Parallel Tempering And Adaptive Spacing For Monte Carlo Simulation

---

Authors:	**Siu Wun Cheung**	Department of Mathematics, CUHK
	Zhao, Yiwei	Department of Mathematics, CUHK
Supervisors:	Dr. **Markus Eisenbach**	ORNL
	Dr. **Ying Wai Li**	ORNL
	Dr. **Kwai Wong**	ORNL/UTK
Hosts:	Dr. **Eduardo D'Azevedo**	ORNL

Content

PRELIMINARY

Abstract

A program has been written in C to simulate various physical models using the Metropolis-Hastings algorithm. Through experiments [1] with different parameters, the temperature dependence analysis of magnetization, Hamiltonian, magnetic suscpetibility and heat capacity of the models is performed. Convergence of the algorithm with parallel tempering and adaptive temperature spacing schemes is analyzed and compared.

Structure

The chief of the program contains a makefile, 3 C files and 2 header files:

`makefile`	
`main.c`	
`core_neighboring.c`	`core_neighboring.h`
`rng_init.c`	`rng_init.h`

The program also consists of 4 directories, namely `Ising_2d`, `Ising_3d`, `Nvct_2d` and `Nvct_3d`. Each directory corresponds to a physical model and consists of an input file, 6 C files and 7 header files:

`file.in`	
	`include.h`
`input.c`	`input.h`
`initialize.c`	`initialize.h`
`data.c`	`data.h`
`calculation.c`	`calculation.h`
`sampling.c`	`sampling.h`
`output.c`	`output.h`

The whole process is managed by `main.c`, which drives `core_neighboring.c` to do the simulation with `MPI` functions.

Firstly, an executable file is constucted through compilation by the makefile with user-specified environments. The executable file finds the corresponding directory, inside which the input file `file.in` is read by `input.c` in the master node and the all the input information is stored as a structure.

[1] All experiments were conducted on the supercomputer Kraken XT5 in Cray Linux Environment 3.1, located in Oak Ridge National Laboratory, Tennessee, United States.

The structure is then sent to all processors. In each processor the structure is read by `initialize.c` for initialization, and `rng_init.c` initializes the random number generator with a distinct random seed.

The Metropolis-Hastings algorithm is carried out by recursive equilibration steps and sampling steps, during which `data.c` records the samples, `calculation.c` does all the mathematical calculations and `sampling.c` determines probabilistic moves.

After the completion of the simulation, results are stored as a structure and gathered by the master node. They are then printed by running `output.c` in the master node.

How to use

The user should first link the header file `include.h` of the model in interest in `core_neighboring.c`, and then specify the model and the machine used in the `makefile`. The user should add any neccssary flags for a new machine in the makefile and make sure the GNU Scientic Library is linked. Typing in the `make clean` and `make` commands successively creates the corresponding executable file.

The user can specify parameters in the input file `file.in` in the corresponding directory. Parameters include:

Vector Dimension n **(Only for Nvct_2d and Nvct_3d)**
Grid length
Interaction strength J
Boltzmann constant k_B
Temperature pattern **(1 for arithmetic, 2 for geometric)**
Minimum temperature
Maximum temperature
Number of equilibration steps between successive exchanges
Number of exchanges between successive spacing adjustments
Number of adaptive temperature spacing adjustments
Adaptive temperature spacing adjustments aggressiveness
Number of sampling steps between successive exchanges
Number of exchanges in sampling time
Random seed

Running the executable file with a certain number of processors does the simulation. Output is generated and printed.

Chapter 0

Overview

0.1 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) algorithms are used for sampling from a certain probability distribution and calculating quantities of interest such as mean and variance. They are particularly useful when direct sampling is difficult.

The idea of Markov chain is to construct a Markov chain whose stationary distribution is set to be the target probability distribution π. After going through a sufficient number of equilibration steps, states generated approximately follow the target distribution, and samples are taken to perform parameter estimation by Bayesian inference. The underlying principle is to design a certain transition probability P to strike the detailed balance condition:

$$\text{For any } x, y \in S, \pi(x)P(x,y) = \pi(y)P(y,x)$$

0.2 n-vector model

The n-vector model is a mathematical model to study ferromagnetism in statisical mechanics. It can also be applied to study various fields such as lattice gas and neuroscience.

The n-vector model begins with a lattice, square in shape for the 2-D model and cube in shape for the 3-D model. In each position of the lattice sits a microstate, which is a magnetized spin pointing in a n-dimensional unit sphere. For n > 2, the model possesses a continum of states even the lattice has finitely many positions.

Dimension n	Name of model
1	Ising model
2	XY model
3	Heisenberg model
4	Standard model

Table 0.1: Names of models for different dimension n

↑ ↑ ↑ ↓
↓ ↑ ↓ ↑
↑ ↓ ↓ ↑
↑ ↓ ↓ ↓

Figure 0.1: A particular state of a 4×4 2-D Ising model

The spins interact with each other accordingly. For simplicity it is assumed that only nearest neighbors interact. The periodic boundary condition is employed, i.e. each spin on an edge of the lattice has a neighbor on the opposite edge.

The magnetization M and the Hamiltonian function H for representing the enrgy of a state σ are given by:

$$M(\sigma) = \sum_i \sigma_i$$

$$H(\sigma) = -J \sum_{<i,j>} \sigma_i \cdot \sigma_j$$

where σ_i is the microstate in position i, and <i, j> means the sum over nearest neighbors. The interaction strength J is positive for a ferromagnetic model such that the spins tend to be like parallel in the ground state.

The states of the models follow the Boltzmann distribution, the canonical ensemble for systems taking discrete values of energy and the most common ensemble in statistical mechanics. The Boltzmann distribution at temperature T is formulated as:

$$P(\sigma; T) = \frac{e^{-\beta H(\sigma)}}{Z(T)};\ \beta = \frac{1}{k_B T}$$

where k_B is the Boltzmann constant and Z(T) is the normalizing constant which is of high difficulty of calculation.

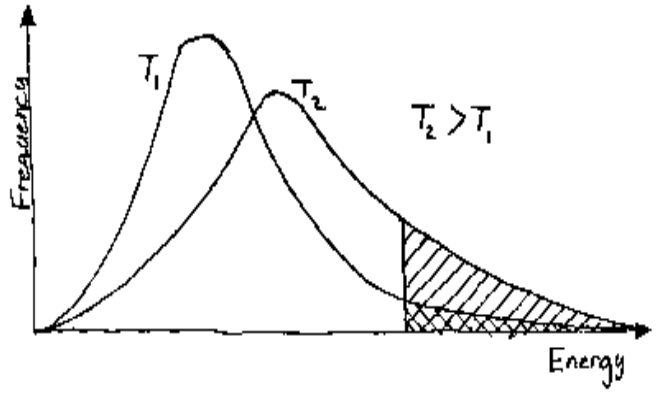

Fig 0.2: Frequency versus energy of the Boltzmann distribution at different temperatures

Chapter 1

Metropolis-Hastings Algorithm

1.1 Introduction

Metropolis-Hastings algorithm is a MCMC method to simulate the Boltzmann distribution. It is employed as the fundamental of this research project. The flow is as follows:

1) Generate an initial state randomly.
2) Go through an equilibration time, during which at each step:
   i. Choose a spin randomly and propose a flip on it.
   ii. Accept the flip with a probability P_{flip}:
   Accept ➔ MC goes to the new state.
   Reject ➔ MC retains the original state.
3) Go through a sampling time, during which at each step:
   i. Choose a spin randomly and propose a flip on it.
   ii. Accept the flip with a probability P_{flip}:
   Accept ➔ MC goes to the new state and a sample is taken.
   Reject ➔ MC retains the original state.
4) Calculate the average quantities of interest:

Mean magnetization per spin $< M > = \frac{\sum M(\sigma)}{\#\ spins \times \#\ samples}$

Mean Hamiltonian per spin $< H > = \frac{\sum H(\sigma)}{\#\ spins \times \#\ samples}$

Magnetic susceptibility $\chi = \frac{\beta \times (<M^2> - <M>^2)}{\#\ spins \times (\#\ samples - 1)}$

Heat capacity $C = \frac{\beta \times (<H^2> - <H>^2)}{T \times \#\ spins \times (\#\ samples - 1)}$

Acceptance ratio for sampling time $R_{samp} = \frac{\#\ samples}{\#\ sampling\ steps}$

It is obvious that the sequence of random variables is Markovian. The Metropolis-Hastings algorithm suggests the acceptance probability P_{flip} to be

$$P_{flip} = \min\{1, e^{-\beta \Delta H}\}$$

where ΔH is the Hamiltonian difference of the flip. It can easily be shown that the resultant Markov chain possesses the detailed balance condition and ergodicity with a unique stationary distribution as the Boltzmann distribution.

1.2 Experiment I: serial Metropolis-Hastings algorithm

The objective of this experiment is to calculate the quantities of interest with serial Metropolis-Hastings algorithm. The set up is as follows:

Model	2D Ising
Grid length	100
Interaction strength J	1.00
Boltzmann constant k_B	1.00
Temperature pattern	Arithmetic
Minimum temperature	0.50
Maximum temperature	5.00
# processors	96
# equilibration steps	10^9
# sampling steps	10^9
Random seed	888

Table 1.1: Set up for Experiment I

Each of the nodes was made to run a distinct temperature system and the results were put altogether as follows:

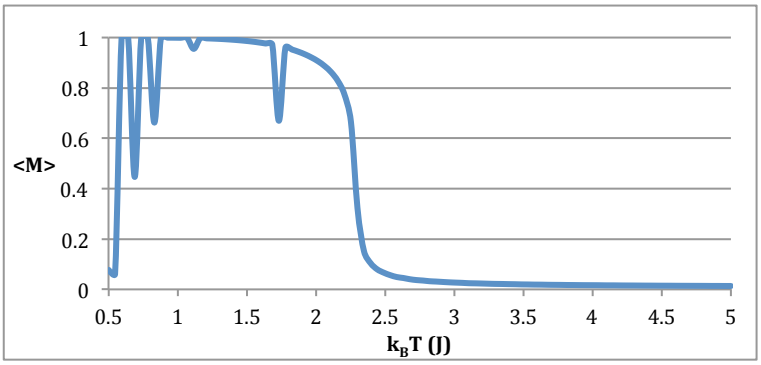

Figure 1.1: Mean magnetization per spin versus temperature

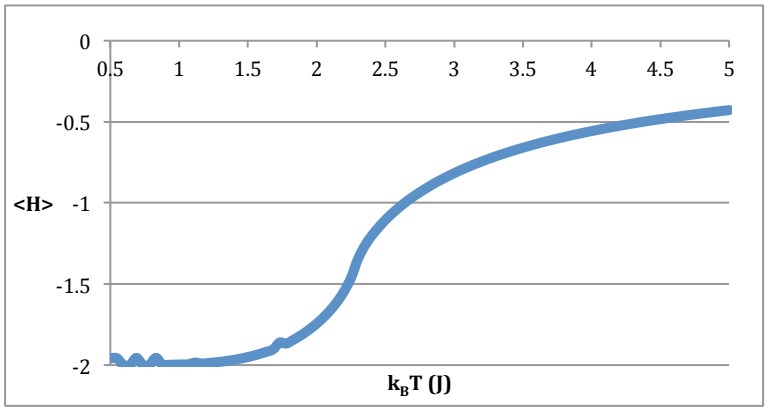

Figure 1.2: Mean Hamiltonian per spin versus temperature

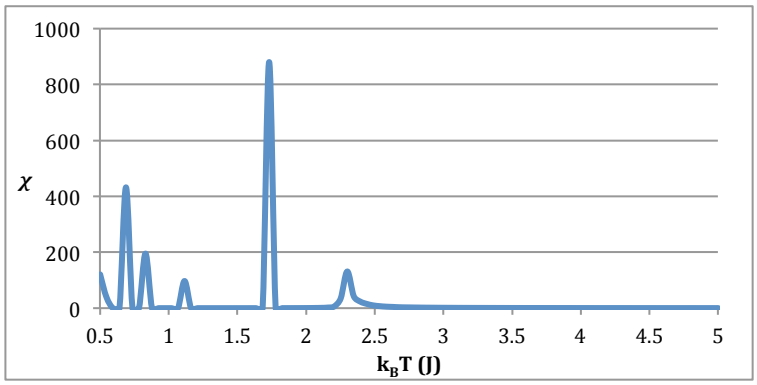

Figure 1.3: Magnetic susceptibility versus temperature

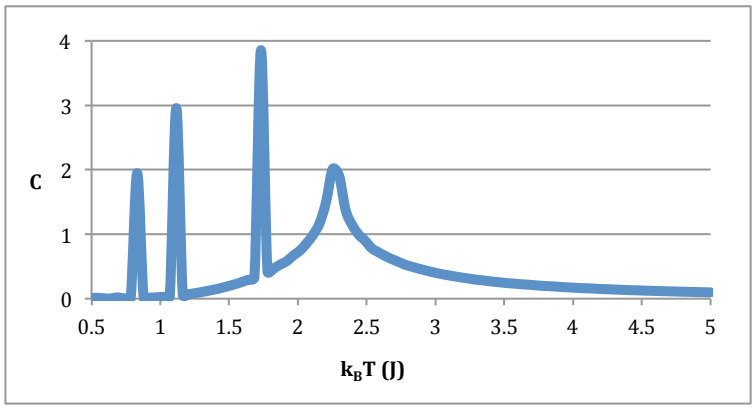

Figure 1.4: Heat capacity versus temperature

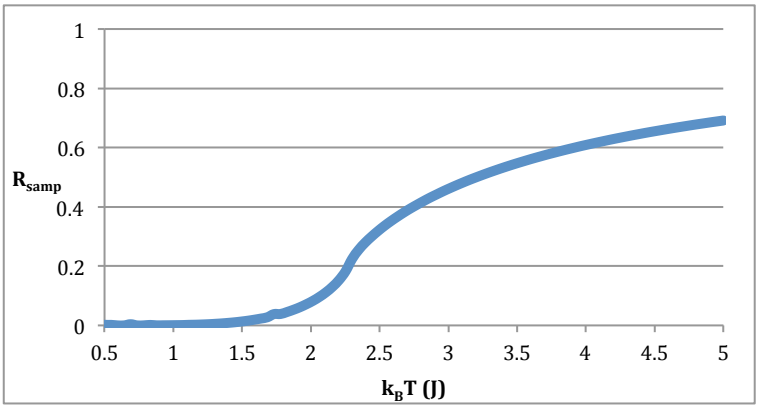

Figure 1.5: Acceptance ratio for sampling time versus temperature

A local, smooth and dramatic change of each of the quantities takes place near the phase transition T_c at which $k_BT_c \approx 2.3$ J.

The highly-oscillated curves, particularly unstable at low temperatures, in figures 1.1, 1.3 and 1.4 look wierd. Figure 1.5 shows that the acceptance ratios for sampling time are very close to 0 at low temperatures. This suggests that the sampling at low temperatures are biased and need to be improved.

Chapter 2

Replica Exchange

2.1 Drawback of Metropolis-Hastings algorithm

Recall the acceptance probability P_{flip} of the Metropolis-Hastings algorithm is given by:

$$P_{flip} = \min\{1, e^{-\beta\Delta H}\};\ \beta = \frac{1}{k_B T}$$

For low temperatures, $\beta \gg 0$ is a very large positive number. If we propose a flip with positive energy difference, i.e. $\Delta H > 0$, we have:

$$\beta\Delta H \gg 0 \implies e^{-\beta\Delta H} \approx 0 \implies P_{flip} \approx 0$$

On the contrary, if we propose a flip with negative energy difference, i.e. $\Delta H < 0$, we have:

$$\beta\Delta H < 0 \implies e^{-\beta\Delta H} > 1 \implies P_{flip} = 1$$

It is very likely to go through successive flips with negative energy difference, forcing the system to visit an energy minimum and be trapped around the state. Consequently it is impossible to generate states according to the Boltzmann distribution, resulting in biased sampling.

2.2 Idea of parallel tempering

Parallel tempering serves to improve convergence of Metropolis-Hastings algorithm in the problem discussed in the last section. A number of systems initialized with distinct temperatures run Metropolis-Hastings algorithm and exchanges of configurations are allowed during the sampling time.

The reason of doing so is to allow configurations at high temperatures to be trasferred to systems with low temperatures as the simulation process goes on, and rescue low temperature from being trapped at undesirable stable states with energy minimum.

2.3 Flow of parallel tempering

The flow of parallel tempering for Metropolis-Hastings algorithm on each temperature system is as follows:

1) Generate an initial state randomly.
2) Go through an equilibration time, during which at each step:
   i. Choose a spin randomly and propose a flip on it.
   ii. Accept the flip with a probability P_{flip}:
   Accept ➔ MC goes to the new state.
   Reject ➔ MC retains the original state.
3) Go through a sampling time partitioned by exchange periods, during which within an exchange period:
   i. Go through the period of a certain amount of recursive sampling steps by:
      a) Choose a spin randomly and propose a flip on it.
      b) Accept the flip with a probability P_{flip}:
      Accept ➔ MC goes to the new state and a sample is taken.
      Reject ➔ MC retains the original state.
   ii. At an exchange process:
      a) Propose a configuration exchange with the systems having the closest temperature alternatively.
      b) Accept the exchange with a probability $P_{exchange}$.

$$P_{exchange} = \min\{1, e^{\delta\beta\delta H}\}$$

where $\delta\beta$ and δH are the difference in β and Hamiltonian respectively.

4) Calculate the average quantities of interest.

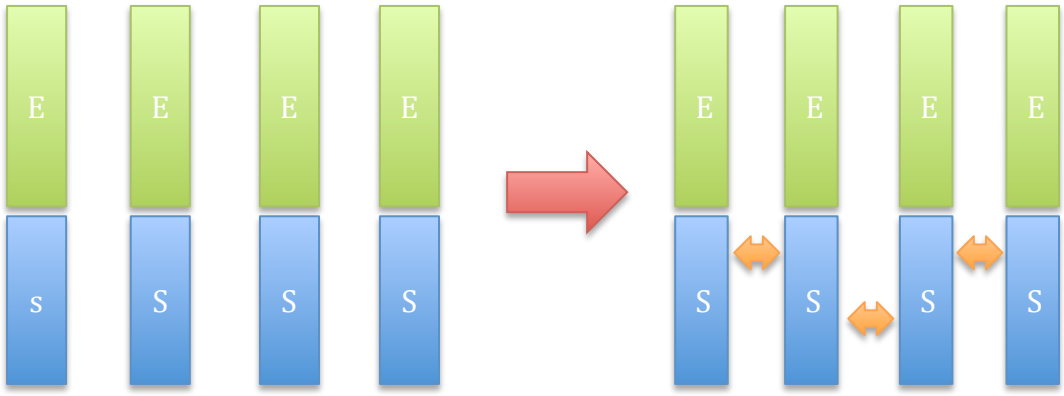

Figure 2.1: Metropolis-Hastings algorithm is now run on a number of processors among which exchanges of configurations are allowed between neighboring temperatures alternatively during sampling time

2.4 Experiment II: parallel Metropolis-Hastings algorithm

The objective of this experiment is to compare the convergence of Metropolis-Hastings algorthm with and without parallel tempering. The set up is as follows:

Model	2D Ising			
Grid length	100			
Interaction strength J	1.00			
Boltzmann constant k_B	1.00			
Temperature pattern	Arithmetic			
Minimum temperature	0.50			
Maximum temperature	5.00			
# processors	96			
# equilibration steps	10^9			
# sampling steps between succssive exchanges	10^9	10^7	10^5	10
# exchanges in sampling time	0	10^2	10^4	10^8
Random seed	888			

Table 2.1: Set up for Experiment II

The total number of sampling steps is fixed at 10^9, in which different number of evenly distributed replica exchanges are inserted to see the functionality of parallel tempering. Results are as follows:

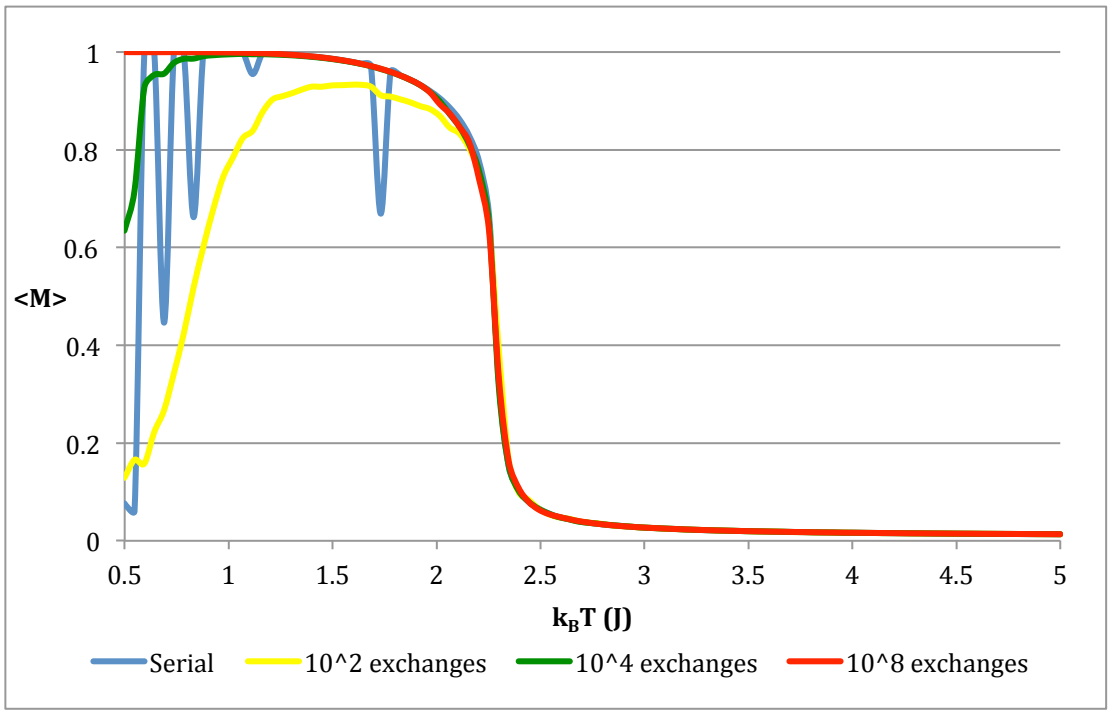

Figure 2.2: Mean magnetization per spin versus temperature at different number of exchanges

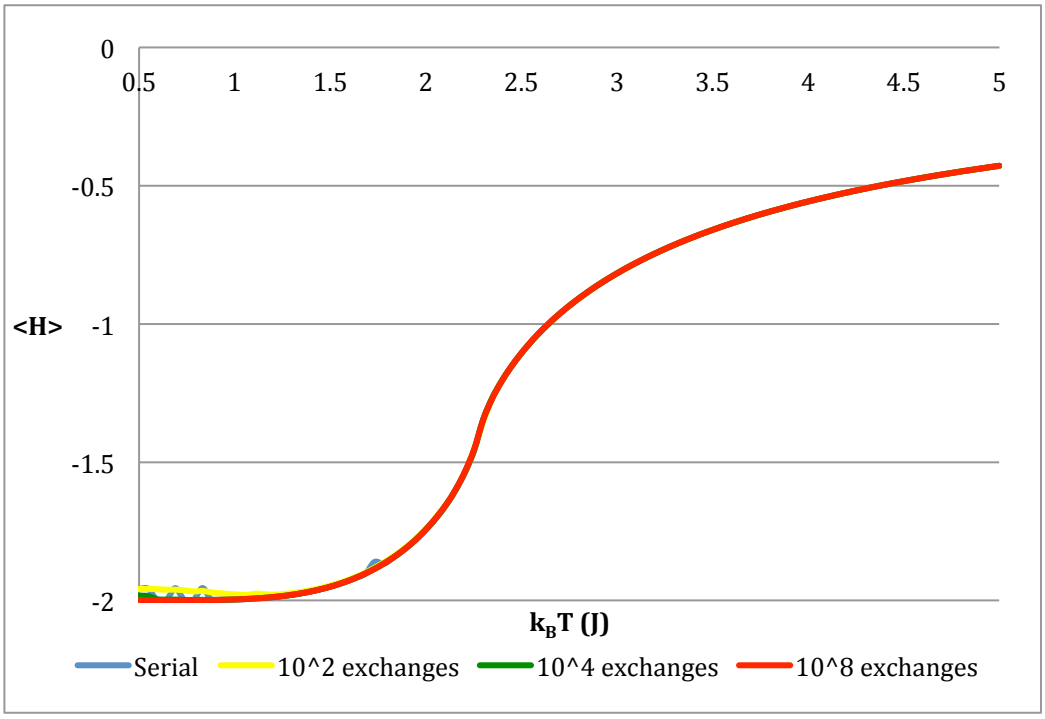

Figure 2.3: Mean Hamiltonian per spin versus temperature at different number of exchanges

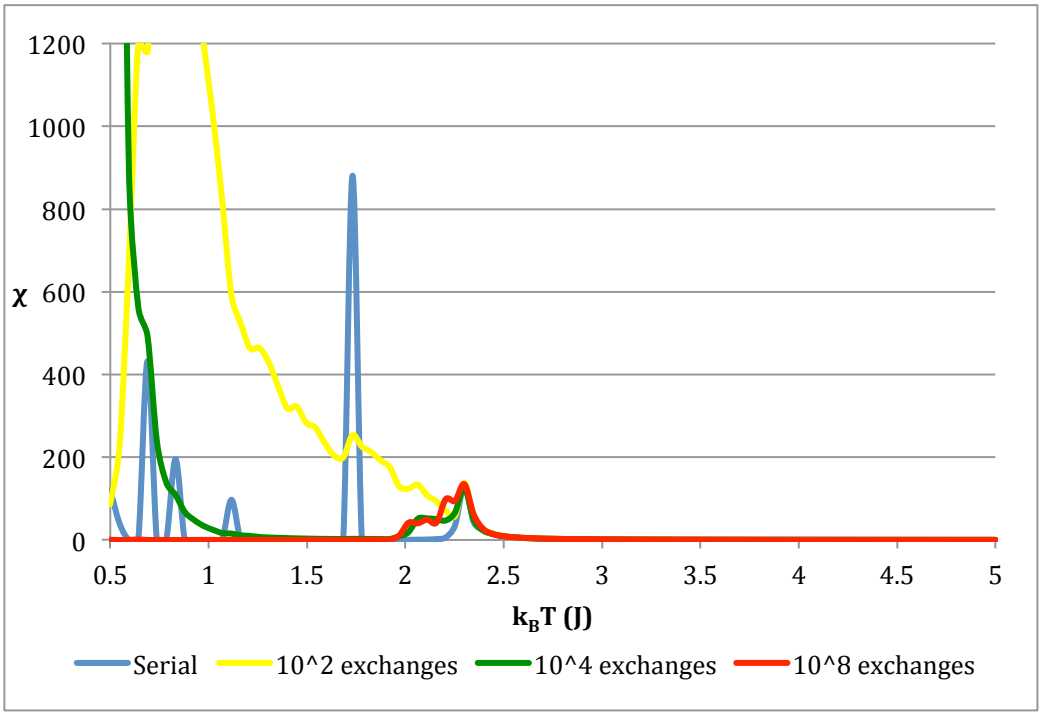

Figure 2.4: Magnetic susceptibility versus temperature at different number of exchanges

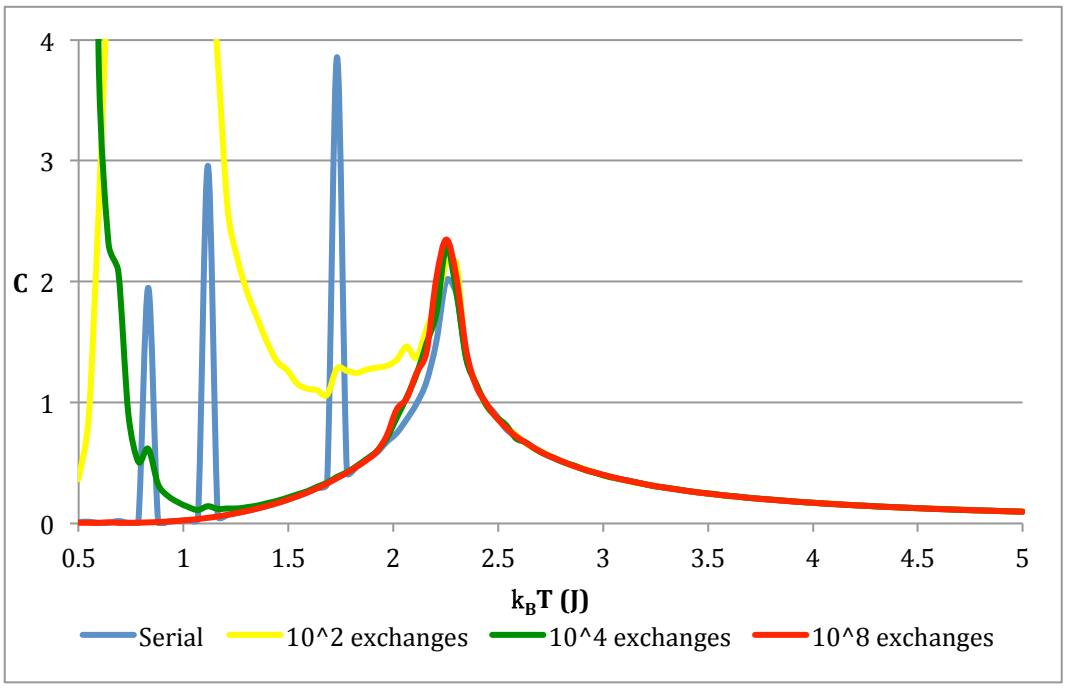

Figure 2.5: Heat capacity versus temperature at different number of exchanges

As shown in figures 2.2 and 2.3, when the number of exchanges increases, perfectly smooth curves is resulted and the convergence of first moments is well seen eventually.

In particular figure 2.2 shows a clear picture of effects of parallel tempering: as parallel tempering is introduced, the badly behaved systems perform better and the smoothness of the curve increases with the number of exchanges. However the well behaved systems are embroiled, because the energy minima are shared among and affect sampling for systems in the low-temperature range. As the number of exchanges is larger, the energy minima are transferred to systems in the high-temperature range more often, increasing the quality of sampling for those in the low-temperature range.

Figures 2.4 and 2.5 show that despite the second moments converge much better eventually with 10^8 exchanges, there are still slight instatbility. Moreover when an insufficient number of exchanges are introduced, the second moments are more chaotic, believed to be a joint result of low acceptance ratio at some systems, and samples are largely different before and after parallel tempering indeed works effectively. Some possible modifications are to increase the density of replica exchanges and the number of sampling steps. But these would dramatically increase the running time. For the purpose of having better convergence with limited computer resources, adjustments are to be made to the equilibration time to ensure the systems are fully equilibrated and earlier samples also follow the Boltzmann distribution.

2.5 Replica exchanges during equilibration

To make sure earlier samples are generated according to the Boltzmann distribution, replica exchanges are inserted into equilibration time as follows:

1) Generate an initial state randomly.
2) Go through a sampling time partitioned by exchange periods, during which within an exchange period:
   i. Go through the period of a certain amount of recursive sampling steps by:
      a) Choose a spin randomly and propose a flip on it.
      b) Accept the flip with a probability P_{flip}:
         Accept ➔ MC goes to the new state.
         Reject ➔ MC retains the original state.
   ii. At an exchange process:
      a) Propose a configuration exchange with the systems having the closest temperature alternatively.
      b) Accept the exchange with a probability $P_{exchange}$.
3) Go through a sampling time partitioned by exchange periods, during which within an exchange period:
   i. Go through the period of a certain amount of recursive sampling steps by:
      a) Choose a spin randomly and propose a flip on it.
      b) Accept the flip with a probability P_{flip}:
         Accept ➔ MC goes to the new state and a sample is taken.
         Reject ➔ MC retains the original state.
   ii. At an exchange process:
      a) Propose a configuration exchange with the systems having the closest temperature alternatively.
      b) Accept the exchange with a probability $P_{exchange}$.
4) Calculate the average quantities of interest.

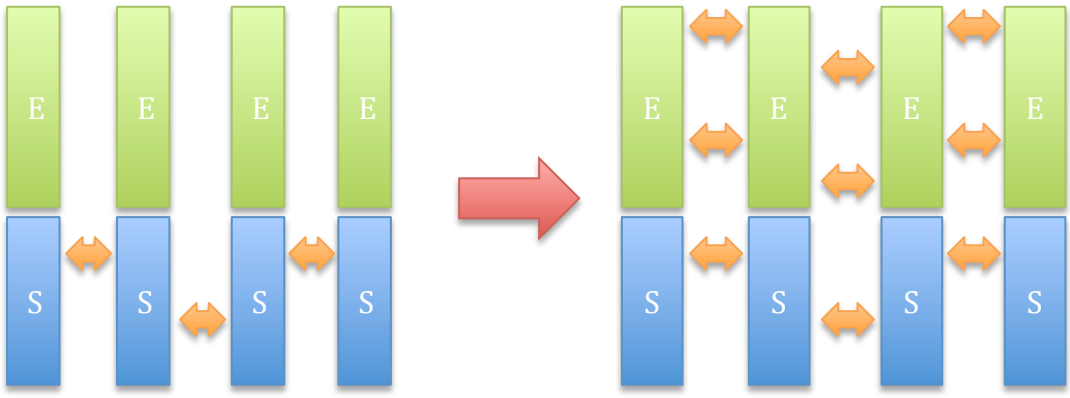

Figure 2.6: Replica exchanges are now inseted into both equilibration time and sampling time

2.6 Experiment III: parallel equilibration

The objective of this experiment is to compare the convergence of second moments of parallel tempering for Metropolis-Hastings algorthm with and without parallel equilibration. The set up is as follows:

Model	2D Ising	
Grid length	100	
Interaction strength J	1.00	
Boltzmann constant k_B	1.00	
Temperature pattern	Arithmetic	
Minimum temperature	0.50	
Maximum temperature	5.00	
# processors	96	
# equilibration steps between successive exchanges	10^8	10^4
# exchanges in equilibration time	0	10^4
# sampling steps between succssive exchanges	10^5	
# exchanges in sampling time	10^4	
Random seed	888	

Table 2.2: Set up for Experiment III

The total number of equilibration steps is fixed at 10^8, in which trials differ in whether evenly distributed replica exchanges are inserted to see the functionality. Results are as follows:

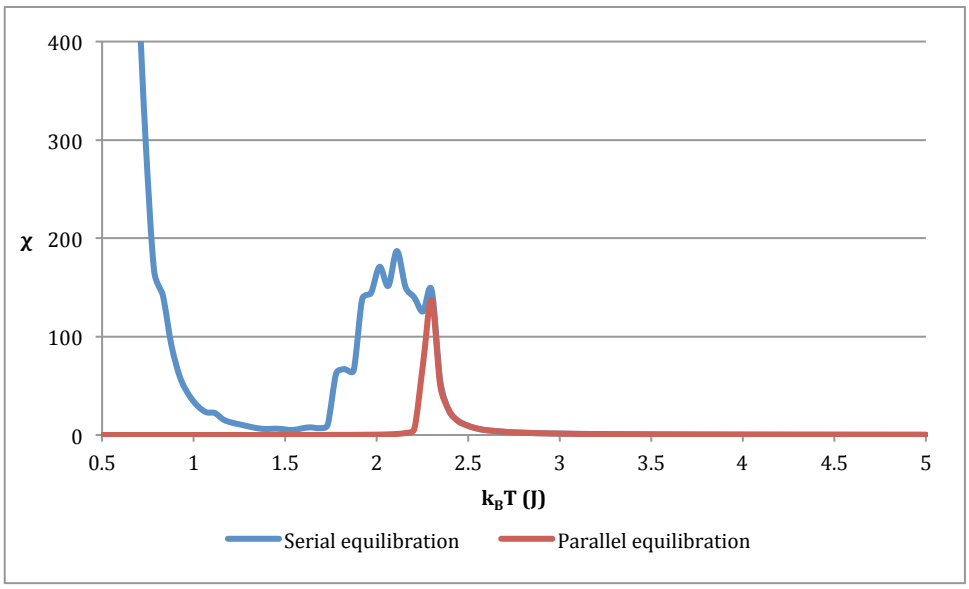

Figure 2.7: Magnetic susceptibility versus temperature with serial and parallel equilibration

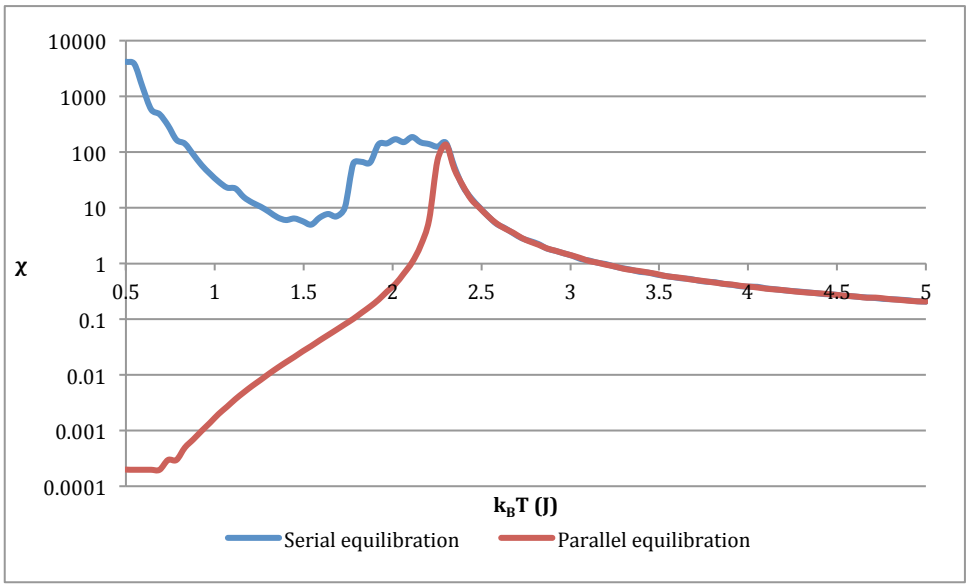

Figure 2.8: Magnetic susceptibility versus temperature with serial and parallel equilibration in logarithmic scale

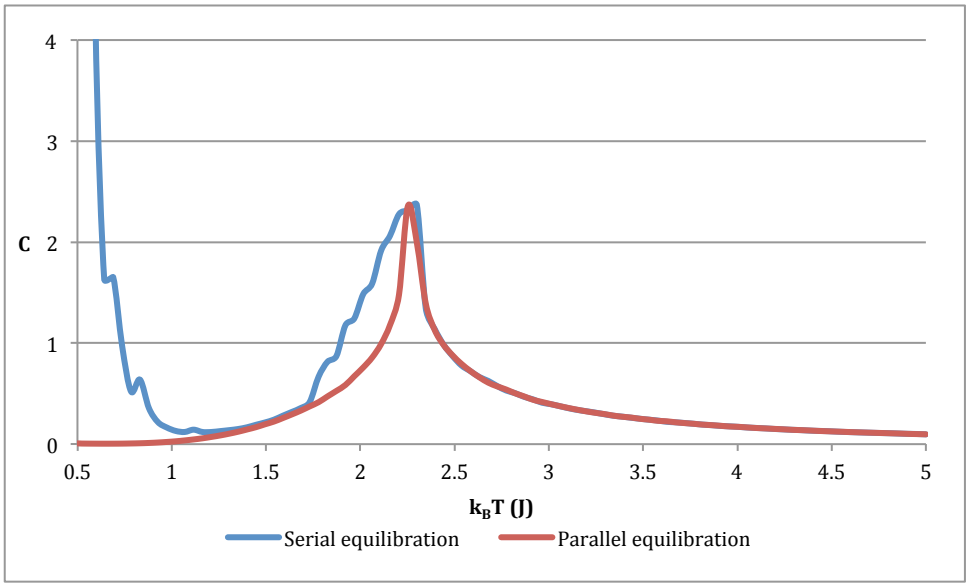

Figure 2.9: Heat capacity versus temperature with serial and parallel equilibration

As shown in figures 2.7 – 2.9, parallel equilibration gives a much better convergence in second moments than serial equilibration with same numbers of total equlibration steps and total sampling steps.

Compared with the results of the best converged ordinary parallel tempering case with 10^9 serial equilibration steps and 10^9 parallel sampling sleps in which 10^8 exchanges take place in Experiment II, the parallel equilibration case with smaller numbers of total exchanges and total steps in this experiment still converges considerably better and with a much shorter time.

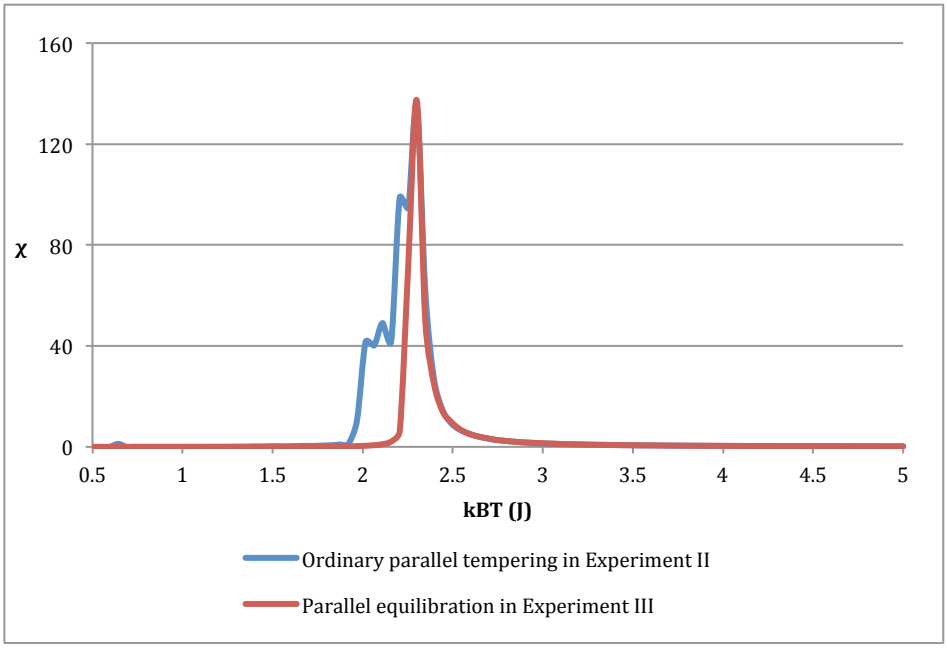

Figure 2.10: Magnetic susceptibility converges better in parallel equilibration than in ordinary parallel tempering with smaller numbers of total exchanges and total steps

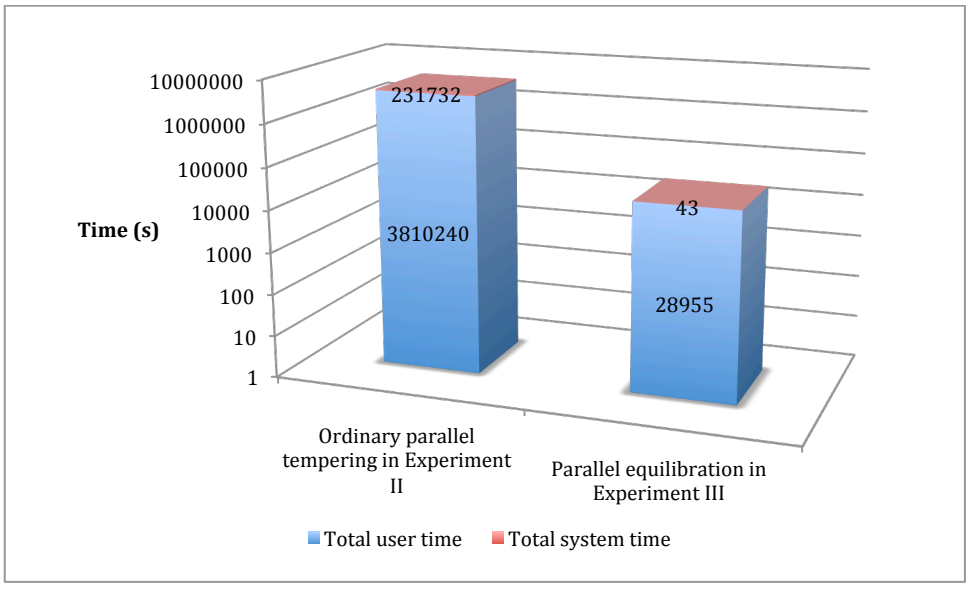

Figure 2.11: Required time is much shorter for parallel equilibration than ordinary parallel tempering to give a better convergence in second moments

Chapter 3

Analysis on Temperature Spacings

3.1 Optimal temperature pattern

As parallel tempering is introduced, there is an additional quantity of interest:

$$\text{Acceptance ratio for exchanges} \quad R_{exch} = \frac{\#\ accepted\ exchanges}{\#\ proposed\ exchanges}$$

This quantity is different for each pair of temperature system. Past researches have pointed out that parallel tempering is most efficient when the acceptance ratios for exchanges in all system pairs are equal in sampling time. It is observed that for each of the experiments on 2D Ising model carried out so far, the exchange ratios for system pairs around the phase transition are much lower (which will be seen not universally true for all models in Chapter 4).

This motivates us to find the optimal temperature pattern which gives a constant acceptance ratio for exchanges, allowing us to further reduce the computational cost especially for problems of large lattices which are in general more difficult to be equilibrated as well as exchanged.

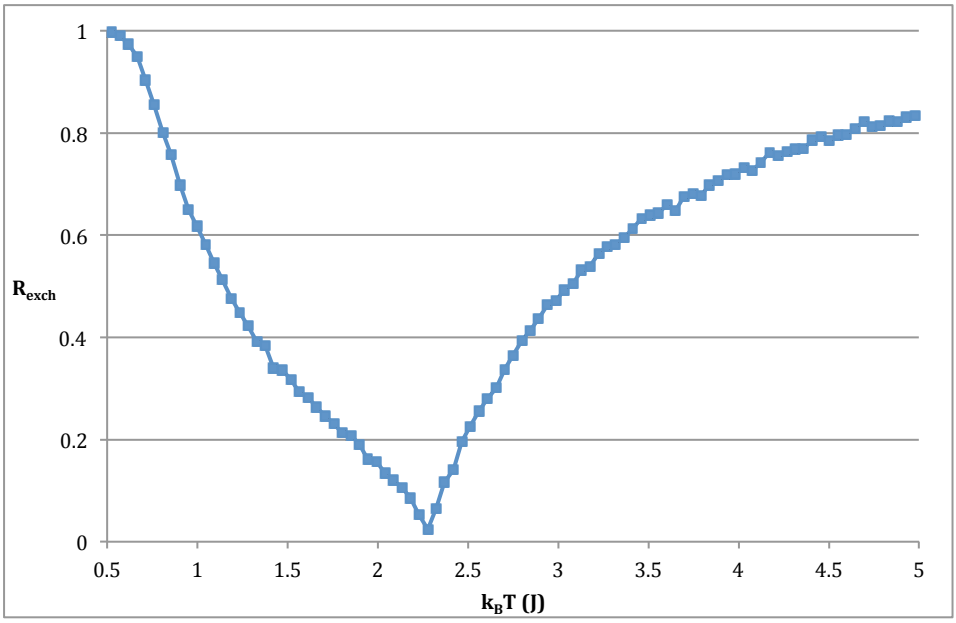

Figure 3.1: Acceptance ratios for exchanges versus temperature in parallel equilibration in Experiment III

3.2 Adaptive temperature spacing scheme

Considering the huge replica exchange difficulty throughout the temperature range, we are motivated to readjust temperature spacing adaptively according to the replica exchange acceptance ratio, setting a readjustment point after every certain amount of replica exchanges (i.e. after we obtain large enough sample of exchange acceptance ratio measurement).

The adaptive adjustment scheme goes as follows:

At every adjustment point, the density function is evaluated using

$$\varphi(T_{rank}) = \frac{1}{R_{acc}(T_{rank}) + C}$$

where $R_{acc}(T_{rank})$ is the exchange acceptance ratio of the temperature pair involving temperature with rank T_{rank} as the lower rank in the pair.

C is a constant specified by user, which will be later called *adjustment constant*. larger C leads to smoother density function, resulting in a less aggressive temperature spacing scheme.

After we have the density function at every temperature rank, we interpolate using piecewise constant function, which will be later used for determining new temperature points.

The initiative is to allow denser temperature points located at places where acceptance ratio is low (i.e. density function has high values). The new temperature spacing, therefore, is determined by finding T_{rank}^{new} such that

$$\int_0^{T_{rank}^{new}} \varphi(s)ds = \frac{T_{rank}}{N-1} \int_0^{N-1} \varphi(s)ds$$

Now that there must exist valid ranks i and such that $i < T_{rank}^{new} < i + 1$, we can readjust temperature by

$$T_{new} = T_i + (T_{rank}^{new} - i)(T_{i+1} - T_i)$$

This scheme has shown stability when the adaptive constant is chosen to be large enough, according to different models we run tests on.