Monte Carlo simulations and error analysis

Matthias Troyer, ETH Zürich

Outline of the lecture

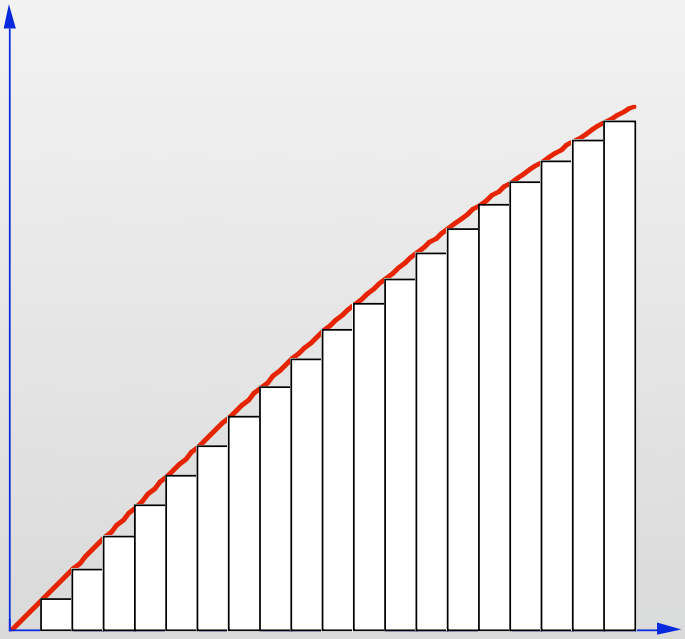
1. Monte Carlo integration
2. Generating random numbers
3. The Metropolis algorithm
4. Monte Carlo error analysis
5. Cluster updates and Wang-Landau sampling
6. The negative sign problem in quantum Monte Carlo

1. Monte Carlo Integration

Integrating a function

- Convert the integral to a discrete sum

$$\int_a^b f(x)dx = \frac{b-a}{N}\sum_{i=1}^{N} f\left(a + i\frac{b-a}{N}\right) + O(1/N)$$



- Higher order integrators:
  - Trapezoidal rule:

$$\int_a^b f(x)dx = \frac{b-a}{N}\left(\frac{1}{2}f(a) + \sum_{i=1}^{N-1} f\left(a + i\frac{b-a}{N}\right) + \frac{1}{2}f(b)\right) + O(1/N^2)$$

  - Simpson rule:

$$\int_a^b f(x)dx = \frac{b-a}{3N}\left(f(a) + \sum_{i=1}^{N-1}(3-(-1)^i) f\left(a + i\frac{b-a}{N}\right) + f(b)\right) + O(1/N^4)$$

High dimensional integrals

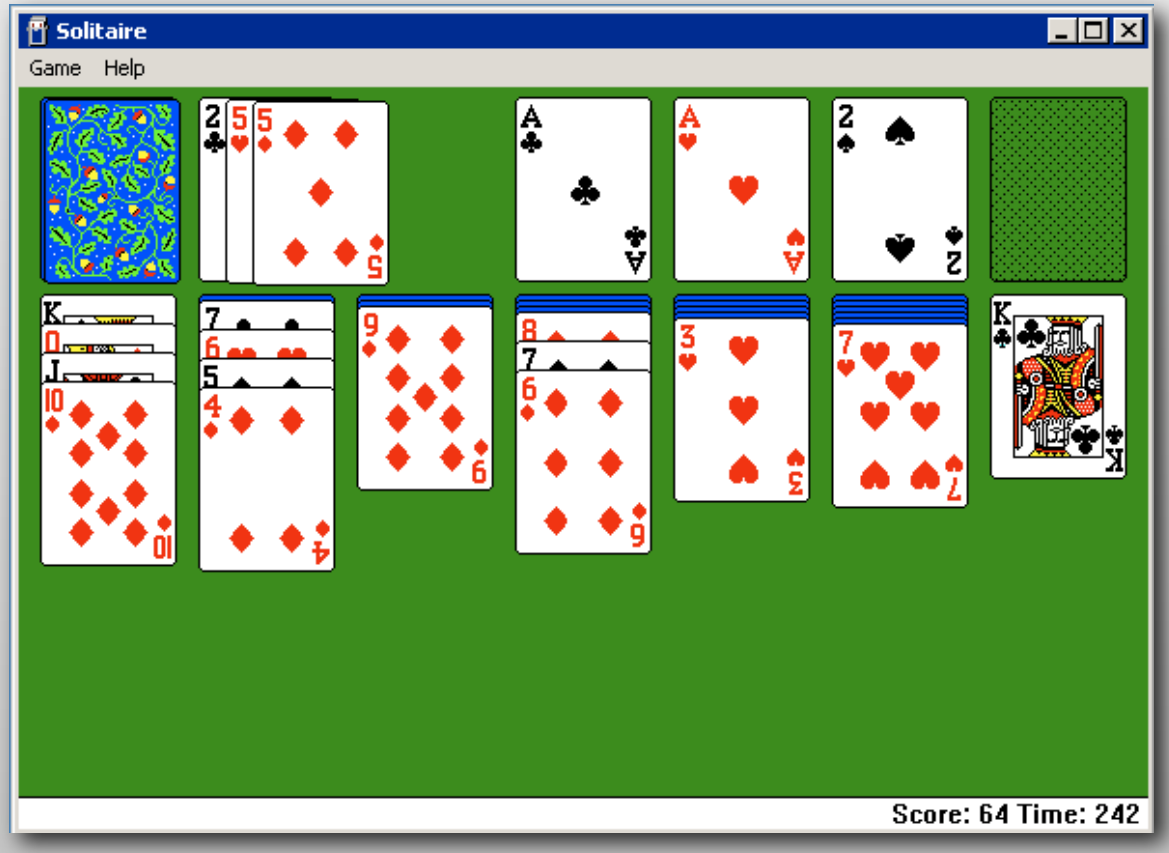
- Simpson rule with M points per dimension
  - one dimension the error is $O(M^{-4})$
  - d dimensions we need $N = M^d$ points
    the error is order $O(M^{-4}) = O(N^{-4/d})$

- An order - n scheme in 1 dimension
  is order - n/d d in d dimensions!

- In a statistical mechanics model with N particles we have $6N$-dimensional integrals ($3N$ positions and $3N$ momenta).
- Integration becomes extremely inefficient!

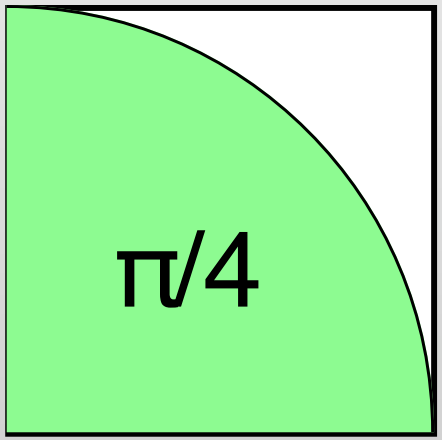
Ulam: the Monte Carlo Method

- What is the probability to win in Solitaire?
  - Ulam's answer: play it 100 times, count the number of wins and you have a pretty good estimate



Throwing stones into a pond

- How can we calculate π by throwing stones?
- Take a square surrounding the area we want to measure:

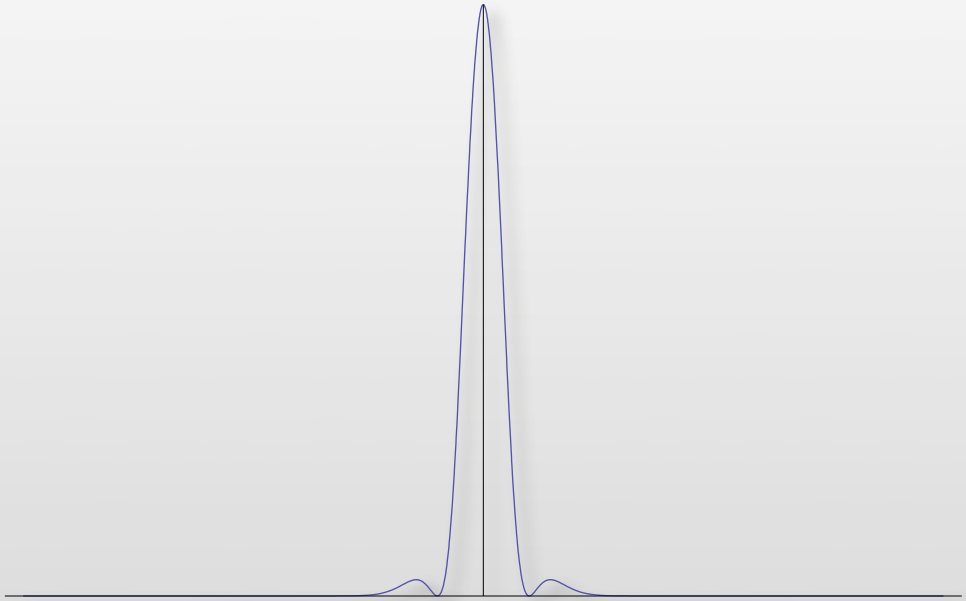


- Choose M pairs of random numbers (x, y) and count how many points (x, y) lie in the interesting area

Monte Carlo integration

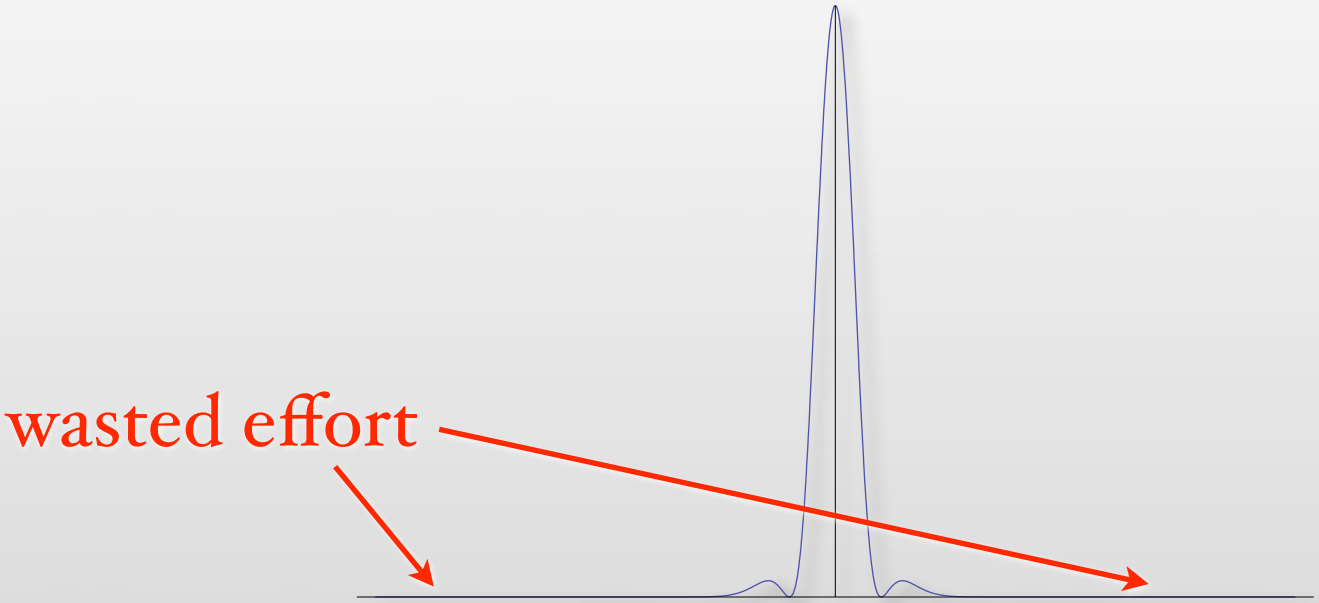
- Consider an integral $$\langle f \rangle = \int_\Omega f(\vec{x}) d\vec{x} \Big/ \int_\Omega d\vec{x}$$
- Instead of evaluating it at equally spaced points evaluate it at M points x_i chosen randomly in Ω:
$$\langle f \rangle \approx \frac{1}{M} \sum_{i=1}^{M} f(\vec{x}_i)$$
- The error is statistical:
$$\Delta = \sqrt{\frac{\mathrm{Var}\, f}{M}} \propto M^{-1/2}$$
$$\mathrm{Var}\, f = \langle f^2 \rangle - \langle f \rangle^2$$
- In d>8 dimensions Monte Carlo is better than Simpson!

Sharply peaked functions



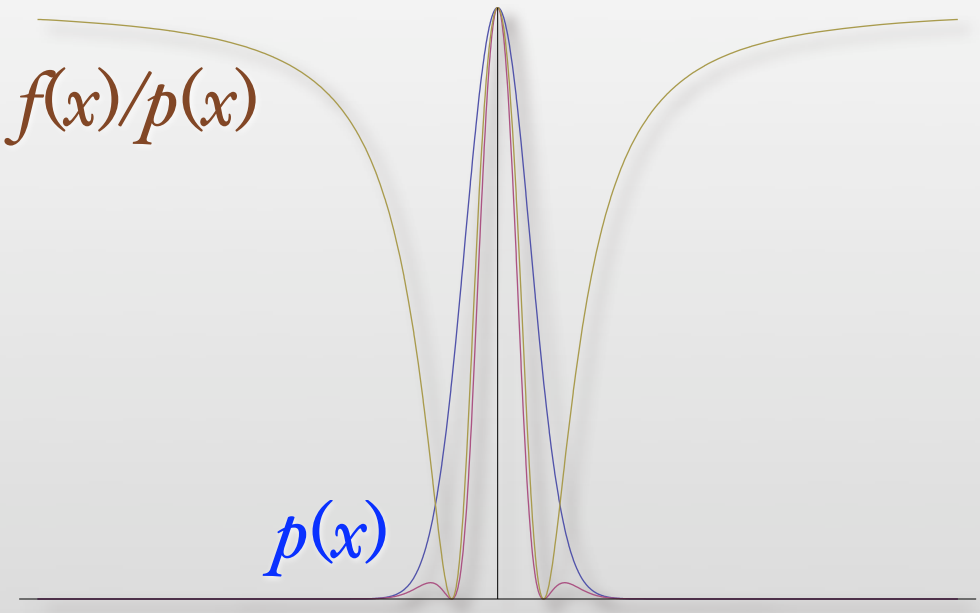
- In many cases a function is large only in a tiny region
- Lots of time wasted in regions where the function is small
- The sampling error is large since the variance is large

Sharply peaked functions



- In many cases a function is large only in a tiny region
- Lots of time wasted in regions where the function is small
- The sampling error is large since the variance is large

Importance sampling



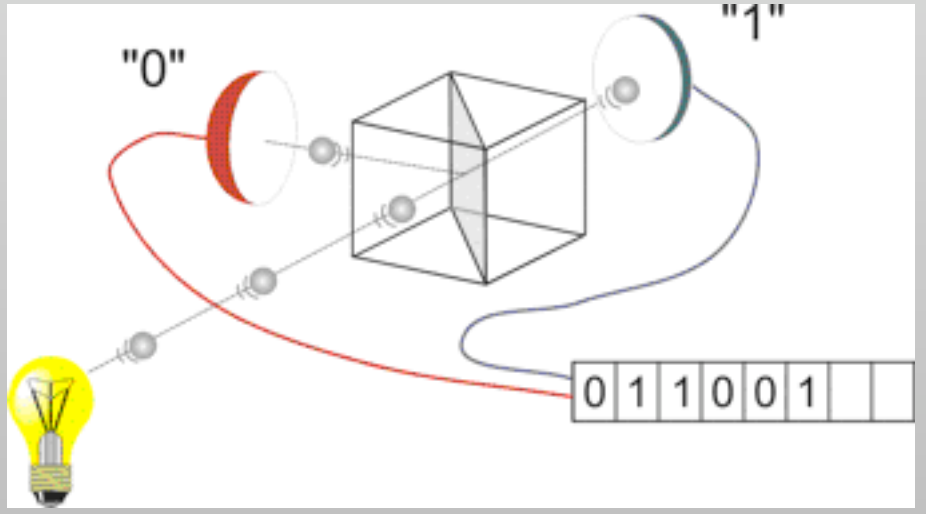
- Choose points not uniformly but with probability $p(x)$:

$$\langle f \rangle = \left\langle \frac{f}{p} \right\rangle_p := \int_\Omega \frac{f(\vec{x})}{p(\vec{x})} p(\vec{x}) d\vec{x} \Big/ \int_\Omega d\vec{x}$$

- The error is now determined by Var f/p
- Find p similar to f and such that p-distributed random numbers are easily available

2. Generating Random Numbers

Random numbers

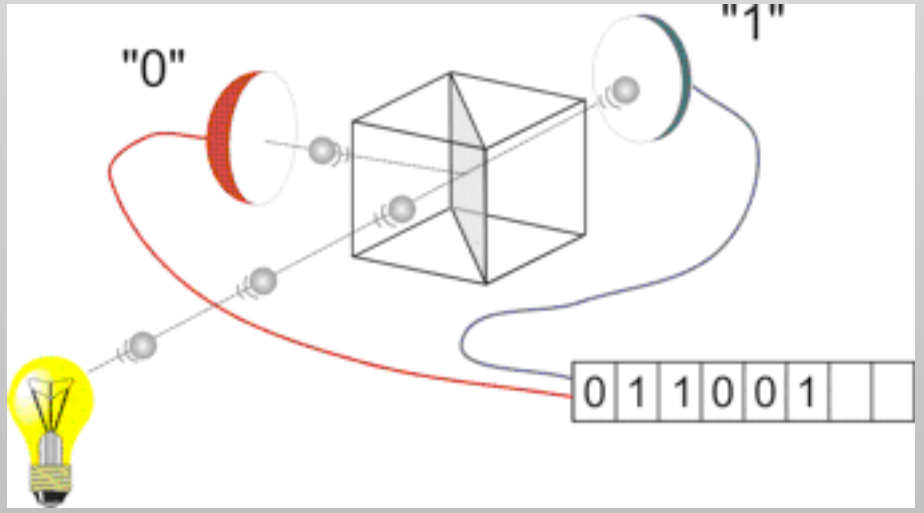


http://www.idquantique.com/

Random numbers

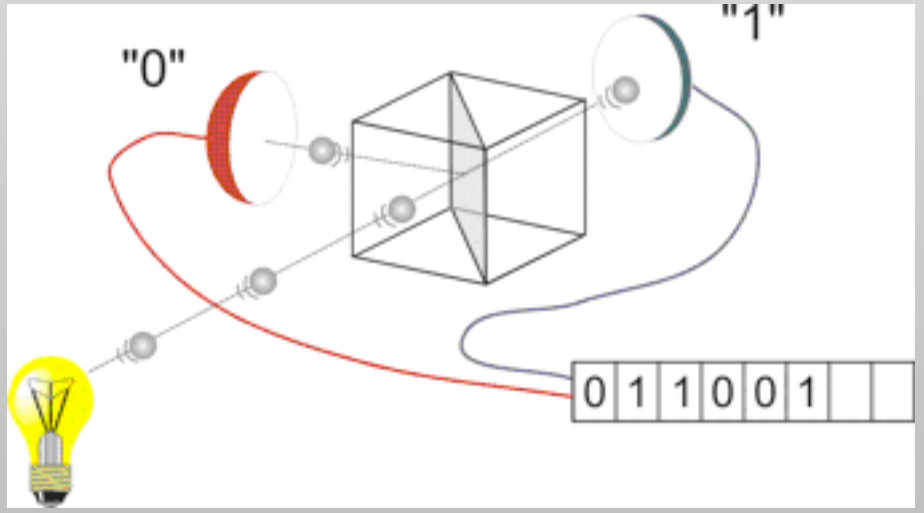
- Real random numbers are hard to obtain
  - classical chaos (atmospheric noise)
  - quantum mechanics

http://www.idquantique.com/



Random numbers

- Real random numbers are hard to obtain
  - classical chaos (atmospheric noise)
  - quantum mechanics
- Commercial products: quantum random number generators
  - based on photons and semi-transparent mirror
  - 4 Mbit/s from a USB device, too slow for most MC simulations



http://www.idquantique.com/

Pseudo Random numbers

Pseudo Random numbers

- Are generated by an algorithm

Pseudo Random numbers

- Are generated by an algorithm
- Not random at all, but completely deterministic

Pseudo Random numbers

- Are generated by an algorithm
- Not random at all, but completely deterministic
- Look nearly random however when algorithm is not known and may be good enough for our purposes

Pseudo Random numbers

- Are generated by an algorithm
- Not random at all, but completely deterministic
- Look nearly random however when algorithm is not known and may be good enough for our purposes
- Never trust pseudo random numbers however!

Linear congruential generators

- are of the simple form $x_{n+1}=f(x_n)$
- A good choice is the GGL generator

$$x_{n+1} = (ax_n + c) \bmod m$$

with a = 16807, c = 0, m = $2^{31}-1$

- quality depends sensitively on a,c,m

- Periodicity is a problem with such 32-bit generators
  - The sequence repeats identically after $2^{31}-1$ iterations
  - With 500 million numbers per second that is just 4 seconds!
  - Should not be used anymore!

Lagged Fibonacci generators

$$x_n = x_{n-p} \otimes x_{n-q} \bmod m$$

- Good choices are
  - (607,273,+)
  - (2281,1252,+)
  - (9689,5502,+)
  - (44497,23463,+)

- Seed blocks usually generated by linear congruential
- Has very long periods since large block of seeds
- A very fast generator: vectorizes and pipelines very well

More advanced generators

- As well-established generators fail new tests, better and better generators get developed
  - Mersenne twister (Matsumoto & Nishimura, 1997)
  - Well generator (Panneton and L'Ecuyer , 2004)

- Based on lagged Fibonacci generators, improved with random bit shuffles

- Deep number theory enters the design of these generators

Pierre L'Ecuyer
(Univ. de Montréal)

Are these numbers really random?

Are these numbers really random?

- No!

Are these numbers really random?

- No!
- Are they random enough?
  - Maybe?

Are these numbers really random?

- No!
- Are they random enough?
  - Maybe?
- Statistical tests for distribution and correlations

Are these numbers really random?

- No!
- Are they random enough?
  - Maybe?
- Statistical tests for distribution and correlations
- Are these tests enough?
  - No! Your calculation could depend in a subtle way on hidden correlations!

Are these numbers really random?

- No!
- Are they random enough?
  - Maybe?
- Statistical tests for distribution and correlations
- Are these tests enough?
  - No! Your calculation could depend in a subtle way on hidden correlations!
- What is the ultimate test?
  - Run your simulation with various random number generators and compare the results

Marsaglia's diehard tests

- **Birthday spacings:** Choose random points on a large interval. The spacings between the points should be asymptotically Poisson distributed. The name is based on the birthday paradox.
- **Overlapping permutations:** Analyze sequences of five consecutive random numbers. The 120 possible orderings should occur with statistically equal probability.
- **Ranks of matrices:** Select some number of bits from some number of random numbers to form a matrix over {0,1}, then determine the rank of the matrix. Count the ranks.
- **Monkey tests:** Treat sequences of some number of bits as "words". Count the overlapping words in a stream. The number of "words" that don't appear should follow a known distribution. The name is based on the infinite monkey theorem.
- **Count the 1s:** Count the 1 bits in each of either successive or chosen bytes. Convert the counts to "letters", and count the occurrences of five-letter "words".
- **Parking lot test:** Randomly place unit circles in a 100 x 100 square. If the circle overlaps an existing one, try again. After 12,000 tries, the number of successfully "parked" circles should follow a certain normal distribution.

Marsaglia's diehard tests (cont.)

- **Minimum distance test:** Randomly place 8,000 points in a 10,000 x 10,000 square, then find the minimum distance between the pairs. The square of this distance should be exponentially distributed with a certain mean.
- **Random spheres test:** Randomly choose 4,000 points in a cube of edge 1,000. Center a sphere on each point, whose radius is the minimum distance to another point. The smallest sphere's volume should be exponentially distributed with a certain mean.
- **The squeeze test:** Multiply 231 by random floats on [0,1) until you reach 1. Repeat this 100,000 times. The number of floats needed to reach 1 should follow a certain distribution.
- **Overlapping sums test:** Generate a long sequence of random floats on [0,1). Add sequences of 100 consecutive floats. The sums should be normally distributed with characteristic mean and sigma.
- **Runs test:** Generate a long sequence of random floats on [0,1). Count ascending and descending runs. The counts should follow a certain distribution.
- **The craps test:** Play 200,000 games of craps, counting the wins and the number of throws per game. Each count should follow a certain distribution.

Non-uniform random numbers

- we found ways to generate pseudo random numbers u in the interval [0,1[
- How do we get other uniform distributions?
  - uniform x in [a,b[: $x = a+(b-a)\,u$
- Other distributions:
  - Inversion of integrated distribution
  - Rejection method

Non-uniform distributions

- How can we get a random number x distributed with $f(x)$ in the interval $[a,b[$ from a uniform random number u?
- Look at probabilities:

$$P[x < y] = \int_a^y f(t)\,dt =: F(y) \equiv P[u < F(y)]$$

$$\Rightarrow x = F^{-1}(u)$$

- This method is feasible if the integral can be inverted easily
  - exponential distribution $f(x)=\lambda \exp(-\lambda x)$
  - can be obtained from uniform by $x=-1/\lambda \ln(1-u)$

Normally distributed numbers

- The normal distribution

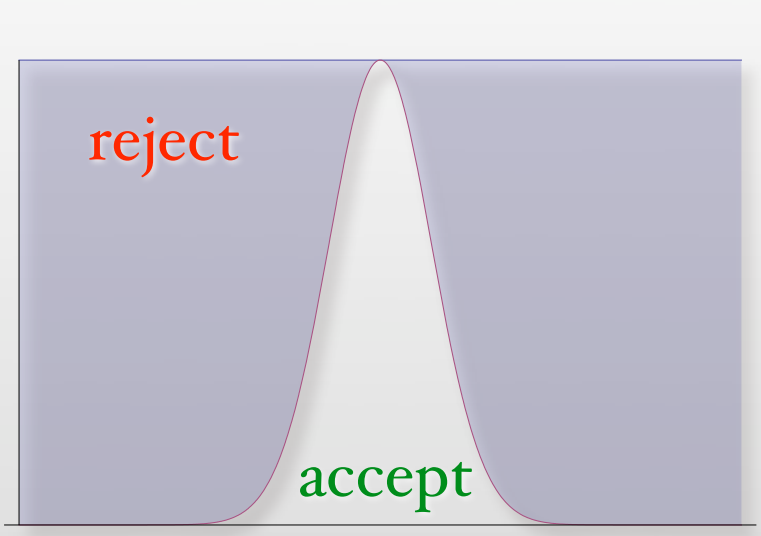
$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2\right)$$

- cannot easily be integrated in one dimension but can be easily integrated in 2 dimensions!
- We can obtain two normally distributed numbers from two uniform ones (Box-Muller method)

$$n_1 = \sqrt{-2\ln(1-u_1)}\sin u_2$$

$$n_2 = \sqrt{-2\ln(1-u_1)}\cos u_2$$

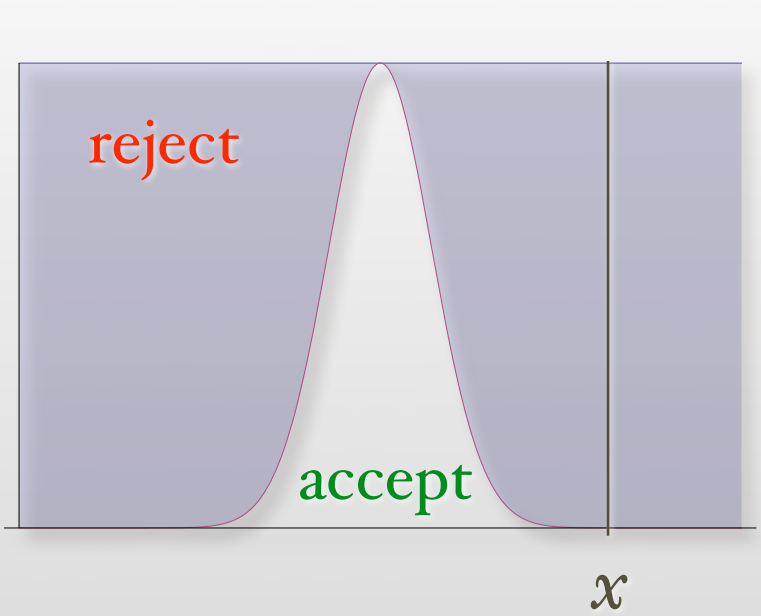
Rejection method (von Neumann)



- Look for a simple distribution h that bounds f: $f(x) < \lambda h(x)$
  - Choose an h-distributed number x
  - Choose a uniform random number number $0 \le u < 1$
  - Accept x if $u < f(x)/\lambda h(x)$,
    otherwise reject x and get a new pair (x,u)

- Needs a good guess h to be efficient, numerical inversion of integral might be faster if no suitable h can be found

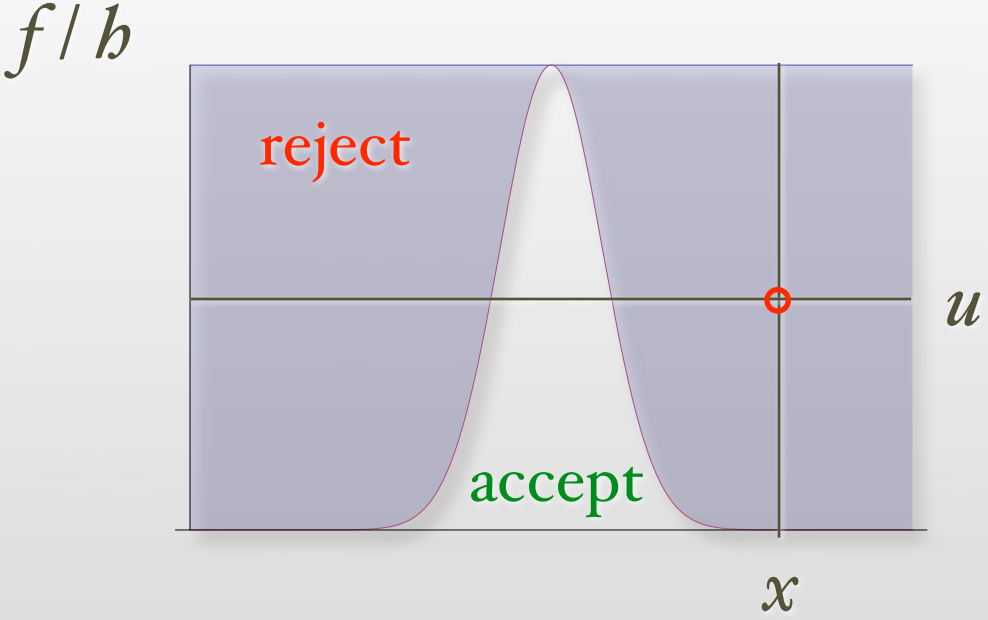
Rejection method (von Neumann)



- Look for a simple distribution h that bounds f: $f(x) < \lambda h(x)$
  - Choose an h-distributed number x
  - Choose a uniform random number number $0 \le u < 1$
  - Accept x if $u < f(x)/\lambda h(x)$,
    otherwise reject x and get a new pair (x,u)

- Needs a good guess h to be efficient, numerical inversion of integral might be faster if no suitable h can be found

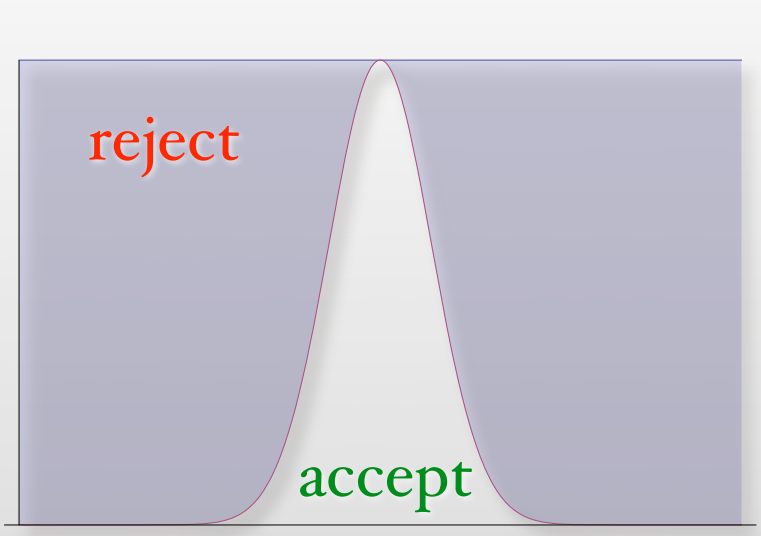
Rejection method (von Neumann)



- Look for a simple distribution h that bounds f: $f(x) < \lambda h(x)$
  - Choose an h-distributed number x
  - Choose a uniform random number number $0 \le u < 1$
  - Accept x if $u < f(x)/\lambda h(x)$,
    otherwise reject x and get a new pair (x,u)

- Needs a good guess h to be efficient, numerical inversion of integral might be faster if no suitable h can be found

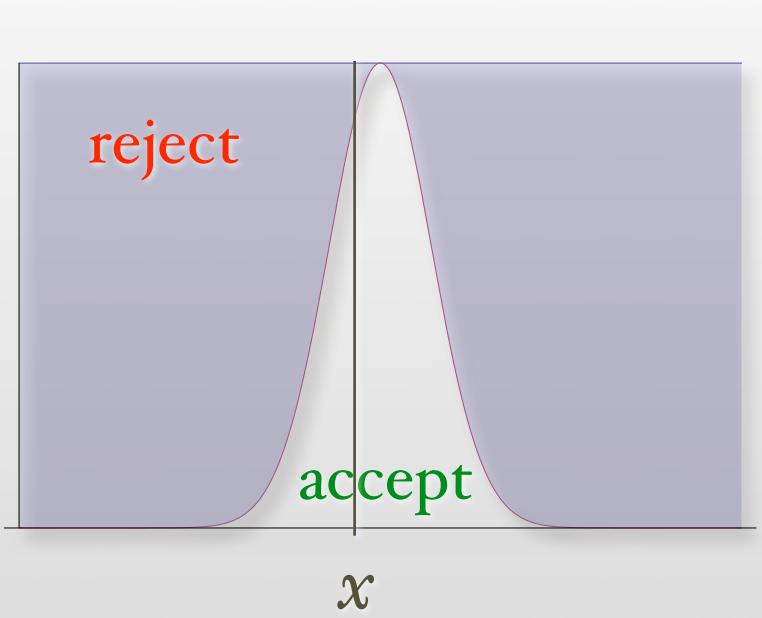
Rejection method (von Neumann)



- Look for a simple distribution h that bounds f: $f(x) < \lambda h(x)$
  - Choose an h-distributed number x
  - Choose a uniform random number number $0 \le u < 1$
  - Accept x if $u < f(x)/\lambda h(x)$,
    otherwise reject x and get a new pair (x,u)

- Needs a good guess h to be efficient, numerical inversion of integral might be faster if no suitable h can be found

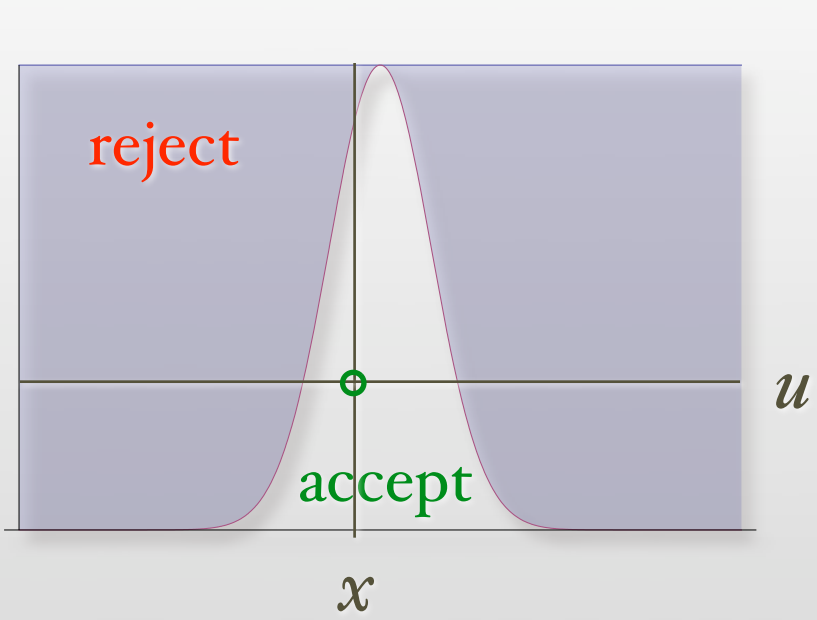
Rejection method (von Neumann)



- Look for a simple distribution h that bounds f: $f(x) < \lambda h(x)$
  - Choose an h-distributed number x
  - Choose a uniform random number number $0 \le u < 1$
  - Accept x if $u < f(x)/\lambda h(x)$,
    otherwise reject x and get a new pair (x,u)

- Needs a good guess h to be efficient, numerical inversion of integral might be faster if no suitable h can be found

Rejection method (von Neumann)



- Look for a simple distribution h that bounds f: $f(x) < \lambda h(x)$
  - Choose an h-distributed number x
  - Choose a uniform random number number $0 \le u < 1$
  - Accept x if $u < f(x)/\lambda h(x)$,
    otherwise reject x and get a new pair (x,u)

- Needs a good guess h to be efficient, numerical inversion of integral might be faster if no suitable h can be found

3. The Metropolis Algorithm

Monte Carlo for classical systems

- Evaluate phase space integral by importance sampling

$$\langle A \rangle = \frac{\int_{\Omega} A(c)p(c)dc}{\int_{\Omega} p(c)dc} \longrightarrow \langle A \rangle \approx \overline{A} = \frac{1}{M}\sum_{i=1}^{M} A_{c_i}$$

- Pick configurations with the correct Boltzmann weight

$$P[c] = \frac{p(c)}{Z} = \frac{\exp(-\beta E(c))}{Z}$$

- But how do we create configurations with that distribution? The key problem in statistical mechanics!

GUEST EDITORS' INTRODUCTION

the Top 10 Algorithms

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

computing in SCIENCE & ENGINEERING

GUEST EDITORS' INTRODUCTION

the Top 10 Algorithms

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

computing in SCIENCE & ENGINEERING

The Metropolis Algorithm (1953)

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

I. INTRODUCTION

THE purpose of this paper is to describe a general method, suitable for fast electronic computing machines, of calculating the properties of any substance which may be considered as composed of interacting individual molecules. Classical statistics is assumed,

II. THE GENERAL METHOD FOR AN ARBITRARY POTENTIAL BETWEEN THE PARTICLES

In order to reduce the problem to a feasible size for numerical work, we can, of course, consider only a finite number of particles. This number N may be as high as several hundred. Our system consists of a square† con-

The Metropolis Algorithm (1953)

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

I. INTRODUCTION

THE purpose of this paper is to describe a general method, suitable for fast electronic computing machines, of calculating the properties of any substance which may be considered as composed of interacting individual molecules. Classical statistics is assumed,

II. THE GENERAL METHOD FOR AN ARBITRARY POTENTIAL BETWEEN THE PARTICLES

In order to reduce the problem to a feasible size for numerical work, we can, of course, consider only a finite number of particles. This number N may be as high as several hundred. Our system consists of a square† con-

Markov chain Monte Carlo

- Instead of drawing independent samples c_i we build a Markov chain

$$c_1 \to c_2 \to \ldots \to c_i \to c_{i+1} \to \ldots$$

- Transition probabilities $W_{x,y}$ for transition $x \to y$ need to satisfy:
  - **Normalization:** $\sum_y W_{x,y} = 1$
  - **Ergodicity:** any configuration reachable from any other

$$\forall x, y \; \exists n \; : \; \left(W^n\right)_{x,y} \neq 0$$

  - **Balance:** the distribution should be stationary

$$0 = \frac{d}{dt} p(x) = \sum_y p(y) W_{y,x} - \sum_y p(x) W_{x,y} \Rightarrow p(x) = \sum_y p(y) W_{y,x}$$

  - Detailed balance is sufficient but not necessary for balance

$$\frac{W_{x,y}}{W_{y,x}} = \frac{p(y)}{p(x)}$$

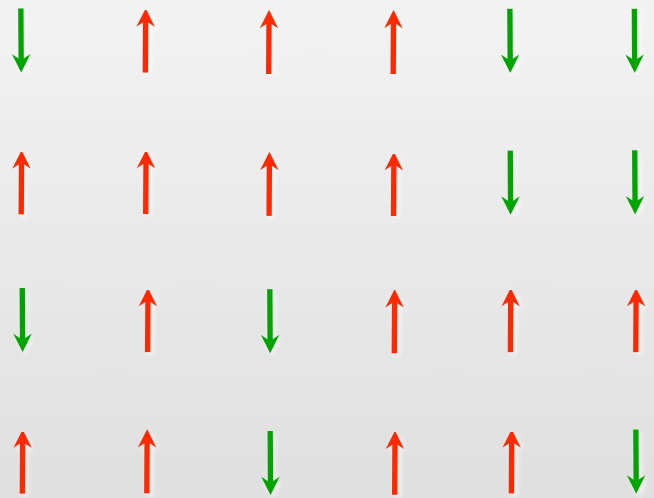
The Metropolis algorithm

- Teller's proposal was to use rejection sampling:
  - Propose a change with an a-priori proposal rate $A_{x,y}$
  - Accept the proposal with a probability $P_{x,y}$
  - The total transition rate is $W_{x,y} = A_{x,y} P_{x,y}$
- The choice

$$P_{x,y} = \min\left[1, \frac{A_{y,x} p(y)}{A_{x,y} p(x)}\right]$$

satisfies detailed balance and was first proposed by Metropolis *et al*

Metropolis algorithm for the Ising model

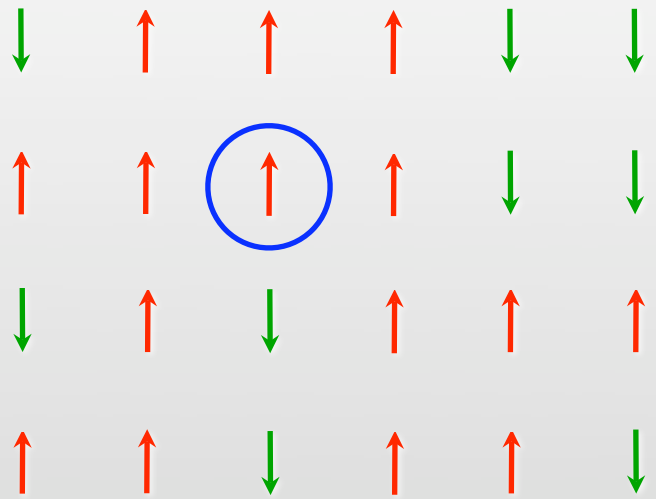


1. Pick a random spin and propose to flip it

2. Accept the flip with probability $P = \min\left[1, e^{-(E_{new} - E_{old})/T}\right]$

3. Perform a measurement independent of whether the proposed flip was accepted or rejected!

Metropolis algorithm for the Ising model

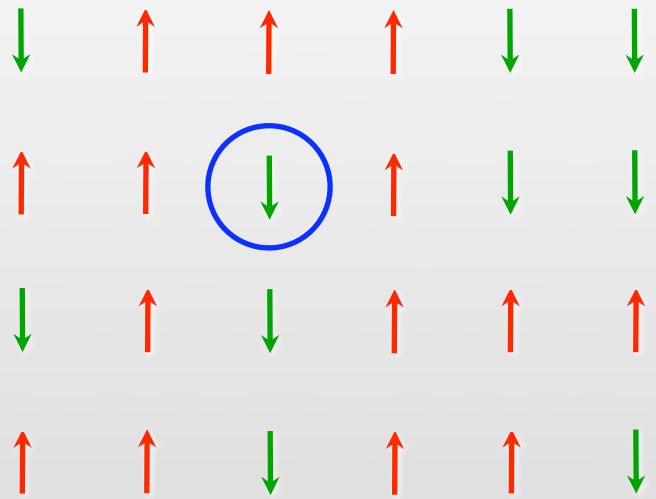


1. Pick a random spin and propose to flip it

2. Accept the flip with probability $P = \min\left[1, e^{-(E_{new} - E_{old})/T}\right]$

3. Perform a measurement independent of whether the proposed flip was accepted or rejected!

Metropolis algorithm for the Ising model



1. Pick a random spin and propose to flip it

2. Accept the flip with probability $P = \min\left[1, e^{-(E_{new} - E_{old})/T}\right]$

3. Perform a measurement independent of whether the proposed flip was accepted or rejected!

Equilibration

- Starting from a random initial configuration it takes a while to reach the equilibrium distribution
- The desired equilibrium distribution is a left eigenvector with eigenvalue 1 (this is just the balance condition)

$$p(x) = \sum_y p(y) W_{y,x}$$

- Convergence is controlled by the second largest eigenvalue

$$p(x,t) = p(x) + O(\exp(-\lambda_2 t))$$

- We need to run the simulation for a while to equilibrate and only then start measuring

4. Monte Carlo Error Analysis

Monte Carlo error analysis

- The simple formula $\Delta A = \sqrt{\frac{\mathrm{Var}\, A}{M}}$

  is valid only for independent samples

- The Metropolis algorithm gives us correlated samples! The number of independent samples is reduced

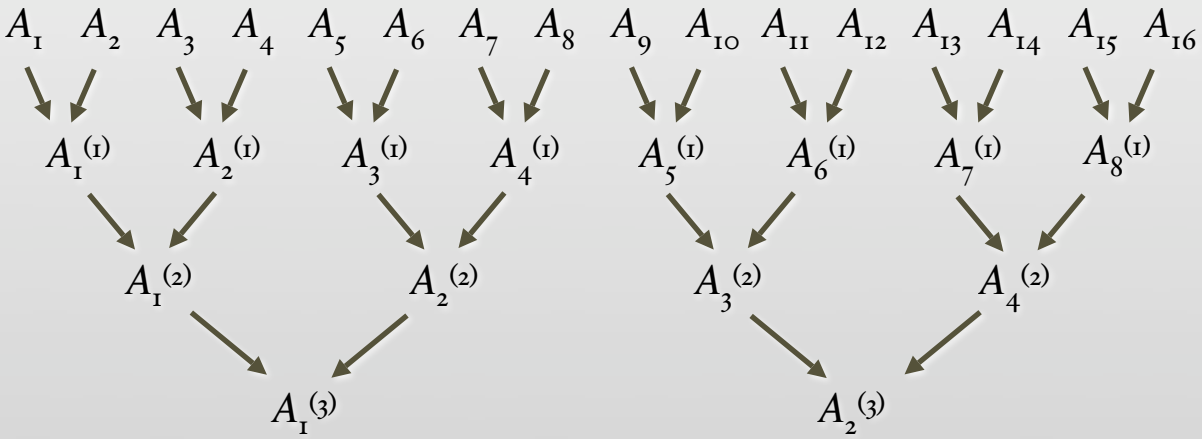
$$\Delta A = \sqrt{\frac{\mathrm{Var}\, A}{M}\left(1 + 2\tau_A\right)}$$

- The autocorrelation time is defined by

$$\tau_A = \frac{\sum_{t=1}^{\infty}\left(\langle A_{i+t} A_i \rangle - \langle A \rangle^2\right)}{\mathrm{Var}\, A}$$

Binning analysis

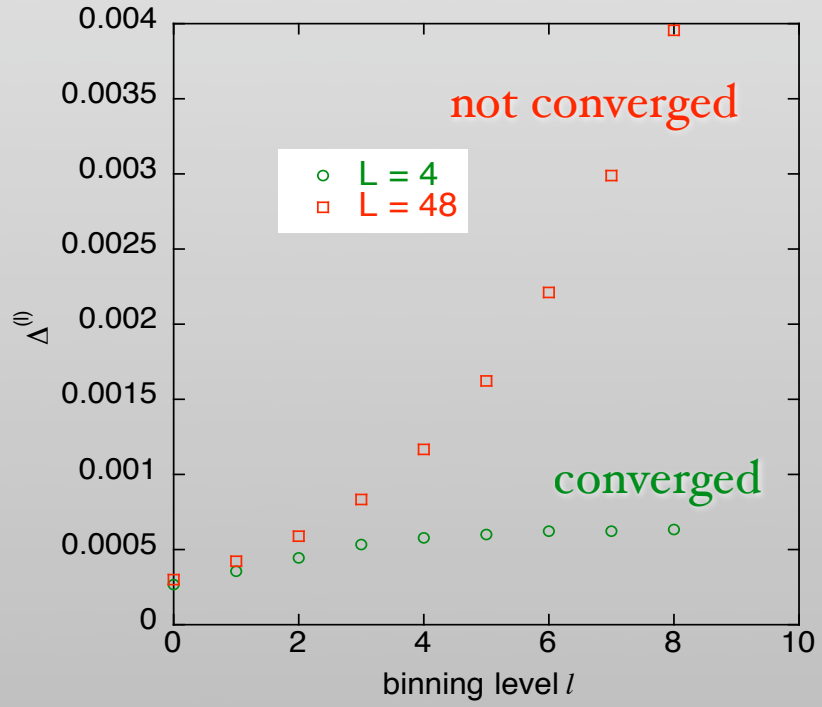
- Take averages of consecutive measurements: averages become less correlated and naive error estimates converge to real error



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

$$\Delta^{(l)} = \sqrt{\operatorname{Var} A^{(l)} / M^{(l)}} \xrightarrow{l\to\infty} \Delta = \sqrt{(1+2\tau_A)\operatorname{Var} A / M}$$

$$\tau_A = \lim_{l\to\infty} \frac{1}{2}\left(\frac{2^l \operatorname{Var} A^{(l)}}{\operatorname{Var} A^{(0)}} - 1\right)$$



a smart implementation needs only
O(log(*N*)) memory for *N* measurements

Seeing convergence in ALPS

- Look at the ALPS output in the first hands-on session
- 48 x 48 Ising model at the critical point
  - local updates:

Name	Count	Mean	Error	Tau	Method
Susceptibility	52529	401.08	11.3 not converged	99.1	binning

  - cluster updates:

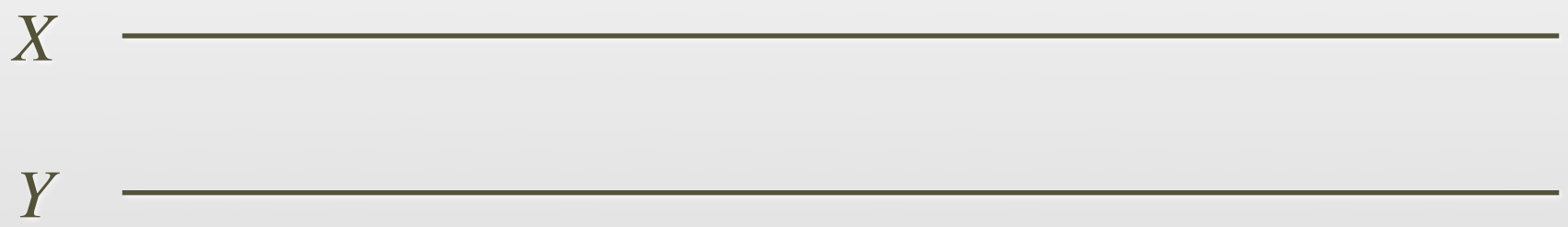
Name	Count	Mean	Error	Tau	Method
Susceptibility	113433	421.642	1.57	0.821	binning

Correlated quantities

- How do we calculate the errors of functions of correlated measurements?
  - specific heat $$c_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2}$$
  - Binder cumulant ratio $$U = \frac{\langle m^4 \rangle}{\langle m^2 \rangle^2}$$
- The naïve way of assuming uncorrelated errors is wrong!
- It is not even enough to calculate all crosscorrelations due to nonlinearities except if the errors are tiny!

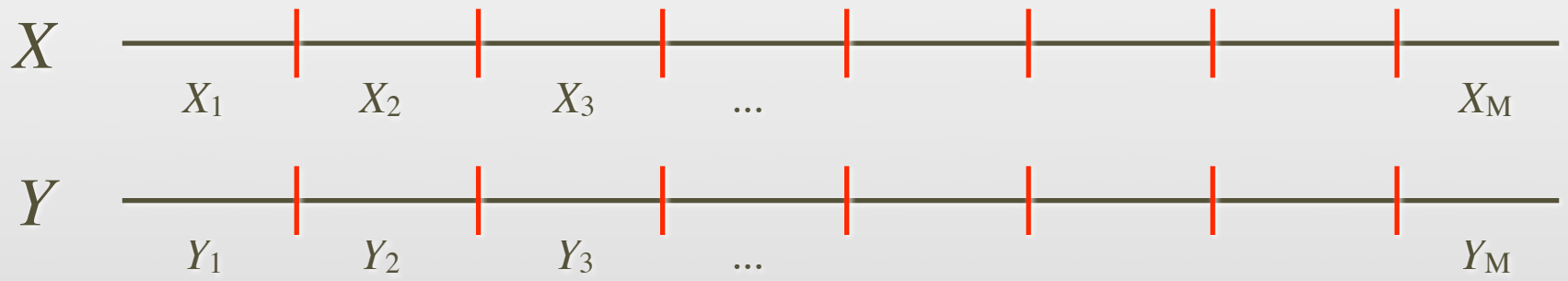
Splitting the time series

Simplest idea: split the time series and evaluate for each segment



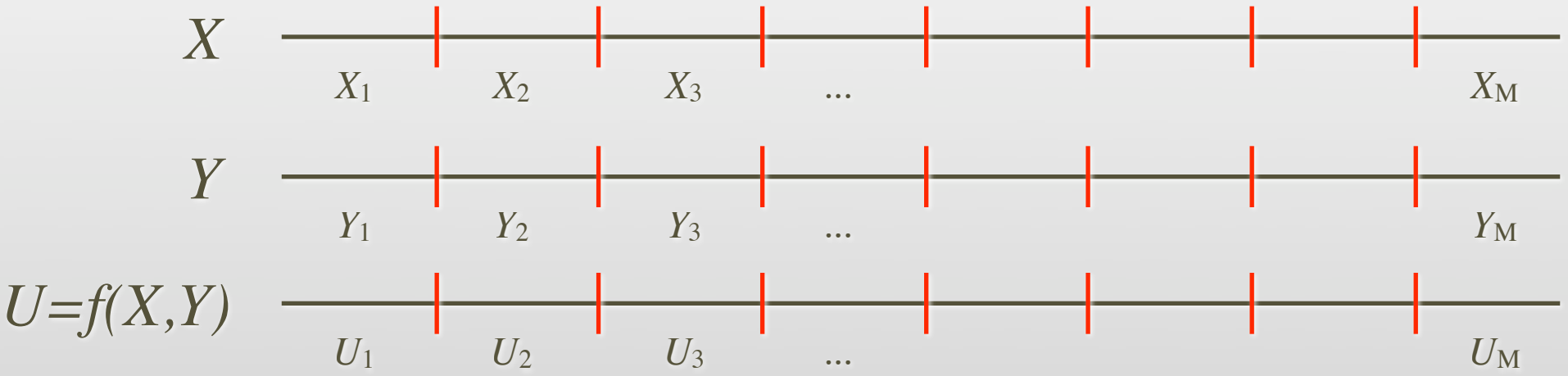
Splitting the time series

Simplest idea: split the time series and evaluate for each segment



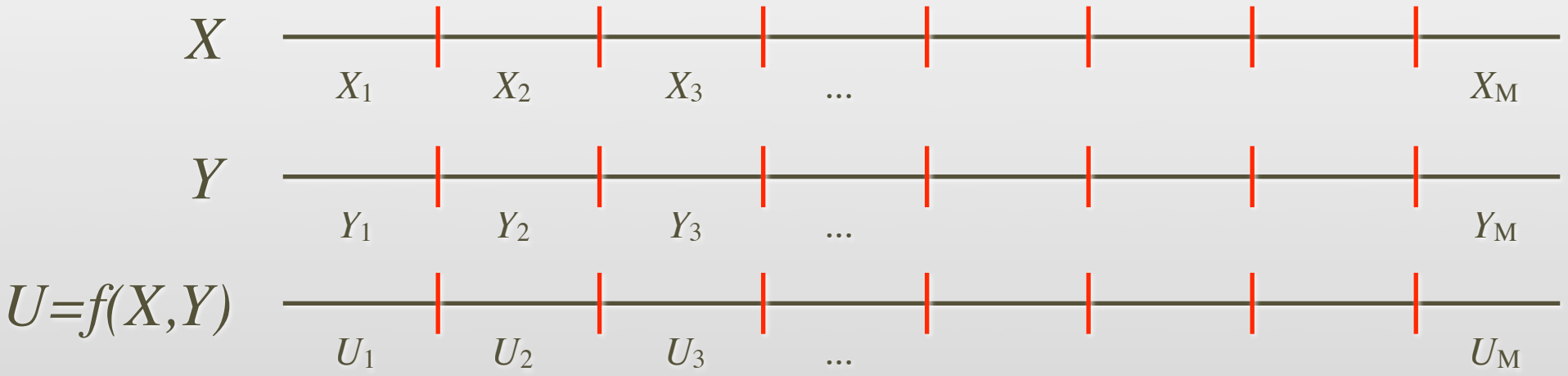
Splitting the time series

Simplest idea: split the time series and evaluate for each segment



Splitting the time series

Simplest idea: split the time series and evaluate for each segment

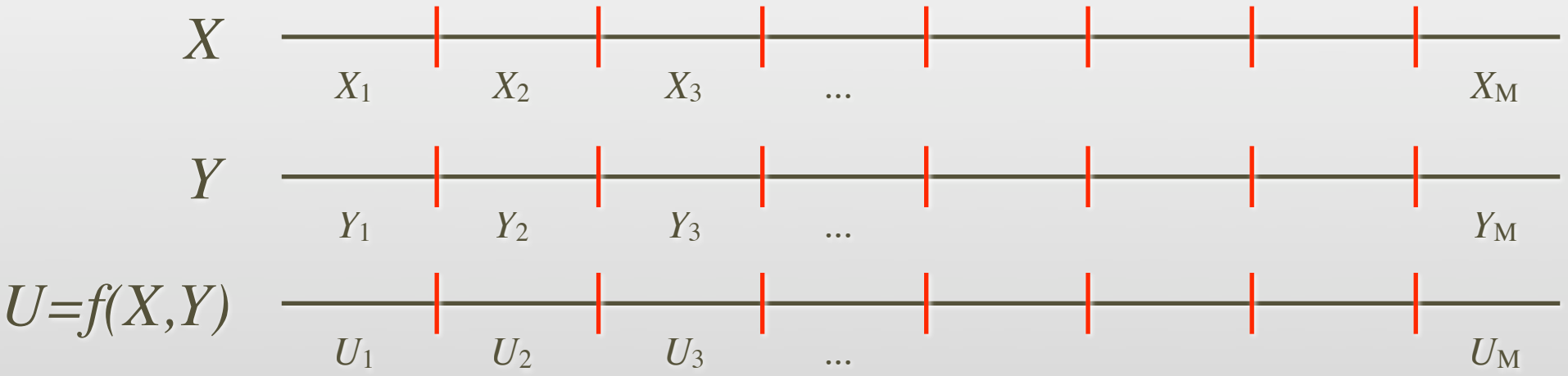


$$\langle U \rangle \approx \overline{U} = \frac{1}{M} \sum_{i=1}^{M} U_i$$

$$\Delta U \approx \sqrt{\frac{1}{M(M-1)} \sum_{i-1}^{M} \left(U_i - \overline{U}\right)^2}$$

Splitting the time series

Simplest idea: split the time series and evaluate for each segment



$$\langle U \rangle \approx \overline{U} = \frac{1}{M}\sum_{i=1}^{M} U_i$$

$$\Delta U \approx \sqrt{\frac{1}{M(M-1)}\sum_{i-1}^{M}\left(U_i - \overline{U}\right)^2}$$

Problem: can be unstable and noisy for nonlinear functions such as X/Y

Jackknife-analysis

Evaluate the function on all and all but one segment

Jackknife-analysis

Evaluate the function on all and all but one segment

$$U_0 = \frac{1}{M}\sum_{i=1}^{M} f(X_i,Y_i)$$

Jackknife-analysis

Evaluate the function on all and all but one segment

$$U_0 = \frac{1}{M}\sum_{i=1}^{M} f(X_i,Y_i)$$

$$U_1 = \frac{1}{M-1}\sum_{i=2}^{M} f(X_i,Y_i)$$

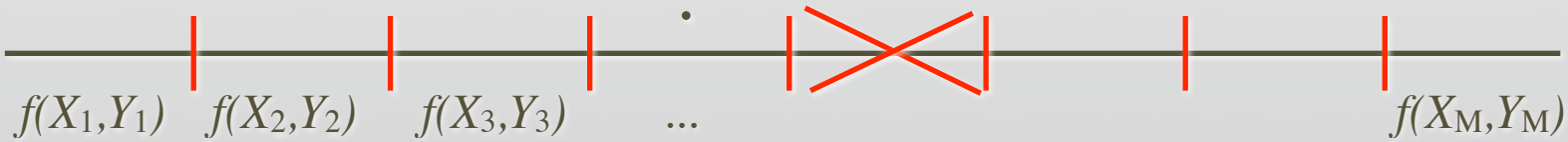
Jackknife-analysis

Evaluate the function on all and all but one segment

$$U_0 = \frac{1}{M}\sum_{i=1}^{M} f(X_i,Y_i)$$

$$U_1 = \frac{1}{M-1}\sum_{i=2}^{M} f(X_i,Y_i)$$

$$U_j = \frac{1}{M-1}\sum_{\substack{i=1 \\ i\neq j}}^{M} f(X_i,Y_i)$$



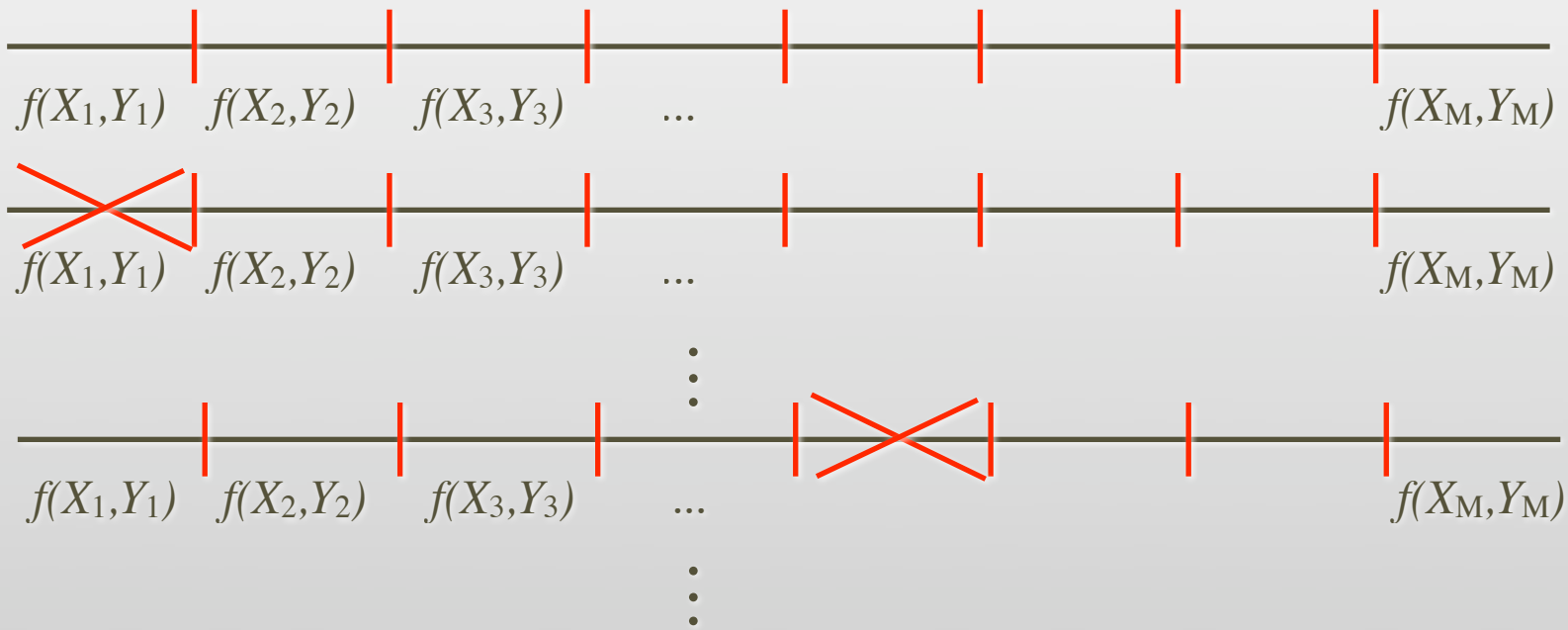
Jackknife-analysis

Evaluate the function on all and all but one segment

$$U_0 = \frac{1}{M}\sum_{i=1}^{M} f(X_i,Y_i)$$

$$U_1 = \frac{1}{M-1}\sum_{i=2}^{M} f(X_i,Y_i)$$

$$U_j = \frac{1}{M-1}\sum_{\substack{i=1 \\ i\neq j}}^{M} f(X_i,Y_i)$$



$$\langle U \rangle \approx U_0 - (M-1)(\overline{U} - U_0)$$

$$\overline{U} = \frac{1}{M}\sum_{i=1}^{M} U_i$$

$$\Delta U \approx \sqrt{\frac{M-1}{M}\sum_{i-1}^{M}\left(U_i - \overline{U}\right)^2}$$

ALPS.Alea library

- The ALPS class library implements reliable error analysis
  - Adding a measurement:

```
alps::RealObservable mag;
…
mag << new_value;
```

  - Evaluating measurements

```
std::cout << mag.mean() << " +/- " << mag.error();
std::cout "Autocorrelation time: " << mag.tau();
```

- Correlated quantities?
  - Such as in Binder cumulant ratios $\langle m^4 \rangle / \langle m^2 \rangle^2$
  - ALPS library uses jackknife analysis to get correct errors

```
alps::RealObsEvaluator binder = mag4/(mag2*mag2);
std::cout << binder.mean() << " +/- " << binder.error();
```

5. Critical slowing down, cluster updates and Wang-Landau sampling

Autocorrelation effects

- The Metropolis algorithm creates a Markov chain

$$c_1 \to c_2 \to \ldots \to c_i \to c_{i+1} \to \ldots$$

- successive configurations are correlated, leading to an increased statistical error

$$\Delta A = \sqrt{\left\langle \left(\overline{A} - \langle A \rangle \right)^2 \right\rangle} = \sqrt{\frac{\mathrm{Var}\, A}{M} (1 + 2\tau_A)}$$

- *Critical slowing down* at second order phase transition

$$\tau \propto L^2$$

- *Exponential tunneling problem* at first order phase transition

$$\tau \propto \exp(L^{d-1})$$

From local to cluster updates

- Energy of configurations in Ising model
  - $-J$ if parallel: ↑ ↑ ↓ ↓
  - $+J$ if anti-parallel: ↓ ↑ ↓ ↑
- Probability for flip
  - Anti-parallel: flipping lowers energy, always accepted

    ↓ ↑ → ↑ ↑ $\quad \Delta E = -2J \Rightarrow P = \min\left(1, e^{-2\Delta E/T}\right) = 1$
  - Parallel:

    ↑ ↑ → ↓ ↑ $\quad \Delta E = +2J \Rightarrow P = \min\left(1, e^{-2\Delta E/T}\right) = \exp(-2\beta J)$

    no change with probability $1 - \exp(-2\beta J)$!!!

From local to cluster updates

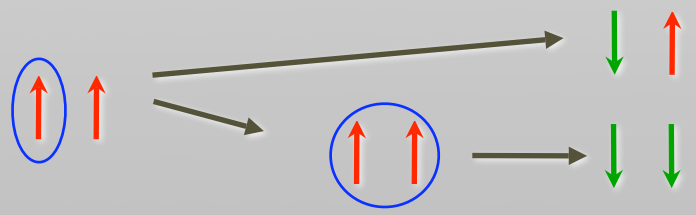
- Energy of configurations in Ising model
  - $-J$ if parallel: ↑ ↑ ↓ ↓
  - $+J$ if anti-parallel: ↓ ↑ ↓ ↑
- Probability for flip
  - Anti-parallel: flipping lowers energy, always accepted

    ↓ ↑ → ↑ ↑ $\quad \Delta E = -2J \Rightarrow P = \min\left(1, e^{-2\Delta E/T}\right) = 1$
  - Parallel:

    ↑ ↑ → ↓ ↑ $\quad \Delta E = +2J \Rightarrow P = \min\left(1, e^{-2\Delta E/T}\right) = \exp(-2\beta J)$

    no change with probability $1 - \exp(-2\beta J)$!!!

Alternative: flip both!



$P = \exp(-2J/T)$

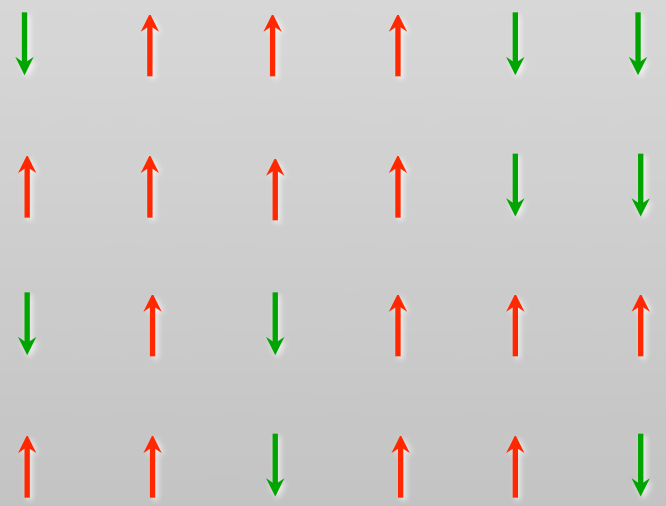
$P = 1 - \exp(-2J/T)$

Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates

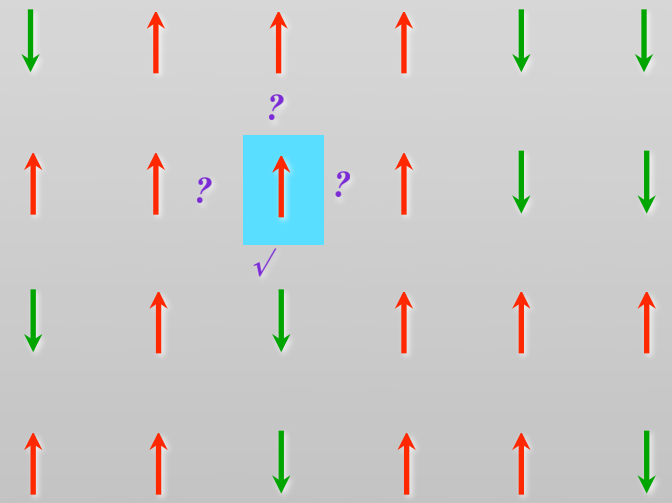
Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Swendsen-Wang Cluster-Updates

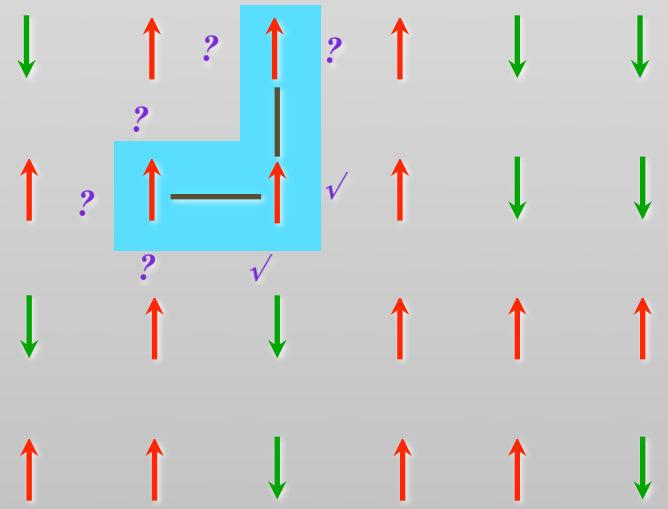
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Shall we flip neighbor?

Swendsen-Wang Cluster-Updates

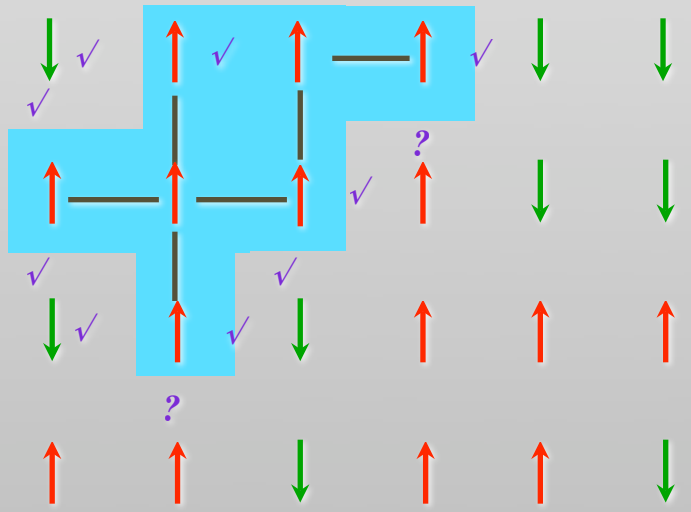
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Shall we flip neighbor?

Swendsen-Wang Cluster-Updates

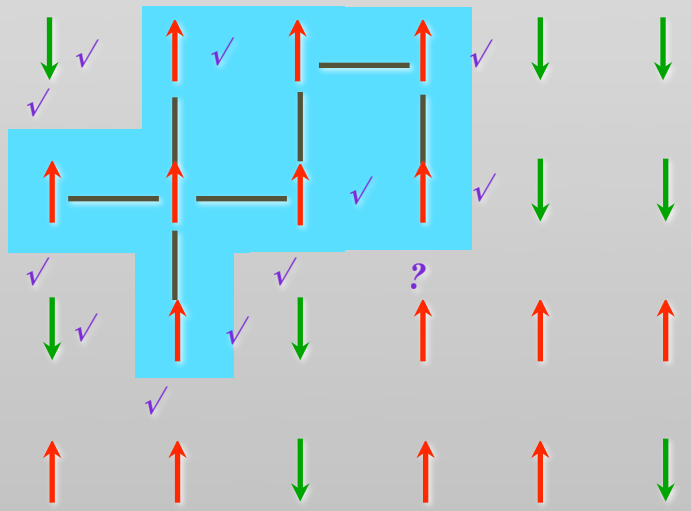
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Shall we flip neighbor?

Swendsen-Wang Cluster-Updates

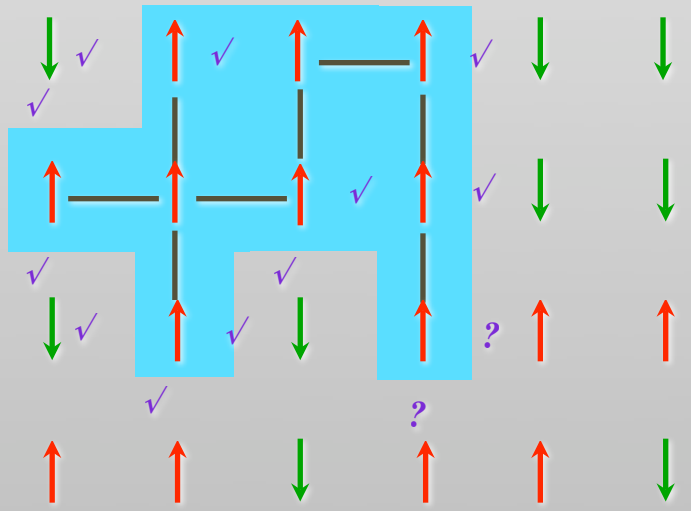
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Shall we flip neighbor?

Swendsen-Wang Cluster-Updates

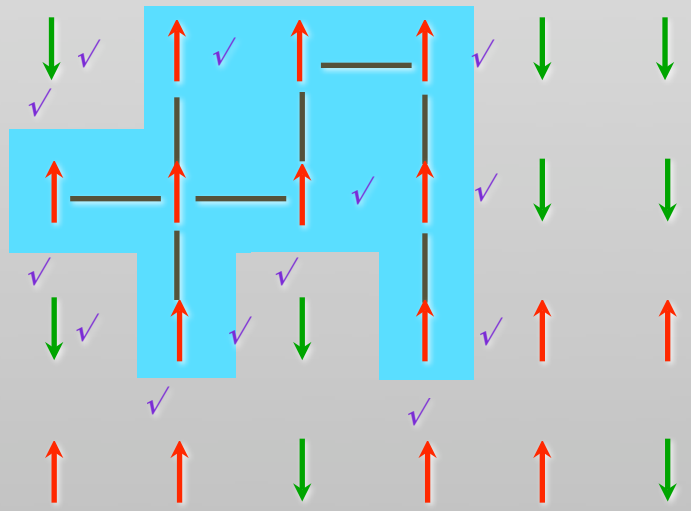
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Shall we flip neighbor?

Swendsen-Wang Cluster-Updates

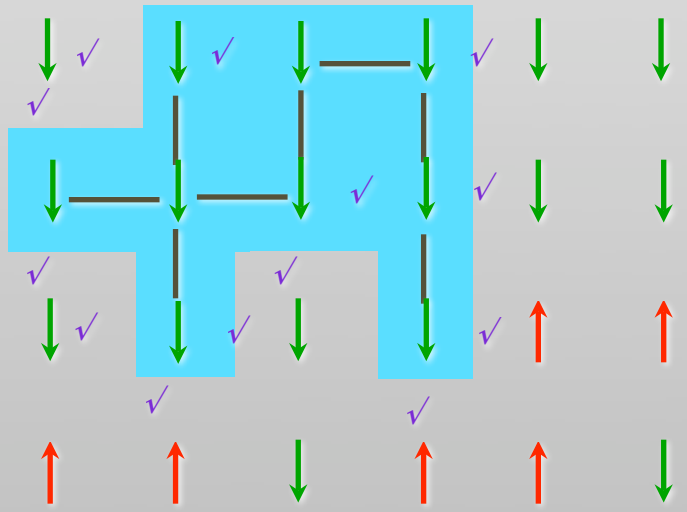
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Shall we flip neighbor?

Swendsen-Wang Cluster-Updates

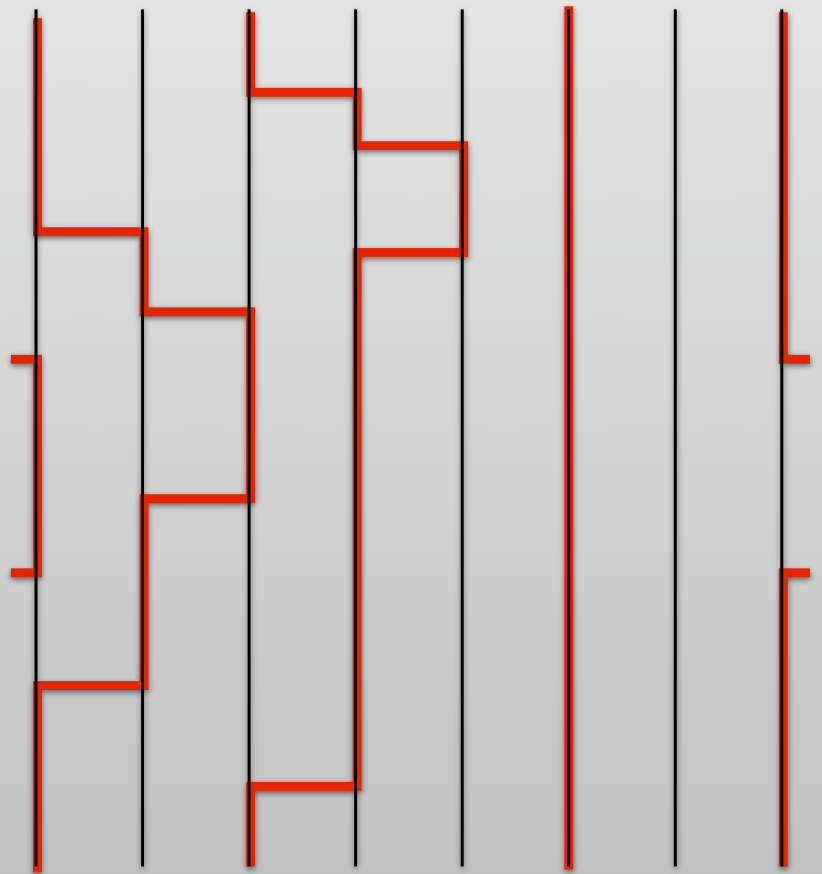
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Done building cluster
Flip all spins in cluster

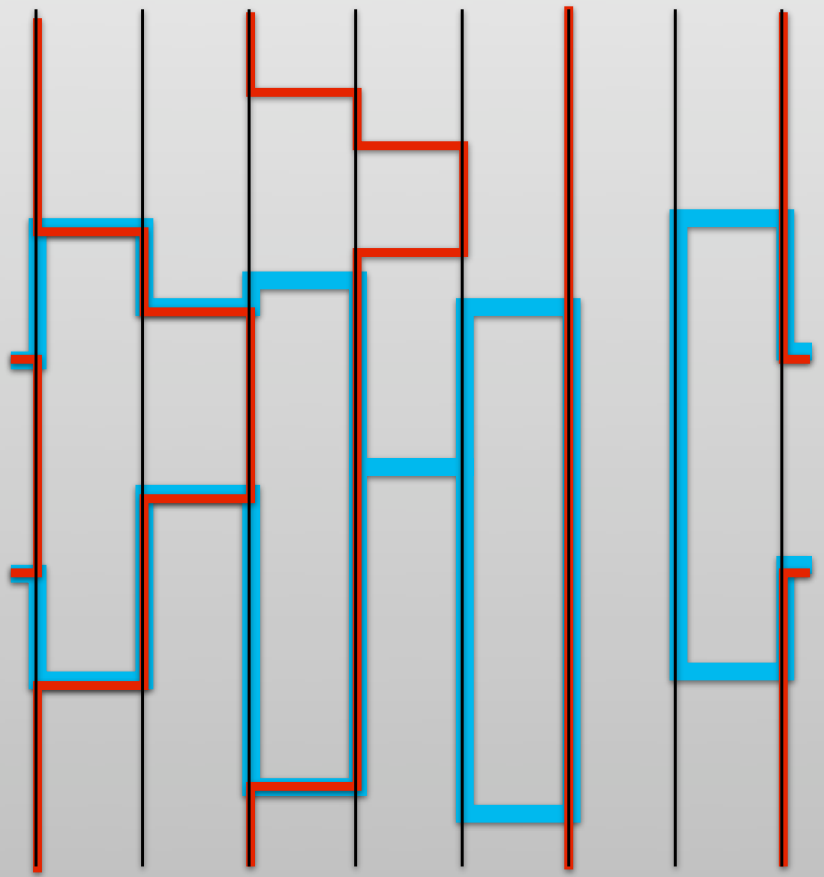
The loop algorithm (Evertz *et al*, 1993)

- Generalization of cluster updates to quantum systems
- Loop-like clusters update world lines of spins



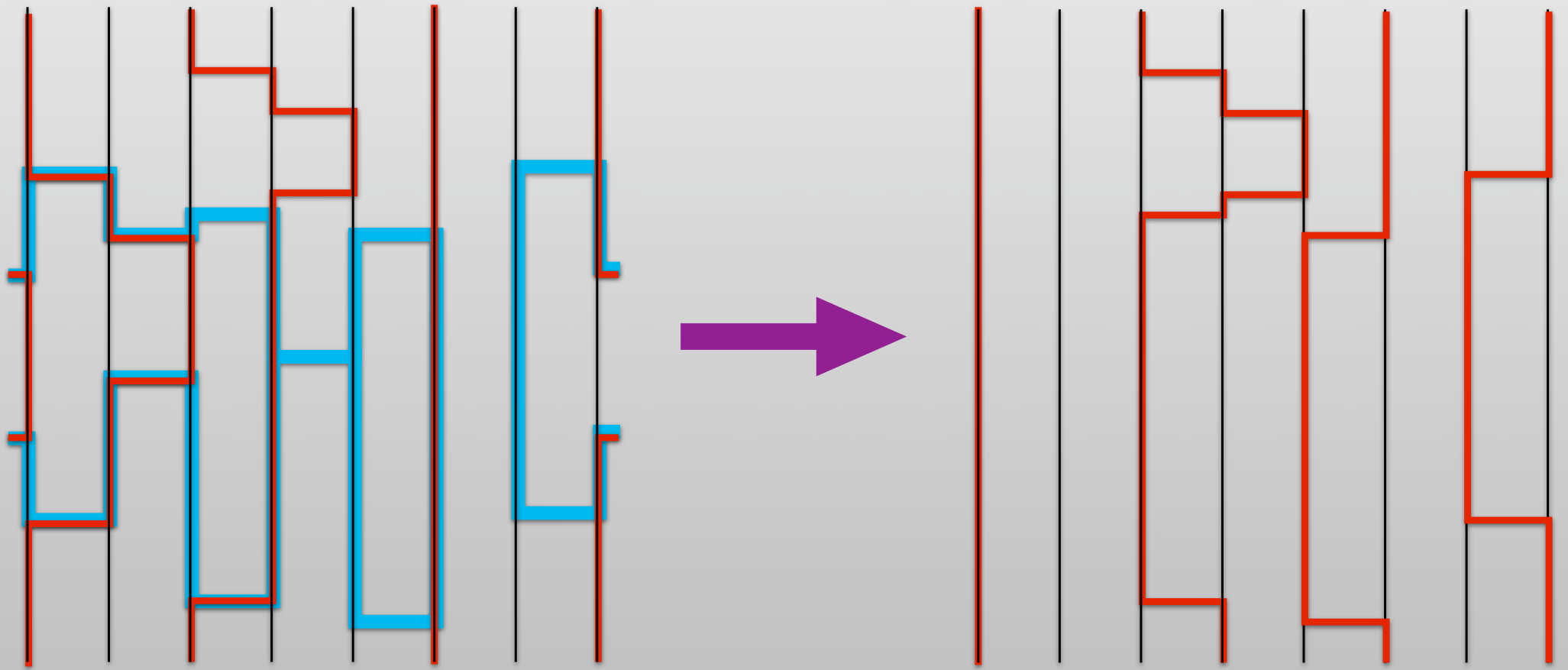
The loop algorithm (Evertz *et al*, 1993)

- Generalization of cluster updates to quantum systems
- Loop-like clusters update world lines of spins



The loop algorithm (Evertz *et al*, 1993)

- Generalization of cluster updates to quantum systems
- Loop-like clusters update world lines of spins

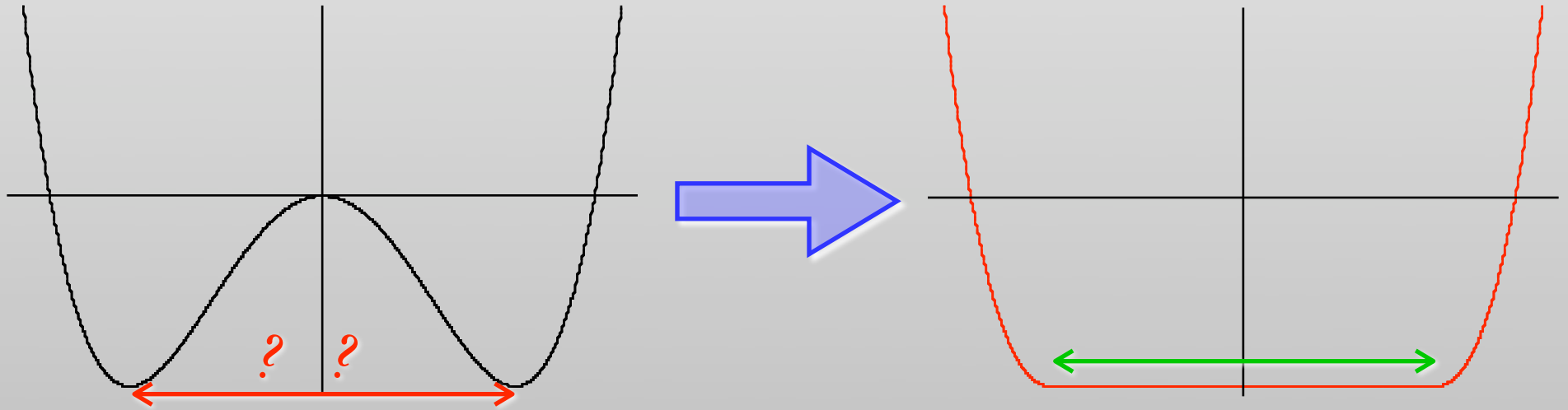


First order phase transitions

- Tunneling problem at a first order phase transition is solved by *changing the ensemble* to create a flat energy landscape
  - Multicanonical sampling (Berg and Neuhaus, Phys. Rev. Lett. 1992)
  - Wang-Landau sampling (Wang and Landau, Phys. Rev. Lett. 2001)
  - Quantum version (MT, Wessel and Alet, Phys. Rev. Lett. 2003)
  - Optimized ensembles (Trebst, Huse and MT, Phys. Rev. E 2004)

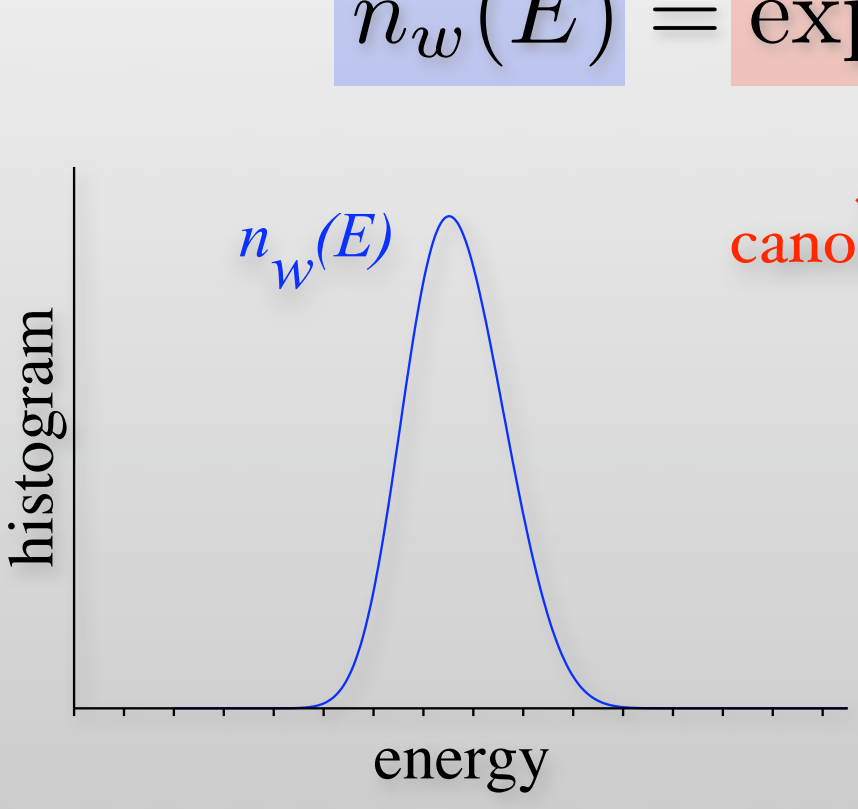
First order phase transitions

- Tunneling problem at a first order phase transition is solved by *changing the ensemble* to create a flat energy landscape
  - Multicanonical sampling (Berg and Neuhaus, Phys. Rev. Lett. 1992)
  - Wang-Landau sampling (Wang and Landau, Phys. Rev. Lett. 2001)
  - Quantum version (MT, Wessel and Alet, Phys. Rev. Lett. 2003)
  - Optimized ensembles (Trebst, Huse and MT, Phys. Rev. E 2004)

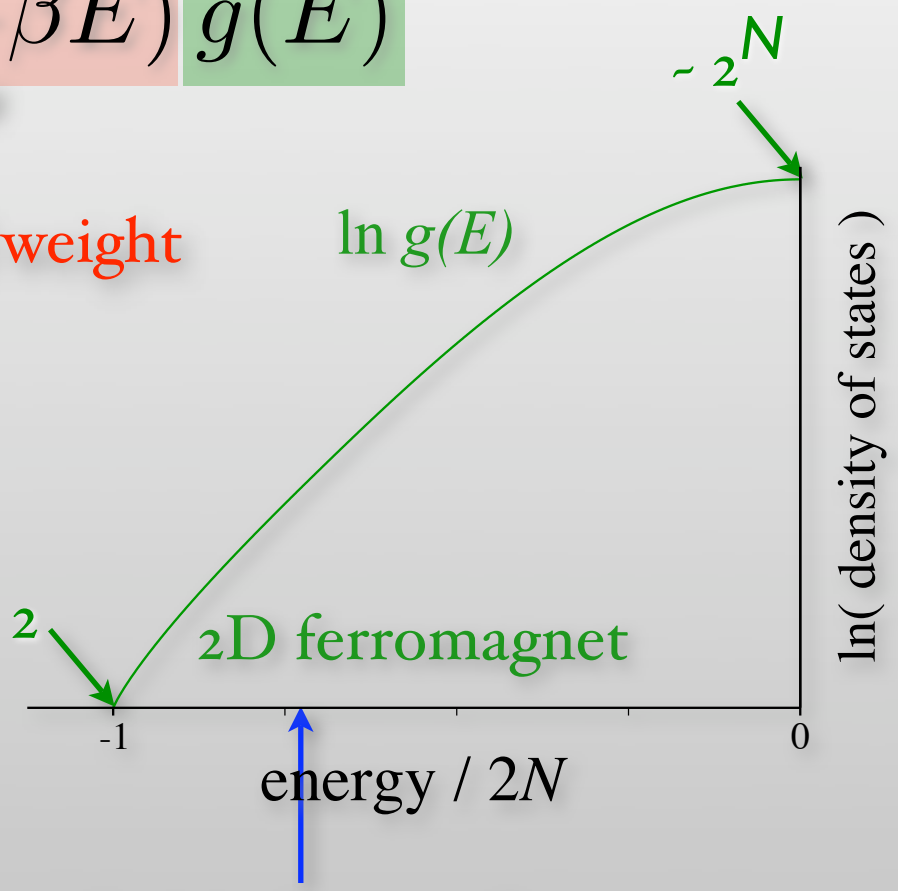


Canonical sampling

$$n_w(E) = \exp(-\beta E)\, g(E)$$



canonical weight

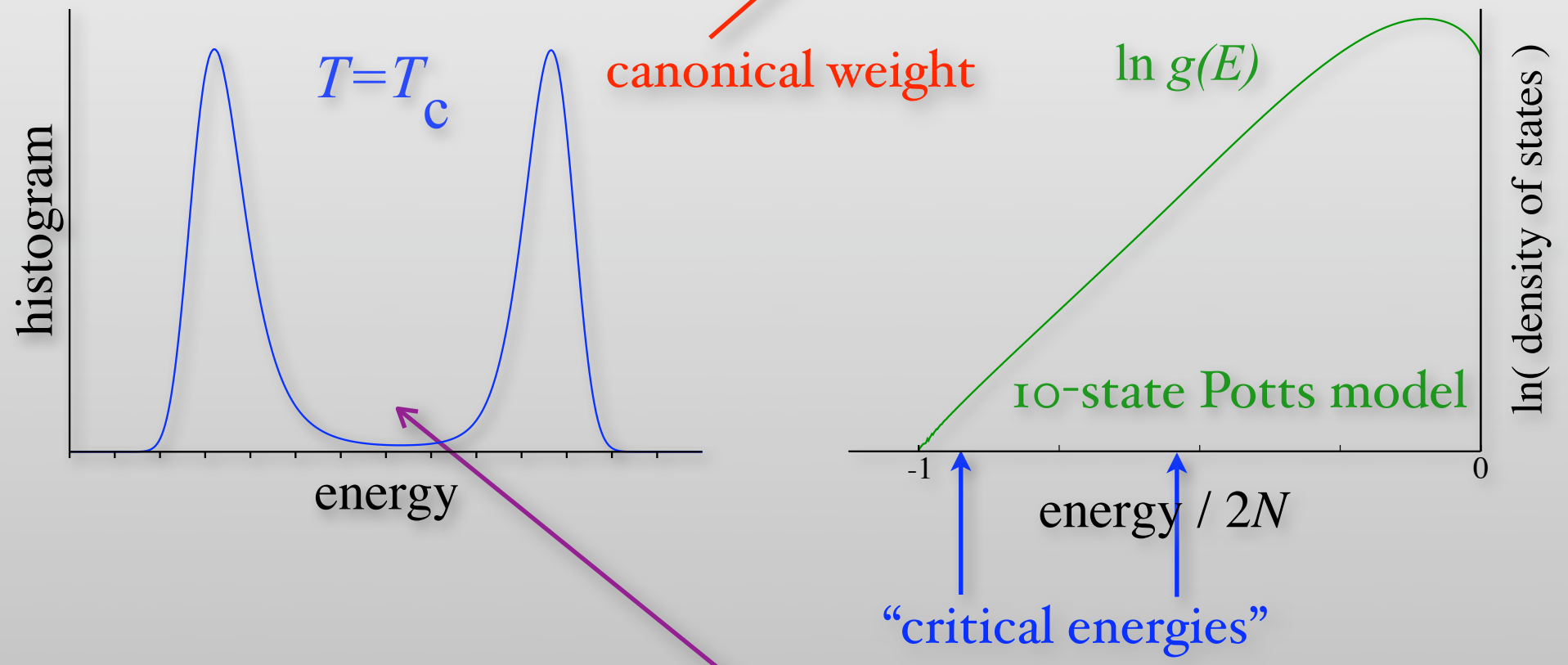


"critical energy"

$$E_c = E(T_c) \sim 0.74 E_0$$

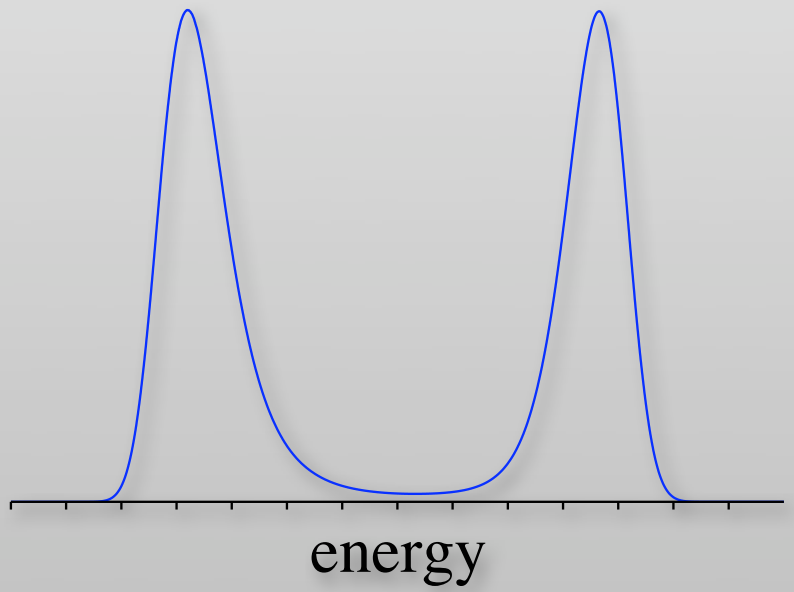
First-order phase transition

$$n_w(E) = \exp(-\beta E)\, g(E)$$

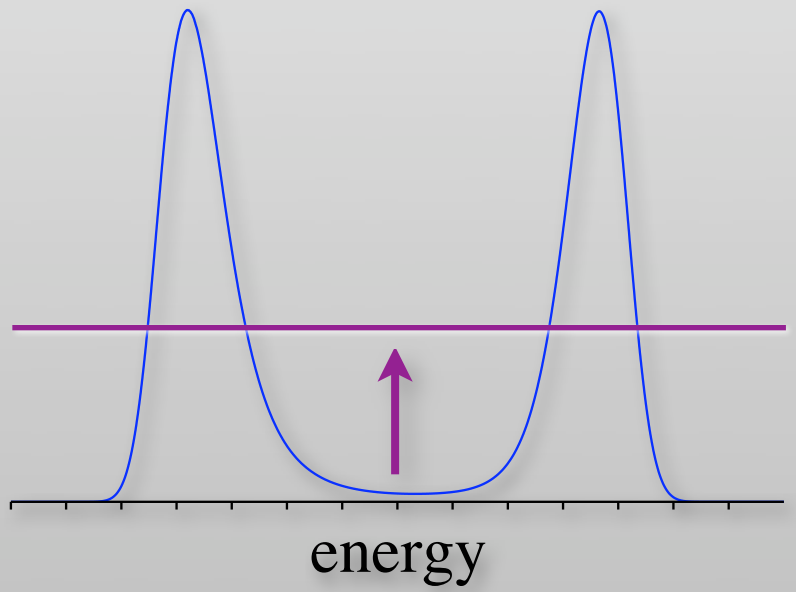


Exponentially suppressed tunneling out of metastable

Flat-histogram sampling



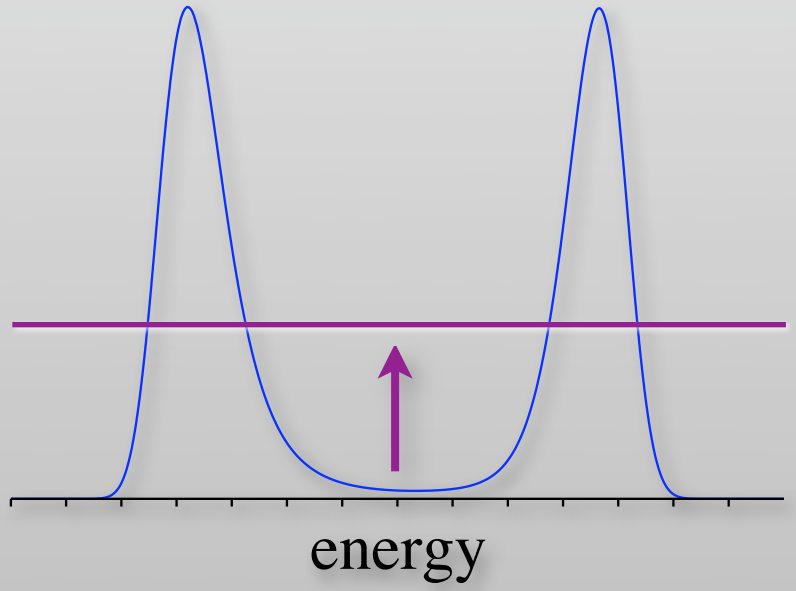
Flat-histogram sampling



Flat-histogram sampling

$$n_w(E) = 1/g(E) \cdot g(E)$$

"flat-histogram" weight

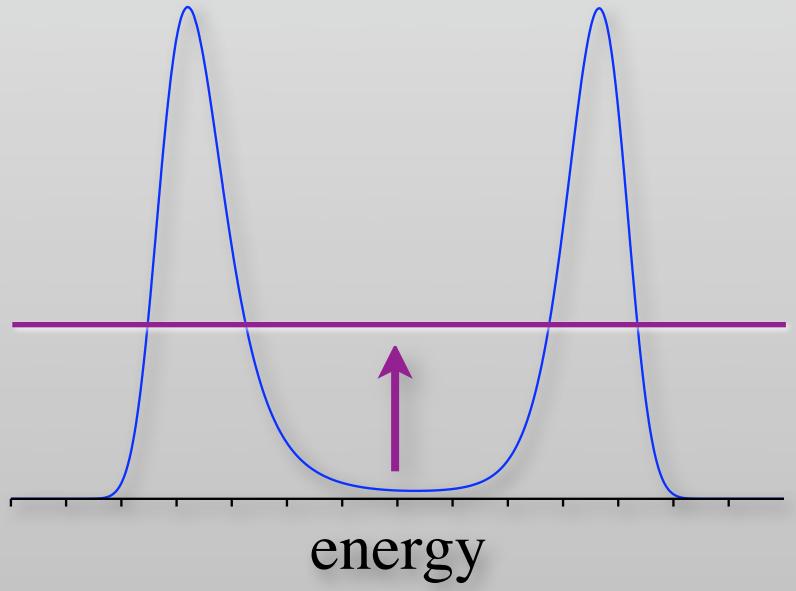


Flat-histogram sampling

$$n_w(E) = 1/g(E) \cdot g(E)$$

"flat-histogram" weight

How do we obtain the weights?

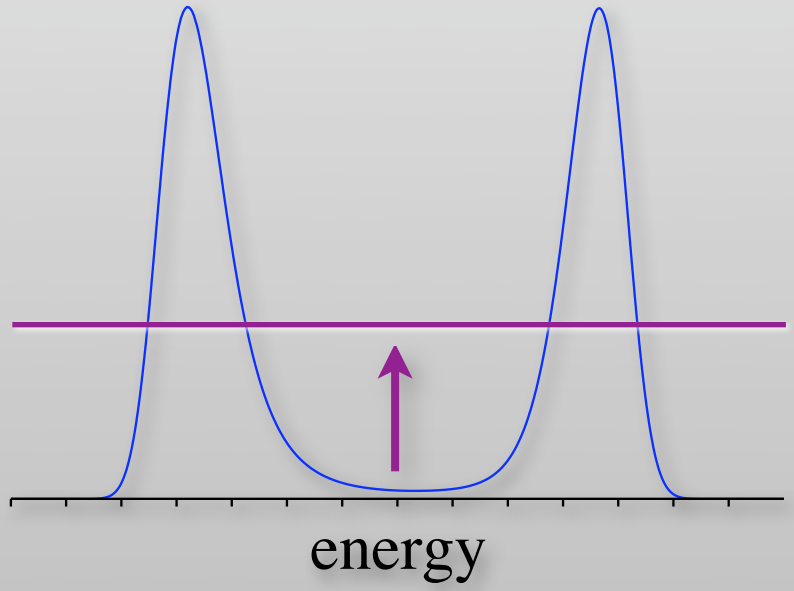


Flat-histogram sampling

$$n_w(E) = 1/g(E) \cdot g(E)$$

"flat-histogram" weight

How do we obtain the weights?



Flat-histogram MC algorithms

- Multicanonical recursions
  B. A. Berg and T. Neuhaus (1992)
- Wang-Landau algorithm
  F. Wang and D.P. Landau (2001)
- Quantum version
  M. Troyer, S. Wessel and F. Alet (2003)

Calculating the density of states

The Wang-Landau algorithm

Calculating the density of states

The Wang-Landau algorithm

- Start with "any" ensemble

$$w(E) = \frac{1}{\tilde{g}(E)} \qquad \tilde{g}(E) = 1$$

Calculating the density of states

The Wang-Landau algorithm

- Start with "any" ensemble

$$w(E) = \frac{1}{\tilde{g}(E)}$$

$\tilde{g}(E) = 1$

estimated density of states

Calculating the density of states

The Wang-Landau algorithm

- Start with "any" ensemble

$$w(E) = \frac{1}{\tilde{g}(E)} \qquad \tilde{g}(E) = 1$$

estimated density of states

- Simulate using Metropolis algorithm

$$p(E_1 \to E_2) = \min\left(1, \frac{w(E_2)}{w(E_1)}\right) = \min\left(1, \frac{\tilde{g}(E_1)}{\tilde{g}(E_2)}\right)$$

Calculating the density of states

The Wang-Landau algorithm

- Start with "any" ensemble

$$w(E) = \frac{1}{\tilde{g}(E)} \qquad \tilde{g}(E) = 1$$

estimated density of states

- Simulate using Metropolis algorithm

$$p(E_1 \to E_2) = \min\left(1, \frac{w(E_2)}{w(E_1)}\right) = \min\left(1, \frac{\tilde{g}(E_1)}{\tilde{g}(E_2)}\right)$$

- Iteratively improve ensemble during simulation

$$\tilde{g}(E) = \tilde{g}(E) \cdot f$$

Calculating the density of states

The Wang-Landau algorithm

- Start with "any" ensemble

$$w(E) = \frac{1}{\tilde{g}(E)} \qquad \tilde{g}(E) = 1$$

estimated density of states

- Simulate using Metropolis algorithm

$$p(E_1 \to E_2) = \min\left(1, \frac{w(E_2)}{w(E_1)}\right) = \min\left(1, \frac{\tilde{g}(E_1)}{\tilde{g}(E_2)}\right)$$

- Iteratively improve ensemble during simulation

$$\tilde{g}(E) = \tilde{g}(E) \cdot f$$

modification factor

Calculating the density of states

The Wang-Landau algorithm

- Start with "any" ensemble

$$w(E) = \frac{1}{\tilde{g}(E)} \qquad \tilde{g}(E) = 1$$

estimated density of states

- Simulate using Metropolis algorithm

$$p(E_1 \to E_2) = \min\left(1, \frac{w(E_2)}{w(E_1)}\right) = \min\left(1, \frac{\tilde{g}(E_1)}{\tilde{g}(E_2)}\right)$$

- Iteratively improve ensemble during simulation

$$\tilde{g}(E) = \tilde{g}(E) \cdot f$$

modification factor

- Reduce modification factor f when histogram is flat.

Wang-Landau in action

Movie by Emanuel Gull (2004)

Wang-Landau in action

Movie by Emanuel Gull (2004)

6. The negative sign problem in quantum Monte Carlo

Quantum Monte Carlo

- Not as easy as classical Monte Carlo

$$Z = \sum_c e^{-E_c / k_B T}$$

- Calculating the eigenvalues E_c is equivalent to solving the problem

- Need to find a mapping of the quantum partition function to a classical problem

$$Z = \mathrm{Tr}\, e^{-\beta H} \equiv \sum_c p_c$$

- "Negative sign" problem if some $p_c < 0$

Quantum Monte Carlo

- Feynman (1953) lays foundation for quantum Monte Carlo
- Map quantum system to classical world lines

THE

PHYSICAL REVIEW

A journal of experimental and theoretical physics established by E. L. Nichols in 1893

SECOND SERIES, VOL. 91, NO. 6 SEPTEMBER 15, 1953

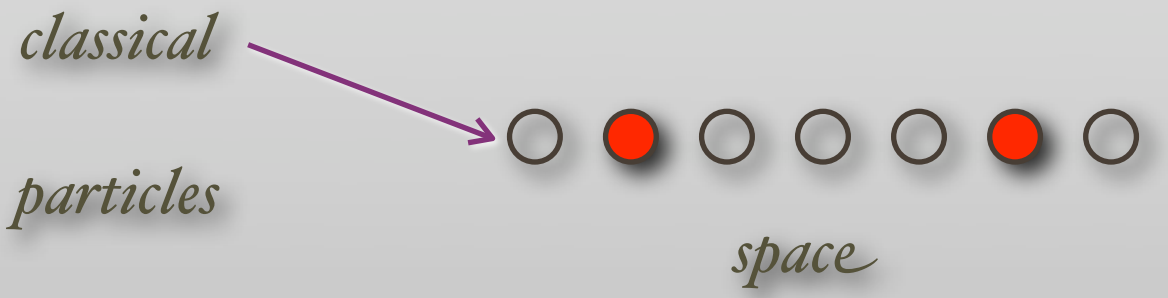
Atomic Theory of the λ Transition in Helium

R. P. FEYNMAN
California Institute of Technology, Pasadena, California

(Received May 15, 1953)

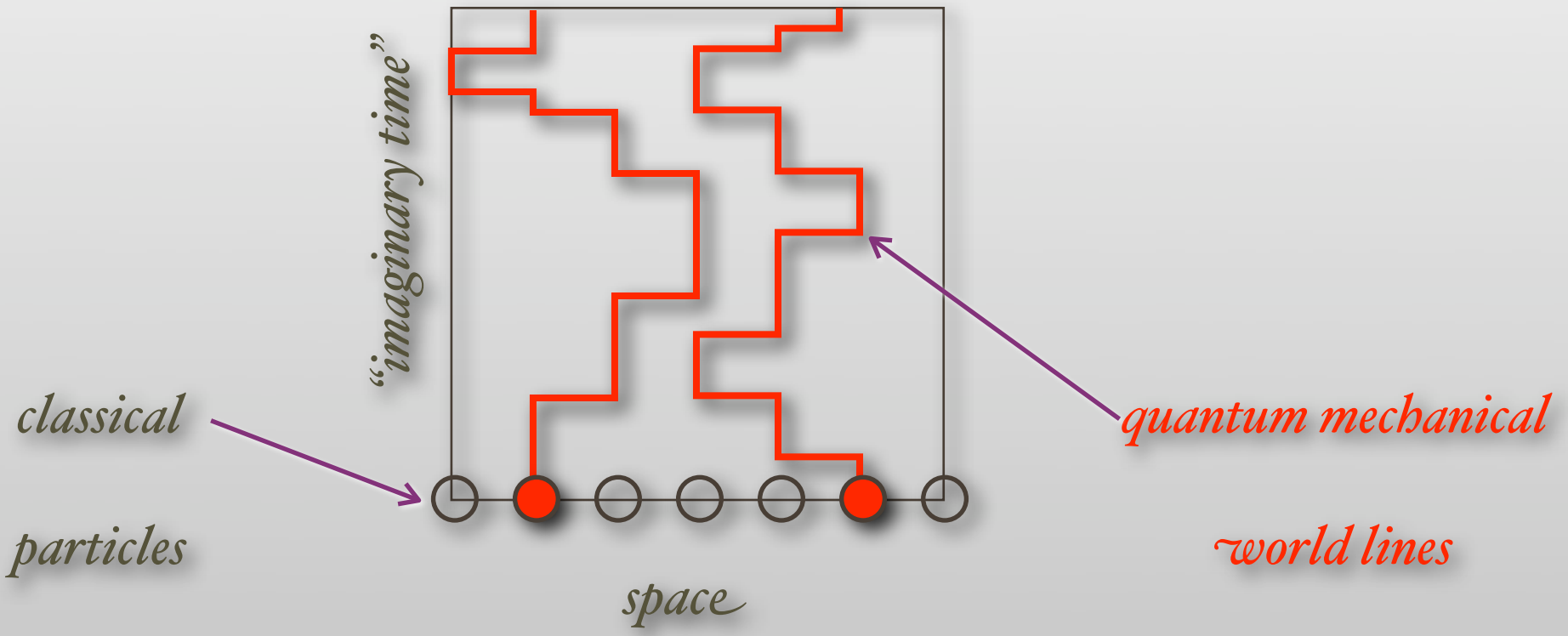
Quantum Monte Carlo

- Feynman (1953) lays foundation for quantum Monte Carlo
- Map quantum system to classical world lines



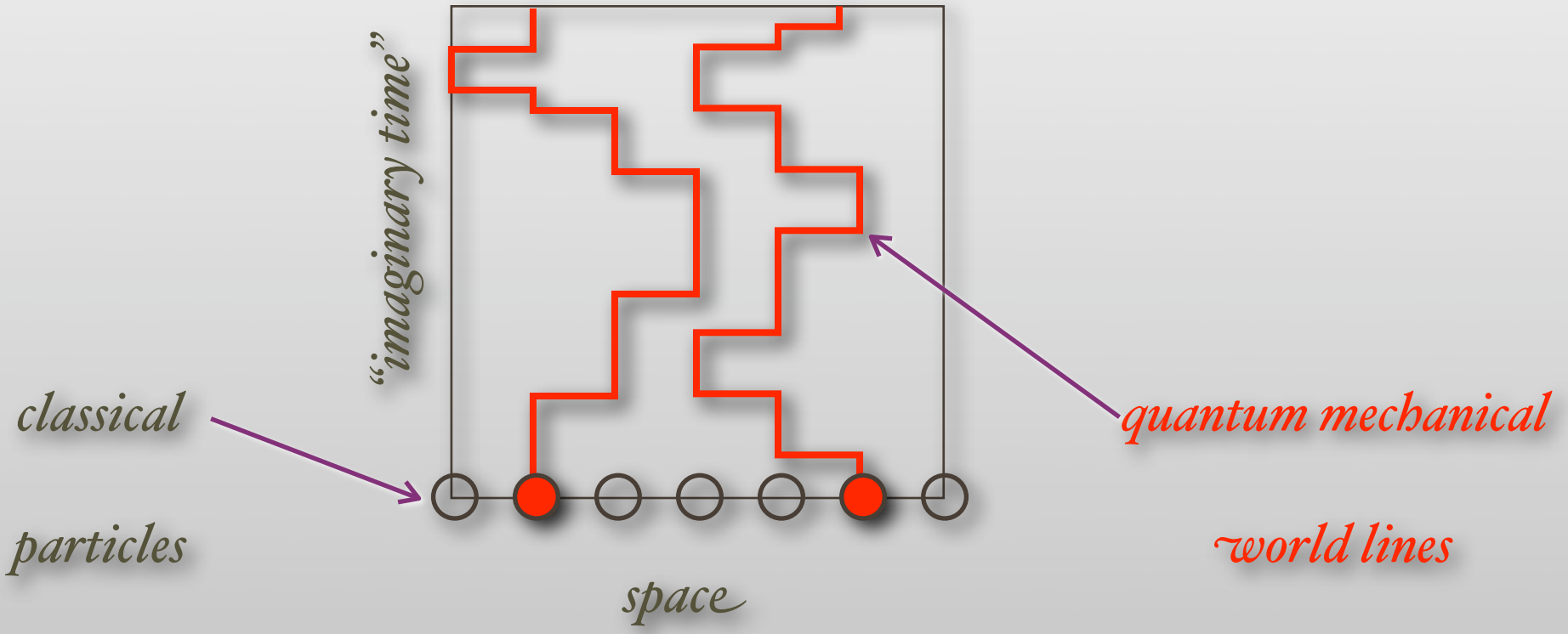
Quantum Monte Carlo

- Feynman (1953) lays foundation for quantum Monte Carlo
- Map quantum system to classical world lines



Quantum Monte Carlo

- Feynman (1953) lays foundation for quantum Monte Carlo
- Map quantum system to classical world lines



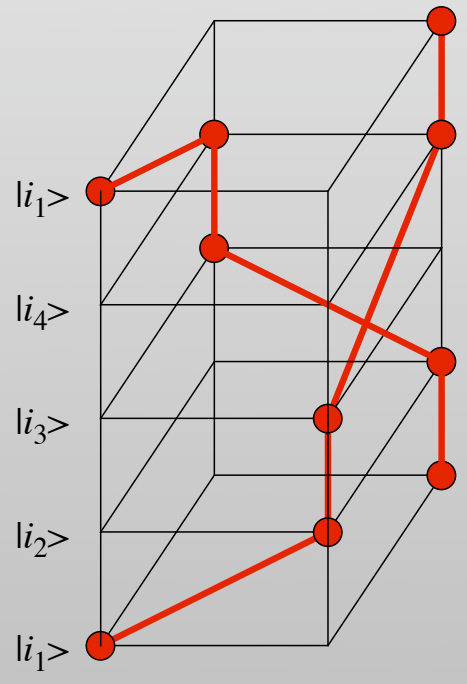
Use Metropolis algorithm to update world lines

The negative sign problem

- In mapping of quantum to classical system

$$Z = \mathrm{Tr} e^{-\beta H} = \sum_i p_i$$

- there is a "sign problem" if some of the $p_i < 0$
  - Appears e.g. in simulation of electrons when two electrons exchange places (Pauli principle)



The negative sign problem

- Sample with respect to absolute values of the weights

$$\langle A \rangle = \sum_i A_i p_i \Big/ \sum_i p_i = \frac{\sum_i A_i \operatorname{sgn} p_i |p_i| \Big/ \sum_i |p_i|}{\sum_i \operatorname{sgn} p_i |p_i| \Big/ \sum_i |p_i|} \equiv \frac{\langle A \cdot sign \rangle_{|p|}}{\langle sign \rangle_{|p|}}$$

- Exponentially growing cancellation in the sign

$$\langle sign \rangle = \frac{\sum_i p_i}{\sum_i |p_i|} = Z/Z_{|p|} = e^{-\beta V (f - f_{|p|})}$$

- Exponential growth of errors

$$\frac{\Delta sign}{\langle sign \rangle} = \frac{\sqrt{\langle sign^2 \rangle - \langle sign \rangle^2}}{\sqrt{M} \langle sign \rangle} \approx \frac{e^{\beta V (f - f_{|p|})}}{\sqrt{M}}$$

- NP-hard problem (no general solution) [Troyer and Wiese, PRL 2005]

The origin of the sign problem

The origin of the sign problem

- We sample with the wrong distribution by ignoring the sign!

The origin of the sign problem

- We sample with the wrong distribution by ignoring the sign!
- We simulate bosons and expect to learn about fermions?
  - will only work in insulators and superfluids

The origin of the sign problem

- We sample with the wrong distribution by ignoring the sign!
- We simulate bosons and expect to learn about fermions?
  - will only work in insulators and superfluids
- We simulate a ferromagnet and expect to learn something useful about a frustrated antiferromagnet?

The origin of the sign problem

- We sample with the wrong distribution by ignoring the sign!
- We simulate bosons and expect to learn about fermions?
  - will only work in insulators and superfluids
- We simulate a ferromagnet and expect to learn something useful about a frustrated antiferromagnet?
- We simulate a ferromagnet and expect to learn something about a spin glass?
  - This is the idea behind the proof of NP-hardness

Working around the sign problem

Working around the sign problem

1. Simulate “bosonic” systems
    - Bosonic atoms in optical lattices
    - Helium-4 supersolids
    - Nonfrustrated magnets

Working around the sign problem

1. Simulate "bosonic" systems
   - Bosonic atoms in optical lattices
   - Helium-4 supersolids
   - Nonfrustrated magnets

2. Simulate sign-problem free fermionic systems
   - Attractive on-site interactions
   - Half-filled Mott insulators

Working around the sign problem

1. Simulate "bosonic" systems
   - Bosonic atoms in optical lattices
   - Helium-4 supersolids
   - Nonfrustrated magnets

2. Simulate sign-problem free fermionic systems
   - Attractive on-site interactions
   - Half-filled Mott insulators

3. Restriction to quasi-1D systems
   - Use the density matrix renormalization group method (DMRG)

Working around the sign problem

1. Simulate "bosonic" systems
   - Bosonic atoms in optical lattices
   - Helium-4 supersolids
   - Nonfrustrated magnets

2. Simulate sign-problem free fermionic systems
   - Attractive on-site interactions
   - Half-filled Mott insulators

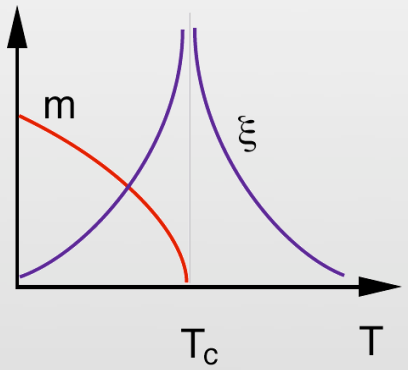
3. Restriction to quasi-1D systems
   - Use the density matrix renormalization group method (DMRG)

4. Use approximate methods
   - Dynamical mean field theory (DMFT)

7. Diverging Length Scales and Finite Size Scaling

Divergence of the correlation length ξ

- Typical length scale ξ divegres at phase transition at Tc



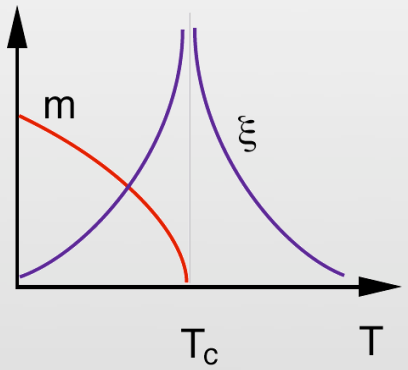
$$m \propto (T_c - T)^{\beta}$$

$$\xi \propto | T - T_c |^{-\nu}$$

- To avoid system size effects we need to have $L >> \xi \to \infty$

Divergence of the correlation length ξ

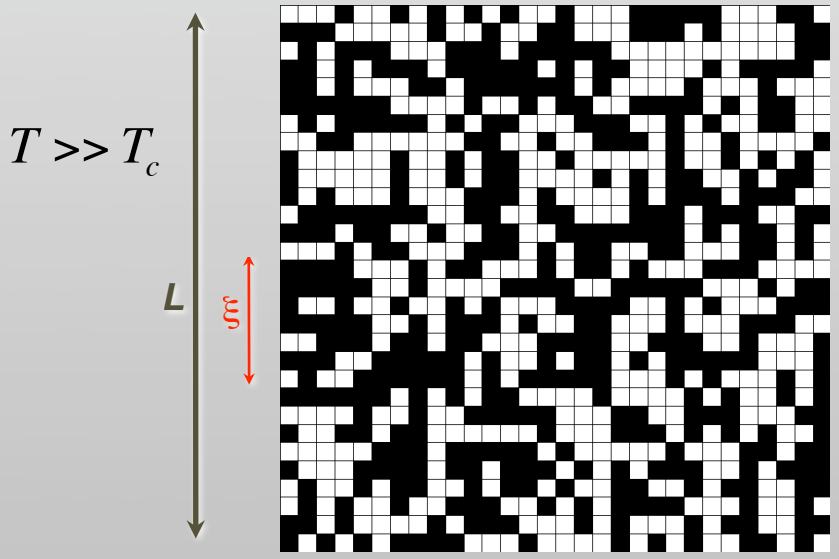
- Typical length scale ξ divegres at phase transition at Tc



$$m \propto (T_c - T)^{\beta}$$

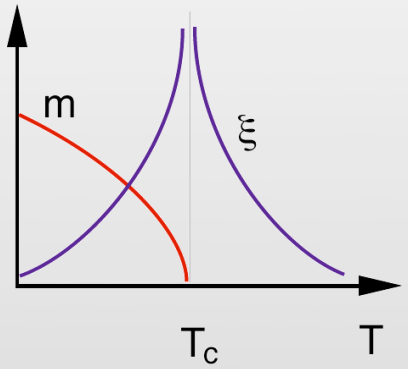
$$\xi \propto | T - T_c |^{-\nu}$$

- To avoid system size effects we need to have $L >> \xi \to \infty$



Divergence of the correlation length ξ

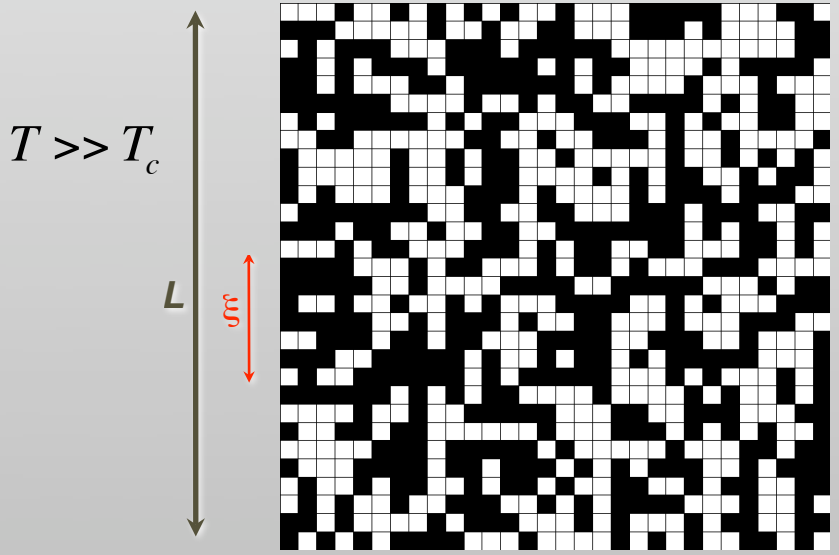
- Typical length scale ξ divegres at phase transition at Tc

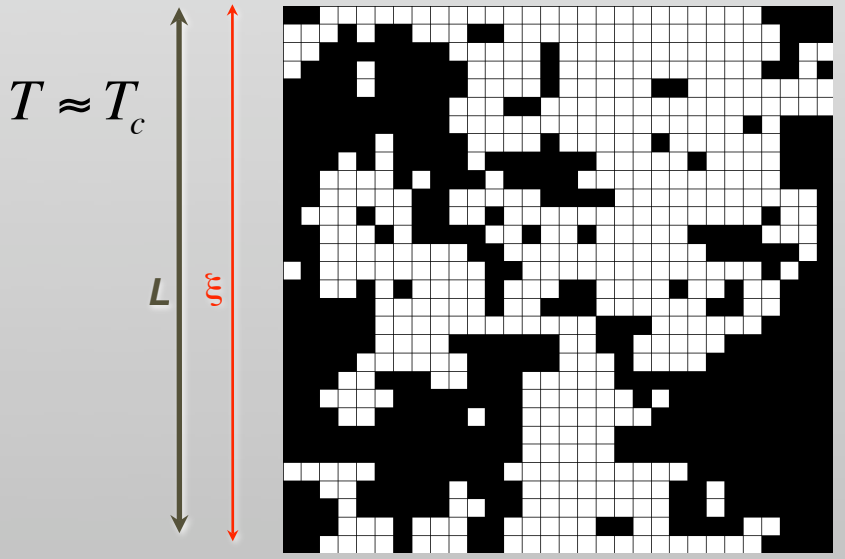


$$m \propto (T_c - T)^{\beta}$$

$$\xi \propto | T - T_c |^{-\nu}$$

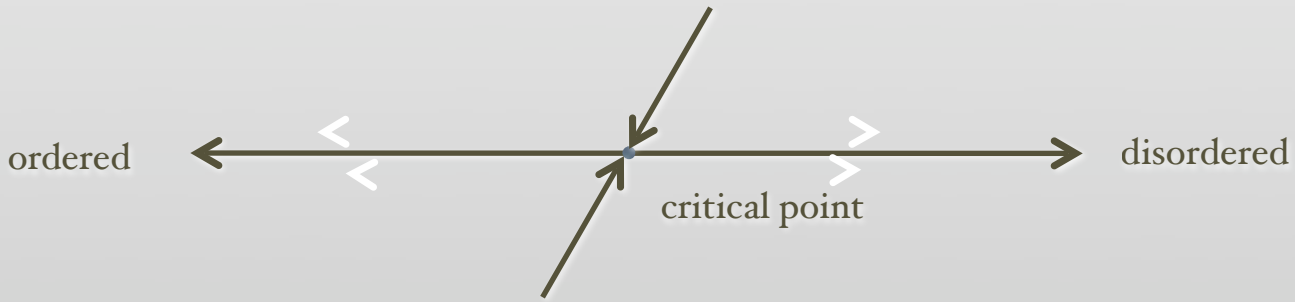
- To avoid system size effects we need to have $L >> \xi \to \infty$





Renormalization group and scaling

- As the length scale ξ diverges, "microscopic details" can be ignored
  - Physics happens at "large" length scale ξ
  - Microscopic length scale a of lattice can be ignored
  - All models with same symmetry converge to the same fixed point



- Fixed point is scale free
  - The only length scale ξ diverges
  - Self-similarity and fractal behavior
  - Power laws are scale free functions

$$m \propto (T_c - T)^{\beta}$$

$$\xi \propto | T - T_c |^{-\nu}$$

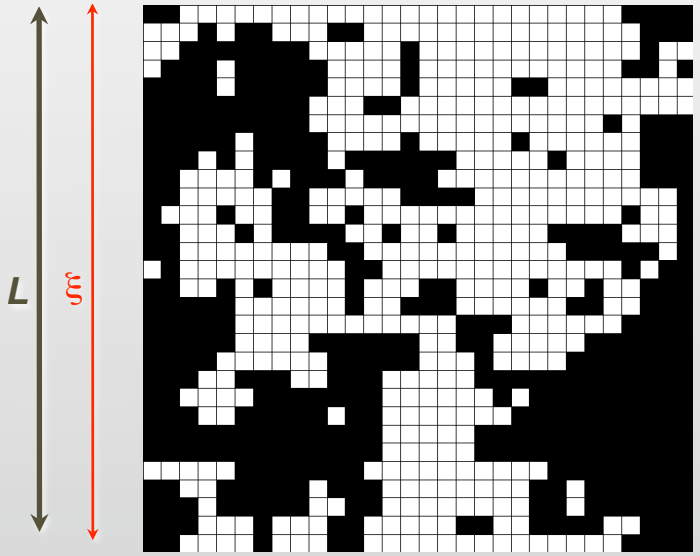
"Finite-size scaling"

- Infinite system

$$M \propto (T_c - T)^{\beta} \qquad \xi \propto (T_c - T)^{-\nu}$$

write M in terms of length scale ξ

$$\Rightarrow M(T) = M(\xi) \propto \xi^{-\beta/\nu}$$



- finite systems: L acts as cutoff to ξ

$$M(T,L) = M(\xi,L) = M(\xi/L) \propto \begin{cases} \xi^{-\beta/\nu} & L >> \xi \\ L^{-\beta/\nu} & L << \xi \end{cases}$$

- We can obtain critical exponents β, ν from finite size effects

A quantum antiferromagnet

- Quantum phase transition in a 2D Heisenberg antiferromagnet
- Susceptibility

$$\chi_s \propto L^{2-\eta}$$

- Structure factor of magnetization

$$S(Q) = L^2 m \propto L^{2-z-\eta}$$

- Scaling fits give z and η
- Additional dynamical critical exponent z is only difference from classical FSS

$$\xi_\tau \propto \xi^z$$