

计算物理作业 9

谢昀城 22307110070

2024 年 12 月 16 日

1 题目 1

1.1 题目描述

The interior of a d -dimensional hypersphere of unit radius is defined by the condition . Write a program that finds the volume of a hypersphere using a Monte Carlo method. Test your program for $d = 2$ and $d = 3$ and then calculate the volume for $d = 4$ and $d = 5$, compare your results with the exact results.

1.2 程序描述

在本程序中，我们使用蒙特卡洛方法计算 d 维超球体的体积，并计算了 1-7 维的体积，并比较了其与标准值的相对误差。其中程序使用线性同余算法生成均匀分布的伪随机数。

本程序源文件为 sphere.py，在终端进入当前目录，使用命令 python -u sphere.py 运行本程序。运行时请保证 Python 第三方库 Numpy,Matplotlib 已安装。程序开发环境为 Python3.12.3，可在 Python3.8 以上版本中运行。

1.3 伪代码

伪随机数生成伪代码:

Algorithm 1 PRN

```
function PRN(n, dim, rmin, rmax, seed)
    INPUT: n (number of rows), dim (number of columns), rmin (minimum range), rmax (maximum range),
    seed (initial seed)

    OUTPUT: result (2D array of pseudo-random numbers)
    m  $\leftarrow 2^{31} - 1, a \leftarrow 48271, b \leftarrow 43, x \leftarrow seed
    result  $\leftarrow []$                                       $\triangleright$  Initialize empty list to store results
    for i  $\leftarrow 0$  To n · dim - 1 do
        x  $\leftarrow (a \cdot x + b) \% m
        Append rmin + (rmax - rmin) · x/m to result
    end for
    result  $\leftarrow \text{Reshape}(\text{result}, n, dim)$             $\triangleright$  Convert the list to a 2D array
    return result
end function$$ 
```

Algorithm 2 intMcN

```

function INTMCN(fn, r, point)
    INPUT: fn (function to integrate), r (function to rescale), point (sampling points)
    OUTPUT: result (Monte Carlo integral value)
    result  $\leftarrow$  Mean([fn(p)/r(p) for p in point]) ▷ Compute average over sampling points
    return result
end function

```

1.4 输入输出实例

对于本程序，运行后会生成图1和图2为”PRN” 和”VolumeDim.png”于当前目录下，分别为伪随机数采样和体积计算结果和相对误差，并在控制台打印计算结果。程序运行截图如图3所示。可以看到，随着维数增加，计算结果的偏差也增加，这是由于高维空间的球体积占总采样体积的比例更小，采样更困难。

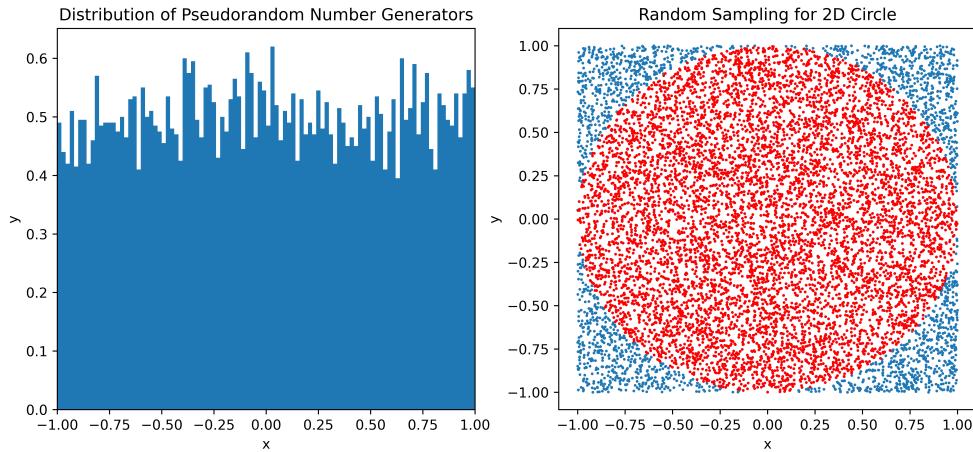


图 1: PRN 采样分布 (a)dimension=1(b)dimension=2

2 题目 2

Write a MC code for a 3D Face-Centered Cubic lattice using the Heisenberg spin model (adopt periodic boundary condition). Estimate the ferromagnetic Curie temperature.

$$H = -J \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j, \quad J = 1, \quad |\vec{S}_i| = 1$$

2.1 程序描述

本程序中，我们通过 metropolis 算法求解三维 fcc 晶格上的海森堡模型，主要求解功能在 Heisenberg 类中实现。程序将计算并可视化 $4 \times 4 \times 4$ 个初基元胞上， $\frac{k_B T}{J} = 1.0, 6.0$ 时的平衡构型。并在不同尺寸下 ($L \times L \times L, L = 2, 3, 4, 5$) 求解 $\langle E \rangle, \langle M \rangle, \langle C \rangle, \langle \chi \rangle$ 随温度变化的关系，以得到相变温度。

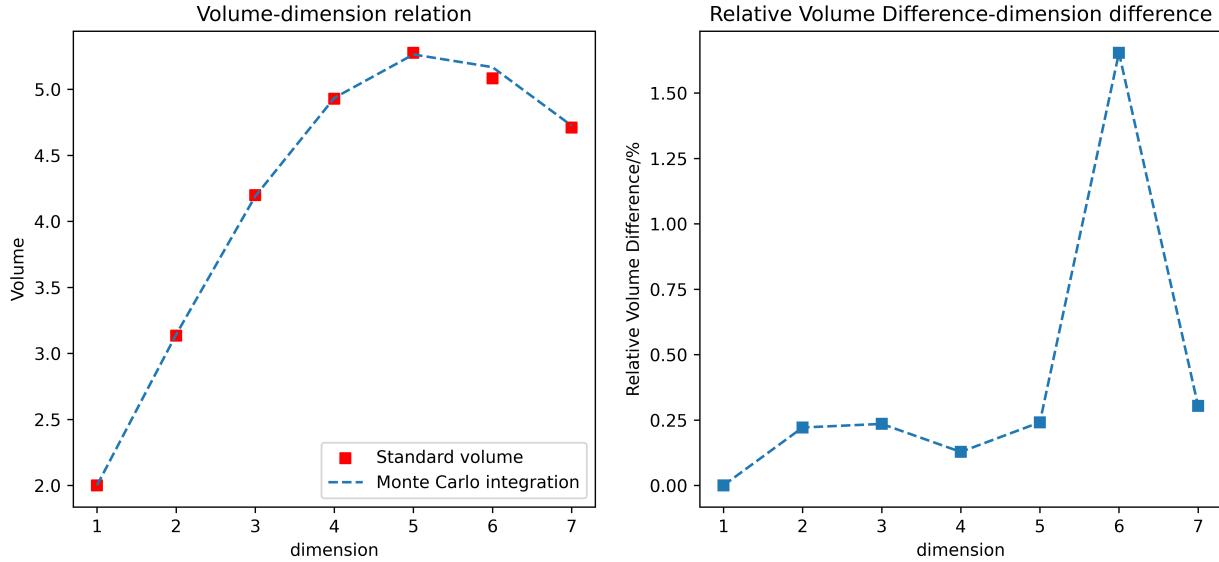


图 2: (a) 不同维度下球体积计算结果 (b) 与标准值相对误差

```
(base) PS C:\Users\ASUS\Desktop\计算物理基础\hw9> python -u .\sphere.py
Dimension:1,Volume:2.0,Standard Volume:1.9999999999999998,Sample Points:100000
Dimension:2,Volume:3.13464,Standard Volume:3.141592653589793,Sample Points:100000
Dimension:3,Volume:4.19864,Standard Volume:4.1887902047863905,Sample Points:100000
Dimension:4,Volume:4.92848,Standard Volume:4.934802200544679,Sample Points:100000
Dimension:5,Volume:5.27648,Standard Volume:5.263789013914325,Sample Points:100000
Dimension:6,Volume:5.08224,Standard Volume:5.167712780049969,Sample Points:100000
Dimension:7,Volume:4.7104,Standard Volume:4.7247659703314016,Sample Points:100000
```

图 3: 题目 1 程序运行截图

其中，热容 C 和磁化率 χ 将通过采样系统能量和磁矩的涨落得到：

$$C = \frac{\langle E^2 \rangle - \langle E \rangle^2}{Nk_B T^2}, \quad \chi = \frac{\langle M^2 \rangle - \langle M \rangle^2}{Nk_B T}$$

程序将首先使用 $1000L^3$ 个 MC 步进行热平衡，然后在平衡构型下运行 $2000L^3$ 个 MC 步，每 L^3 步采样体系的能量和平均磁矩。为了避免磁畴产生而难以达到平衡，系统将从 $\vec{S}_i = (1, 0, 0)$ 的基态构型开始。

由于系统尺寸 $L \rightarrow \infty$ 时， χ 将在 T_c 处发散，因此我们将取 χ 最大时的温度作为体系相变温度。

本程序将使用 Marsaglia 算法生成球面上的均匀分布。

本程序源文件为 Heisenberg.py，在终端进入当前目录，使用命令 `python -u Heisenberg.py` 运行本程序。运行时请保证 Python 第三方库 Numpy,Numba,Matplotlib 已安装。程序开发环境为 Python3.12.3，可在 Python3.8 以上版本中运行。

2.2 伪代码

Marsaglia 算法生成球面均匀分布

Algorithm 3 Marsaglia

function MARSAGLIA

INPUT: None

OUTPUT: *result* (array of three numbers generated using Marsaglia method)

while True **do**

```

v1 ← 2 · Random() − 1
v2 ← 2 · Random() − 1
s ← v12 + v22
s2 ←  $\sqrt{1 - s}$ 
if s < 1 then
    return Array([2 · v1 · s2, 2 · v2 · s2, 1 − 2 · s])
end if
end while
end function

```

求解 Heisenberg 模型

Algorithm 4 Heisenberg Class

Class: Heisenberg

Attributes:

L: Lattice size (integer)

T: Temperature (float)

J: Interaction constant (default = −1.0)

baiss: Array defining lattice basis

lattice: FCC lattice positions

spins: Spin vectors at lattice sites

neighbors: List of neighbor indices for each lattice site

Methods:

function INIT(*L*, *T*, *J* = −1.0)

self.L ← *L*, *self.T* ← *T*, *self.J* ← *J*

self.baiss ← Array([[0, 0, 0], [0, 0.5, 0.5], [0.5, 0, 0.5], [0.5, 0.5, 0]])

 GENERATEFCCLATTICE

 SETSPINS

 INITIALIZENEIGHBORS

end function

function GENERATEFCCLATTICE

lattice ← Empty List

for *i* ← 0 to *L* − 1 **do**

for *j* ← 0 to *L* − 1 **do**

for *k* ← 0 to *L* − 1 **do**

for *b* in *self.baiss* **do**

 Append [*i*, *j*, *k*] + *b* to *lattice*

end for

end for

end for

end for

```

self.lattice ← Array(lattice)
end function

function INITIALIZENEIGHBORS
neighbors ← Empty List
nbDistance ← Array([[0, 0.5, 0.5], [0.5, 0, 0.5], [0.5, 0.5, 0]]), tol ← 10-5
for i, ri in Enumerate(self.lattice) do
    neighbor ← Empty List
    for j, rj in Enumerate(self.lattice) do
        rij ← ri − rj
        APPLYPBC(rij)
        if i ≠ j and abs(rij) is close to any nbDistance within tol then
            Append j to neighbor
        end if
    end for
    Append neighbor to neighbors
end for
self.neighbors ← Array(neighbors)
end function

function APPLYPBC(rij)
for i ← 0 to 2 do
    if rij[i] > L/2 then
        rij[i] ← rij[i] − L
    else if rij[i] < −L/2 then
        rij[i] ← rij[i] + L
    end if
end for
end function

function CALCULATEENERGY
sn ← self.spins[self.neighbors]
return J · Sum(self.spins[:, None, :] · sn)/2/Length(self.lattice)
end function

function CALCULATEMAGNETIZATION
return Sum(self.spins, axis = 0)/Length(self.lattice)
end function

function CALCULATEDELTAE(i, spin1, spin2)
dS ← spin2 − spin1
dE ← J · Sum(dS · self.spins[self.neighbors[i]])
return dE
end function

function METROPOLISSTEP

```

```

index ← Random Integer in Range(Length(self.spins))
spin1 ← self.spins[index]
spin2 ← Marsaglia()
dE ← CALCULATEDELTAE(index, spin1, spin2)
if dE < 0 or Random() < exp(-dE/T) then
    self.spins[index] ← spin2
end if
end function

```

2.3 输入输出实例

对于本程序,运行后会生成图4,5,6,7为”Marsaglia.png”,”T=1.0.png”,”T=6.0.png”,”fingTc.png”于当前目录下,分别为球面随机向量的生成, $T=1.0, 6.0$ 时的平衡结构, 不同 $L, \langle E \rangle, \langle |M| \rangle, C, \chi$ 随 T 的变化。程序运行截图如图8所示。

可以看到,在低温时, 磁矩朝向几乎一致(图5), 高温时则呈现混乱分布(图6)。

随着温度的升高, $\langle |M| \rangle$ 在 T_c 附近下降到 0, $\langle E \rangle$ 随温度的增加速率减慢。而 C 和 χ 在 T_c 附近有明显的峰值, 且随着系统尺寸 L 的增加, 峰值宽度减小, 峰值增加, 表现出明显的有限尺寸效应(图7)。

最终得到相变温度约为:

$$T_c \approx 3.19 \frac{J}{k_B}$$

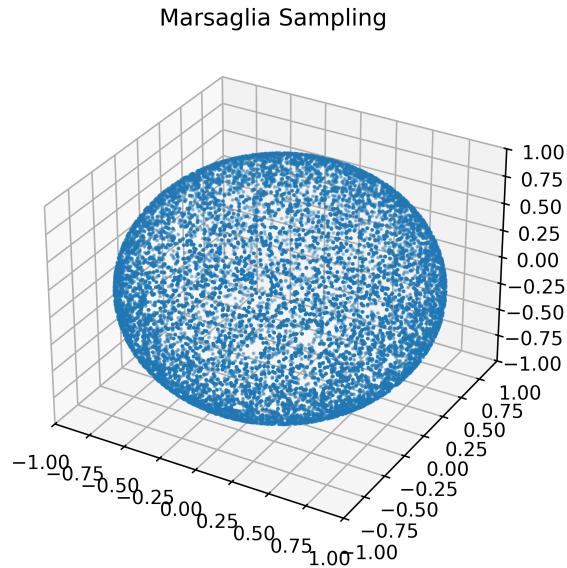


图 4: Marsaglia 算法生成的球面均匀分布

3D Visualization of Spins ($T=1.0, J=-1.0, L=4, N=256$)

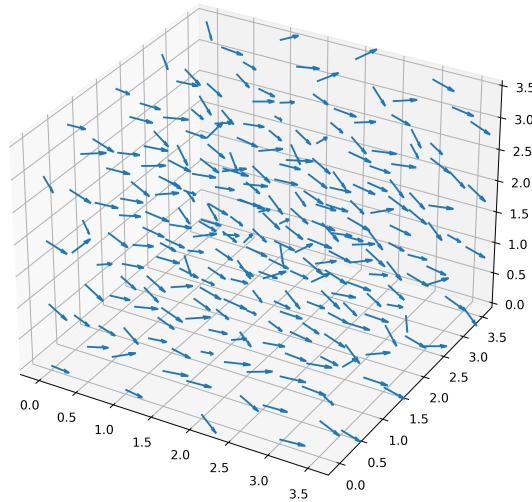


图 5: $L=4, T=1.0$ 时的平衡结构, 磁矩朝向较为有序

3D Visualization of Spins ($T=6.0, J=-1.0, L=4, N=256$)

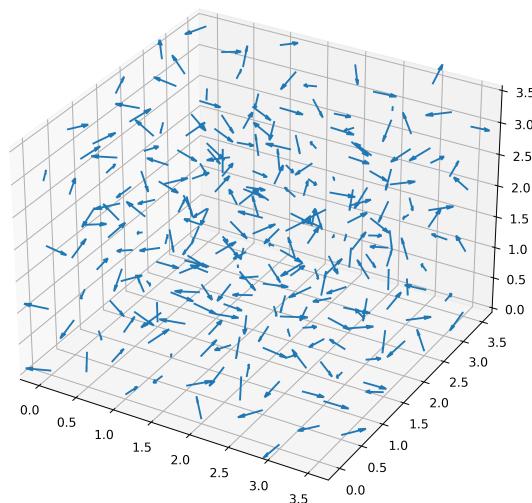


图 6: $L=4, T=6.0$ 时的平衡结构, 磁矩呈混乱分布

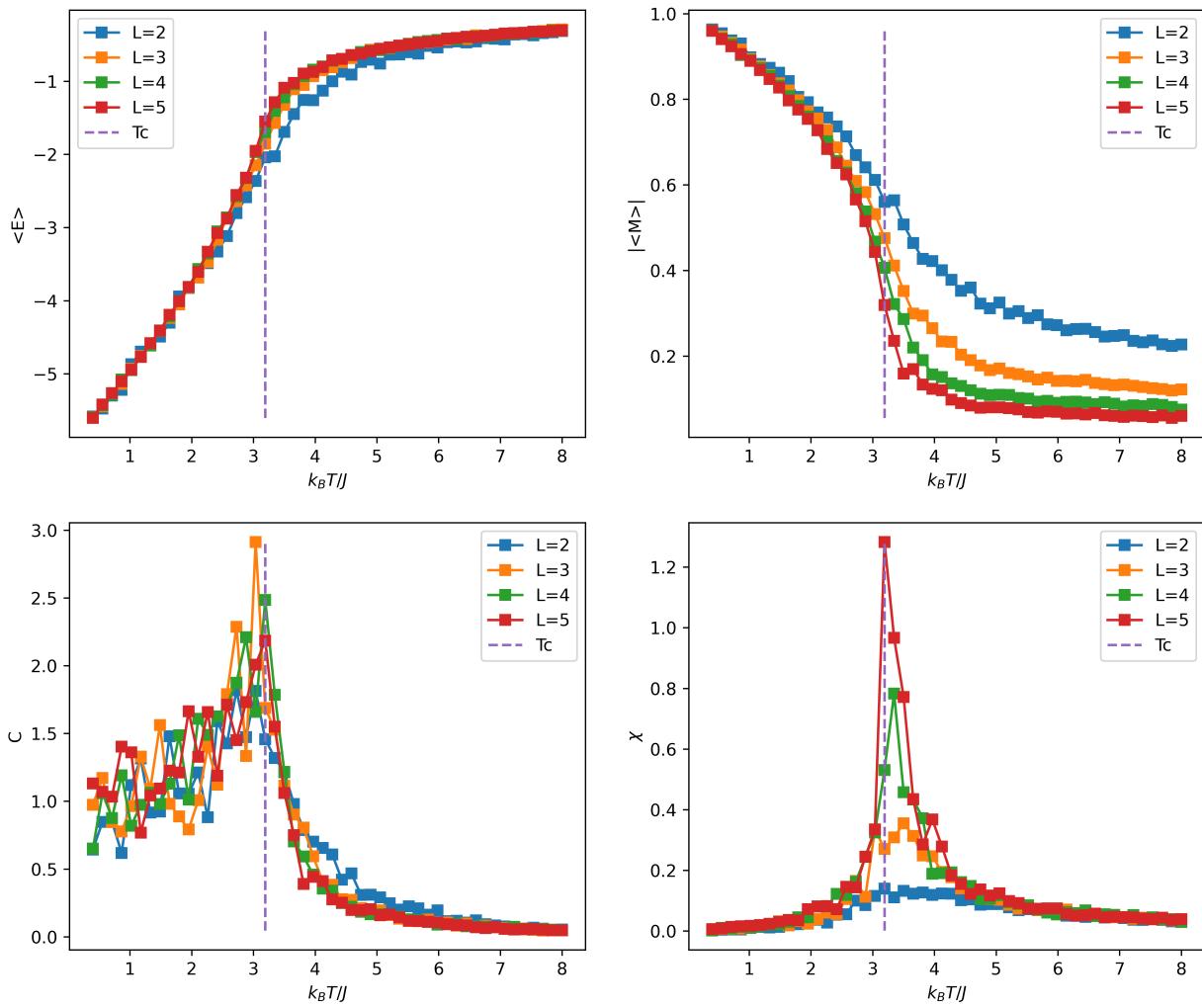


图 7: 不同 $L, \langle E \rangle, \langle |M| \rangle, C, \chi$ 随 T 的变化

```
calculate at T=4.4326530612244905,L=5,J=1.0,sites=500
calculate at T=4.587755102040816,L=5,J=1.0,sites=500
calculate at T=4.742857142857143,L=5,J=1.0,sites=500
calculate at T=4.8979591836734695,L=5,J=1.0,sites=500
calculate at T=5.053061224489796,L=5,J=1.0,sites=500
calculate at T=5.208163265306123,L=5,J=1.0,sites=500
calculate at T=5.363265306122449,L=5,J=1.0,sites=500
calculate at T=5.518367346938776,L=5,J=1.0,sites=500
calculate at T=5.673469387755103,L=5,J=1.0,sites=500
calculate at T=5.828571428571429,L=5,J=1.0,sites=500
calculate at T=5.983673469387756,L=5,J=1.0,sites=500
calculate at T=6.138775510204082,L=5,J=1.0,sites=500
calculate at T=6.293877551020408,L=5,J=1.0,sites=500
calculate at T=6.448979591836735,L=5,J=1.0,sites=500
calculate at T=6.604081632653061,L=5,J=1.0,sites=500
calculate at T=6.759183673469388,L=5,J=1.0,sites=500
calculate at T=6.914285714285715,L=5,J=1.0,sites=500
calculate at T=7.069387755102041,L=5,J=1.0,sites=500
calculate at T=7.224489795918368,L=5,J=1.0,sites=500
calculate at T=7.379591836734694,L=5,J=1.0,sites=500
calculate at T=7.534693877551021,L=5,J=1.0,sites=500
calculate at T=7.689795918367348,L=5,J=1.0,sites=500
calculate at T=7.844897959183674,L=5,J=1.0,sites=500
calculate at T=8.0,L=5,J=1.0,sites=500

#####
Tc=3.1918367346938776
#####
```

图 8: 题目 2 程序运行截图