2. Monte Carlo Simulations

in “equilibrium” statistical physics

Dong-Hee Kim

Gwangju Institute of Science and Technology

References

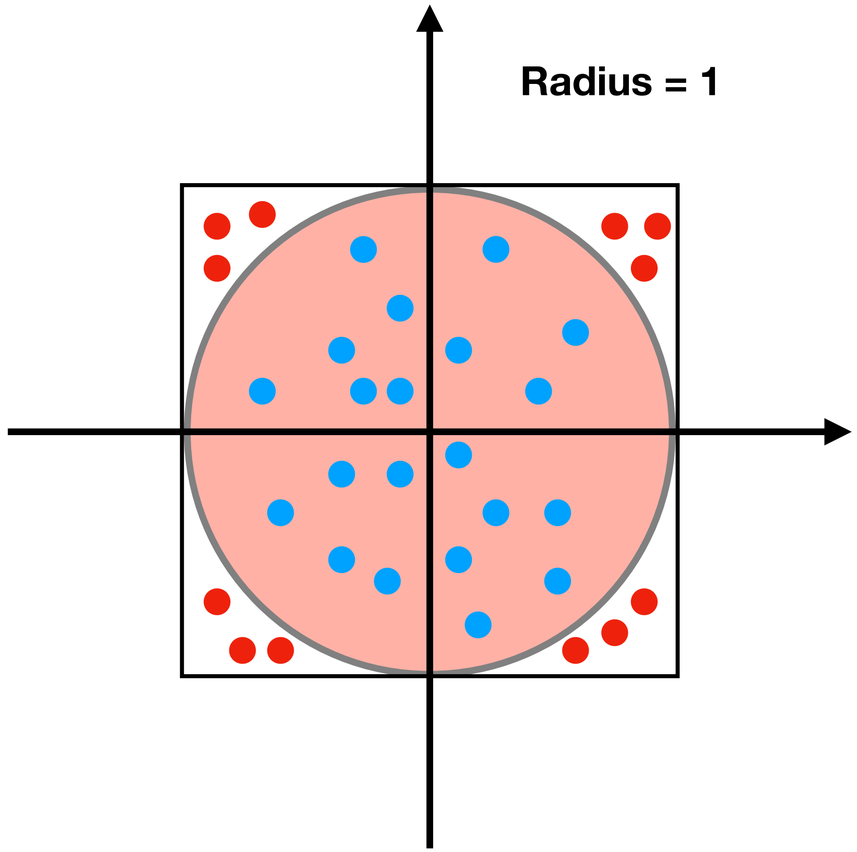
- M. E. J. Newman and G. T. Barkema, "Monte Carlo Methods in Statistical Physics" (Oxford University Press, 1999)
- L. Böttcher and H. J. Herrmann, "Computational Statistical Physics" (Cambridge University Press, 2021)
- W. Janke, in "Computational Physics" (Springer, 1996)
- Lectures by Mattias Troyer, Bernd Berg, Kari Rummukainnen

Outline

- Basic concepts in Markov Chain Monte Carlo
- Implementing a Monte Carlo code to simulate the Ising model
- Error estimation schemes
- Finite-size-scaling analysis of the second-order transitions
- Critical slowing down and cluster update algorithms
- Some other issues

2.1. Random numbers

Integration by rolling dice



Compute the area of the unit circle with random numbers.

1. Generate *two* random numbers from the uniform distribution between -1 and 1.
2. Use them as x and y coordinates.
3. See if (x,y) is inside of the circle.
4. Generate many such random points, and count how many are inside of the circle.
5. **Area** = **4** x **number of points found in the circle / total number of points**

Pseudo-random numbers

- “**P**seudo”-**R**andom **N**umber **G**enerators. That is what we’re looking for.
- There are many algorithms and implementations.
- There are notorious ones causing *systematic failures* in Monte Carlo simulations, such as **R250** and **RANDU**. (e.g. Ferrenberg *et al.* PRL 1992) See “examples of a *bad* generator” in H. G. Katzgraber, arXiv:1005.4117.
- There are many tests for “quality” check, but those do not guarantee that it is random enough. **R250** was a “good” PRNG as it passed all common quality tests at that time.

PRNG

- It is deterministic. It generates “a sequence” based on a given “seed”. The sequence is generated recursively, i.e. $x_i = f(x_{i-1}, x_{i-2}, \ldots, x_{i-k})$.
- It has a finite period.
- A simple example: a linear congruential generator, RAN0.

$$X_{i+1} = (16807\, X_i) \mod (2^{31} - 1)$$

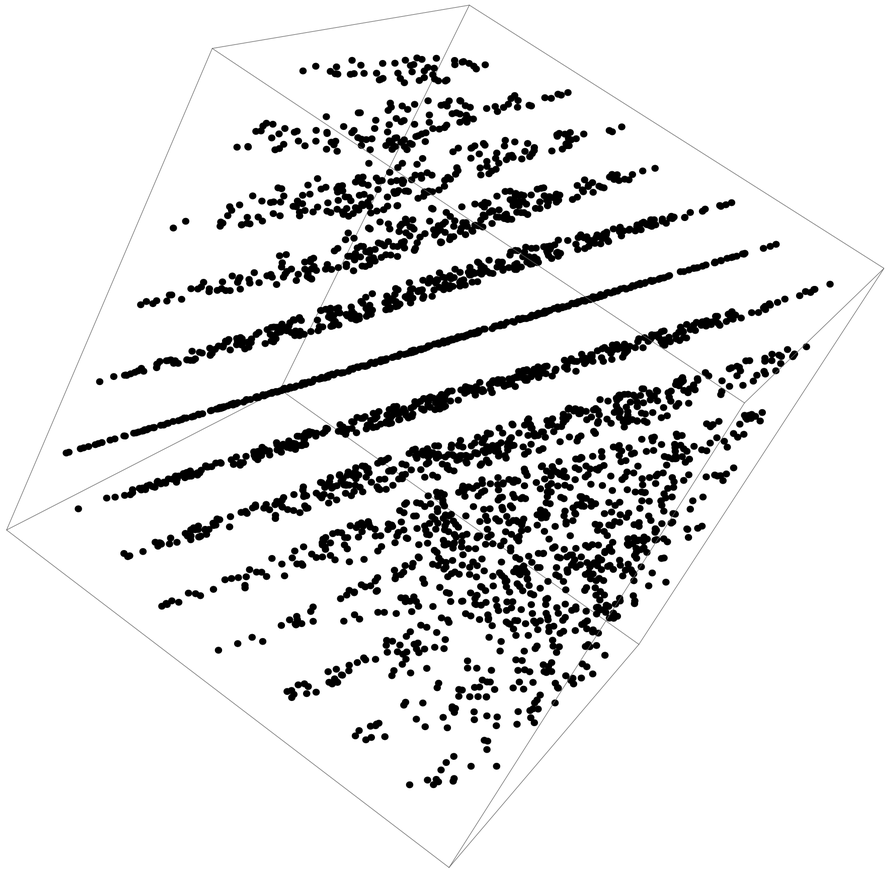
(period = 2^{31} - 1: very short)

- Good PRNG: good statistics (no correlation), long period, high performance, well-tested in your field of research.

A bad PRNG?

[From Helmut G. Katzgraber, arXiv:1005.4117]

Figure 1: 10^3 triplets of successive random numbers produced with RANDU plotted in three-dimensional space. If the random numbers were perfectly random, no planes should be visible. However, when viewed from the right angle, planes emerge, thus showing that the random numbers are strongly correlated.



Strategies for parallel computing

- Random seeding: each CPU has its own seed. *Hope for the best.*
- **Parametrization**: each CPU has different parameters for the same PRNG.
- **Block splitting**:

- **Leapfrogging**:

Popular PRNG

- **Merssene Twister** generator (MT19937, Matsumoto and Nishimura 1998)
  - Serial generator, available by default in C++11, python, etc.
  - Very fast, Very long period (2^{19937}-1), Very low correlation
- **SPRNG** (The Scalable Parallel Random Number Generators Library)
  - Parallel (parameterization), available at http://www.sprng.org/
- **TRNG** (Tina's random number generator library, fully C++)
  - Parallel (block splitting, leapfrogging), https://www.numbercrunch.de/trng/

C++ vs. Python

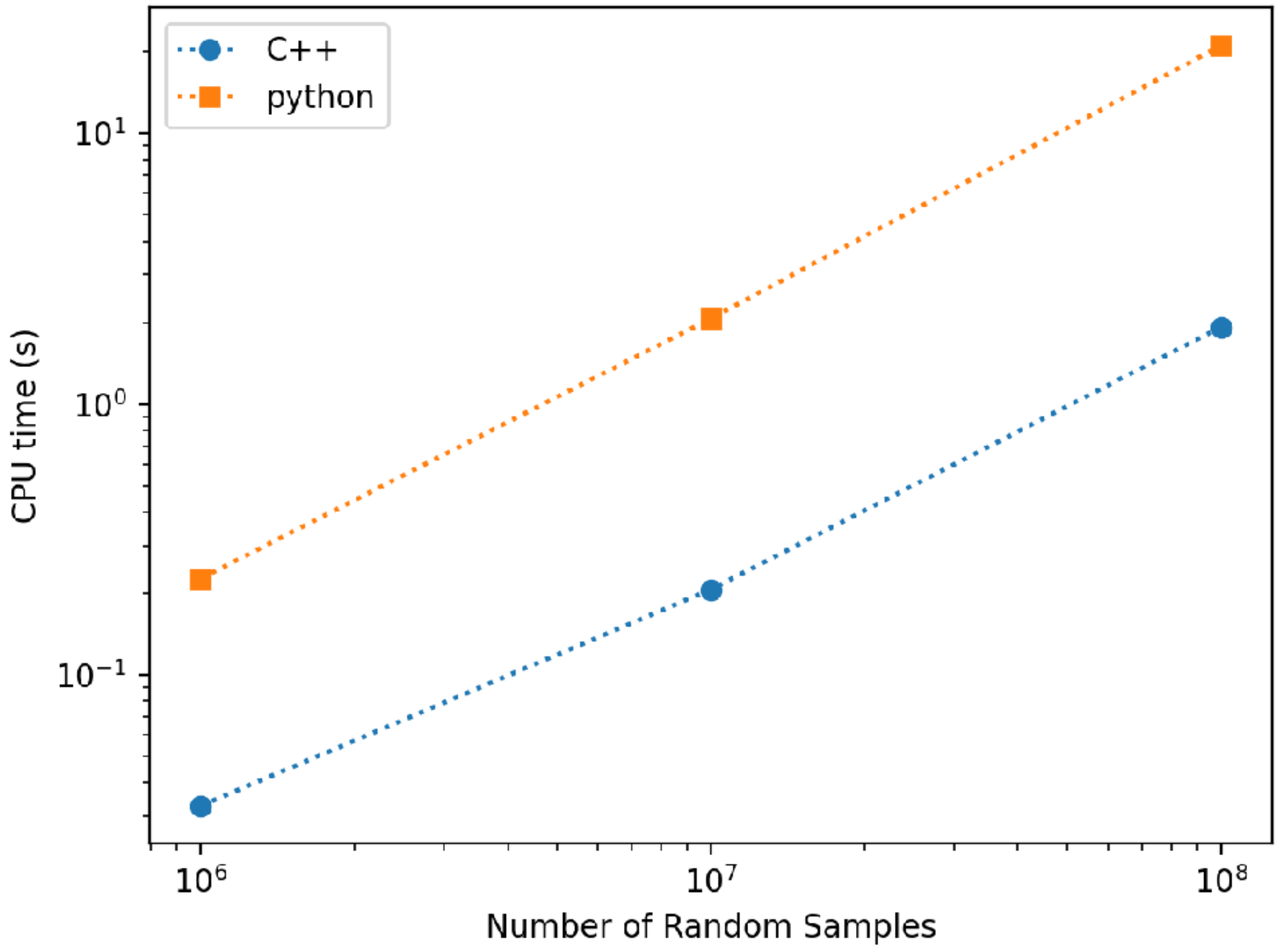
Computing π with random numbers:

```cpp
#include <iostream>
#include <random>

int main(int argc, char* argv[]) {

    const int N = atoi(argv[1]);

    std::random_device rd;
    std::mt19937 mt(rd());
    std::uniform_real_distribution<double> rng;

    int count = 0;
    for (int i = 0; i < N; ++i) {
        const double x = rng(mt);
        const double y = rng(mt);
        if (x*x + y*y < 1.0) count++;
    }

    std::cout << "AREA = "
              << 4.0 * count / N << std::endl;
}
```

```python
import random as rng
import sys

N = int(sys.argv[1])

count = 0

for i in range(N):
    x = rng.random()
    y = rng.random()
    if x**2 + y**2 < 1 :
        count = count + 1

print('AREA = %f' % (4.0*count/N))
```

```
>> ./pi.x [N]  # C++

>> python3 pi.py [N]  # python
```

C++ vs. Python



Computing π with random numbers:

C++ is 10 times faster than python.

Python's for-loop has a HUGE overhead.

Do not use python for serious Monte Carlo simulations.

Non-uniform distribution

- So far, we have been playing with a random number from a uniform distribution [0,1).
- What if you need a random number following an arbitrary distribution?
- Transformation method. e.g. Box-Muller transformation
- Rejection sampling method. (purely numerical)

Transformation

Change of variables in P.D.F

$$|p_x(x)\,dx| = |p_y(y)\,dy|$$

e.g. exponential distribution

$$p(x)\,dx = a\exp[-ax]\,dx$$
$$= d[e^{-ax}] = 1 \cdot du$$

uniform distribution

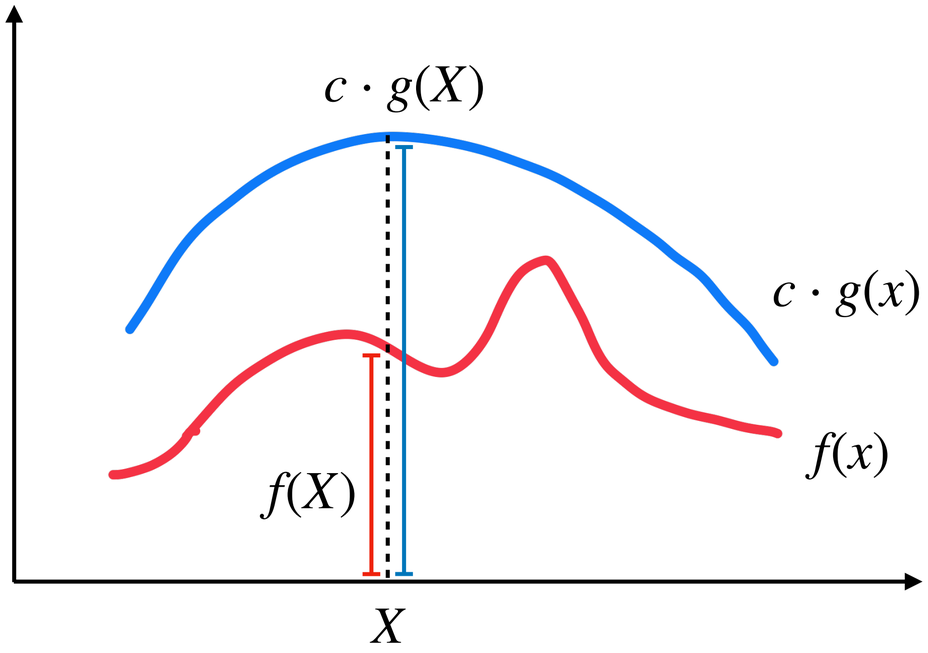
$$\rightarrow \quad x = -\frac{1}{a}\ln(1-u)$$

e.g. Box-Muller transformation for the Gaussian distribution

$$P(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) \quad \rightarrow \quad x = \sqrt{-2\ln(1-u)}\sin(2\pi v)$$

It does not work for all probability distribution! — This method requires an inverse function.

Rejection sampling



Conditions

1. *We know how to make a random sample from **g(x)**.*
2. *There exists a constant **c** such that **f(x) < c g(x)**.*

Algorithm

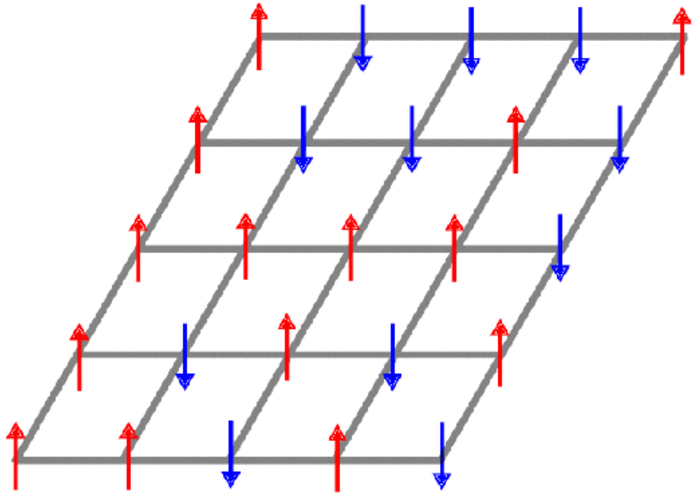
1. Draw a random number ***X*** from PDF ***g(x)***.
2. Accept X with a probability ***f(X) / [c g(X)]***;
   (1) Draw a uniform random number ***r***
   (2) If ***r < f(X) / [c g(X)]***, accept ***X***.
   (3) Reject, otherwise.

2.2. Markov Chain Monte Carlo

Ising model in two dimensions

Hamiltonian

$$E = -J \sum_{\langle i,j \rangle} s_i s_j - h \sum_i s_i \qquad h = 0$$



Canonical ensemble

$$p(\{s_1, s_2, \ldots, s_N\}) = \frac{1}{Z} \exp[-\beta E(\{s_1, s_2, \ldots, s_N\})]$$

Measurement

$$\langle O \rangle = \sum_{s_1=\pm 1} \sum_{s_2=\pm 1} \cdots \sum_{s_N=\pm 1} O(\{s_1, s_2, \ldots, s_N\})\, p(\{s_1, s_2, \ldots, s_N\}) \rightarrow \langle O \rangle_{\mathrm{MC}}$$

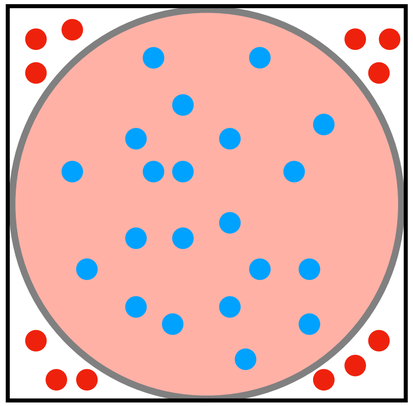
What are the hurdles?

Compute this: $$\langle O \rangle = \sum_{s_1=\pm 1} \sum_{s_2=\pm 1} \cdots \sum_{s_N=\pm 1} O(\{s_1, s_2, \ldots, s_N\})\, p(\{s_1, s_2, \ldots, s_N\})$$

where $$p(\{s_1, s_2, \ldots, s_N\}) = \frac{1}{Z} \exp[-\beta E(\{s_1, s_2, \ldots, s_N\})]$$

Number of microstates = $\mathbf{2^N}$: IT'S IMPOSSIBLY LARGE!

What about just generating a random spin sequence and computing a sum like we did before:



Importance sampling

- Uniform generation of random spin sequence would not solve the issue:
  The uniform sampling does not change the system size scaling $\sim 2^N$.

- Boltzmann weight is exponential: central region contributes much more than the tail part. Can we just sample the Boltzmann distribution directly?

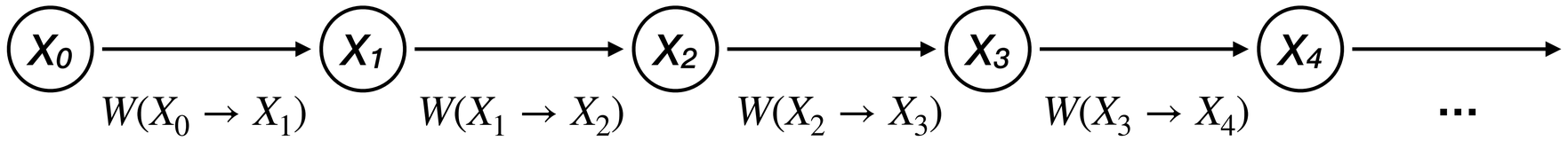
- **Importance sampling** (this is what “Monte Carlo” really means in practice)
  — a strategy to visit “important” states more often.

e.g. importance sampling of the canonical ensemble

$$\langle O \rangle = \frac{1}{Z} \sum_{\{s_i\}} O \exp(-\beta E) \approx \frac{1}{N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} O_i \equiv \langle O \rangle_{\mathrm{MC}}$$

Markov Chain Monte Carlo

Start from a state X and propose a new state Y with a **selection probability** $g(X \to Y)$.
Accept or reject the proposed state with an **acceptance probability** $A(X \to Y)$.

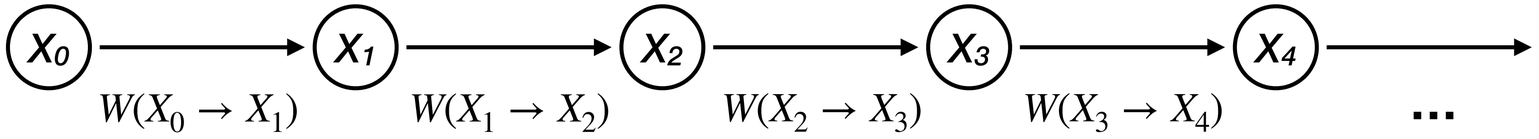


Transition probability $W(X \to Y) = g(X \to Y)\, A(X \to Y)$

$$\langle O \rangle_{\mathrm{MC}} = \frac{1}{N_{\mathrm{MC}}} \left[O(X_{t_o}) + O(X_{t_o+1}) + O(X_{t_o+2}) + O(X_{t_o+3}) + \cdots + O(X_{t_o+N_{\mathrm{MC}}}) \right]$$

Markov Chain Monte Carlo

- Master equation



$$\frac{dp(X,t)}{d\tau} = \sum_{Y \neq X} p(Y)W(Y \to X) - \sum_{Y \neq X} p(X)W(X \to Y)$$

Equilibrium → 0

$$p(X) = p^{\mathrm{eq}}(X)$$

equilibrium distribution

$$p^{\mathrm{eq}}(X) \propto \exp[-\beta E(X)]$$

- Ergodicity: there exists n such that $W^{(n)}(X \to Y) > 0$.
- Normalization: $\sum_Y W(X \to Y) = 1$
- Homogeneity: $\sum_Y p(Y)W(Y \to X) = p(X)$

Detailed Balance Condition

Balance Condition [stationary p(X)]

$$\sum_{Y \neq X} p(Y) W(Y \to X) = \sum_{Y \neq X} p(X) W(X \to Y)$$

"Equilibrium"

$$p(X) \to p^{\mathrm{eq}}(X)$$

Detailed Balance Condition (sufficient condition)

$$p^{\mathrm{eq}}(Y) W(Y \to X) = p^{\mathrm{eq}}(X) W(X \to Y)$$

→ **"Reversibility"**

"No net stochastic flux"

$$W(X \to Y)\, W(Y \to Z)\, W(Z \to X)$$
$$= W(X \to Z)\, W(Z \to Y)\, W(Y \to X)$$

c.f. "irreversible" MC

Canonical ensemble

$$\frac{W(X \to Y)}{W(Y \to X)} = \frac{p^{\mathrm{eq}}(Y)}{p^{\mathrm{eq}}(X)} = \exp[-\beta(E_Y - E_X)]$$

Metropolis algorithm

M(RT)² algorithm [N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller (1953)]

Canonical ensemble

$$\frac{W(X \to Y)}{W(Y \to X)} = \frac{p^{\mathrm{eq}}(Y)}{p^{\mathrm{eq}}(X)} = \exp[-\beta(E_Y - E_X)]$$

You can propose any A that satisfies this equation!

$$\frac{W(X \to Y)}{W(Y \to X)} = \frac{g(X \to Y)\,A(X \to Y)}{g(Y \to X)\,A(Y \to X)} = \frac{A(X \to Y)}{A(Y \to X)} = e^{-\beta(E_Y - E_X)}$$

symmetric

The choice of M(RT)² :

$$A(X \to Y) = \min\left[1, \exp[-\beta(E_Y - E_X)]\right]$$

Choices of Acceptance Probability

You can propose any A that satisfies this equation:

$$\frac{A(X \to Y)}{A(Y \to X)} = e^{-\beta(E_Y - E_X)}$$

M(RT)² :

$$A(X \to Y) = \min\left[1, \exp(-\beta \Delta E)\right]$$

Heat-Bath (Glauber) :

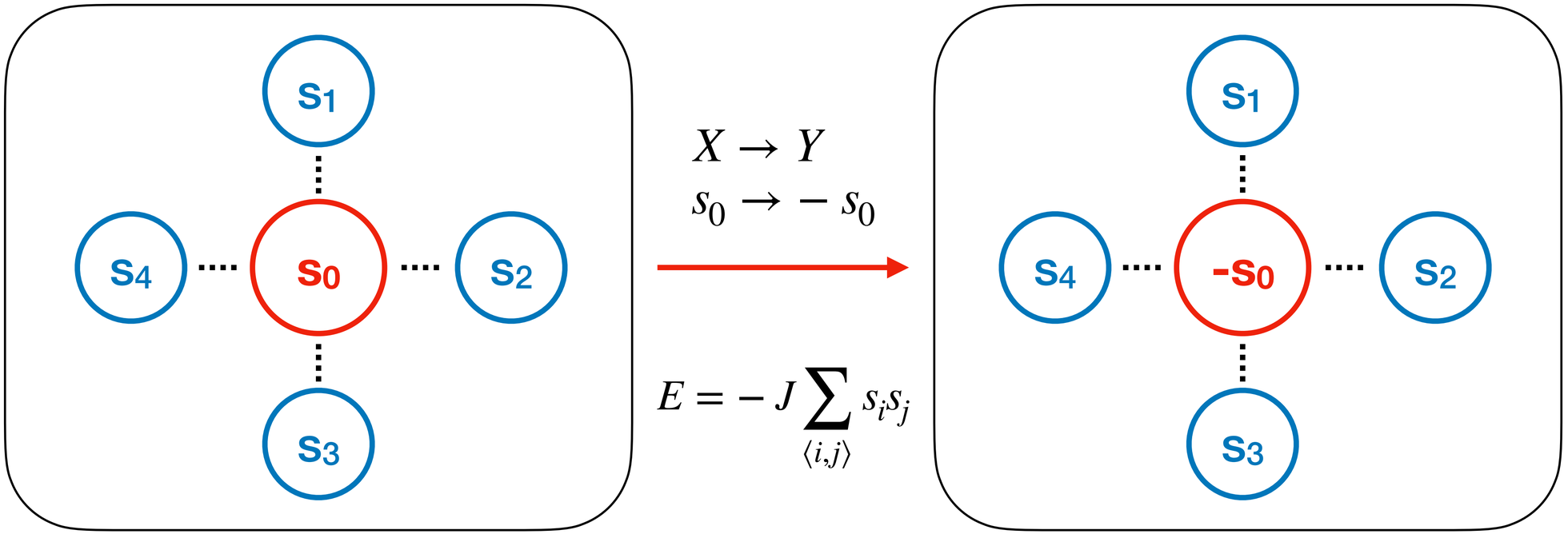
$$A(X \to Y) = \frac{\exp(-\beta \Delta E)}{1 + \exp(-\beta \Delta E)}$$

2.3. Simulating the Ising model

Simulating Ising model

local update with a single-spin flip

$$\Delta E \equiv E_Y - E_X = [-J(-s_0)(s_1 + s_2 + s_3 + s_4)] - [-J(s_0)(s_1 + s_2 + s_3 + s_4)] = \boxed{2Js_0 \sum_{k \in \text{n.n.}} s_k}$$



Common update strategies

How do you <u>sweep</u> the lattices to update spins?

1. RANDOM

Visit lattices sites randomly. Slow.

2. SEQUENTIAL

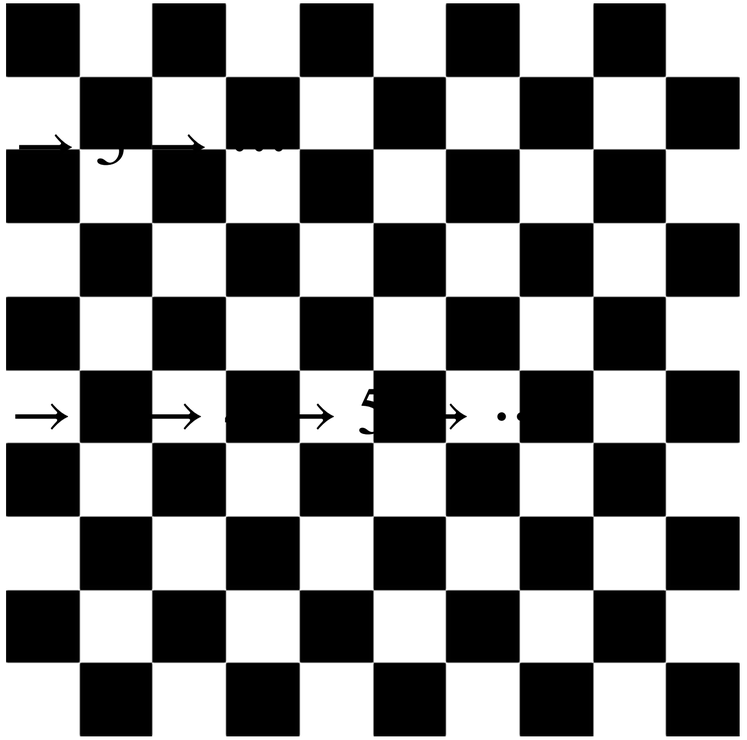
Visit every site sequentially. Weak detailed balance.

3. SUBLATTICE (CHECKERBOARD)

Visit every site in sublattices sequentially.
Weak detailed balance. Some advantages in vectorization.

$3 \to 1 \to 10 \to 14 \to$

$1 \to 2 \to$



One Monte Carlo Step (MCS)

A typical choice of unit time in Monte Carlo simulations

= One full sweep of the lattice sites

```
for mcs in range(N_mcs) :
    for i in range(L) :
        for j in range(L) :
            deltaE = (compute ΔE by s_{i,j} → − s_{i,j})
            if deltaE < 0 :
                (accept the spin flip)
            else if exp(−beta∗deltaE) < random() :
                (accept the spin flip)
```

one move

one sweep, one MCS

Convergence of MCMC

A physicist-friendly argument: O. Narayan and A. P. Young, PRE 64, 021104 (2001)

def.

$$G(t) = \sum_l \frac{1}{p_l^{\mathrm{eq}}} \left(p_l(t) - p_l^{\mathrm{eq}} \right)^2 = \sum_l \frac{[p_l(t)]^2}{p_l^{\mathrm{eq}}} - 1 \quad \leq 0$$

master equation (unit step = one move)

$$p_l(t+1) = \sum_m p_m(t)\, W(m \to l) \quad \longleftarrow \quad p_l(t+1) - p_l(t) = \sum_{m \neq l} \left[p_m(t)\, W(m \to l) - p_l(t)\, W(l \to m) \right]$$

detailed balance

$$p_l^{\mathrm{eq}}\, W(l \to m) = p_m^{\mathrm{eq}}\, W(m \to l)$$

Convergence of MCMC

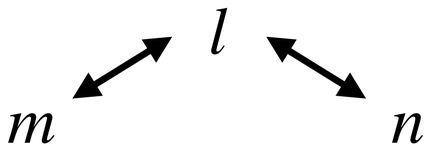
master equation

$$p_l(t+1) = \sum_m p_m(t)\, W(m \to l)$$

detailed balance

$$p_l^{\mathrm{eq}}\, W(l \to m) = p_m^{\mathrm{eq}}\, W(m \to l)$$

$$\Delta G = G(t+1) - G(t) = -\frac{1}{2} \sum_{l,m,n} W(l \to m)\, W(l \to n)\, p_l^{\mathrm{eq}} \left(\frac{p_m(t)}{p_m^{\mathrm{eq}}} - \frac{p_n(t)}{p_n^{\mathrm{eq}}} \right)^2 \leq 0$$



should appear at an arbitrary large time

+ ergodicity

$$p(X) \to p^{\mathrm{eq}}(X)$$

O. Narayan and A. P. Young, PRE 64, 021104 (2001)

Weak Detailed Balance

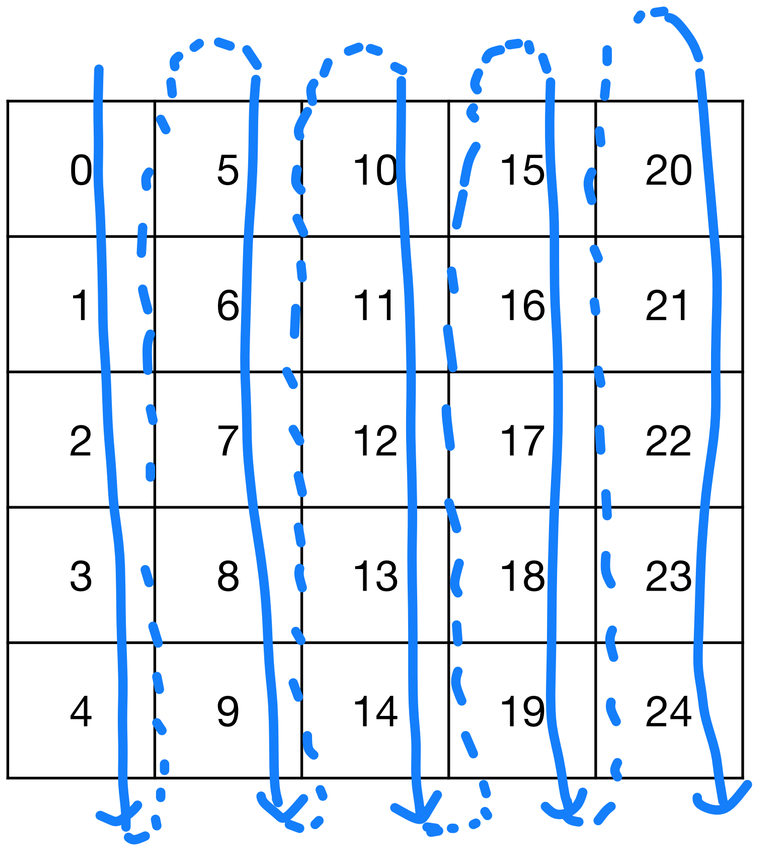
master equation $$p_l(t+1) = \sum_m p_m(t)\, W(m \to l)$$

sequential update

$$\mathbf{p}(t_{\mathrm{mcs}}+1) = \mathbf{W}_{\mathrm{sweep}}\,\mathbf{p}(t_{\mathrm{mcs}}) = \mathbf{W}_N \mathbf{W}_{N-1} \cdots \mathbf{W}_1 \mathbf{W}_0\, \mathbf{p}(t_{\mathrm{mcs}})$$

$\mathbf{W}_i$: detailed balance is valid.

$\mathbf{W}_{\mathrm{sweep}}$: detailed balance is broken.



Thus, the inequality still works in every Monte Carlo "move".

$$\Delta G = G(t+1) - G(t) = -\frac{1}{2} \sum_{l,m,n} W(l \to m)\, W(l \to n)\, p_l^{\mathrm{eq}} \left(\frac{p_m(t)}{p_m^{\mathrm{eq}}} - \frac{p_n(t)}{p_n^{\mathrm{eq}}} \right)^2 \leq 0$$

A typical structure of an MC code

0. Initialization

Ordered initial state or *disordered* (*random*) initial state?

1. Thermalization

```
sweep()
```

Allow *relaxation* time for a spin configuration to be thermalized.

Ideally, we want $p(X) \rightarrow p^{\mathrm{eq}}(X)$.

2. Measurement

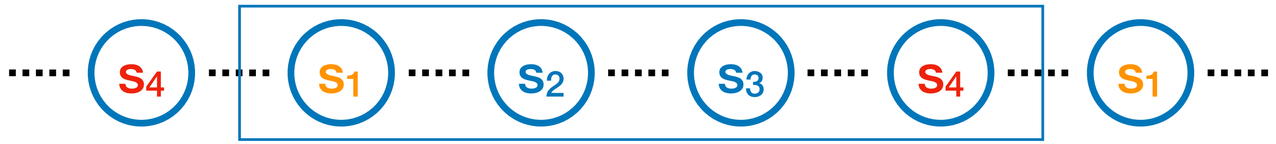
```
sweep()
measure()
```

This is a production run. Measure what you want every ***m*** MCS.

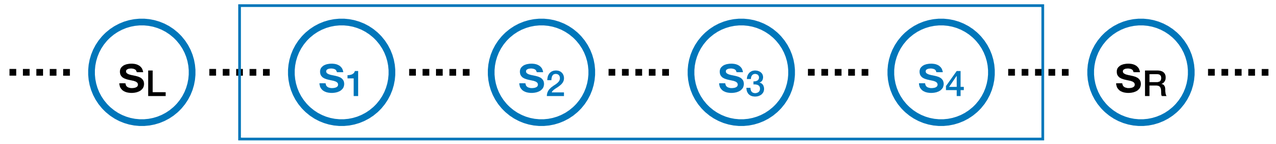
(***m = 1*** is just fine.)

Boundary Conditions

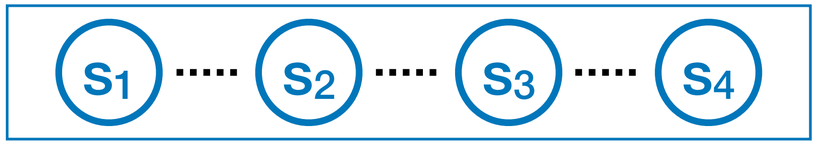
- Periodic Boundary Condition



- Fixed Boundary Condition



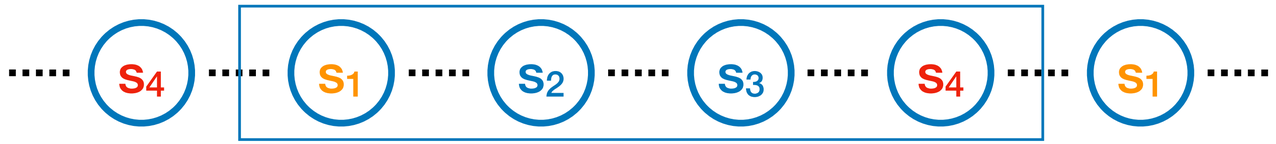
- Open Boundary Condition



- Helical Boundary Condition

Boundary Conditions

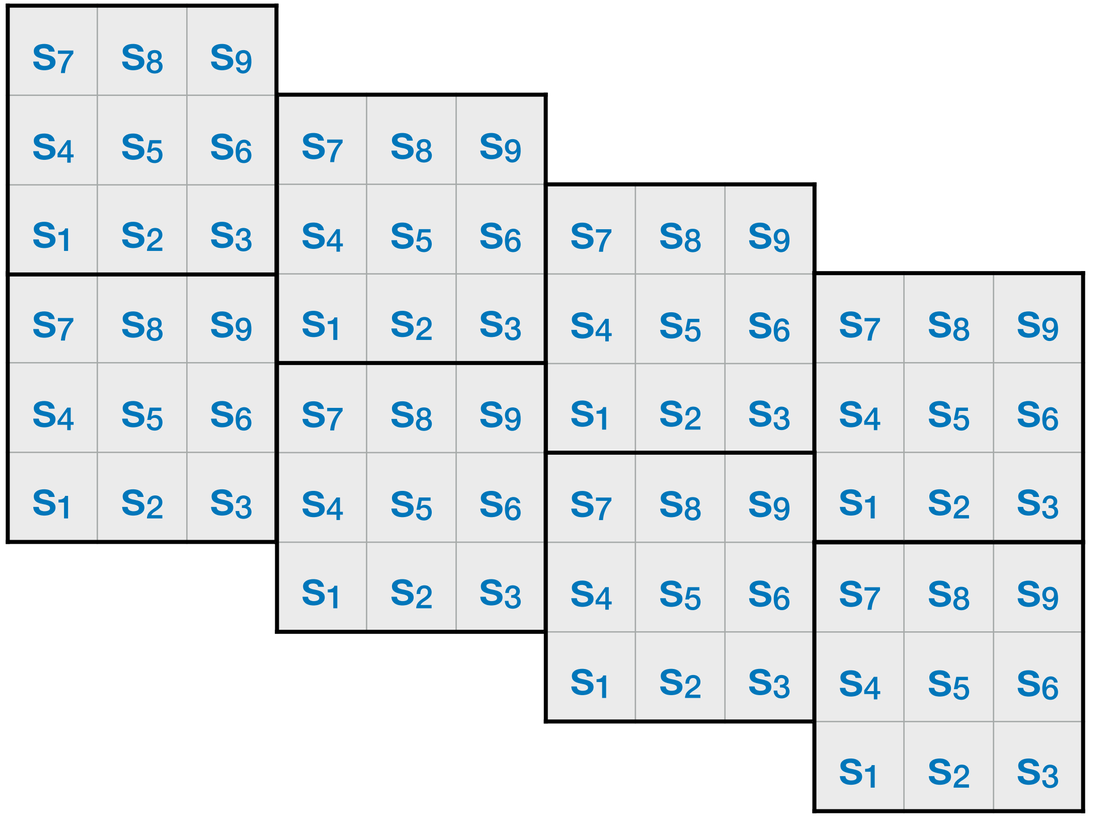
- **Periodic Boundary Condition**



- Fixed Boundary Condition

- Open Boundary Condition

- **Helical Boundary Condition**



Measurements

- magnetization (order parameter) $\langle m \rangle = \frac{1}{N} \left\langle \left| \sum_{i=1}^{N} s_i \right| \right\rangle$
- magnetic susceptibility $\chi = \beta N \left(\langle m^2 \rangle - \langle m \rangle^2 \right)$
- specific heat $c = \frac{\beta^2}{N} \left(\langle E^2 \rangle - \langle E \rangle^2 \right)$
- Binder's (fourth-order) cumulant $U_4 = 1 - \frac{\langle m^4 \rangle}{3 \langle m^2 \rangle^2}$
- correlation function $G_c^{(2)}(i,j) = \langle s_i s_j \rangle - \langle s_i \rangle \langle s_j \rangle$

Python Implementation [M(RT)^2]

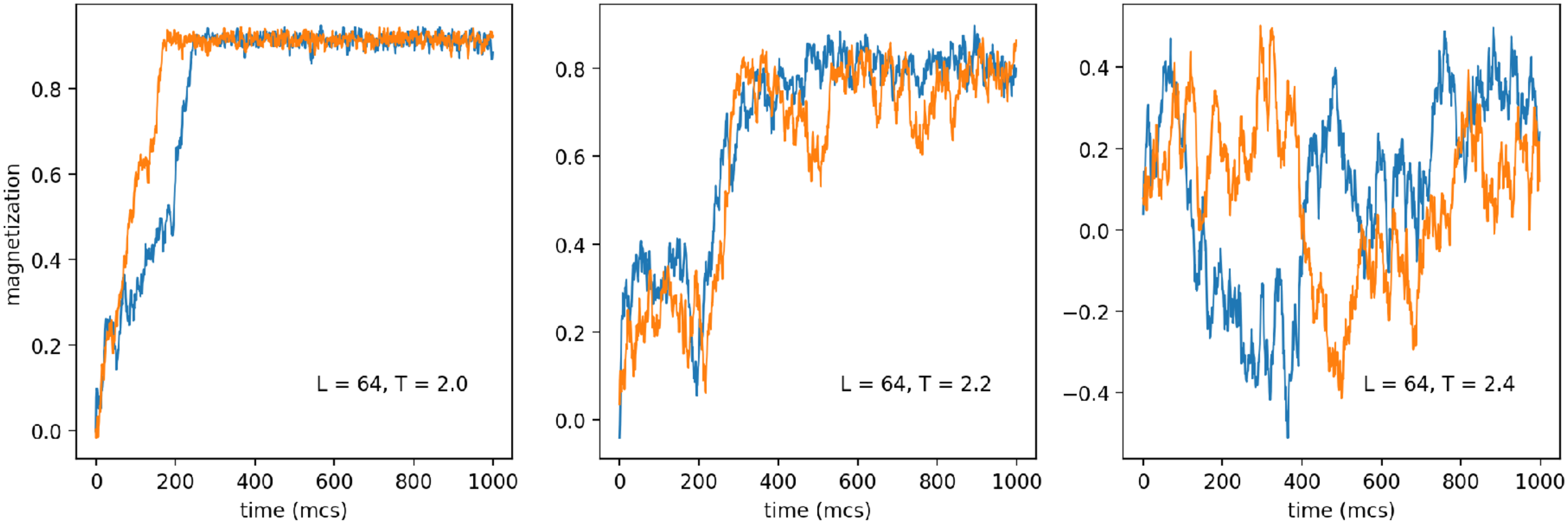
IT'S VERY SLOW. NOT RECOMMENDED. BUT, ...

```
import numpy as np
import matplotlib.pyplot as plt

L = 64              # L x L lattices
T = 2.0             # temperature

mcs_max = 1000

N = L * L           # total number of site

# index of nearest neighbor spins

neighbor = [ [ (i + 1) % L + (i // L) * L,
               (i - 1) % L + (i // L) * L,
               (i + L) % N,
               (i - L) % N
             ] for i in range(N) ]
```

```
# initial state : random
spin = 2 * np.random.randint(2, size = N) - 1
mag = np.sum(spin)
magarr = np.zeros(mcs_max)

# M(RT)^2 single spin update

pc = np.exp(-2.0 / T * np.arange(5))

for mcs in range(mcs_max) :
    for i in range(N) :
        snn = spin[i] * np.sum(spin[neighbor[i]])
        if snn <= 0 :
            mag = mag - 2 * spin[i]
            spin[i] = -spin[i]
        elif pc[snn] > np.random.random() :
            mag = mag - 2 * spin[i]
            spin[i] = -spin[i]
    magarr[mcs] = mag / N
```

See what happens ...

In fact, MC measurements are correlated, often very much.



2.4. Analyzing Monte Carlo data

Equilibration time

- Monte Carlo measurements should be started after the Markov chain provides a distribution sufficiently close to the asymptotic distribution.

- The number of equilibration (thermalization) steps should be several (>10) times of an autocorrelation time measured from the very beginning.

$$\tau_{\mathrm{eq}} \gg \frac{\sum_{t=1}^{\infty} \left(\langle A_0 A_t \rangle - \langle A \rangle^2 \right)}{\langle A_0^2 \rangle - \langle A \rangle^2}$$

- This is one of the bottleneck for parallelization with massive independent MC running across many CPUs. Every cpu needs its own thermalization.

Autocorrelation time

- A measure of how long it takes from one state to a significantly different another state. *Ideally*, it should be a time interval between different measurements, which is practically not possible.

- Usually the equilibration time is much longer than the autocorrelation time.

- Autocorrelation function

$$C(t) = \langle A(t_i)A(t_i + t)\rangle - \langle A\rangle^2 \propto \exp(-t/\tau_{\text{exp}})$$

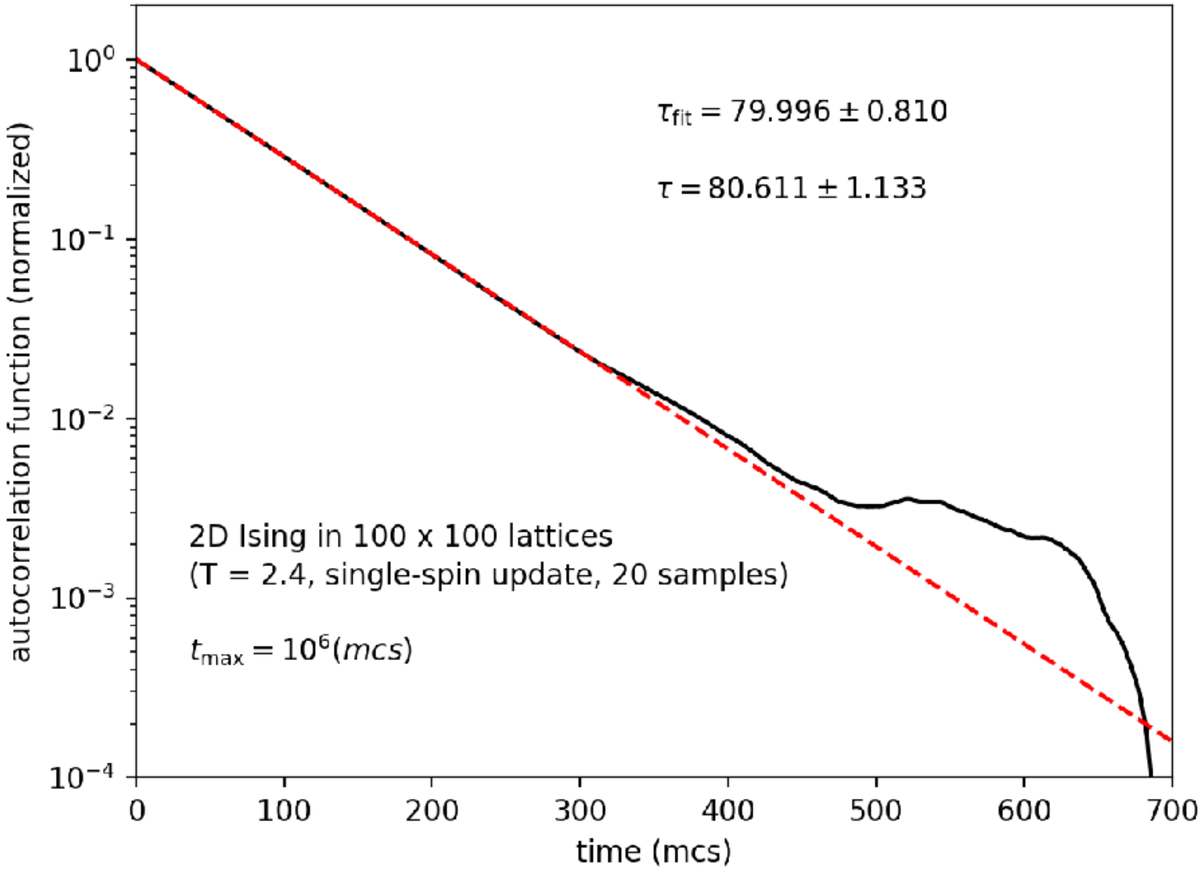
- Integrated autocorrelation time

$$\tau_{\text{int}} = 1 + 2\sum_{t=1}^{\infty}\frac{C(t)}{C(0)} \approx 2\int_0^{\infty}\frac{C(t)}{C(0)}dt = 2\tau_{\text{exp}}$$

Exponentially decaying autocorrelation function

$$C(t) = \langle A(t'+t)A(t')\rangle - \langle A\rangle^2 \propto \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} A(t'+t)A(t') - \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} A(t'+t)\cdot\frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t} A(t')$$

(Newman and Barkema)



Trouble happens when *t* gets 'close' to t_{max} :

- *very large statistical fluctuations*
- *poor accuracy at its tail*
- *Thus, you have to run it for a very long time (as long as many correlation times)!*
- Left figure: t_{max} = 1,000,000 mcs.

Error estimate of uncorrelated measurements

You have done M *uncorrelated* measurements on A : $\{A_1, A_2, \ldots, A_M\}$. → sample mean $\bar{A} = \frac{1}{M}\sum_{i=0}^{M} A_i$.

$$\langle \bar{A} \rangle = \frac{1}{M}\sum_i \langle A_i \rangle = \langle A \rangle$$ best estimate of a sample mean = true expectation value

Standard error : standard deviation of a *true* estimate

$$\sigma_A^2 = \langle A^2 \rangle - \langle A \rangle^2$$ → ***We do not know it.***

Our error estimate : deviation of $\bar{A}$ from $\langle A \rangle$

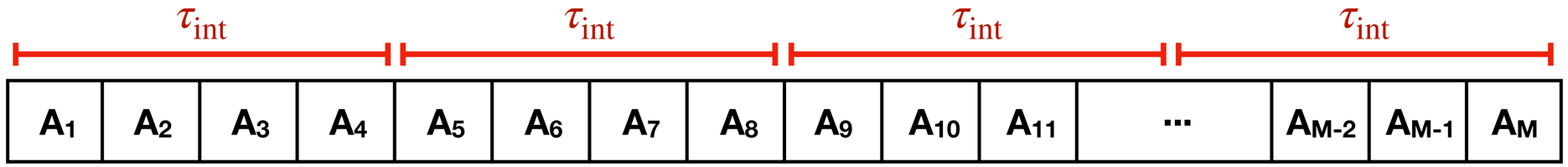
We know it for the measurements!

$$\Delta_A^2 = \left\langle \left(\bar{A} - \langle A \rangle\right)^2 \right\rangle = \frac{\sigma_A^2}{M} \approx \frac{s_A^2}{M-1}$$

$$s_A^2 = \overline{(A^2)} - (\bar{A})^2 = \frac{1}{M}\sum_i A_i^2 - \left(\frac{1}{M}\sum_i A_i\right)^2$$

$$\langle s_A^2 \rangle = \frac{M-1}{M}\left[\langle A^2 \rangle - \langle A \rangle^2\right] = \frac{M-1}{M}\sigma_A^2$$

Binning analysis



Error estimate for correlated samples:

$$\Delta_A^2 \equiv \left\langle (\bar{A} - \langle A \rangle)^2 \right\rangle = \frac{\sigma_A^2}{M} + \frac{1}{M^2} \sum_{i \neq j} \left(\langle A_i A_j \rangle - \langle A \rangle^2 \right)$$

Assumption:
correlation decay rapidly as $|i - j| \to \infty$.

$$(\text{second term}) = \frac{2}{M^2} \sum_{i=1}^{M} \sum_{t=1}^{M-i} \left(\langle A_i A_{i+t} \rangle - \langle A \rangle^2 \right) \approx \frac{2}{M} \sum_{t=1}^{\infty} \left(\langle A_1 A_{1+t} \rangle - \langle A \rangle^2 \right)$$

$$\Delta_A = \sigma_A \sqrt{\frac{1 + 2\tau_A}{M}}$$

← ***corrected error***

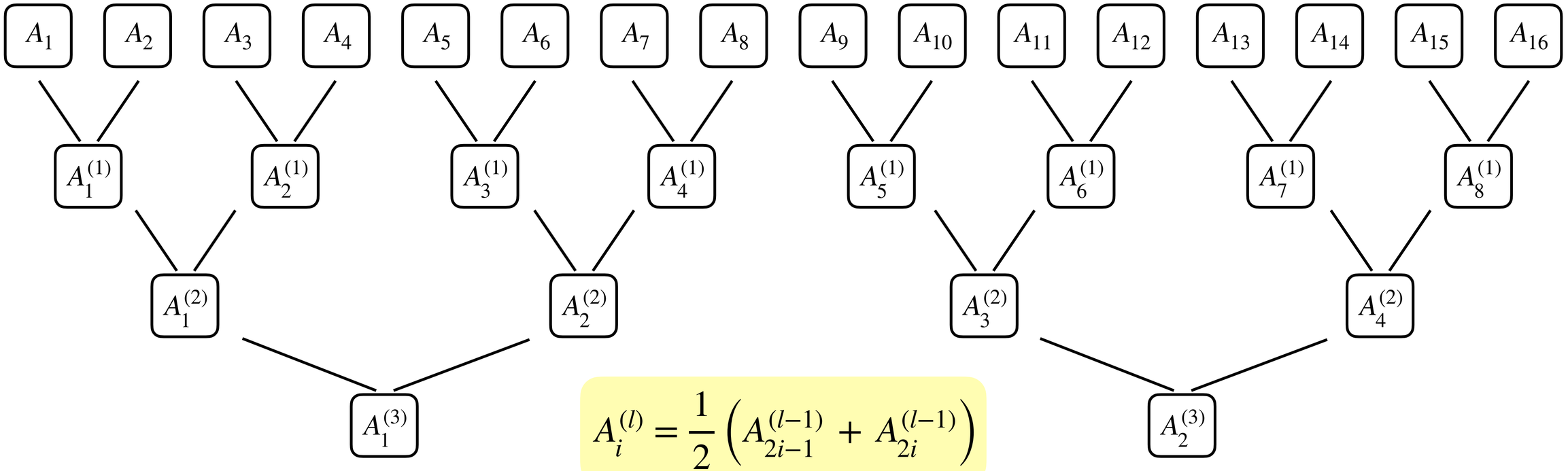
$$\frac{2\sigma_A^2}{M} \tau_A$$

V. Ambegaokar and M. Troyer, Am. J. Phys. 78, 150 (2010)

Binning analysis

V. Ambegaokar and M. Troyer, Am. J. Phys. 78, 150 (2010) [arXiv:0906.0943]

Measuring error estimate and integrated autocorrelation time at once.



$$A_i^{(l)} = \frac{1}{2}\left(A_{2i-1}^{(l-1)} + A_{2i}^{(l-1)}\right)$$

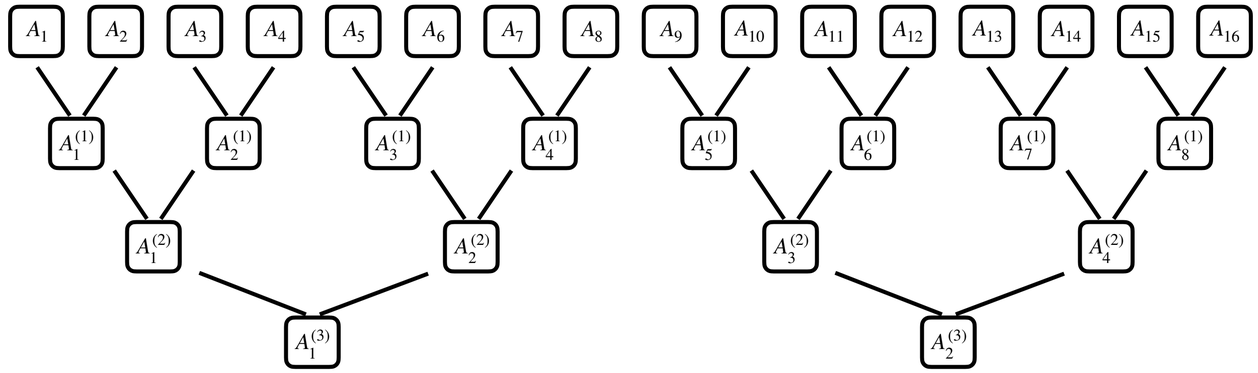
Less correlated as it proceeds to the next layer.

Binning analysis

V. Ambegaokar and M. Troyer, Am. J. Phys. 78, 150 (2010) [arXiv:0906.0943]

$$A_i^{(l)} = \frac{1}{2}\left(A_{2i-1}^{(l-1)} + A_{2i}^{(l-1)}\right)$$

$$i = 1,\ldots,M_l \equiv M/2^l$$



Error estimate at the *l*-th layer if uncorrelated

$$\Delta_A^{(l)} = \left[\frac{1}{M_l(M_l-1)}\sum_{i=1}^{M_l}\left(A_i^{(l)} - \overline{A^{(l)}}\right)^2\right]^{1/2} \qquad s_A^2$$

$$l \to \infty \longrightarrow \Delta_A = \sigma_A\sqrt{\frac{1+2\tau_A}{M}}$$

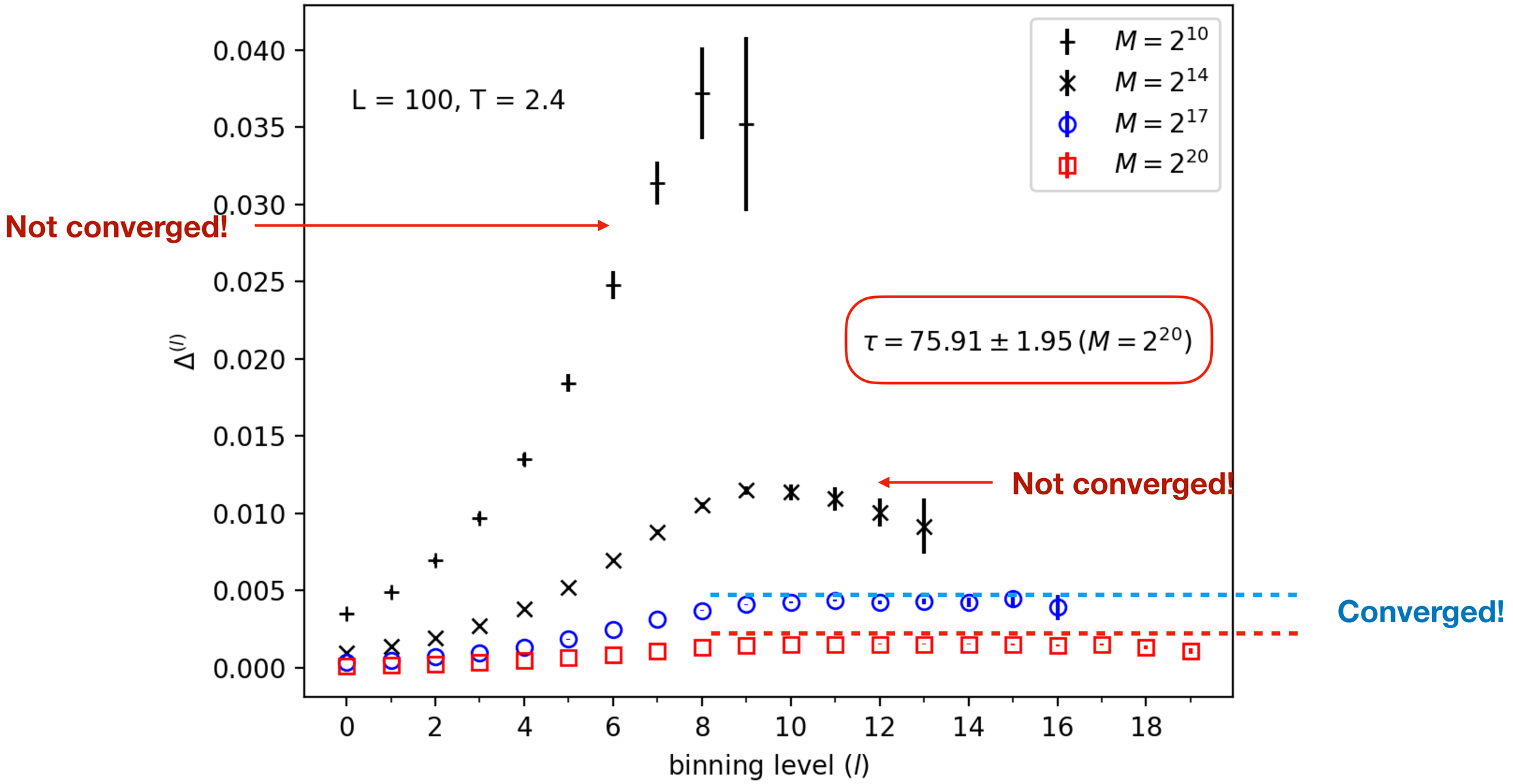
Integrated autocorrelation time

$$\tau_A = \frac{1}{2}\lim_{l\to\infty}\left[\left(\frac{\Delta_A^{(l)}}{\Delta_A^{(0)}}\right)^2 - 1\right]$$

$$\tau_A^{(\mathrm{int})} \equiv 1 + 2\tau_A$$

Keep track of $\Delta_A^{(l)}$ to test the convergence!

Binning analysis : an example



Calculation of Measurement Errors

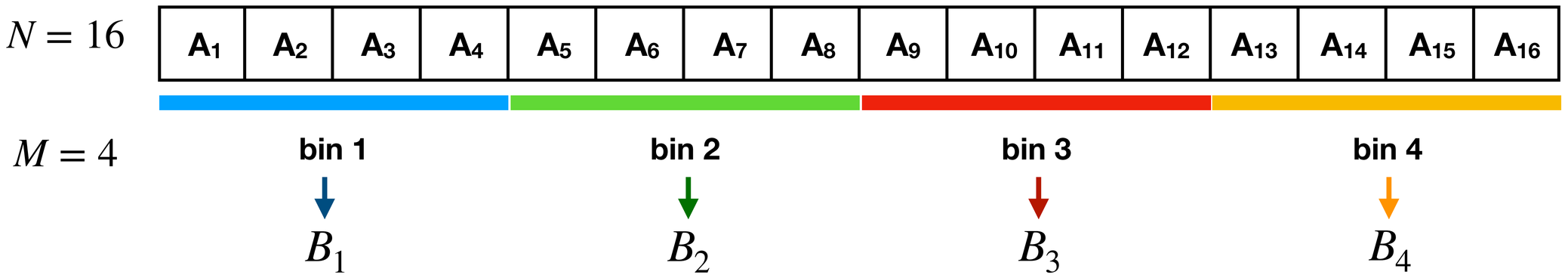
- What if your quantity is a function of other estimates, *e.g. specific heat, magnetic susceptibility, and Binder's cumulant.*
- Magnetic susceptibility $\chi = \beta N \left(\langle m^2 \rangle - \langle m^2 \rangle \right)$: how can you estimate the error of this quantity? The binning analysis is not applicable.
- Blocking method, bootstrap resampling method, **jackknife method**.
- Blocking method: *intuitive*, *simple*, but (a naive version is) *unreliable.*
- Jackknife method: *easy*, *fast*, but *independent samples* needed.

Blocking method

1. Prepare bins (blocks) of consecutive measurements

$$N/M \gg \tau_{\mathrm{int}}$$

2. Compute a quantity (e.g. susceptibility, specific heat) within each block



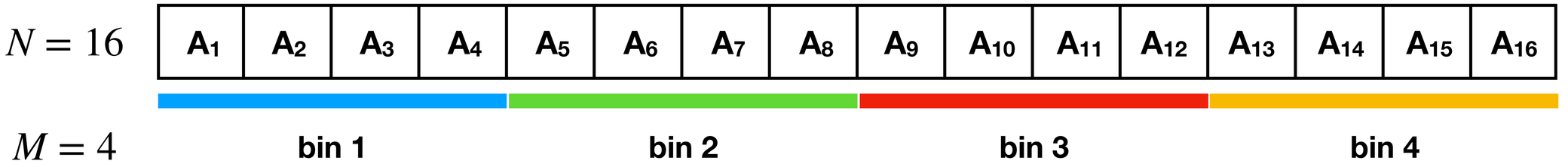
Option 1. Assume no correlation between B's. → $\Delta_B^2 \approx \dfrac{s_B^2}{N_B - 1}$ **(block size dependent!)**

(not recommended)

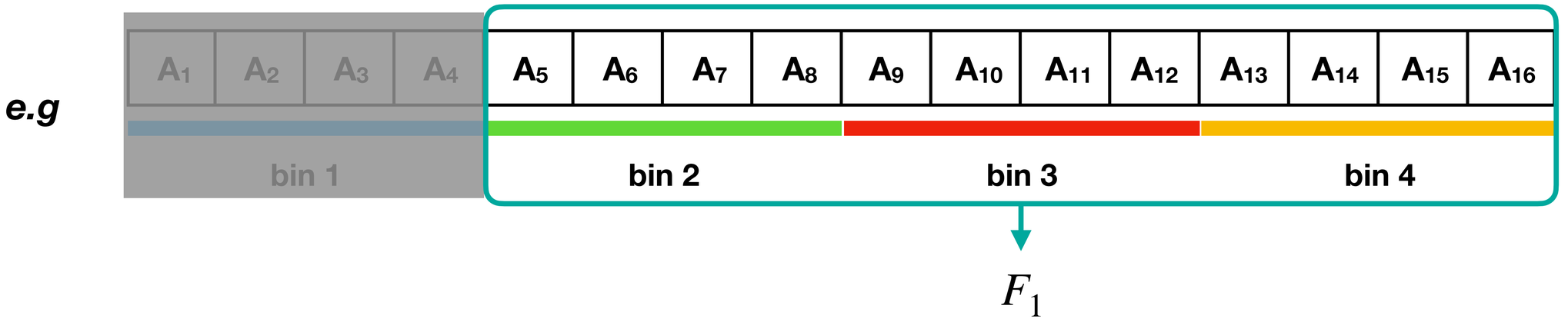
<u>Option 2. Do the binning analysis with the data set of B's</u>

Jackknife method

1. Split the measurements into M bins of size $N/M \gg \tau_{\mathrm{int}}$.



2. Try to discard each bin to prepare a measurement of F_i with remaining data



Jackknife method

3. Jackknife error estimate

Jackknife measurements: $F_0, F_1, F_2, \ldots, F_M$ *(*F_0 *: computation with a full data set)*

Expectation value

$$\langle F \rangle \approx MF_0 - (M-1)\bar{F}$$
$$\approx F_0 \approx \bar{F}$$

Error estimate

$$\Delta_F \approx \sqrt{\sum_{i=1}^{M} \left(F_i - \bar{F}\right)^2}$$

Jackknife or Bootstrap ?

Jackknife

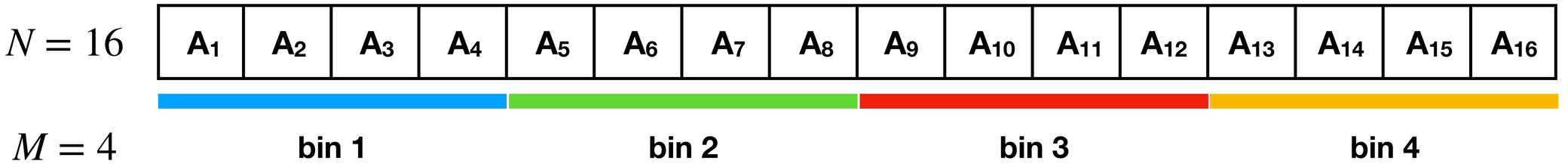
+ quick and easy
+ more suitable for small samples
- poor for non-smooth estimator

Bootstrap

+ all purposes, no assumption
+ more precise
- computationally more intensive

Bootstrap resampling

1. Split the measurements into M bins of size $N/M \gg \tau_{\mathrm{int}}$.



2. Pick randomly M bins with replacement. Compute the quantity of interest.

3. Repeat Step 1-2 (a large number) N_{b} times. Now you have done N_{b} measurements.

4. Compute the error estimate at a given confidence interval with the distribution.

5. Often, the bootstrap estimate of the error is measured as:

$$\Delta_F = \sqrt{\frac{1}{N_b - 1} \sum_{i=0}^{N_b} \left(F_i - \langle F \rangle \right)^2}$$

Finite-size-scaling analysis

- Correlation length is always *finite* in a finite-size system.
- Strictly speaking, a phase transition does not exist in a finite-size system.
- Although, you can still define a *pseudo-critical point* which is getting closer and closer to a true critical point as the system size increases.
- Critical behaviors can be also studied with the *finite-size-scaling ansatz* of macroscopic observables.

Pseudo-critical point & Finite-size-scaling ansatz

At the critical point, the correlation length ξ diverges as $\xi \sim |T - T_c|^{-\nu}$ in the thermodynamic limit.

However, in a finite-size system with length scale L, it has to be like $\xi_L \sim L$ at the largest.

A "pseudo"-critical point : $\xi_L \propto |T_L^* - T_c|^{-\nu} \propto L \longrightarrow T_L^* = T_c + aL^{-1/\nu}$

$|T_L^* - T_c| L^{1/\nu} = O(1)$

How does an observable showing criticality like, $X \sim |T - T_c|^{-x}$, behave in a finite-size system?

Finite-Size-Scaling ansatz : $X_L(T) = L^y X_o(L/\xi) \longrightarrow X_L(T) = L^{x/\nu} \tilde{X}_o\left(|T - T_c| L^{1/\nu}\right)$

$y = x/\nu$

$X_L(T_L^*) = L^y \tilde{X}_o(L/\xi_L) \propto L^y \longleftrightarrow X_L(T_L^*) \propto |T_L^* - T_c|^{-x} \propto \xi_L^{x/\nu} \propto L^{x/\nu}$

Finite-size-scaling ansatz

leading-order behavior in $t \equiv (T - T_c)/T_c$

- magnetization

$$\langle m \rangle = \frac{1}{L^d} \left\langle \left| \sum_{i=1}^{L^d} s_i \right| \right\rangle \longrightarrow m_L = L^{-\beta/\nu} \mathscr{M}_o \left(tL^{1/\nu} \right)$$

- magnetic susceptibility

$$\chi = \beta L^d \left(\langle m^2 \rangle - \langle m \rangle^2 \right) \longrightarrow \chi_L = L^{-\gamma/\nu} \mathscr{X}_o \left(tL^{1/\nu} \right)$$

- specific heat

$$c = \frac{\beta^2}{L^d} \left(\langle E^2 \rangle - \langle E \rangle^2 \right) \longrightarrow c_L = L^{-\alpha/\nu} \mathscr{C}_o \left(tL^{1/\nu} \right)$$

$$c_L^* \sim \log L \ \ (\alpha = 0)$$

***2D Ising model ($\alpha = 0$) : $c(t) \sim -\log|t|$

The fourth-order cumulant

a.k.a. Binder cumulant, Binder parameter

$$U_L = 1 - \frac{\langle m^4 \rangle_L}{3\langle m^2 \rangle_L^2} \quad \xrightarrow{p_L(m,t) = L^{\beta/\nu}\tilde{p}_o(|m|L^{\beta/\nu}, tL^{1/\nu})} \quad U_L = \mathcal{U}_o\left(tL^{1/\nu}\right)$$

i.e. a true critical point = a system-size-invariant crossing point in U_L

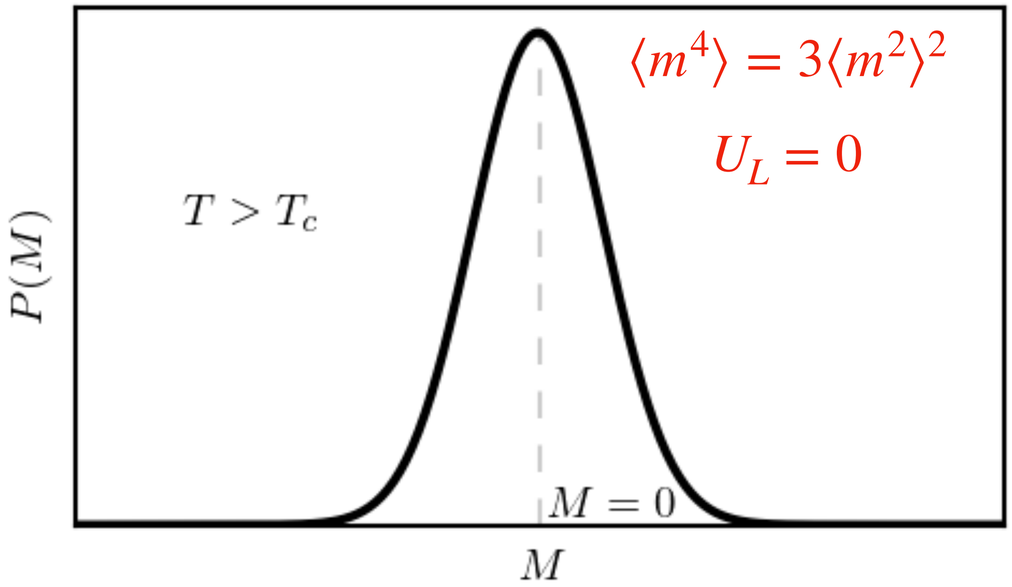
$$U_L(T) = \begin{cases} \frac{2}{3} & \text{for } T < T_c \\ \text{const.} & \text{for } T = T_c \\ 0 & \text{for } T > T_c \end{cases}$$

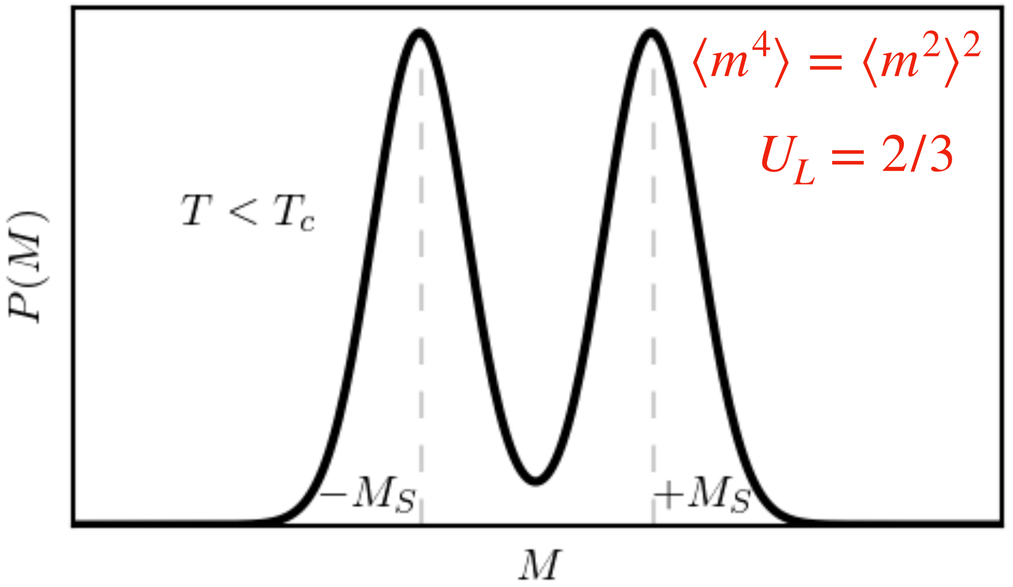
*figures taken from L. Böttcher's lecture note

The fourth-order cumulant

a.k.a. Binder cumulant, Binder parameter

$$U_L = 1 - \frac{\langle m^4 \rangle_L}{3\langle m^2 \rangle_L^2}$$





$$p_L(m) = \sqrt{\frac{L^d}{2\pi\sigma_L^2}} \exp\left(-\frac{L^d m^2}{2\sigma_L^2}\right)$$

$$p_L(m) = \frac{1}{2}\sqrt{\frac{L^d}{2\pi\sigma_L^2}} \left\{ \exp\left[-\frac{L^d(m-m_s)^2}{\sigma_L^2}\right] + \exp\left[-\frac{L^d(m-m_s)^2}{\sigma_L^2}\right] \right\}$$

FSS analysis

1. **Locate the critical point, as precisely as you can.**

   - Crossing point of Binder cumulants: (T_c and $1/\nu$)
   - Maximum point of susceptibility or specific heat. (T_c and $1/\nu$)

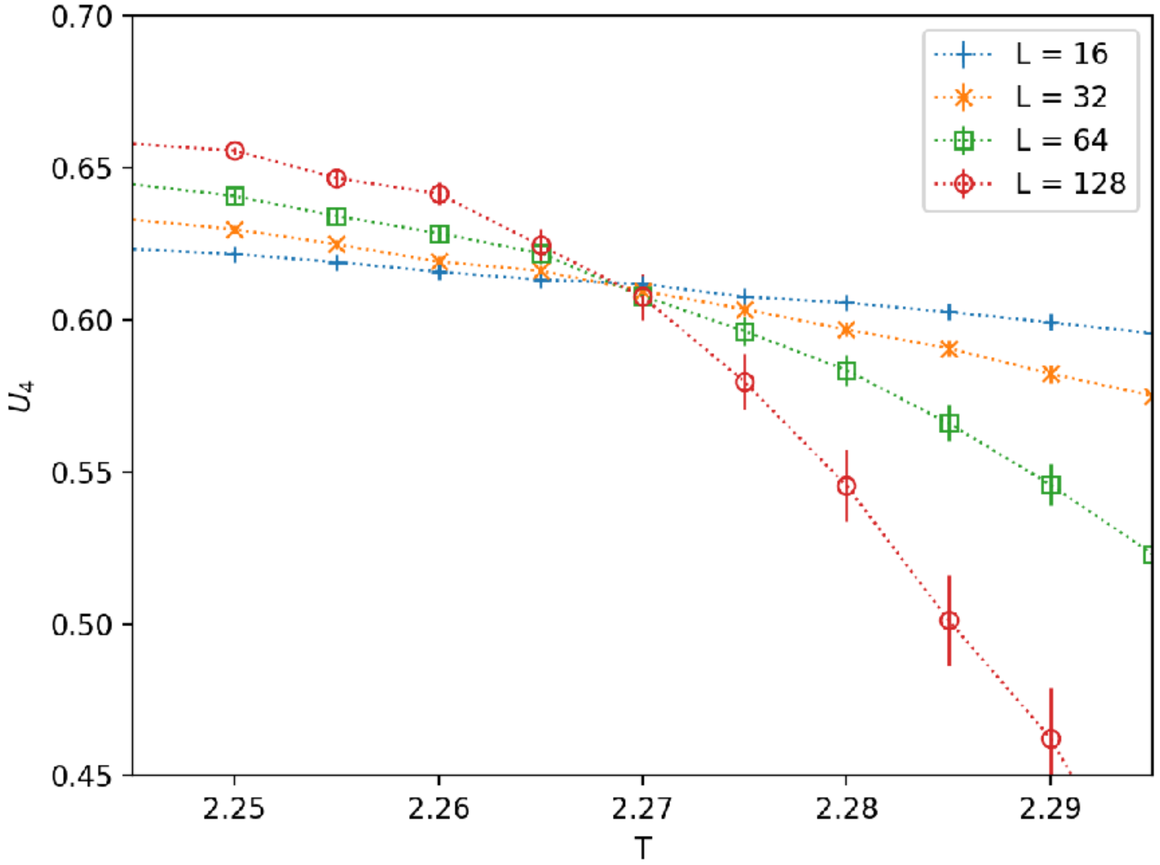
2. **Estimate the critical exponents**

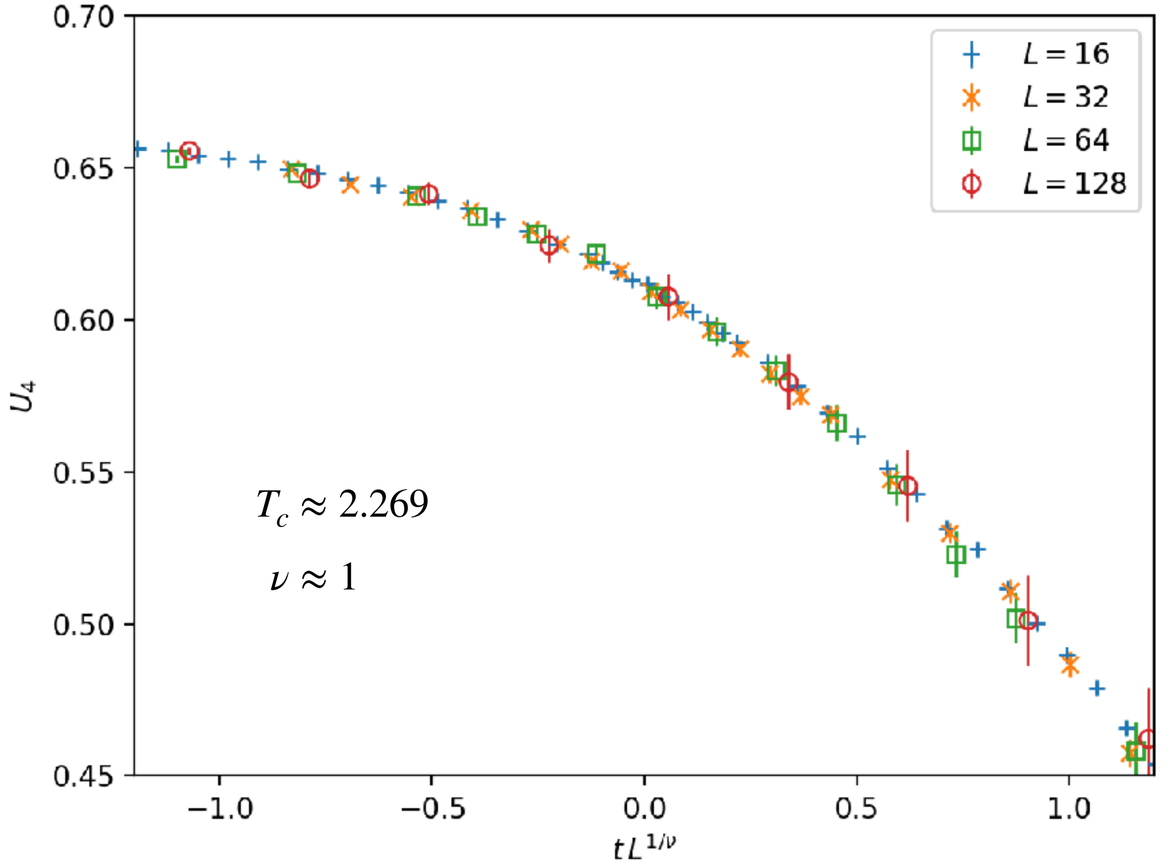
   - Maxima of susceptibility and specific heat (γ/ν and α/ν)

3. **Verify them or measure them agian by with the finite-size-scaling curve collapse**

Example : fourth-order cumulant

2D Ising model, single-spin update, equilibration = 100,000 MCS, measurement = 1,000,000 MCS





Why not just trying the FSS collapse?

- Quality of the FSS collapse is often judged by one's eyes 😅
- There are some ways to systematically measure the quality. For example,

$$\sigma^2 = \frac{1}{x_{\max} - x_{\min}} \int_{x_{\min}}^{x_{\max}} \sum_L \tilde{\chi}_L^2(s) - \left[\sum_L \tilde{\chi}_L(x) \right]^2 \mathrm{d}x$$

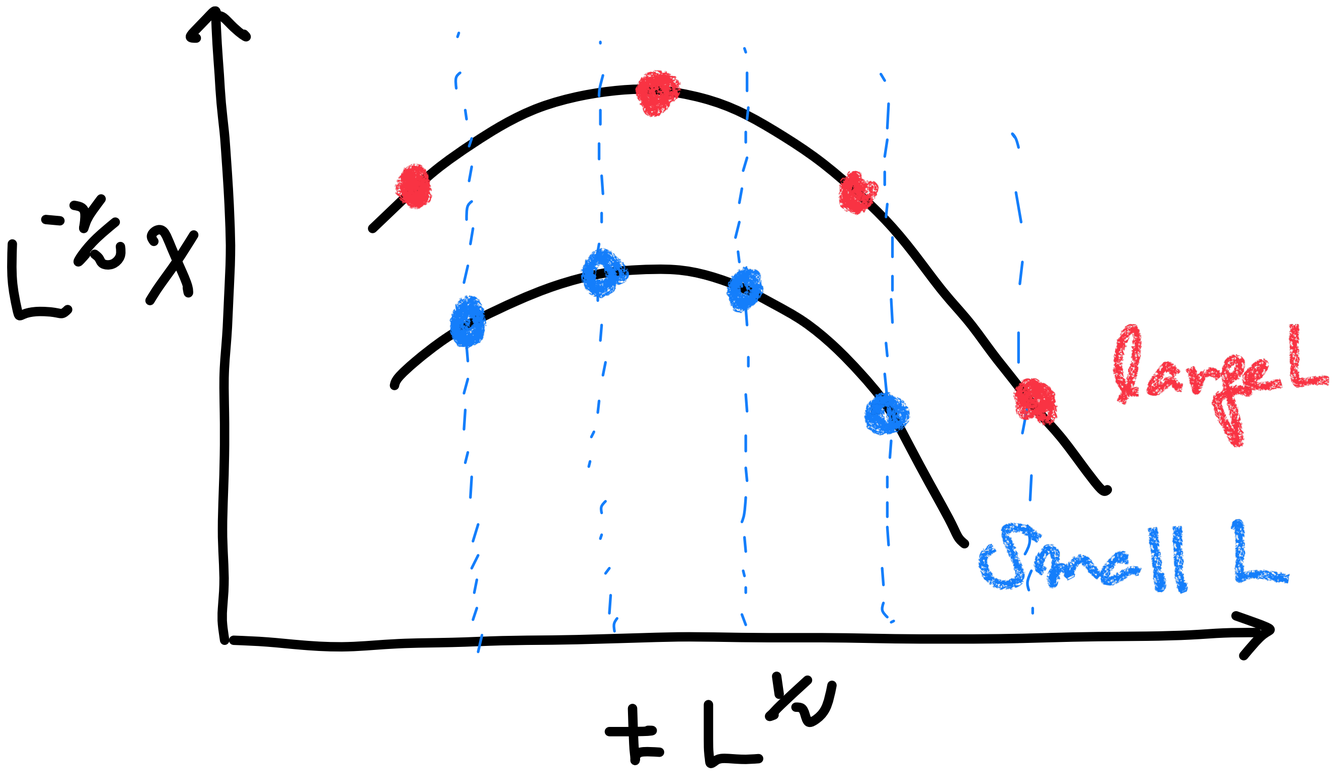
(See Ch.8 of Newman & Barkema)

or $$S = \frac{1}{\mathcal{N}} \sum_{i,j} \frac{(y_{ij} - Y_{ij})^2}{dy_{ij}^2 + dY_{ij}^2}$$

pyfssa: https://pyfssa.readthedocs.io/en/stable/

Kawashima and Ito, JPSJ 62, 435 (1993)
Houdayer and Hartmann PRB 70, 014418 (2004)
Srafino et al., arXiv:1905.09512 [PNAS 118 (2021)]

Why not just trying the FSS collapse?



We have a problem...

Rescaled points of different L's are not on the same line:

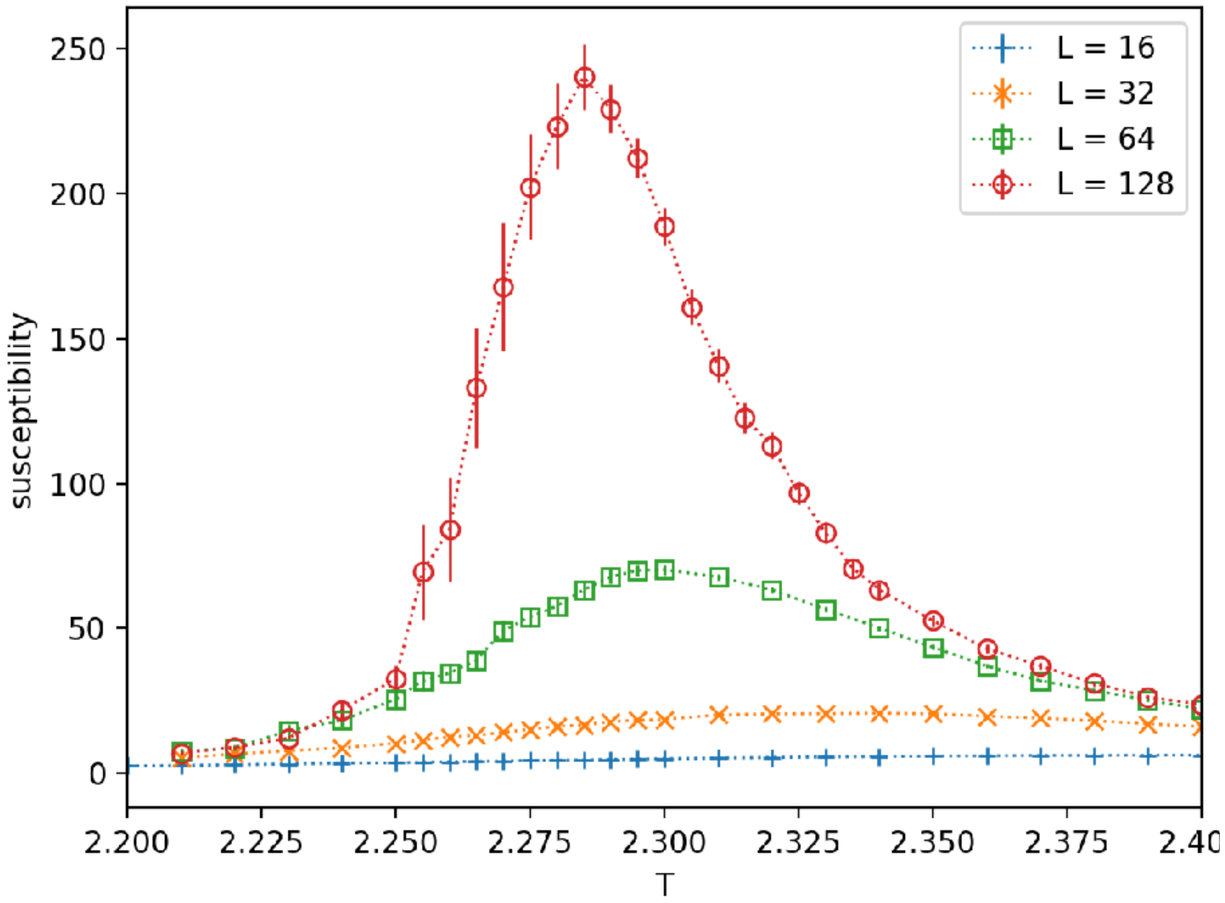
The deviation can't be computed directly.

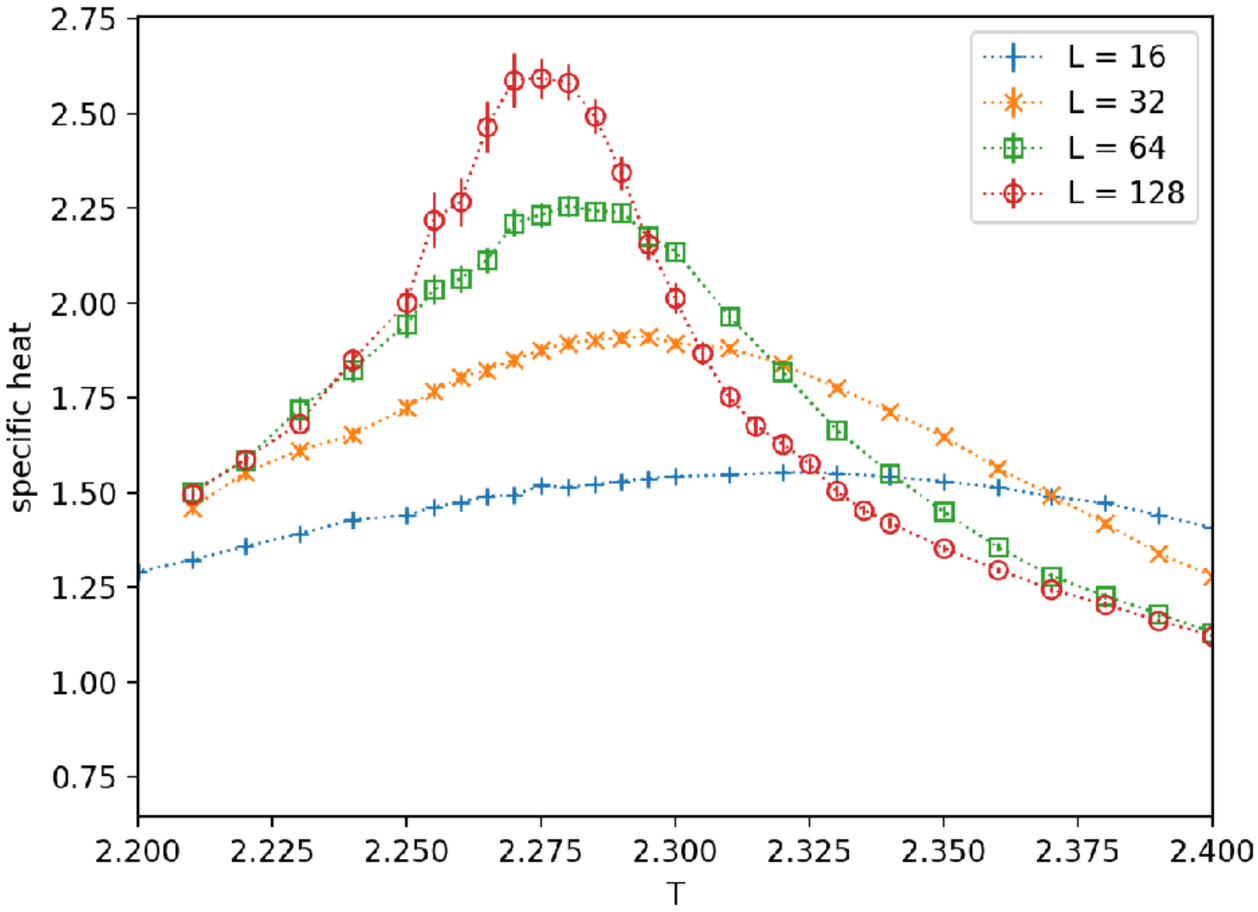
1. histogram reweighting
2. interpolation

2.5. Critical slowing down & Cluster update algorithms

Example: susceptibility and specific heat

Error bars around the critical point increases rapidly with increasing system size L!





2D Ising model, single-spin update, equilibration = 100,000 MCS, measurement = 1,000,000 MCS

Critical slowing down

- At the critical point, the *correlation time* diverges, too!

$$\tau \sim |t|^{-z\nu} \qquad z : \textit{dynamical exponent}$$

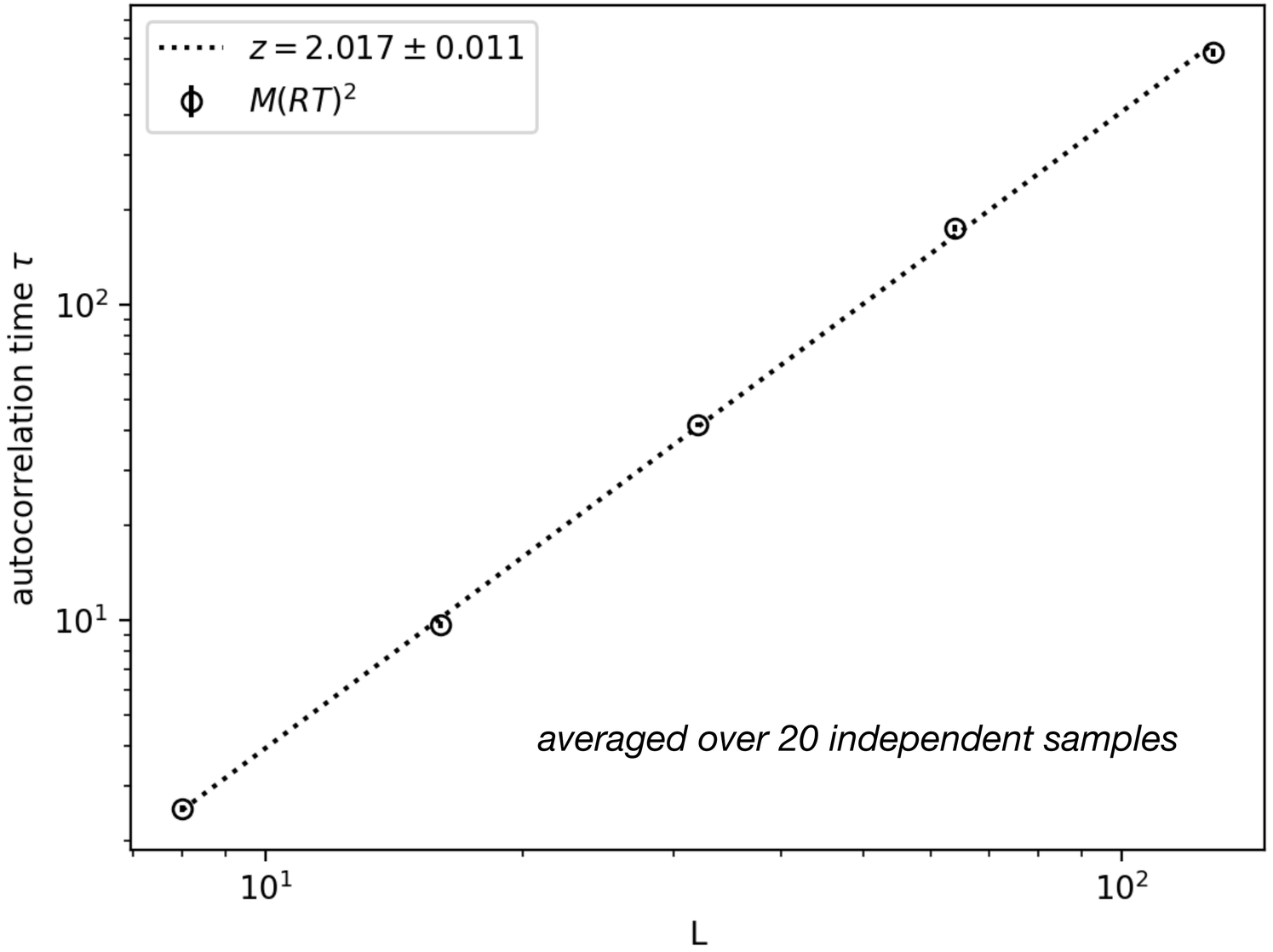
- It is getting worse as the system size increases!

$$\xi \sim |t|^{-\nu} \longrightarrow \tau \sim \xi^z \sim L^z$$

- This is a major issue with the single-spin update scheme.
  - It would need extremely long MC sweeps to simulate a large-size system.

⟶ ***What about updating many spins instead of a single spin?***

Critical slowing down



Autocorrelation time computed at T_c using the binning analysis.

(with single-spin updates)

$$\tau \sim L^z$$

$$z \approx 2$$

Cluster algorithms

- Why not updating many spins at once, instead of a single spin?
- Cluster update algorithms ➔ much shorter autocorrelation time
  ➔ significantly reduces the critical slowing down
- Popular algorithms in classical spin systems: Swendsen-Wang, Wolff cluster updates
- Although, the cluster update schemes are model-dependent. *There are cases where a cluster update strategy is not available.*

Fortuin-Kasteleyn representation

$$p = 1 - e^{-2\beta J}$$

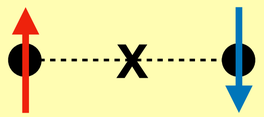
$$Z = \sum_{\{s_i\}} e^{\beta J \sum_{\langle i,j \rangle} s_i s_j} = \sum_{\{s_i\}} \prod_{\langle i,j \rangle} e^{\beta J} \left[(1-p) + p\delta_{s_i,s_j} \right] = \sum_{\{s_i\}} \sum_{n_{ij}} \prod_{\langle i,j \rangle} e^{\beta J} \left[(1-p)\delta_{n_{ij},0} + p\delta_{s_i,s_j}\delta_{n_{ij},1} \right]$$

$$= \sum_{\{s_i\}} \sum_{n_{ij}} \prod_{\langle i,j \rangle} \left[e^{-\beta J}(1-\delta_{s_i,s_j})\delta_{n_{ij},0} + e^{\beta J}\delta_{s_i,s_j} \left\{ (1-p)\delta_{n_{ij},0} + p\delta_{n_{ij},1} \right\} \right] \equiv \sum P(\boldsymbol{\sigma}, \mathbf{n})$$

→ ***bond percolation at a given spin configuration***

different neighboring spins

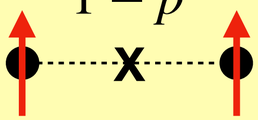
deleted (n=0)



same neighboring spins

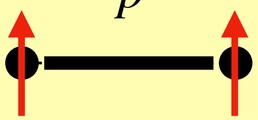
deleted (n=0)

$1-p$



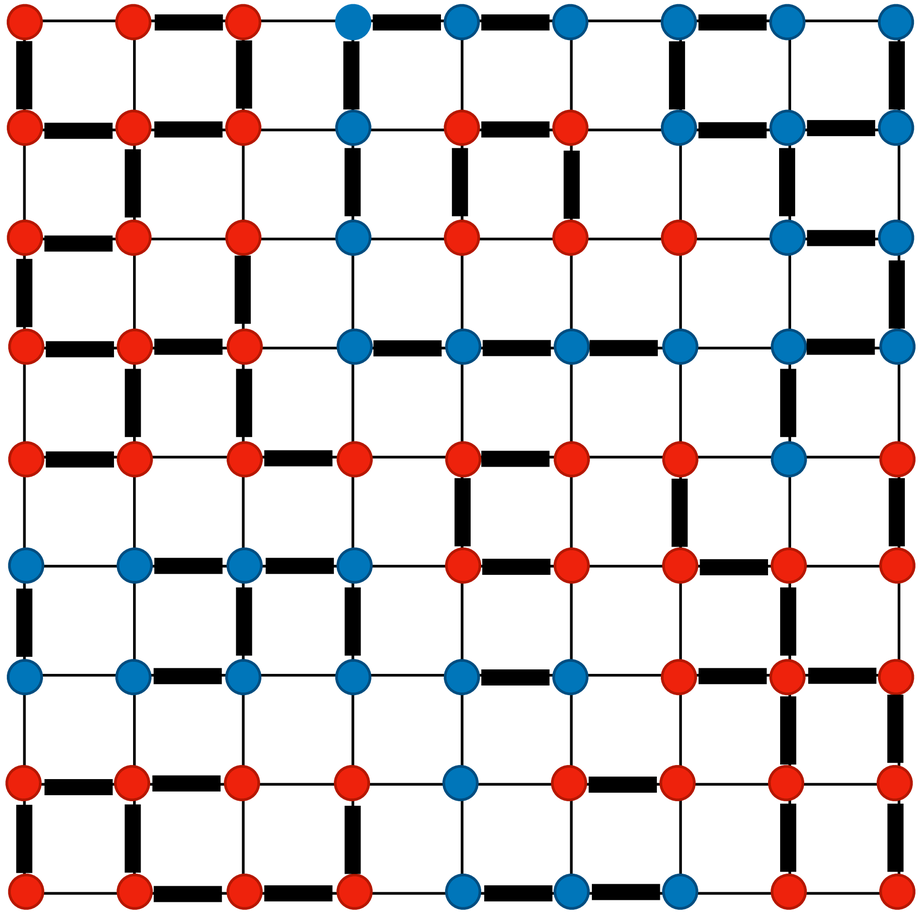
active (n=1)

p



Swendsen-Wang algorithm

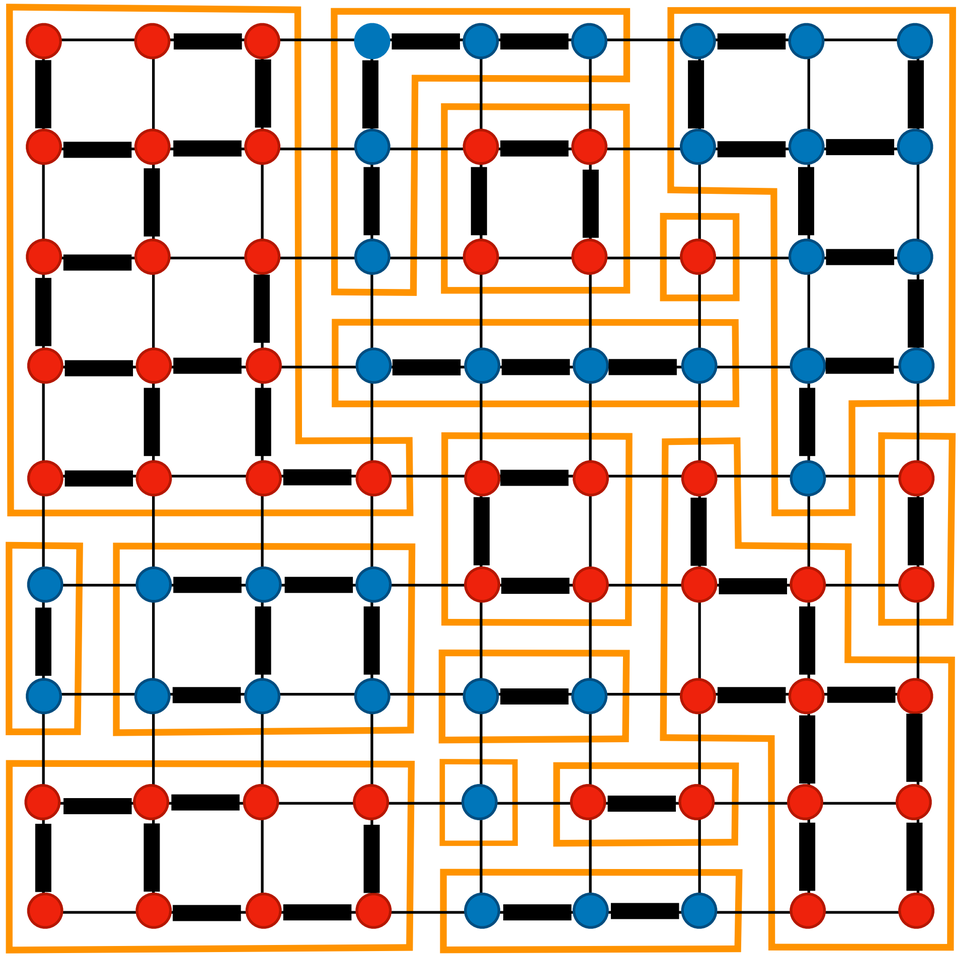
1. Mark active bonds with probability ***p*** for neighboring bonds between the same spins. $p = 1 - \exp[-2\beta J]$



R. H. Swendsen and J.-S. Wang, PRL 58, 86 (1987).
J.-S. Wang and R. H. Swendsen, Physica A 167, 565 (1990).

Swendsen-Wang algorithm

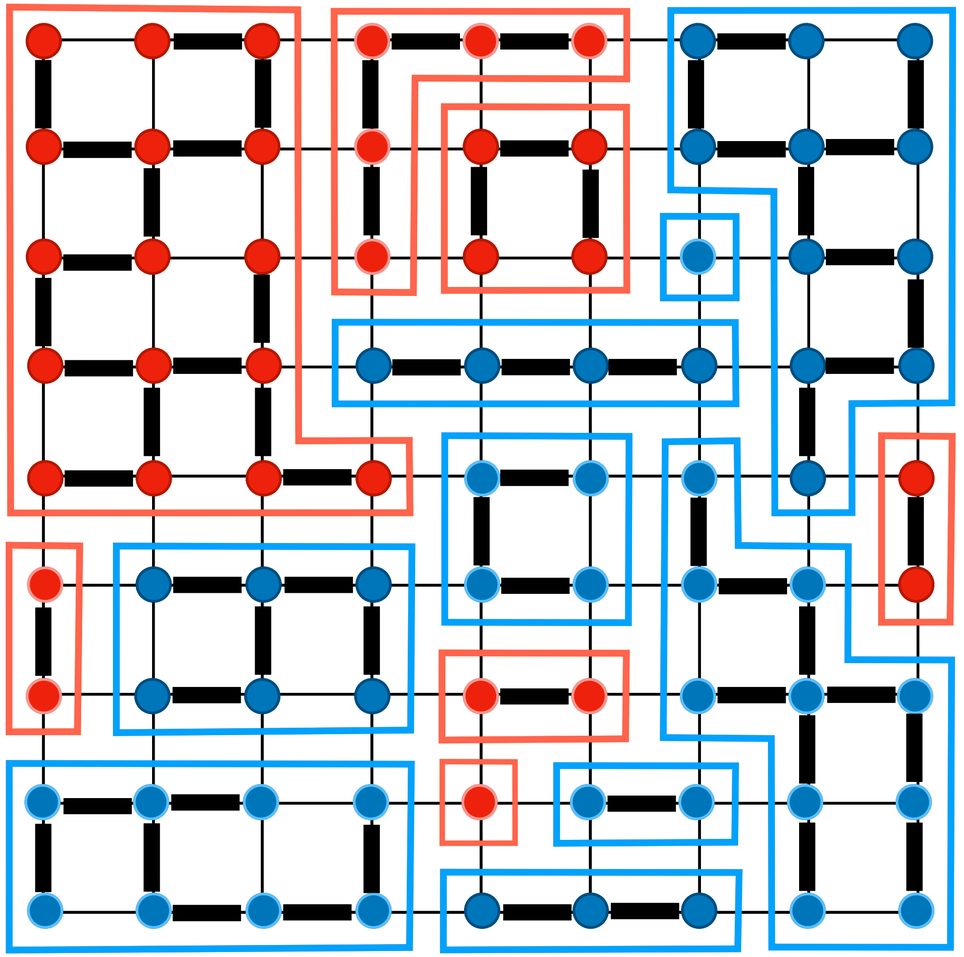
1. Mark active bonds with probability ***p*** for neighboring bonds between the same spins. $p = 1 - \exp[-2\beta J]$
2. Identify ALL clusters connected through the active bonds.



R. H. Swendsen and J.-S. Wang, PRL 58, 86 (1987).
J.-S. Wang and R. H. Swendsen, Physica A 167, 565 (1990).

Swendsen-Wang algorithm

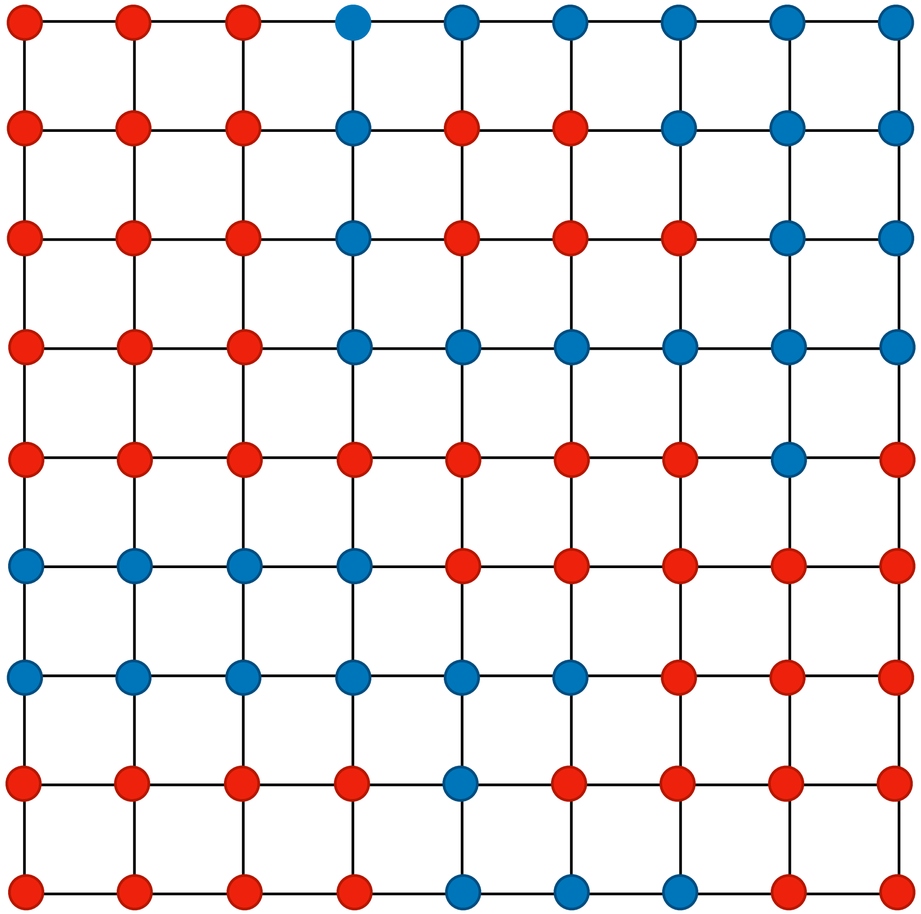
1. Mark active bonds with probability ***p*** for neighboring bonds between the same spins. $p = 1 - \exp[-2\beta J]$
2. Identify ALL clusters connected through the active bonds.
3. Assign a new spin +1 or -1 randomly for each cluster.



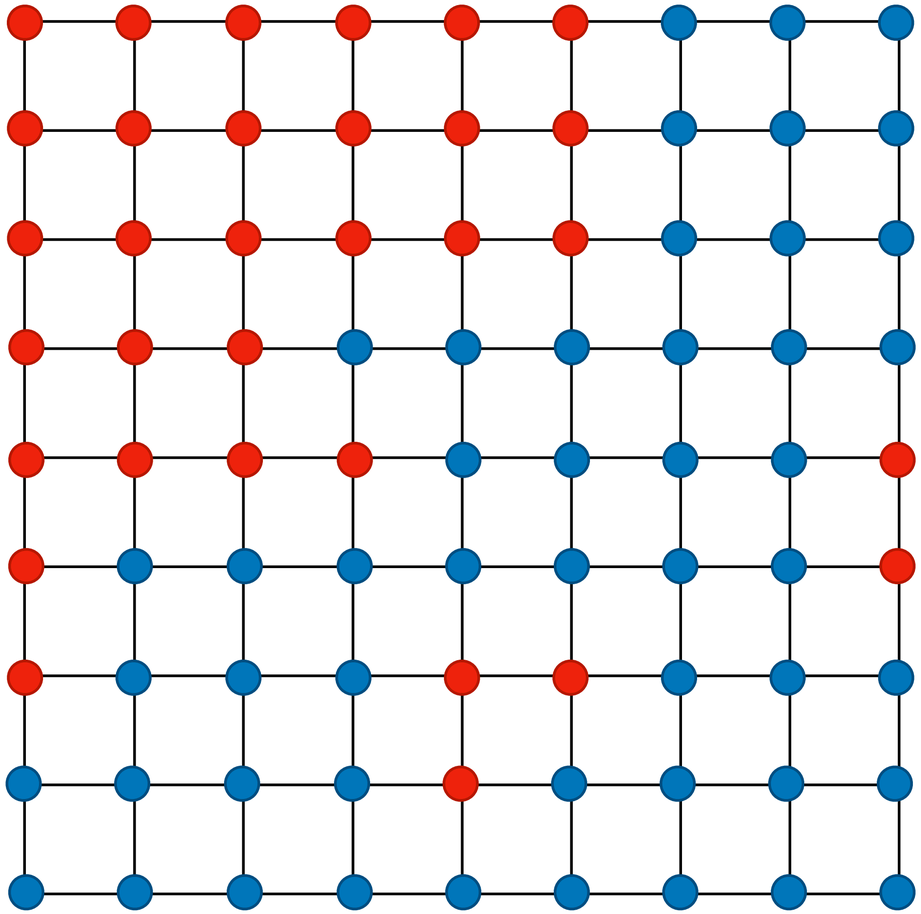
R. H. Swendsen and J.-S. Wang, PRL 58, 86 (1987).
J.-S. Wang and R. H. Swendsen, Physica A 167, 565 (1990).

Swendsen-Wang algorithm

OLD



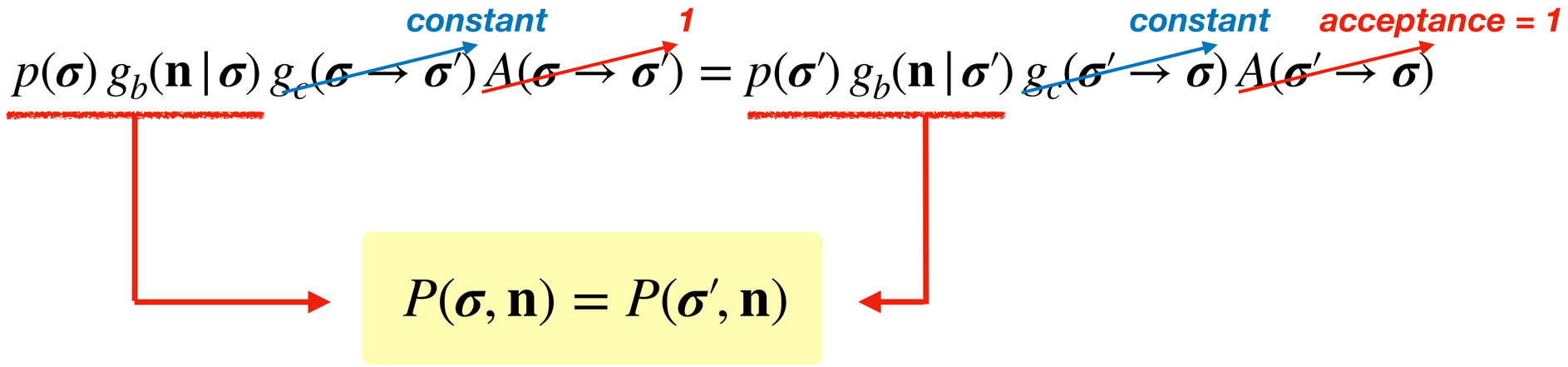
NEW



Detailed Balance

detailed balance condition $$p(\boldsymbol{\sigma})W(\boldsymbol{\sigma} \to \boldsymbol{\sigma}') = p(\boldsymbol{\sigma}')W(\boldsymbol{\sigma}' \to \boldsymbol{\sigma})$$

SW update

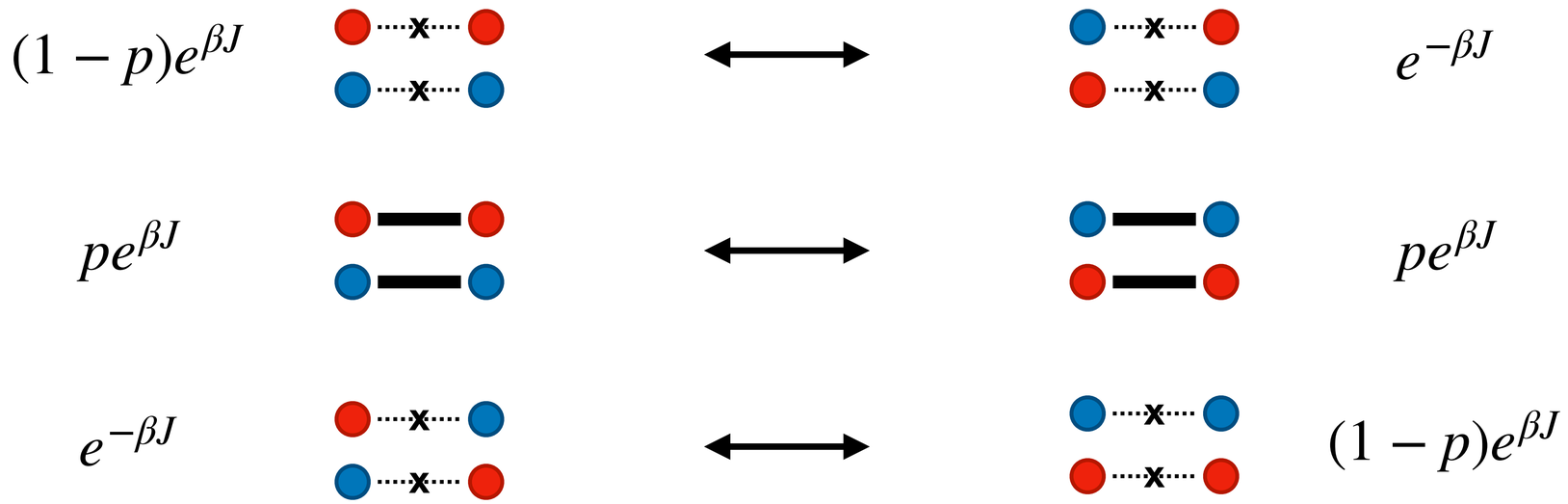


$$Z = \sum_{\{s_i\}} \sum_{n_{ij}} \prod_{\langle i,j \rangle} \left[e^{-\beta J}(1 - \delta_{s_i,s_j})\delta_{n_{ij},0} + e^{\beta J}\delta_{s_i,s_j} \left\{ (1-p)\delta_{n_{ij},0} + p\delta_{n_{ij},1} \right\} \right] \equiv \sum P(\boldsymbol{\sigma}, \mathbf{n})$$

Detailed Balance

$$Z = \sum_{\{s_i\}} \sum_{n_{ij}} \prod_{\langle i,j \rangle} \left[e^{-\beta J}(1 - \delta_{s_i,s_j})\delta_{n_{ij},0} + e^{\beta J}\delta_{s_i,s_j} \left\{ (1-p)\delta_{n_{ij},0} + p\delta_{n_{ij},1} \right\} \right] \equiv \sum P(\boldsymbol{\sigma}, \mathbf{n})$$

$$P(\boldsymbol{\sigma}, \mathbf{n}) = P(\boldsymbol{\sigma}', \mathbf{n}) \quad \longleftarrow \quad \textbf{\textit{Try}}\ p = 1 - e^{-2\beta J}\ \textbf{!}$$



Numerical Bottleneck

- In the Swendsen-Wang algorithm, "all" clusters must be identified.
- Cluster identification can be a major bottleneck.
- Hoshen-Kopelmann algorithm for cluster labelling
- Newman-Ziff algorithm
- Wolff update is preferred if available because of its simplicity.

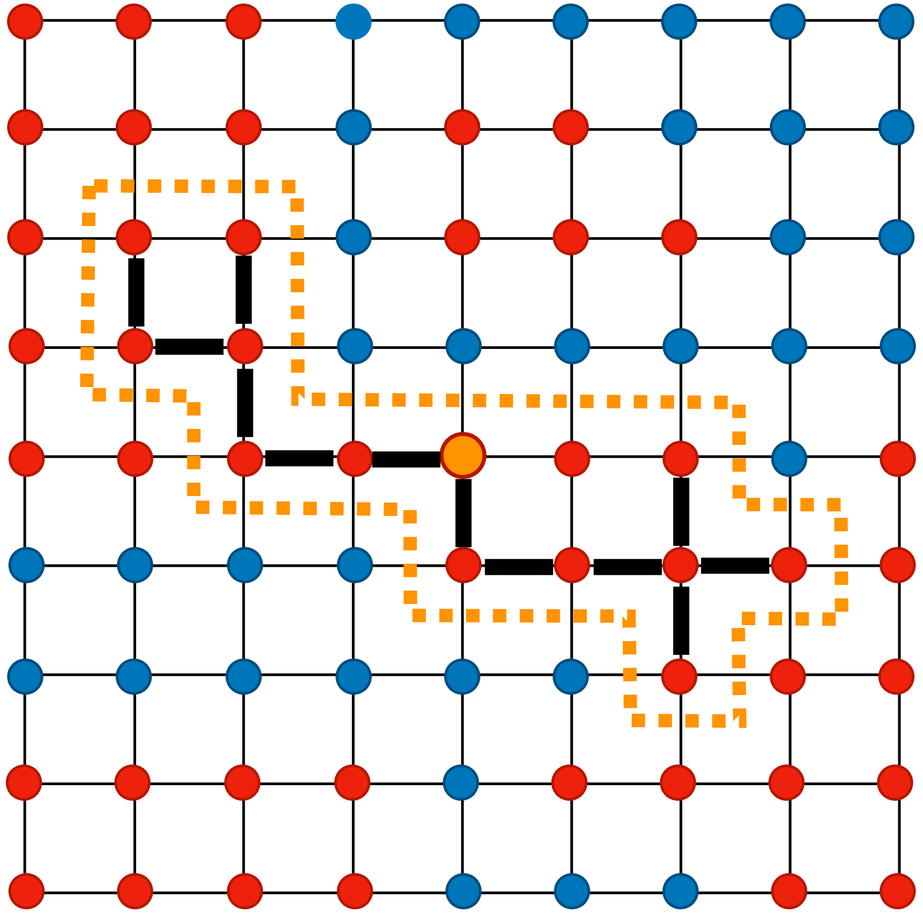
Wolff algorithm

1. Choose a random site to be a starting site.

2. From the starting site, make a cluster with the neighboring sites of the same spin through active bonds chosen with probability
   $p = 1 - \exp(-2\beta J)$.

3. Expand the cluster by inspecting the neighbors of newly joined sites until it stops growing.

4. Flip the spin of the cluster.



U. Wolff, PRL 62, 361 (1989).

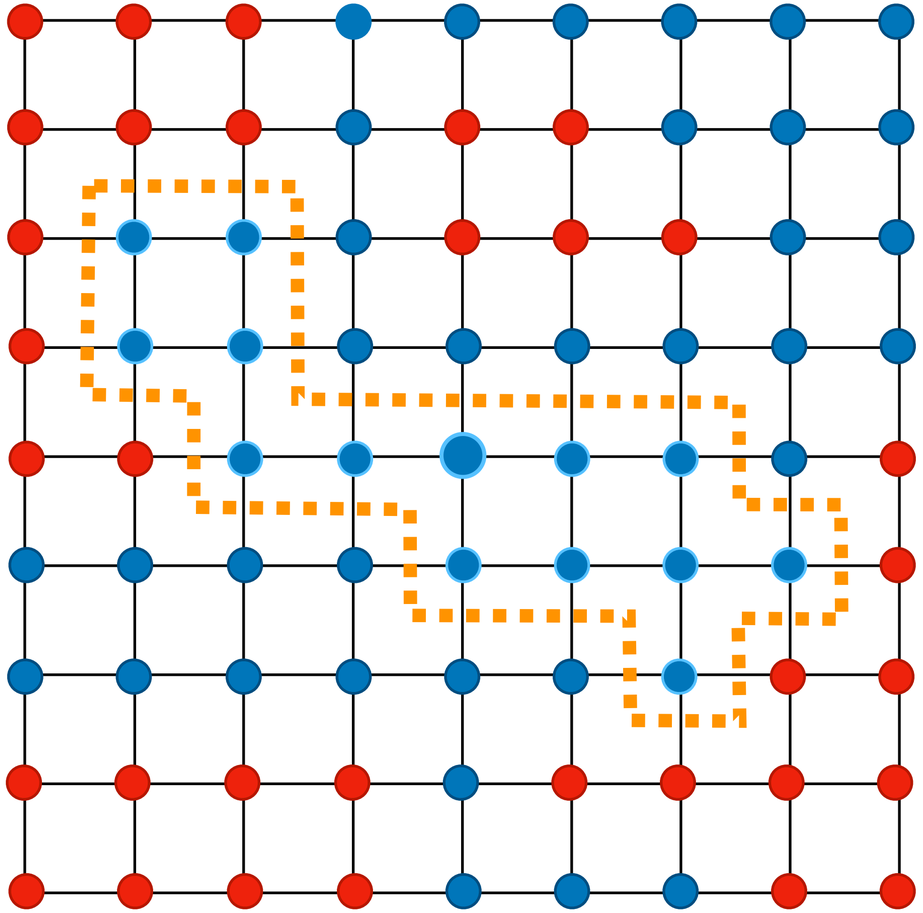
Wolff algorithm

1. Choose a random site to be a starting site.

2. From the starting site, make a cluster with the neighboring sites of the same spin through active bonds chosen with probability
   $p = 1 - \exp(-2\beta J)$.

3. Expand the cluster by inspecting the neighbors of newly joined sites until it stops growing.

4. Flip the spin of the cluster.



U. Wolff, PRL 62, 361 (1989).

Python implementation

```
import numpy as np
from collections import deque

def wolff(k, S, A, p, m) :
    nc = 1
    s0 = S[k]
    deq = deque()
    deq.append(k)
    S[k] = -s0
    while deq :
        i = deq.popleft()
        for j in A[i] :
            if S[j] == s0 :
                if np.random.random() < p :
                    deq.append(j)
                    S[j] = -s0
                    nc = nc + 1
    m = m - 2 * nc * s0
    return m, nc
```

```
mcs_max = 100000

pc = 1.0 - np.exp(-2.0 / T)

# run like this:

for mcs in range(mcs_max) :
    loc = np.random.randint(N)
    mag, nc = wolff(loc, spin, neighbor, pc, mag)
```

Settings such as spin, mag, neighbor, T, L, and N are the same as defined in the implementation of MRT².

Detailed Balance

OLD STATE (A)

stopping dead bonds (red-red) = d_A

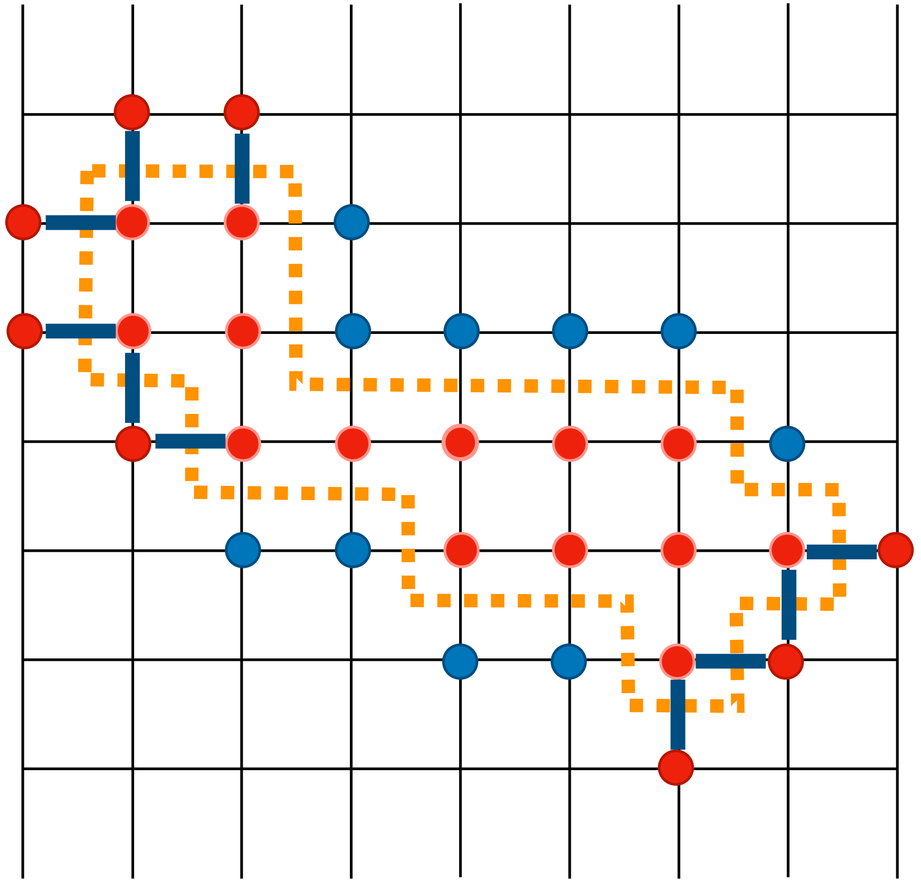
other dead bonds (red-blue) = d_B

transition probability

$$W(\boldsymbol{\sigma}_A \to \boldsymbol{\sigma}_B) \propto (1-p)^{d_A}$$

energy

$$E_A = -Jd_A + Jd_B + (\text{irrelevant})$$



Detailed Balance

NEW STATE (B)

stopping dead bonds (blue-blue) = d_B

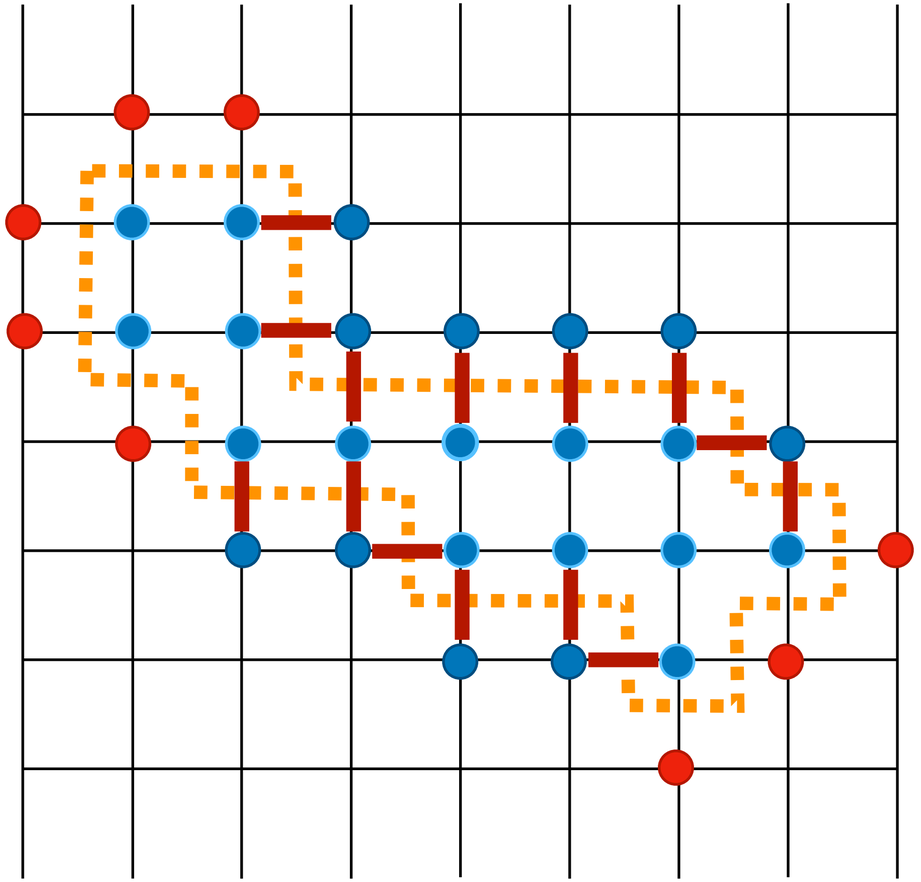
other dead bonds (blue-red) = d_A

transition probability

$$W(\boldsymbol{\sigma}_B \rightarrow \boldsymbol{\sigma}_A) \propto (1-p)^{d_B}$$

energy

$$E_B = -Jd_B + Jd_A + (\text{irrelevant})$$



Detailed Balance

transition probability

$$W(\boldsymbol{\sigma}_A \to \boldsymbol{\sigma}_B) \propto (1-p)^{d_A} \qquad W(\boldsymbol{\sigma}_B \to \boldsymbol{\sigma}_A) \propto (1-p)^{d_B}$$

energy

$$E_A = -Jd_A + Jd_B + (\text{irrelevant}) \qquad E_B = -Jd_B + Jd_A + (\text{irrelevant})$$

detailed balance

$$\frac{W(\boldsymbol{\sigma}_A \to \boldsymbol{\sigma}_B)}{W(\boldsymbol{\sigma}_B \to \boldsymbol{\sigma}_A)} = \exp[-\beta(E_B - E_A)]$$

→ $(1-p)^{d_A - d_B} = \exp[-2\beta J(d_A - d_B)]$ ↔ $p = 1 - e^{-2\beta J}$

Improved estimators

Swendsen-Wang

two spin term: $s_i s_j = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are on the same cluster} \\ 0 & \text{otherwise} \end{cases}$

$$\langle m^2 \rangle = \frac{1}{N^2} \sum_{ij} \langle s_i s_j \rangle = \frac{1}{N^2} \left\langle \sum_k n_k^2 \right\rangle$$

n_k : number of spins in cluster *k*

Wolff

$$\langle n \rangle = \left\langle \sum_k p_k n_k \right\rangle = \frac{1}{N} \left\langle \sum_k n_k^2 \right\rangle = N \langle m^2 \rangle \longrightarrow \chi = \beta \left(\langle n \rangle - N \langle m \rangle^2 \right)$$

$p_k = \frac{n_k}{N}$: probability of choosing a cluster *k* of size n_k

Dynamical exponent

Wolff single-cluster update algorithm

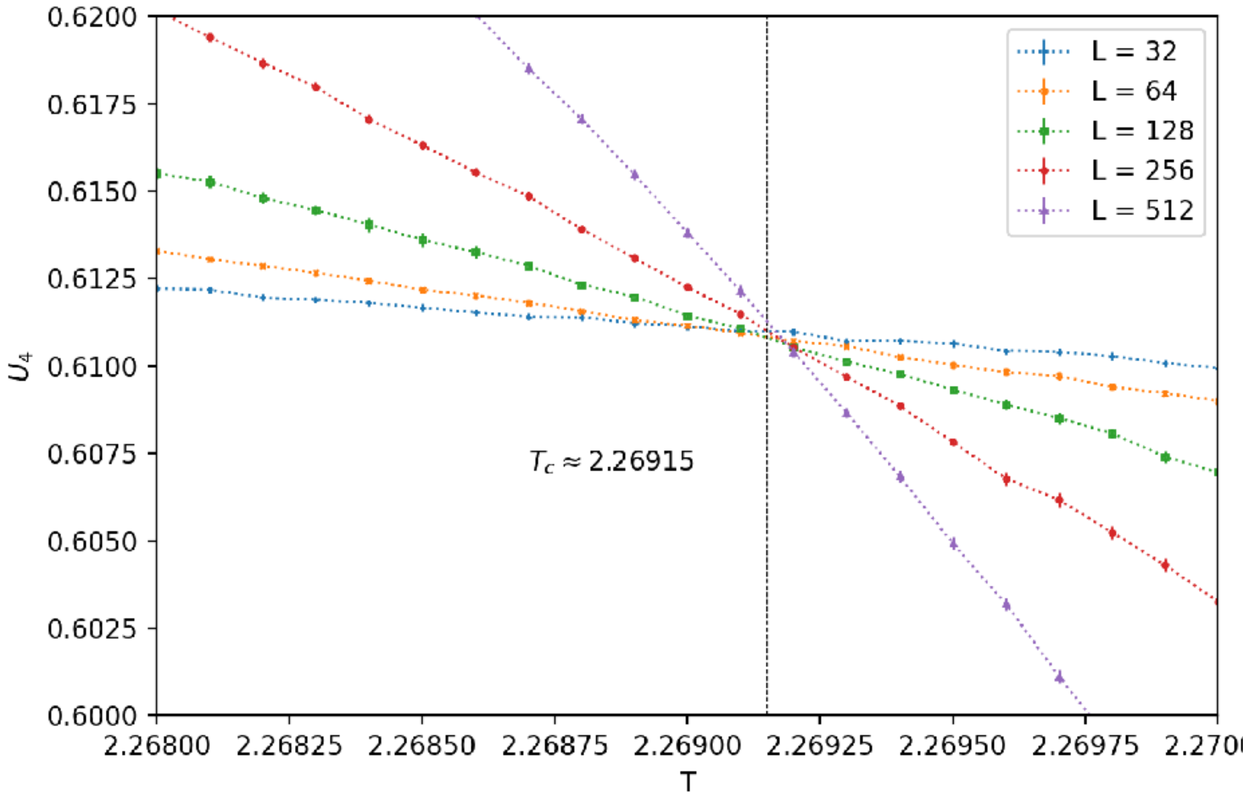
$$\tau = \tau_{\text{steps}} \frac{\langle n \rangle}{L^d} = \tau_{\text{steps}} \frac{\chi}{\beta L^d} \xrightarrow{\text{at } T \to T_c^+} L^z \sim L^{z_{\text{steps}}} \cdot L^{\gamma/\nu} \cdot L^{-d}$$

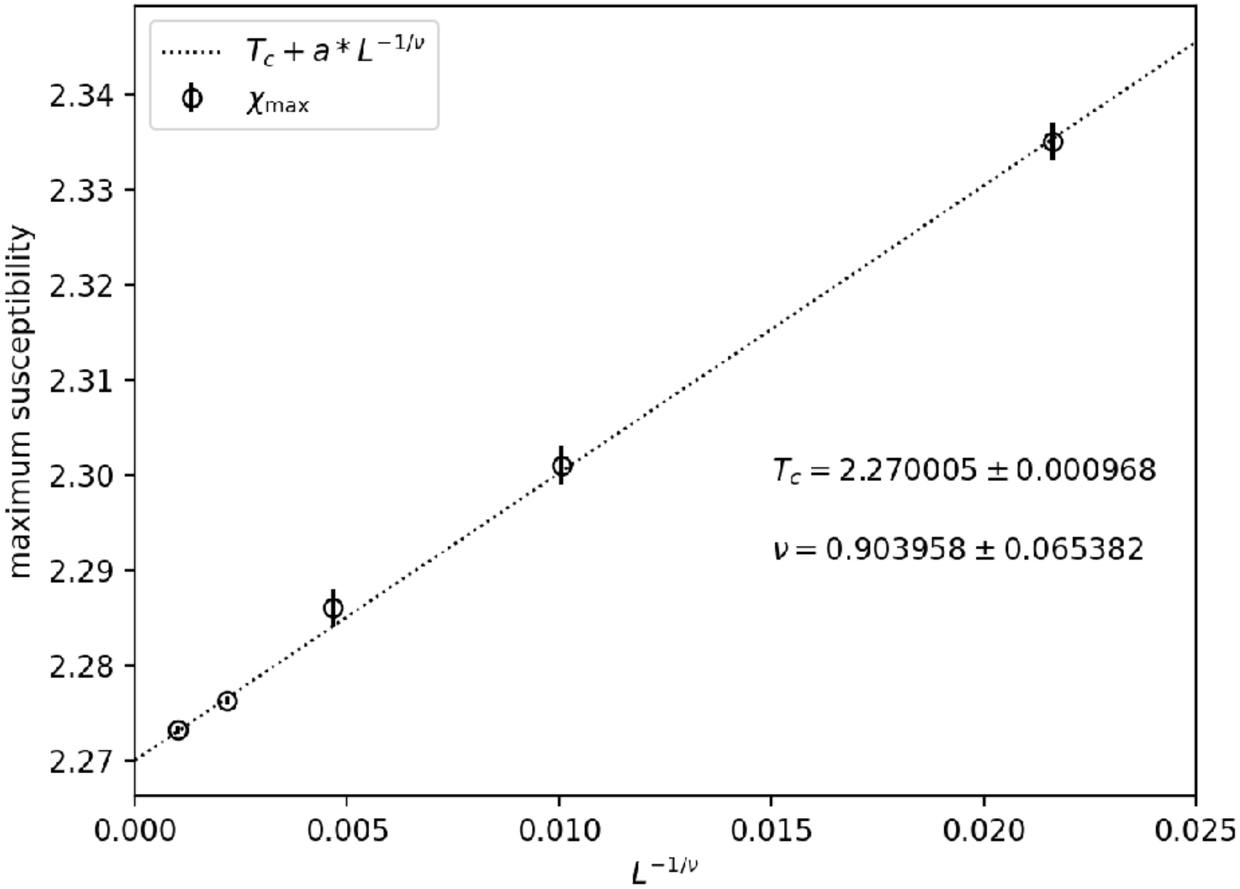
$$\chi = \beta \langle n \rangle \quad (T > T_c)$$

$$z = z_{\text{steps}} + \frac{\gamma}{\nu} - d$$

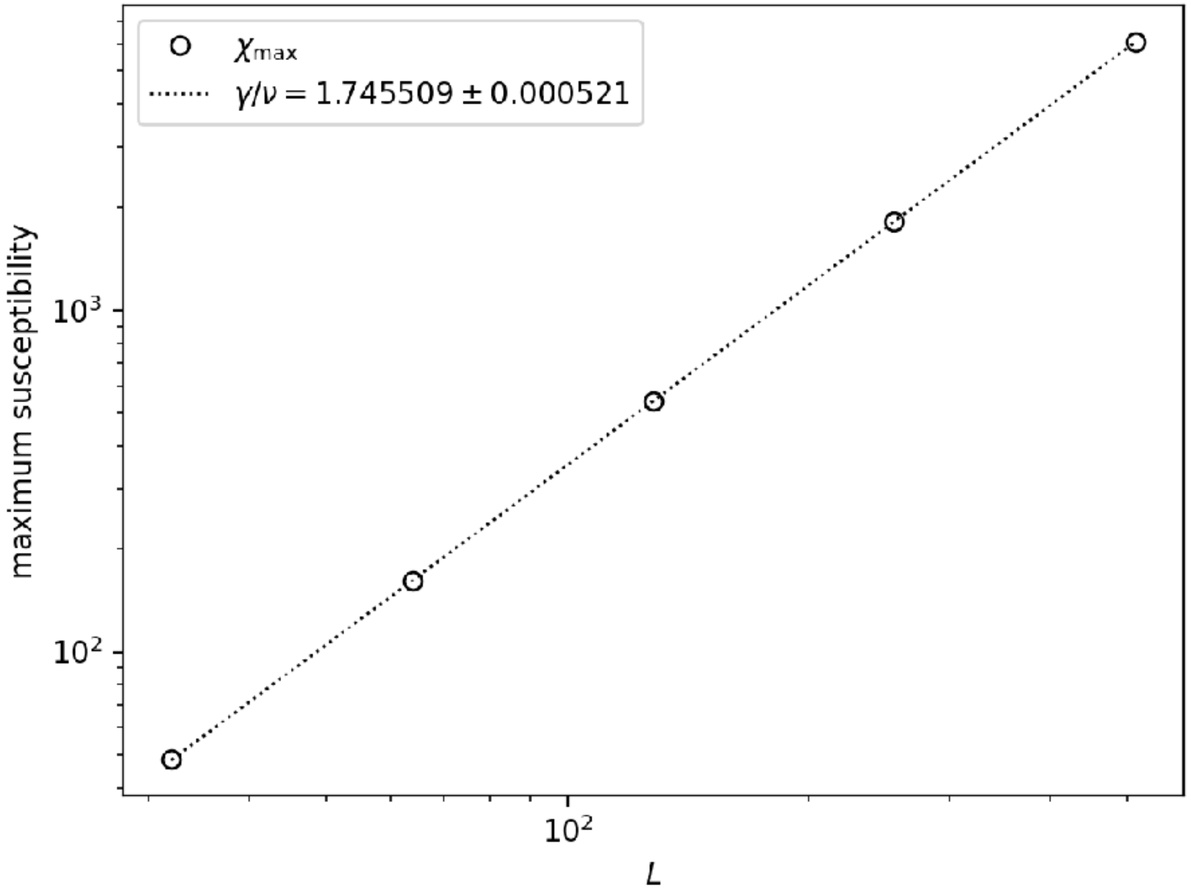
Dimension	M(RT)²	Wolff	Swendsen-Wang
d = 2	z = 2.167(1)	z = 0.25(1)	z = 0.25(1)
d = 3	z = 2.02(2)	z = 0.33(1)	z = 0.54(2)

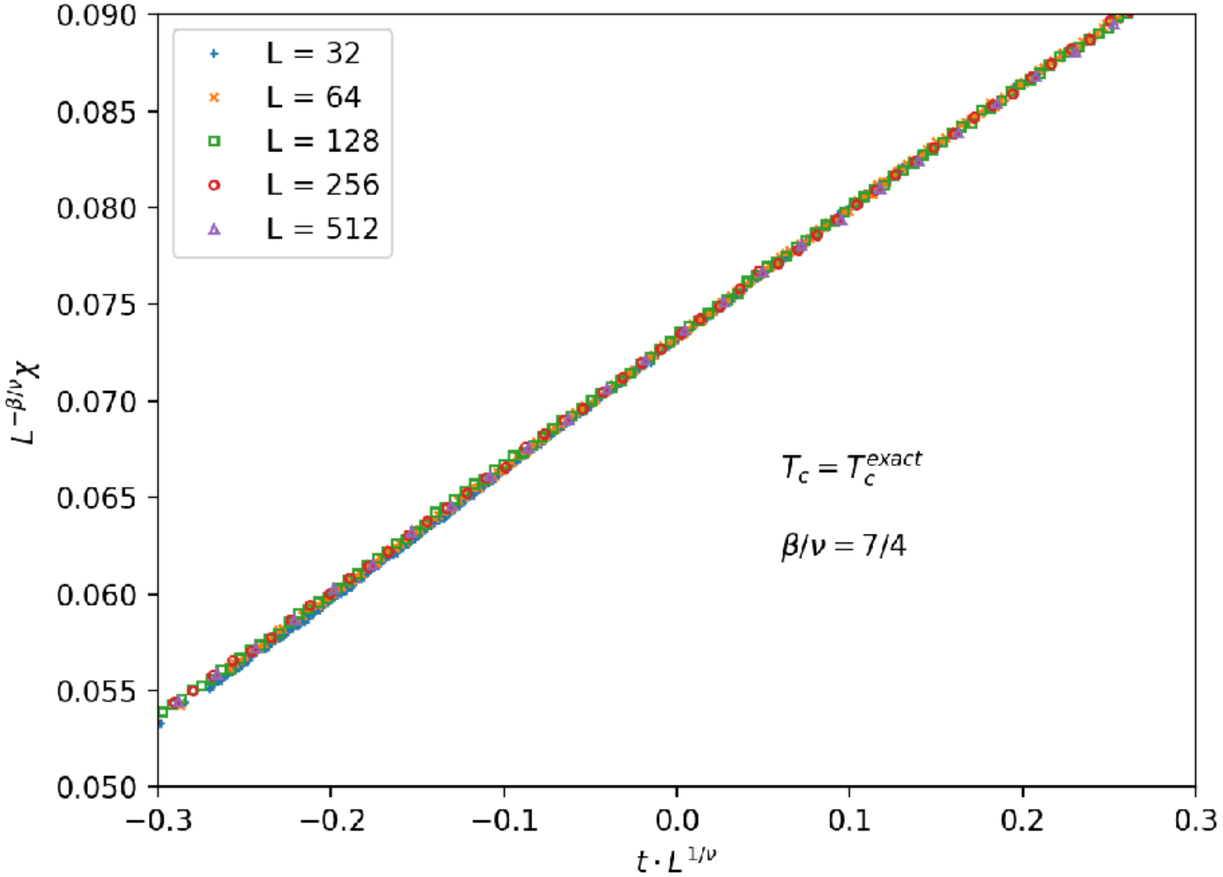
How much improved:





How much improved:





Remarks

- Pay enough attention to the equilibration and autocorrelation time.
- Be careful with issues like critical slowing down and impenetrable barrier.
- No one accepts Monte Carlo data without the error estimates.
- Finite-size-scaling analysis is sometimes very delicate. Practice.
- We have only covered "classic" techniques of Monte Carlo simulations. There are many "modern" developments. A lot to study 😀