《计算物理》第 6 次作业

1900017812 高乐耘*

2023 年 1 月 10 日

I. 蒙特卡洛积分

A. 一维蒙特卡洛积分

点击此链接得到以下积分的精确数值结果：

$$\mu_f = I = \int_0^1 \mathrm{e}^{-100(x-0.5)^2}\mathrm{d}x = 0.1772453850902791 \tag{1}$$

点击此链接得到以下积分的精确数值结果：

$$\mu_f^2 + \sigma_f^2 = \int_0^1 \mathrm{e}^{-200(x-0.5)^2}\mathrm{d}x = 0.12533141373155 \tag{2}$$

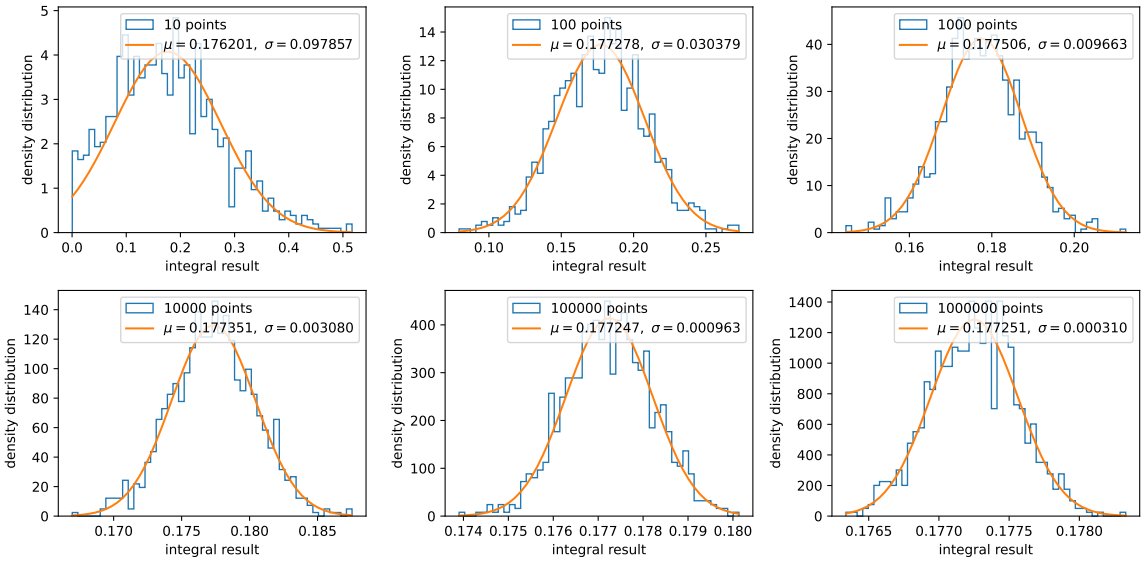

图 1： 不同撒点数下 1000 次积分结果的频率密度分布直方图及其高斯拟合曲线

程序 mcint/1d.c 使用最简单的方法完成了对 I 的蒙特卡洛积分。分别选择撒点数 10，100，1000，10000，100000，1000000，在 $[0,1)$ 区间内产生相应数量的随机数 x，并分别计算其对应的函数值 $y = \mathrm{e}^{-100(x-0.5)^2}$，对 y 取平均后立即得到积分结果。每个撒点数重复实验 1000 次，得到积分结果分布直方图，与拟合的高斯分布曲线一同绘制于图 1 中。

*电子邮件地址: seeson@pku.edu.cn 手机号: 13759115414

图 1 中各实际分布的平均值和标准差列于表 I 中。对于撒点数 N，平均误差可以表示为：

$$e_{\text{obs}} = \mu_{\text{obs}} - I \tag{3}$$

标准差的预期值为：

$$\sigma_s = \frac{1}{\sqrt{N}}\sigma_f \tag{4}$$

为方便比较，表 I 中一并给出了这两个量的计算结果。

表 I: 不同撒点数下积分结果的平均值和标准差

撒点数 N	10	100	1000
平均值 μ_{obs}	0.17620137369137473	0.17727837398565746	0.17750566684307933
标准差 σ_{obs}	0.09785658396760209	0.03037926156842198	0.009662751147223531
平均误差 e_{obs}	-0.0010440113989043653	3.2988895378366356e-05	0.0002602817528002399
预期标准差 σ_s	0.09691000319665079	0.030645633815561502	0.009691000319665079
撒点数 N	10000	100000	1000000
平均值 μ_{obs}	0.17735060035620115	0.17724740235052067	0.1772514541220718
标准差 σ_{obs}	0.003080245047499931	0.0009626821919801426	0.0003101951168015013
平均误差 e_{obs}	0.00010521526592205399	2.017260241571339e-06	6.069031792704793e-06
预期标准差 σ_s	0.00306456338155615	0.0009691000319665078	0.00030645633815561504

$\sigma_s \propto N^{-1/2}$ 是蒙特卡洛积分的重要特征，表 I 中标准差 σ_{obs} 与预期标准差 σ_s 的精确符合很好地验证了这一结论。由于蒙特卡洛方法固有的涨落特征，平均误差 e_{obs} 的绝对值并非随撒点数 N 的增大而单调减小，但撒点数充分大时，e_{obs} 一般在较小的量级。同时，e_{obs} 的数量级始终接近或小于 σ_s 的数量级，这验证了在中心极限定理的前提下 e_{obs} 是 σ_s 很好的估计。整体上看来，基于这种最简单抽样方法的蒙特卡洛方法对本题中的一维积分是有效的，但显然不是最高效的。

B. 多维蒙特卡洛积分

显然，根据上题结果，可以轻松得到（较高数值精度下的）精确值

$$\mu_f = I = 0.17724538509027919 = 1.7265311851403458\text{e-}07 \tag{5}$$

$$\mu_f^2 + \sigma_f^2 = 0.125331413731559 = 7.630261930997968\text{e-}09 \tag{6}$$

现在我们看到 $\sigma_f \gg \mu_f$，被积函数的光滑性非常差，这将使得后续蒙特卡洛算法的收敛遇到较大的困难。假设我们想要收敛到大约 10% 的精度，令 $\sigma_s = 0.1\mu_f$，得到：

$$N = \left(\frac{\sigma_f}{\sigma_s}\right)^2 = 100\left(\frac{\sigma_f}{\mu_f}\right)^2 = 25596990 \sim 10^7 \tag{7}$$

按照相同的算法，对于给定的取点数目 N，随机产生 N 对 9 元组 $\boldsymbol{x}$，便可在 $\sigma_s = \sigma_f/\sqrt{N}$ 的精度下估计出这个积分值：

$$\hat{I} = \langle y \rangle\,,\ y := \mathrm{e}^{-100(\boldsymbol{x}-0.5)^2} \tag{8}$$

上式中向量与标量的代数运算按照 numpy 的广播规则理解。同样，重复实验 1000 次将获得：

$$\mu_{\text{obs}} = \left\langle \hat{I} \right\rangle, \ \sigma_{\text{obs}} = \sqrt{\left\langle \hat{I}^2 \right\rangle - \left\langle \hat{I} \right\rangle^2}, \ e_{\text{obs}} = \mu_{\text{obs}} - I \tag{9}$$

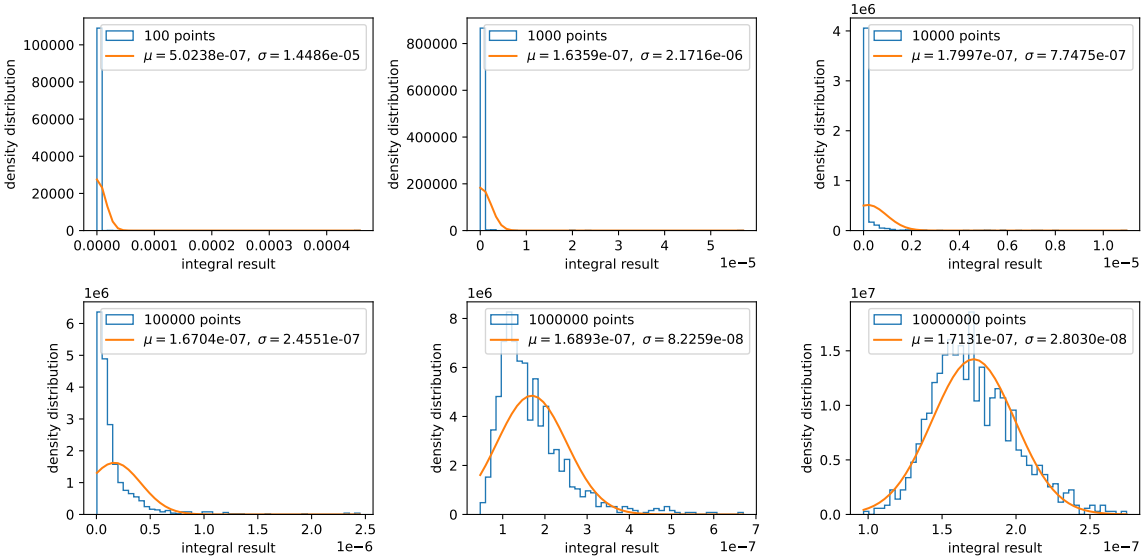

图 2: 不同撒点数下 1000 次积分结果的频率密度分布直方图及其高斯拟合曲线

程序 mcint/9d.c 以撒点数 100，1000，10000，100000，1000000，10000000 实现了这一算法。其积分结果分布直方图，与拟合的高斯分布曲线一同绘制于图 2 中。当撒点数小于 1000000 时，由于 $\sigma_s > \mu_f$，不出意外地并未出现收敛结果；当撒点数达到 1000000 时，σ_s 和 μ_f 数量级相当，对应图中显示的 $\sigma_{\text{obs}} \sim \mu_{\text{obs}}$，显然不符合中心极限定理的前提，其结果分布曲线与高斯分布曲线相差较大；当撒点数达到 10000000 时，由于 σ_s 比 μ_f 小一个量级，对应图中显示的 $\sigma_{\text{obs}} \sim 0.1\mu_{\text{obs}}$，此时中心极限定理预言的渐进高斯分布行为开始显现，分布曲线的对称性明显增强。

表 II: 不同撒点数下积分结果的平均值和标准差

撒点数 N	100	1000	10000
平均值 μ_{obs}	5.023790705055333e-07	1.6358522258947532e-07	1.799655067541637e-07
标准差 σ_{obs}	1.4485512471052439e-05	2.1716118257155676e-06	7.747493710673112e-07
平均误差 e_{obs}	3.2972595199149876e-07	-9.067895924559254e-09	7.312388240129113e-09
预期标准差 σ_s	8.735119988814485e-06	2.762287479951831e-06	8.735119988814485e-07
撒点数 N	100000	1000000	10000000
平均值 μ_{obs}	1.6704287471532882e-07	1.6893498316495875e-07	1.7131225770657013e-07
标准差 σ_{obs}	2.4550887141693176e-07	8.225859487009784e-08	2.802986147518743e-08
平均误差 e_{obs}	-5.610243798705764e-09	-3.718135349075824e-09	-1.3408608074644453e-09
预期标准差 σ_s	2.7622874799518307e-07	8.735119988814485e-08	2.7622874799518308e-08

$\sigma_s \propto N^{-1/2}$ 是的重要特征被表 II 中标准差 σ_{obs} 与预期标准差 σ_s 的精确符合再次有力验证。由于上表中的所有撒点数均不足以达到 $\sigma_s = 0.1\mu_f$ 的标准，随着 N 的增加，平均误

差 e_{obs} 的绝对值如预期一样单调地减小。同时，当 N 充分大（$N = 10000000$）时，e_{obs} 的数量级接近 σ_s 的数量级，这再次验证了在中心极限定理的前提下 e_{obs} 是 σ_s 很好的估计。整体上看来，这种算法对本题的积分而言是非常低效的，可能的改进策略为通过适当的第一类变量代换，使得抽样函数集中在 $\boldsymbol{x} = 0.5$ 附近密集撒点，来改善新的被积函数的光滑性。

II. 二维 Potts 模型

A. 双自旋态蒙特卡洛模拟

程序 potts/q2.c 中给出了判断是否接受一个自旋状态的改变并重新计算能量和磁化强度的高效实现。不考虑计算机多级存储器性能随格点阵列规模 n 的增大因局部性的损失不可避免地下降时（此时你会不禁感慨，拥有一款超大三级缓存的 CPU 才是多么幸运的一件事），这一实现的时间复杂度是 $\mathcal{O}(1)$ 的：

```
#define likely(e)    (__builtin_expect((e), 1))
static void lmove(int n, int L[n][n], int i, int j, int v,
                  int *Ep, int *Mp, double T)
{
  // 对于温度T, 接收或拒绝改变L[i][j] -> v, 并更新能量*Ep和磁化强度*Mp
  int E = *Ep, M = *Mp;
  if(likely(i > 0))       E -= (v == L[i - 1][j]) - (L[i][j] == L[i - 1][j]);
  if(likely(i < n - 1))   E -= (v == L[i + 1][j]) - (L[i][j] == L[i + 1][j]);
  if(likely(j > 0))       E -= (v == L[i][j - 1]) - (L[i][j] == L[i][j - 1]);
  if(likely(j < n - 1))   E -= (v == L[i][j + 1]) - (L[i][j] == L[i][j + 1]);
  if(E <= *Ep || drand48() < exp((*Ep - E) / T)) {
    M += v - L[i][j];
    L[i][j] = v;
    *Ep = E, *Mp = M;
  }
}
```

由于这段代码为程序花费时间最多的代码，我们将函数声明为 `static` 并指定适当的命令行参数使得编译器在合适的时候进行内联和有关上下文的优化。代码中所有的分支都使用了与 GLIBC 相同的方法利用 `__builtin_expect()` 进行了优化。按照 Metropolis 算法，当且仅当新状态能量降低或随机数以波尔兹曼分布给定的概率落在一定区间内时，才接收状态的转移。在每个蒙特卡洛周期内，坐标二元组 (i, j) 和新的自旋值 v 都是随机生成的。

程序 potts/q2.c 分别选取正方形格点矩阵边长 $n = 10, 40, 80$，将系统的各个自旋值初始化为 $[0, q)$ 区间内的随机整数值（这与题目中的 $(0, q]$ 不同，仅为实现高效方便考虑，不影响计算结果）；而后对于温度下限 $T = T_{\text{L}}$，重复 10000000 个蒙特卡洛周期后，获得能量的平均值 E 和磁化强度 M 的平均值；接着令 $T := T + \Delta T$，（不重新初始化）再次重复 10000000 个蒙特卡洛周期得到相应平均值；重复这一过程，直到刚好超过温度上限：$T = T_{\text{H}} + \Delta T$。程序将扫描区间 $[T_{\text{L}}, T_{\text{H}}]$ 等分为 100 份，故取 $\Delta T = (T_{\text{H}} - T_{\text{L}})/100$。最后按照一阶向前差分方法得到一系列亥姆霍兹自由能的二阶导数：

$$C := \frac{\mathrm{d}E}{\mathrm{d}T} \;\Rightarrow\; C_i \approx \frac{E_{i+1} - E_i}{\Delta T},\ \left(\frac{\mathrm{d}M}{\mathrm{d}T}\right)_i \approx \frac{M_{i+1} - M_i}{\Delta T} \tag{10}$$

每次微微上调温度时不重新初始化系统，避免了在较低温度区间系统由 $M > 0$ 的状态跃迁到 $M < 0$ 的状态或反之造成二阶导数的假间断（注意小于临界温度时系统为自发磁化状态），在现实中这一过程在动力学上的确很难发生。

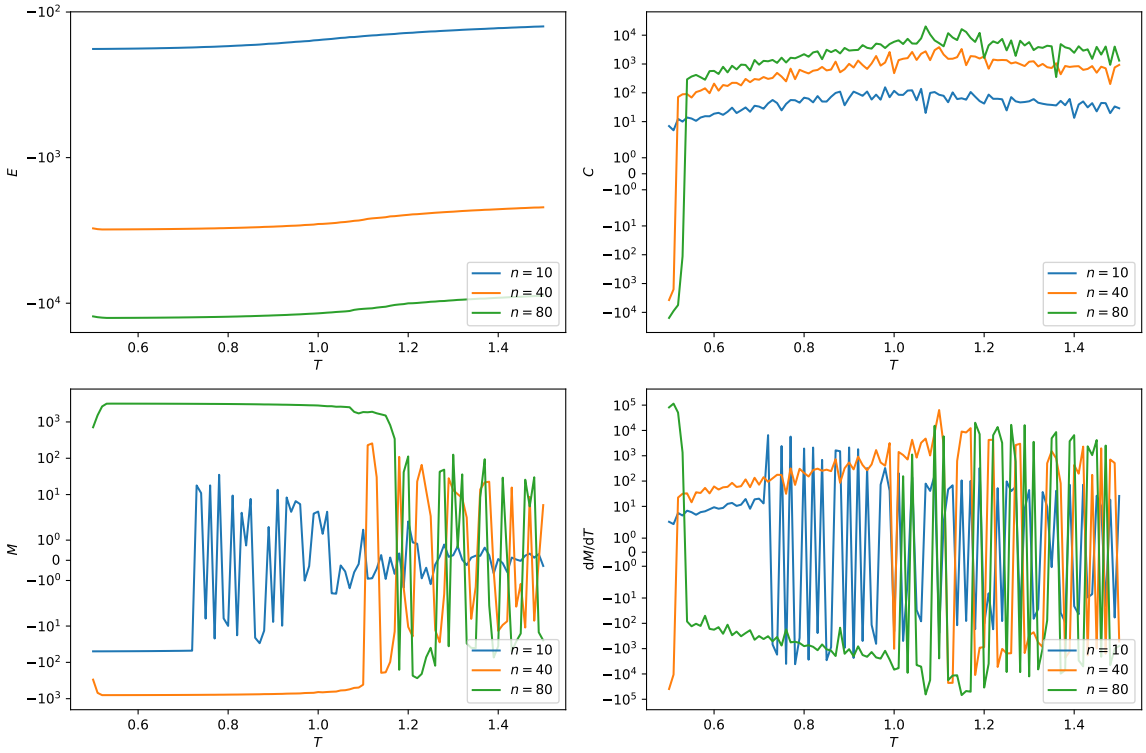

图 3: $q = 2$ 时不同大小的 Potts 格点方阵蒙特卡洛模拟结果（随机数种子 1428576）

E-T 图显示，系统在 $[0.5, 1.5]$ 的大部分子区间内，能量随温度的上升而增加，且连续性非常好。这表明如果有相变，必定不是一级相变。C-T 图显示，系统的热容在 $[0.5, 1.5]$ 的大部分子区间内为正，与 E-T 图中的能量总体上升趋势对应；热容随温度的增加表现出一定的抖动，多次实验表明对于不同的随机数种子，其抖动的模式差异很大，这表明这一抖动并非物理原因所致，而是蒙特卡洛算法造成的缺陷。我已尝试继续增大蒙特卡洛周期数，并未有效消除这一抖动。在这一抖动的背景下，热容可能的不连续性被隐藏了，从而无法直接判断是否存在二级相变。

M-T 图显示，系统在接近零温时表现出自发磁化，出现两个不同的磁化方向。如前所述，我设计的算法尽力地避免了系统在两种不同磁化方向之间转换，除非温度足够高使得系统的涨落足以克服这一转换过程的势垒。观察图像，不同 $n = 10$ 时的相变温度大约在 0.7 左右，$n = 40$ 时的相变温度大约在 1.1 左右，$n = 80$ 时的相变温度大约在 1.2 左右。从图中明显看出，虽然 M 表现出剧烈的振荡，其绝对值是大致连续的，符合二级相变的特征；但在二级相变发生处 M 的振荡导致 M 的导数连续性难以判断，结合 $\mathrm{d}M/\mathrm{d}T$-T 图来看，从上面的实验数据难以肯定这是一个二级相变。

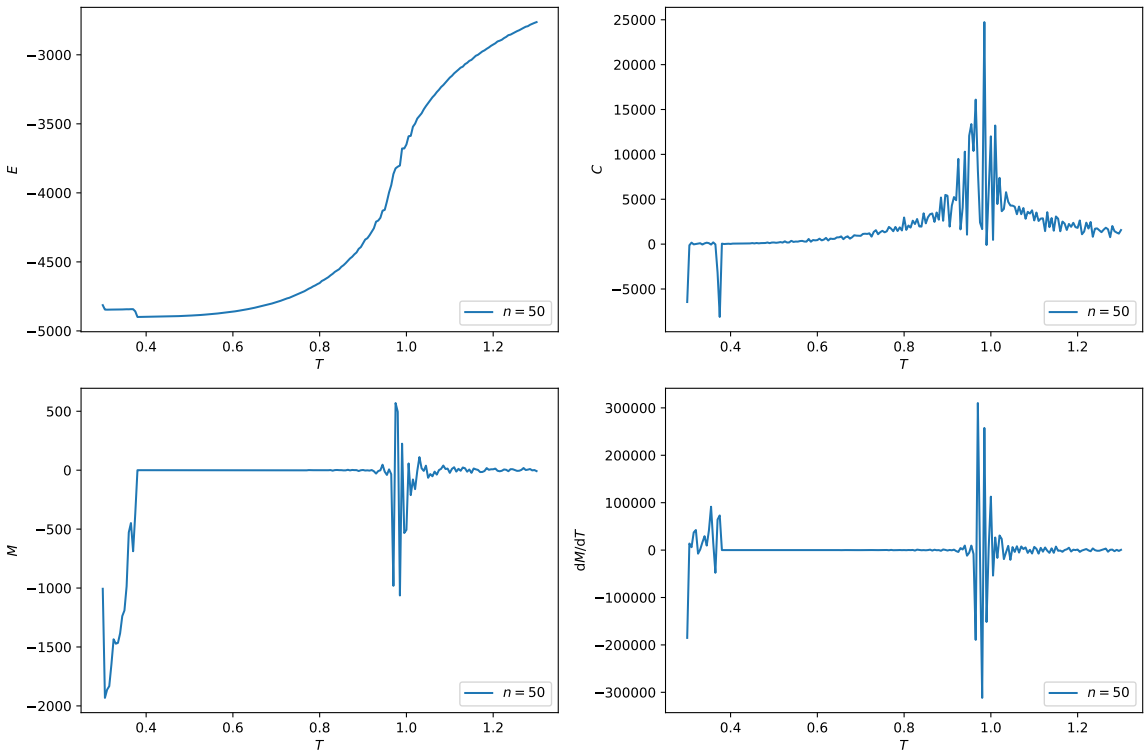

图 4: $q = 3$ 时 $n = 50$ 的 Potts 格点方阵蒙特卡洛模拟结果（随机数种子 1428576）

B. 与伊辛模型比较

二维伊辛模型描述了临界温度 $T_{\mathrm{C}} = 2.269$ 附近的二级相变，作参数变换 $J \to J/2$ 后临界温度变为 $T_{\mathrm{C}} = 1.134$，$n = 40$ 和 $n = 80$ 时的双自旋态 Potts 模型与之符合较好。$n = 10$ 时由于格点数目过少，不满足平均场近似所需的前提条件，求得的临界温度偏离较大。

C. 多自旋态蒙特卡洛模拟

由于我设计的算法不需要使用 ΔE-q 表，故无需具体考虑每个 q 值对应的 ΔE 范围。图 4，图 5 和图 6 分别给出了 $q = 3, 6, 10$ 时的 Potts 格点方阵蒙特卡洛模拟结果。经过实验，选取了共同的较优参数组合 $n = 50$ 或 $n = 200$，扫描范围大小 1，扫描区间数 200 和蒙特卡洛周期数 40000000，如此扫描的精度可以达到 0.005，可以获得带有两位有效数字的临界温度结果。n 增大时程序显著变慢，且必须相应增大蒙特卡洛周期数以尽量趋近于“各态历经”；扫描精度过小时蒙特卡洛涨落将严重影响求导精度；蒙特卡洛周期数增加时运行时间线性增长。考虑到这些参数之间复杂的关系，没有系统性的方法找到最优的参数配置。目前的参数配置已足以显示物理图像的典型特征和在一定的精度下求得临界温度。产生这些结果的程序为 potts/q3.c，potts/q6.c 和 potts/q10.c。

图 4 显示，$q = 3$ 时，能量的连续性提示这很有可能是一个二级相变。热容和磁化强度的温度导数在临界温度附近出现尖峰，提示这确实为一个二级相变。从原始数据读出，热容峰值位于 $T_{\mathrm{C}} = 0.98$ 处，磁化强度的温度导数峰值位于 0.97 处，两者非常接近。由于 E-T 图图像光滑性较好，选择 $T_{\mathrm{C}} = 0.98$ 作为实验结果。$q = 3$ 与 $q = 2$ 最大的不同在于，自发磁化在达到临界温度前很早的地方就消失了，但在临界温度附近磁化强度出现了振荡；其次，

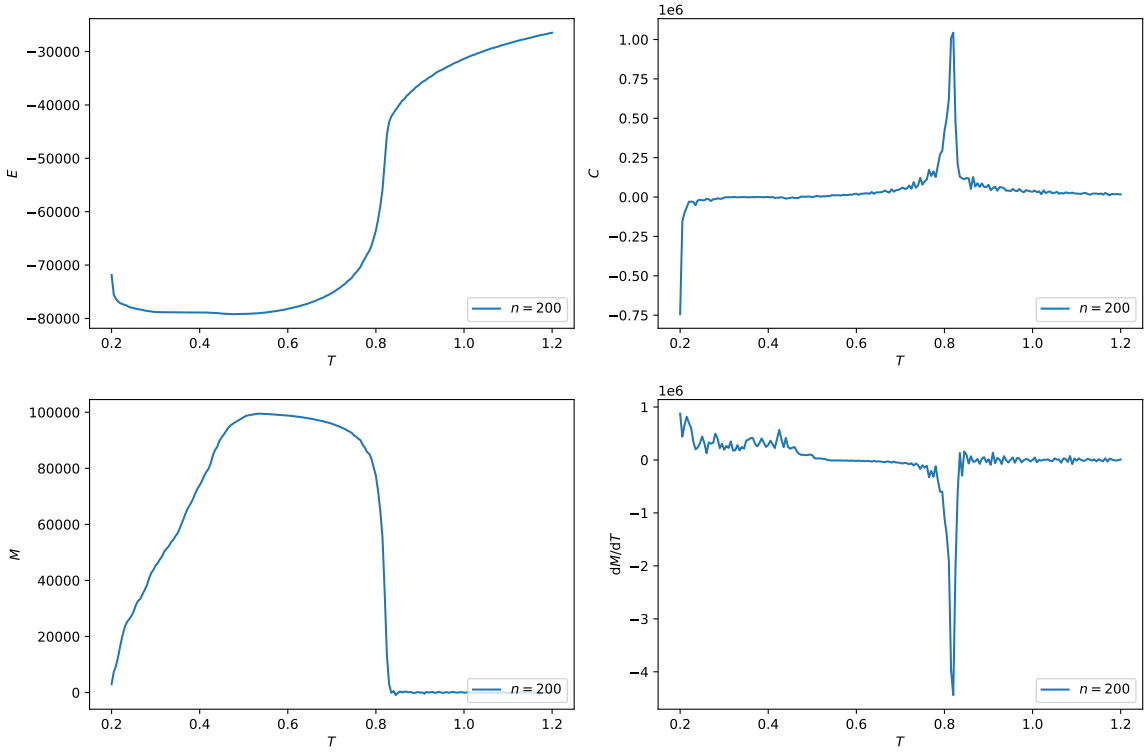

图 5: $q = 6$ 时 $n = 200$ 的 Potts 格点方阵蒙特卡洛模拟结果（随机数种子 1428576）

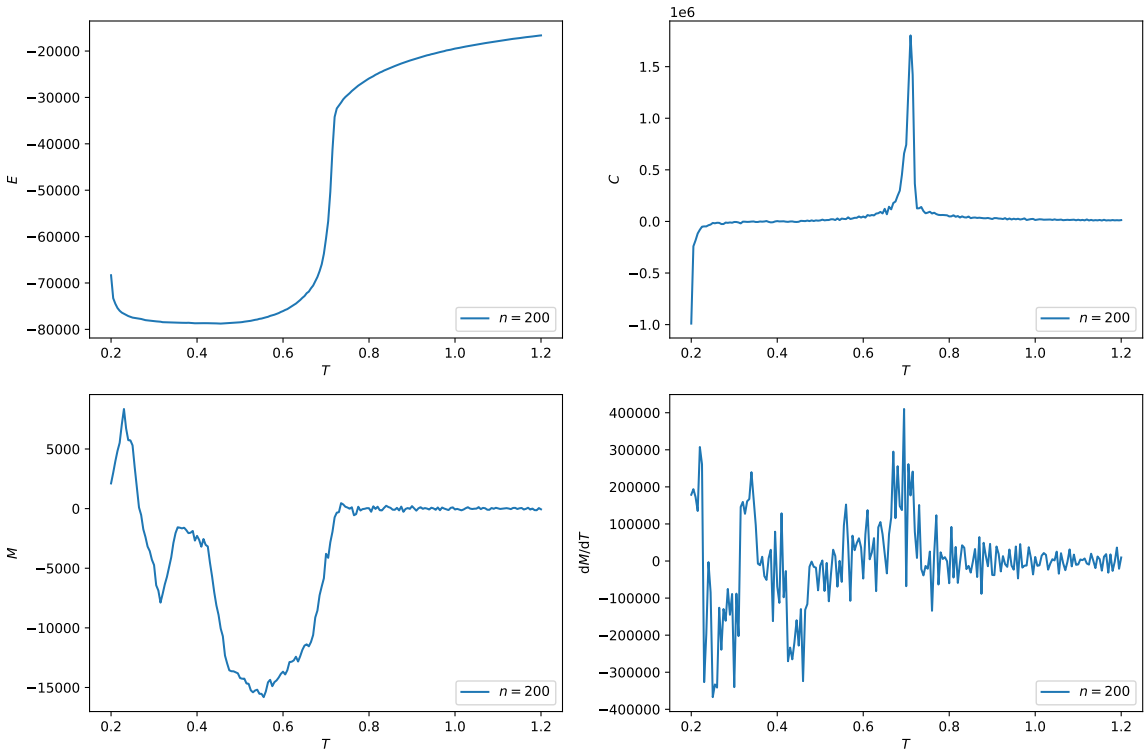

图 6: $q = 10$ 时 $n = 200$ 的 Potts 格点方阵蒙特卡洛模拟结果（随机数种子 1428576）

$q = 3$ 时热容在临界温度处显示了明显的尖峰，而 $q = 2$ 时临界温度附近没有看到热容曲线明显的奇异性。

图 5 和 6 显示，$q = 6$ 和 $q = 10$ 时临界温度附近发生的相变为明显的一级相变：在临界温度附近，能量有向上跃变，热容出现明显尖峰；磁化强度接近临界温度处迅速趋于零，其温度导数出现明显尖峰。结合图像从热容 C 的原始数据中读出，$q = 6$ 时 $T_{\mathrm{C}} = 0.82$，$q = 10$ 时 $T_{\mathrm{C}} = 0.71$。$q = 6$ 时，$\mathrm{d}M/\mathrm{d}T$ 尖峰的位置与 $T_{\mathrm{C}} = 0.82$ 完全重合；$q = 10$ 时 $\mathrm{d}M/\mathrm{d}T$ 的尖峰位置为 0.70，与 $T_{\mathrm{C}} = 0.71$ 非常接近。结合图 6 中 E-T 图图像质量更好的特征，应当选择 $T_{\mathrm{C}} = 0.71$ 作为 $q = 10$ 时的临界温度结果。

A 项目源代码

参见 `https://github.com/lyazj/numphy-06`。