Wang-Landau 몬테카를로 법의 κ-카바이드에의 적용에 대한 연구

A study on the application of Wang-Landau Monte Carlo method on κ-carbide

A study on the application of Wang-Landau Monte Carlo method on κ-carbide

By

Lee, Jee Yong
Computational Metallurgy
Graduate Institute of Ferrous Technology
Pohang University of Science and Technology

A thesis submitted to the faculty of Pohang University of Science and Technology in partial fulfillments of the requirements for the degree of Master of Science in the Graduate Institute of Ferrous Technology (Computational Metallurgy)

Pohang, Korea
December 19th, 2011

Approved by

Prof. H.K.D.H. Bhadeshia	Prof. In Gee Kim
Major Advisor	Co-Advisor

A study on the application of Wang-Landau Monte Carlo method on κ-carbide

Lee, Jee Yong

This dissertation is submitted for the degree of Master of Science at the Graduate Institute of Ferrous Technology of Pohang University of Science and Technology. The research reported herein was approved by the committee of Thesis Appraisal

19.12.2011

Thesis Review Committee

Chairman: Dong Woo Suh Dongwoo Suh

Member: In Gee Kim

Member: Se Kyun Kwon

MFT Jee Yong Lee
20100945 A study on the application of Wang-Landau
Monte Carlo method on κ-carbide
Department of Ferrous Technology
(Computational Metallurgy) 2012
Advisor: Prof. Bhadeshia, H.K.D.H.; Prof. Kim, In Gee
Text in English

Abstract

The Monte Carlo (MC) method is a class of computational algorithms that rely on repeated random sampling to simulate complex statistical behaviors and compute their results. In physics and material science, the MC method is especially useful for simulating systems with many coupled degrees of freedom. In this article, the application of MC method on the physical problems was dealt, by introducing the relationship with thermodynamics and statistical mechanics. The Wang-Landau algorithm, which is one of the non-Boltzmann reweighting sampling methods, is known for its efficiency and wide applicability on various problems. It estimates the density of states (DOS) of the objective material system by histogram reweighting, and calculate properties of the system at wide range of temperatures with the estimated DOS. We developed the program code of the Wang-Landau algorothm and tested the feasibility of the code with simple models. And we made a ‘cell gas’ model which deals each cell of structure as a non-

interacting gas. Finally, with the code and the model, we calculated basic thermodynamic properties of κ-carbide.

Contents

Nomenclature

A	A certain property of a thermodynamic system
C	The specific heat
d	The size of brittle particle of the particle
DOS	Density of states
E	The total energy of the system
e	The entire set of microstates
F	The Helmholtz free energy
G	The Gibbs free energy
$g(E)$	The density of states at energy level E
H	The Hamiltonian of the system
h	The external potential
J	The interaction constant
k_{B}	The Boltzmann constant
L	The size of one side of the lattice
M	The total magnetization
m	The size of sample in statistics
N	The number of particles of the system, or the number of lattice sites
P	The probability

p	The pressure of the system
Q	The partition function for classical system
S	The entropy
T	The temperature
T_{c}	The critical temperature
U	The internal energy
V	The volume of the system
β	$1/k_{\mathrm{B}}T$
$\varepsilon(\sigma_i)$	The energy of state σ_i
μ	The chemical potential
ν	A microstate
σ	The standard deviation, or the tensile stress acting on the particle
σ_i	The spin of the lattice site i
τ	The time
ψ	The wavefunction of Schrödinger equation
$\Omega(E)$	The number of states of the system at energy level E

1. Introduction

1.1 Monte Carlo method

"Monte Carlo" (MC) method is a class of computational algorithms that rely on repeated random sampling to simulate complex statistical behaviors and compute their results. Thus, MC involves simulation of an objective system. A large number of different problems, especially mathematical and physical systems, are simulated this way: multidimensional integrals, the stock market, phase transitions in materials, radiation damage, space exploration and many other such problems have all been the subject of MC simulations.

In physics and materials science, the MC method is especially useful for simulating systems with many coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, and cellular structures. So, it tends to be used when it is infeasible to compute an exact result with deterministic algorithms, and it models phenomena with significant uncertainty in inputs.

The MC method was attempted in the 1930s by Enrico Fermi while studying neutron diffusion, but he did not publish anything on it. And again in the 1940s it was made and used by John von Neumann, Nicholas Metropolis, and Stanislaw Ulam, in the Manhattan Project – nuclear weapon

development. Ulam and his Manhattan project team used the MC method to study radiation shielding and neutron penetration. They had most of the necessary data, but the problem could not be solved with analytical calculations. Then, they had the idea of making use of random experiments. The main idea of the MC method was proposed by the “genius” Ulam, who thought of it while he was convalescing from an illness, and playing solitaire, so the MC method is similar to playing card games in its random and statistical nature. The name “Monte Carlo” came from the famous Monte Carlo Casino.

But actually, its primitive idea had already popped out about 200 years before Ulam’s thought, by Comte de Buffon. This idea is a famous one in mathematics history, which is named “Buffon’s needle problem” after the name of the inventor. Its question is: “Suppose we have a floor made of parallel strips of wood, each the same width, and we drop a needle onto the floor. What is the probability that the needle will lie across a line between two strips?” And in a more mathematical way: “Given a needle of length l dropped on a plane ruled with parallel lines t units apart, what is the probability that the needle will cross a line?” This was the earliest problem in geometric probability to be solved; it can be solved using integral geometry. But it can also be solved statistically by a sampling of random throwing experiments, where, as the number of samples gets bigger, the result of the

solution gets more accurate. In 1901, Italian mathematician Mario Lazzarini performed Buffon's needle experiment, and after 3408 tosses, he gathered all samples he made and used of the result to calculate the value of π. This is possible when the length *l* of needle is shorter than the length *t* between two strips. Here, the method of getting π from the sampling of Buffon's needle experiment will not be discussed, but the next chapter will discuss this in another easier way that shows the essence of the MC method well.

References for section 1.1: Newman and Barkema, Ch. 1; Wikipedia: Monte Carlo method

1.2 Calculation of π with Monte Carlo method

Monte Carlo method is implemented in many ways, but generally follows a common pattern:

1. Define a domain of possible inputs.
2. Generate inputs randomly from a probability distribution over the domain.
3. Perform a deterministic computation on the inputs.
4. Aggregate the results.

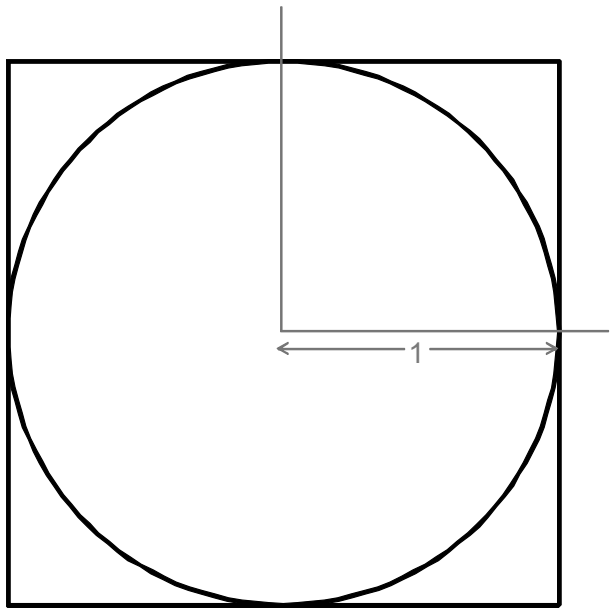

Fig. 1. A circle in a square, used to count the number of darts for calculating π

Consider a circle in a square of Figure 1. The area of the square is definitely 4, and the area of the circle is π. From this fact, it is possible to calculate the value of π by "integration by darts". If we throw darts, roughly uniformly in the unit square of Figure 1, some darts will be in the circle, and the others will not. So the darts will be classified into two groups, and the ratio of the darts in the circle to the darts in the square can be measured. By the geometric probability this ratio should converge into the value of π/4 as the number of sample gets bigger, and from this the value of π can be estimated. Of course, by symmetry, the same process can be done with only a quadrant of Figure 1.

Here,

(1) the domain of inputs is the square which circumscribes the circle. (2, 3) And the random inputs are generated and computed by throwing darts many times, (4) and the results are aggregated for achieving the goal.

To get an accurate approximation, MC simulation should have two properties from its statistical nature. First, the inputs should be random enough. If some darts are thrown onto the center of the circle intentionally, the darts will not be distributed uniformly, and then the estimation of the result would be not good. Second, the number of inputs should be large enough. The estimation gets more accurate as the number of darts increases, and if the darts are thrown for just a few times, the result would be unreliable.

This is the general procedure of the MC method, and is very beneficial when the direct calculation is difficult. However, for many situations, it has a potential problem. This will be discussed and solved in the following chapter.

References for section 1.2: Newman and Barkema, Ch. 1; Wikipedia: Monte Carlo method

2. Theoretical Backgrounds

2.1 Two kinds of sampling strategies of the MC method

2.1.1 Simple Sampling

The pattern 1-4 of chapter 1.2 shows the common procedure of the MC simulation. However, in the problem of estimating π, the darts are just "simply" sampled. This means that when each dart is thrown, it has always equal probability on any part of the square, so the sampling is purely random. This strategy is called "simple sampling" and it does not matter at all in the previous π problem, but in many other problems, it does. So it is not typically used, but sometimes by economists.

2.1.2 Importance sampling

In most cases of MC simulation, sampling is done by "importance sampling".

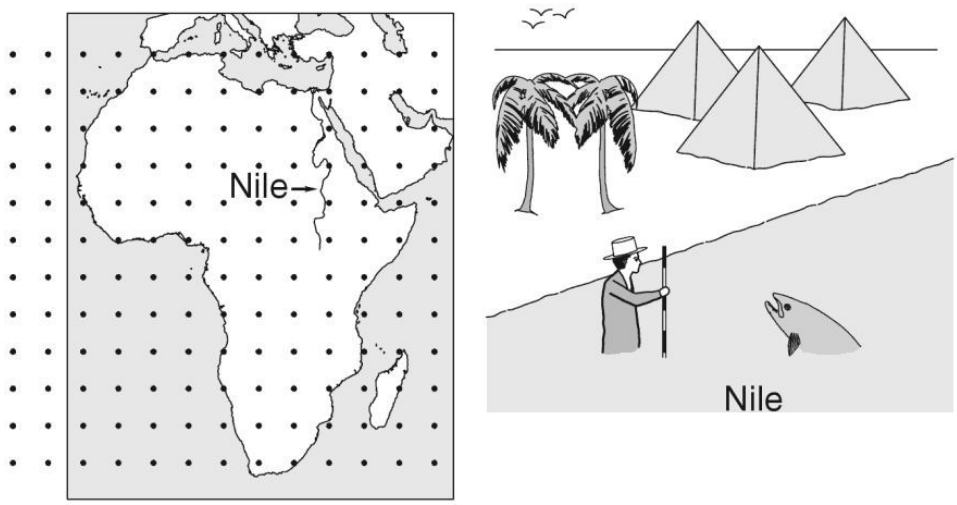

Fig.2 Measuring the depth of the Nile: a comparison of simple sampling (left) with the importance sampling (right) [D. Frenkel and B. Smith. Understanding Molecular Simulation. Academic press]

Figure 2 shows the two methods of averaging the depth of the Nile. In both cases the cartographer measures the depth with a stick. The left one shows simple sampling: in this case the cartographer walks around the whole of Africa to measure the depth, so most of the times he is not in the Nile, and gets a lot of zeros. For sure, it is not the good way of measuring. The smarter way of doing it is: find the Nile first. This is putting a bias on sampling. And then once in the Nile, just stay in the Nile and keep on measuring. By doing like this, the cartographer still can have some amount of randomness to walk around in the Nile, so he can get the samples to average, but he is biased so

he keeps on staying in the Nile. This is what the right figure of Fig.2 shows, and is the main idea of importance sampling.

The typical way of doing importance sampling in MC simulation is the Metropolis sampling. This will be discussed after we see the relationship between the thermodynamics and the MC method.

References for section 2.1: Frenkel and Smit, Ch. 2 and 3

2.2 Meeting with thermodynamics and statistical mechanics

2.2.1 Thermodynamics, statistical mechanics and the atomistic simulation

Thermodynamics describes the effects of transfer of heat and of work done on, or by, the material bodies or the radiation. So it interrelates macroscopic variables, such as temperature, volume and pressure, which describe the physical properties of thermodynamic systems – material bodies and radiation. And these macroscopic physical properties are actually, a consequence of the microscopic state of the system. In other words, the state of microscopic elements, such as atoms, molecules, electrons, or some other particles that consist of the system, determines the macroscopic state of the system. So, if we observe the change of the microscopic state of the system,

we will be able to predict macroscopic changes also. Then how can we go from microscopic description to macroscopic behavior? This question is answered by statistical mechanics. Statistical mechanics describes the macroscopic state (macrostate) as an average of a group of several microscopic states (microstates). So it can turn the thermodynamics problem into a mechanics problem of many bodies.

Meanwhile, when simulation focuses on the consisting particles, such as atoms, to describe the macroscopic behaviors, it is typically called as the "atomistic simulation". One of the frequently used atomistic simulation methods, the molecular dynamics (MD), describes the change of the thermodynamic system as the change of positions and velocities of particles (in classical MD), or the change of the wavefunction of many particles (in quantum MD), along the time line. So in MD, macroscopic properties such as energy E and volume V can be calculated by statistical mechanics, as averages over a dynamic trajectory along time τ from the simulation:

$$E = \frac{1}{t}\int_0^t E(\tau)d\tau \tag{1}$$

$$V = \frac{1}{t}\int_0^t V(\tau)d\tau \tag{2}$$

But this average only includes phenomena which occur over the time scale of the MD simulation. Thus, to make properties averaged for all possible

phenomena, MD simulation should be done for a very long time. If we only need this long time average, not the evolution of the system along the time line, and if some excitations of the system are beyond the time scale of MD, then it will be more efficient to sample microscopic states that are statistically significant for the long time averages, from the space of states. And this can be done by the MC simulation.

2.2.2 Conjugate pairs, ensembles, and ergodicity

In statistical mechanics, each microscopic state has a certain probability to emerge. Thus, if enough microscopic states are sampled, they will be sampled according to their probabilities, so they will show some probability distribution. For sure, the probability distribution for the microscopic system and its Hamiltonian are related to the macroscopic free energy function. Here, variables appear in pairs, that we call the "conjugate (variable) pairs". They are the couples that appear together in the first Law of thermodynamics:

$$dU = TdS + (-pdV) + \mu dN + \cdots \quad (3)$$

where U is the energy (internal energy) of the system, T is the temperature, S is the entropy, p is the pressure, V is the volume, μ is the chemical potential, and N is the number of particles.

The first Law of thermodynamics indicates the net energy flow of the system, so it shows the change of the internal energy of the system from

many different contributions. The TdS term is the heat flow, the $-pdV$ term is the mechanical or volume work, μdN term is the chemical work, and so on. And the conjugate pairs are $(T,S),(-p,V),(\mu,N),\cdots$. Each of them consists of one extensive variable and one intensive variable. Here, S,V,N are extensive variables, which are scaled with system size, and T,p,μ are intensive variables, which are not. For statistical sampling to work, one conjugate of each pair must be specified. In other words, either T or S, p or V, and μ or N should be fixed. This is because these fixed variables in the macroscopic condition translate as boundary constraints for sampling.

It is sometimes easier to use the entropy formulation of the first Law. This is simply obtained by rearranging the energy formulation:

$$dS = \frac{1}{T}dU + \frac{p}{T}dV - \frac{\mu}{T}dN + \cdots \tag{4}$$

In this formulation, the conjugate pairs are $(1/T,U),(p/T,V),(-\mu/T,N),\cdots$.

These boundary constraints determine the "ensemble". The ensemble is the collection of all possible microscopic states in which the system can be, given the thermodynamic (macroscopic) boundary conditions. This ensemble theory describes a system at equilibrium by probability distribution. There are three main ensembles in physics:

the micro canonical ensemble, E(E,V,N);

the canonical ensemble, $\mathrm{E}(\mathrm{T}, V, N)$; and

the grand canonical ensemble, $\mathrm{E}(T, V, \mu)$

(Here, the variables in parentheses are fixed.)

The micro canonical ensemble has the energy, volume, and number of particles fixed. For example, the Newtonian dynamics system in a closed box, which is the basic set of molecular dynamics, meets this condition. The canonical ensemble releases the constraint of fixed energy, but has volume, number of particles and temperature fixed. So it allows the control of the average of energy with temperature. In this case, the Newtonian system in a box with non-elastic walls, equilibrated at temperature T, meets the condition. The grand canonical ensemble releases even the constraint of fixed number of particles, so it should take into account the field that sets the average number of particles: fixed chemical potential. It can deal with open systems, such as the surface of a material.

Of course, other ensembles can be made by fixing different variables. For example, by fixing N, P, T, we can make an ensemble which is called the "isothermal-isobaric ensemble".

Then how can we average over the ensemble? It should be done with correct weights: probabilities that a system is in particular microstates. And the probability to find the system in some energy level ν shows the Boltzmann distribution, and is given by

$$P_\nu = \frac{\exp(-\beta H_\nu)}{\sum_{\nu \in e} \exp(-\beta H_\nu)} \tag{5}$$

where H_ν indicates the Hamiltonian of state ν, and β is $1/k_B T$. And the denominator of P_ν is the sum over all states. It is called as the "partition function" of the distribution:

$$Q = \sum_{\nu \in e} \exp(-\beta H_\nu) \tag{6}$$

The system does not have a unique energy mean, because it is probabilistic, so thermodynamic quantities should be defined as expectation values. But in the thermodynamic limit of infinite system size, the relative fluctuations in these averages will go to zero.

Also the free energy of the system can be obtained from Q:

$$F = -\frac{1}{\beta} \ln(Q) \tag{7}$$

If thermodynamic boundary conditions are decided, then the Hamiltonian in P_ν and Q, which should include all the things that can fluctuate in the system, can be derived from the relevant Legendre transform of the entropy.

This Legendre transform shifts a function with one of its parameters as an independent variable, to a new function dependent on a new variable. This new function is the partial derivative of the original function with respect to the independent variable. And it becomes the difference between the original function and the product of the old and new variables.

From the energy formulation of (3), the internal energy U depends on S, V, and N (=$\{N_i\}$). And by Legendre transforms, the Helmholtz free energy and the Gibbs free energy are obtained, and their independent variables change:

$$F(T,V,N) = U - TS \tag{8}$$

$$G(T,p,N) = U - TS + pV \tag{9}$$

And from the entropy formulation of (4), the entropy S depends on U, V, and N. By Legendre transforms, several free entropies are obtained:

$$A\left(\frac{1}{T},V,N\right) = S - \frac{1}{T}U = -\frac{F}{T} \tag{10}$$

$$K\left(\frac{1}{T},\frac{p}{T},N\right) = S - \frac{p}{T}V = -\frac{G}{T} \tag{11}$$

$$L\left(\frac{1}{T},V,\frac{\mu}{T}\right) = S - \frac{1}{T}U - \frac{\mu}{T}N \tag{12}$$

(10) is called the Helmholtz free entropy or the Massieu potential, and (11) is called the Gibbs free entropy or the Planck potential. From these free entropies, we can get the relevant Hamiltonians for sampling. For example, the canonical ensemble is a function of T,V,N, so the Helmholtz free entropy fits. And the Hamiltonian of the canonical ensemble is obtained from the Legendre transform from S:

$$-\frac{1}{T}U = -\frac{1}{T}\langle E\rangle \qquad \rightarrow \qquad H_\nu = E_\nu \tag{13}$$

where E_ν is the energy of eigenstate ν. So the probability and the partition function become:

$$P_\nu = \frac{\exp(-\beta E_\nu)}{Q}, \qquad Q = \sum_{\nu \in e} \exp(-\beta E_\nu) \tag{14}$$

Meanwhile, the grand canonical ensemble is a function of T, V, μ, so the free entropy of (12) fits. And the Hamiltonian of grand canonical ensemble is obtained from the Legendre transform from *S*:

$$-\frac{1}{T}U + \frac{\mu}{T}N = -\frac{1}{T}(\langle E \rangle - \mu N) \qquad \rightarrow \qquad H_\nu = E_\nu - \mu N \tag{15}$$

So the probability and the partition function become:

$$P_\nu = \frac{\exp\big(-\beta(E_\nu - \mu N)\big)}{Q}, \qquad Q = \sum_{\nu \in e} \exp\big(-\beta(E_\nu - \mu N)\big) \tag{16}$$

And thermodynamic quantities are averaged over the ensemble. For example, in the canonical ensemble, the average internal energy of the system is the expectation value of the energy:

$$U \equiv \langle E \rangle = \sum_{\nu \in e} P_\nu E_\nu = \frac{\exp(-\beta E_\nu)}{\sum_{\nu \in e} \exp(-\beta E_\nu)} E_\nu = -\frac{\partial \ln(Q)}{\partial \beta} \tag{17}$$

In summary, constant thermodynamic variables become macroscopic boundary constraints, and they decide the relevant ensemble. And this ensemble decides the probability with which the sampling is done over the ensemble. Thus we can get the average value of thermodynamic quantities.

However, one more concept is needed: the “ergodicity hypothesis” of thermodynamics. It describes a dynamical system which has the same behavior averaged over time as averaged over phase space. In other words, if

a certain system is “ergodic”, its microstates are equiprobable over a long period of time. And this means the system will reach all of its states if we wait long enough. Because we know that the system will reach all its states, we can just sum over the probability of each state, rather than doing the full dynamics along the time line. Statistical mechanics can only be applied on ergodic sytems.

References for section 2.2: Frenkel and Smit, Ch. 2-5; Allen and Tildesley, Ch. 2

2.3 Importance sampling of statistical mechanics

As we discussed in Chapter 2.1, sampling can be done simply or by importance. The problem of simple sampling arises also in the sampling of physical systems: we would sample states with high entropy by simple sampling. For example, the Ising model, that describes magnetic behaviors with lattice of spins:

$$H = -\sum_{i,j} J\sigma_i \sigma_j \tag{18}$$

(The external magnetic field term is omitted)

has the lowest energy when the model is in ferromagnetic state, which indicates when the spins are all up, or all down. And the average of energy lies between the lowest energy and the highest energy.

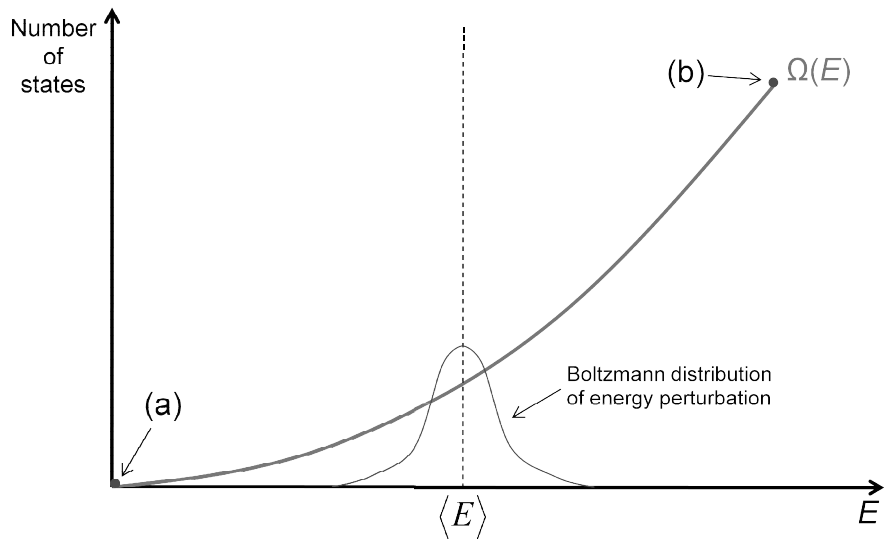

Fig. 3. Number of states of the Ising model and the Boltzmann distribution

As we can see in the Figure 3, when we simply sample this model, states with high energy such as states near (b), the paramagnetic state with random spins, would occur more frequently, proportional to the number of states $\Omega(E)$. This is because they are more probable when we just randomly capture states. On the other hand, states near (a), the ferromagnetic state, would occur seldom. This is clear from

$$\frac{S(E)}{k_{\mathrm{B}}} = \ln \Omega(E) \tag{19}$$

and,

$$\frac{d \ln \Omega(E)}{dE} = \frac{1}{k_{\mathrm{B}} T} > 0 \tag{20}$$

which means that the number of states with a given energy is an increasing function of an energy. So, that means with simple sampling, states near (a)

would almost never picked since there are almost no states.

But to have correct average, states of both cases are equally needed. Even if states near (b) occur many times in the simulation, they will have little meaning according to the Boltzmann distribution. Furthermore, when we want to study the low temperature phenomena, where the model longs to be ferromagnetic, with all spins aligned, we would almost never end up with ferromagnetic configuration by simple random picking. So a lot of states will be sampled in the phase space but the relevant one will never be sampled.

Then how about picking states with a biased probability, to sample relevant ones? This is done by the importance sampling, as shown in Figure 4. The most frequently used importance sampling method in physical material simulation is the Metropolis algorithm. It walks through phase space visiting each state from the ensemble with a probability proportional to $\exp(-\beta E)$, rather than picking states randomly and later weighing them by a probability of Boltzmann distribution.

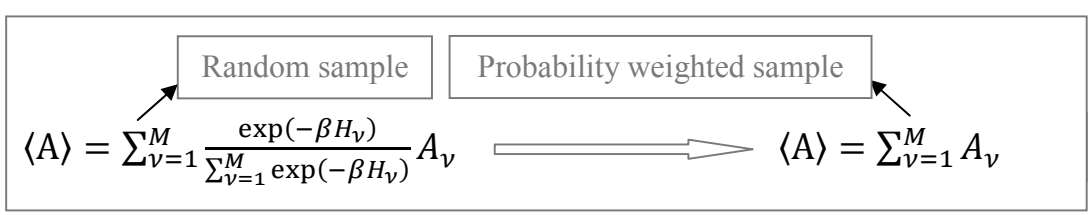

Fig. 4. Difference between simple sampling and importance sampling of statistical mechanics

2.3.1 Metropolis algorithm

MC simulation with Metropolis algorithm [Metropolis et al., 1953] starts with a random starting state, say *i*. And pick trial state, say *j*, from *i* with some rate $W^{o}_{i\to j}$, then accept *j* with some probability $P_{i\to j}$. It repeats the "pick-and-accept(or not)" process until some criterion is satisfied. Metropolis algorithm has some criteria for picking and accepting, and also the criteria for appropriate probability distribution.

There are two conditions for generating proper probability distribution. The first one is "equal *a priori* probabilities":

$$W^{o}_{i\to j} = W^{o}_{j\to i} \tag{21}$$

where $W^{o}_{i\to j}$ is the pick rate at which the system choose new states from the state *i*. We first have to design mechanism for picking potential moves. And the second one is the "detailed balance":

$$P_i P_{i\to j} = P_j P_{j\to i} \tag{22}$$

In the detailed balance condition, P_i and P_j mean the probability that the system will be in the state *i* and *j*, respectively, And $P_{i\to j}$ and $P_{j\to i}$ mean the rate at which the system transfer from the state *i* into the state *j*, and from *j* into *i*, respectively. The LHS of the detailed balance condition is the number of times the system goes out of *i* into *j*, because it is the multiplication of the probability that the system is in *i* with the rate at which

the system transfer to j. So the RHS is the number of times the system goes out of j into i. And when these two are equal, net flow of probability density would be zero, so we can have a steady state distribution. In other words, if this detailed balance holds for all pairs of i and j, the resulting probability distribution will not be changing anymore, if we do a lot of samplings. This When $W_{i\to j}^{\mathrm{o}}$ and $P_{i\to j}$ satisfy the above two criteria, the Metropolis algorithm will produce an equilibrium distribution.

There are also two conditions to accept a new state and transfer to it. The first one is:

$$\mathrm{P}_{i\to j} = 1 \text{ when } E_j < E_i \tag{23}$$

This means the transfer to the lower energy state is always accepted.

And the second one is,

$$\mathrm{P}_{i\to j} = \exp\big(-\beta(E_j - E_i)\big) \text{ when } E_j > E_i \tag{24}$$

(19) is easily derived from the detailed balance condition. In (17), $P_{j\to i}$ is 1 by (18). And $P_i = \exp(-\beta E_i)/Q$, $P_j = \exp\big(-\beta E_j\big)/Q$ from the Boltzmann distribution. By rearranging, $\mathrm{P}_{i\to j}$ becomes $\exp\Big(-\beta\big(E_j - E_i\big)\Big)$.

By putting all the things discussed above together, we can get the procedure of Metropolis MC simulation:

1. Start with some random configuration.
2. Choose perturbation (trial state) of the system.

3. Compute energy for that perturbation.
4. If $\Delta E < 0$, accept perturbation.

   If $\Delta E > 0$, accept perturbation with probability $\exp\left(\frac{-\Delta E}{k_B T}\right)$, the Boltzmann factor.
5. Go back to 2

For one's information, the name "Metropolis" came from the person: Nicholas Metropolis, who made the algorithm and was one of the main members of the Manhattan project.

2.3.2 The reweighting method: Umbrella sampling, multi canonical algorithm and Wang-Landau algorithm

The importance sampling of the MC method is also called as Boltzmann sampling (BS), since microstates are sampled with Boltzmann weight. This works well for many systems, however, a lot of other approaches have been made, since the resolution limit of MC simulations near phase transitions needs many runs of the simulation to precisely characterize peaks in response functions such as the specific heat. Great improvements have become available when it was discovered that entire distribution of properties, not just mean values, can be useful: for predicting the behavior of the system at a temperature other than that at which the simulation was

performed. This is called “reweighting”, and may be done after a simulation is complete or during the simulation as one of its integral process. By reweighting method, so-called non-Boltzmann sampling (NBS), which samples systems with a non-Boltzmann factor, is possible.

The fundamental basis for the reweighting method is to realize that the properties of the systems will be determined by a distribution function in an appropriate ensemble. For example, in the canonical ensemble, the probability to observe a particular state in the Ising ferromagnet with interaction constant J at temperature T, is proportional to the Boltzmann weight $\exp(-KE)$ where $K = J/k_B T$, the dimensionless coupling. The probability to observe simultaneously the system with total (dimensionless) energy $E = -\sum \sigma_i \sigma_j$ and total magnetization $M = \sum \sigma_i$ is then:

$$P_K(E, M) = \frac{g(E, M)}{Q(K)} \exp(-KE) \tag{25}$$

where $g(E, M)$ is the density of states, which is the number of configurations, with energy E and magnetization M. Like this, the density of states contains all the information about the systems, and also the effect of temperature can be included straightforwardly. The reweighting method is also useful when sampling a part of phase space relevant for a particular property, or when sampling phase space with a lot of local minima more efficiently because NBS can make it much easier to get out of local minima.

In addition, free energy differences throughout the covered range of states can be easily obtained. This is useful for thermodynamics studies, since these studies often require the knowledge of free energy changes. [Landau and Binder, 2009 & Abreu and Escobedo, 2006]

The umbrella sampling is one of the first reweighting methods. Bennet (1976) tried to estimate free energy difference ΔF between two systems, labeled A and B, with partition functions Q_{A} and Q_{B}, by the method called "overlapping distribution method", which is often used to calculate free energy by the MC method. Torrie and Valleau [Torrie and Valleau, 1977] replaced the Boltzmann factor of the system by a non-negative weight function $\pi(\mathbf{r}^{\mathrm{N}})$ to modify the Markov chain which is constructed in the sampling in a way that one samples both the part of configuration space accessible to system A and the part accessible to system B. By this approach, the average of samples becomes:

$$\langle \mathrm{A} \rangle = \frac{\sum_{\nu=1}^{M} \exp\left(-\beta(H_\nu - H_\nu^o)\right) A_\nu}{\sum_{\nu=1}^{M} \exp\left(-\beta(H_\nu - H_\nu^o)\right)} \tag{26}$$

which is the result of sampling of $\Delta H = H - H^o$ with some Hamiltonian H^o. So this sampling is definitely non-Boltzmann.

After a while, Berg and Neuhaus [Berg and Neuhaus, 1991] suggested using a distribution which yields a flat histogram. This scheme is known as the "multi canonical (MUCA) algorithm", and uses a different choice for

sampling:

$$P_\nu = \frac{1}{g(E_\nu)} \tag{27}$$

where $g(E_\nu)$ is the density of states (DOS) of the system. This approach generates states equally distributed on energy, thus provides an overview of energy. However, on most systems, $g(E_\nu)$ is unknown. There have been many solutions to it, and Wang and Landau (2001) offered a way to obtain the DOS during the simulation. In this "Wang-Landau algorithm", the generation of states does not depend on the temperature term β, so the average values of every thermodynamic variable can be calculated for every temperature.

The framework of Wang-Landau algorithm is a random walk in energy space with a flat histogram. [Wang and Landau, 2001a, 2001b. & Landau et al., 2004] The classical partition function can be written either as a sum over all states or a sum over all energy levels, so the partition function of (6) can be written:

$$Q = \sum_{\nu \in \mathrm{e}} \exp(-\beta H_\nu) = \sum_{E} g(E) \exp(-\beta E) \tag{28}$$

And we begin Wang-Landau algorithm with simple guess for the DOS, usually $g(E) = 1$, and improve it following the procedure:

1. Set $g(E) = 1$ for all energy levels, and choose a modification factor f, e.g. $f_0 = e^1$
2. Start with some configuration
3. Set the histogram $\mathrm{h}(E) = 0$ for all energy levels
4. Choose perturbation(trial state) of the system
5. Compute energy for the perturbation
6. Calculate the ratio of the DOS:

$$\eta = \frac{g(E_1)}{g(E_2)}$$

   which results if the system perturbs, and where E_1 is the energy of present state, and E_2 is the energy of trial state
7. If $\eta \geq 1$, accept perturbation

   If $\eta < 1$, accept the perturbation with probability η
8. Set $g(E_{\text{present}}) \rightarrow g(E_{\text{present}}) * f$, $\mathrm{h}(E_{\text{present}}) \rightarrow \mathrm{h}(E_{\text{present}}) + 1$
9. If the histogram is not 'flat', go to 3

   If the histogram is 'flat', decrease f, e.g. $f_{n+1} = \sqrt{f_n}$, and go to 3
10. Repeat steps 3-9 until $f = f_{\text{final}} \sim \exp(10^{-8}) \cong 1.000\ 000\ 01$
11. Calculate properties using the final resultant DOS $g(E)$

As we can see in the step 5-6, the probability of accepting perturbation is:

$$\mathrm{P}(E_1 \to E_2) = \min\left(\frac{g(E_1)}{g(E_2)}, 1\right) \tag{29}$$

This is from the detailed balance of the Wang-Landau algorithm:

$$\frac{1}{g(E_1)}\mathrm{P}(E_1 \to E_2) = \frac{1}{g(E_2)}\mathrm{P}(E_2 \to E_1) \tag{30}$$

Where $1/g(E_1)$ is the probability at the energy level E_1 and $\mathrm{P}(E_1 \to E_2)$ is the transition probability from E_1 to E_2. Since the probabilities at all energy levels are reciprocal to the relevant DOS of them, if some energy levels are less visited than others, the probability to visit these energy levels is higher than frequently visited energy levels. Thus all energies can be almost equally visited, and this is checked by the histogram's flatness. However, a perfectly flat histogram cannot be achieved, so we should set some criterion. Typically this is when the minimum entry is ~ 80% of the mean value. In the early stages, the detailed balance is not satisfied, but as $f \to 1$, it is recovered.

2.3.3 A practical problem: random number generation

The MC method needs real random numbers, but "real random" is not possible in computers. So, we generate pseudo random numbers by computing and use them in MC simulation. A definition of good random number generator depends on the application, but there are some criteria:

periodicity, uniformity, consistency with the laws of statistics, and non-correlativity. We can test these by simple tests.

In the sampling of physical systems, random numbers are used to make the probabilistic decision. The criteria of all sampling methods accept trial perturbation with some probability, and this is done by making random numbers: when the generated random number is smaller than the value of probability, then we are in the probability, thus accept the perturbation. On the other hand, if the generated number is bigger than the probability, then we are outside of it, thus do not accept the perturbation.

References for section 2.3: Frenkel and Smit, Ch. 3 and 5; Newman and Barkema, Ch. 2; Landau and Binder, Ch 2, 4, and 7

2.4 Analyzing results

2.4.1 Obtaining results

What we get after the Metropolis MC simulation is a trajectory-like perturbation of some property along the increasing number of sampling, which is called, by convention, the “MC time”. This is not the physical time of the system, but the number of sampling by MC iteration, which tells the relative duration time of simulation. One MC iteration gets one sample of the system, so the MC time is also called as MC step or MC sweep (MCS).

The result perturbation is averaged over to get correct thermodynamic averages. Figure 5 shows a typical result of Metropolis MC simulation. Practically, people often cut off the first part (part (a) of Figure 5) and remove it, since the simulation start with a random state and may be far away from the average energy state. Thus, by cutting that off and sampling only part (b), which is the part after the system has relaxed towards the equilibrium, we can get much faster relaxation of the average.

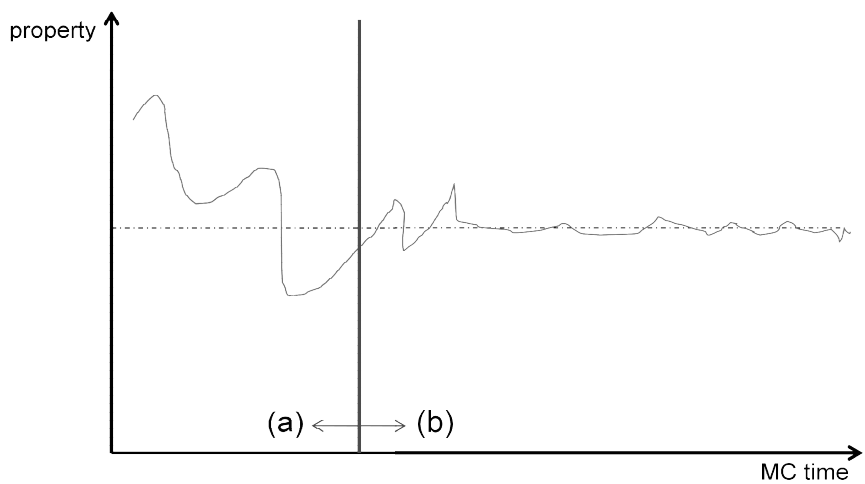

Fig. 5. The graph of a typical Metropolis MC simulation result as a function of MC time

The Wang-Landau MC simulation makes a relative density of states of the system. And this can be rescaled to the absolute one by the total number of possible states or the number of ground states.

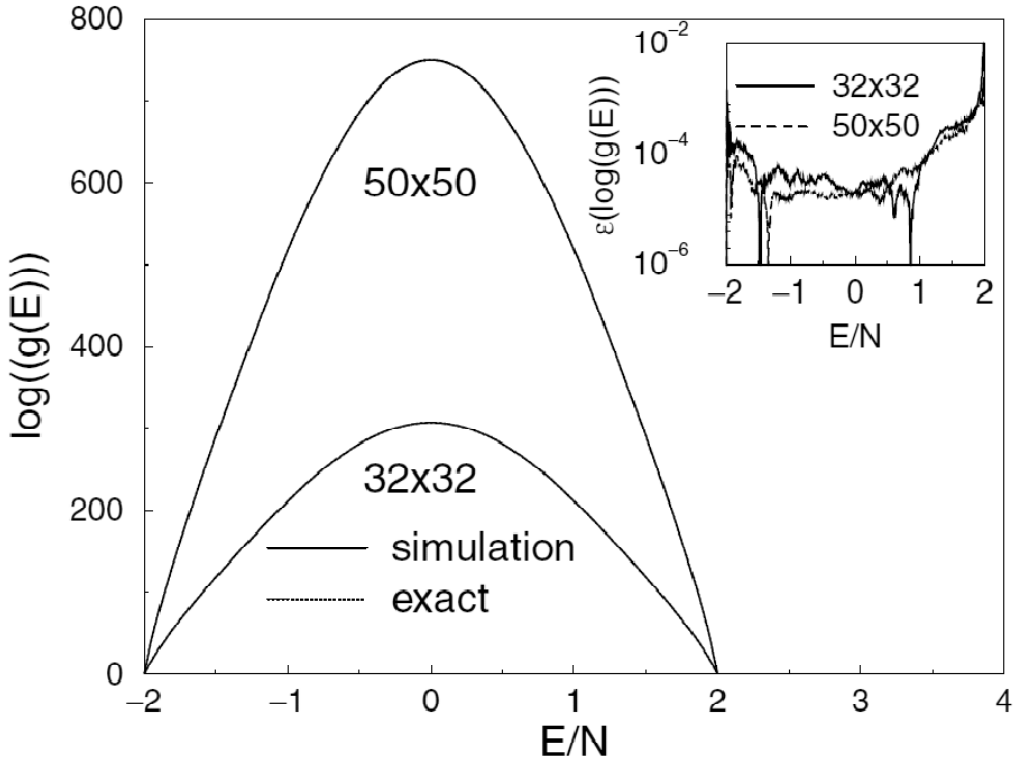

Fig. 6. The resultant DOS by Wang-Landau algorithm for 2D Ising model, with 32×32 and 50×50 lattices
[Wang and Landau, 2001]

2.4.2 Detecting phase transitions

We detect phase transitions by looking at physical properties, just like for a real system. The energy of the system is discontinuous at first order transitions, but not at second order transitions. The concentration is also discontinuous at first order transitions, when we work at constant chemical potential. Another thing to track is the specific heat of the system. Specific heat is infinite at first order transitions. Actually, specific heat is the fluctuation of the energy; the measure of how much the energy of the system

fluctuates around the average:

$$C = \frac{1}{N}\left(\frac{\partial U}{\partial T}\right) = \frac{1}{N}\left(\frac{\langle E^2\rangle - \langle E\rangle^2}{k_{\mathrm{B}}T^2}\right) \tag{31}$$

So, this property can be obtained from the energy distribution. In addition, it has log-like infinite singularity for second order transitions.

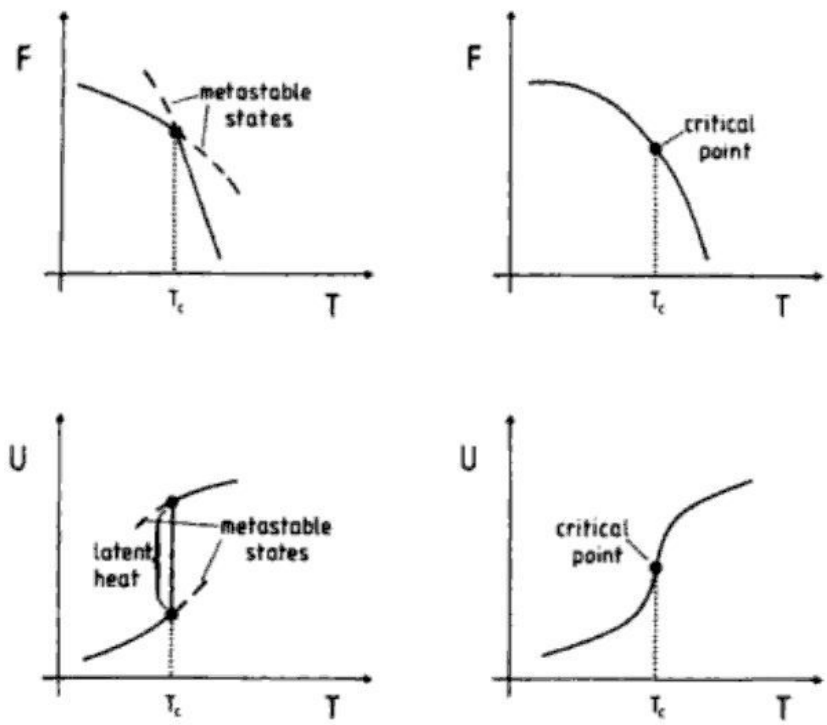

Fig. 7. Schematic view of the change of the free energy and the (internal) energy of the system undergoing a first order transition (left) and a second order transition (right) [Landau and Binder, 2009]

2.4.3 Evaluation of the accuracy

One of attractive aspects of the MC simulation is that it is a simulation

method with almost no approximations in its approach. So the MC simulation can give solutions as accurately as we want, or as we can afford. Then, how can we measure how good is the MC sampling to get properties of ensemble? To answer this question, we should know from where the accuracy comes.

Of course, the Hamiltonian of the model of the system should be correct and accurate. And once it is decided, the error comes from its finite samples, in other words its finite time, and from its finite size, in other words its periodic boundaries. Here, we discuss only for the error on quantities that are in the microscopic states thus able to average, like volume, energy, concentration, and so on, but not for ones that are not, like specific heat, free energy, entropy, and so on. The latter quantities can be evaluated indirectly from the former ones. And we introduce some terminologies from statistics:

$\bar{A}$: True average of some property A from the true distribution of the ensemble (or the population)

$\langle A \rangle$: Average collected in sampling

σ_{dist}: standard deviation A from the true distribution

σ_{av}: standard deviation of A from the sampled distribution

m: sample size

How different are $\langle A \rangle$ with $\bar{A}$? The property A fluctuates with some spread around the true average $\bar{A}$.

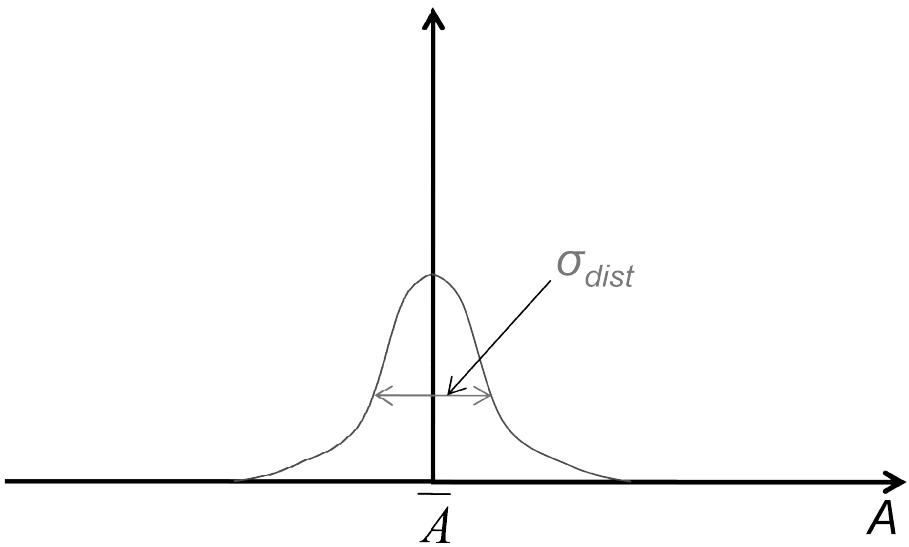

Fig. 8. The distribution of some property A around its true average.

Thus A shows some distribution and has standard deviation σ_{dist}. If A is the energy of the system, then σ_{dist} will be the specific heat. And if we take finite samples from this distribution of A, we will not get $\bar{A}$ but some other average, $\langle A \rangle$. If samples are uncorrelated:

$$\sigma_{av}^2 = \overline{(\langle A \rangle - \bar{A})^2} = \frac{\sigma_{dist}^2}{m} \approx \frac{\langle A^2 \rangle - \langle A \rangle^2}{m} \tag{32}$$

from the knowledge of statistics. So, if we know m, then we can estimate the error. And the simulation has all these quantities in it. This is true unless samples are correlated, in other words, if samples are really picked randomly from the ensemble.

However, samples are correlated. For instance, the states in the Markov

chain of a Metropolis MC simulation are not independent. State is obtained from previous state through some procedure of perturbation. When the system tries to perturb with very high energy in a Markov chain of the MC simulation, the perturbation will almost never be accepted. Thus, the system has a high degree of correlation, and the MC simulation would not converge as well as the formula of random sampling indicates. In other words, the quality of a sample gets diluted by what is called the "correlation time" of A:

$$\tau_A = \int_0^\infty C_A(\tau)\, d\tau \tag{33}$$

where the correlation function $C_A(\tau)$ of A is:

$$C_A(\tau) = \lim_{t\to\infty} \frac{1}{t} \int_0^t \frac{(A(t) - \langle A\rangle)(A(\mathrm{t}+\tau) - \langle A\rangle)}{\langle A^2\rangle - \langle A\rangle^2}\, dt \tag{34}$$

And the quality of average becomes:

$$\sigma_{av}^2 = \frac{\langle A^2\rangle - \langle A\rangle^2}{m}(1 + 2\tau_A) \tag{35}$$

And again, these all can be calculated from the simulation. Thus, we can evaluate the quality of sampling by these equations.

References for section 2.4: Newman and Barkema, Ch. 2; Binder and Heermann, Ch. 3; Landau and Binder, Ch 2 and 7

3. Simulation

3.1 Verification of the Wang-Landau MC code

We programmed our own Wang-Landau MC code with C++ programming language, with the random number generator made by Agner Fog. It is distributed freely under the terms of the GNU General Public License as published by the Free Software Foundation. Also, the EXTNUM library of Southwest Biotechnology and Informatics Center of New Mexico State University was used for dealing with big numbers, which are out of the range of C++ built-in real number. This library extends the exponent to have range from $10^{-646456993}$ to $10^{646456992}$.

In order to verify that the code works, we tested it with some simple models. Once the density of states of the system is obtained, the partition function of the system is obtained by

$$Q = \sum_{E} g(E)\exp(-E/k_B T) \tag{36}$$

and from this partition function, the Helmholtz free energy is obtained by

$$F(T) = -k_{\mathrm{B}} T \ln Q \tag{37}$$

The internal energy of the system is the average, or the expectation value of the energy, thus

$$U(T) \equiv \langle E \rangle_T = \frac{\sum_E g(E)\exp(-E/k_B T)}{Q} \tag{38}$$

and the specific heat is the fluctuation of the energy, so

$$C(T) \equiv \frac{\partial U(T)}{\partial T} = \frac{\langle E^2 \rangle_T - \langle E \rangle_T^2}{k_B T^2} \tag{39}$$

And finally, from the definition of the Helmholtz free energy, the entropy of the system is obtained by

$$S(T) = \frac{U(T) - F(T)}{T} \tag{40}$$

3.1.1 2D Ising model

As discussed briefly in chapter 2.3, the Ising model shows the behavior of the ferromagnet with lattices of spins which are up or down. Its Hamiltonian is

$$H = -\sum_{i,j} J_{ij} \sigma_i \sigma_j - \sum_j h_j \sigma_j \tag{41}$$

The first term of RHS is the interaction term, and the second term is the external potential (magnetic field) term. In the verification, the interaction coefficient J_{ij} is same as 1 for all pairs of interactions, and the second term was omitted for simplicity:

$$H = -J \sum_{i,j} \sigma_i \sigma_j \ , \qquad J = 1 \tag{42}$$

The results are discussed in the Chapter 4.

3.1.2 2D Potts model

The Potts model is an upgraded version of the Ising model, and here the spins can have many states. Its Hamiltonian is

$$H_p = -\sum_{i,j} J_{ij}\delta(\sigma_i, \sigma_j) - \sum_j h_j \sigma_j \qquad (43)$$

where $\delta(\sigma_i, \sigma_j)$ is the Kronecker delta, which equals 1 whenever $\sigma_i = \sigma_j$, and 0 otherwise. And σ_i, σ_j can have q different states: $1,2,\dots,q$. Here, the interaction coefficient is 1 for all pairs of interactions, and the external field term was omitted for simplicity.

$$H_p = -J_p \sum_{i,j} \delta(\sigma_i, \sigma_j)\,, \qquad J_p = 1 \qquad (44)$$

The results for $q = 10$ case are discussed in the Chapter 4.

References for section 3.1: Wang and Landau, 2001a and 2001b

3.2 κ-carbide as a realistic sample

Carbide is a compound of carbon and some less electronegative elements. And κ-carbide is a kind of carbide, which can be found in high aluminum-high manganese steel products. These steel products have been expected to be used widely, since they can have high corrosion resistance, cheap price, and especially, light weight. [Frommeyer and Brüx, 2006] And in these products, the κ-carbide acts as a crucial phase.

The formula of κ-carbide is $(Fe,Mn)_3AlC$; aluminum atoms are located on each corner of the cubic unit cell, iron and manganese are placed on the face centers, and carbon is at octahedral or tetrahedral site. This type of structure is called $E2_1$ or the "anti-perovskite-type" structure. Figure 9 shows the structure of Fe_2MnAlC κ-carbide.

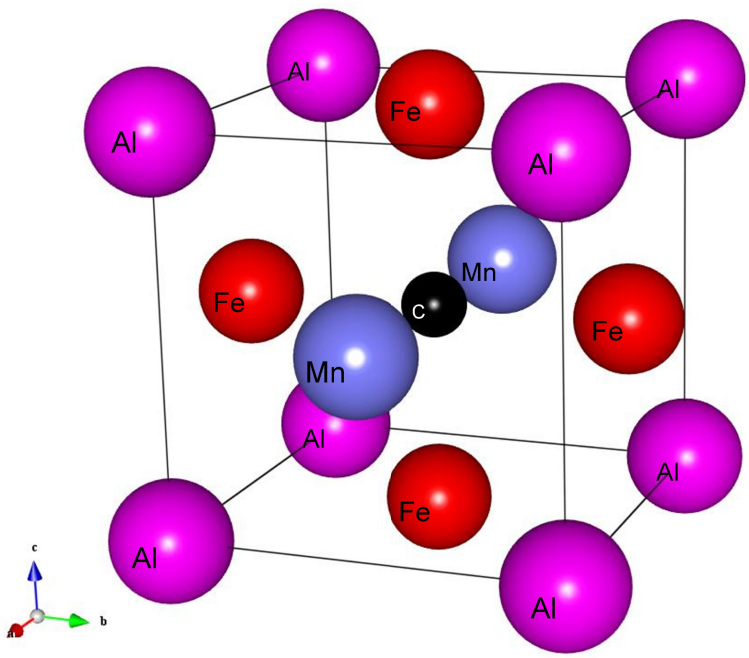

Fig. 9. The structure of Fe_2MnAlC κ-carbide

While the perovskite structure, such as $BaTiO_3$, has trasition metal at the center of the unit cell and non-metallic atom at the face center, the anti-perovskite structure has transition metal at the face center and non-metallic atom at the center of the unit cell.

Carbide is harder than pure iron, thus strengthens the steel. The experiment on fine κ-carbide (Frommeyer and Brüx, 2006) showed that the nano-size κ-carbide regularly distributed and coherent with austenite matrix sustains homogeneous shear band acquired by dislocation glide in the high-strength Fe-Mn-Al-C light-weight TRIPLEX steels. Thus, the ductility is highly improved keeping the high specific energy absorption of the TWIP steel. And also, Kimura et al. (2004) reported that the lamellar structure of ferrite and κ-carbide make steel more ductile and crack-resistant.

However, carbide also can be the initial point of crack in the steel. The effect of brittle particles in a ductile matrix on fracture was studied [Wallin et al., 1986]:

$$P_{fr} = 1 - \exp\left\{-\left(\frac{d}{\bar{d}}\right)^3 \left(\frac{\bar{d}}{d_N}\right)^3 \left(\frac{\sigma - \sigma_{min}}{\sigma_0 - \sigma_{min}}\right)^m\right\} \tag{45}$$

where P_{fr} is the probability of fracture, d is the size of the brittle particle, $\bar{d}$ is the average size of particles, σ is the tensile stress acting on a particle, σ_{min} is the minimum fracture stress of the particle, m is Weibull inhomogeneity factor, and σ_0 and d_N are normalizing parameters. According to (45), the probability of fracture increases when the size of the brittle particle increases, or when brittle particles gather to be as one big particle. Brittle κ-carbide particles gather as band-type structures on phase boundaries, and they can be initial points of cracks. [Han et al., 2010]

3.3 Simulation details

3.3.1 A brief introduction to the density functional theory

The density functional theory (DFT) or first-principles calculations, is a quantum mechanical way of modeling to investigate the electronic structure of many-body systems, and it calculates the properties of a many-electron system using functionals of the electron density. It has become one of the most popular computational methods available in condensed matter physics, chemistry and material science. The energy and the formation enthalpy, and some other properties of the materials are obtained by calculating the eigenvalue of the equation of the wavefunction, which is so called the 'Kohn-Sham equation' [Kohn and Sham, 1965]:

$$\left\{-\frac{1}{2}\nabla^2 + v_s(\mathbf{r})\right\}\psi_i(\mathbf{r}) = \epsilon_i\psi_i(\mathbf{r}) \qquad (46)$$

We obtained the energy of 6 kinds of κ-carbide unit cell of 3.3.2 by DFT [Seo, 2011], with all electron full potential linearized augmented planewave (FLAPW) method [Wimmer et al., 1981, Weinert et al., 1982], and the generalized gradient approximation (GGA). [Perdew et al., 1996]

3.3.2 Modeling Fe_2MnAlC κ-carbide: A cell gas model

Fe_2MnAlC κ-carbide has 3 kinds of octahedral sites for carbon atom, as shown in the Figure 10.

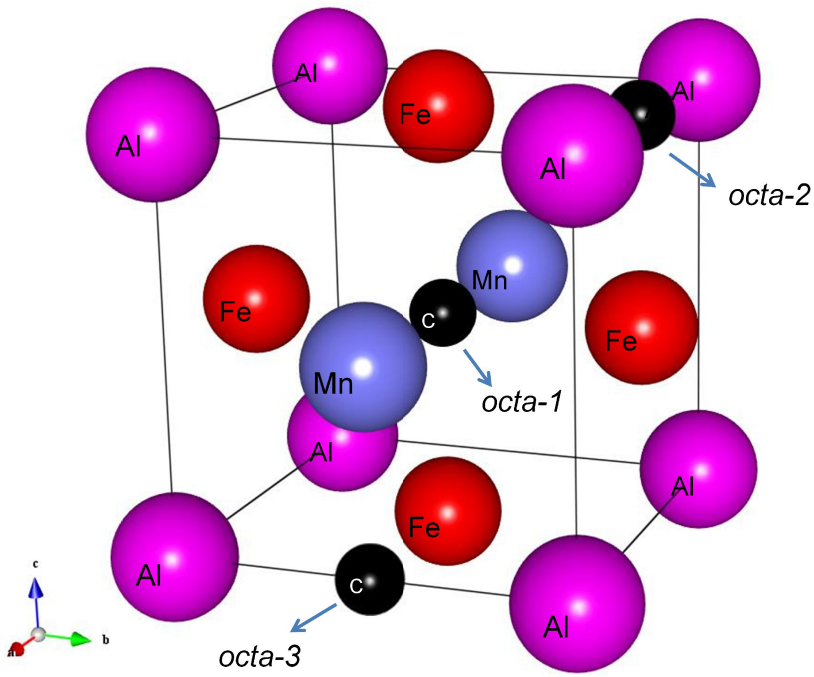

Fig. 10. Octa-1, Octa-2, and Octa-3 position of carbon in Fe_2MnAlC κ-carbide unit cell.

We can think of 6 kinds of Fe_2MnAlC κ-carbide unit cell: carbon can be either in the octa-1, or the octa-2, or the octa-3 site, and the unit cell can be either ferromagnetic or paramagnetic. The total energy state was calculated for the 6 cells respectively, by the calculation of DFT [Seo, 2011]. Based on that result, we modeled a bigger structure of Fe_2MnAlC κ-carbide, which is a cubic of 10×10×10 Fe_2MnAlC κ-carbide unit cells. Each unit cell may have one of 6 energy states, according to 6 kinds of Fe_2MnAlC κ-carbide unit cell, and it may also change to another energy state among 6 kinds. We set the

initial energy states of unit cells of the model randomly, then changed and sampled the state of each unit cell with Monte Carlo method, according to the Wang-Landau algorithm. In this model, for its simplicity, all cells have no interactions between them, thus we call this model as a "cell gas model". This is a very simple model:

$$H = \sum_i \varepsilon(\sigma_i) \tag{47}$$

which is not near from the real Fe_2MnAlC κ-carbide. However, it can be used as a starting point of studying the thermodynamics of the real κ-carbide system. Since all unit cells of the system are one of 6 kinds of Fe_2MnAlC κ-carbide, thus all of them have one carbon atom. So, the system is carbon-saturated.

4. Results

4.1 Verification results

4.1.1 2D Ising model

Figure 11-14 show the verification simulation results of 2D Ising model, and compare them with the work of Wang and Landau (2001). Here, *L* means the size of one side of the lattice.

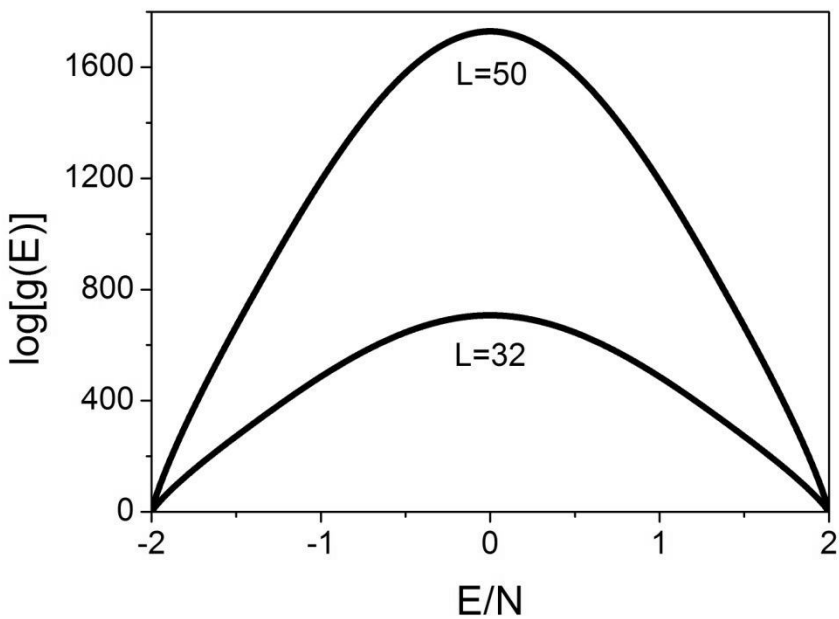

Fig. 11. Density of states (DOS) from this work

Our simulation code resulted values of density of states (DOS) of 32×32 and 50×50 2D Ising model. They are well matched.

In the Figure 12, the resultant DOS simulated with the tolerance of 80% and 95%, are shown respectively. They show almost no difference.

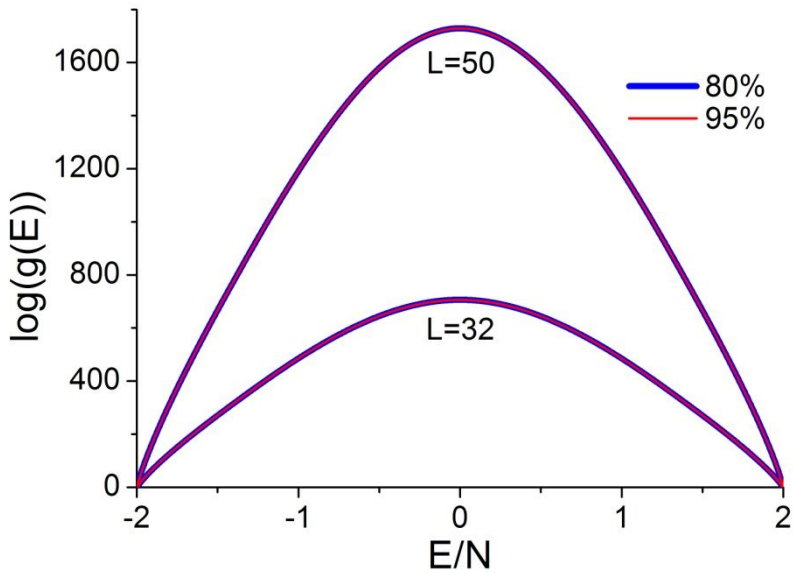

Fig. 12. DOS simulated with tolerance of 80% and 95%

Figure 13 and 14 compares the free energy and the specific heat of 32×32 and 50×50 lattices from our simulation, respectively, with the result of 256×256 lattice of Landau and Wang. The temperature is defined in the unit of J/k_{B} with $J = 1$, where J is the interaction parameter and k_{B} is the Boltzmann constant. Here, thermodynamic properties are calculated and divided by $N = L \times L$, the number of unit cells in the model.

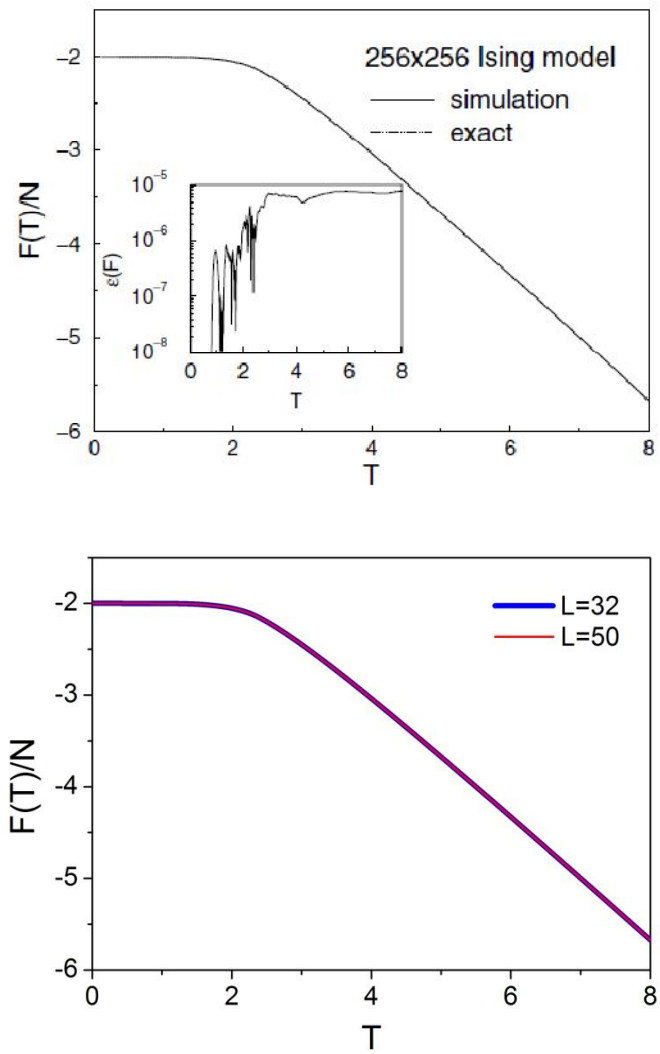

Fig. 13. (a) Free energy curve from the work of Wang and Landau, (b) free energy curve from this work

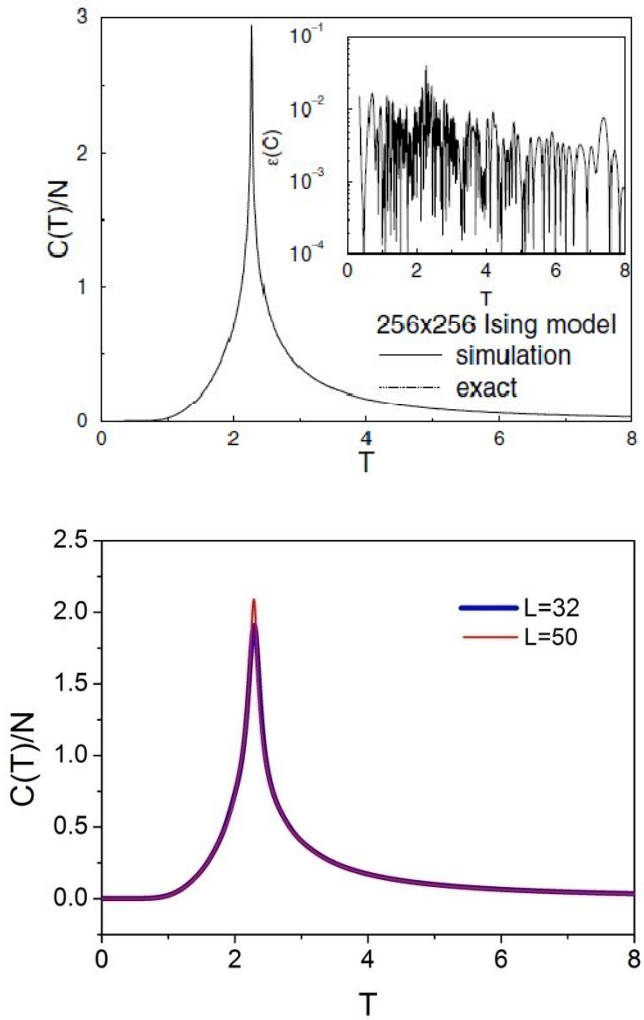

Fig. 14. (a) Specific heat curve from the work of Wang and Landau,
(b) specific heat curve from this work

The simulation code works well with the 2D Ising model, and as the lattice size gets bigger, the result gets closer to the work of Wang and Landau. Figure 15 has the result of the calculation of the internal energy and the entropy from our code. They show the second order phase transition phenomena, which is the ferromagnetic transition.

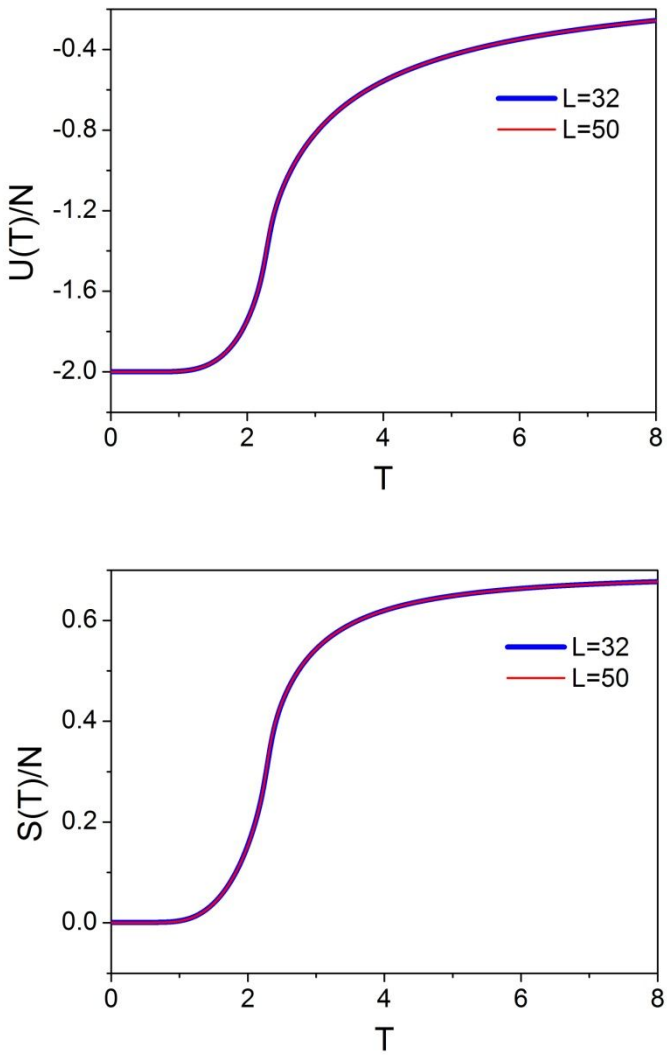

Fig. 15. (a) Internal energy curve and (b) entropy curve from this work

4.1.2 2D Potts model

Figure 16-20 show the verification simulation results of 2D Potts model, and compare them with the work of Wang and Landau.

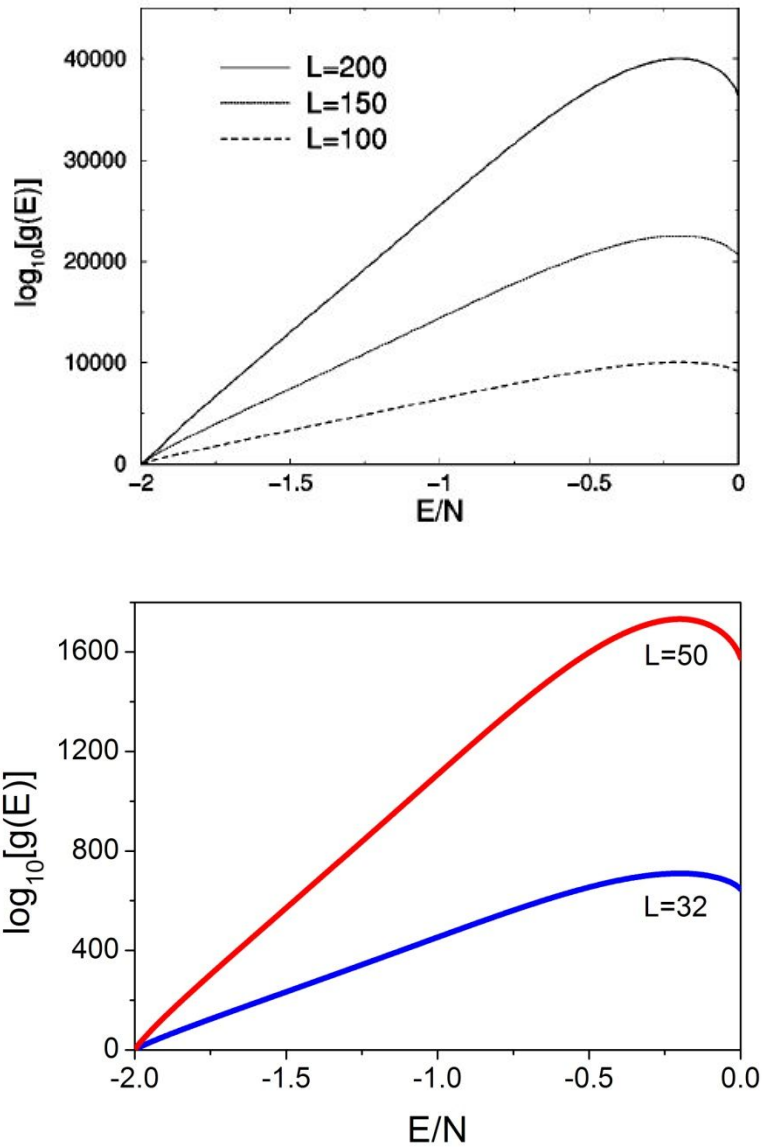

Fig. 16. (a) Density of states (DOS) from the work of Wang and Landau, (b) DOS from this work

Our simulation code resulted values of density of states (DOS) of 32×32 and 50×50 2D q=10 Potts model, which are shown and compared with the values from the work of Landau and Wang, in Figure 15. We could not do as big as Wang and Landau did, since the parallelization of the program is not

done yet, however, the simulation result shows well the DOS of the Potts model.

Fig. 17-20 show the free energy, internal energy, entropy, and specific heat, respectively. Here, all values are calculated and divided by $N = L \times L$, the number of unit cells in the model.

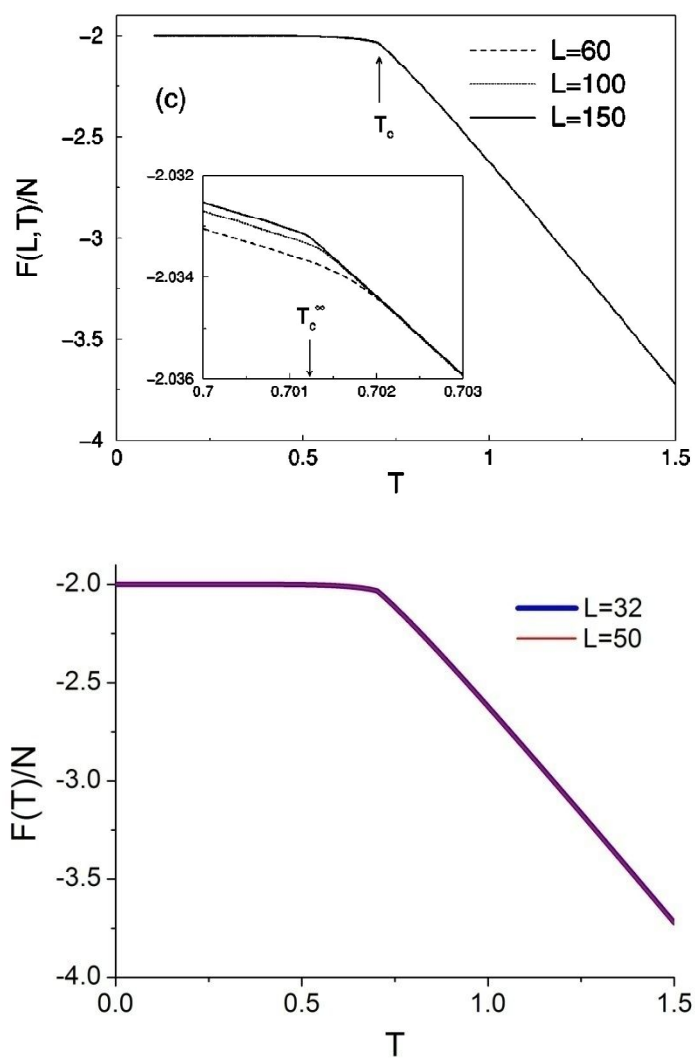

Fig. 17. (a) Free energy curve from the work of Wang and Landau, (b) free energy curve from this work

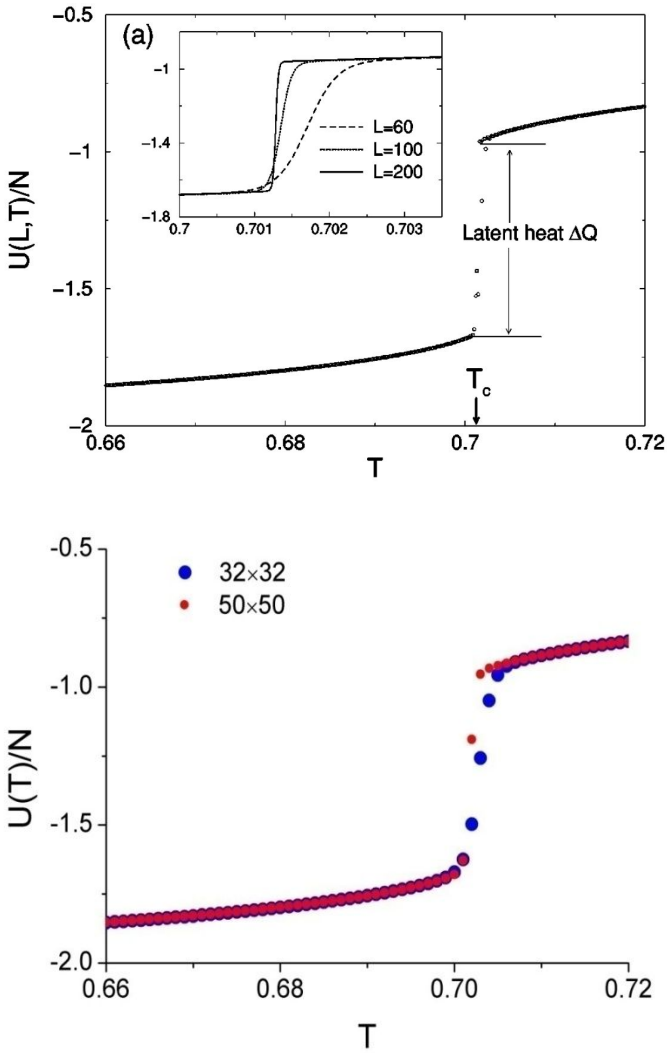

Fig. 18. (a) Internal energy curve from the work of Wang and Landau, (b) internal energy curve from this work

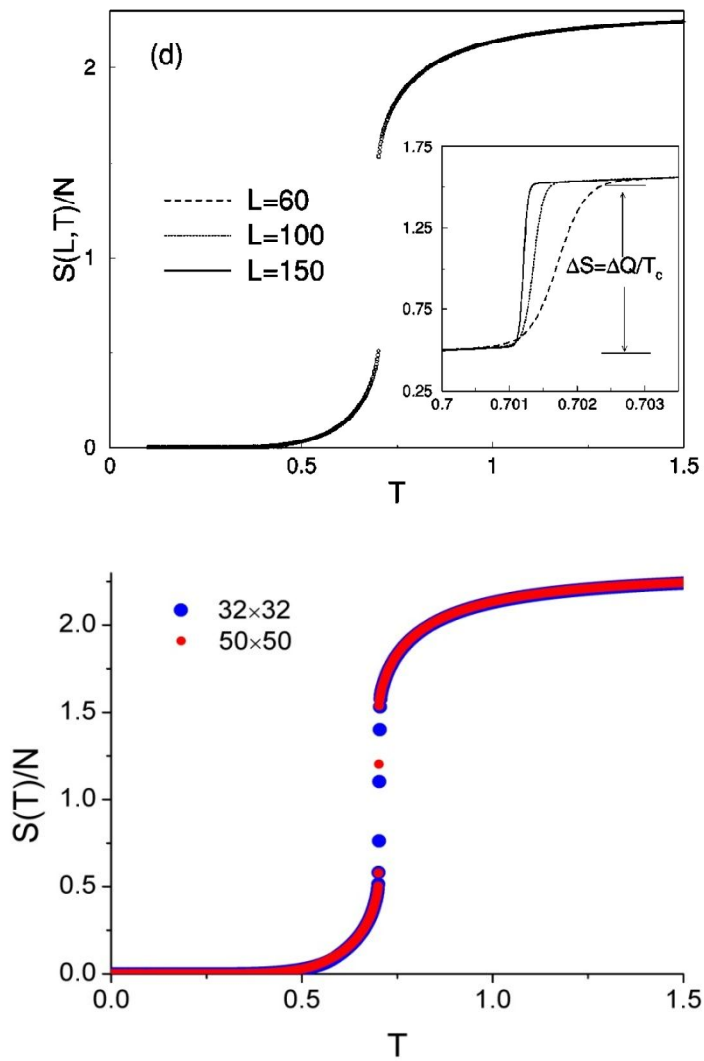

Fig. 19. (a) Entropy curve from the work of Wang and Landau, (b) entropy curve from this work

They are well matched, and from the internal energy and the entropy curves, it is clear that this system has first order transition near $T_c \cong 0.701$.

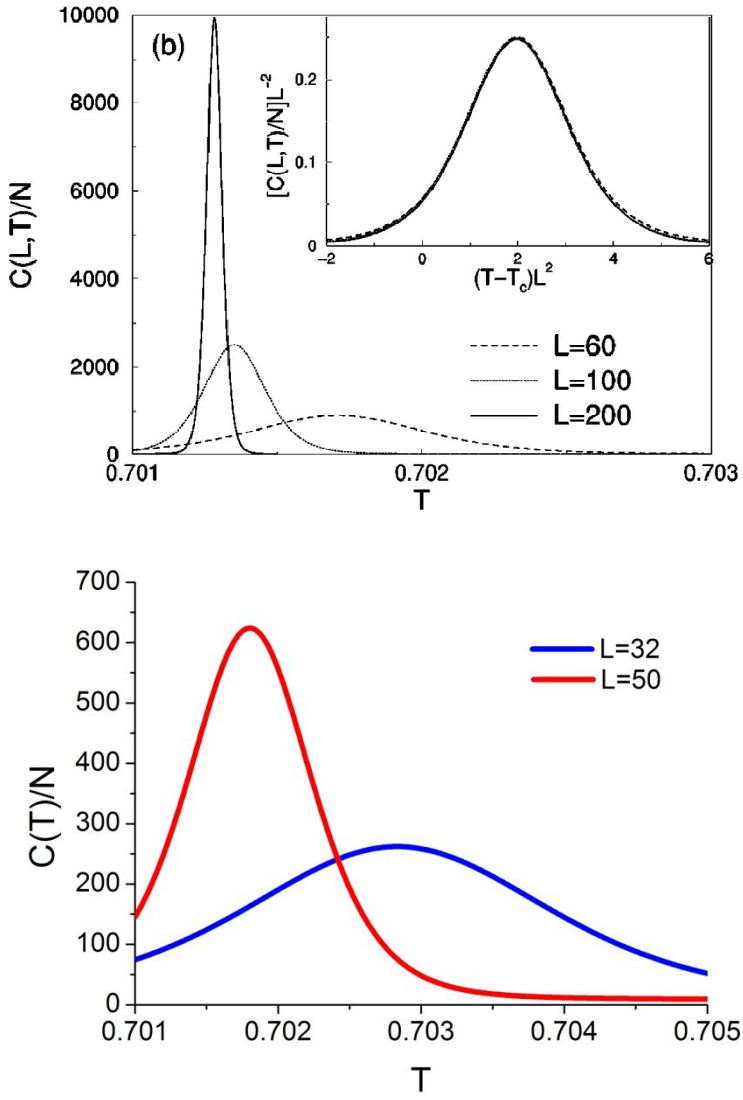

Fig. 20. (a) Specific heat curve from the work of Wang and Landau,
(b) specific heat curve from this work

The specific heat gets closer to the work of Wang and Landau as the lattice size gets bigger.

4.2 Fe_2MnAlC κ-carbide simulation results

The simulation was done with 10×10×10 3D lattices, in which each lattice can have 6 states of Fe_2MnAlC κ-carbide. The difference between the lowest energy $-1.737454381 \times 10^{-11}$ J and the highest energy $-1.737410840 \times 10^{-11}$ J was divided into 8000 levels, and the DOS of whole energy levels was calculated. Figure 21 shows the result, with normalising the lowest energy as 0, and the highest energy as 1.

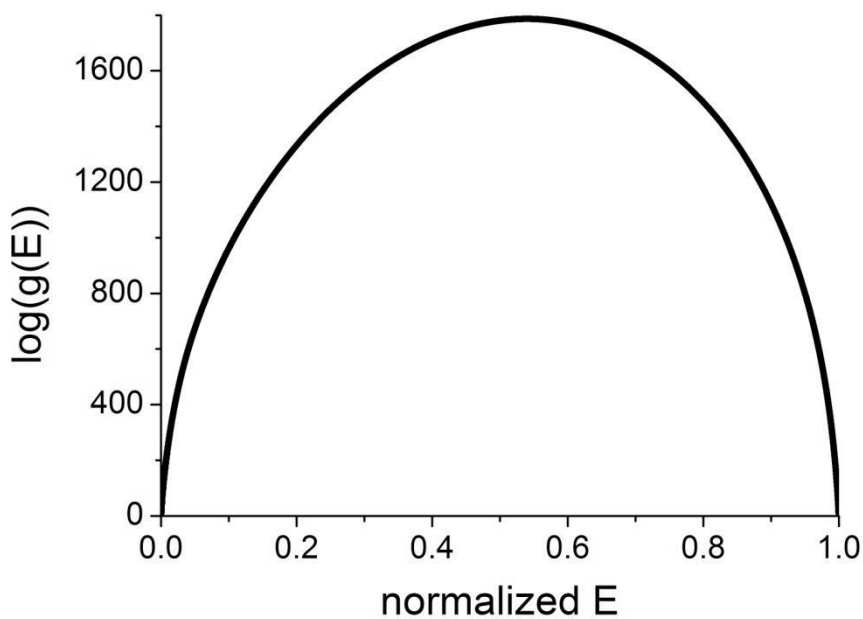

Fig. 21. DOS of the cell gas model of Fe_2MnAlC κ-carbide

The energy levels of 6 states are not in symmetry, thus the form of the DOS is not symmetric. From the resultant DOS, the Helmholtz free energy was obtained (Figure 22). Here, the free energy F, internal energy U, the

entropy S, and the specific heat C are calculated for one mole of Fe_2MnAlC κ-carbide, and divided with N, the number of unit cells in the model, which is 10×10×10=1000 in this simulation. The unit of temperature is given in k_BT (J) for convenience.

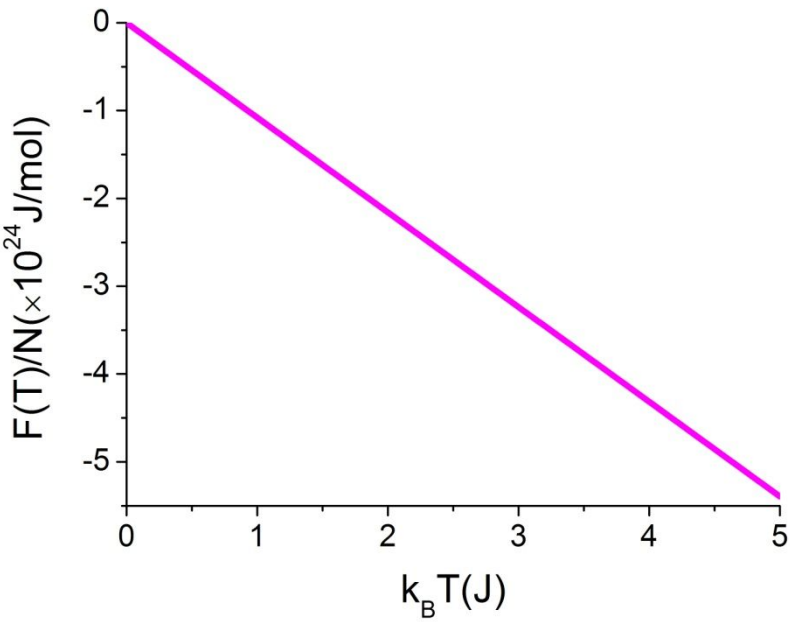

Fig. 22. The free energy of the cell gas model of Fe_2MnAlC κ-carbide

It decreases simply, and linearly. The system of the cell gas model has no interactions between cells, so the phase transition would occur at almost zero temperature. The specific heat curve of Figure 23 clearly shows this phenomenon.

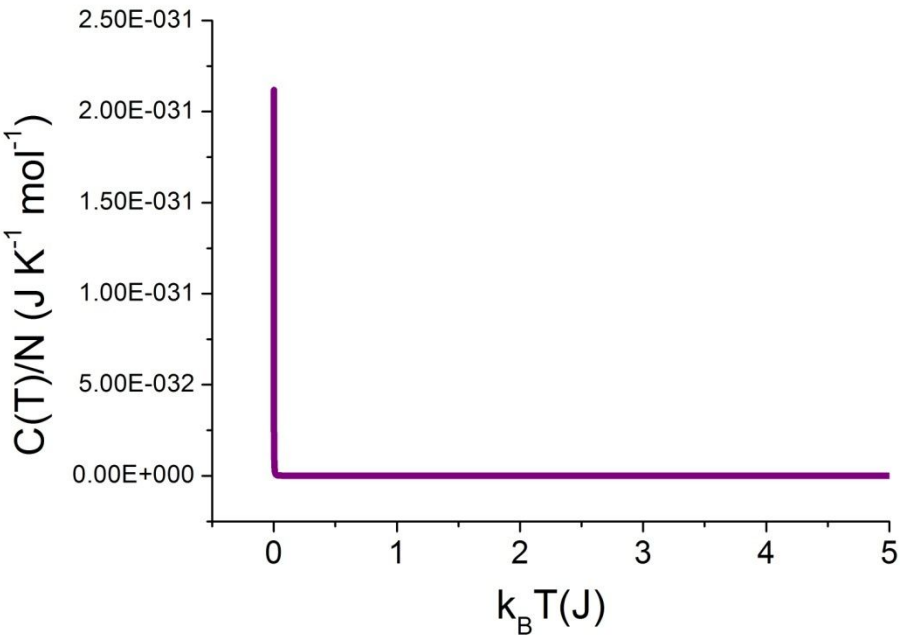

Fig. 23. The specific heat of the cell gas model of Fe_2MnAlC κ-carbide

The specific heat is very close to zero, since the system has no interactions. Moreover, it is almost same at high temperatures, but as the temperature gets lower near zero temperature, it arises. Thus, the singularity point of the specific heat curve would be at zero temperature. From the more detailed curve of Figure 24, the transition may start at about 0.0022 k_BT (J).

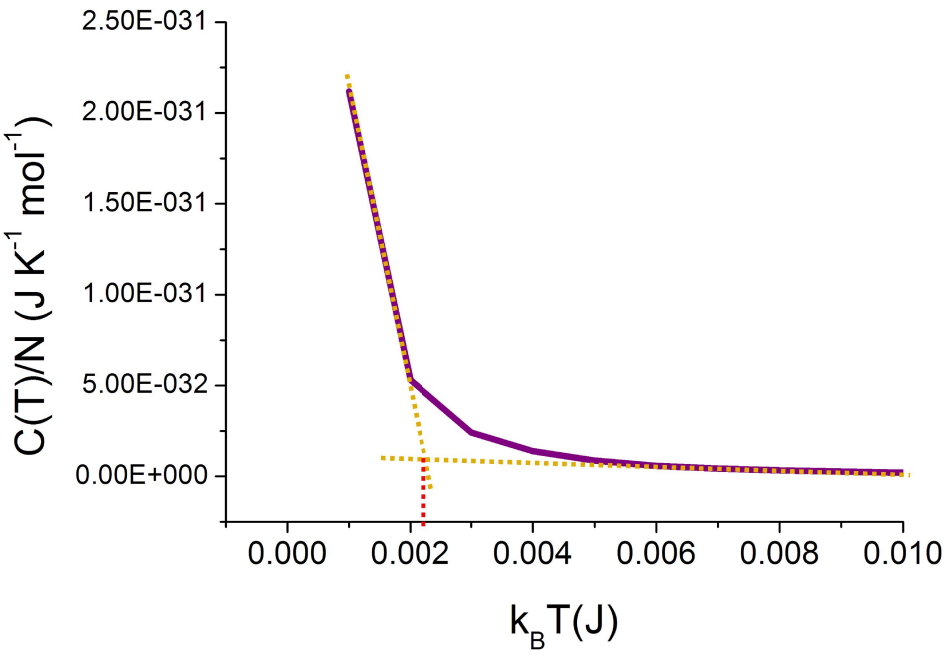

Fig. 24. More detailed specific heat curve of the cell gas model of Fe_2MnAlC κ-carbide

The value of the internal energy and the entropy was calculated for the temperature near zero. They are almost same for all temperatures. Since

$$C(T) \equiv \frac{\partial U}{\partial T} \tag{48}$$

the specific heat is almost zero for the temperature over zero.

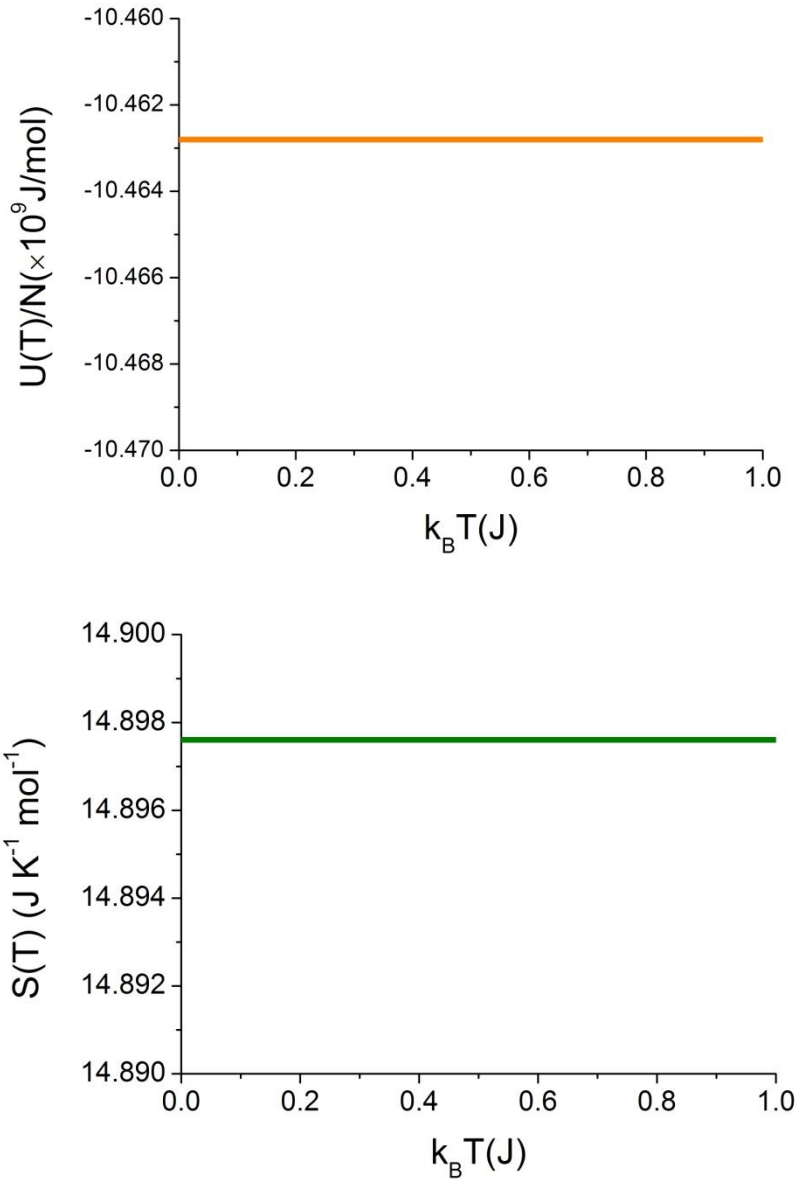

Fig. 25. Internal energy and entropy curve of the cell gas model of Fe_2MnAlC κ-carbide

5. Conclusive summary

The basic knowledge on the Monte Carlo method was studied, and the simulation code of the Wang-Landau algorithm was made, and verified. And finally, with a simple model, the calculation of the thermodynamic properties of Fe_2MnAlC κ-carbide was attempted.

The cell gas model simulation has its limitation that it does not have any information on interaction. And also, it is hard to obtain the information. However, this model can be a starting point of approaching to the real material system, and it predicts well the temperature dependence of the free energy, internal energy, specific heat, and entropy, showing its non-interacting characteristics.

For the further work, we will accomplish the parallelization of the simulation code. And with the special quasi-random structure (SQS) or the cluster variation method, which calculates interactions between supercells of the alloy, we will make use of the Monte Carlo simulation for more realistic Fe_2MnAlC κ-carbide system.

References

Abreu, C. R., Escobedo, F. A. (2006). A general framework for non-Boltzmann Monte Carlo sampling. *J. Chem. Phys*. 2006 Feb 7;124(5):054116.

Allen, M. P., Tildesley, D. J. (1987). *Computer simulation of liquids*. Oxford University Press.

Berg, B. A., Neuhaus, T., (1991). Multicanonical algorithms for first order phase transitions. Phys. Rev. B., 267. 249-253.

Binder, K., and Heerman, D. W. (2010). *Monte Carlo Simulation in Statistical Physics: An introduction.* Springer

Chang, K. M., Chao, C. G., and Liu, T. F. (2010). Excellent combination of strength and ductility in an Fe-9Al-28Mn-1.8C alloy. *Scripta Mater.*, 63(2):162-165

Frenkel, D. and Smit, B. (2002). *Understanding molecular simulation.* Academic Press.

Frommeyer, G. and Brüx, U. (2006). Microstructures and mechanical properties of high-strength Fe-Mn-Al-C light-weight TRIPLEX steels. *Steel research international*, vol. 77:627-633

Han, S. Y., Shin, S. Y., Lee, S., Kim, N. J., Kwak, J. –H. and Chin, K. –G. (2010). Fracture mechanisms of cold-rolled light-weight steel plates

containing different carbon content. *Kor. J. Met. Mater,*. Vol. 48, No. 5, pp. 377~386

Kohn, W. and Sham, L. (1965). Self-consistent equations including exchange and correlation effects. *Phys. Rev*, 140(4A):A1133–A1138.

Kimura, Y., Handa, K., and Mishima, Y. (2004). Micro structure control and ductility improvement of the two-phase gamma Fe/kappa-(Fe, Mn)3AlC alloys in the Fe-Mn-Al-C quaternary system. *Intermetallics*, 12(6):607-617

Landau, D. P. and Binder, K. (2009). *A guide to Monte-Carlo simulations in statistical physics*. Cambridge University Press.

Landau, D. P., Tsai, S. H., and Exler, M. (2004). A new approach to Monte Carlo simulations in statistical physics: Wang-Landau sampling. *Am J. Phys*, 72(10):1294–1302.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., and a. Teller (1953). Equation of state calculations by fast computing machines. The Journal of Chemical physics.

Newman, M. E. J, and Barkema, G. T. (1999). *Monte Carlo methods in statistical physics*. Oxford University Press.

Perdew, J., Burke, K., and Wang, Y. (1996). Generalized gradient approximation for the exchange-correlation hole of a many-electron system. *Phys. Rev. B.*, 54(23):16533–16539.

Seo, S. -W. (2011). First principles calculation on thermodynamic properties

and magnetism of κ-carbide and Monte-Carlo cell gas model. Master thesis. POSTECH.

Torrie, G. M., Valleau, J. P. (1977). Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *J. Comp. Phys.* 23, 187-199.

Wang, F. and Landau, D. P. (2001a). Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys Rev Lett*, 86(10):2050-2053.

Wang, F. and Landau, D. P. (2001b). Determining the density of states for classical statistical models: A random walk algorithm to produce a flat histogram. *Phys Rev E*, 64(5):056101

Wallin, K., Saario, T., and Torronen, K. (1986). Fracture of brittle particles in a ductile matrix. *Int. J. Fracture*, 32(3):201-209.

Weinert, M., Wimmer, E., and Freeman, A. (1982). Total-energy allelectron density functional method for bulk solids and surfaces. *Phys. Rev. B.*, 26(8):4571–4578.

Wikipedia. http://en.wikipedia.org

Wimmer, E., Krakauer, H., Weinert, M., and Freeman, A. J. (1981). Fullpotential self-consistent linearized-augmented-plane-wave method for calculating the electronic-structure of molecules and surfaces - O2 molecule. *Phys. Rev. B.*, 24(2):864–875.

Appendix

This is the source code of the simulation program code of Wang-Landau algorithm, which is written in C++ language, and aided by the random number generator of Agner Fog and the EXTNUM library.

Agner Fog's random number generator:

http://www.agner.org/random

EXTNUM big number library:

http://research.nmsu.edu/molbio/bioinfo/bioinformatics/extnum/code.html

1. Program source code files

WL.h

```
#include <fstream>
#include <iomanip>
#include <cstring>
#include <cmath>
#include <ctime>

#include <vector>

#include "randoma.h"
#include "real.h"

using namespace std;

class WangLandau
{
        ifstream infile_env;
        ifstream infile_ENERGY;
        ofstream outfile_log;
        ofstream outfile_result;

        // temporary variable
        char tempString1[40], tempString2[40];

        // number of histograms
        int num_HISTOGRAM_Bars;

        // number of energy states
        int num_ENERGY_States;
```

```
        // energy state
        vector<double> ENERGY_States;


        int L, N;                                   // L: L of L*L*L number of unit cell

        double log_f;                               // log_f: log of modification factor
        double kB;                                  // kB: Boltzmann Constant
        double ENERGY_GroundState;
        double ENERGY_HighestState;
        double ENERGY_Range;

        int ***CELL;
        unsigned int *HISTOGRAM;
        double *LOG_DOS;

        double ENERGY_Present;
        double ENERGY_Trial;

        int index_ENERGY_Present;
        int index_ENERGY_Trial;

        double num_sweep;
        double mcs;


public:
        WangLandau();
        ~WangLandau();

        void get_env();
        void get_ENERGY();
        void read_input();
        void initialize_variable();

        void flip_spin();
        double HAMILTONIAN();
        double deltaE(int state1, int state2);
        bool is_flat();
        void do_step();
        void write_result();
};
```

WL.cpp

```
#include "CLASS_WL.h"

void WangLandau::get_env()
{

        //get environment
        int i;
        char tempString[40];
        infile_env.open("env.txt");

        for(i=0;i<2;i++)
```

```
                tempString[i] = infile_env.get();
        tempString[i] = '\0';

        L = 0;
        if(!strcmp(tempString, "L:")) {
                infile_env >> tempString;
                for(i=0; tempString[i] != '\0'; i++) {
                        L *= 10;
                        L += tempString[i] - 48;
                }
        }
        else {
                cout << "Input file is incorrect." << endl;
                exit(1);
        }

        num_HISTOGRAM_Bars = 0;
        infile_env.get();           // deal with the new line character
        for(i=0;i<10;i++)
                tempString[i] = infile_env.get();
        tempString[i] = '\0';
        if(!strcmp(tempString, "HISTOGRAM:")) {
                infile_env >> tempString;
                for(i=0; tempString[i] != '\0'; i++) {
                        num_HISTOGRAM_Bars *= 10;
                        num_HISTOGRAM_Bars += tempString[i] - 48;
                }
        }
        else {
                cout << "Input file is incorrect." << endl;
                exit(2);
        }

        infile_env.close();
}
void WangLandau::get_ENERGY()
{
        //get energy
        int i;
        bool flag1 = TRUE, flag2 = TRUE;
        double temp_ENERGY;
        char tempString[40];
        char tempChar;

        infile_ENERGY.open("energy.txt");

        int count = 0;
        while(1){

                for(i=0;    ; i++) {
                        tempString[i] = infile_ENERGY.get();
                        if(tempString[i] == EOF)
                        {
                                infile_ENERGY.seekg(-1L,ios::end);
                                tempChar = infile_ENERGY.get();
                                if(tempChar== '\n') {
                                        flag1 = FALSE;
                                        break;
                                }
                                else{
                                        flag2 = FALSE;
                                        break;
```

```
                                        }
                                }
                                if(tempString[i] == '\n')
                                        break;
                        }

                        if(flag1 == FALSE) break;

                        count++;
                        tempString[i] = '\0';

                        temp_ENERGY = atof(tempString);
                        ENERGY_States.push_back(1);
                        ENERGY_States[count-1]=temp_ENERGY;

                        if(flag2 == FALSE) break;
                }
                num_ENERGY_States = count;

                infile_ENERGY.close();
}

void WangLandau::read_input()
{
                get_env();
                get_ENERGY();
}

void WangLandau::initialize_variable()
{
                int ENERGY_RandSeed = time(NULL);
                static CRandomSFMTA0 ENERGY_RandObject(ENERGY_RandSeed);


                N = L*L*L;
                log_f = 1.0;                    // log_f: log of modification factor
                kB = 1.3806504e-23;     // kB: Boltzmann Constant
                ENERGY_GroundState = ENERGY_States[0] * N;
                ENERGY_HighestState = ENERGY_States[num_ENERGY_States - 1] * N;
                ENERGY_Range = ENERGY_HighestState - ENERGY_GroundState;


                // Set DOS g(E) = 1, as a simple guess
                for(int i=0;i<num_HISTOGRAM_Bars;i++)
                        LOG_DOS[i] = 0.0;

                ////////////////////////////////// scheme(2) Choose an initial state
                for(int i=0;i<L;i++)
                        for(int j=0;j<L;j++)
                                for(int k=0;k<L;k++)
                                        CELL[i][j][k] = ENERGY_RandObject.IRandomX(0, 5);
                ENERGY_Present = HAMILTONIAN();

                index_ENERGY_Present=(int)( (ENERGY_Present -
ENERGY_GroundState)/(ENERGY_Range/num_HISTOGRAM_Bars) );
                if(index_ENERGY_Present > num_HISTOGRAM_Bars-1)                  // index overflow: if all cells
are at Highest states, index = N -> make it N-1
                        index_ENERGY_Present = num_HISTOGRAM_Bars-1;
}

bool WangLandau::is_flat()
{
```

```
	double average_HISTOGRAM = 0.8*(double)num_sweep/(double)num_HISTOGRAM_Bars;

	for(int i=0;i<num_HISTOGRAM_Bars;i++) {
		if(HISTOGRAM[i] < average_HISTOGRAM)
			return false;
	}
	return true;
}


void WangLandau::do_step()
{
	double tempDOUBLE;

	double mcsMax = 0.0;
	mcs = 0.0;
	do{
		num_sweep=0.0;
		// initialize, reset Histogram
		for(int i=0;i<num_HISTOGRAM_Bars;i++)
			HISTOGRAM[i] = 0;
		do{
			mcsMax = mcs+max(100000/N, 1);
			for( ; mcs<mcsMax; mcs += 1.0)
				flip_spin();

			tempDOUBLE = num_sweep - floor(num_sweep/10000000) * 10000000;
			if(tempDOUBLE == 0.0)
			{
				outfile_log << endl;
				outfile_log << "Sweeps of " << setprecision(0) << num_sweep
<< "are done:"<< endl;
				for(int i=0;i<num_HISTOGRAM_Bars;i++)
					outfile_log << HISTOGRAM[i] << '\t';
			}
		}while(!is_flat());

		////////////////////////////// scheme(9) If the histogram is 'flat', decrease f, e.g. f = sqrt(f)
		log_f = log_f * 0.5;

		outfile_log << endl;
		outfile_log << "//////////////////////////////////////////////";
		outfile_log << endl;
		outfile_log << "Now updates factor..";
		outfile_log << endl;
		outfile_log << "new update factor : " << setprecision(16) << exp(log_f);

	}while(log_f>1.0e-8);

	outfile_log << endl << endl << endl;
	outfile_log << "///////////////End of Calculation/////////////////" << endl;

}

void WangLandau::flip_spin()
{
	static int ENERGY_RandSeed = time(NULL);
	static int lattice_RandSeed = time(NULL)+1;
	static int ratio_RandSeed = time(NULL)+2;
	static CRandomSFMTA0 ENERGY_RandObject(ENERGY_RandSeed);
	static CRandomSFMTA1 Lattice_RandObject(lattice_RandSeed);
	static CRandomSFMTA1 Ratio_RandObject(ratio_RandSeed);
```

```
	int i, j, k, tempINT;
	int new_state;
	int previous_transaction;

	for(int steps = 0; steps<N ; steps++)
	{
		num_sweep += 1.0;
		tempINT = Lattice_RandObject.IRandomX(0, N-1);
		i = tempINT/(L*L);
		j = (tempINT - i*L*L)/L;
		k = tempINT - i*L*L - j*L;
		previous_transaction = CELL[i][j][k];

		new_state = ENERGY_RandObject.IRandomX(0, 5);
		CELL[i][j][k] = new_state;
		ENERGY_Trial= ENERGY_Present + deltaE(previous_transaction, new_state);
		index_ENERGY_Trial=(int)( (ENERGY_Trial -
ENERGY_GroundState)/(ENERGY_Range/num_HISTOGRAM_Bars) );
		// if index overflows
		if(index_ENERGY_Trial > num_HISTOGRAM_Bars-1)
			index_ENERGY_Trial = num_HISTOGRAM_Bars-1;

		if(LOG_DOS[index_ENERGY_Present] > LOG_DOS[index_ENERGY_Trial])
		{
			ENERGY_Present = ENERGY_Trial;
			index_ENERGY_Present = index_ENERGY_Trial;
		}
		else if(Ratio_RandObject.Random() < exp(LOG_DOS[index_ENERGY_Present]-
LOG_DOS[index_ENERGY_Trial] )) //accept
		{
			ENERGY_Present = ENERGY_Trial;
			index_ENERGY_Present = index_ENERGY_Trial;
		}
		else
			CELL[i][j][k] = previous_transaction;

		LOG_DOS[index_ENERGY_Present] += log_f;
		HISTOGRAM[index_ENERGY_Present] += 1;
	}
}

void WangLandau::write_result()
{
	////////////////////////////////PRINTING RESULTS////////////////////////////////
	outfile_result.setf(ios::fixed);
	outfile_result.precision(0);
	outfile_result << "Total sweep: " << mcs << endl;

	// rescaling
	extnum* real_LOG_DOS = NULL;
	real_LOG_DOS = new extnum[num_HISTOGRAM_Bars];
	for(int i=0;i<num_HISTOGRAM_Bars;i++)
		real_LOG_DOS[i]=0;
	for(int i=0;i<num_HISTOGRAM_Bars;i++)
		real_LOG_DOS[i] = LOG_DOS[i]-LOG_DOS[0]+log(2.0);
	outfile_result << endl;
	outfile_result << "Energy states:" << endl;
	for(unsigned int i=0;i < ENERGY_States.size() ; i++)
		outfile_result << ENERGY_States[i] << endl;
	outfile_result << endl;
	outfile_result << "==================================" << endl;
```

```
		outfile_result << "Histogram | result_log(DOS)    | real_log(DOS)" << endl;
		outfile_result << "===================================" << endl;

		outfile_result.precision(16);
		for(int i=0;i<num_HISTOGRAM_Bars;i++)
			outfile_result << setprecision(0) << i << "\t" << setprecision(16) << LOG_DOS[i] << "\t"
<< real_LOG_DOS[i] << endl;

		outfile_result << "===================================" << endl << endl << endl;


		delete[] real_LOG_DOS;
}

WangLandau::WangLandau()
{
		char tempString1[40];
		char tempString2[40];

		read_input();

		sprintf(tempString1, "log_L%d_H%d.txt", L, num_HISTOGRAM_Bars);
		sprintf(tempString2, "result_L%d_H%d.txt", L, num_HISTOGRAM_Bars);

		outfile_log.open(tempString1);
		outfile_result.open(tempString2);

		cout << "L: " << L << endl;
		cout << "number of Histogram Bars: " << num_HISTOGRAM_Bars << endl;

		outfile_result << "L: " << L << endl;
		outfile_result << "number of Histogram Bars: " << num_HISTOGRAM_Bars << endl;

		cout << "Calculating process is going on...Please check your log file occasionally to see the interim
results.";


		CELL = new int**[L];
		for(int i=0;i<L;i++){
			CELL[i] = new int*[L];
			for(int j=0;j<L;j++)
				CELL[i][j] = new int[L];
		}

		HISTOGRAM = new unsigned int[num_HISTOGRAM_Bars];
		LOG_DOS = new double[num_HISTOGRAM_Bars];

		initialize_variable();
}

WangLandau::~WangLandau()
{
		for(int i=0;i<L;i++){
			for(int j=0;j<L;j++)
				delete[] CELL[i][j];
			delete[] CELL[i];
		}
		delete[] CELL;

		delete[] HISTOGRAM;
		delete[] LOG_DOS;
```

```
	outfile_log.close();
	outfile_result.close();
}



double WangLandau::HAMILTONIAN()
{
	double temp = 0.0;
	for(int i=0;i<L;i++)
		for(int j=0;j<L;j++)
			for(int k=0;k<L;k++)
				temp += ENERGY_States[CELL[i][j][k]];
	return temp;
}
double WangLandau::deltaE(int state1, int state2)
{
	return ENERGY_States[state2] - ENERGY_States[state1];
}
```

2. Needed environment files: example

energy.txt – includes possible energy states

```
-1.258431810e9
-1.258427161e9
-1.258413450e9
-1.258412230e9
-1.258403740e9
-1.258400273e9
```

env.txt – includes lattice size and number of histogram size

```
L: 10
HISTOGRAM: 8000
```

CURRICULUM VITAE

Name: Lee, Jee Yong

E-mail: leejy83@postech.ac.kr

Date of birth: 18th September, 1983

Place of birth: Gwangju, South Korea

Education

M. S. 2012, POSTECH (Pohang, Korea), Graduate Institute of Ferrous Technology, Computational Metallurgy Group

B. S. 2010, POSTECH (Pohang, Korea), Department of Computer Science and Engineering

Acknowledgement

I am sincerely thankful to my advisor, Professor Kim, In Gee, for his guide and help. I am also grateful to my director Professor Bhadeshia, H. K. D. H., and Professor Suh, Dong-Woo. I would like to express my thanks to all the members of Computational Metallurgy Laboratory (CML) and Graduate Institute of Ferrous Technology (GIFT) for their friendship and support.

제가 이렇게 자라고 공부할 수 있도록 길러주신 친가, 외가 조부모님, 부모님께 엎드려 마음 깊이 감사 드립니다. 또한 어려서부터 저와 함께 했던 유일한 여동생 은지에게, 그리고 자주보지는 못하지만 가까운, 우리 친척 어른들과 형, 누님, 동생들에게 감사를 전합니다.

지금의 제가 있을 수 있도록 많은 것들을 배우게 해준 제 많은 친구들, 특별히 고등학교부터 계속 붙어있는 수용이와 대학에 와서 많은 생활을 함께했던 석범이, 주호, 매일 같이 운동하는 정우에게, 또 많은 오랜 친구들에게, 녹록하지 않았던 중고등학교 시절, 생경했던 첫 대학생활의 희로애락을 함께한 10분반과 컴퓨터공학과, 전혀 다른 세상이었던 군대, 그 외 여러 곳에서 만났던 우리 동기들, 선배님들, 후배님들께 많은 빚을 졌습니다.

제게 복음의 맛을 가르침과 삶으로 보여주신 DFC 여러 지체들, 특별히 함께 말씀을 나눴던 원정, 달라, 장우, 준섭, 종록, 그리고 도기현 선교사님과 백성균 선교사님께 감사 드리며, 함께 같은 길을 추구하는 법을 보여주신 수요 예배팀과 신우회 교수님들께도 많은 빚을 졌습니다. 지금껏 배운 것들, 앞으로 열심히 살도록 하겠습니다.

학부시절 많은 것들을 배울 수 있었던 북포항감리교회 지체분들, 함께 공동체를 세워가는 포항소망교회 지체분들께 마음 깊이 감사드립니다. 여러분과 함께 생활하며 배운 것은 셀 수도 없이 많습니다. 제게 대낮

같은 진리를 확고하게 심어준 큰믿음교회에도 감사하고 싶습니다.

어떤 것이 옳은 길인지를 고민하는 길 위에 알게 된 많은 분들께, 특별히 민주화를 위해, 정의가 강물처럼 흐르는 사회를 만들기 위해, 이름 없이 빛도 없이 젊음을 바친 여러분들께 마음 깊이 감사 드립니다.

마지막으로 많은 면에서 심히 부족한 저와 함께 일하고 있는 철강대학원 CML연구실 사람들, 교수님들께, 특별히 직접 일을 함께하는 김인기 교수님과 윤원석 박사님께 감사 드립니다.

그리고, 위에 쓴 모든 것의 이유와 근원, 처음과 나중이 되시는 하나님께 영광을 드립니다.

Praise God!