

Ising模型

丁历杰

1 模型简介

对于一系列晶格点的集合 S ,其中每一个晶格点都有一个所有与之相邻的晶格点的集合 T ,并另这些晶格点形成一个 d 维的晶格.对于每一个晶格点 $k \in S$ 都有一个离散参数 $\sigma_k \in (+1, -1)$,代表每一个晶格点的自旋.而所有的离散参数集合 $\Sigma \equiv \sigma_{k \in S}$ 为体系的自旋组态.对于每两个相邻的晶格点 $i, j \in S$,引入相互作用参数 J_{ij} ,并假设每个晶格点都和外加磁场 h_j 作用,则体系的Hamilton量可以写作:

$$H(\sigma) = - \sum_{\langle i, j \rangle} J_{ij} \sigma_i \sigma_j - \mu \sum_j h_j \sigma_j \quad (1.0.1)$$

其中 $\langle i, j \rangle$ 代表 i 和晶格点是相邻的,在Hamilton量的计算中只对每一堆晶格点计算一次.第二项是磁场与自旋的相互作用能, μ 是晶格点的磁矩.

Ising模型 [1]是统计力学中关于相变的较为基础的模型.它的最早的提出者是Wilhelm Lenz(1920).后来他让他的学生Ernst Ising对一维的Ising模型进行求解,但是并没有发现相变现象,因此也没有得到更多物理学家的关注.随后,Lars Onsager于1944年对二维的Ising模型进行了解析求解($H = 0$ 的情况, $H \neq 0$ 尚无解析解),并同时发现了二维Ising模型中的相变现象,从而引起了更多学者的关注.之后,随着Landau、Ginzburg等人的努力,人们发现了Ising模型与量子场论之间的联系,并创立了平行的统计场论.

2 统计分析

在Ising模型中,粒子数固定,粒子自旋(广义坐标)的取值范围固定,所以如果定义了温度 T 固定(达到热平衡)的话,那么有其组成的系综就是一个正则系综.其相空间中的概率分布为Boltzman分布:

$$P_T(\Sigma) = \frac{e^{-H(\Sigma)/(k_B T)}}{Z} \quad (2.0.2)$$
$$Z = \sum_{\Sigma} e^{-H(\Sigma)/(k_B T)}$$

因而对于任何一个可观测量 Ω ,都可以写作:

$$\langle \Omega \rangle = \sum_{\Sigma} \Omega(\Sigma) P_T(\Sigma) \quad (2.0.3)$$

2.1 关于参数的讨论

在Hamilton量中参数 J_{ij} 和 h_i 值得进一步探讨.

$$\begin{cases} J_{ij} > 0, \text{称该体系为铁磁性} \\ J_{ij} < 0, \text{称该体系为反铁磁性} \\ J_{ij} = 0, \text{说明晶格点间无相互作用} \end{cases}$$

$$\begin{cases} h_i > 0, \text{则晶格点} i \text{趋于正向} \\ h_i < 0, \text{则晶格点} i \text{趋于反向} \\ h_i = 0, \text{则说明没有外加力场作用} \end{cases}$$

2.2 模型的简化

常见的简化模型为:

1. 没有外加力场作用: $H(\sigma) = -\sum_{\langle i,j \rangle} J_{ij} \sigma_i \sigma_j$
2. 进一步的,所有晶格相互作用是相同的: $H(\sigma) = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j$

2.3 一维Ising模型解析解

对于正则系综,体系的特征函数是Helmoholtz自由能 F ,达到平衡态时 F 取极小值. [2]

$$F = -k_B T \ln Z \quad (2.3.1)$$

2.3.1 $h=0$

对于 $h=0$ 的情况,可以在一维自由边界条件下进行求解, N 个自旋排成一行,端点之间没有相互作用,此时系统的Hamilton量为:

$$E(\Sigma) = -J \sum_{i=1}^{N-1} \sigma_i \sigma_{i+1} = -J(\sigma_1 \sigma_2 + \cdots + \sigma_{N-1} \sigma_N) \quad (2.3.2)$$

令 $K \equiv J/k_B T$, $\sigma'_i = \sigma_i \sigma_{i-1} (i \geq 2)$,则系统的配分函数:

$$Z = \sum_{\sigma_1, \dots, \sigma_N} e^{K(\sigma_1 \sigma_2 + \cdots + \sigma_{N-1} \sigma_N)} = 2 \prod_{i=2}^N \sum_{\sigma'_i} e^{K \sigma'_i} = 2[e^K + e^{-K}]^{N-1} \quad (2.3.3)$$

进而由(2.3.1)式,对于 $N \rightarrow \infty$ 每个粒子的平均自由能 $f = \frac{F}{N} = -k_B T \ln[e^K + e^{-K}]$,除了 $T = 0, \infty$ 外 f 及其温度的高阶导数没有奇点,所以不存在相变.

2.3.2 $h \neq 0$

对于 $h \neq 0$ 的情况,可以周期边界条件进行求解,,此时系统的Hamilton量为:

$$E(\Sigma) = -J \sum_{i=1}^N \sigma_i \sigma_{i+1} - H \sum_{i=1}^N \sigma_i \quad (2.3.4)$$

令 $K \equiv J/K_B T, I = H/K_B T$,则系统的配分函数:

$$Z = \sum_{\sigma_1, \dots, \sigma_N} e^{I\sigma_1} e^{K\sigma_1\sigma_2} e^{I\sigma_2} e^{K\sigma_2\sigma_3} \dots e^{I\sigma_N} e^{K\sigma_N\sigma_1} = \sum_{\sigma_1, \dots, \sigma_N} V_{\sigma_1, \sigma_2} V_{\sigma_2, \sigma_3} \dots V_{\sigma_N, \sigma_1} \quad (2.3.5)$$

其中的 $V_{\sigma_i, \sigma_j} = e^{\frac{I\sigma_i}{2}} e^{K\sigma_i\sigma_j} e^{\frac{I\sigma_j}{2}}$ 可以视 σ_i, σ_j 的取值视为矩阵:

$$V = \begin{bmatrix} e^{K+I} & e^{-K+I} \\ e^{-K+I} & e^{K-I} \end{bmatrix}$$

进而:

$$Z = \sum_{\sigma_1, \dots, \sigma_N} \langle \sigma_1 | V | \sigma_2 \rangle \langle \sigma_2 | V | \sigma_3 \rangle \dots \langle \sigma_N | V | \sigma_1 \rangle = \sum_{\sigma_1} \langle \sigma_1 | V^N | \sigma_1 \rangle = \text{Tr} V^N = \lambda_+^N + \lambda_-^N$$

$$\lambda_{\pm} = e^K \cosh I \pm \sqrt{e^{2K} \sinh^2 I + e^{-2K}}$$

进而由(2.3.1)式,对于 $N \rightarrow \infty$ 每个粒子的平均自由能

$$f = \frac{F}{N} \approx -k_B T \ln e^K \cosh I + \sqrt{e^{2K} \sinh^2 I + e^{-2K}} (\lambda_+ > \lambda_-)$$

同样地除了 $T = 0, \infty$ 外 f 及其温度的高阶导数没有奇点,所以不存在相变.

3 模拟分析

下面对二维的Ising模型进行模拟计算: 所采用的Hamilton量

$$E(\Sigma) = -J \sum_{i,j} \sigma_{i,j} (\sigma_{i-1,j} + \sigma_{i+1,j} + \sigma_{i,j-1} + \sigma_{i,j+1}) - H \sum_{i,j} \sigma_{i,j} \quad (3.0.6)$$

3.1 程序结构

为了加强程序的模块化, 编写了如下的类来进行算法实现。

```

1  class ising
2  {
3  private:
4      struct neighbor
5      {
6          int spin;
7          int ux, uy;
8          int dx, dy;
9          int lx, ly;
10         int rx, ry;
11     } **mark;
12     int size;
13     int up_count;
14     double up_rate;
15     double T;
16     double J;
17     double H;
18     /*
19     neighbor is a struct for every lattice info storage
20     and mark is a 2 dimensional neighbor pointer
21     which contain all the map
22     size is the size of the table
23     m is the average spin
24     T is the temperature
25     h is the megnetic field
26     */
27 public:
28     ising( int size_,
29           double up_rate_,
30           double T_,
31           double J_,
32           double H_);
33     void refresh( double T_,
34                  double J_,
35                  double H_);
36     //refresh the object's parameters as you wish!
37     //will set all spins up
38     void reset( double T_,
39                double J_,
40                double H_);
41     //reset the object's parameters as you wish

```

```

42 //will not reset spins' direction
43
44
45 int get_size(); //size getting
46 int get_up_count(); //up_count getting
47 double get_up_rate(); //up_rate getting
48 double get_T();
49 double get_J();
50 double get_H();
51 double get_z(); //z = (exp(2J/T)-1)
52 double get_energy(); //get the energy
53 double get_m(); //get m = average(s)
54
55 //Algorithms
56 void metropolis(int steps);
57 void swendsen_wang(int steps);
58 void wolff(int steps);
59 void worm(int steps);
60
61
62
63 //output data methods
64 void output_mark_pos(std::string filename);
65 void output_n_E_m( int init_step ,
66                   int steps ,
67                   int per_step ,
68                   std::string filename);
69 void output_T_E_C_m(int equi_step ,
70                   int ave_step ,
71                   int step_per_ave ,
72                   double T_start ,
73                   double T_end ,
74                   double delta_T ,
75                   std::string filename);
76 void output_H_E_C_m(int equi_step ,
77                   int ave_step ,
78                   int step_per_ave ,
79                   double H_start ,
80                   double H_end ,
81                   double delta_H ,
82                   std::string filename);

```

```

83
84 void output_T_H_E_C_m(int equi_step,
85                        int ave_step,
86                        int step_per_ave,
87                        double T_start,
88                        double T_end,
89                        double delta_T,
90                        double H_start,
91                        double H_end,
92                        double delta_H,
93                        std::string filename);
94 void output_H_loop_E_C_m(int equi_step,
95                          int ave_step,
96                          int step_per_ave,
97                          double H_start,
98                          double H_end,
99                          double delta_H,
100                         std::string filename);
101 };

```

3.2 模拟算法

细致平衡 由主方程 [5]可以到的细致平衡解:

$$\frac{P(x \rightarrow x')}{P(x' \rightarrow x)} = \frac{P(x')}{P(x)} \quad (3.2.1)$$

3.2.1 Metropolis

Metropolis算法 [3]是一种Markov Chain Monte Carlo(MCMC)算法,由Metropolis等人在1953年提出 [4].

算法导出 为了得到Metropolis算法,将细致平衡中的转移概率分为两个部分:提议概率 $g(x \rightarrow x')$ 和接受概率 $A(x \rightarrow x')$.

$$P(x \rightarrow x') = g(x \rightarrow x')A(x \rightarrow x')$$

进而:

$$\frac{A(x \rightarrow x')}{A(x' \rightarrow x)} = \frac{P(x') g(x' \rightarrow x)}{P(x) g(x \rightarrow x')}$$

选择满足上述条件的接受概率,便可满足细致平衡条件,而Metropolis算法为:

$$A(x \rightarrow x') = \min \left(1, \frac{P(x')}{P(x)} \frac{g(x' \rightarrow x)}{g(x \rightarrow x')} \right) \quad (3.2.2)$$

具体算法 在本次的模拟中,具体采用的算法是.

1. 以均匀分布随机选取一个晶格点,计算它的局域能量 E_{old}
2. 反转它,计算新的局域能量 E_{new}
3. 以概率 $\min(1, e^{-(E_{new}-E_{old})/k_B T})$ 接受新的状态
4. 重复上述步骤

3.2.2 Swendsen-Wang

Swendsen-Wang方法是处理临界慢化问题的一种加速方法,整个系统被分为一系列的团簇集团,每个团簇集团再被随机的赋予新的自旋值.由Swendsen和Wang于1987年提出 [6],具体的团簇产生由逾渗模型中的Hoshen-Kopelman的逾渗集团标记法 [7]产生.

具体算法

1. 产生初始自旋构型(比如可以先由Metropolis法热化)
2. 进行自旋键逾渗变换
3. 由Hoshen-Kopelman的逾渗集团标记法将所有联键的集团逐一标识
4. 对每个集团分别产生(+1,-1)的随机整数,赋予该集团中所有的晶格点该自旋值.
5. 重复2至4步

3.2.3 Worff

Worff算法也是一种集团算法,由Worff于1989年提出 [8].与Swendsen-wang 算法所不同的是,Worff算法一次只反转一个集团,并且集团的产生并不是连接所有的键逾渗中的联键集团. [9]

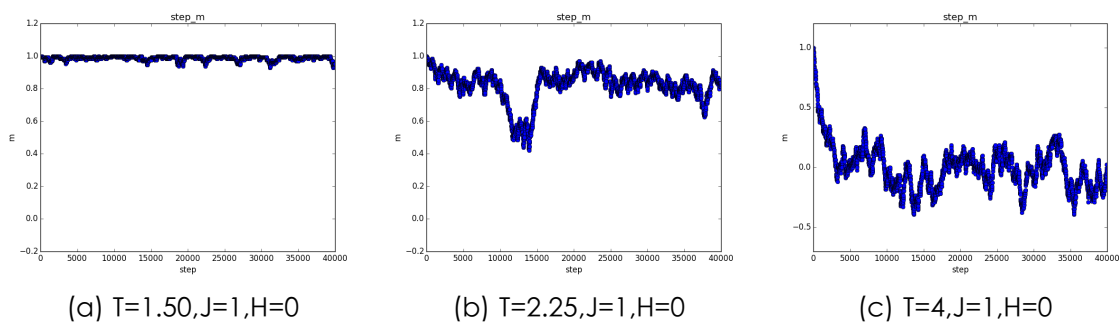
具体算法

1. 选取一个随机晶格点*i*,将其放入集团标记数组C
2. 遍历所有未遍历过的C中晶格点的邻居晶格点*j*,
3. 如果 $\sigma_j = \sigma_i$, 则以 $p = 1 - \exp(-2J/k_B T)$ 的概率接受它加入C(若拒绝了之后便不会再遍历到)
4. 当C中晶格点的邻居中没有未遍历过的了,C中所有的自旋反向.
5. 重复上述步骤

3.3 模拟分析

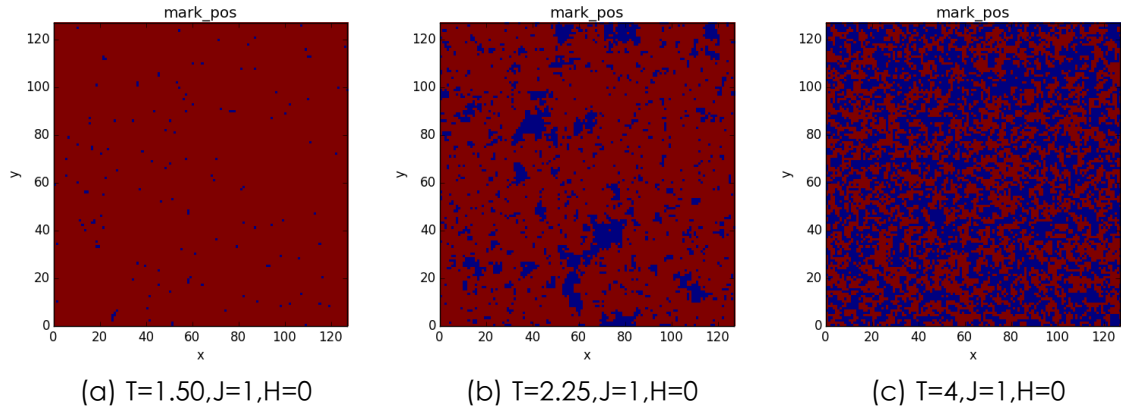
不同温度下,由于涨落的不同,从而构成平衡态的状态大不相同.

Figure 1: *m*随演化步数的变化



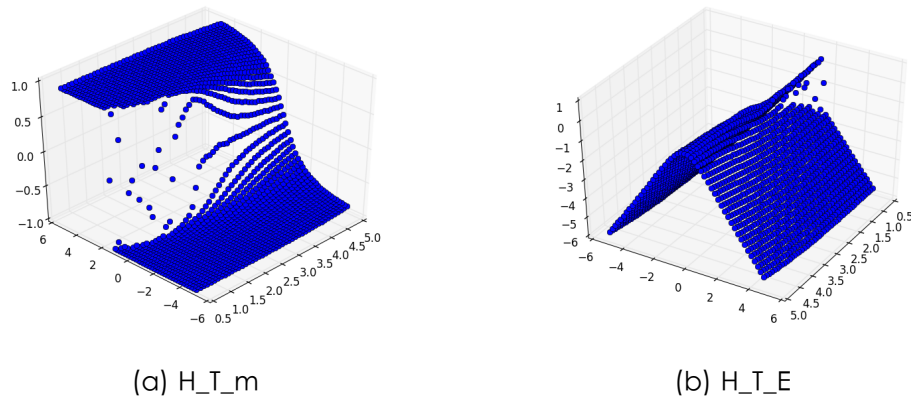
可以看到不同温度下平均磁矩*m*随演化步数的变化

Figure 2: 体系稳态状态



演化了 $1e8$ 步后体系稳态状态

Figure 3: m 关于 (H,T) 变化

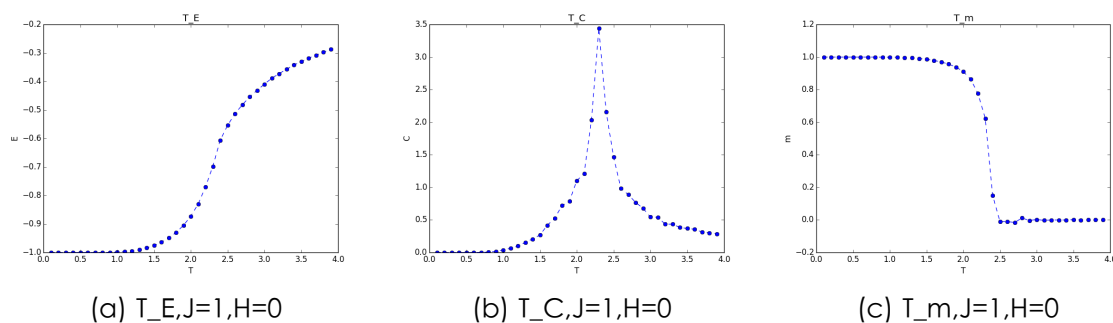


而平均自旋 m 关于参数 (H,T) 的变化($Size = 64$) 可以看到,在 T 较小($<T_c$)的部分, m 随 H 的变化出现了断裂,也就是说存在一级相变,而在较大的 T 出,都是连续的,所以是存在二级相变. E 页同样出项了断裂.

3.3.1 二级相变

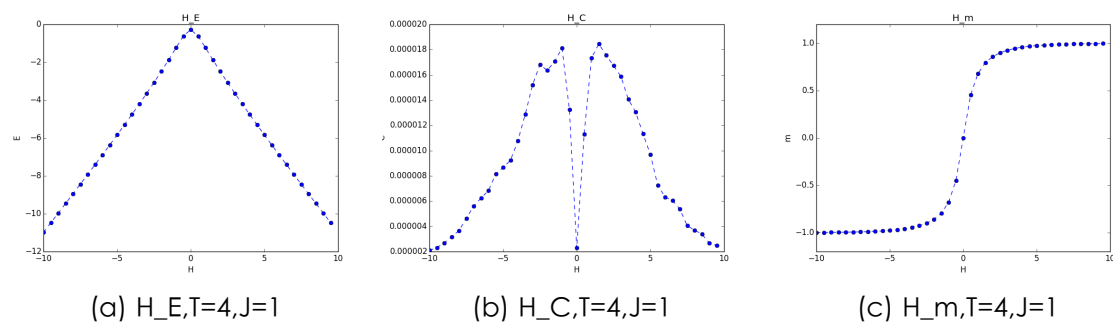
随着温度的改变,观察平衡态统计量的变化(E :平均能量, C :比热,平均自旋),模拟的大小为 128×128 .

Figure 4: $H=0$,随温度变化



$H=0$ 时,随温度的变化

Figure 5: $T > T_c$,随磁场变化

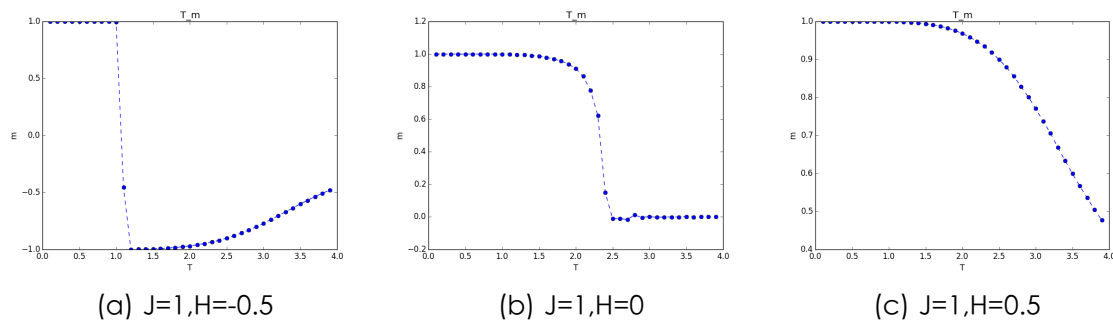


$T > T_c$ 时,随磁场的变化

3.3.2 一级相变

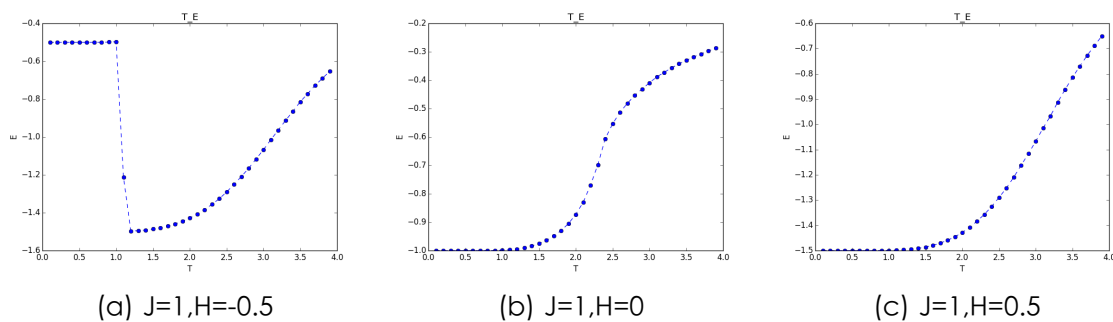
随着温度的降低,以及磁场的改变,体系将出项一级相变:

Figure 6: m_T 趋势随磁场变化



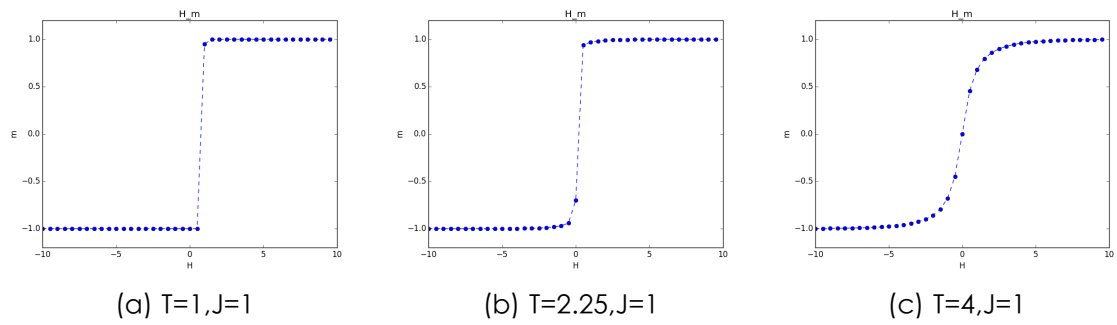
改变磁场, m 随 T 的变化趋势改变

Figure 7: E_T 趋势随磁场变化



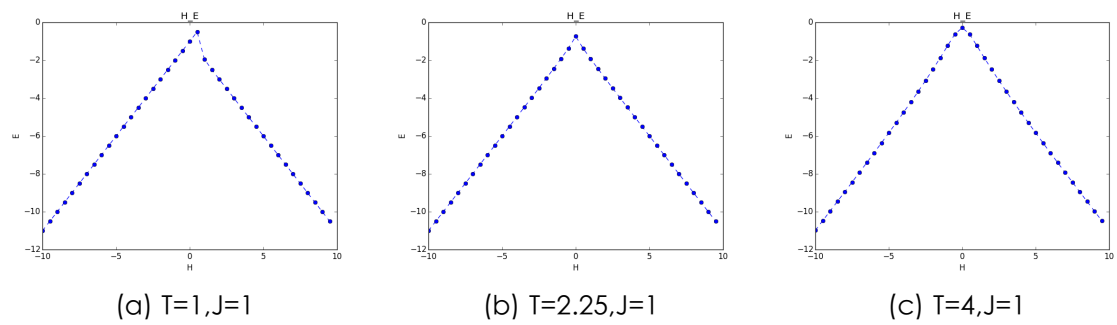
改变磁场, E 随 T 的变化趋势改变

Figure 8: m_H 趋势随温度变化



改变温度, m 随 H 的变化趋势改变

Figure 9: E_H 趋势随温度变化

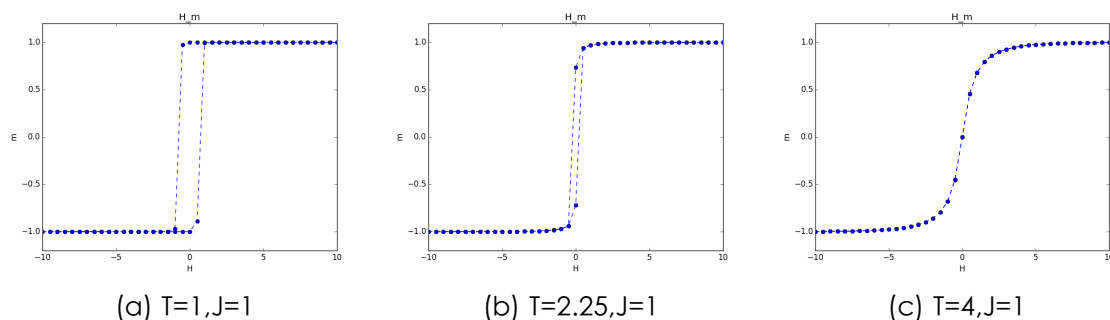


改变温度, E 随 H 的变化趋势改变

3.3.3 磁滞回线

当对磁场H为一级相变时,演增大和减小磁场的方向改变磁场: 不难发现,磁滞回线随着

Figure 10: 磁滞回线



温度的增高逐渐消失,这是因为在模拟中自旋反转的机率正比于 $\exp(-\Delta E/k_B T)$, T 越小,该机率越小,系统处于亚稳态的惯性越强,故更倾向于留在原来态.这样从正反方向演化,也就出现了相变点的不同,进而产生了磁滞回线.

4 总结

概要的介绍了Ising模型,并用Monte Carlo方法对二维Ising模型进行了模拟,讨论了稳态组态,一级相变,二级相变等若干问题. Ising模型是统计物理中同时具备了:表述简单,内涵丰富,应该广泛三种有点. 其扩展模型有:

1. XY模型: 自旋可以在平面中任意取向.
2. Heisenberg模型: 自旋在三维空间中任意取向.
3. Potts模型: 自旋不再是 ± 1 而是可以取 q 个离散状态.
4. 时钟模型: 二维中间中 q 个离散取向(XY模型+Potts模型)
5. Ising自旋玻璃: Ising模型中 J_{ij} 取满足Gauss分布的随机变量.

相关的应用也有很多,比如磁化模拟,投票模型,神经网络模型等.

参考文献

- [1] https://en.wikipedia.org/wiki/Ising_model
- [2] 汪志诚. 热力学 · 统计物理[M]// 高等教育出版社, 2008.第九章 9.5
- [3] https://en.wikipedia.org/wiki/Metropolis-Hastings_algorithm#cite_note-Metropolis-1
- [4] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics 21, 1087–1092 (1953).
- [5] https://en.wikipedia.org/wiki/Master_equation
- [6] Swendsen, R. H. & Wang, J.-S. Nonuniversal critical dynamics in Monte Carlo simulations. Phys. Rev. Lett. 58, 86–88 (1987).
- [7] Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. (1976).
- [8] Ulli Wolff. Collective Monte Carlo Updating for Spin Systems. 62, 361–364. 15 p (1989).
- [9] Krauth, W. Cluster Monte Carlo algorithms. arXiv cond-mat.stat-mech, (2003).