# 计算物理 HW10 – Week8

PB18020735 杨世哲

2021 年 12 月 3 日

## 1 积分

代码：Integration.m

### 1.1 积分一

$$I = \int_{-\infty}^{+\infty} g(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

$p(\boldsymbol{x})$ 为随机向量 $(x_1, x_2, x_3, x_4)$ 的概率密度函数；

$$p(\boldsymbol{x}) = \frac{24}{\pi^2} \exp\left(-x_1^2 - 4x_2^2 - 9x_3^2 - 16x_4^2\right)$$

$$g(\boldsymbol{x}) = \frac{1}{96} x_1^2 \exp\left(x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4\right)$$

结果：

```
Warning: Unable to generate a stable distribution
> In StableMetropolis (line 25)
  In Integration (line 12)
sqrt(2).*1.*Sigma_1 = 0.992131
sqrt(2).*2.*Sigma_2 = 0.999561
sqrt(2).*3.*Sigma_3 = 1.015063
sqrt(2).*4.*Sigma_4 = 1.002172
I_0 = 6.372254e-03
```

$x_1 \sim x_4$ 的抽样结果：

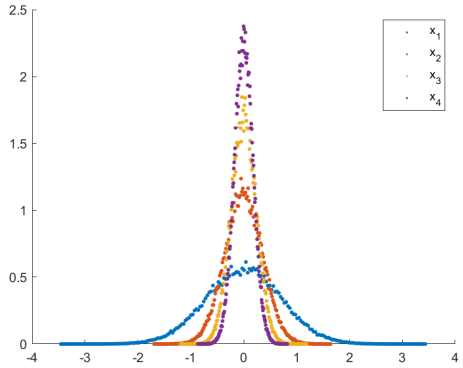



图 1: $x_1 \sim x_4$ 的分布

用 Mathematica 得到解析解:

Listing 1: 第一个积分的解析解

```
In[1]:=Integrate[1/(4 \[Pi]^2) a^2 Exp[-a^2 - 4 b^2 - 9 c^2 - 16 d^2 + a*
    b + a*c + a*d + b*c + b*d + c*d], {a, -Infinity, Infinity}, {b, -
    Infinity, Infinity}, {c, -Infinity, Infinity}, {d, -Infinity, Infinity
    }]
Out[1]=1516/(8001 Sqrt[889])
In[2]:=N[%, 20]
Out[2]=0.0063548316578727350288
```

可见相差约 1% 量级.

## 1.2 积分二

$$I = \int_{\Lambda/s}^{\Lambda} d\boldsymbol{k}_1 \int_{\Lambda/s}^{\Lambda} d\boldsymbol{k}_2 p(\boldsymbol{k}_1) p(\boldsymbol{k}_2) g(\boldsymbol{k}_1, \boldsymbol{k}_2)$$

采用球坐标:

$$p(\boldsymbol{k}) = f(k)\Theta(\theta)\Phi(\phi)$$

$$f(k) = \left[\Lambda - \mu \arctan\left(\frac{\Lambda}{\mu}\right) - \frac{\Lambda}{s} + \mu \arctan\left(\frac{\Lambda}{\mu s}\right)\right]^{-1} \frac{k^2}{k^2+\mu^2} \quad k \in \left[\frac{\Lambda}{s}, \Lambda\right]$$

$$\Theta(\theta) = \frac{1}{2}\sin\theta \quad \theta \in [0, \pi]$$

$$\Phi(\phi) = \frac{1}{2\pi} \quad \phi \in [0, 2\pi]$$

被积函数:

$$g(\boldsymbol{k}_1, \boldsymbol{k}_2) = \frac{1}{4\pi^4}\left[\Lambda - \mu \arctan\left(\frac{\Lambda}{\mu}\right) - \frac{\Lambda}{s} + \mu \arctan\left(\frac{\Lambda}{\mu s}\right)\right]^2 \frac{1}{(\boldsymbol{p} - \boldsymbol{k}_1 - \boldsymbol{k}_2)^2 + \mu^2}$$

取 $\boldsymbol{p} = (p, 0, 0)$，$p$ 为区间 $[0.68, 0.99]$ 的 9 等分点及区间端点，输出每一步迭代后的积分值，每个 $p$ 值、各 $p$ 值对应的最终积分结果以及相对误差估计：

Listing 2: 第二个积分的计算结果

```
IterTime = 1     p = 6.800000e-01     I = 1.457426e-04     RelError ~
   3.669566e-04
IterTime = 1     p = 7.144444e-01     I = 1.421287e-04     RelError ~
   1.759022e-03
IterTime = 1     p = 7.488889e-01     I = 1.383955e-04     RelError ~
   1.058097e-03
IterTime = 1     p = 7.833333e-01     I = 1.349446e-04     RelError ~
   7.948064e-04
IterTime = 1     p = 8.177778e-01     I = 1.312422e-04     RelError ~
   8.688226e-04
IterTime = 1     p = 8.522222e-01     I = 1.279494e-04     RelError ~
   6.884763e-04
IterTime = 1     p = 8.866667e-01     I = 1.243521e-04     RelError ~
   2.398737e-04
IterTime = 1     p = 9.211111e-01     I = 1.210773e-04     RelError ~
   3.254913e-04
IterTime = 1     p = 9.555556e-01     I = 1.177937e-04     RelError ~
   7.102131e-04
IterTime = 1     p = 9.900000e-01     I = 1.145585e-04     RelError ~
   3.385503e-04
```

可见抽样得到的分布稳定后，第一次迭代的结果即满足精度要求 (1%).

# 2 抽样

代码：sampling.m

## 2.1 抽样一

$$P(x) = 0.7e^{-10(x+10)^2} + 0.9e^{-x^4+3x^2}$$

利用 Mathematica 计算归一化系数：

```
In[1]:=Integrate[Exp[-x^4 + 3 x^2], {x, -Infinity, Infinity}]
```

```
Out[1]=1/2 Sqrt[3/2] E^(9/8) \[Pi] (BesselI[-(1/4), 9/8] + BesselI[1/4,
    9/8])
In[2]:=?BesselI
Out[2]=BesselI[n,z] gives the modified Bessel function of the first kind
    Subscript[I, n](z).
```

结果如下：

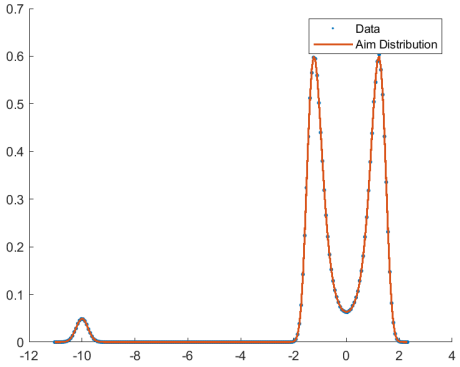


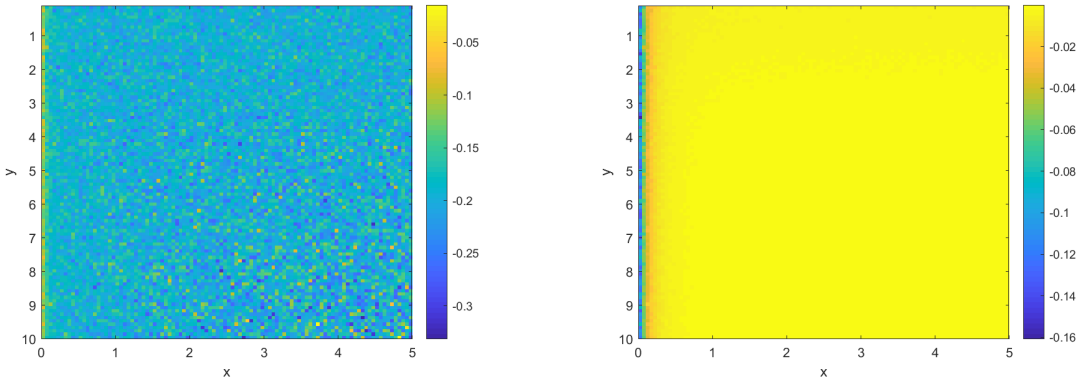

图 2: 分布一抽样结果

## 2.2 分布二

$$P(x,y) = A\left(\frac{1}{x^2+m^2}+\frac{1}{y^2}\right) \quad x\in[0,a], y\in[\varepsilon,b]$$

归一化系数：

$$A = \left[\frac{a}{\varepsilon} - \frac{a}{b} + \frac{b}{m}\arctan\left(\frac{a}{m}\right)\right]^{-1}$$




(a) 各点分布的相对误差　　(b) 各点误差与概率密度函数最大值的比

图 3: 分布二抽样结果

# 3 Ising Model

取相互作用强度 $J = 1$，玻尔兹曼常数 $k = 1$. 具体计算思路课上有讲，不再赘述.

## 3.1 2D Ising Model

取格点数为 $20 \times 20$，通过 Metropolis 方法产生 $10^5$ 个系统组成的系综，计算磁矩的系综平均值 $M$. 在 $T \in [0.001, 5]$ 之间取 5000 个 $T$ 值重复上述计算，作 $M - T$ 散点图：

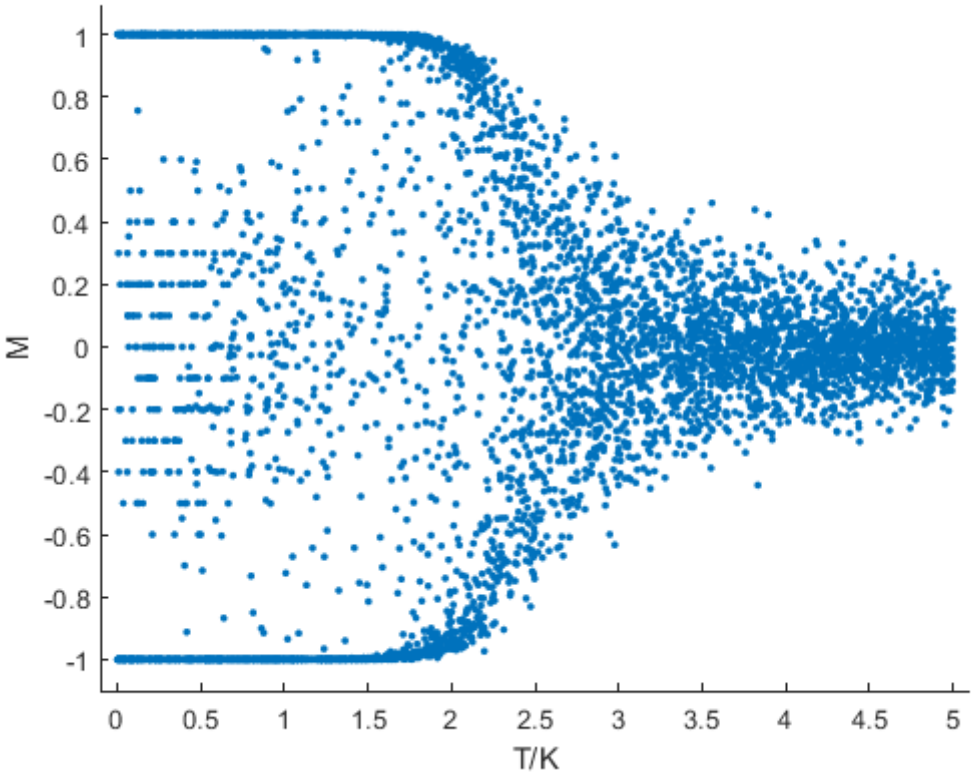



图 4: 不同温度下二维 Ising Model 的磁矩系综平均值

由图像可知，在 $2K \sim 3.5K$ 之间存在明显的相变点.

有序相区域的模拟结果并非全部落在两条曲线上，猜测是因为一些模拟的初值与有序态相差较大，导致在 $10^5$ 次翻转后仍然无法到达稳态. 另外，在低温区出现了几条平行于 $T$ 轴的散点列，猜测其原因为进入了“亚稳态”：即，自旋向上和向下的粒子各自聚集在一起而非相互混杂. 这时除非翻转发生在左右边界处，任何翻转后，系统能量增加值均会到达单次翻转的最大可能增加值. 设此增加值为 $\Delta E$，低温下接收这一翻转的概率为 $\exp(-\beta J \Delta E) \to 0$. 因此，处于边界的粒子数越少，系统发生翻转的概率越小. 系统为一正方形粒子阵，最短分界线为平行于一条边的线段. 由于系统边长为 20，这样的分界线意味着 $s = +1$ 的粒子比 $s = -1$ 的粒子多出（或不足）的数量为总粒子数的 $0, 0.1, 0.2, \cdots, 0.9, 1$ 倍. 此即系统能在 $M = 0, 0.1, 0.2, \cdots$ 处保持直线散点列的原因.

系综平均需要遍历相空间中所有可能的点，当 Metropolis 的采样次数不足时，模拟的结果不是对“遍历”的良好近似. 改系综内的系统数为 $10^4$，得到的模拟结果如下：

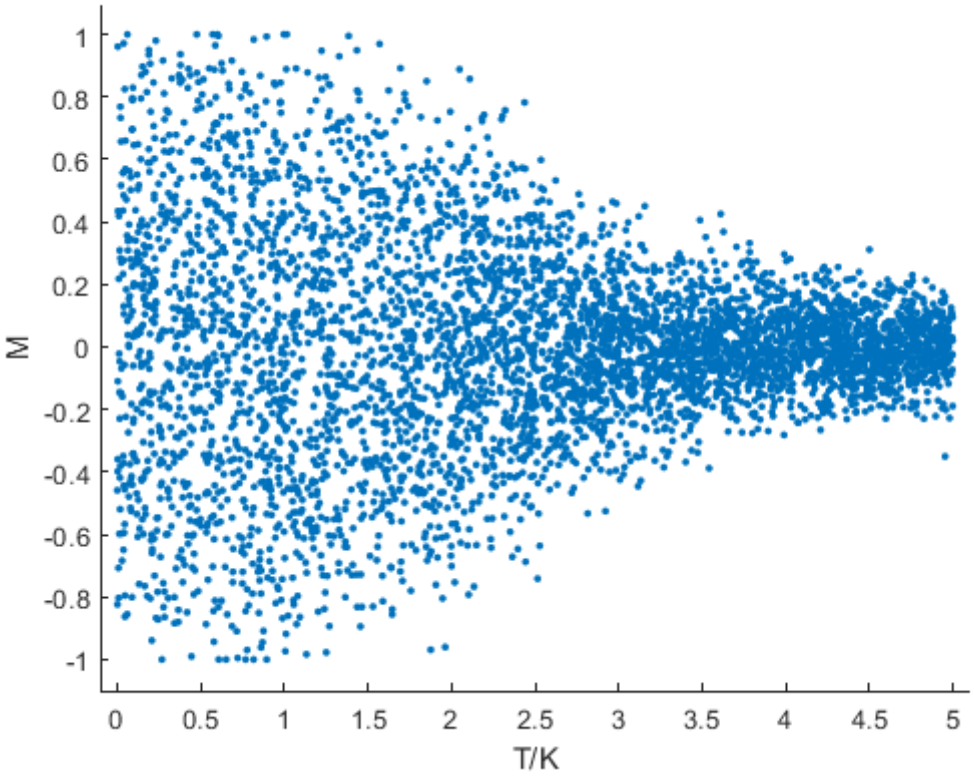



图 5: 系综大小为 $10^4$ 时，不同温度下二维 Ising Model 的磁矩模拟结果

可见 $10^4$ 大小的系综并不能得到良好的系综平均.

## 3.2 3D Ising Model

取点阵大小为 $10 \times 10 \times 10$，系综大小和温度采样点数不变，得到图像：

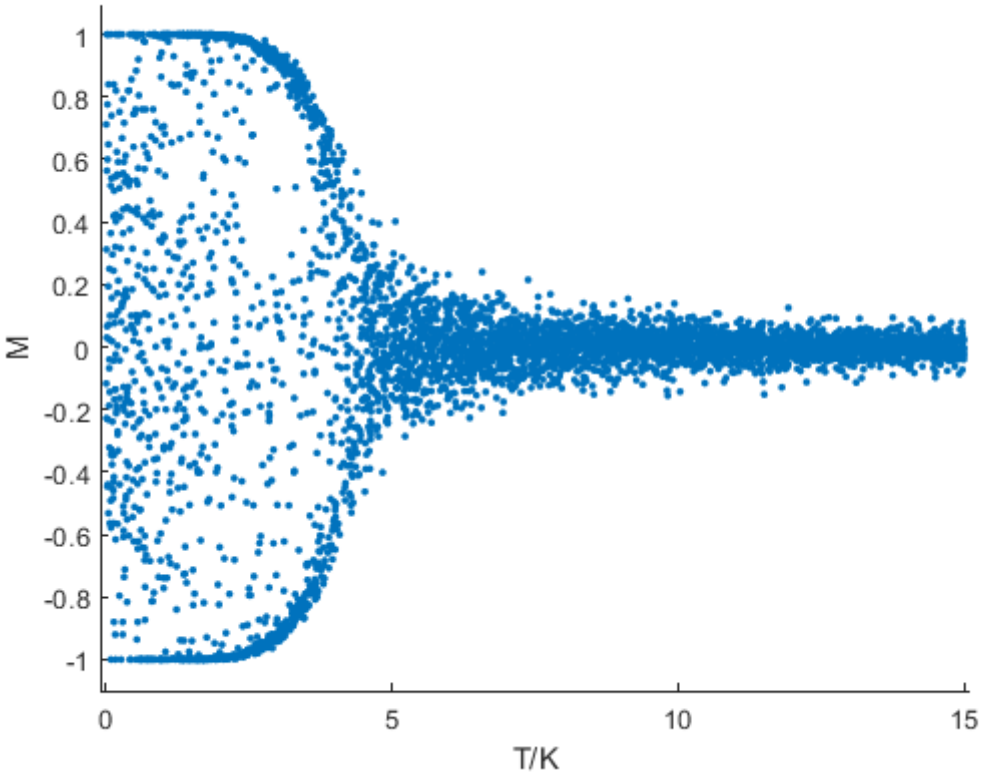



图 6: 不同温度下三维 Ising Model 的磁矩系综平均值

从图像读出相变点介于 $4K \sim 5K$.

# 4 相变

$$\left(p + \frac{a}{V^2}\right)(V - b) = RT$$

临界点满足（《热力学与统计物理》，汪志诚，第五版，P91）:

$$\left(\frac{\partial p}{\partial V}\right)_T = 0 \quad \left(\frac{\partial^2 p}{\partial V^2}\right) = 0$$

得到相变点:

$$p_c = \frac{1}{27}\frac{a}{b^2}$$
$$V_c = 3b$$
$$T_c = \frac{8}{27}\frac{a}{Rb}$$

现在令 $p = p_c + \Delta p$, $V = V_c + \Delta V$, $T = T_c + \Delta T$, 得到:

$$p_c + \Delta p \approx \frac{1}{27}\frac{a}{b^2} + \frac{R}{2b}\Delta T - \frac{R}{4b^2}\Delta T \Delta V + \left(-\frac{1}{36}\frac{a}{b^4} + \frac{1}{32}\frac{R\Delta T}{b^3}\right)(\Delta V)^2 + o(\Delta V)^3$$

因此:

$$\kappa = \kappa(V,T) = -\frac{1}{V}\left(\frac{\partial p}{\partial V}\right)_T \approx \frac{4b^2}{VR}\frac{1}{T - T_c}$$

$$\Delta V = \Delta V(p,T) =$$

令 $\Delta T = 0$ 得到:

$$\Delta V \approx 9b^2\sqrt{-\frac{p - p_c}{a}}$$

# 5 代码

## 5.1 Metropolis 抽样函数

Listing 3: 高维随机向量的 Metropolis 抽样函数–Metropolis.m

```
function x = Metropolis(N, dim, p, New, x0)
  %dim是随机向量的维数
  %N抽取样本点的个数
  %p为（未归一化的）概率密度函数的函数句柄
  %New为产生x_new的函数句柄
  %x0可选输入，当需要接续上一次Metropolis抽样时，x0为上一次抽样的末尾点
  x = zeros(N, dim);
  %第一个抽样值
  if nargin == 4                %不接续抽样
    x(1, :) = New(rand(1, dim));
  elseif nargin == 5            %接着上一个Markov链抽样
    x(1, :) = New(x0);
    xi = rand;
    if xi >= p(x(1, :))/p(x0)
      x(1, :) = x0;
    end
  end
  %Mertopolis抽样
  for k = 2:N
    x(k, :) = New(x(k-1, :));
    xi = rand;
```

```
      if xi >= p(x(k, :))/p(x(k-1, :))
        x(k, :) = x(k-1, :);
      end
    end
  end
```

## 5.2 计算样本分布的函数

Listing 4: 计算高维随机向量样本分布的函数–Distribution.m

```
function [x, fx] = Distribution(xi, N)
  %确定样本xi的取值范围，并划分为小区间，计算落在每个小区间内的xi的个数，
     归一化得到区间中心点处xi分布函数的近似值
  %xi为矩阵，其每一行为随机向量的一个抽样值
  %N为行向量，第k个元素表示计算第k维度分布时划分的子区间数
  dim = size(xi, 2);                    %随机向量的维数
  x0step = (max(xi) - min(xi))./N;      %每一维的小区间长度
  xstart = min(xi) - x0step./N;         %每一维的区间起始值
  xend = max(xi) + x0step./N;           %每一维的区间终止值
  y = cell(1, dim);                     %各个小区间的端点
  ShapeArray = cell(1, dim);            %调整向量维数的辅助数组
  for k = 1:dim
    ShapeArray{k} = 1;
  end
  x = cell(1, dim);                     %第k个元素表示第k维度各小区间中点
  ind = zeros(size(xi));                %用于计数的数组
  indcff = zeros(size(N, 2), 1);        %数组寻址的辅助数组
  %由于数组寻址需要，第一维需单独进行计算
  k = 1;
  y{k} = linspace(xstart(k), xend(k), N(k)+1);
  %调整区间第一维端点数组的维数
  if dim ~= 1
    ShapeArray{k} = N(k);
    x{k} = reshape((y{k}(2:end) + y{k}(1:end-1))./2, ShapeArray{:});
    ShapeArray{k} = 1;
  else
    x{k} = (y{k}(2:end) + y{k}(1:end-1)).'./2;
  end
```

```
    %确定各样本点在第一维度的哪个小区间内
    for j = 1:numel(y{k})
      ind(:, k) = (ind(:, k)) + (xi(:, k) > y{k}(j));
    end
    indcff(k) = 1;
    for k = 2:dim
      y{k} = linspace(xstart(k), xend(k), N(k)+1);
      %调整区间第k维端点数组的维数
      if dim ~= 1
        ShapeArray{k} = N(k);
        x{k} = reshape((y{k}(2:end) + y{k}(1:end-1))./2, ShapeArray{:});
        ShapeArray{k} = 1;
      else
        x{k} = (y{k}(2:end) + y{k}(1:end-1)).'./2;
      end
      %确定各样本点在第k维度的哪个小区间内
      for j = 1:numel(y{k})
        ind(:, k) = (ind(:, k)) + (xi(:, k) > y{k}(j));
      end
      ind(:, k) = ind(:, k) - 1;
      indcff(k) = prod(N(1:k-1).');
    end
    %计算分布
    N1 = prod(N.');                          %N中各元素相乘，得到总区间数
    fx = zeros(N1, 1);                       %存储各区间元素数目的数组
    %!!!注意：数组维数过高时，fx内存可能溢出
    ind1 = ind*indcff;                       %计算每个xi在哪个小区间内
    for k = 1:numel(ind1)
      fx(ind1(k)) = fx(ind1(k)) + 1;
    end
    %归一化
    x1step = (xend - xstart)./N;
    NormFac = sum(fx, 'all').*prod(x1step.');
    fx = fx./NormFac;
    if ~isscalar(N)
    fx = reshape(fx, N);
    end
  end
```

## 5.3 产生稳定的 Metropolis 抽样序列的函数

为了得到稳定分布，采用下列步骤判断样本分布 $(x, fx)$ 是否稳定：

1. 计算目标概率密度函数在 $x$ 处的值 $fx_{\text{check}}$
2. 若 $fx$ 与 $fx_{\text{check}}$ 相差不超过 $fx_{\text{check}}$ 最大值的 5%，则分布稳定

Listing 5: 稳定的 Metropolis 抽样函数–StableMetropolis.m

```
function xi = StableMertopolis(N, dim, p, New, Dist, RelTol)
  %产生稳定的随机向量xi，其每一行为一个随机向量值
  %N为样本数量
  %dim为随机向量维数
  %p为期望的概率密度函数句柄（可以不归一化）
  %New是产生x_new的随机函数句柄、
  %Dist为归一化的目标概率密度函数
  %RelTol为相对误差，可选输入
  DistPoints = 80.*ones(1, dim);     %计算样本分布时的采样点数
  if nargin == 5
    RelTol = 5e-2;                   %样本分布与解析分布的最大相对误差
  end
  MaxIterTime = 5;                   %最大迭代次数
  Nx = N;                            %每次迭代新增的抽样点数
  %-------------第一次抽样-------------%
  IterTime = 0;                              %迭代次数
  xi = Metropolis(Nx, dim, p, New);          %抽样
  [x, fx] = Distribution(xi, DistPoints); %计算样本分布
  fx_check = Dist(x{:});                     %计算标准分布函数在格点上的值
  fxmax = max(fx_check, [], 'all');          %标准分布函数在格点上的最大值
  sig = sum(abs(fx_check - fx) > RelTol.*fxmax, 'all');
  %-------------当抽样质量不达标，增加样本数量，重复操作-------------%
  while sig ~= 0
    IterTime = IterTime + 1;
    %到达最大迭代次数时报错并退出
    if IterTime > MaxIterTime
      warning('Unable to generate a stable distribution');
      break;
    end
    Nx = N.*(2.^IterTime);
    xi = [xi; Metropolis(Nx, dim, p, New, xi(end, :))];
    [x, fx] = Distribution(xi, DistPoints);
```

```
        fx_check = Dist(x{:});
        fxmax = max(fx_check, [], 'all');
        sig = sum(abs(fx_check - fx) > RelTol.*fxmax, 'all');
      end
    end
```

## 5.4 Ising Model

以下是计算任意维数 Ising Model 系综平均磁矩的函数：

Listing 6: IsingModel.m

```
%Ising Model
format long;
% clear;

%参数设置
N = {10, 10, 10};%每个维度的格点数
Nensemble = 1e5;%系综里系统的数目
% beta_J = 5;

%总粒子数
dim = numel(N);
Ndot = 1;
for k = 1:dim
  Ndot = Ndot.*N{k};
end
%自旋数组
% Sigma = ones(Ndot, dim) - 2.*(rand(Ndot, dim) <= 0.5);
Sigma = ones(Ndot, 1) - 2.*(rand(Ndot, 1) <= 0.5);
%系综数组，每一行存储单个系统自旋分量的平均值
Ensemble = zeros(Nensemble, 1);
Ensemble(1, :) = mean(Sigma, 1);
%系综能量数组，每一行存储单个系统的能量
EnsembleH = zeros(Nensemble, 1);
%Cofficient for array searching
indcff = ones(dim, 1);
for k = 2:dim
indcff(k) = N{k-1}.*indcff(k-1);
```

```
end

%用于计算系统能量的辅助数组
%单个系统能量 <- IndH <- IndRepmat + GenIndVec <- IndArray
GenIndVec = cell(dim, 2);
%生成索引矢量
IndArray = cell(1, dim);
for k = 1:dim
  IndArray{k} = 1;
end
for k = 1:dim
  IndArray{k} = N{k};
  GenIndVec{k, 1} = true(IndArray{:});
  GenIndVec{k, 2} = GenIndVec{k, 1};
  GenIndVec{k, 1}(end) = false;
  GenIndVec{k, 2}(1) = false;
  IndArray{k} = 1;
end
%将索引矢量重复排列，生成计算能量的索引数组
IndRepmat = N;
IndH = cell(dim, 2);
for k = 1:dim
IndRepmat{k} = 1;
IndH{k, 1} = reshape(repmat(GenIndVec{k, 1}, IndRepmat{:}), 1, Ndot);
IndH{k, 2} = reshape(repmat(GenIndVec{k, 2}, IndRepmat{:}), 1, Ndot);
IndRepmat{k} = N{k};
end

%初始化系统能量
for k = 1:dim
  EnsembleH(1) = EnsembleH(1) - sum(Sigma(IndH{k, 1}, :).*Sigma(IndH{k,
     2}, :), 'all');
end


%自旋翻转位置的坐标范围
MinInd = zeros(1, dim);
MinInd(1) = 1;
```

```
MaxInd = ones(1, dim);
MaxInd(1) = N{1};
for k = 2:dim
  MaxInd(k) = N{k} - 1;
end

%Metropolis方法生成系综
NewInd = zeros(1, dim);
for k = 2:Nensemble
  %选择要自旋翻转的粒子
  NewInd(1) = randi(N{1});
  for j = 2:dim
    NewInd(j) = randi(N{j}) - 1;
  end
  OneDNewInd = NewInd*indcff;
  Sigma(OneDNewInd) = - Sigma(OneDNewInd);
  %计算能量变化
  DeltaH = 0;
  VicNewInd = NewInd;
  for j = 1:dim
    if NewInd(j) ~= MinInd(j)
      VicNewInd(j) = NewInd(j) - 1;
      VicOneDNewInd = VicNewInd*indcff;
      DeltaH = DeltaH - Sigma(VicOneDNewInd).*Sigma(OneDNewInd);
      VicNewInd(j) = NewInd(j);
    end
    if NewInd(j) ~= MaxInd(j)
      VicNewInd(j) = NewInd(j) + 1;
      VicOneDNewInd = VicNewInd*indcff;
      DeltaH = DeltaH - Sigma(VicOneDNewInd).*Sigma(OneDNewInd);
      VicNewInd(j) = NewInd(j);
    end
  end
  DeltaH = DeltaH.*2;
  %判断是否接受翻转
  if rand < exp(-beta_J.*DeltaH)
    EnsembleH(k) = EnsembleH(k-1) + DeltaH;
  else
```

```
        Sigma(OneDNewInd, :) = - Sigma(OneDNewInd, :);
        EnsembleH(k) = EnsembleH(k-1);
    end
    %计算新的系统的能量
    Ensemble(k, :) = mean(Sigma, 1);
end

%计算系综平均
EnsembleH_calc = -beta_J.*EnsembleH;
EnsembleH_calc = EnsembleH_calc - max(EnsembleH_calc, [], 'all');%直接按
    系综平均公式计算会产生溢出
Z_calc = exp(EnsembleH_calc);
EnsAve = mean(Ensemble.*Z_calc, 'all')./mean(Z_calc, 'all');
fprintf('beta*J = %e     Sigma_ave = %e\n', beta_J, EnsAve);
```