

相变与临界现象研究中蒙特卡洛方法介绍

Wenan Guo

Beijing Normal University

2019 May

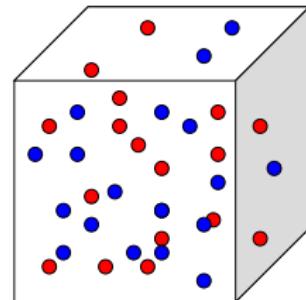


References:

- H.W.J. Blöte, Lecture notes: Nonlocal Monte carlo Methods
- A. W. Sandvik, <http://physics.bu.edu/~sandvik>

Manybody system, statistical physics

- Classical systems, any observable
 $[A, H] = 0$, each microstate Γ has certain energy $E(\Gamma) = H(\Gamma)$



- 平衡态配分函数(**partition function**)

$$Z = \sum_{\Gamma} e^{-E(\Gamma)/k_B T} = \sum_{\Gamma} W(\Gamma)$$

正则分布(**Canonical distribution**)

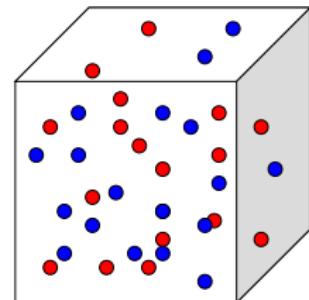


$$p_{\text{eq}}(\Gamma) = W(\Gamma)/Z$$

expection value: $\langle A \rangle = \sum_{\Gamma} A(\Gamma) \frac{W(\Gamma)}{Z}$

Manybody system, statistical physics

- Quantum systems, eigenenergies and eigenstates are not easy to obtain



- 平衡态配分函数(partition function)

$$Z = \text{Tr } e^{-H/k_B T} = \sum_{\alpha} \langle \alpha | e^{-\beta H} | \alpha \rangle$$

$|\alpha\rangle$ complete basis

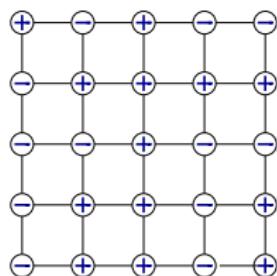
Thermal expectation value of a quantum observable



$$\langle A \rangle = \frac{1}{Z} \text{Tr}\{ A e^{-\beta H} \}$$

Classical Ising model

描写单轴铁磁体（可以在任意晶格上，以二维正方晶格为例）



$$H = -J \sum_{\langle i,j \rangle} s_i s_j - B \sum_k s_k; \quad s_k = \pm 1$$

微观状态 $\Gamma = (s_1, s_2, \dots, s_N)$.

总磁矩 $M(\Gamma) = \sum_k s_k$

- $J > 0$, 铁磁(ferromagnetic)
- $J < 0$, 反铁磁(antiferromagnetic)

当系统处于热平衡

$$\langle M \rangle = \sum_{\Gamma} M(\Gamma) p_{eq}(\Gamma)$$

- ▶ 在磁性和相变理论中非常重要
- ▶ 也是其它统计物理问题的有效模型：格气(lattice gas), 合金, 原子在表面的吸附问题等

Why Monte Carlo?

Monte Carlo simulation: 直接求和或积分不可能或不容易时的办法

以Ising model 为例，考虑任意物理量 A , 计算它的统计平均

$$\langle A \rangle = \sum_{\Gamma} A(\Gamma) p(\Gamma)$$

直接求和不现实: 2^N 个微观状态 (位形)

重要性抽样(Importance sampling)

[Metropolis, Rusenbluth, Rosenbluth, Teller, and Teller, Phys.Rev.1953]

- 构造一个随机过程，得到一系列微观状态 $\Gamma_1, \Gamma_2, \dots, \Gamma_M$.
 Γ_{t+1} 由 Γ_t 按照一定的跃迁几率(transition prob.) $T(\Gamma_{t+1}, \Gamma_t)$ 得到
当 $M \rightarrow \infty$, 任一给定位形 Γ 出现的频率 $\frac{N(\Gamma)}{M} = \frac{e^{-E(\Gamma)/k_B T}}{Z}$.
- 实现按位形的正则分布概率来抽取位形，而不是在相空间等概率地抽取位形

$$\boxed{\langle A \rangle \approx A_M = \sum_{\Gamma} \frac{N(\Gamma)}{M} A(\Gamma) = \frac{1}{M} \sum_l^M A(\Gamma_l)}$$

怎样实现这样的序列?

要实现这种序列, **关键是选取** $T(\Gamma', \Gamma)$: 更新 Γ 得到 Γ' 的几率

- 满足归一化条件 $\sum_{\Gamma'} T(\Gamma', \Gamma) = 1$

定义 $P_t(\Gamma)$ 为第 t 个微观状态为 Γ 的几率, 我们有

$$P_{t+1}(\Gamma') = \sum_{\Gamma} T(\Gamma', \Gamma) P_t(\Gamma)$$

此为主方程. 这样的过程称为**Markov**过程.

写成矢量形式

$$\vec{P}_t = \mathbf{T} \cdot \vec{P}_{t-1} = \mathbf{T}^t \cdot \vec{P}_0$$

可以预期

$$t \rightarrow \infty, \quad \vec{P}_t(\Gamma) \rightarrow \vec{P}_{eq}(\Gamma)$$

$$\vec{P}_{eq} = \mathbf{T} \cdot \vec{P}_{eq}$$

意味着 \vec{P}_{eq} 是 \mathbf{T} 的本征矢

我们的目标:对于任意初始几率分布 $P_0(\Gamma)$

$$P_{eq}(\Gamma) = \frac{e^{-E(\Gamma)/T}}{Z} = \frac{W(\Gamma)}{Z}$$

可以证明满足两个条件即可实现

- ▶ 各态历经(ergodic): 任意一个微观状态可以通过一系列的跃迁来达到(矩阵不可约)
- ▶ 细致平衡(detailed balancing)

$$T(\Gamma', \Gamma)P_{eq}(\Gamma) = T(\Gamma, \Gamma')P_{eq}(\Gamma')$$

即从 Γ 流到 Γ' 的几率等于反过来从 Γ' 流到 Γ 的几率
也就是

$$T(\Gamma', \Gamma)W(\Gamma) = T(\Gamma, \Gamma')W(\Gamma')$$

实现细致平衡：两种常见选择

跃迁几率可以分两步完成

$$T(\Gamma', \Gamma) = P_{\text{accept}}(\Gamma', \Gamma)P_{\text{select}}(\Gamma', \Gamma)$$

- $P_{\text{select}}(\Gamma', \Gamma)$: 从各种可能的新位形里选择 Γ' 的几率
- $P_{\text{accept}}(\Gamma', \Gamma)$: 接受新位形 Γ' 的几率

保证细致平衡的两种常见方式

Metropolis:

$$P_{\text{accept}}(\Gamma', \Gamma) = \min\left[\frac{P_{\text{select}}(\Gamma, \Gamma')W(\Gamma')}{P_{\text{select}}(\Gamma', \Gamma)W(\Gamma)}, 1\right]$$

Heat bath:

$$P_{\text{accept}}(\Gamma', \Gamma) = \frac{P_{\text{select}}(\Gamma, \Gamma')W(\Gamma')}{P_{\text{select}}(\Gamma, \Gamma')W(\Gamma') + P_{\text{select}}(\Gamma', \Gamma)W(\Gamma)}$$

实现细致平衡：两种常见选择

跃迁几率可以分两步完成

$$T(\Gamma', \Gamma) = P_{\text{accept}}(\Gamma', \Gamma)P_{\text{select}}(\Gamma', \Gamma)$$

- $P_{\text{select}}(\Gamma', \Gamma)$: 从各种可能的新位形里选择 Γ' 的几率
- $P_{\text{accept}}(\Gamma', \Gamma)$: 接受新位形 Γ' 的几率
- local update Algorithm (局域更新算法):
从 N 个自旋中挑选一个自旋尝试翻转:

$$P_{\text{select}}(\Gamma', \Gamma) = P_{\text{select}}(\Gamma, \Gamma') = 1/N, \text{ 此时上式化简为}$$

- Metropolis:

$$P_{\text{accept}}(\Gamma', \Gamma) = \min\left[\frac{W(\Gamma')}{W(\Gamma)}, 1\right]$$

- Heat bath:

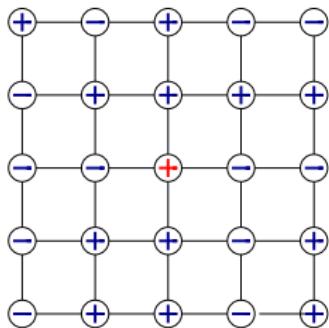
$$P_{\text{accept}}(\Gamma', \Gamma) = \frac{W(\Gamma')}{W(\Gamma') + W(\Gamma)}$$

这就是通常所谓单自旋蒙特卡罗模拟算法

Metropolis algorithm: 以Ising model为例

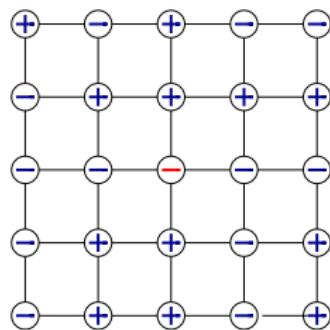
算法:

- 1 在位形 $\Gamma = (s_1, s_2, \dots, s_i, \dots, s_N)$ 中任选一个自旋, 比如 s_i .



Metropolis algorithm: 以Ising model为例

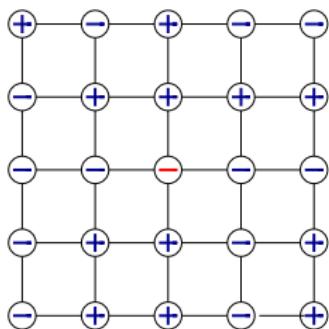
算法:



- 1 在位形 $\Gamma = (s_1, s_2, \dots, s_i, \dots, s_N)$ 中任选一个自旋, 比如 s_i .
- 2 翻转它, $\Gamma' = (s_1, s_2, \dots, -s_i, \dots, s_N)$. 这就是按 $p_{\text{select}} = \frac{1}{N}$ 的几率选择一个新位形

Metropolis algorithm: 以Ising model为例

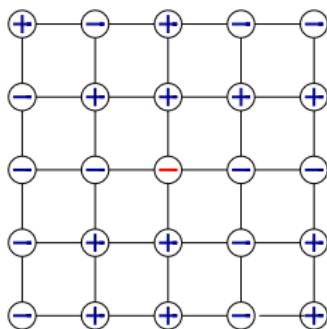
算法:



- 1 在位形 $\Gamma = (s_1, s_2, \dots, s_i, \dots, s_N)$ 中任选一个自旋, 比如 s_i .
- 2 翻转它, $\Gamma' = (s_1, s_2, \dots, -s_i, \dots, s_N)$. 这就是按 $p_{\text{select}} = \frac{1}{N}$ 的几率选择一个新位形
- 3 计算 $\Delta E = E(\Gamma') - E(\Gamma)$, 得到
$$\frac{W(\Gamma')}{W(\Gamma)} = e^{-\Delta E/k_B T}$$

Metropolis algorithm: 以Ising model为例

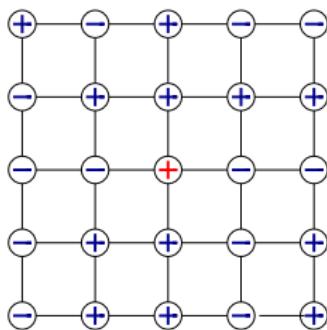
算法:



- 1 在位形 $\Gamma = (s_1, s_2, \dots, s_i, \dots, s_N)$ 中任选一个自旋, 比如 s_i .
- 2 翻转它, $\Gamma' = (s_1, s_2, \dots, -s_i, \dots, s_N)$. 这就是按 $p_{\text{select}} = \frac{1}{N}$ 的几率选择一个新位形
- 3 计算 $\Delta E = E(\Gamma') - E(\Gamma)$, 得到 $\frac{W(\Gamma')}{W(\Gamma)} = e^{-\Delta E/k_B T}$
- 4 设定 $P_{\text{accept}}(\Gamma', \Gamma) = \min[\frac{W(\Gamma')}{W(\Gamma)}, 1]$:
 - ▶ 如果 $\Delta E \leq 0$, 接受 Γ' ;
 - ▶ 如果 $\Delta E > 0$, 按几率 $e^{-\Delta E/k_B T}$ 接受 Γ' ; 否则拒绝 Γ' (或者说取 $\Gamma' = \Gamma$).

Metropolis algorithm: 以Ising model为例

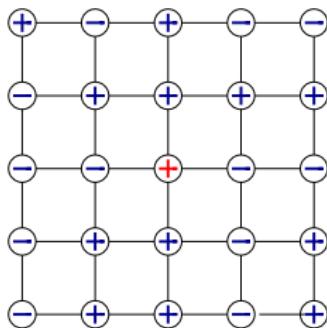
算法:



- 1 在位形 $\Gamma = (s_1, s_2, \dots, s_i, \dots, s_N)$ 中任选一个自旋, 比如 s_i .
- 2 翻转它, $\Gamma' = (s_1, s_2, \dots, -s_i, \dots, s_N)$. 这就是按 $p_{\text{select}} = \frac{1}{N}$ 的几率选择一个新位形
- 3 计算 $\Delta E = E(\Gamma') - E(\Gamma)$, 得到 $\frac{W(\Gamma')}{W(\Gamma)} = e^{-\Delta E/k_B T}$
- 4 设定 $P_{\text{accept}}(\Gamma', \Gamma) = \min[\frac{W(\Gamma')}{W(\Gamma)}, 1]$:
 - ▶ 如果 $\Delta E \leq 0$, 接受 Γ' ;
 - ▶ 如果 $\Delta E > 0$, 按几率 $e^{-\Delta E/k_B T}$ 接受 Γ' ; 否则拒绝 Γ' (或者说取 $\Gamma' = \Gamma$).
- 5 回到1, 循环到足够的位形产生.

Metropolis algorithm: 以Ising model为例

算法:



- 1 在位形 $\Gamma = (s_1, s_2, \dots, s_i, \dots, s_N)$ 中任选一个自旋, 比如 s_i .
- 2 翻转它, $\Gamma' = (s_1, s_2, \dots, -s_i, \dots, s_N)$. 这就是按 $p_{\text{select}} = \frac{1}{N}$ 的几率选择一个新位形
- 3 计算 $\Delta E = E(\Gamma') - E(\Gamma)$, 得到
$$\frac{W(\Gamma')}{W(\Gamma)} = e^{-\Delta E/k_B T}$$
- 4 设定 $P_{\text{accept}}(\Gamma', \Gamma) = \min[\frac{W(\Gamma')}{W(\Gamma)}, 1]$:
 - ▶ 如果 $\Delta E \leq 0$, 接受 Γ' ;
 - ▶ 如果 $\Delta E > 0$, 按几率 $e^{-\Delta E/k_B T}$ 接受 Γ' ; 否则拒绝 Γ' (或者说取 $\Gamma' = \Gamma$).
- 5 回到1, 循环到足够的位形产生.

► P_{accept} 也可以选为热浴(Heat Bath)法:

$$P_{\text{accept}}(\Gamma', \Gamma) = \frac{e^{-E(\Gamma')/k_B T}}{e^{-E(\Gamma')/k_B T} + e^{-E(\Gamma)/k_B T}} = \frac{1}{1 + e^{\Delta E/k_B T}}$$

- MC 时间: 一个Monte Carlo 步(1 Monte Carlo step)指进行完 N 次自旋翻转尝试(平均每个自旋尝试一次翻转)
 - 在实现平衡后开始测量观测量(确保 $t > \tau_2$ 平衡时间)
 - Bining: 把 M 步MC叫作一个bin, 最终计算bin平均和统计误差
-
- 程序流程
 - ▶ 生成任意的一个初始态
 - ▶ 执行一定的蒙特卡罗步(实现平衡)
 - ▶ 运行一定数目的bins
 - ▶ 每个bin包含 M 步
 - ▶ 在每一MC步(或者很少几步)后测量物理量
 - ▶ 完成每一个bin后保存bin平均
 - ▶ 计算最后的平均和统计误差

show

提纲

Manybody system, statistical physics

Monte Carlo simulation of the classical Ising model

Importance sampling

Local update algorithm: Metropolis, heat bath

模拟结果与分析, 有限尺寸标度(finite-size scaling)

3. 模拟结果与分析, 有限尺寸标度(finite-size scaling)

Cluster algorithm

提纲

Manybody system, statistical physics

Monte Carlo simulation of the classical Ising model

Importance sampling

Local update algorithm: Metropolis, heat bath

模拟结果与分析, 有限尺寸标度(finite-size scaling)

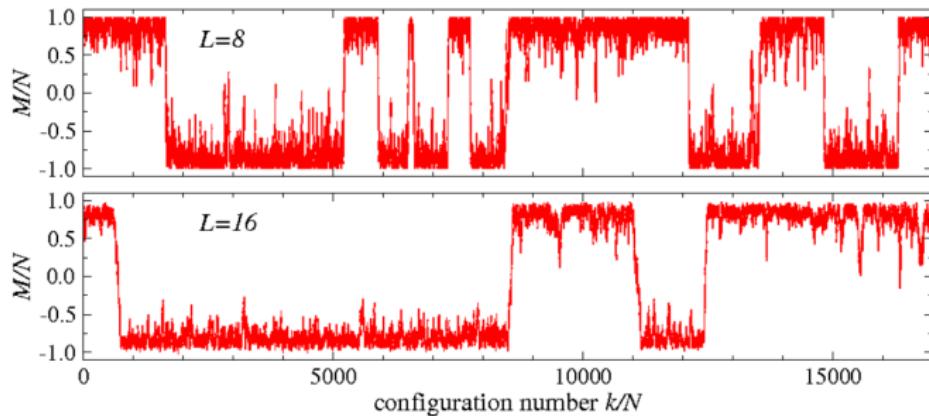
3. 模拟结果与分析, 有限尺寸标度(finite-size scaling)

Cluster algorithm

磁化强度的时间演化: 对称性的破缺

$$T/k_B T = 2.2 < T_c$$

- $\langle m \rangle = 0$, 但是 m 反转所需时间随尺寸增加
- 对于足够大的系统, 上下对称发生**破缺!**

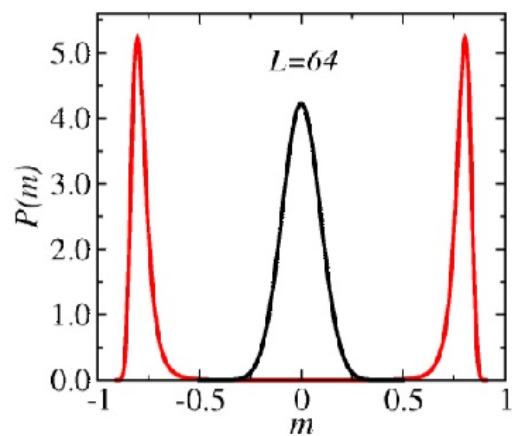
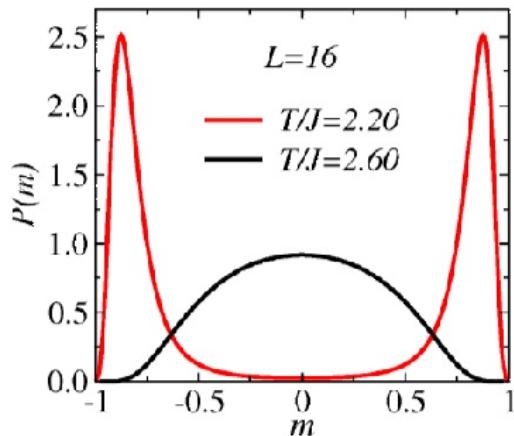


- 磁化强度及其分布依赖于温度和尺寸（有限尺寸效应）

▶ $T > T_c$ 在 $m = 0$ 附近形成单峰

▶ $T < T_c$ 在 $\pm \bar{m}$ 形成对称双峰

▶ m 分布



- 对称破缺：模拟中大尺寸系统只能测到一个峰，非常小的几率在两峰间跃迁

测量物理观测量

- 磁化强度为序参量(order parameter)

$$m = \frac{M}{N} = \frac{\sum_i^N s_i}{N}$$

但是它的统计平均（期望值）在有限系统里等于零，因为对称性只在热力学极限下真正破缺

如何在有限尺寸下区分对称破缺相与对称相？

模拟中我们计算 $\langle |m| \rangle$, 或者 $\langle m^2 \rangle$

测量物理观测量

- 磁化率

$$H = -J \sum_{\langle i,j \rangle} s_i s_j - B \sum_k s_k; \quad s_k = \pm 1$$

$$\chi = \frac{\partial m}{\partial B} = \frac{1}{N} \frac{\partial}{\partial B} \sum_{\Gamma} \frac{M e^{-\beta H}}{Z}$$

$$= \frac{\beta}{N} \left(\frac{\sum_{\Gamma} M^2 e^{-\beta H}}{Z} - \left(\frac{\sum_{\Gamma} M e^{-\beta H}}{Z} \right)^2 \right)$$

$$= \frac{1}{k_B T N} (\langle M^2 \rangle - \langle M \rangle^2)$$

即正比于磁矩的涨落

考慮到有限尺寸下 $\langle M \rangle = N \langle m \rangle = 0$, 模拟中我们计算

$$\chi = \frac{N}{k_B T} \langle m^2 \rangle$$

这就是MC中磁化率的estimator

测量物理观测量

- 比热

$$c = \frac{1}{N} \frac{dU}{dT} = \frac{1}{N} \frac{d}{dT} \frac{\sum_{\Gamma} E(\Gamma) e^{-E(\Gamma)/T}}{Z} = \frac{1}{NT^2} (\langle E^2 \rangle - \langle E \rangle^2)$$

表示成能量密度的函数

$$c = \frac{N}{T^2} (\langle e^2 \rangle - \langle e \rangle^2)$$

这就是MC中比热的estimator

计算统计平均和误差

具体计算：考虑 M 个“bins”，每个包含 n 个测量位形

由于测量位形出现几率（频率）正比于正则分布，有

$$A_i = \frac{1}{n} \sum_l A(\Gamma_l) \approx \langle A \rangle.$$

A_i 是第*i*个bin的测量平均值

计算统计平均和误差

我们有统计独立的平均 $A_i, i = 1, \dots, M$, 计算它们的平均值

$$\bar{A} = \frac{1}{M} \sum_i^M A_i$$

A_i 的标准偏差为

$$\sigma' = \sqrt{\frac{1}{M} \sum_i^M (A_i - \bar{A})^2} = \sqrt{(\bar{A}^2 - \bar{A}^2)}$$

我们关心的是 \bar{A} 的标准偏差: 误差; 基于中心极限定理

$$\sigma = \frac{\sigma'}{\sqrt{M - 1}}$$

成立条件: 要保证 A_i 统计独立, 即每个 bin 需要足够长

什么叫足够长? 我们要考虑 ‘自关联时间 (auto-correlation length)’

程序实现

- Main program

```
open(1,file='ising.in',status='old')
read(1,*)L,temp, initsteps, bins, binsteps
close(1)
call initialize(temp)
do i=1,initsteps
    call mcstep
enddo
do j=1,bins
    call cleandata
    do i=1,binsteps
        call mcstep
        call measure
    enddo
    call writebindata(n,binsteps)
enddo
```

系统变量可以用Module 描述

Module systemvariables

integer :: L ! 尺寸

integer :: N ! 自旋数 $N = L * L$

real(8) :: pflip(-4:4) ! 事先计算的接受几率

integer, allocatable :: spin(:) !自旋数组，可调

```
end module systemvariables
```

程序实现

- Initialization tasks

```
subroutine initialize(temp)
```

```
use systemvariables
```

```
N = L * L
```

```
do i = -4, 4, 2
```

```
    pflip(i) = exp(-i * 2./temp)
```

```
enddo
```

```
call init_random(seed)
```

```
allocate(spin(0:N-1))
```

```
do i=0,N-1
```

```
    call random_number(r)      ! spins are labeled  $s = 0, \dots, N - 1$ 
```

```
    spin(i) = 2 * int(2. * r) - 1 !spin(s) 是 s 的状态 = ±1
```

```
enddo
```

! 计算接受概率数组:

4个紧邻自旋的状态有:

4 正, 3正1负, 2正2负, 1正3负, 4负.

$s_1 + s_2 + s_3 + s_4 = 4, 2, 0, -2, -4$

如果被选中自旋为 s , 那么

令 $i = s(s_1 + s_2 + s_3 + s_4)$,

$\Delta E = 2 * i$

程序实现

- Performs one Monte Carlo step

```
subroutine mcstep()
do i=1,n
call random_number(r)
s=int(r*n) ! 随机挑选一个自旋s
x=mod(s,L); y=s/L      !计算横纵坐标
s1=spin(mod(x+1,L)+y*L) ! 计算4个最近邻, 可事先计算保存在数组中
s2=spin(mod(x-1+L,L)+y*L)
s3=spin(x+mod(y+1,L)*L)
s4=spin(x+mod(y-1+L,L)*L)
call random_number(r)
if (r<=pflip(spin(s)*(s1+s2+s3+s4))) spin(s)=-spin(s)
! 根据随机数r和跃迁几率pflip( $i = \Delta E / 2$ )决定是否接受  $s \rightarrow -s$ 
enddo
```

程序实现

- Measures observables (energy, magnetization, and their squares)

```
subroutine measure
```

```
real(8) :: enrg1,enrg2,magn1,magn2
```

```
common/measurments/enrg1,enrg2,magn1,magn2
```

```
e=0; m=0
```

```
do s=0,n-1
```

```
x=mod(s,L); y=s/L
```

```
e=e-spin(s)*(spin(mod(x+1,L)+y*L))
```

```
e=e-spin(s)*(spin(x+mod(y+1,L)*L))
```

```
enddo
```

```
m=sum(spin)
```

```
enrg1=enrg1+dble(e) !累计能量
```

```
enrg2=enrg2+dble(e)**2 ! 累计能量平方
```

```
magn1=magn1+abs(m) ! 累计磁化强度
```

```
magn2=magn2+dble(m)**2 ! 累计磁化强度平方
```

程序实现

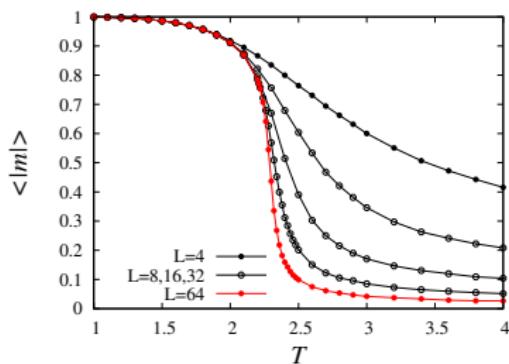
- Writes bin averages to a file

```
subroutine writebindata(n,steps)
real(8) :: enrg1,enrg2,magn1,magn2
common/measurments/enrg1,enrg2,magn1,magn2 !公用块，可用module代替
open(1,file='bindata.dat',position='append')
!以下计算统计平均
enrg1=enrg1/(dble(steps)*dble(n))
enrg2=enrg2/(dble(steps)*dble(n)**2)
magn1=magn1/(dble(steps)*dble(n))
magn2=magn2/(dble(steps)*dble(n)**2)
write(1,'(4f18.12)')enrg1,enrg2,magn1,magn2
close(1)
```

- These bin data should be processed (giving final averages and statistical errors) with a separate program.

自发磁化，有限尺寸行为

- Ising model的自发磁化

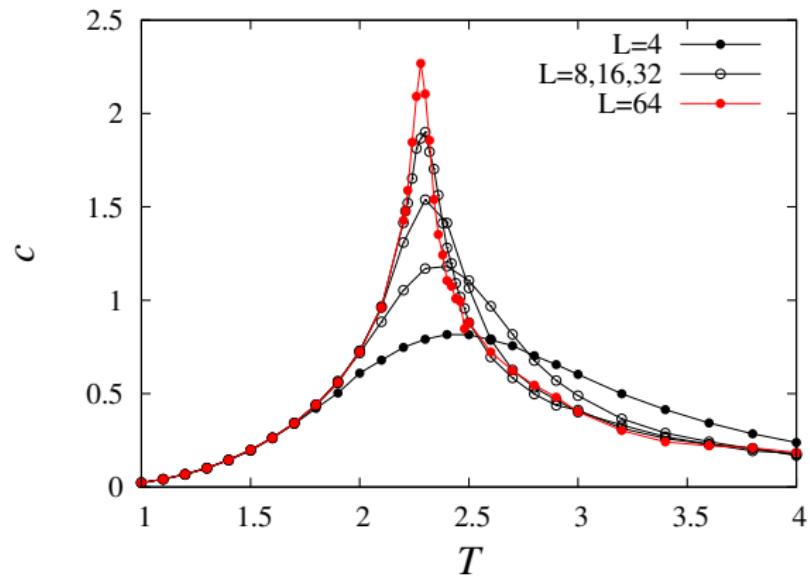


热力学极限下理论公式($T_c = 2.269..$)

$$\langle m \rangle = \left(1 - \frac{1}{\sinh^4(2/T)}\right)^{1/8}, \quad T \leq T_c$$

$$\langle m \rangle = 0, \quad T > T_c$$

比热



$$c \propto |T - T_c|^\alpha$$

临界指数的计算: 临界指数 β

热力学极限下

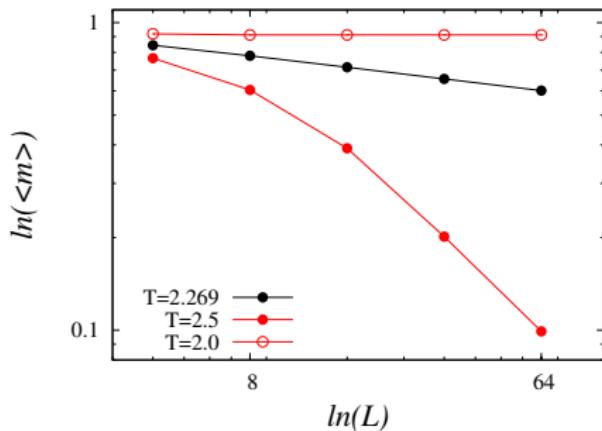
$$\langle m \rangle \propto (T_c - T)^\beta \propto \xi^{-\beta/\nu}, \quad \beta = \frac{1}{8} \quad T < T_c$$

有限系统, 在 $T = T_c$, 可以理解为用 L 替代 ξ

$$\langle |m| \rangle \propto L^{-\beta/\nu} + \dots$$

- 利用这一关系, 在 T_c 模拟一系列尺寸系统, 我们可以定出 β

(作对数图: $\ln |m|$ vs. $\ln L$)



临界点的确定： Binder ratio

Binder Ratio : $Q = \frac{\langle m^2 \rangle}{\langle |m| \rangle^2}$, 或者, $Q = \frac{\langle m^4 \rangle}{\langle m^2 \rangle^2}$

$$\chi = \frac{N}{T}(\langle m^2 \rangle - \langle m \rangle^2) \propto t^{-\gamma}$$

有限尺寸标度(finite-size scaling, 相变点以 L 代替 $\xi \propto t^{-\nu}$)

$$\chi(L, T_c) = \frac{N}{T}(\langle m^2 \rangle - \langle m \rangle^2) \propto L^{\gamma/\nu}$$

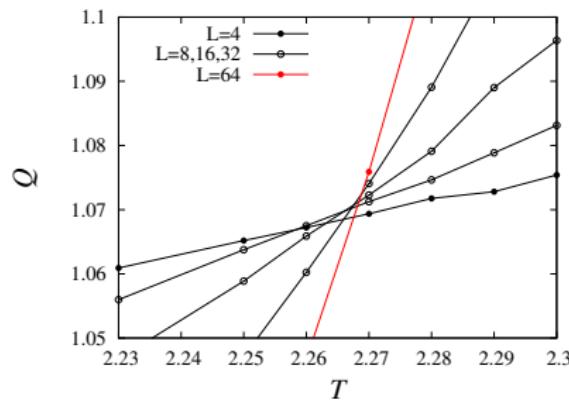
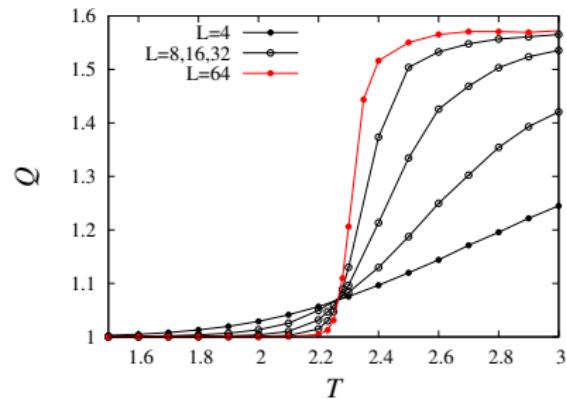
$$\langle m^2 \rangle \propto L^{\gamma/\nu-d} \quad \langle |m| \rangle \propto L^{\gamma/(2\nu)-d/2}$$

Q 在临界点与尺寸无关!

$Q \rightarrow 1$ 当 $T \rightarrow 0$; $Q \rightarrow constant$ 当 $T \rightarrow \infty$, 由于 m 的高斯分布

临界点的确定：Binder ratio

$$\text{Binder Ratio : } Q = \frac{\langle m^2 \rangle}{\langle |m| \rangle^2}$$



自关联时间

设某个物理观测量在第 t MC步的值为 $Q(t)$

自关联函数测量它需要花多长‘时间’与之前数值独立?

$$A_Q(\Delta t) = \frac{\langle Q(t + \Delta t)Q(t) \rangle - \langle Q(t) \rangle^2}{\langle Q(t)^2 \rangle - \langle Q(t) \rangle^2}$$

一般满足

$$A_Q(\Delta t) \sim e^{-\Delta t/\tau}, \quad \tau \text{ 自关联时间}$$

Integrated autocorrelation time (决定独立样本数 $\approx n/(2\tau_{int})$)

$$\tau_{int} = \frac{1}{2} + \sum_{\Delta t=1}^{\infty} A_Q(\Delta t) \approx \tau$$

- 临界慢化(critical slowing down)

$\tau_{int} \propto \xi^z \rightarrow \infty$, 热力学极限下, 当 $T \rightarrow T_c$

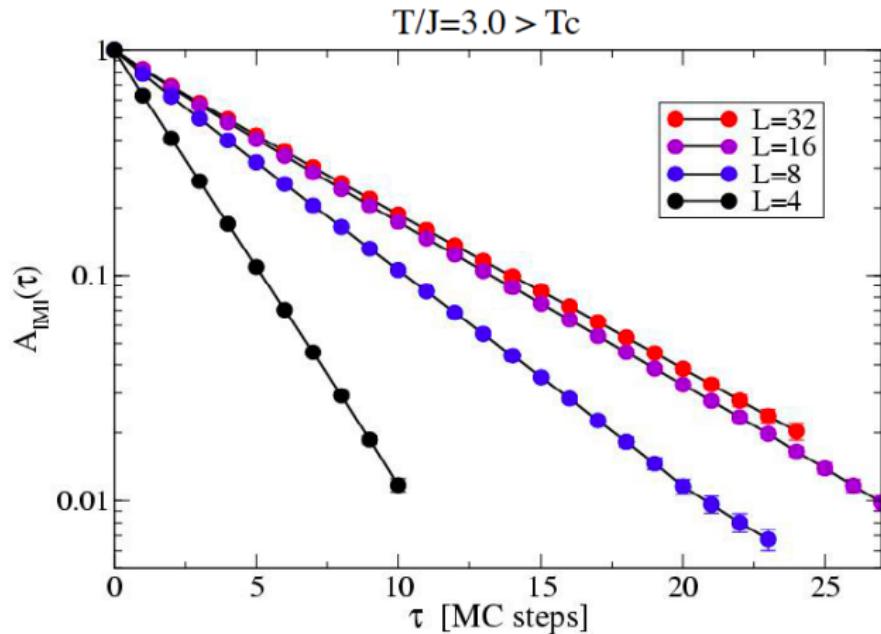
- 对于处于临界点 T_c 的有限尺寸系统, Q 为序参量

$$\tau_{int} \sim L^z$$

z 动力学临界指数, 由实现平衡的动力学, 亦即算法, 决定!

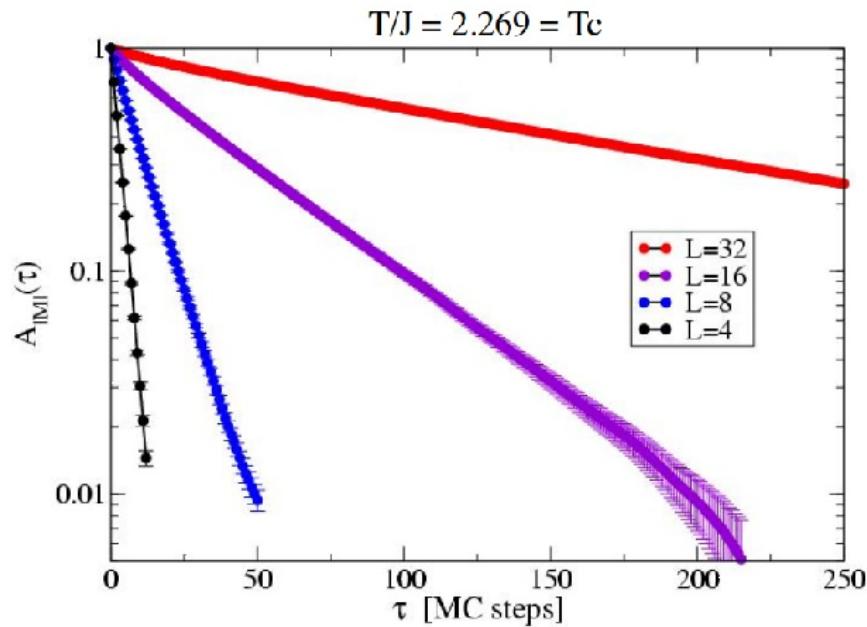
Metropolis算法的弱点, 考虑 $A_{|M|}(\Delta t)$

当温度高于 T_c



关联时间迅速收敛到与尺寸无关的值

Metropolis算法的弱点, 考虑 $A_{|M|}(\Delta t)$



$\tau \propto L^z, z \approx 2.2$: 对于局域算法(local algorithm), 由于每次尝试的 Γ' 与 Γ 相差很小(local 算法) 自关联时间很长

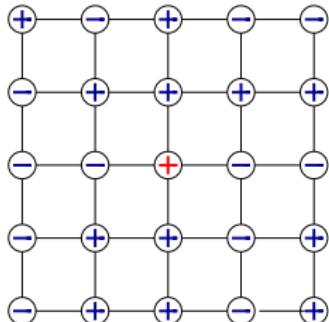
- 需要经过正比于 L^{d+z} 次操作后才可以得到一个独立的位形

Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤：

- 1 任意选取一个自旋*i*



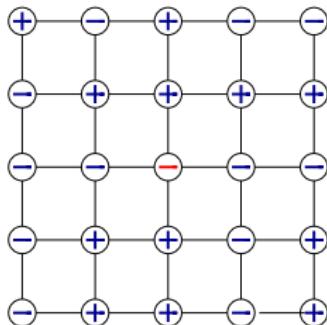
Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤：

1 任意选取一个自旋 i

2 翻转之: $s'_i = -s_i$



Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤:

1 任意选取一个自旋 i

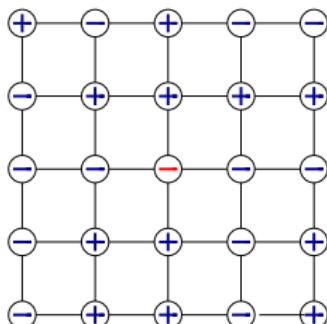
2 翻转之: $s'_i = -s_i$

3 对 i 的所有最近邻 k 作如下操作:

► 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成

i 翻转 $s_k \rightarrow s'_k = -s_k$.

ii 将 k 记录到地址列表('堆栈(stack)')



Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤:

1 任意选取一个自旋 i

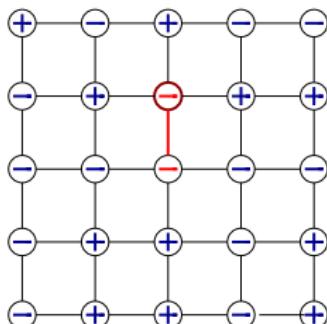
2 翻转之: $s'_i = -s_i$

3 对 i 的所有最近邻 k 作如下操作:

► 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成

i 翻转 $s_k \rightarrow s'_k = -s_k$.

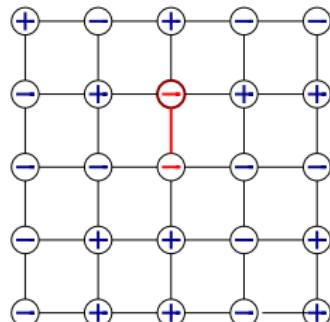
ii 将 k 记录到地址列表('堆栈(stack)')



Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤：

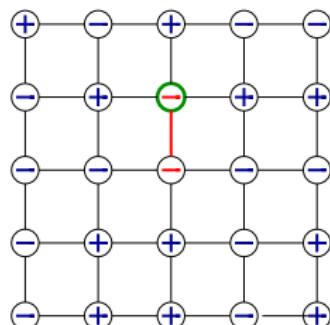


- 1 任意选取一个自旋 i
- 2 翻转之: $s'_i = -s_i$
- 3 对 i 的所有最近邻 k 作如下操作:
 - 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成
 - i 翻转 $s_k \rightarrow s'_k = -s_k$.
 - ii 将 k 记录到地址列表('堆栈(stack)')
 - 如果 $s_k = s'_i$ 什么也不作

Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤:

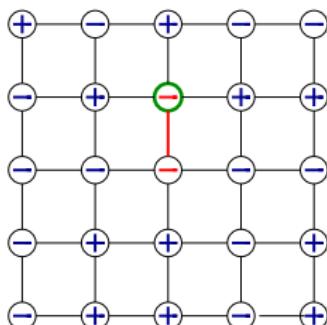


- 1 任意选取一个自旋 i
- 2 翻转之: $s'_i = -s_i$
- 3 对 i 的所有最近邻 k 作如下操作:
 - 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成
 - i 翻转 $s_k \rightarrow s'_k = -s_k$.
 - ii 将 k 记录到地址列表('堆栈(stack)')
 - 如果 $s_k = s'_i$ 什么也不作
- 4 从堆栈中读取一个地址 l

Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤:

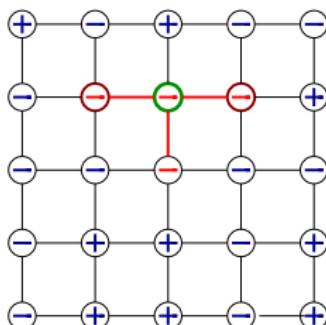


- 1 任意选取一个自旋*i*
- 2 翻转之: $s'_i = -s_i$
- 3 对*i*的所有最近邻*k*作如下操作:
 - 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成
 - i 翻转 $s_k \rightarrow s'_k = -s_k$.
 - ii 将*k*记录到地址列表('堆栈(stack)')
 - 如果 $s_k = s'_i$ 什么也不作
- 4 从堆栈中读取一个地址*l*
- 5 对*l*执行第3步

Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤:

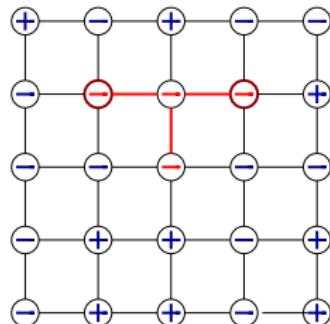


- 1 任意选取一个自旋 i
- 2 翻转之: $s'_i = -s_i$
- 3 对 i 的所有最近邻 k 作如下操作:
 - 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成
 - i 翻转 $s_k \rightarrow s'_k = -s_k$.
 - ii 将 k 记录到地址列表('堆栈(stack)')
 - 如果 $s_k = s'_i$ 什么也不作
- 4 从堆栈中读取一个地址 l
- 5 对 l 执行第3步

Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤：

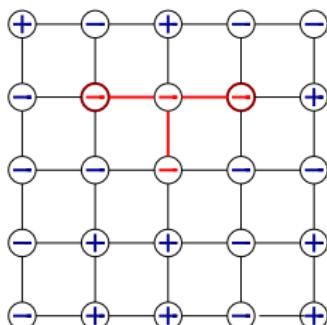


- 1 任意选取一个自旋 i
- 2 翻转之: $s'_i = -s_i$
- 3 对 i 的所有最近邻 k 作如下操作:
 - 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成
 - i 翻转 $s_k \rightarrow s'_k = -s_k$.
 - ii 将 k 记录到地址列表('堆栈(stack)')
 - 如果 $s_k = s'_i$ 什么也不作
- 4 从堆栈中读取一个地址 l
- 5 对 l 执行第3步
- 6 从堆栈中删除 l

Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤：

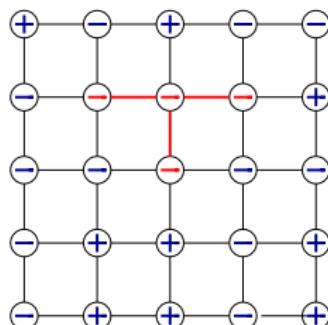


- 1 任意选取一个自旋 i
- 2 翻转之: $s'_i = -s_i$
- 3 对 i 的所有最近邻 k 作如下操作:
 - 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成
 - i 翻转 $s_k \rightarrow s'_k = -s_k$.
 - ii 将 k 记录到地址列表('堆栈(stack)')
 - 如果 $s_k = s'_i$ 什么也不作
- 4 从堆栈中读取一个地址 l
- 5 对 l 执行第3步
- 6 从堆栈中删除 l
- 7 重复4-6, 直到堆栈为空

Cluster algorithm: Wolff algorithm

在一步里翻转一个集团，而不是一个一个地翻 $N = L^d$ 次

步骤：



- 1 任意选取一个自旋 i
- 2 翻转之: $s'_i = -s_i$
- 3 对 i 的所有最近邻 k 作如下操作:
 - 如果 $s_k = s_i$ 则以几率 $1 - e^{-2K}$, $K = J/k_B T$ 完成
 - i 翻转 $s_k \rightarrow s'_k = -s_k$.
 - ii 将 k 记录到地址列表('堆栈(stack)')
 - 如果 $s_k = s'_i$ 什么也不作
- 4 从堆栈中读取一个地址 l
- 5 对 l 执行第3步
- 6 从堆栈中删除 l
- 7 重复4-6, 直到堆栈为空

程序实现

```
s=rn()*n+1 !! rn()是随机数，随机选择一个自旋s (位置)
icsp=spin(s) !! icsp 是自旋s原来的状态
spin(s)=-icsp !! 翻转s
nstack=0 !! 堆栈里的成员数和当前成员标号
js=s !! 当前考虑的自旋 (位置)
104 continue
do 107 inb=1,4
    ks=nbor(inb,js) !! js 的4个邻居位置，事先算好保存在数组nbor中
    if ((spin(ks).eq.icsp).and.(rn().lt.bp)) then !! bp 是几率 $1 - e^{-2K}$ 
        nstack=nstack+1 !! ks 放入堆栈，成员数加一
        istn(nstack)=ks !! ks 是第nstack个成员，保存在数组istn中
        spin(ks)=-icsp !! 翻转ks
    endif
107 continue !! 完成对s 的4个邻居的操作
if (nstack.eq.0) goto 110 !! 堆栈里的成员处理完后，结束Wolff步骤
js=istn(nstack) !! 从堆栈中读取一个成员
nstack=nstack-1 !! 待操作的成员数减少一个
goto 104 !! 取处理当前成员
```

细致平衡的证明

定义 $P(C, \Gamma)$ 为形成 C 内部连接棒位形的几率. 设边界棒(一端属于 C , 另一端不属于)的数目为 $n = n_a + n_p$, n_a 是两端自旋反号的棒数, n_p 是同号的棒数.

将同号自旋排除在 C 之外的几率: e^{-2Kn_p} , 反号自旋以几率1排除

将 C 翻转得到新位形, 整个过程的几率

$$T(\Gamma', \Gamma) = e^{-2Kn_p} P(C, \Gamma) \frac{1}{N}.$$

反之, 计算从 Γ' 到 Γ 的几率

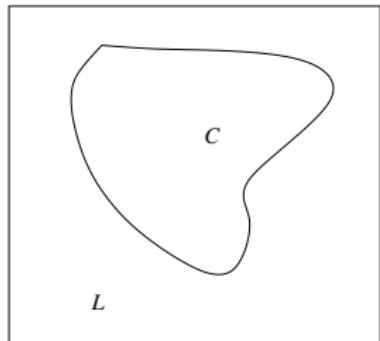
$$T(\Gamma, \Gamma') = e^{-2Kn_a} P(C, \Gamma') \frac{1}{N}.$$

注意到内部连接几率不变

$$P(C, \Gamma) = P(C, \Gamma').$$

因此

$$T(\Gamma', \Gamma) = e^{2K(n_a - n_p)} T(\Gamma, \Gamma').$$



C 表示Wolff集团, 是晶格 L 的子集

细致平衡的证明

由于集团的翻转只改变边界的相互作用能，

$$P_{\text{eq}}(\Gamma)/P_{\text{eq}}(\Gamma') = e^{-(E(\Gamma)-E(\Gamma'))/k_B T} = e^{2(n_p-n_a)K}$$

因此

$$P_{\text{eq}}(\Gamma)T(\Gamma', \Gamma) = P_{\text{eq}}(\Gamma')T(\Gamma, \Gamma')$$

魔术的秘诀在哪里？

配分函数

$$Z = \sum_S W(S), \quad S \text{是自旋位形}$$

以几率 $1 - e^{-2K}$ 连接棒之后

$$Z = \sum_{S,B} W(S)P(B|S) \equiv \sum_{S,B} W(S, B)$$

这里利用了从 S 生成棒位形 B 的几率 $P(B|S)$ 是归一的:

$$\sum_B P(B|S) = 1$$

位形空间多了一个维度 B , 而配分函数不变!

- 集团算法把 $\Gamma = \{S, B\} \rightarrow \Gamma' = \{S', B\}$
- 可以证明 $W(S, B) = W(S', B)$, 所以这个 $S \rightarrow S'$ 的跃迁几率是 $T(\Gamma', \Gamma) = 1$.
- 哲学就是“高一维”的位形空间里两个等几率的位形, 可以轻松互相跃迁, 而其自旋位形相差很大!

Cluster algorithm: Swendsen-Wang algorithm

与Wolff算法非常接近

- 1 从自旋位形开始
- 2 构造集团：与Wolff 算法一样，但是找到所有集团
- 3 以 $1/2$ 几率翻转集团
- 4 回到[1]

Swendsen-Wang

Swendsen-Wang

Swendsen-Wang

Thank you!