计算物理作业十四

于浩然　　PB19020634　　2021.12.12

1　作业题目

设体系的能量为 $H(x,y)=-2(x^2+y^2)+\frac{1}{2}(x^4+y^4)+\frac{1}{2}(x-y)^4$, 取 $\beta=0.2,1,5$, 采用 Metropolis 抽样法计算 $\langle x^2\rangle,\langle y^2\rangle,\langle x^2+y^2\rangle$. 抽样时在二维平面上依次标出 Markov 链点分布，从而形象地理解 Markov 链.

2　算法简介

2.1　Metropolis 抽样规则

采用对称建议分布 T，非对称接受概率 A，由待满足的几率分布 p 的形式决定，即满足如下式：

$$W_{ij}=\begin{cases}T_{ij} & ,p_j>p_i\\ T_{ij}(p_j/p_i) & ,p_j<p_i\end{cases},\qquad A_{ij}=\min\{1,p_j/p_i\} \tag{1}$$

$$W_{ii}=1-\sum_{j\neq i}W_{ij} \tag{2}$$

2.2　抽样方法

具体抽样方法如下：设初始点坐标为 (x_0,y_0)，已经产生了 $\mathbf{x_1},\cdots,\mathbf{x_n}$ 这些点后，可在最后一个点附近构造一个试探解 $\mathbf{x_t}=\mathbf{x_n}+(\xi_x,\xi_y)\Delta x$，$\Delta x$ 是固定步长，$\xi_x,\xi_y\in(-1/2,1/2)$ 是均匀分布的随机数. 设体系满足 Boltzmann 分布：

$$p(x,y)\propto\exp\{-\beta H(x,y)\} \tag{3}$$

通过如下步骤得到 x_{n+1}:

(1) 产生 $[-1/2,1/2]$ 上均匀分布的随机数 ξ_x,ξ_y，令 $\mathbf{x_t}=\mathbf{x_n}+(\xi_x,\xi_y)\Delta x$;

(2) 计算 $\Delta E=\beta(H(\mathbf{x_t})-H(\mathbf{x_n}))$ 若 $\Delta E<0$ 则取 $\mathbf{x_{n+1}}=\mathbf{x_t}$;

(3) 若 $\Delta E>0$，则产生一个 $[0,1]$ 上均匀分布的随机数 ξ，若 $\xi<e^{-\Delta E}$ 则 $\mathbf{x_{n+1}}=\mathbf{x_t}$，反之则 $\mathbf{x_{n+1}}=\mathbf{x_t}$.

2.3　积分计算

设上述抽样共进行 N 步，可通过平均值法计算定积分：

$$\int_{-\infty}^{+\infty}f(x,y)\mathrm{d}x\mathrm{d}y=\frac{1}{N}\sum_{i=1}^{N}f(\mathbf{x_i}) \tag{4}$$

3 编程实现

使用 FORTRAN90 进行编程，

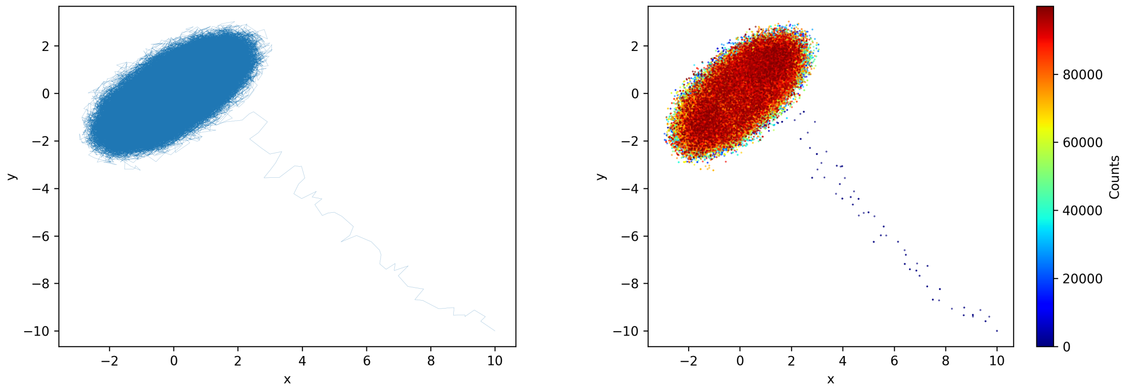
- `SUBROUTINE Sample`: 根据 Metropolis 抽样规则产生抽样点；
- `SUBROUTINE Integrate`: 根据已产生的抽样求三个积分的子程序.

由模块 `Metropolis` 包装，在主程序中分别实现 $\beta = 0.2, \beta = 1.0, \beta = 5.0$ 的情况，并用 python 绘图. 程序详见附件/附录.

4 计算结果

适当选取 Δx，并选取能量高的点 $(x_0 = 10, y_0 = -10)$ 开始模拟，将绘制的图像显示如下，左侧为将所有抽样点连线的折线图，右为根据抽样次序变换点颜色的散点图.

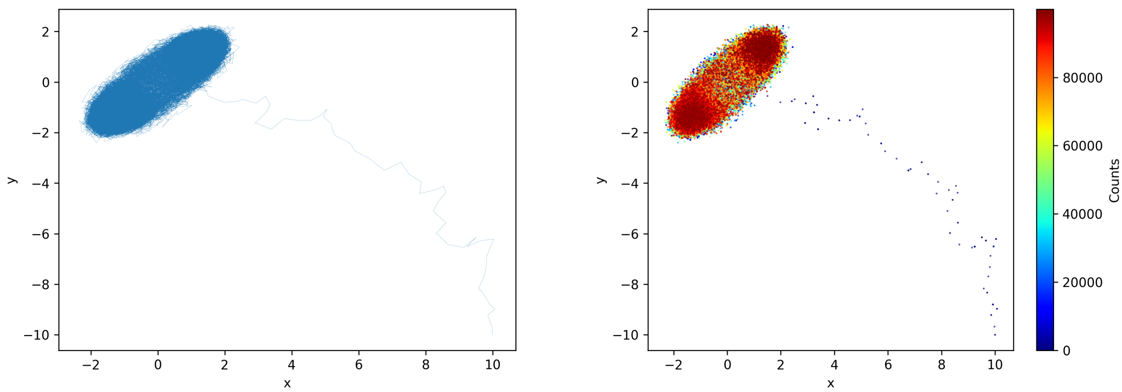
- $\beta = 0.2$



(a) 抽样点连线图 (b) 抽样点分布图

图 1: $\beta = 0.2$ 时 Markov 链点图 $(\Delta x = 1.0)$

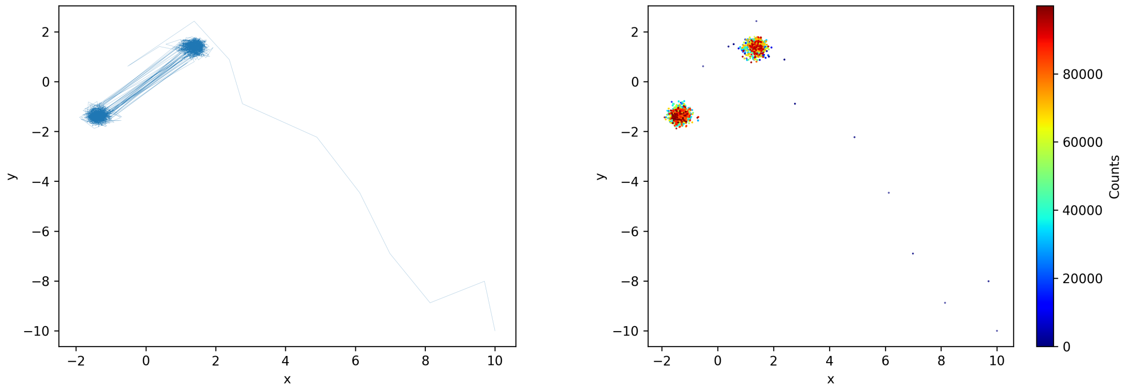
- $\beta = 1.0$



(a) 抽样点连线图 (b) 抽样点分布图

图 2: $\beta = 1.0$ 时 Markov 链点图 $(\Delta x = 1.0)$

- $\beta = 5.0$



(a) 抽样点连线图 (b) 抽样点分布图

图 3: $\beta = 5.0$ 时 Markov 链点图 ($\Delta x = 5.0$)

所得积分值如下表:

表 1: 积分值计算结果

β	$\langle x^2 \rangle$	$\langle y^2 \rangle$	$\langle x^2 + y^2 \rangle$
0.2	1.6242	1.5795	3.2038
1.0	1.7148	1.7283	3.4432
5.0	1.9650	1.9743	3.9394

5 结论

本题我们进行了 Metropolis 方法的模拟 (模拟退火法) 并绘制了 Markov 链, 对 Markov 链有了更深的理解.

6 源代码

FORTRAN90 源代码:

```fortran
MODULE Metropolis
IMPLICIT NONE
CONTAINS
    SUBROUTINE Sample(x0, y0, beta, num, step, filename)
        CHARACTER(LEN=*), INTENT(IN) :: filename
        REAL(KIND=8), INTENT(IN) :: beta, x0, y0, step
        REAL(KIND=8) :: rand(3 * num), xt(2), x(0:num, 2), d,
            seed
        INTEGER(KIND=4), INTENT(IN) :: num
        INTEGER(KIND=4) :: i
        x(0, 1) = x0
        x(0, 2) = y0 ! 初始化起步点
        CALL RANDOM_NUMBER(seed)
        ! 用FORTRAN自带的随机数生成器生成16807生成器的种子
        CALL Schrage(3 * num, int(2147483647 * seed), rand)
        DO i = 1, num
            xt(1) = x(i-1, 1) + step * (rand(i) - 0.5)
            xt(2) = x(i-1, 2) + step * (rand(2 * i) - 0.5)
            ! xt储存建议的一步，是否接受取决于随机数的判断
            d = beta * (H(xt(1), xt(2)) - H(x(i-1, 1), x(i-1, 2)
                )) ! 计算能量差d
            IF(d < 0) THEN
                x(i, :) = xt(:) ! 若能量减小则直接接收
            ELSE
                IF(rand(3 * i) < EXP(-d)) THEN
                    x(i, :) = xt(:)
                    ! 若能量增加，使用rand(3*i)与Bolzmann因子EXP
                        (-d)比较来进行判断
                ELSE
                    x(i, :) = x(i-1, :)
                    ! 若前面两次判断都为假，则抽样失败，点与上一
                        个点相同
                END IF
            END IF
        END DO
        OPEN (1, file=filename)
```

```
        WRITE (1, *) x
        CLOSE (1)
    END SUBROUTINE Sample

    SUBROUTINE Integrate(num, filename)
        CHARACTER(LEN=*) :: filename
        INTEGER(KIND=4) :: num
        INTEGER(KIND=4) :: i
        REAL(KIND=8), DIMENSION(0:num, 2) :: x
        REAL(KIND=8) :: i1, i2, i3
        OPEN (1, file=filename)
        READ (1, *) x
        CLOSE (1)
        i1 = 0
        i2 = 0
        i3 = 0
        DO i = 1, num
            i1 = real(i1 * (i-1)) / i + x(i, 1)**2 / i
            i2 = real(i2 * (i-1)) / i + x(i, 2)**2 / i
            i3 = real(i3 * (i-1)) / i + (x(i, 1)**2 + x(i, 2)
               **2) / i
            ! 按步更新平均值, 可防止求和溢出
        END DO
        print *, 'i1 = ', i1
        print *, 'i2 = ', i2
        print *, 'i3 = ', i3
    END SUBROUTINE Integrate

    REAL(KIND=8) FUNCTION H(x, y)
        REAL(KIND=8), INTENT(IN) :: x, y
        H = -2 * (x**2 + y**2) + 0.5 * (x**4 + y**4) + 0.5 * (x
          - y)**4
    END FUNCTION H
END MODULE Metropolis

SUBROUTINE Schrage(num, z0, rand)
    !Schrage随机数生成器子程序,将均匀随机数序列存放在数组rand中
    IMPLICIT NONE
    INTEGER(KIND=4) :: N = 1, num
```

```
    INTEGER :: m = 2147483647, a = 16807, q = 127773, r = 2836, In(num), z0
    REAL(KIND=8), INTENT(INOUT) :: rand(num)
    In(1) = z0 !将传入值z0作为种子
    rand(1) = REAL(In(1))/m
    DO N = 1, num - 1
        In(N + 1) = a * MOD(In(N), q) - r * INT(In(N) / q)
        IF (In(N + 1) < 0) THEN !若值小于零，按Schrage方法加m
            In(N + 1) = In(N + 1) + m
        END IF
        rand(N + 1) = REAL(In(N + 1))/m !得到第N+1个随机数
    END DO
END SUBROUTINE Schrage

PROGRAM MAIN
    USE Metropolis
    IMPLICIT NONE
    CALL Sample(10.0_8, -10.0_8, 0.2_8, 100000, 1.0_8, '0_2.dat')
    print *, 'beta = 0.2:'
    CALL Integrate(100000, '0_2.dat')
    CALL Sample(10.0_8, -10.0_8, 1.0_8, 100000, 1.0_8, '1_0.dat')
    print *, 'beta = 1.0:'
    CALL Integrate(100000, '1_0.dat')
    CALL Sample(10.0_8, -10.0_8, 5.0_8, 100000, 5.0_8, '5_0.dat')
    print *, 'beta = 5.0:'
    CALL Integrate(100000, '5_0.dat')
END PROGRAM MAIN
```

python 绘图脚本代码：

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import math
```

```python

plt.rcParams['savefig.dpi'] = 300
plt.rcParams['figure.dpi'] = 300

dat = np.loadtxt('0_2.dat')
x = dat[0:100000]
y = dat[100001:200001]
plt.xlabel('x')
plt.ylabel('y')
plt.plot(x, y, linewidth=0.1)
plt.savefig('0_2.eps')
plt.show()
plt.scatter(x, y, c=range(100000), cmap=mpl.cm.jet, s=0.1)
plt.colorbar(label="Counts", orientation='vertical')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('0_2_1.eps')
plt.show()

dat = np.loadtxt('1_0.dat')
x = dat[0:100000]
y = dat[100001:200001]
plt.xlabel('x')
plt.ylabel('y')
plt.plot(x, y, linewidth=0.1)
plt.savefig('1_0.eps')
plt.show()
plt.scatter(x, y, c=range(100000), cmap=mpl.cm.jet, s=0.1)
plt.colorbar(label="Counts", orientation='vertical')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('1_0_1.eps')
plt.show()

dat = np.loadtxt('5_0.dat')
x = dat[0:100000]
y = dat[100001:200001]
plt.xlabel('x')
plt.ylabel('y')
```

```python
plt.plot(x, y, linewidth=0.1)
plt.savefig('5_0.eps')
plt.show()
plt.scatter(x, y, c=range(100000), cmap=mpl.cm.jet, s=0.1)
plt.colorbar(label="Counts", orientation='vertical')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('5_0_1.eps')
plt.show()
```