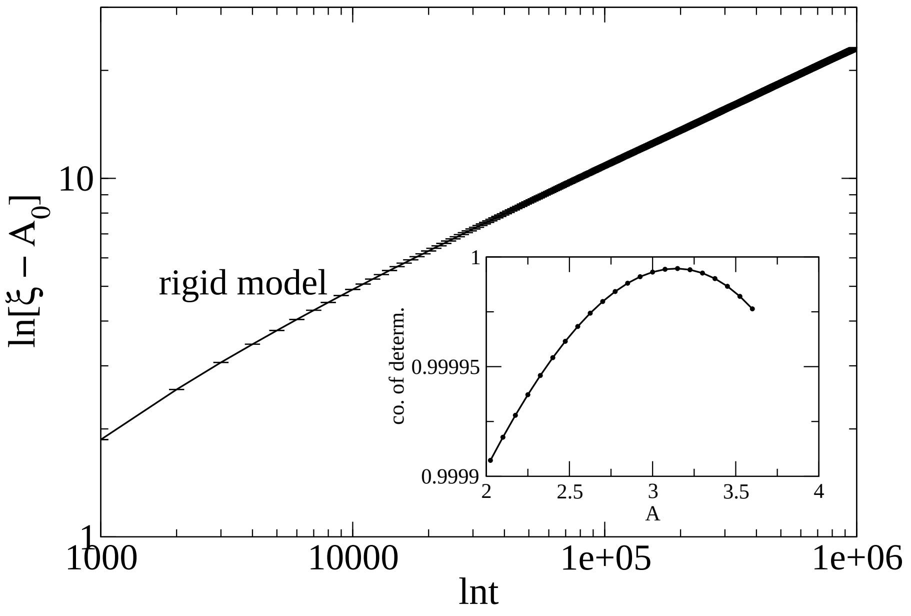

Figure 5.8: For the rigid model. $\ln \xi'(t)$ vs. $\ln t$. The inset shows $\Re^2$ for different $A$ values. The start fitting point is $t = 50000$ MCS. Different start fitting points ranged from 30000 to 80000 MCS are tested and $A_0$ differs only by 4% at most.

Table 5.4: $\xi(t)$ vs. $t$ for the rigid model for one sample

| $t$ | $\xi(t)$ |
|---|---|
| 1000 | 5.06043 |
| 2000 | 5.76313 |
| 3000 | 6.22384 |
| 4000 | 6.57068 |
| 5000 | 6.92243 |
| 6000 | 7.18181 |
| 7000 | 7.43524 |
| 8000 | 7.69064 |
| 9000 | 7.87531 |
| 10000 | 8.06324 |

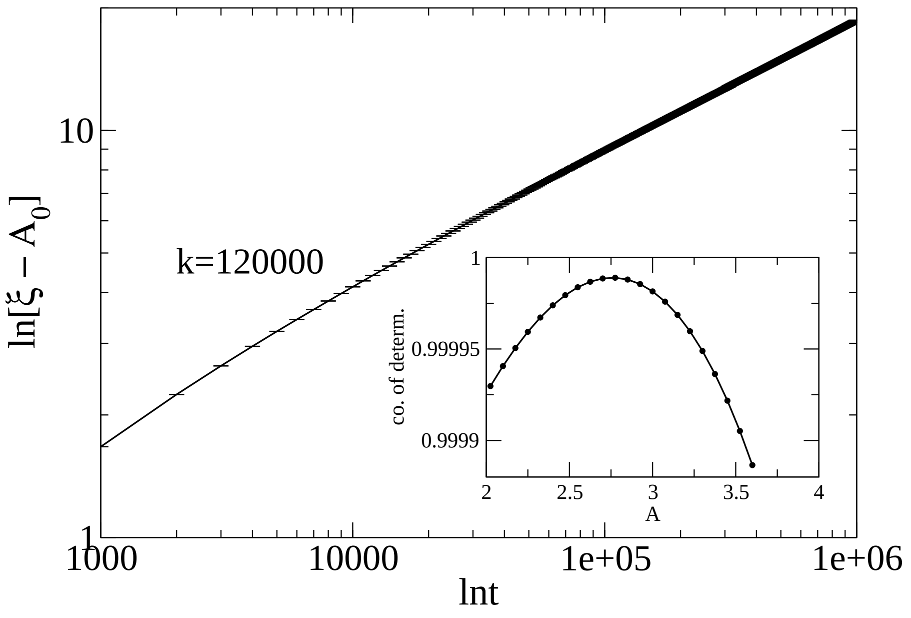

Figure 5.9: For the model with $k = 120000$. $\ln \xi'(t)$ vs. $\ln t$. The inset shows $\Re^2$ for different $A$ values. The start fitting point is $t = 50000$ MCS. Different start fitting points ranged from 30000 to 80000 MCS are tested and $A_0$ differs only by 4% at most.

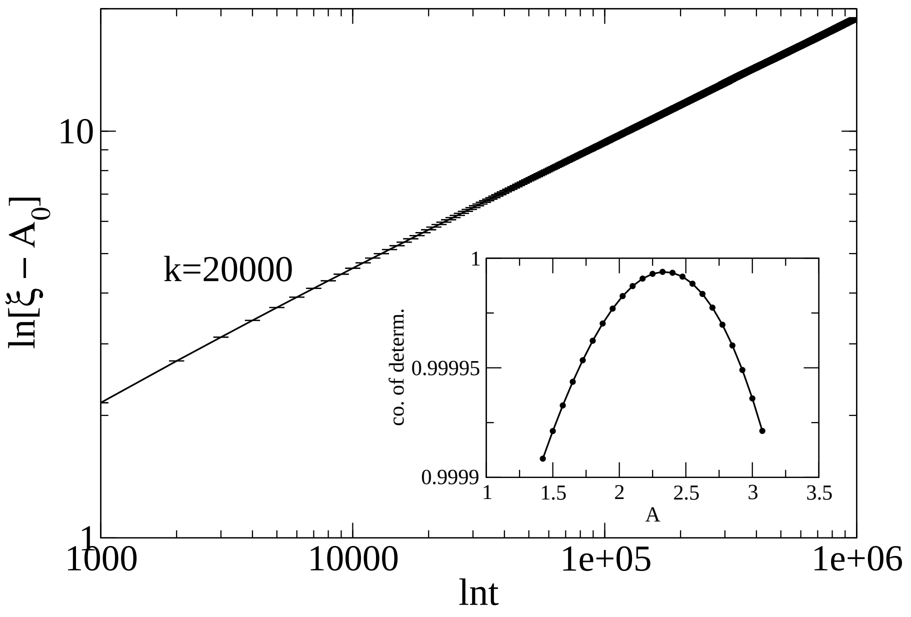

Figure 5.10: For the model with $k = 20000$. $\ln \xi'(t)$ vs. $\ln t$. The inset shows $\Re^2$ for different $A$ values. The start fitting point is $t = 50000$ MCS. Different start fitting points ranged from 30000 to 80000 MCS are tested and $A_0$ differs only by 4% at most.

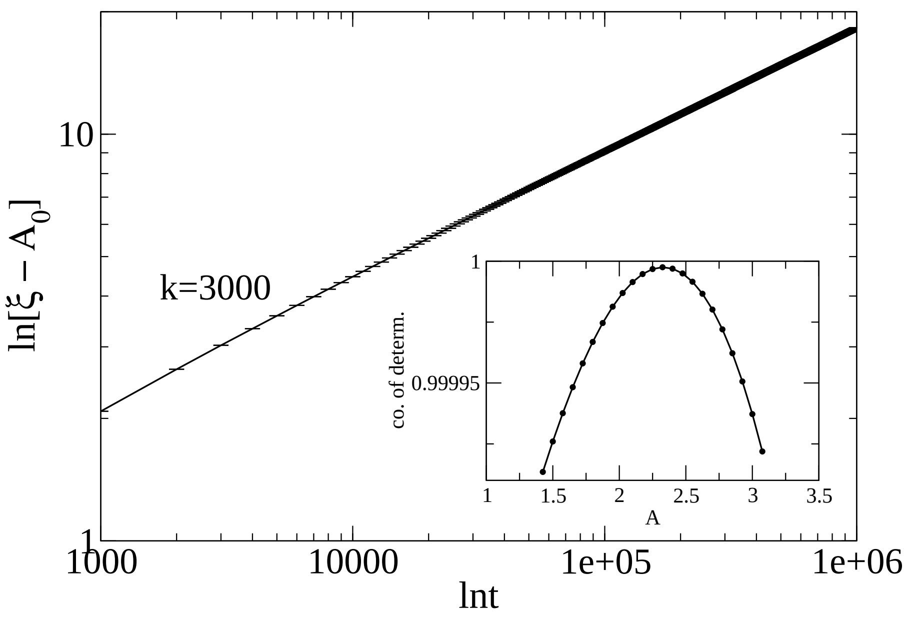

Figure 5.11: For the model with $k = 3000$. $\ln \xi'(t)$ vs. $\ln t$. The inset shows $\Re^2$ for different $A$ values. The start fitting point is $t = 50000$ MCS. Different start fitting points ranged from 30000 to 80000 MCS are tested and $A_0$ differs only by 4% at most.

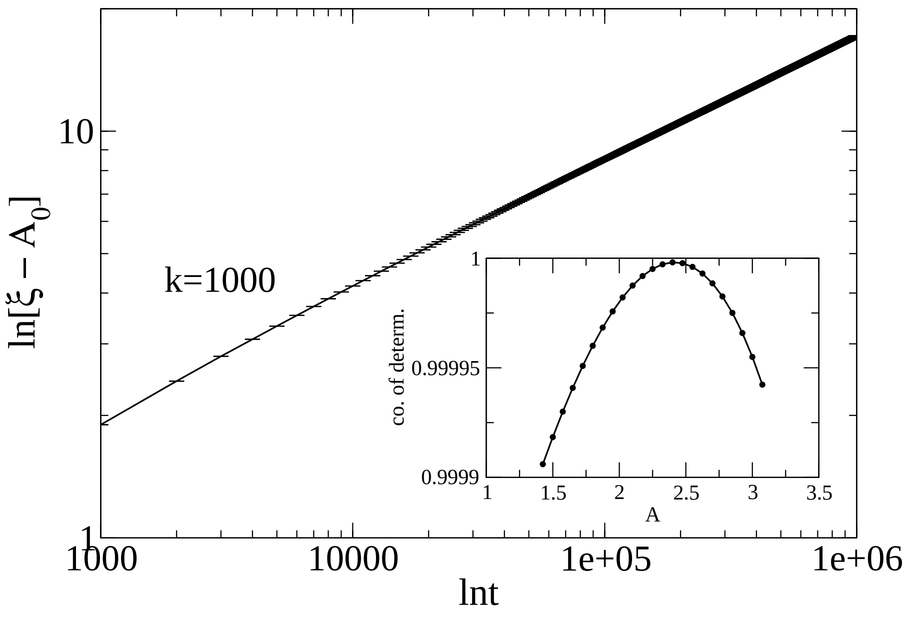

Figure 5.12: For the model with $k = 1000$. $\ln \xi'(t)$ vs. $\ln t$. The inset shows $\Re^2$ for different $A$ values. The start fitting point is $t = 50000$ MCS. Different start fitting points ranged from 30000 to 80000 MCS are tested and $A_0$ differs only by 4% at most.

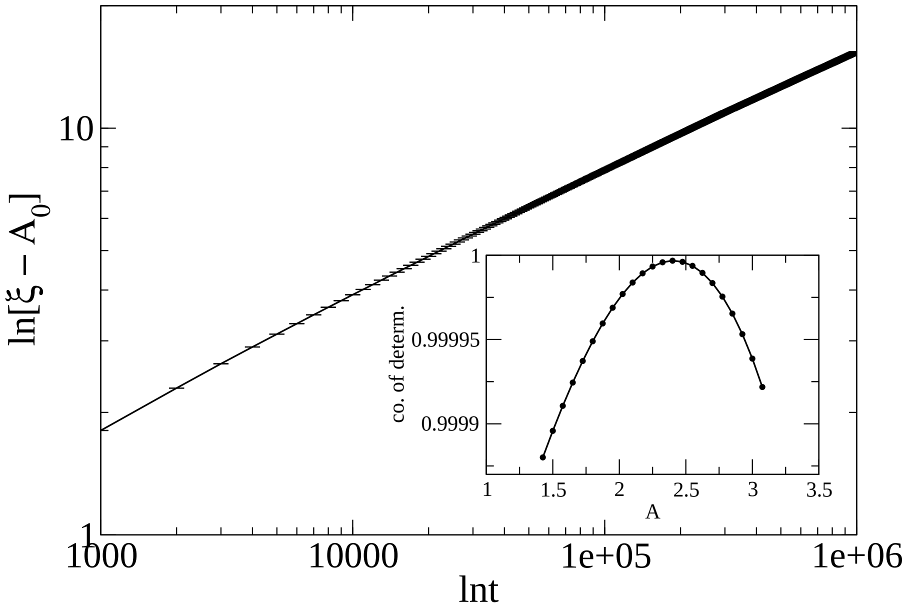

Figure 5.13: For the model with $k = 500$. $\ln \xi'(t)$ vs. $\ln t$. The inset shows $\Re^2$ for different $A$ values. The start fitting point is $t = 50000$ MCS. Different start fitting points ranged from 30000 to 80000 MCS are tested and $A_0$ differs only by 4% at most.

All the insets are obtained from fitting Eq. 5.2 with the starting point $t = 50000$ MCS. Different start fitting points ranged from 30000 to 80000 MCS are tested and there is no much difference in the results. Specifically, the change of $A_0$ with different start fitting points is only by 4% at most. The $\Re^2$ scores are rather good for each curve and it drops down slowly and smoothly in both directions away from its peak. We take the $A$ value at the peak as the one to be used in our fitting. And the second nearest neighbors of $A$ in both directions, which are also the points starting from where the curve begins to drop, are manually taken as error bar of $A$. We show $A$'s for our models in Table 5.5. The error bar

Table 5.5: $A$ in Eq. 1.2 for systems with different elasticities

| $k$ | $A$ |
|---:|---:|
| rigid | 3.2 |
| 120000 | 2.8 |
| 20000 | 2.3 |
| 3000 | 2.3 |
| 1000 | 2.4 |
| 500 | 2.4 |

of $A_0$ is taken as $\pm 0.1$. And correspondingly, we shown $n$'s in Table 5.6.

Table 5.6: $n$ and the standard error for systems with different elasticities

| $k$ | $n$ | error |
|---:|---:|---:|
| rigid | 0.329 | 0.003 |
| 120000 | 0.320 | 0.004 |
| 20000 | 0.306 | 0.004 |
| 3000 | 0.304 | 0.004 |
| 1000 | 0.303 | 0.004 |
| 500 | 0.298 | 0.003 |

We can tell that the overall trend of $n$ decreases with more elastic models. However, our technique of analysis is not good enough to give a high-resolution estimation of $n$ to make clear distinctions among models with varying elasticity constants. A more accurate way of

analyzing the data is needed to get a better estimation of $n$ values, considering the current difficulty of late-time simulation of large scale systems with adequate samples.
There are several other methods that are used in previous studies. Here we include some of them to show the results.

## 5.3 Extrapolation

A method of obtaining the domain growth exponent is by calculating $n_{eff}$ [3]. The Lifshitz-Slyozov theory has been generalized qualitatively to apply to the case of equal fractions of the two phases by Huse [3]. Huse's theory assumes the following:

$$\frac{\partial R}{\partial t} = C_2/R^2(t) + C_3/R^3(t) + O(R^{-4}) \tag{5.3}$$

where $C_2$ corresponds to the contribution to growth from diffusion between domains through the bulk and $C_3$ corresponds to the first-order correction, due to the transport of spins along the interface between domains. Huse found the growth law from solving it:

$$n_{eff}(t) = \frac{1}{3} - C/R(t) \tag{5.4}$$

where $n_{eff}(t)$ is defined as:

$$n_{eff}(t) = \partial ln[R(t)]/\partial(\ln t) \tag{5.5}$$

and $C = \frac{1}{3}A$. By taking the domain growth exponent as an variable, this can be simply generalized to:

$$n_{eff}(t) = n - nA/R(t) \tag{5.6}$$

To quantify the domain growth exponent $n$, we calculate $n_{eff}$ and then extrapolate the value for $n$. So as a numerical approximation of $n_{eff}$ in Eq. 5.5, we use $t_f = 2t$ and $t_i = \frac{1}{2}t$ and $n_{eff}$ is calculated as:

$$n_{eff}(t) = \ln[R(t_f)/R(t_i)]/\ln(t_f/t_i) \tag{5.7}$$

Eq. 5.7 is the numerical way of computing the first order derivative of $\ln[R(t)]$ with respect to $\ln t$. $t_f$ and $t_i$ are chosen in this way to avoid fluctuations and reflect the overall trend of the time series curve shown below.

In Fig 5.14 we show $n_{eff}$ as a function of $1/R$ for the rigid model and the model with elastic constant $k = 500$. We also plot $n_{eff}$ vs. $t^{-1/3}$ for both models in Fig 5.15. By least square fitting of the linear functions in Fig 5.14 we can get the $y$ intercepts or extrapolated late-time exponents. It is quite clear that we have different domain growth exponents for these two different models. And from the graph we can tell that the domain grows with different behavior at early and late time.

In order to closely observe the overall trend of domain growth at different time points, we draw a time series plot of $n$ as a function of $t_{start}$ by varying the start fitting time $t_{start}$. The time series plot of $n$ vs. $t_{start}$ is obtained as follows: for any fixed time point $t_0$, in order to obtain each $n(t_0)$, we first calculate $n_{eff}(t)$ for each $t > t_0$, and then extrapolate $n$ from it. Then we let $n(t_0) = n$. Repeat this procedure for all $0 < t_0 < 3 \times 10^5$ MCS, and then we end up with the time series curve.

In Fig. 5.16 we plot curves for systems with different elasticities. In particular, we have the rigid model and the elastic models with $k = 120000$, $k = 20000$, $k = 3000$, $k = 1000$ and $k = 500$. For each of them we have simulations up to $3 \times 10^5$ MCS and $10^6$ MCS. If we represent the number of short runs as the first figure and the number of long runs as the second figure, we have 0/246 samples for the rigid model, 250/220 for the one with $k = 120000$, 255/125 for $k = 20000$, 150/170 for $k = 3000$, 0/278 for $k = 1000$ and 220/140 for $k = 500$ model. Each curve starts off a low point and then levels off for a certain period, and finally end up with larger error bars at around $3 \times 10^5$ MCS. Curves after that are not shown here because very large error bars become dominant and they are not as informative. The reason we have large error bars at late times is because we don't have as many samples of late times as early times. Besides that, for each independent sample, the statistics for late time domain growth is not as good as the early times. This is because for

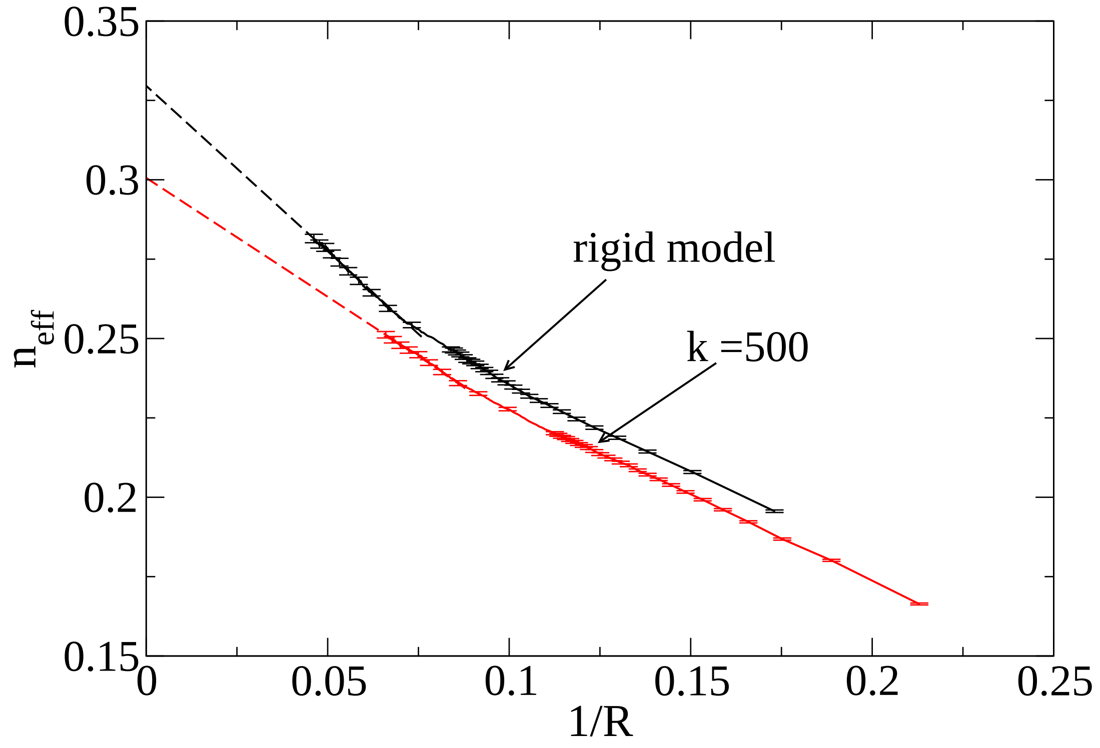

Figure 5.14: $n_{eff}$ as a function of $1/R$. Only part of the error bars are included for a better view. The dashed lines are linear fits to them with the start fitting points at the late stage. Clearly the $k = 500$ model has the $y$ intercept lower than the rigid model. That indicates the domain growth exponent $n$ is smaller for the compressible model. The intercepts we show in the plot are 0.329 and 0.301. For both systems, the line end up on different $y$ intercept points if we start the fitting procedure from different time.

each system, we don't have that many domains of large size compared with those of smaller size. For the rigid model $n = \frac{1}{3}$ is achieved within error bar in the middle leveled-off period. Clearly the elastic models have different, smaller $n$ values from the rigid model. In addition, $n$ decreases with increasing elasticity. The only exception is for the system with $k = 1000$ we cannot judge when it begins to level off, and it looks like the $k = 1000$ model violates the overall decreasing trend of $n$ among these models with increasing elasticity. It

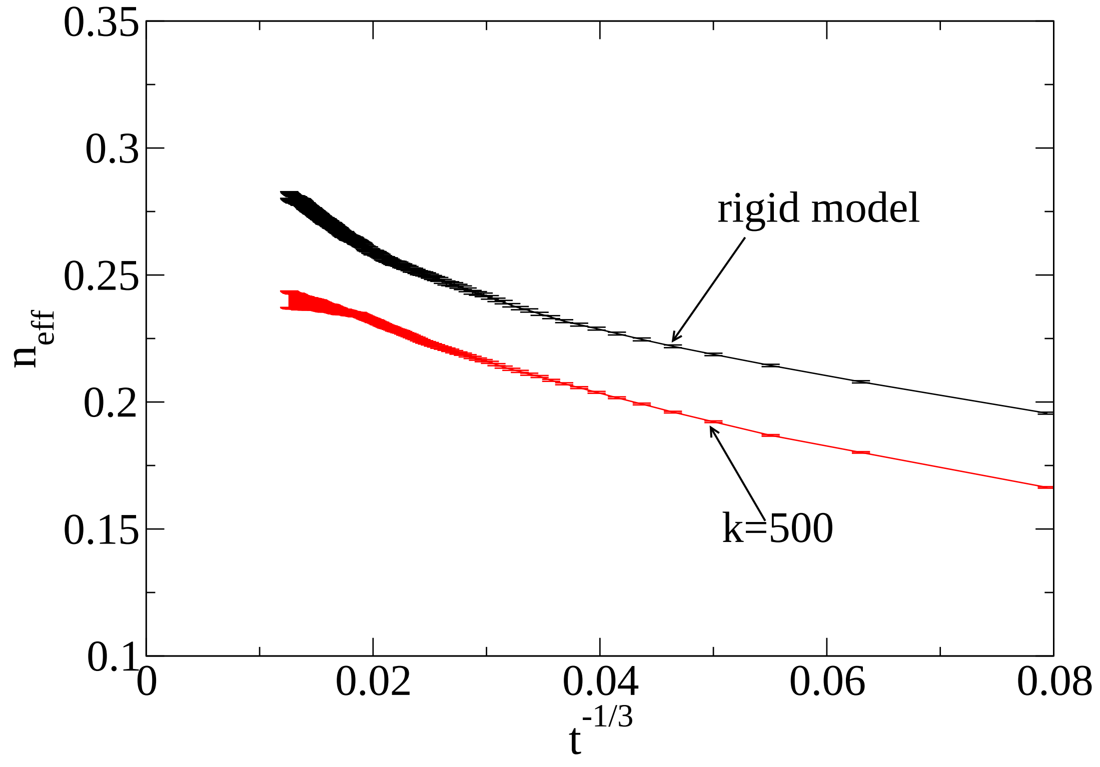

Figure 5.15: $n_{eff}$ as a function of $t^{-1/3}$. Clearly the $k = 500$ model has the $y$ intercept lower than the rigid model. That indicates the domain growth exponent $n$ is smaller for the compressible model.

maybe because our sample numbers are still not big enough to adequately reflect the system's domain growth behavior.

## 5.4 Local Exponent Analysis

We also perform a ratio analysis [1], which is more sensitive to the local behavior of domain growth. The "local exponent" is defined as:

$$n_{loc}(t) = \frac{\log[\frac{\xi\prime(t+\Delta)}{\xi\prime(t-\Delta)}]}{\log[\frac{t+\Delta}{t-\Delta}]} \tag{5.8}$$

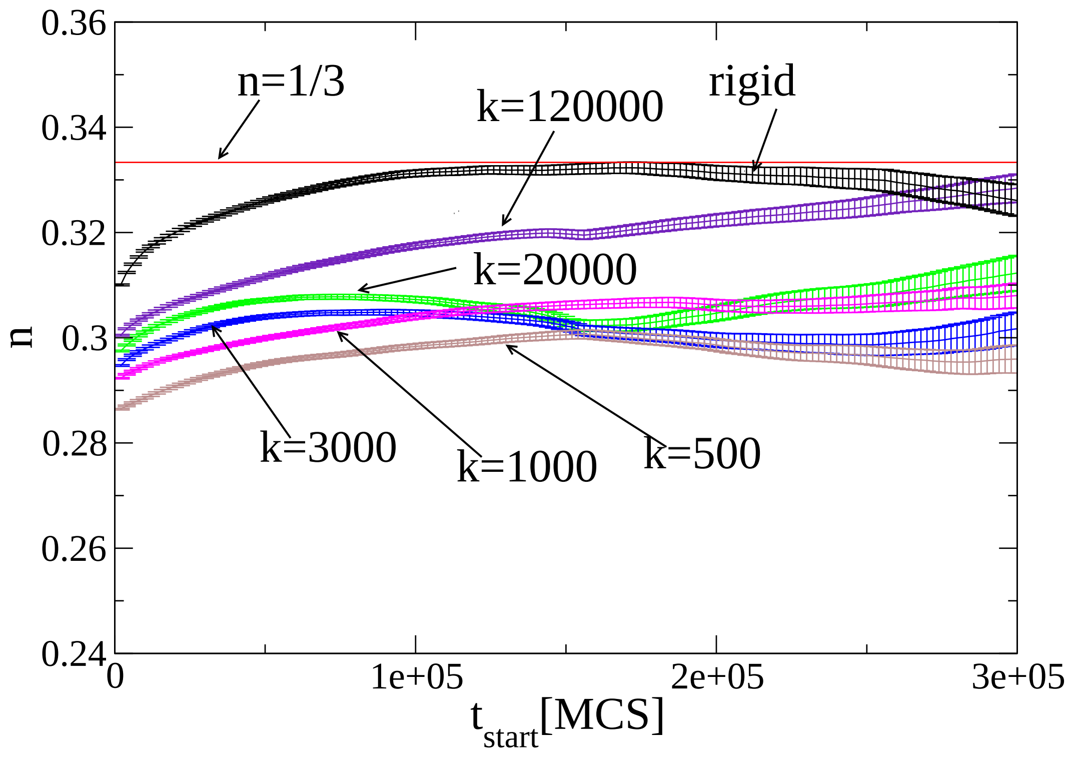

Figure 5.16: $n$ as extrapolated from Eq. 5.6. Time series plot for $n$ vs. $t_{start}$. $n = \frac{1}{3}$ is achieved in the steady period of the rigid model curve. The models with more elasticity have smaller $n$ values. The error bars on each curve become larger at late times and so we only show the results up to $3 \times 10^5$ MCS.

where $\xi\prime(t) = \xi(t) - A$. It is a numerical approximation of the first order derivative of $\log[\xi'(t)]$ with respect to $\log[t]$. $\Delta$ is the step width used in the numerical analysis. We use the same $A$ as the ones that we use above in Sec. 5.2.

Because the ratio method is very sensitive to small statistical errors in $\xi(t)$, we calculate the bin average of the final sequence. So $\overline{n}_{loc}(t)$ is the average of $n_{loc}(t_i)$'s where $t_i$ scattered symmetrically around $t$. Bin size means how many $t_i$'s are used in calculating $\overline{n}_{loc}(t)$. Here we use bin size of 16 points (every point is separated $10^3$ MCS from each

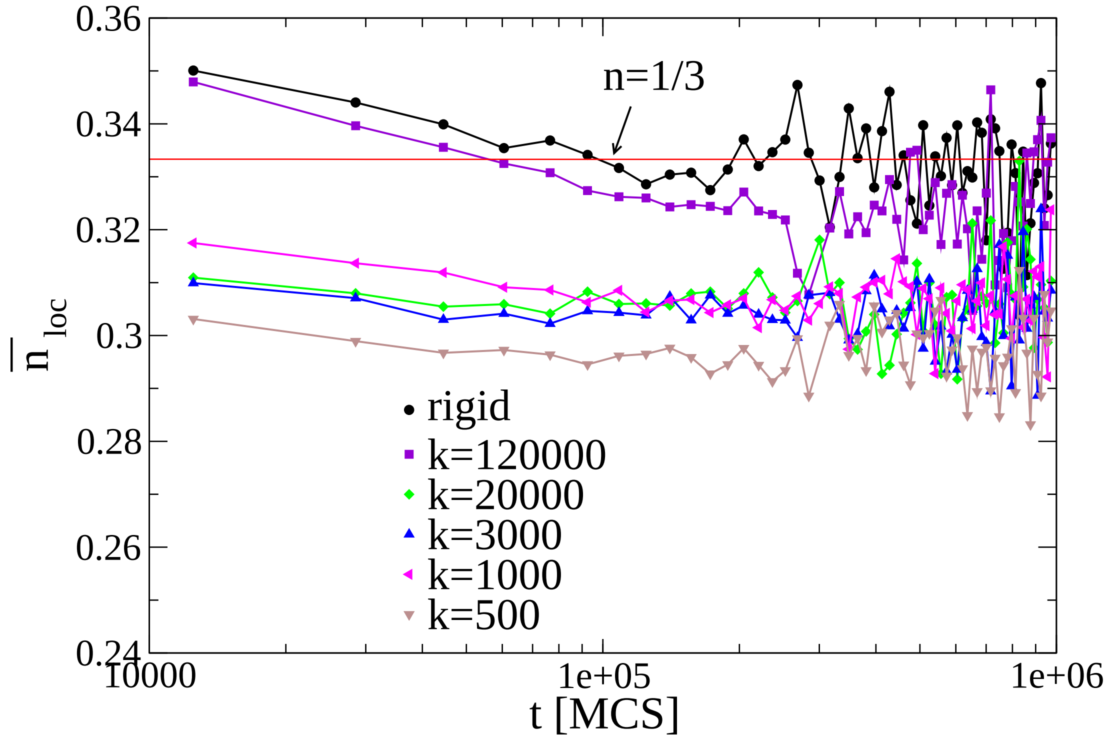

Figure 5.17: The ratio method. $\Delta = 4 \times 10^3$ MCS, bin averaged over 16 points (original point separation $10^3$ MCS).

other.) and $\Delta = 4 \times 10^3$ MCS. Fig. 5.17 demonstrates that $\overline{n}_{loc}$ for elastic models gradually deviates from $\frac{1}{3}$ with increasing elasticity. $\Delta$ is chosen to be $4 \times 10^3$ in order to avoid large fluctuations for small $\Delta$ and a log ratio plot with a different $\Delta$, e.g. $1 \times 10^3$ MCS is shown in Fig. 5.18, too. The plots for different choices of $\Delta$ look quite similar except that large $\Delta$ gives us small fluctuations.

To better observe the overall trend shown in each time series plot by the ratio method, we use cubic spline smoothing [17] to fit the time series curves.

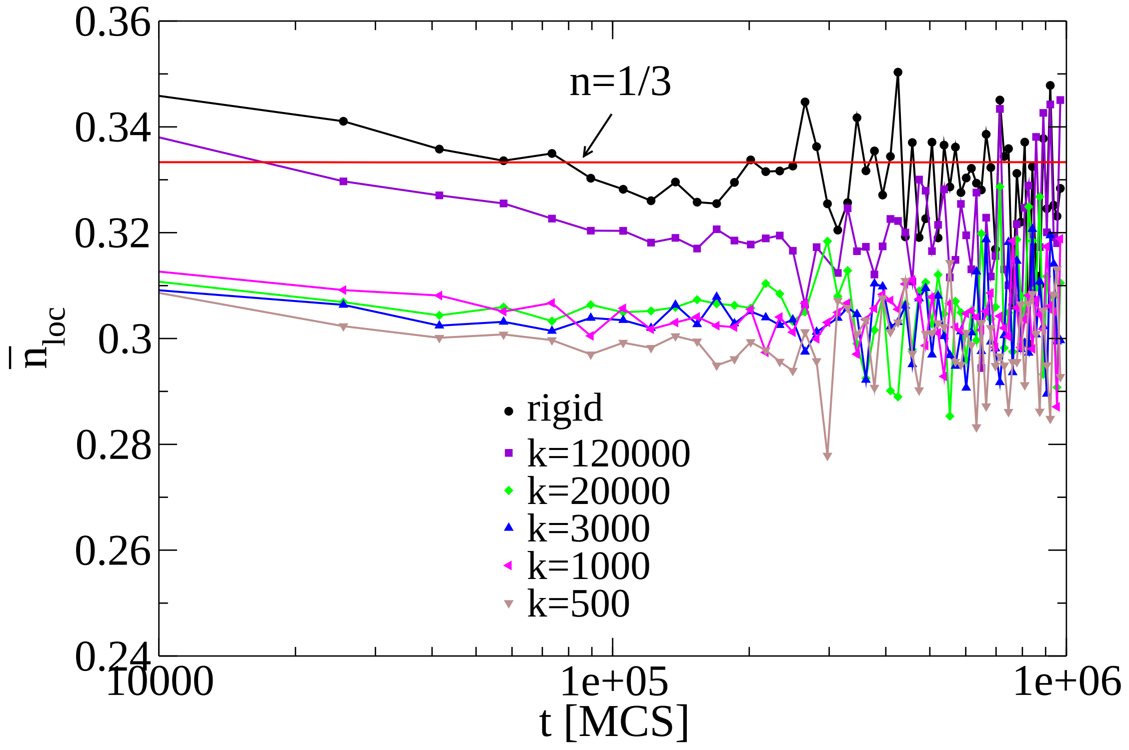

Figure 5.18: The ratio method. $\Delta = 1 \times 10^3$ MCS, bin averaged over 16 points (original point separation $10^3$ MCS).

Again Fig. 5.19 clearly indicates that $\overline{n}_{loc}(t)$ decreases with increasing elasticity, and on the other hand, for rigid model, $\overline{n}_{loc} = \frac{1}{3}$ is within its error bar. In this way we are able to get a longer steady period compared with the previous figures. The late time behavior of each system is close to the estimated $n$ value in Sec. 5.2.

## 5.5 $n$ AS A FUNCTION OF $1/k$

The results from the previous two sections don't suggest any inconsistency with our estimate of $n$ in Sec. 5.2. We can tell that the overall trend of $n$ decreases with more elastic

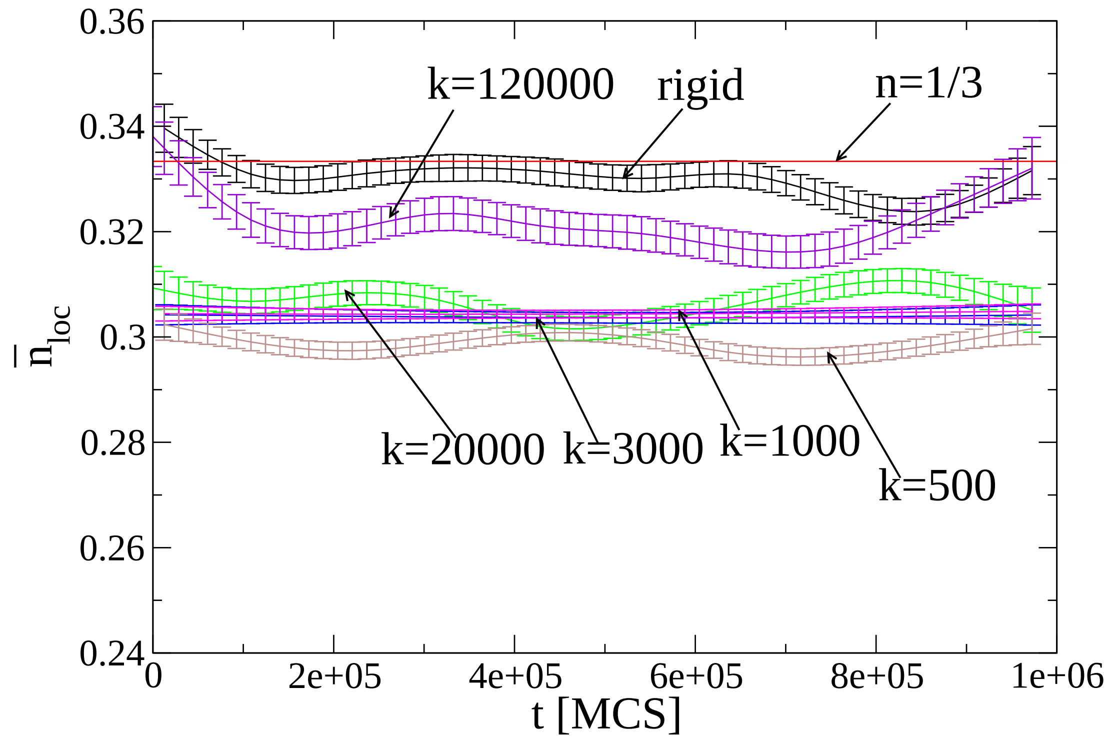

Figure 5.19: The smoothing cubic spline fitting of log ratio plot.

models. However, our technique of analysis is not good enough to give a high-resolution estimation of $n$ to make clear distinctions among models with close elasticity constants. A more accurate way of analyzing the data is needed to get a better estimation of $n$ values, considering the current difficulty of late-time simulation of large scale systems with adequate samples. We plot $n$ vs. $1/k$ in the Fig. 5.20.

From the results that we have so far, there could be two possibilities for the relationship between the domain growth exponent $n$ and the elasticity $k$. The first is that although $n = \frac{1}{3}$ for the rigid system, $n$ is approximately 0.30 for the compressible models. The system with $k = 120000$ may, in the long run, drop down and meet up with the other

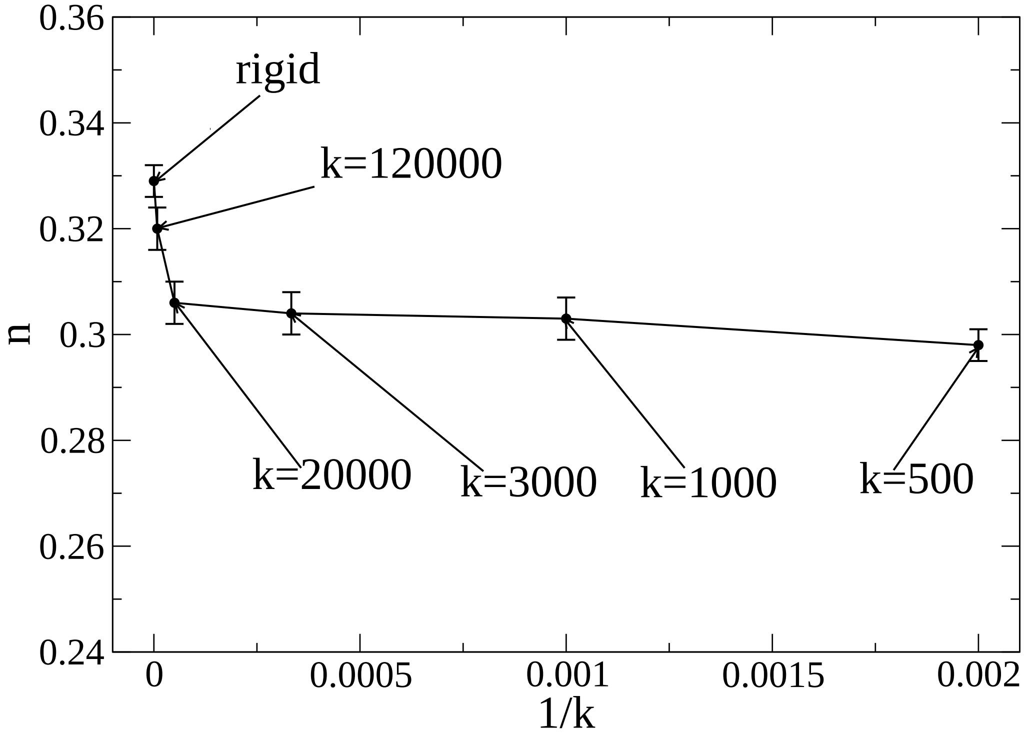

Figure 5.20: $n$ for different $1/k$

elastic systems at around $n = 0.3$. The other possibility is that $n$ drops down continuously as a function of $1/k$, so it could decrease exponentially or as a large power of $1/k$, or in some other function forms.

# Chapter 6

# Conclusion

We investigate the static and dynamic properties of a two-dimensional, compressible triangular Ising net.

The static behavior of the compressible model is not affected by the elastic force between the nearest neighbors. The Binder cumulant crossing and the collapsing of the scaled probability distribution of order parameter clearly indicate that all the compressible models belong to the two-dimensional rigid Ising model universality class. On the other hand, through our simulation of the phase separation, we find that the compressibility obviously affects the dynamic behavior of the systems. We observe the late-time domain growth exponent $n = \frac{1}{3}$ by Lifshitz-Slyozov growth law for the rigid model. For compressible models, we observe clear deviations from $n = \frac{1}{3}$, and $n$ decreases with greater elasticity of the model. Because of the nature of the problem, and also because of the computational facilities with limited power that we have, we can't guarantee that we have good enough statistics to reflect the asymptotic domain growth behavior of the model. From what we currently have, we believe there could be two different possible relationships between $n$ and $1/k$. The first is that $n \sim 0.3$ for every compressible models and when $1/k$ approaches 0, $n$ suddenly jumps up to $\frac{1}{3}$ for the rigid model. The second is that $n$ drops exponentially or as a power of $1/k$, or in some other function forms.

Besides that, our model can be easily generalized to include lattice mismatch by using different preferred bond length $l$ for different spin pairs. How the combination of compressibility and lattice mismatch can alter the domain growth exponent is of great interest and both further theoretical and computational efforts are needed.

# Bibliography

[1] D.P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics* (Cambridge University Press, New York, 2000), 2nd ed.

[2] M. Lifshitz and V.V. Slyozov, J. Phys. Chem. Solids **19**, 35 (1961).

[3] D.A. Huse, Phys. Rev. B **34**, 7845 (1986).

[4] J.G. Amar, F.E. Sullivan, and R.D. Mountain, Phys. Rev. B **37**, 196 (1988).

[5] A.J. Bray, A.D. Rutenberg, Phys. Rev. E **49**, R27 (1994).

[6] P. Nielaba, P. Fratzl, and J.L. Lebowitz, J. Stat. Phys. **95**, 23 (1999).

[7] S.J. Mitchell and D.P. Landau, Phys. Rev. Lett. **97**, 025701 (2006).

[8] H. Nishimori and A. Onuki, Phys. Rev. B **42**, 980 (1990).

[9] C. Sagui and R.C. Desai, Phys. Rev. Lett. 74, 1119 (1995).

[10] S.K. Das, S. Puri, J. Horbach, and K. Binder, Phys. Rev. Lett. **96**, 016107 (2006).

[11] B. Dünweg and D.P. Landau, Phys. Rev. B **48**, 14182 (1993).

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

[13] M. Laradji, D.P. Landau, and B. Dünweg, Phys. Rev. B **51**, 4894 (1995).

[14] F. Tavazza, D.P. Landau, and J. Adler, Phys. Rev. B **70**, 184103 (2004).

[15] E.M. Vanderworp and K.E. Newman, Phys. Rev. B **55**, 14222 (1997).

[16] R. Steel and J. Torrie, *Principles and Procedures of Statistics* (McGraw-Hill, New York, 1960).

[17] T.J. Hastie and R.J. Tibshirani, *Generalized Additive Models* (Chapman and Hall, 1990).

[18] P.C. Kelires and J. Tersoff, Phys. Rev. Lett. **63**, 1164 (1989).

[19] P.C. Weakliem and E.A. Carter, Phys. Rev. B **45**, 13458 (1992).

[20] E.M. Vandeworp and K.E. Newman, Phys. Rev. B **55**, 14222 (1997).

[21] C. Tzoumanekas and P.C. Kelires, Phys. Rev. B **66**, 195209 (2002).

[22] X. Zhu, F. Tavazza, D.P. Landau, and B. Dünweg, Phys. Rev. B **72**, 104102 (2005).

[23] L. Cannavacciuolo and D.P. Landau, Phys. Rev. B **71**, 134104 (2004).

[24] X. Zhu, D.P. Landau and N.S. Branco, Phys. Rev. B **73**, 064115 (2006),

[25] D.J. Bergman and B.I. Halperin, PHys. Rev. B **13**, 2145 (1976).

[26] A.D. Bruce and N.B. Wilding, Phys. Rev. Lett. **68** 193 (1992).

[27] K. Binder, Z. Phys. B **43**, 119 (1981).

[28] C. Wagner, Z. Elektrochem, **65**, 581 (1961).

[29] A.J. Bray, Adv. Pys. **43**, 357 (1994).

[30] A. Onuki, *Phase Transition Dynamics* (Cambridge University Press, Cambridge, England, 2002).

[31] J.D. Gunton, M.S. Miguel, and P.S. Sahni, in *Phase Transitions and Critical Phenomena*, edited by C. Domb and J.L. Lebowitz (Academic, New York, 1983), Vol. 8.

[32] K. Kawasaki, *Phase Transitions and Critical Phenomena* (Academic Press, London, 1972)

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

[34] N. Metropolis and S. ulam, Journal of the American Staistical Association **44**, 335 (1949).

[35] Alan M. Ferrenberg, Robert H. Swendsen(1988), Phys Rev.Lett, **61**,23

[36] J.G. Kissner, 1992, Ph.D Thesis, University of Manchester

[37] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer Series in Statistics, 2009)

[38] T.W. Burkhardt and B. Derrida, Phys. Rev. B **32**, 7273 (1985).

[39] A.D. Bruce, J. Phys. A **18**, L873 (1985).

[40] D.P. Landau and D. Stauffer, J. Phys. (Paris) **50**, 509 (1989).

[41] E. Brezin and J. Zinn-Justin, Nucl. Phys. B *257*, 867 (1985).

[42] A.M. Ferrenberg and D.P. Landau, Phys. Rev. B **44**, 5081 (1991), and references therein.

[43] G. Kamieniarz and H.W.J. Bölte, J. Phys. A: Math. Gen. **21**, 233 (1993).

[44] H.E. Stanley, *An Introduction to Phase Transitions and Critical Phenomena* (Oxford University Press, Oxford, 1971).

[45] M.E. Fisher, Rev. Mod. Phys. **46**, 597 (1974).

[46] M.E. Fisher, J. Math. Phys. **4**, 278 (1963).

[47] M.E. Fisher, J. Math. Phys. **5**, 944 (1964).

[48] M.E. Fisher, Rep. Prog. Phys. **30**, 615 (1967).

[49] L.P. Kadanoff et al. Rev. Mod. Phys. **39**, 395 (1967).

# Appendices

DOC.H

```
//// this is for a ferromagnetic triangular model;
const int  L=512;
const int N=L*L;
const double J0=-200.0;    //100-120
//#define J0       -1.0    //100-120
const double Jaa=J0;
const double Jbb=J0;
const double Jab=J0;
const double Jba=J0;    // Jba = Jab;
#define gamma 4.5
const double k=500.0 ;
const double  l0=1.0;
const double  laa=l0;
const double  lbb=l0;
const double  lab=l0;
const double  lba=l0;  // lab = lba;
extern int im[L+1];
extern int ip[L+1];
```

COUPLEDATA.CC

```
//coupledata.cc include the main function;
int spinaccpt = 0;
int spintry = 0;
int moveaccpt = 0;
int movetry = 0;
int rescaletry = 0;
int rescaleaccpt = 0;
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include <time.h>
#include <iostream>
#include <fstream>
#include "r1279.cc"
using namespace std;
#include "doc.h"
int im[L+1],ip[L+1];
double xaver = 1.0;
double yaver = 0.866*xaver;
double T=16.378;
int to;
double x[ L + 1 ][ L + 1 ], y[ L + 1 ][ L + 1 ];
int lattice[ L + 1 ][ L + 1 ];
int type[ L + 1 ][ L + 1 ][ 4 ];
double dis[ L + 1 ][ L + 1 ][ 4 ];
double mag_energy[ L + 1 ][ L + 1 ][ 4 ],
       elas_energy[ L + 1 ][ L + 1 ][ 4 ];
double temp_magen[ L + 1 ][ L + 1 ][ 4 ];
double temp_elasen[ L + 1 ][ L + 1 ][ 4 ];
double temp_dis[ L + 1 ][ L + 1 ][ 4 ];
double temp_x[ L + 1 ][ L + 1 ], temp_y[ L + 1 ][ L + 1 ];
int temptype[L+1][L+1][4];
double r10[L/2],C10[L/2];
double r01[L/2],C01[L/2];
double r11[L/2],C11[L/2];
double rn11[L/2],Cn11[L/2];
double ther[L/2],theC[L/2];
double samps[L/2];
#include "initialize.cc"
#include "spin_exchange.cc"
#include "spin_move.cc"
#include "global_rescale.cc"
#include "updatedata.cc"
int main(char* argc, char* argv[] ){
int seed=atoi(argv[1]);
rinitialize(seed);
cout<<"L="<<L<<" T="<<T<<" k="<<k<<" J0="<<J0<<" gamma="<<gamma<<" with\
random number "<<seed<<endl;
const int equistep=5000;
const int totalstep=1005000;
int round=0;
initialize( );
update( equistep, totalstep );
}
```

UPDATEDATA.CC

```
//this "data" version will process the data automatically
//without storing it in disk space;
int kmax=30;
void write4corr(int mcs)
```

```
{
  int i,j;
  char s[512];

//  cout<<mcs<<endl;
  for(i=0;i<kmax;++i){
 /*    fprintf(ofp,"%f %f %f %f %f %f\n",
             r01[i],C01[i],
             r10[i],C10[i],
             rn11[i],Cn11[i]);
*/
    cout<<r01[i]<<" "<<C01[i]<<" "<<r10[i]<<" "<<C10[i]<<
    " "<<rn11[i]<<" "<<Cn11[i]<<endl;
  }
}
void calcCdir(int di,int dj)
{
  int i,j,k;
  int ti,tj;
  double r,rx,ry;
  int tmpi,tmpj;
  int rindex;
  double fact;
  double diag;
  double X=xaver*L;
  double Y=yaver*L;
  double Xhalf=X/2.0;
double Yhalf=Y/2.0;
if(di==1&&dj==0)
  fact = 0.866/yaver;
else if(di==0&&dj==1)
  fact = 1.0/xaver;
else if(di==-1&&dj==1){
  diag = sqrt((yaver*yaver)+
          (xaver-0.5*yaver/0.866)*(xaver-0.5*yaver/0.866));
  fact = 1.0/diag;
}
for(k=0;k<L/2;++k)
{
  ther[k]=0.0;
  theC[k]=0.0;
  samps[k]=0.0;
};
//index over all atomic positions
for(i=1;i<=L;i++)
  for(j=1;j<=L;j++)
  {
    //index over all relative positions along the di,dj direction
    for(k=0;k<kmax;k++)
    {
      ti = i;
      tj = j;
      if(di==1&&dj==0){
         ti += k;
         if(ti>L)
             ti -= L;
      }
      else if(di==0&&dj==1){
         tj += k;
         if(tj>L)
             tj -= L;
      }
      else if(di==-1&&dj==1){
         ti -= k;
         tj += k;
         if(ti<1)
             ti += L;
         if(tj>L)
             tj -= L;
      }
      rx=fabs(x[i][j]-x[ti][tj]);
      ry=fabs(y[i][j]-y[ti][tj]);
      if(rx>=Xhalf)
         rx-=X;
      if(ry>=Yhalf)
         ry-=Y;
      r=sqrt(rx*rx+ry*ry);
           //index in the storage arrays
      rindex=(int) ((r+0.5)*fact);
      if(rindex<0)
      {
```

```
            exit(1);
          };
          if(rindex<L/2)
          {
            samps[rindex]+=1.0;
            ther[rindex]+=r;
            theC[rindex]+=1.0*lattice[i][j]*lattice[ti][tj];
          }
          else
          {
          };
        };
      };
   for(k=0;k<kmax;++k)
   {
      ther[k]/=1.0*samps[k];
      theC[k]/=1.0*samps[k];
   };
}
void calcCs(void)
{
   int k;
   calcCdir(1,0);
   for(k=0;k<=kmax;k++)
   {
     r10[k]=ther[k];
     C10[k]=theC[k];
```

```
  };
  calcCdir(0,1);
  for(k=0;k<=kmax;k++)
  {
    r01[k]=ther[k];
    C01[k]=theC[k];
  };
  calcCdir(-1,1);
  for(k=0;k<=kmax;k++)
  {
    rn11[k]=ther[k];
    Cn11[k]=theC[k];
  };
}
void update( int equistep, int totalstep ){
  int i,j,w;
  int i1,j1,k1;
  double r, r1, r2;
  int mcs;
  int t;
    // pick up a random spin;
  for( mcs = 1; mcs <= totalstep; mcs ++ ){  //1500
    if(mcs==equistep+1)
      T=5.422;
    for( t = 1; t <= N; t ++ ){
      r1 = ranf() * N;
      for( w = 1; w <= L; w ++ ){
          if( r1/L > ( w - 1 ) * 1.0 && r1/L <= w * 1.0 )
              i = w;    // i is the x index;
      }
      r2 = r1 - 1.0 * ( i - 1 ) * L;
      for( w = 1; w <= L; w ++ ){
          if( r2 > ( w - 1 ) * 1.0 && r2 <= w * 1.0 )
              j = w;    // j is the y index;
      }
      r = ranf();
      if( r <= 0.5 ){
          spin_exchange(  i, j );
      }
      else{
          spin_move(  i, j );
      }
    }
    global_rescale();
    if( mcs - equistep >=1000 && (mcs-equistep)%1000 == 0 )
    {
      cout<<mcs<<endl;
      cout<<xaver<<" "<<yaver<<endl;
  //        for(register int i1=1; i1<=L;++i1)
     //        for(register int j1=1; j1<=L;++j1)
          //   cout<<x[i1][j1]<<" "<<y[i1][j1]<<" "<<lattice[i1][j1]<<endl;
      ////now I want to process the  data;
      calcCs();
      write4corr(mcs);
    }
  }
}
```

SPIN_MOVE.CC

```
// propose a spin move;
void spin_move( int i, int j  ){
   double r;
   double disx, disy, newdis[ 7 ];
   double temp_magen[ 7 ];
   double temp_elasen[ 7 ];
   int    index0, index1, index2, index3;
   int    ini,inj;
   int order;
   double deltah = 0.0;
   double diffx,diffy;
   double range = sqrt(xaver*xaver+yaver*yaver)*0.3;
   double  r1 = range * ( 1.0 - 2.0 * ranf() );
   double  r2 = range * ( 1.0 - 2.0 * ranf() );
   double  x1 = x[ i ][ j ] + r1;
   double  y1 = y[ i ][ j ] + r2;
   double  xmove = x1 - xaver*(i-3)/2.0-j*xaver;
   double  ymove = y1 - yaver*(L-i);
   for( int t = 1; t <= 6; t ++ ){
     if( t == 1 ){
        index0 = i;
        index1 = j;
        index2 = im[ i ];
        index3 = ip[ j ];
        order = 1;
        ini = i;
        inj = j;
        diffx = xaver/2.0;
        diffy = yaver;
    }
    if( t == 2 ){
        index0 = i;
        index1 = j;
        index2 = i;
        index3 = ip[ j ];
        order = 2;
        ini = i;
        inj = j;
        diffx=xaver;
        diffy=0.0;
    }
    if( t == 3 ){
        index0 = i;
        index1 = j;
        index2 = ip[ i ];
        index3 = j;
        order = 3;
        ini = i;
        inj = j;
        diffx=xaver/2.0;
        diffy=-1.0*yaver;
    }
    if( t == 4 ){
        index2 = ip[ i ];
        index3 = im[ j ];
        index0 = i;
        index1 = j;
        order = 1;
        ini = ip[ i ];
        inj = im[ j ];
        diffx=-1.0*xaver/2.0;
        diffy=-1.0*yaver;
    }
    if( t == 5 ){
        index2 = i;
        index3 = im[ j ];
        index0 = i;
        index1 = j;
        order = 2;
        ini = i;
        inj = im[ j ];
        diffx=-1.0*xaver;
        diffy=0.0;
    }
    if( t == 6 ){
        index2 = im[ i ];
```

```
      index3 = j;
      index0 = i;
      index1 = j;
      order = 3;
      ini = im[ i ];
      inj = j;
      diffx=-1.0*xaver/2.0;
      diffy=yaver;
    }
    // new distance;
    double xmove1 = x[index2][index3] - xaver*(index2-3)/2.0
                   -index3*xaver;
    double ymove1 = y[index2][index3] - yaver*(L-index2);
    disx = xmove1 + diffx - xmove;
    disy = ymove1 + diffy - ymove;
    newdis[ t ] = sqrt( disx * disx + disy * disy );
    temp_magen[ t ] = mag_energy[ ini ][ inj ][ order ] *
      exp( -1 * gamma * ( newdis[ t ] - dis[ ini ][ inj ][ order ] ) );
    temp_elasen[ t ] = k/2.0 * ( newdis[ t ] - l0 ) *
      ( newdis[ t ] - l0 );
    deltah = deltah + temp_magen[ t ] -
      mag_energy[ ini ][ inj ][ order ]  +
      temp_elasen[ t ] - elas_energy[ ini ][ inj ][ order ];
  }
  r = ranf();
  if( r <= exp( -1.0 * deltah / T ) ){
     moveaccpt++;
```

```
for( int t = 1; t <= 6; t ++ ){
   if( t == 1 ){
      index0 = i;
      index1 = j;
      order = 1;
   }
   if( t == 2 ){
      index0 = i;
      index1 = j;
      order = 2;
   }
   if( t == 3 ){
      index0 = i;
      index1 = j;
      order = 3;
   }
   if( t == 4 ){
      index0 = ip[ i ];
      index1 = im[ j ];
      order = 1;
   }
  if( t == 5 ){
     index0 = i;
     index1 = im[ j ];
     order = 2;
  }
  if( t == 6 ){
```

```
            index0 = im[ i ];
            index1 = j;
            order = 3;
          }
          x[ i ][ j ] = x1;
          y[ i ][ j ] = y1;
          mag_energy[ index0 ][ index1 ][ order ] = temp_magen[ t ];
          elas_energy[ index0 ][ index1 ][ order ] = temp_elasen[ t ];
          dis[ index0 ][ index1 ][ order ] = newdis[ t ];
      }
    }
    else{
    }
}
```

SPIN_EXCHANGEE.CC

```
//propose a spin exchange;
void spin_exchange( int i, int j  ){
double r = ranf();
int temp;
double deltah1=0.0;
double deltah2=0.0;
double deltah = 0.0;
double pair = ranf();
if(pair>0.333333&&pair<0.666666){
 if( lattice[ i ][ j ] == lattice[ i ][ ip[ j ] ] ){
     deltah = 0.0;
```

```
        spinaccpt ++;
    }
    else{
        deltah1 = -2.0*( mag_energy[i][j][1] + mag_energy[i][j][3]
        + mag_energy[im[i]][j][3] + mag_energy[i][im[j]][2] +
        mag_energy[ip[i]][im[j]][1]+ mag_energy[i][ip[j]][1] +
        mag_energy[i][ip[j]][2] + mag_energy[i][ip[j]][3] +
        mag_energy[im[i]][ip[j]][3] + mag_energy[ip[i]][j][1] );
        deltah = deltah1;
        if( r <= exp( -1.0 * deltah / T ) ){
          temp = lattice[ i ][ ip[ j ] ];
          lattice[ i ][ ip[ j ] ] = lattice[ i ][ j ];
          lattice[ i ][ j ] = temp;
          mag_energy[ i ][ j ][ 1 ] = -1.0*mag_energy[ i ][ j ][ 1 ];
          mag_energy[ i ][ j ][ 3 ] = -1.0*mag_energy[ i ][ j ][ 3 ];
          mag_energy[ i ][ ip[ j ] ][ 1 ] =
              -1.0*mag_energy[ i ][ ip[ j ] ][ 1 ];
          mag_energy[ i ][ ip[ j ] ][ 2 ] =
              -1.0*mag_energy[ i ][ ip[ j ] ][ 2 ];
          mag_energy[ i ][ ip[ j ] ][ 3 ] =
               -1.0*mag_energy[ i ][ ip[ j ] ][ 3 ];
          mag_energy[ im[ i ] ][ j ][ 3 ] =
               -1.0*mag_energy[ im[ i ] ][ j ][ 3 ];
          mag_energy[ i ][ im[ j ] ][ 2 ] =
               -1.0*mag_energy[ i ][ im[ j ] ][ 2 ];
          mag_energy[ ip[ i ] ][ im[ j ] ][ 1 ] =
               -1.0*mag_energy[ ip[ i ] ][ im[ j ] ][ 1 ];
        mag_energy[ im[ i ] ][ ip[ j ] ][ 3 ] =
            -1.0*mag_energy[ im[ i ] ][ ip[ j ] ][ 3 ];
        mag_energy[ ip[ i ] ][ j ][ 1 ] =
            -1.0*mag_energy[ ip[ i ] ][ j ][ 1 ];
        spinaccpt ++;
      }
      else{
      // reject;
      }
    }
  }
  if(pair<0.333333){
    if( lattice[ i ][ j ] == lattice[ ip[i] ][ j ] ){
          deltah = 0.0;
          spinaccpt ++;
     }
    else{
          deltah1 = -2.0*( mag_energy[i][j][1] + mag_energy[i][j][2] +
          mag_energy[im[i]][j][3] +  mag_energy[i][im[j]][2] +
          mag_energy[ip[i]][im[j]][1] + mag_energy[ip[i]][im[j]][2] +
          mag_energy[ip[i]][j][1] + mag_energy[ip[i]][j][2] +
          mag_energy[ip[i]][j][3] + mag_energy[ip[ip[i]]][im[j]][1]  );
          deltah = deltah1;
          if( r <= exp( -1.0 * deltah / T ) ){
            temp = lattice[ ip[i] ][ j ];
            lattice[ ip[i] ][ j ] = lattice[ i ][ j ];
            lattice[ i ][ j ] = temp;
            mag_energy[ i ][ j ][ 1 ] = -1.0*mag_energy[ i ][ j ][ 1 ];
            mag_energy[ i ][ j ][ 2 ] = -1.0*mag_energy[ i ][ j ][ 2 ];
            mag_energy[ im[i] ][ j ][ 3 ] =
                                         -1.0*mag_energy[ im[i] ][ j ][ 3 ];
            mag_energy[ i ][ im[ j ] ][ 2 ] =
                                       -1.0*mag_energy[ i ][ im[ j ] ][ 2 ];
            mag_energy[ ip[i] ][ im[ j ] ][ 1 ] =
                                   -1.0*mag_energy[ ip[i] ][ im[ j ] ][ 1 ];
            mag_energy[ ip[ i ] ][ im[j] ][ 2 ] =
                                   -1.0*mag_energy[ ip[ i ] ][ im[j] ][ 2 ];
            mag_energy[ ip[i] ][ j ][ 1 ] =
                                         -1.0*mag_energy[ ip[i] ][ j ][ 1 ];
            mag_energy[ ip[ i ] ][ j ][ 2 ] =
                                       -1.0*mag_energy[ ip[ i ] ][ j ][ 2 ];
            mag_energy[ ip[ i ] ][ j ][ 3 ] =
                                         -1.0*mag_energy[ ip[i] ][ j ][ 3 ];
            mag_energy[ ip[ip[ i ]] ][ im[j] ][ 1 ] =
                              -1.0*mag_energy[ ip[ip[ i ] ] ][ im[j] ][ 1 ];
            spinaccpt ++;
          }
          else{
      // reject;
          }
    }
  }
  if(pair >0.666666){
    if( lattice[ i ][ j ] == lattice[ im[i] ][ ip[j] ] ){
          deltah = 0.0;
          spinaccpt ++;
    }
    else{
          deltah1 = −2.0*( mag_energy[i][j][2] + mag_energy[i][j][3] +
          mag_energy[im[i]][j][3] +  mag_energy[im[i]][j][2] +
          mag_energy[ip[i]][im[j]][1]+ mag_energy[i][im[j]][2]+
          mag_energy[im[i]][ip[j]][1] + mag_energy[im[i]][ip[j]][2] +
          mag_energy[im[i]][ip[j]][3] + mag_energy[im[im[i]]][ip[j]][3] );
          deltah = deltah1;
          if( r <= exp( −1.0 * deltah / T ) ){
            temp = lattice[ im[i] ][ ip[j] ];
            lattice[ im[i] ][ ip[j] ] = lattice[ i ][ j ];
            lattice[ i ][ j ] = temp;
            mag_energy[ i ][ j ][ 3 ] = −1.0*mag_energy[ i ][ j ][ 3 ];
            mag_energy[ i ][ j ][ 2 ] = −1.0*mag_energy[ i ][ j ][ 2 ];
            mag_energy[ im[i] ][ j ][ 3 ] =
                                  −1.0*mag_energy[ im[i] ][ j ][ 3 ];
            mag_energy[ im[i] ][ j ][ 2 ] =
                                  −1.0*mag_energy[ im[i] ][ j ][ 2 ];
            mag_energy[ ip[i] ][ im[ j ] ][ 1 ] =
                              −1.0*mag_energy[ ip[i] ][ im[ j ] ][ 1 ];
            mag_energy[ i ][ im[j] ][ 2 ] =
                                  −1.0*mag_energy[ i ][ im[j] ][ 2 ];
            mag_energy[ im[i] ][ ip[j] ][ 1 ] =
                                −1.0*mag_energy[ im[i] ][ ip[j] ][ 1 ];
            mag_energy[ im[ i ] ][ ip[j] ][ 2 ] =
                                −1.0∗mag_energy[ im[ i ] ][ ip[j] ][ 2 ];
              mag_energy[ im[i] ][ ip[j] ][ 3 ] =
                                  −1.0∗mag_energy[ im[i] ][ ip[j] ][ 3 ];
              mag_energy[ im[im[ i ]] ][ ip[j] ][ 3 ] =
                            −1.0∗mag_energy[ im[im[ i ] ] ][ ip[j] ][ 3 ];
              spinaccpt ++;
           }
           else{
         // reject;
           }
  }
  }
}
```

GLOBAL_RESCALE.CC

```
//propose a volume change;
void global_rescale(){
 double l;
 double deltah = 0.0,deltah1=0.0,deltah2=0.0;
 double disx,disy;
 int     index2, index3;
 double diffx, diffy;
 double xmove, ymove, xmove1,ymove1;
 double  r1 = 1.0 + ( 1.0 − 2 ∗ ranf() ) ∗ 0.025;
 double  r2 = 1.0 + ( 1.0 − 2 ∗ ranf() ) ∗ 0.025;
 double  xaver1 = xaver∗r1;
 double yaver1 = yaver∗r2;
```

```
for(int i = 1; i <= L; ++i ){
   for( int j = 1; j <= L; ++j ){
       temp_x[i][j] = r1 * x[i][j];
       temp_y[i][j] = r2 * y[i][j];
   }
}
for( int i = 1; i <= L; ++i ){
   for( int j = 1; j <= L; ++j ){
      for( int w = 1; w <= 3; ++w ){
         if( w == 1 ){
             index2 = im[i];
             index3 = ip[j];
             diffx = xaver1/2.0;
             diffy = yaver1;


         }
         else if( w == 2 ){
             index2 = i;
             index3 = ip[j];
             diffx = xaver1;
             diffy = 0.0;
         }
         else if( w == 3 ){
             index2 = ip[i];
             index3 = j;
             diffx = xaver1/2.0;
             diffy = -1.0*yaver1;
```

```
            }
            xmove1 = temp_x[index2][index3]-xaver1*(index2-3)/2.0
                         -index3*xaver1;
            ymove1 = temp_y[index2][index3]-yaver1*(L-index2);
            xmove = temp_x[i][j]-xaver1*(i-3)/2.0-j*xaver1;
            ymove = temp_y[i][j]-yaver1*(L-i);
            disx = xmove1 + diffx - xmove;
            disy = ymove1 + diffy - ymove;
            temp_dis[ i ][ j ][ w ] = sqrt( disx * disx + disy * disy );
            temp_magen[ i ][ j ][ w ] = exp( -1.0 * gamma *
                  ( temp_dis[ i ][ j ][ w ]
                  - dis[ i ][ j ][ w ] ) ) * mag_energy[ i ][ j ][ w ];
            l = 10;
            temp_elasen[ i ][ j ][ w ] = k/2.0 *
                  ( temp_dis[ i ][ j ][ w ] - l )
                  * ( temp_dis[ i ][ j ][ w ] - l );
            deltah1 = deltah1 + temp_magen[i][j][w]-mag_energy[i][j][w];
            deltah2 = deltah2 + temp_elasen[i][j][w]-elas_energy[i][j][w];
          }
        }
      }
      deltah = deltah1 + deltah2;
      deltah = deltah - N * T * log(r1*r2);
      double r = ranf();
      if( r <= exp( -1.0 * deltah / T) ){
         // accept;
         for( int i = 1; i <= L; i ++ ){
```

```
        for( int j = 1; j <= L; j ++ ){
           for( int w = 1; w <= 3; w ++ ){
             dis[ i ][ j ][ w ] = temp_dis[ i ][ j ][ w ];
             mag_energy[ i ][ j ][ w ] = temp_magen[ i ][ j ][ w ];
             elas_energy[ i ][ j ][ w ] = temp_elasen[ i ][ j ][ w ];

           }
           x[ i ][ j ] = temp_x[ i ][ j ];
           y[ i ][ j ] = temp_y[ i ][ j ];
         }
      }
      xaver = xaver1;
      yaver = yaver1;
      rescaleaccpt ++;
   }
   else {
     }
  }
```

SPIN_FLIP.CC

```
void spin_flip( int i, int j ){
 double r = ranf();
 int temp;
 double deltah = 0.0;
 double oldenergy,newenergy;
 oldenergy = mag_energy[i][j][1] + mag_energy[i][j][2]
  + mag_energy[i][j][3]+mag_energy[im[i]][j][3] +
     mag_energy[i][im[j]][2]+mag_energy[ip[i]][im[j]][1];
   deltah =  -2.0* oldenergy ;
   if( r <= exp( -1.0 * deltah / T ) ){
      lattice[i][j] = -1 * lattice[i][j];
      mag_energy[ i ][ j ][ 1 ] = -1.0*mag_energy[ i ][ j ][ 1 ];
      mag_energy[ i ][ j ][ 3 ] = -1.0*mag_energy[ i ][ j ][ 3 ];
      mag_energy[ i ][ j ][ 2 ] = -1.0*mag_energy[ i ][ j ][ 2 ];
      mag_energy[ im[ i ] ][ j ][ 3 ] =
                              -1.0*mag_energy[ im[ i ] ][ j ][ 3 ];
      mag_energy[ i ][ im[ j ] ][ 2 ] =
                              -1.0*mag_energy[ i ][ im[ j ] ][ 2 ];
      mag_energy[ ip[ i ] ][ im[ j ] ][ 1 ] =
                         -1.0*mag_energy[ ip[ i ] ][ im[ j ] ][ 1 ];
      spinaccpt ++;
   }
   else{
       // reject;
   }
}
```