计算物理 A 第十六题

杨旭鹏 PB17000234

2019 年秋季

目录

1 题目描述

设体系能量为 $H(x,y)=\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}$（以 kT 为单位），采用 Metropolis 抽样法计算 $\langle x^2\rangle,\langle y^2\rangle,\langle x^2+y^2\rangle$，并与解析结果进行比较。抽样时在 2 维平面上依次标出 Markov 链点分布，从而形象地理解 Markov 链。

2 理论推导

设体系满足 Boltzman 分布，则有：

$$p(x,y)=\frac{1}{Z}e^{-H(x,y)/kT} \tag{1}$$

其中 Z 为配分函数，即：

$$Z=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)}dxdy=2\pi\sigma_x\sigma_y \tag{2}$$

则可知有：

$$p(x,y)=\frac{e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)}}{2\pi\sigma_x\sigma_y} \tag{3}$$

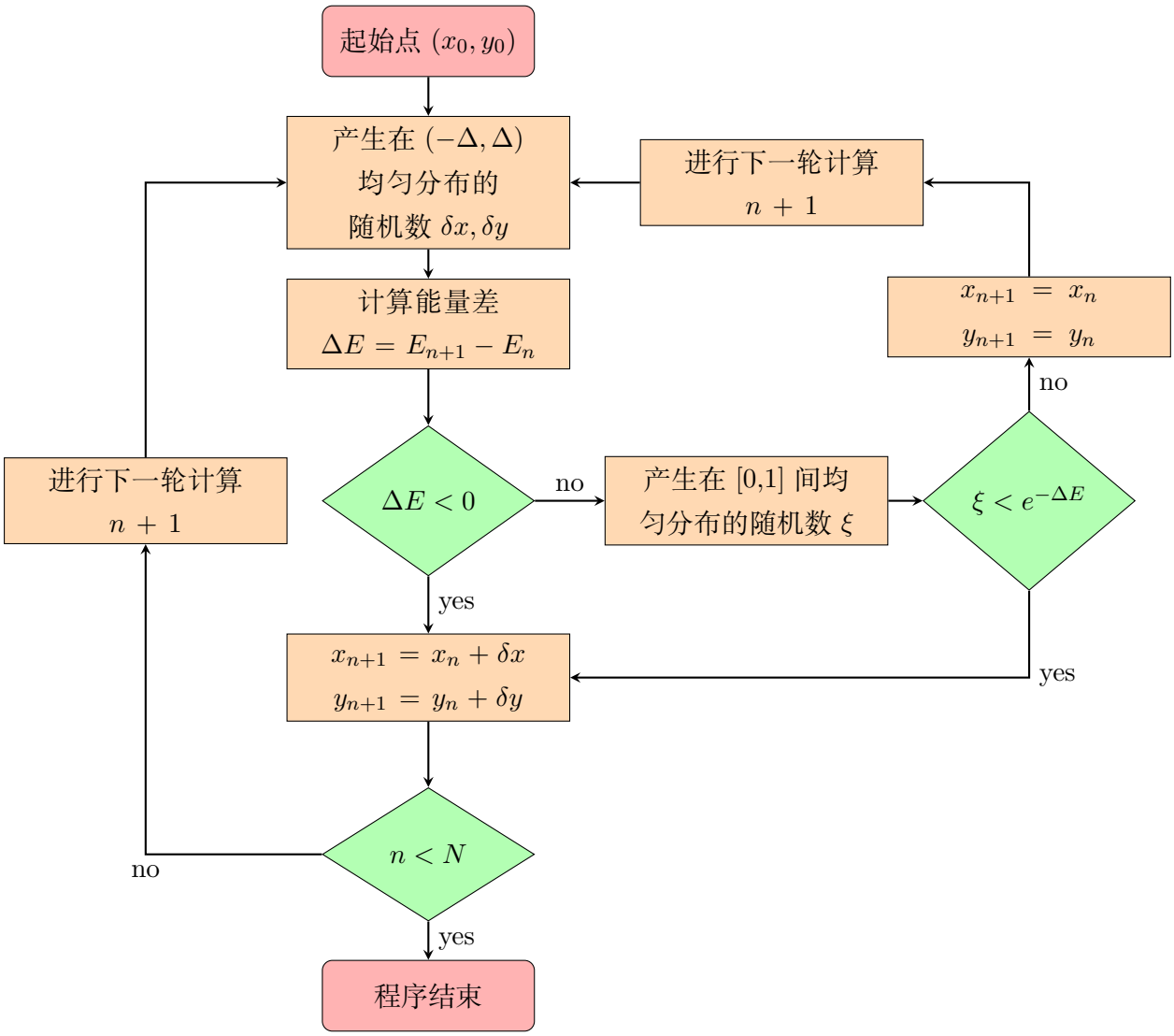
则有：

$$\left\langle x^2\right\rangle=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}x^2\frac{e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)}}{2\pi\sigma_x\sigma_y}dxdy=\sigma_x^2 \tag{4}$$

$$\left\langle y^2\right\rangle=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}y^2\frac{e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)}}{2\pi\sigma_x\sigma_y}dxdy=\sigma_y^2 \tag{5}$$

$$\left\langle x^2+y^2\right\rangle=\sigma_x^2+\sigma_y^2 \tag{6}$$

3 Metropolis 算法



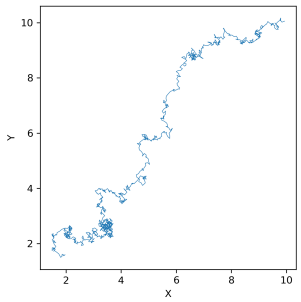
算法流程图如上所示。

4 程序使用方法

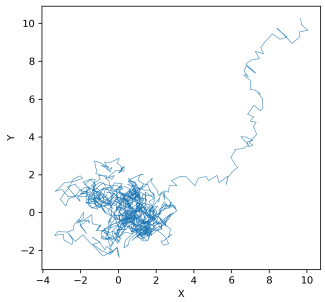
此程序设计为参数在程序代码中直接赋值形式，每次需在程序源码中进行参数调整，进而编译运行，以简化每次输入的过程（重复运行时不用在此输入）。需要调整的参数包括 main 函数中的 Markov 链总链节数 N，初始链节的坐标，体系能量表达式中的参数 σ_x, σ_y，每一步的步长参数 Δ，算统计量所需排除的初始链节个数。调整完参数后，编译运行，程序会自动输出 Markov 链节 (x, y) 坐标至不同文件并在屏幕上打印出统计量的计算结果。

5 程序结果与讨论

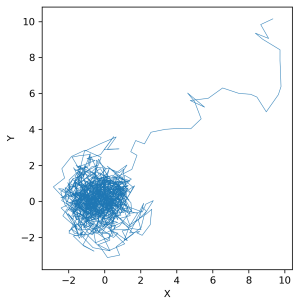
当设定起始点的坐标为 $(10, 10), \sigma_x = \sigma_y = 1$ 时，得到结果：



(a) $\Delta = 0.1$ 得到的 Markov 链

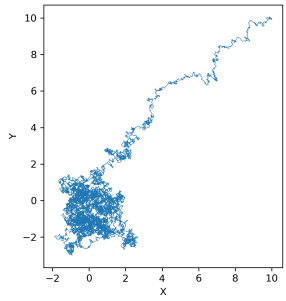


(b) $\Delta = 0.5$ 得到的 Markov 链

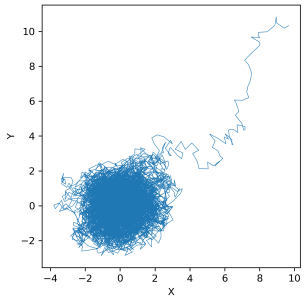


(c) $\Delta = 1$ 得到的 Markov 链

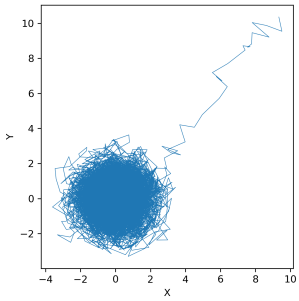
图 1: $N = 10^3$ 得到的 Markov 链



(a) $\Delta = 0.1$ 得到的 Markov 链



(b) $\Delta = 0.5$ 得到的 Markov 链



(c) $\Delta = 1$ 得到的 Markov 链

图 2: $N = 10^4$ 得到的 Markov 链

可以很明显的看出随着步长的增加，Markov 链节更加快速的收敛到概率值最大的地方附近，并在相同链节数的情况下，在概率值附近更加均匀。由于最后得到的是抽样得到 Markov 链节，而不是 Markov 链本身，故此时可认为步数大一点会使程序抽样效率更高。

为了计算统计量，我们将前 n 个链节数据删除，以排除初始位置对于统计量计算的影响。例如对于 $N = 10^6$ 的情况，我们删除前 n 个链节有可视化结果：

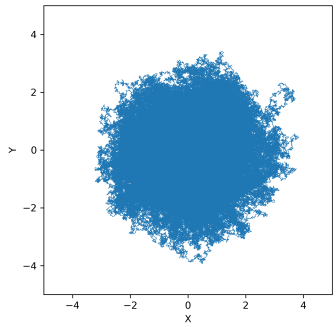
可以看出此时已基本消除了起始位置带来的影响。

经计算得到统计量的计算结果：

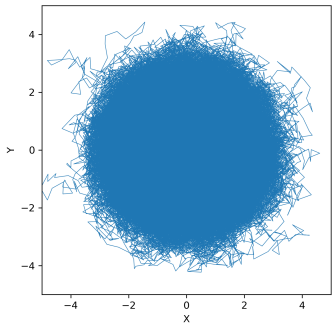
Δ	n	$\langle x^2 \rangle$	$\langle y^2 \rangle$	$\langle x^2 + y^2 \rangle$
0.05	5000	1.02657	0.84716	1.87372
0.1	2000	1.06450	0.93417	1.99867
0.5	2000	0.99612	1.01576	2.01188
1	1500	0.99551	0.99904	1.99456
5	1500	1.00770	1.00148	2.00918
50	1500	0.93853	0.93817	1.87671

表 1: 共产生 10^6 个链节，删除前 n 个链节得到的统计量计算结果一览表

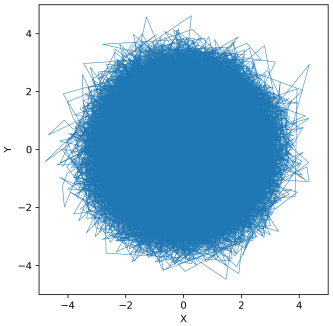
从表格中可以看出，步长比较合适时，得到的统计量值才与理论值较为接近。此点也很好理解，当步长比较小的时候，每步移动的距离有限，相同步数下很难形成比较均匀的图样，需很多的步数统计量才能逼近理论值；而步长比较大时，由于移动距离比较大的情



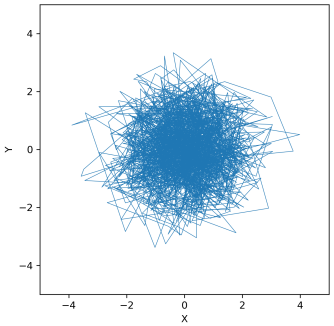
(a) $\Delta = 0.05, n = 5000$ 得到的 Markov 链



(b) $\Delta = 0.5, n = 2000$ 得到的 Markov 链



(c) $\Delta = 5, n = 1500$ 得到的 Markov 链



(d) $\Delta = 50, n = 1500$ 得到的 Markov 链

图 3: $N = 10^6$ 删去前 n 个链节得到的 Markov 链

况均会不予移动到新的地方，故重复链节很多，导致统计量计算误差。

综上，利用 Metropolis 方法进行抽样时，步长的选择是一门学问：既不能太大，（太大得到的分布不够好）也不能太小（太小需要更多的步数才能得到好的分布），需要根据具体情况具体定下步长的取值。

6 附录

A Metropois 算法抽样 C 语言源程序

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>


//写文件子程序，输入写成文件名称字符串str，数据来源于数组num，数据总数n
int my_filewriter(char str[],double num[],int n){
    FILE * fp;
    fp = fopen(str,"w+");

    for(int i=0;i<(n-1);i++)
    {
        fprintf(fp,"%lf,",num[i]);

    }
    fprintf(fp,"%lf",num[n-1]); //最后一个数据后不加 ","
    fclose(fp);
    return 0;
}



int main(int argc, const char * argv[]) {
    time_t t;
    srand((unsigned) time(&t));
    double x = 10; //起始点的(x,y)坐标
    double y = 10;
    double sigmax = 1; //sigmax,sigmay的值
    double sigmay = 1;
    double dE; //能量差
    double dx,dy; //坐标x,y某一步的该变量
    double delta = 50; //每一步的步长范围在[-delta,delat]之间
    double flag;
    int N = 1000000; //总步数
    int n = 1500;  //计算统计量删去起始点数的个数
    double avgx2 = 0; //计算产生数据点的<x^2>
    double avgy2 = 0; //计算产生数据点的<y^2>
    double avgsum = 0; //计算产生数据点的<x^2+y^2>

```

```
41    double *positionx = malloc(sizeof(double)*N); //存放markov链的节点坐标数组
42    double *positiony = malloc(sizeof(double)*N);
43
44
45    for(int i=0;i<N;){
46        dx = 2*delta*rand()/(double)RAND_MAX-delta;
47        dy = 2*delta*rand()/(double)RAND_MAX-delta;
48
49        dE = ( pow(x+dx,2)-pow(x,2) )/(2*sigmax) + ( pow(y+dy,2)-pow(y,2)
              )/(2*sigmay);
50        if(dE<0){
51            x += dx;
52            y += dy;
53            positionx[i] = x;
54            positiony[i] = y;
55            i++;
56        }
57        else{
58            flag = rand()/(double)RAND_MAX;
59            if(flag < exp(-dE) ){
60                x += dx;
61                y += dy;
62            }
63            positionx[i] = x;
64            positiony[i] = y;
65            i++;
66        }
67    }
68
69    my_filewriter("x.dat", positionx, N);
70    my_filewriter("y.dat", positiony, N);
71
72    if(n >= N){ //参数检查
73        printf("Parameters wrong!!\n");
74        return 0;
75    }
76
77    for(int i = n-1;i<N;i++){ //计算统计量
78        avgx2 += pow(positionx[i],2)/(double)(N-n);
79        avgy2 += pow(positiony[i],2)/(double)(N-n);
80    }
81    avgsum = avgx2 + avgy2;
82    printf("average of x2 is:%.5lf\n",avgx2);
83    printf("average of y2 is:%.5lf\n",avgy2);
84    printf("average of x2+y2 is:%.5lf\n",avgsum);
```

```
85
86
87
88     return 0;
89 }
```

B 可视化绘图及数据处理 Python 程序源码

```python
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams['savefig.dpi'] = 300 #图片像素
plt.rcParams['figure.dpi'] = 300 #分辨率
# 默认的像素：[6.0,4.0]，分辨率为100，图片尺寸为 600&400
fig1 = plt.figure()
fig2 = plt.figure()

ax1 = fig1.add_subplot(111)
ax2 = fig2.add_subplot(111)

X = []
Y = []
delta = 0.01 #步长
N = 6  #总链节数为10^N

with open('problem 16 Metropolis/'+str(delta)+'-10'+str(N)+'-x.dat', 'r') as f:
    while True:
        lines = f.readline() # 整行读取数据
        if not lines:
            break
        X = [float(i) for i in lines.split(',')] # 将整行数据分割处理
    X = np.array(X) # 将数据从list类型转换为array类型。


with open('problem 16 Metropolis/'+str(delta)+'-10'+str(N)+'-y.dat', 'r') as f:
    while True:
        lines = f.readline() # 整行读取数据
        if not lines:
            break
        Y = [float(i) for i in lines.split(',')] # 将整行数据分割处理
    Y = np.array(Y) # 将数据从list类型转换为array类型。


n = 2000

ax2.plot(np.delete(X, np.s_[0:n:1]), np.delete(Y, np.s_[0:n:1]), lw=0.5)
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_aspect('equal')
fig2.savefig(str(delta)+'-10'+str(N)+'-2.png')
```