SUN YAT-SEN UNIVERSITY

《数据挖掘导论》

（**Exercises for Monte Carlo Methods**）

学 院 名 称 ： 数据科学与计算机学院

专业（班级） ： 16 软件工程电子政务

学 生 姓 名 ： 孙肖冉

学 号 ： 16340198

时 间 ： 2019 年 3 月 14 日

Exercise 1.

The Monte Carlo method can be used to generate an approximate value of pi. The figure below shows a unit square with a quarter of a circle inscribed. The area of the square is 1 and the area of the quarter circle is pi/4. Write a script to generate random points that are distributed uniformly in the unit square. The ratio between the number of points that fall inside the circle (red points) and the total number of points thrown (red and green points) gives an approximation to the value of pi/4. This process is a Monte Carlo simulation approximating pi. Let N be the total number of points thrown. When N=50, 100, 200, 300, 500, 1000, 5000, what are the estimated pi values, respectively? For each N, repeat the throwing process 100 times, and report the mean and variance. Record the means and the corresponding variances in a table.

蒙特卡洛方法可以用于产生接近pi的近似值。图1显示了一个带有1/4内切圆在内的边长为1的正方形。正方形的面积是1，该1/4圆的面积为pi/4。通过编程实现在这个正方形中产生均匀分布的点。落在圈内（红点）的点和总的投在正方形（红和绿点）上的点的比率给出了pi/4的近似值。这一过程称为使用蒙特卡洛方法来仿真逼近pi实际值。令N表示总的投在正方形的点。当投点个数分别是20, 50, 100, 200, 300, 500, 1000, 5000时，pi值分别是多少？对于每个N，每次实验算出pi值，重复这个过程20次，并在表中记下均值和方差。

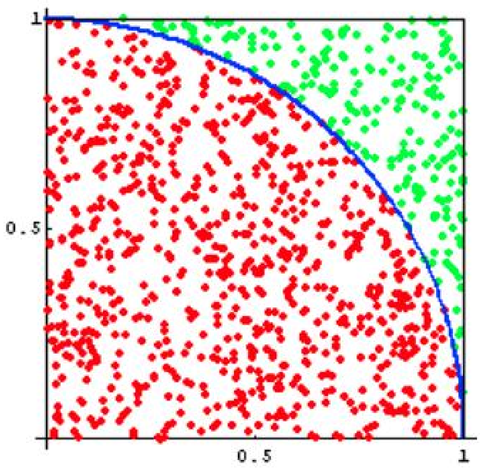

Figure 1 蒙特卡洛方法求解pi

源代码：

```
#include<bits/stdc++.h>
using namespace std;
#define COUNT 1000
//定义每次取样所用的点的个数

//判断采样的点是否在圆内
bool exit_in_circle(double x , double y){
	double length = x*x+y*y;
	return length < 1;
}

int main(){

	//用于生成随机数
	srand(time(NULL));
	for (int j = 0 ; j < 20 ; j++){
		int sum = 0 ;
		for (int i = 0 ; i < COUNT ;i++){
			double x = (double)rand()/RAND_MAX;
			double y = (double)rand()/RAND_MAX;
			if(exit_in_circle(x,y)){
				sum++;
			}
		}

		//计算pi值
		double result = (double)sum/(double)COUNT*4;
		cout << result << endl;
	}

}
```

数据统计：

N \次数	1	2	3	4	5
20	2.8	2.4	2.6	3.2	3
50	3.2	3.68	3.2	2.8	3.2
100	3	3.04	3.4	3.32	3.36
200	3.24	3.24	3.08	3.08	2.92
300	3.08	3.10667	3.34667	3.21333	3.10667
500	3.08	3.264	3.304	3.048	3.184
1000	3.108	3.056	3.128	3.212	3.104
5000	3.172	3.1696	3.104	3.1384	3.112

6	7	8	9	10	11
3	3.2	2.4	3	2.8	3.2
3.36	2.88	2.88	3.04	2.96	3.44
3.12	2.8	3	3.4	3.48	3.16
3.12	3.02	3.2	3.16	3.14	3.02
3.17333	3.06667	2.90667	3.21333	3.2	3.4
3.168	2.968	3.168	3.152	3.104	3
3.16	3.18	3.116	3.172	2.076	3.152
3.152	3.16	3.1328	3.1328	3.128	3.1624

12	13	14	15	16	17
3.4	3	3.6	3	2.2	3.6
3.12	2.96	2.96	2.88	3.6	3.12
3.08	3.12	3.04	3.28	3	3.28
3.24	3.26	3.28	3.14	3	3.24
3.09333	3.04	3.14667	2.93333	3.13333	3.09333
3.096	3.04	3.24	3.048	3.064	2.968
3.108	3.124	3.2	3.144	3.208	3.152
3.1632	3.1368	3.16	3.1256	3.124	3.1248

18	19	20	方差	均值
3.8	3	2.8	0.164	3
3.36	2.96	3.12	0.057984	3.136
2.96	3.16	3.08	0.030124	3.154
3.1	3.18	3.32	0.010819	3.149
3.10667	3.10667	3.2	0.012675524	3.1333335
3.096	3.104	3.256	0.00904384	3.1176
3.12	3.176	3.068	0.05567116	3.053538462
3.136	3.1696	3.1848	0.000471198	3.14444

Exercise 2.

We are now trying to integrate the another function by Monte Carlo method:

$$\int_0^1 x^3$$

A simple analytic solution exists here: $\int_{x=0}^{1} x^3 = 1/4$. If you compute this integration using Monte Carlo method, what distribution do you use to sample x? How good do you get when N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100, respectively? For each N, repeat the Monte Carlo process 20 times, and report the mean and variance of the integrate in a table.

我们现在尝试通过蒙特卡洛的方法求解如下的积分：

$$\int_0^1 x^3$$

该积分的求解我们可以直接求解，即有$\int_{x=0}^{1} x^3 = 1/4$。如果你用蒙特卡洛的方法求解该积分，你认为x可以通过什么分布采样获得？如果采样次数是分别是N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100，积分结果有多好？对于每个采样次数N，重复蒙特卡洛过程100次，求出均值和方差，然后在表格中记录对应的均值和方差。

源代码：

```
#include <bits/stdc++.h>
using namespace std;
#define COUNT 100

//判断是否位于函数图像的下方
bool under_the_line(double x, double y){
	double num = x*x*x;
	return y < num ;
}
int main(){
	srand(time(NULL));
	for(int j = 0 ; j<20 ; j++){
		int sum = 0 ;
		for (int i = 0 ; i < COUNT ;i++){
			double x = (double)rand()/RAND_MAX;
			double y = (double)rand()/RAND_MAX;
			if(under_the_line(x,y)){
				sum++;
			}
		}
		double result = (double)sum/COUNT;
		cout << result << endl;
	}
	return 0;
}
```

数据统计：

N \次数	1	2	3	4	5
5	0.2	0.2	0.2	0.6	0
10	0.1	0.2	0.5	0.1	0.1
20	0.1	0.3	0.1	0.45	0.2
30	0.2	0.3	0.266667	0.366667	0.266667
40	0.15	0.225	0.25	0.25	0.3
50	0.24	0.36	0.34	0.38	0.24
60	0.266667	0.216667	0.233333	0.216667	0.283333
70	0.314286	0.285714	0.2	0.271429	0.228571
80	0.325	0.15	0.25	0.3125	0.225
90	0.144444	0.255556	0.3	0.288889	0.166667
100	0.27	0.25	0.21	0.19	0.19

6	7	8	9	10	11
0	0.4	0.2	0.4	0.2	0.4
0.2	0.2	0.5	0.1	0.2	0.2
0.15	0.3	0.2	0.15	0.15	0.15
0.3	0.3	0.2	0.266667	0.2	0.366667
0.225	0.2	0.425	0.275	0.275	0.3
0.16	0.2	0.24	0.16	0.3	0.3
0.2	0.25	0.2	0.3	0.2	0.216667
0.171429	0.285714	0.242857	0.185714	0.257143	0.257143
0.1375	0.225	0.275	0.2625	0.3625	0.3375
0.222222	0.266667	0.266667	0.222222	0.233333	0.2
0.25	0.21	0.23	0.2	0.3	0.29

12	13	14	15	16	17
0.2	0.4	0.2	0	0.6	0
0.2	0.3	0.2	0.1	0.4	0.3
0.15	0.25	0.2	0.15	0.2	0.25
0.3	0.233333	0.266667	0.233333	0.233333	0.3
0.125	0.125	0.3	0.175	0.25	0.3
0.26	0.26	0.26	0.18	0.34	0.26
0.116667	0.35	0.216667	0.233333	0.216667	0.216667
0.271429	0.3	0.257143	0.328571	0.257143	0.257143
0.2875	0.2875	0.325	0.3	0.2625	0.1875
0.277778	0.222222	0.233333	0.255556	0.244444	0.222222
0.26	0.28	0.3	0.32	0.3	0.29

18	19	20	均值	方差
0. 2	0. 6	0. 2	0. 26	0. 0364
0. 4	0. 2	0. 3	0. 24	0. 0154
0. 3	0. 25	0. 3	0. 215	0. 007025
0. 3	0. 233333	0. 2	0. 2666667	0. 002444456
18	0. 3	0. 25	0. 245	0. 00485
0. 2	0. 22	0. 18	0. 261	0. 004299
0. 4	0. 2	0. 333333	0. 23750005	0. 002690959
0. 3	0. 257143	0. 228571	0. 2557143	0. 001487754
0. 3	0. 2875	0. 25	0. 264375	0. 003348047
18	0. 288889	0. 233333	0. 2383333	0. 001475009
0. 2	0. 19	0. 25	0. 247	0. 002091

Exercise 3:

We are now trying to integrate a more difficult function by Monte Carlo method that may not be analytically computed:

$$\int_{x=2}^{4}\int_{y=-1}^{1} f(x,y) = \frac{y^2 * e^{-y^2} + x^4 * e^{-x^2}}{x * e^{-x^2}}$$

Can you compute the above integration analytically? If you compute this integration using Monte Carlo method, what distribution do you use to sample (x,y)? How good do you get when the sample sizes are N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100, 200 respectively? For each N, repeat the Monte Carlo process 100 times, and report the mean and variance of the integrate.

我们现在尝试通过蒙特卡洛的方法求解如下的更复杂的积分：

$$\int_{x=2}^{4}\int_{y=-1}^{1} f(x,y) = \frac{y^2 * e^{-y^2} + x^4 * e^{-x^2}}{x * e^{-x^2}}$$

你能够通过公式直接求解上述的积分吗？如果你用蒙特卡洛的方法求解该积分，你认为(x, y)可以通过什么分布采样获得？如果点（x, y）的采样次数是分别是N = 10, 20, 30, 40, 50, 60, 70, 80, 100, 200, 500, 积分结果有多好？对于每个采样次数N，重复蒙特卡洛过程100次，求出均值和方差，然后在表格中记录对应的均值和方差。

方法一：

源代码：

```
#include <bits/stdc++.h>
using namespace std;

#define COUNT 500

double get_result(double x , double y ){
	double e;
	e=((y*y*exp((-1)*y*y)+pow(x,4)*exp((-1)*x*x)))/(x*exp((-1)*x*x));
	return e;
}

int main(){
	srand(time(NULL));
	double sum ;
	double result ;
	for(int j = 0 ; j < 20 ; j++){
		for (int i = 0 ; i < COUNT ;i++){
			double x = (double)rand()/RAND_MAX*2+2;
			double y = (double)rand()/RAND_MAX*2-1;

			result = get_result(x,y);
			sum += result;
		}

		double fina;
		fina = sum/COUNT*4;
		cout << fina << endl;
	}
	return 0;
}
```

数据统计：

N\次数	1	2	3	4	5
10	299824	397553	881189	974488	1.01E+06
20	24107.2	92723.3	121870	138793	321814
30	193646	383439	484730	554564	637943
40	113763	222730	318065	482063	584229
50	111795	313562	517067	6.07E+05	673003
60	54906.5	123272	339072	385318	532250
70	123261	167124	340291	421138	490381
80	83494	212097	279632	362005	447745
100	165669	255418	370252	458767	541323
200	98664	216012	313958	391540	496065
500	110541	239580	344910	452162	566768

6	7	8	9	10	11
1.10E+06	1.20E+06	1.29E+06	1.44E+06	1.44E+06	1.54E+06
430594	476957	583757	596149	754247	1.02E+06
786706	911886	986541	1.24E+06	1.33E+06	1.37E+06
751672	866054	969312	998535	1.03E+06	1.16E+06
820256	1.03E+06	1.12E+06	1.16E+06	1.30E+06	1.52E+06
704849	806990	891836	1.06E+06	1.21E+06	1.33E+06
564228	660534	793609	869830	1.02E+06	1.19E+06
571804	727554	836930	969479	1.08E+06	1.25E+06
633304	735659	922667	1.05E+06	1.18E+06	1.27E+06
583419	642286	759065	850498	997675	1.10E+06
665413	770599	861379	965199	1.08E+06	1.22E+06

12	13	14	15	16	17
1.77E+06	1.89E+06	1.95E+06	1.97E+06	1.97E+06	1.99E+06
1.16E+06	1.26E+06	1.43E+06	1.47E+06	1.54E+06	1.74E+06
1.38E+06	1.69E+06	1.88E+06	2.04E+06	2.13E+06	2.29E+06
1.19E+06	1.24E+06	1.32E+06	1.44E+06	1.55E+06	1.63E+06
1.64E+06	1.76E+06	1.92E+06	2.02E+06	2.09E+06	2.23E+06
1.43E+06	1.50E+06	1.69E+06	1.82E+06	1.94E+06	2.17E+06
1.37E+06	1.44E+06	1.58E+06	1.64E+06	1.72E+06	1.85E+06
1.40E+06	1.51E+06	1.66E+06	1.80E+06	1.90E+06	1.98E+06
1.36E+06	1.46E+06	1.61E+06	1.77E+06	1.91E+06	1.98E+06
1.22E+06	1.33E+06	1.45E+06	1.59E+06	1.68E+06	1.81E+06
1.33E+06	1.45E+06	1.57E+06	1.69E+06	1.82E+06	1.94E+06

18	19	20	均值	方差
2.18E+06	2.23E+06	2.42E+06	1497130.7	3.40071E+11
1.93E+06	2.14E+06	2.48E+06	986035.075	5.22422E+11
2.42E+06	2.46E+06	2.58E+06	1387856.75	5.52439E+11
1.72E+06	1.77E+06	1.98E+06	1066002.65	2.73092E+11
2.34E+06	2.42E+06	2.45E+06	1402051.7	5.22457E+11
2.34E+06	2.55E+06	2.66E+06	1276992.175	6.29891E+11
1.95E+06	2.03E+06	2.16E+06	1118686.3	4.12157E+11
2.03E+06	2.20E+06	2.30E+06	1180417	4.93366E+11
2.09E+06	2.23E+06	2.33E+06	1216957.45	4.55808E+11
1.91E+06	2.06E+06	2.18E+06	1084240.1	3.97955E+11
2.06E+06	2.15E+06	2.27E+06	1178342.55	4.31561E+11

方法二：

源代码：

```
#include <bits/stdc++.h>
using namespace std;

#define COUNT 500

double get_result(double x , double y ){

	double e;

	e=((y*y*exp((-1)*y*y)+pow(x,4)*exp((-1)*x*x)))/(x*exp((-1)*x*x));

	return e;
}

bool exit_under_fx(double z , double e){
	return z < e;
}

int main(){
	srand(time(NULL));
	for(int j = 0 ; j < 20 ; j++){
		int sum = 0;
		for(int i = 0 ; i < COUNT ; i++){
			double x = (double)rand()/RAND_MAX*2+2;
			double y = (double)rand()/RAND_MAX*2-1;
			double z = rand()%800000;
			double e = get_result(x,y);
			if(exit_under_fx(z,e)){
				sum++;
			}

		}
		double result = (double)sum/(double)COUNT*4*800000;
		cout << result<<endl;
	}
}
```

数据统计：

N\次数	1	2	3	4	5
10	0	640000	1.28E+06	640000	960000
20	640000	800000	640000	800000	1.12E+06
30	853333	533333	1.07E+06	533333	106667
40	480000	560000	1.20E+06	480000	640000
50	768000	704000	576000	640000	512000
60	693333	1.01E+06	746667	640000	693333
70	548571	594286	457143	640000	502857
80	640000	640000	520000	720000	560000
100	640000	800000	864000	608000	384000
200	752000	672000	576000	736000	544000
500	691200	672000	678400	665600	780800

6	7	8	9	10	11
0	0	960000	320000	640000	640000
800000	960000	640000	1.28E+06	1.12E+06	800000
426667	426667	426667	853333	640000	640000
800000	320000	800000	320000	560000	800000
768000	640000	1.02E+06	704000	640000	832000
746667	586667	586667	533333	533333	586667
548571	731429	411429	868571	777143	914286
640000	760000	560000	760000	560000	520000
832000	800000	608000	672000	672000	704000
816000	752000	656000	704000	720000	752000
608000	723200	582400	710400	646400	768000

12	13	14	15	16	17
1.28E+06	960000	960000	320000	960000	320000
320000	960000	960000	800000	800000	960000
746667	533333	213333	640000	640000	746667
640000	800000	480000	720000	640000	560000
576000	640000	896000	640000	512000	576000
693333	906667	640000	693333	480000	693333
731429	685714	411429	731429	594286	960000
640000	720000	720000	560000	800000	680000
896000	640000	704000	640000	704000	704000
592000	560000	784000	672000	736000	576000
678400	704000	672000	672000	627200	736000

18	19	20	均值	方差
320000	960000	640000	640000	1.536E+11
1.12E+06	640000	800000	848000	46336000000
533333	960000	320000	592000.15	55153916623
960000	640000	640000	652000	41136000000
1.02E+06	768000	704000	707200	20674560000
853333	640000	960000	695999.8	19477233067
731429	777143	594286	660571.55	23713954188
560000	480000	760000	640000	8800000000
576000	800000	704000	697600	12861440000
608000	592000	752000	677600	6714240000
755200	704000	704000	688960	2438246400

方法一使用的是随机产生点的x,y坐标，并将这个点对应的函数图像的Z坐标求出来，多次计算求平均值，得出一个结果，然后与4（及在xy平面内的投影面积）相乘得出对应的结果。

方法二是随机生成一个以xy平面内的投影面为底，以在此范围函数图像应的最大的z值为高（800000）的长方体内的点的坐标，计算处于函数图像下方（即位于被积空间内部）的概率，然后乘以长方体的体积，得到相应的结果。