量子物理计算方法选讲实验报告

李昊恩　2021010312

Classical Monte Carlo, Task 5

目录

1 引言

1.1 模型

本次实验采用经典 Monte Carlo 方法，求解二维经典 Ising 模型. 具体来说，我们考虑一个 $L \times L$ 的方形格点，其哈密顿量为：

$$\mathscr{H} = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j + H \sum_i \sigma_i, \tag{1}$$

其中 $\langle i, j \rangle$ 表示近邻相互作用，σ_i 表示 i 位点的自旋，自旋取 ± 1.

在本实验中，我们希望模拟其在各温度下的平均能量、平均磁化、热容和极化率.

我们首先用经典统计力学推导这些物理量的表达式. 事实上，该模型的配分函数为：

$$Z = \sum_{\{\sigma_i\}} e^{-\beta E(\{\sigma_i\})}, \tag{2}$$

这里的记号 $\{\sigma_i\}$ 用来指代二维格点的每个位置的自旋构成的集合（也就是组态），每个自旋的取值为 ± 1. 因此，能量平均值为：

$$\langle E \rangle = \frac{1}{Z} \sum_{\sigma_i} E(\{\sigma_i\}) e^{-\beta E(\{\sigma_i\})}. \tag{3}$$

热容定义为 $\langle E\rangle$ 关于温度 T 的导数：

$$C := \frac{\partial \langle E\rangle}{\partial T} = \frac{\partial \langle E\rangle}{\partial \beta}\frac{\partial \beta}{\partial T} = -\frac{1}{kT^2}\frac{\partial \langle E\rangle}{\partial \beta}, \tag{4}$$

而：

$$\begin{aligned}\frac{\partial\langle E\rangle}{\partial\beta} &= \frac{1}{Z^2}\begin{bmatrix} \left(-\sum_{\{\sigma_i\}}\left[E\left(\{\sigma_i\}\right)\right]^2 e^{-\beta E(\{\sigma_i\})}\right) Z \\ -\left(\sum_{\{\sigma_i\}} E\left(\{\sigma_i\}\right) e^{-\beta E(\{\sigma_i\})}\right)\left(-\sum_{\{\sigma_i\}} E\left(\{\sigma_i\}\right) e^{-\beta E(\{\sigma_i\})}\right) \end{bmatrix} \\ &= \langle E\rangle^2 - \left\langle E^2\right\rangle \end{aligned} \tag{5}$$

所以：

$$C = \frac{1}{kT^2}\left(\left\langle E^2\right\rangle - \langle E\rangle^2\right). \tag{6}$$

类似地，平均磁化的表达式为：

$$\langle M\rangle = \frac{1}{Z}\sum_{\{\sigma_i\}} m(\{\sigma_i\}) e^{-\beta E(\{\sigma_i\})}, \tag{7}$$

平均磁化率为：

$$\begin{aligned}\chi = \frac{\partial\langle M\rangle}{\partial H} &= \frac{1}{Z^2}\begin{bmatrix} \left(-\beta\sum_{\{\sigma_i\}} m\left(\{\sigma_i\}\right)\frac{\partial E(\{\sigma_i\})}{\partial H} e^{-\beta E(\{\sigma_i\})}\right) Z \\ -\left(\sum_{\{\sigma_i\}} m\left(\{\sigma_i\}\right) e^{-\beta E(\{\sigma_i\})}\right)\left(-\beta\sum_{\{\sigma_i\}}\frac{\partial E(\{\sigma_i\})}{\partial H} e^{-\beta E(\{\sigma_i\})}\right) \end{bmatrix} \\ &= \beta\frac{1}{Z^2}\left[\left(\sum_{\{\sigma_i\}}\left[m\left(\{\sigma_i\}\right)\right]^2 e^{-\beta E(\{\sigma_i\})}\right) Z - \left(\sum_{\{\sigma_i\}} m\left(\{\sigma_i\}\right) e^{-\beta E(\{\sigma_i\})}\right)^2\right] \\ &= \frac{1}{kT}\left(\left\langle M^2\right\rangle - \langle M\rangle^2\right) \end{aligned} \tag{8}$$

二维情形下，经典 Ising 模型具有相变，其相变临界温度的严格解为：

$$T_c = \frac{2J}{\log\left(1+\sqrt{2}\right)} \simeq 2.269J. \tag{9}$$

1.2 Markov 链蒙特卡洛（MCMC）

Metropolis(-Hastings) 算法是 Markov 链蒙特卡洛方法的一种实现方式. 它采用一种接收—拒绝式的更新步骤，能够实现重要性采样（impotrance sampling）. 具体来说，在每一步迭代中，我们首先依据目前状态 x，随机生成 Markov 链中的一个状态 y，也就是考虑状态转移 $Q(x\to y)$，然后计算所谓的 “接受概率”：

$$A(x\to y) := \min\left\{1, \frac{P(y)}{P(x)}\frac{Q(y\to x)}{Q(x\to y)}\right\}, \tag{10}$$

在 Metropolis 算法中，分布 Q 是对称的（$Q(x\to y) = Q(y\to x)$，例如 x 为中心的 Gauss 分布），于是 $A(x\to y)$ 进一步约化为：

$$A(x\to y) = \min\left\{1, \frac{P(y)}{P(x)}\right\}. \tag{11}$$

我们以随机数生成的方式来判定是否接受新的状态 y：按照 $(0,1)$ 上的均匀分布生成一个随机数 $u \in (0,1)$，如果：1）$u \leq A(x \to y)$，接受新的状态，并且更新 $x \mapsto y$，进入下一步迭代更新；2）如果 $u > A(x \to y)$，拒绝新的状态，仍然保持状态 x 进入下一步迭代.

在经典 Ising 模型的模拟例子中，Markov 链上的每个状态就是一种自旋排布方式（也就是组态，configuration），然后每次从 $\{\sigma_x\}$ 出发，随机尝试翻转其中一个自旋作为这一步的 $\{\sigma_y\}$，并依照

$$\alpha = \frac{P(\{\sigma_y\})}{P(\{\sigma_x\})} = \exp[-\beta[E(\{\sigma_y\}) - E(\{\sigma_x\})]] = \exp(-\beta \Delta E), \tag{12}$$

其中 $\Delta(E)$ 代表翻转自旋带来的能量变化，其表达式为：

$$\Delta E = 2\sigma_j \left(J \sum \sigma_{\text{near}} - H \right), \tag{13}$$

这里 H 表示外场.

然后，从 $\mathrm{U}(0,1)$ 中随机生成一个随机数 $r \in (0,1)$.

- 如果 $r < \alpha$，则接受这次翻转，定义新的 $\{\sigma_x\}$ 为 $\{\sigma_y\}$；
- 如果 $r > \alpha$，则拒绝这次翻转，定义新的 $\{\sigma_x\}$ 为 $\{\sigma_x\}$.

1.3 临界指数

临界指数是用来描述物理量在临界点附近行为的指数. 我们用约化临界温度 $t = \frac{T - T_c}{T_c}$ 作为参数.

在临界温附近，经典 Ising 模型的各物理量的临界指数是：

$$M \propto (-t)^{\beta}, \quad t \to -0, \tag{14}$$

$$\chi \propto t^{-\gamma}, \quad t \to +0, \tag{15}$$

$$\chi \propto (-t)^{-\gamma'}, \quad t \to -0, \tag{16}$$

$$M \propto H^{\delta}. \tag{17}$$

在本实验中，我们关心 $\langle M \rangle$ 的临界指数，严格解给出当 $T < T_c$ 时，$\beta = \frac{1}{8}$，当 $T > T_c$ 时，$\beta = -\frac{7}{8}$.

2 代码实现

在本次实验中，我们取 $J = 1$，$H = 0$，探究当外场趋于 0 时，二维经典 Ising 模型的行为.

首先定义计算 ΔE，$\langle E \rangle$，$\langle M \rangle$ 以及进行每一步 MCMC 更新的函数，代码思路见注释.

```
J = 1.0

```

```
# initialize the configuration by randomly choosing the spin patterns at each site
def init_config_ising(L):
    return np.random.choice([1,-1], size = (L,L))

# calculate the energy difference after flipping the spin at a certain site
def delta_E(config, site):
    delta_E = 0.0
    L = np.shape(config)[0]
    for nbs in [(1,0),(-1,0),(0,1),(0,-1)]: # take summation over each neighborhood of this site, don't forget to multiply by 2
        delta_E += config[(site[0] + nbs[0])%L, (site[1] + nbs[1])%L] * J * config[site] * 2
    return delta_E

def MCMC_update(config, T):
    # implement of metropolis algorithm
    L = np.shape(config)[0]
    random_site = (np.random.randint(L), np.random.randint(L))
    energy = delta_E(config, random_site)
    r = np.random.rand() # random number sampled from U(0,1)
    alpha = np.exp(-energy/T) # alpha =  the proportion of probabilities
    if r < alpha or energy < 0:
        config[random_site] *= -1
    return config

def get_energy(config):
    L = np.shape(config)[0]
    energy = 0.0
    for i in range(L):
        for j in range(L):
            energy += - config[i,j] * config[(i+1)%L, j] - config[i,j] * config[i, (j+1)%L] # get the exact energy of a configuration
    return energy

def get_magn(config):
    magn = np.sum(config) # get the exact magnetization of a configuation
    return magn
```

随后，用前面定义的函数进行 MC 迭代。由于 MC 迭代更新十分耗时，我们在这里只模拟几个较小的体系 $L = (2, 4, 6)$：

```
n_iter = 100000
L_list = [2, 4, 6]
```

```
T_list = np.linspace(1, 3.5, 100)
Tc = 2/np.log(1+np.sqrt(2))

def MCMC(L, T, n_iter, n_avg=10000): # implement of the whole MCMC iteration

    state = init_config_ising(L)

    for _ in range(n_iter):
        state = MCMC_update(state, T) # implement MCMC update for n_iter = 100000 times (following the suggestion given in the lecture note)

    energies = []
    magnetizations = []
    for _ in range(n_avg*10): # for calculation of the physical observations, further sample 10 * n_average configurations
        state = MCMC_update(state, T)
        sample_energy = get_energy(state)
        sample_magnetization = get_magn(state)
        energies.append(sample_energy)
        magnetizations.append(sample_magnetization)

    subsample_energies = np.random.choice(energies, size=n_avg)
    subsample_magnetizations = np.abs(np.random.choice(magnetizations, size=n_avg))
    avg_energy = np.mean(subsample_energies) / (L*L)
    avg_magnetization = np.mean(subsample_magnetizations) / (L*L)
    specific_heat = np.var(subsample_energies) / (T ** 2)
    susceptibility = np.var(subsample_magnetizations) / T

    return state, avg_energy, avg_magnetization, specific_heat, susceptibility # output the final physical observations of one MCMC iteration


# setting of the figures (5 subfigures)
fig = plt.figure(figsize=(15, 30))
ax_energy = fig.add_subplot(421)
ax_energy.set_xlabel(r'$T$')
ax_energy.set_ylabel(r'$\langle E\rangle$')
ax_magnetization = fig.add_subplot(422)
ax_magnetization.set_xlabel(r'$T$')
ax_magnetization.set_ylabel(r'$\langle M \rangle$')
ax_specific_heat = fig.add_subplot(423)
ax_specific_heat.set_xlabel(r'$T$')
```

```
ax_specific_heat.set_ylabel(r'$\langle C\rangle$')
ax_susceptibility = fig.add_subplot(424)
ax_susceptibility.set_xlabel(r'$T$')
ax_susceptibility.set_ylabel(r'$\langle\chi\rangle$')
ax_exponent = fig.add_subplot(212)
ax_exponent.set_xlabel(r'$N|T-T_c|$')
ax_exponent.set_ylabel(r'$N^{1/8}\langle M\rangle$')
ax_exponent.set_xscale('log') # logarithm coordinate in the critical exponent graph
ax_exponent.set_yscale('log')

###### start MC iteration ######
for L in L_list:
    avg_energies=[]
    avg_magnetizations=[]
    specific_heats=[]
    susceptibilities=[]

    for T in tqdm(T_list):
        state, avg_energy, avg_magnetization, specific_heat, susceptibility = MCMC(L, T, n_iter)
        avg_energies.append(avg_energy)
        avg_magnetizations.append(avg_magnetization)
        specific_heats.append(specific_heat)
        susceptibilities.append(susceptibility)

    print('L = {}, iteration {} MC iteration done.'.format(L, n_iter))
    # plot
    ax_energy.scatter(T_list, avg_energies, label=f'L = {L}')
    ax_magnetization.scatter(T_list, avg_magnetizations, label=f'L = {L}')
    ax_specific_heat.scatter(T_list, np.array(specific_heats)/(L*L), label=f'L = {L}')
    ax_susceptibility.scatter(T_list, np.array(susceptibilities)/(L*L), label=f'L = {L}')
    ax_exponent.scatter(np.abs(T_list-Tc) * (L*L), np.array(avg_magnetizations) * ((L*L) ** (1/8)), label=f'L = {L}')

# exact solution (beta = 1/8, -7/8)
ax_exponent.plot(np.abs(T_list-Tc) * (L*L), (np.abs(T_list-Tc) * (L*L)) ** (1/8), label=r'exact at $T<T_c$')
ax_exponent.plot(np.linspace(0.1, 3.5-Tc) * (L*L), 10*(np.linspace(0.1, 3.5-Tc) * (L*L)) ** (-7/8), label=r'exact at $T>T_c$')
ax_energy.legend()
```

```
ax_magnetization.legend()
ax_specific_heat.legend()
ax_susceptibility.legend()
ax_exponent.legend()
plt.show()
```

我们首先进行 10^5 次迭代，此时应当体系接近平衡状态，再次采样 10^5 次，并根据这些样本近似计算 $\langle E\rangle$，$\langle M\rangle$，以及 C 和 χ：

$$\langle E\rangle \approx \frac{1}{N}\sum_{i=1}^{N} E_{i,\text{sampled}}, \tag{18}$$

$$\langle M\rangle \approx \frac{1}{N}\sum_{i=1}^{N} M_{i,\text{sampled}}, \tag{19}$$

$$C \approx \frac{1}{T^2}\frac{1}{N}\sum_{i=1}^{N}(E-\langle E\rangle)^2 = \frac{1}{T^2}\mathrm{Var}(E_i), \tag{20}$$

$$\chi \approx \frac{1}{T}\mathrm{Var}(M_i). \tag{21}$$

大数定律保证了这些近似是合理的.

随后，我们对每个温度进行一次蒙特卡洛迭代，得到这些物理量关于温度的函数，并绘制成曲线.

3 结果

运行以上代码，得到的结果为：

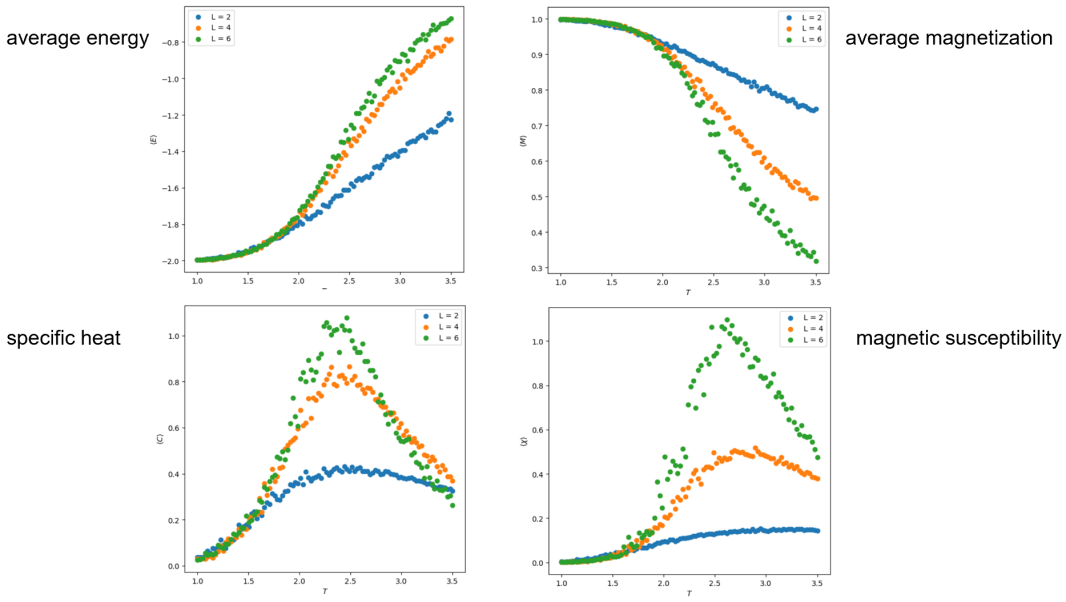

图 1: 模拟结果（物理量）

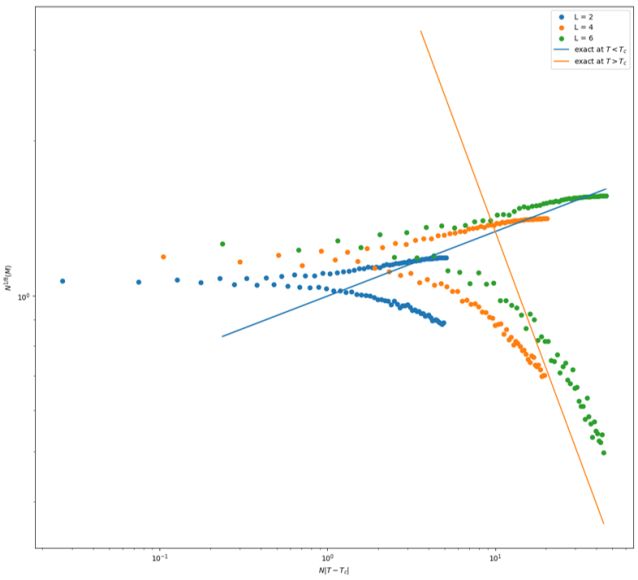

图 2: 模拟结果（临界指数）

可以看到，其在临界点附近的行为的确可以和严格解符合地较好. 另外，这里的模拟结果和讲义中给出的模拟结果也是类似的.

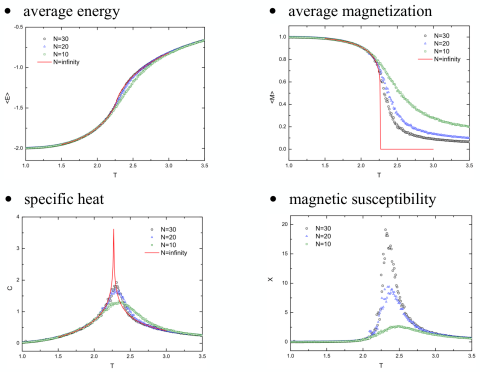

图 3: 讲义模拟结果（物理量）

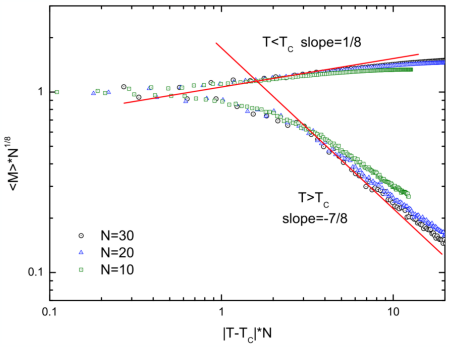

图 4: 讲义模拟结果（临界指数）