经典蒙特卡罗应用

吴亦航（15346037）

考虑 D 维积分

$$I=\int_0^a \mathrm{d}x_1\int_0^a \mathrm{d}x_2\cdots\int_0^a \mathrm{d}x_D\, f\left(x_1,x_2,\cdots,x_D\right)$$

其中

$$f\left(x_1,x_2,\cdots,x_D\right)=\prod_{i=1}^{D}\left[\left(1+\frac{x_i}{2}\right)e^{-x_i}\right]$$

此积分的严格值为

$$I_{exact}=\left(\frac{3}{2}-\left(\frac{3}{2}+\frac{a}{2}\right)e^{-a}\right)^D$$

根据蒙特卡罗思想，可得

$$I_m=\frac{\left(1-e^{-a}\right)^D}{m}\sum_{l=1}^{m}\left[\prod_{i=1}^{D}\left(1+\frac{x_i^{(l)}}{2}\right)\right]\Bigg|_{x_l\text{服从}\mathrm{p}(x)}$$

相空间为 D 维的$\left[0,a\right]^D$连续空间。

算法如下：

(1) 给出一个初始位型 $\vec{x}_0$，本程序用随机函数生成

$$\left(x_1,x_2,\cdots,x_n\right),x_i\in\left[0,a\right]$$

(2) 选择更新方式，选取局域移动的方式

$$\vec{x}=\vec{x}^{(l)}+\Delta\vec{x}$$

$$\Delta x_i=\Delta\left(\eta_i-0.5\right)\times 2,\eta_i\in\left[0,1\right],i=1,\cdots,D$$

$$\Delta=const,0<\Delta\le a$$

若更新后的值超出$\left[0,a\right]^D$的范围，则有两种方式进行折回：反射折回（路径经过边域反射）和周期返回（路径超过边域，则去另一边）

(3) 计算试探更新构型与原构型的权重比例，判断是否接受更新

(4) 按照 N_{equi},N_0,m 三参数选取的原则从 Markov 链中取 x_k，计算 I_m。

(5) 独立做 L 次，求 $\overline{I}_{mL}$，估计 I_m 的涨落 σ_m

作 业 部 分

1　$D=5, a=2.0, \Delta=0.1a$，取 Markov 链[1, 10000]和[10000, 20000]之间的连续点作统计直方图，看是否服从 p(x)分布。

下面分别示出周期折回和反射折回的结果

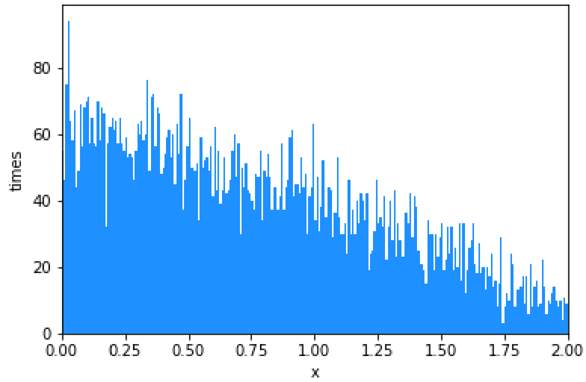

图 1.a 周期折回 Markov 链[1, 10000]

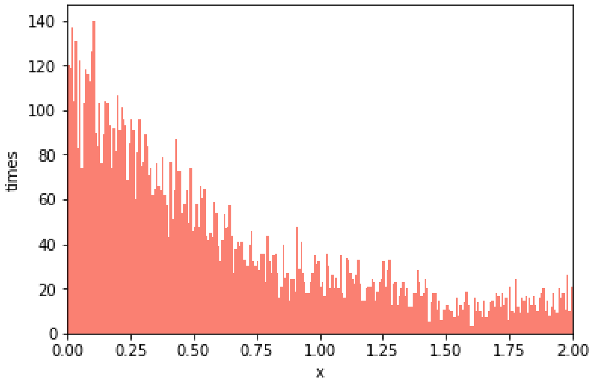

图 1.b 周期折回 Markov 链[10000, 20000]

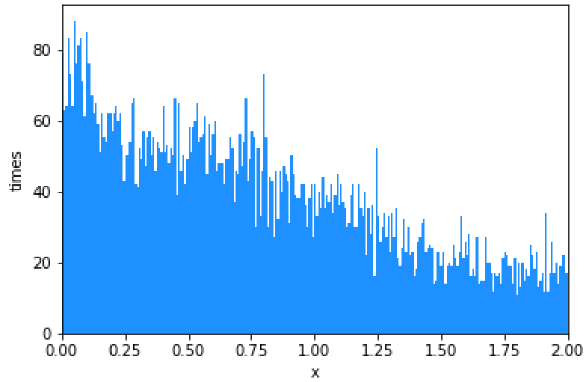

图 1.a 反射折回 Markov 链[1, 10000]

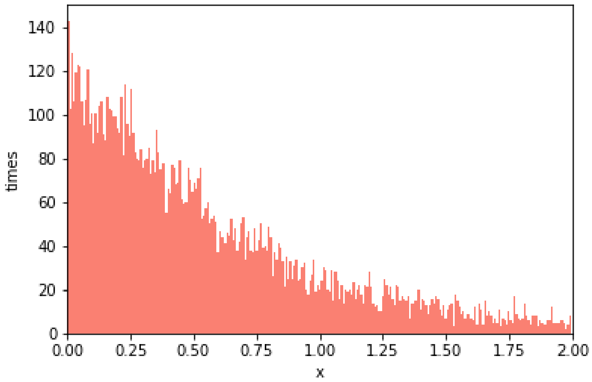

图 1.b 反射折回 Markov 链[10000, 20000]

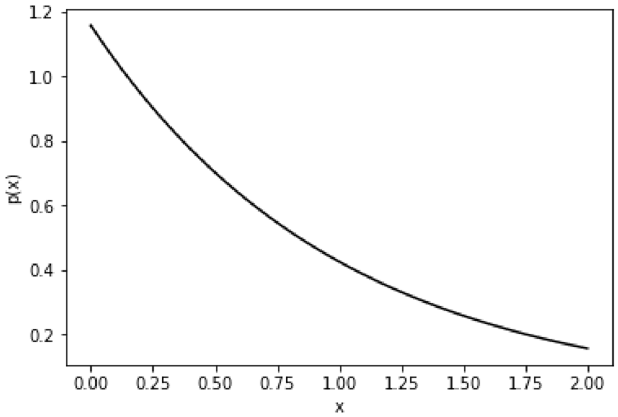

图 1.c $p(x)$的分布

我认为周期折回和反射折回的区别不大，但周期仿佛会更好，并且[10000, 20000]的数据会更趋近于 p(x)的分布。这是由于 10000 步后，会更加随机且稳定。下面的计算均采用周期折回。

2　$D=5, a=2.0, \Delta=0.1a$，$N_{equi}=10000, N_0=50, m=10000$，分别用 L=200，L=400 做出 I_m 的统计直方图。

这里（包括后面所有的计算）运用了不同的随机数种子。

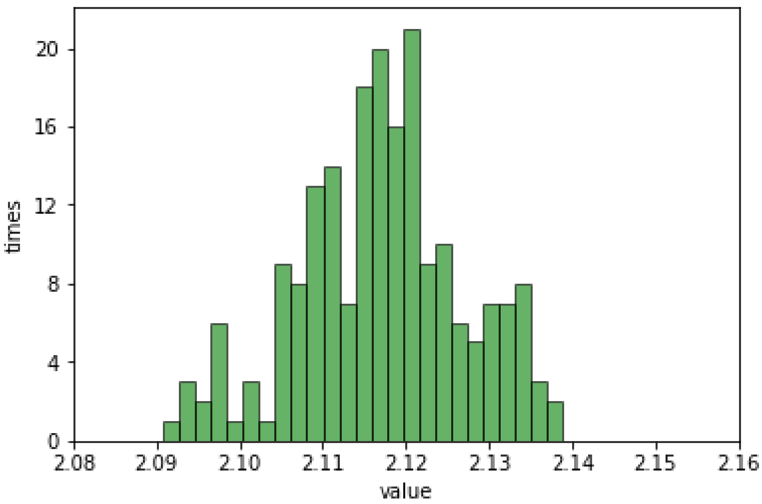

图 2.a L=200，统计直方图

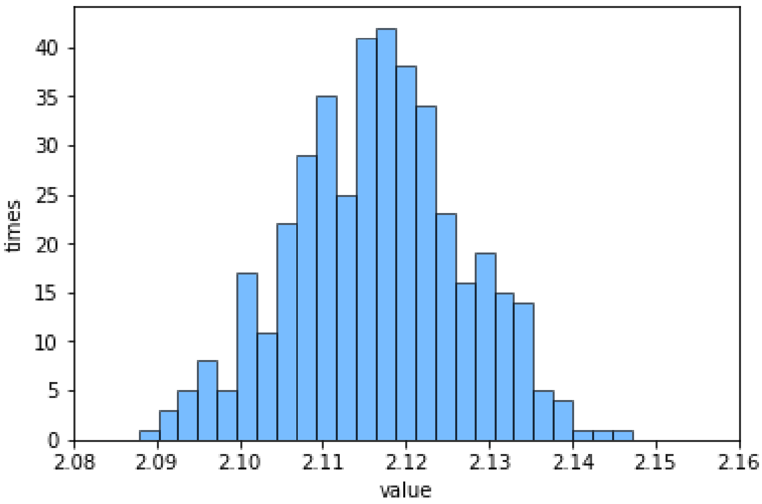

图 2.b L=400，统计直方图

L=200 组，平均值为 2.11696，标准差为 0.009964。

L=400 组，平均值为 2.11641，标准差为 0.010472。

认为这两组差别不大。

3　$a=2.0, \Delta=0.1a, N_{equi}=10000, N_0=50, L=200$，改变 m 的值，做 D=5 和 D=8 的 $\sigma_m \sim m$ 的关系曲线，验证蒙特卡罗标准差 $\sim 1/\sqrt{m}$ 的行为。给出蒙特卡罗积分结果，并比较计算误差 $E_r=\left|I_{mL}-I_{exact}\right|$ 与 $\sigma_m/\sqrt{L}$ 的关系。

实验结果如下：

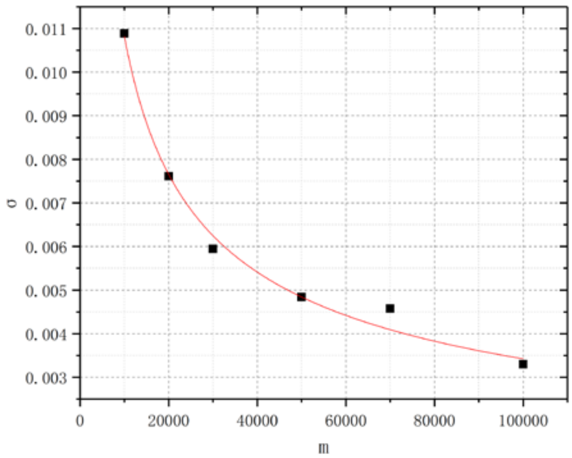

图 3.a D=5

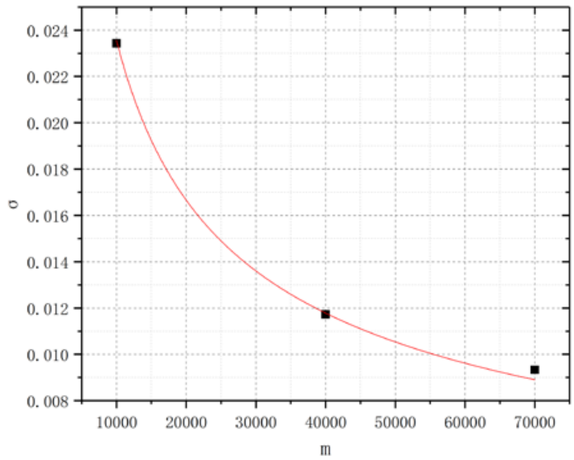

图 3.b D=8

根据图 3.a 和图 3.b，可以认为蒙特卡罗标准差 $\sim 1/\sqrt{m}$ 的行为。我们给出更为精确的结果：

图 3.a 的拟合曲线为 $\sigma_m=A/\sqrt{m}$，其中 $A=1.08\pm0.01$， $R^2=0.99052$。

图 3.b 的拟合曲线为 $\sigma_m=A/\sqrt{m}$，其中 $A=2.35\pm0.02$。

D	m	I_{mL}	σ_m	$E_r=\left\|I_{mL}-I_{exact}\right\|$	$\sigma_m/\sqrt{L}$
5	10000	2.115608	0.010886	0.000178	0.00077
	20000	2.114866	0.007614	0.000564	0.000538
	30000	2.116183	0.005944	0.000753	0.00042
	50000	2.115431	0.004842	7.5E-07	0.000342
	70000	2.115368	0.004576	6.2E-05	0.000324
	100000	2.115455	0.003296	2.49E-05	0.000233
8	10000	3.315941	0.023426	0.00024	0.001656
	40000	3.314518	0.011726	0.001662	0.000829
	70000	3.315709	0.009333	0.000471	0.00066

表 1　E_r 与 $\sigma_m/\sqrt{L}$ 的关系

4　考察 N_{equi} 的选取对结果的影响。$D=5, a=2.0, \Delta=0.1a, m=10000, N_0=50, L=200$，选取 N_{equi} 分别为 2000，4000，8000，16000，看 σ_m 的变化

N_{equi}	m	σ_m
200	2.11582	0.010531
2000	2.11468	0.010696
4000	2.11515	0.011834
8000	2.11613	0.011049
16000	2.11586	0.011161

表 2

认为 σ_m 随 N_{equi} 没有明显变化。

5　考察 N_0 的影响

6　考察Δ的影响。$D=5, a=2.0, N_{equi}=10000, m=10000, N_0=50, L=200$，计算接受率$\alpha$。

$\Delta(a)$	σ_m	α
0.05	0.01616	0.843883
0.1	0.01058	0.723201
0.2	0.00998	0.559003
0.4	0.01091	0.407480
0.6	0.00850	0.351765
0.8	0.00883	0.342627
1	0.00995	0.372850

表 3

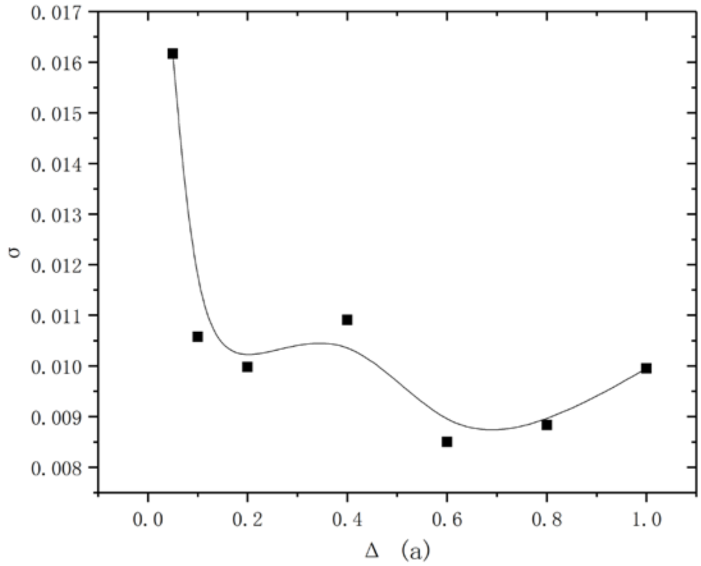

图 4.a　$\Delta \sim \sigma$

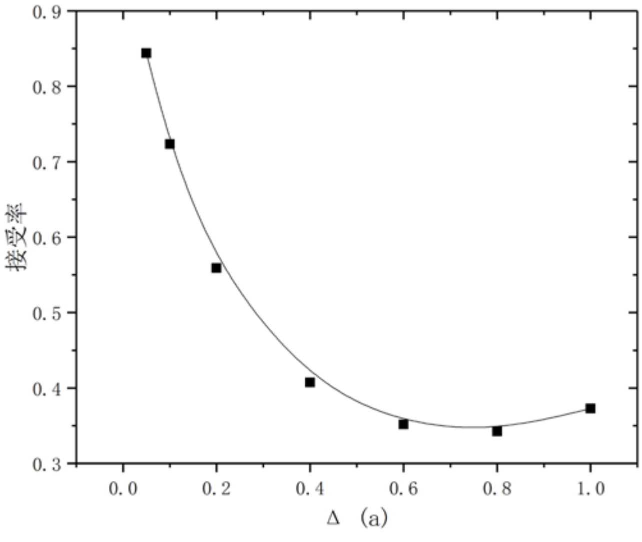

图 4.b　$\Delta \sim \alpha$

示例代码

```
double *next_point(double *x0, int D, double a, double delta, string method)
{
    //给出马尔可夫链的下一个点
    //x0:输入; D:维度; a:取值上界; delta:delta; method:"reflect" or "period";
    double fRand(double, double);
    int i;
    double *x_temp, *dx, e_sum;
    extern int accept;
    extern int total;

    x_temp = new double[D];
    dx = new double[D];

    for (i = 0; i < D; i++) {
        dx[i] = delta * (fRand(0, 1) - 0.5) * 2;
        x_temp[i] = x0[i] + dx[i];
    }

    //反射折回or周期折回
    if (method == "reflect") {
        for (i = 0; i < D; i++) {
            if (x_temp[i] > a) x_temp[i] = 2 * a - x_temp[i];
            else if (x_temp[i] < 0) x_temp[i] = -x_temp[i];
        }
    }
    else {
        for (i = 0; i < D; i++) {
            if (x_temp[i] > a) x_temp[i] -= a;
            else if (x_temp[i] < 0) x_temp[i] += a;
        }
    }

    //是否更新值
    total += 1;
    e_sum = 0;
    for (i = 0; i < D; i++) {
        e_sum += x_temp[i] - x0[i];
    }
    if (exp(-e_sum) >= fRand(0, 1)){
        accept += 1;
        for (i = 0; i < D; i++)
            x0[i] = x_temp[i];
```

```
    }

    delete x_temp, dx;
    x_temp = NULL;
    dx = NULL;
    return x0;
}


//游走,然后输出到屏幕和文件
    ofstream outfile("data1.csv", ios::out);
    //cout << setiosflags(ios::fixed) << setiosflags(ios::left) << setprecision(4);
    for (j = 0; j < d; j++) {
        //cout << setw(8) << x[j];
        if (j < d - 1) outfile << x[j] << ",";
        else outfile << x[j] << endl;
    }
    //cout << endl;
    for (i = 0; i < 20000; i++) {
        x = next_point(x, d, a, 0.1*a, "period");
        for (j = 0; j < d; j++) {
            //cout << setw(8) << x[j];
            if (j < d - 1) outfile << x[j] << ",";
            else outfile << x[j] << endl;
        }
        //cout << endl;
    }
    outfile.close();


int L = 160; //运行次数

    ofstream outfile("data6_0.05a.csv", ios::app);
    for (int k = L; k < L + 40; k++) {
        srand(k + 4600);

        //新建随机值x[D]
        x = new double[D];
        for (i = 0; i < D; i++)
            *(x + i) = fRand(0, a);

        int iter_sum = 0, iter = 0;
        double sum = 0, prod, result;
```

```
    while (iter_sum < 10000) {
        x = next_point(x, D, 2, 0.1, "period");
        iter += 1;
        if (iter < 10000) continue;
        if ((iter - 10000) % 50 == 0) {
            prod = 1;
            for (j = 0; j < D; j++) prod *= (1 + x[j] / 2);

            sum += prod;
            iter_sum += 1;
        }
    }
    result = pow((1 - exp(-2)), D)*sum / 10000;
    cout << "计算值(" << k << "): " << result << ',' << double(accept) / double(total) << endl;
    outfile << k << ',' << result << ',' << double(accept)/double(total) << endl;
    delete[] x;
    x = NULL;
}
outfile.close();
```