

计算物理 A 第三题

杨旭鹏 PB17000234

2019 年秋季

1 题目描述

在球坐标系 (ρ, θ, ϕ) 下产生球面上均匀分布的随机坐标点，给出其直接抽样方法。

2 算法及程序思路

2.1 算法

2.1.1 16807 产生器

16807 产生器属于线性同余法产生器的特例。而线性同余法方法为：

$$\begin{aligned} I_{n+1} &= (aI_n + b) \bmod m \\ x_n &= I_n / m \end{aligned} \tag{1}$$

其中整数 $I_i \in [0, m - 1]$, a, b, m 为算法中的可调参数，其选取直接影响产生器的质量。选取参数：

$$\begin{cases} a = 7^5 = 16807 \\ b = 0 \\ m = 2^{31} - 1 = 2147483647 \end{cases} \tag{2}$$

即为所谓的 16807 产生器。由于直接利用 1 编写程序时计算 $(aI_n \bmod m)$ 时很容易造成数据溢出，故采取 Schrage 方法进行具体编程的实现：

$$aI_n \bmod m = \begin{cases} a(I_n \bmod q) - r[I_n/q], & if \geq 0 \\ a(I_n \bmod q) - r[I_n/q] + m, & otherwise \end{cases} \tag{3}$$

其中 $m = aq + r$, 即 $q = m/a = 127773$, $r = m \bmod a = 2836$ 。即可利用此方法产生伪随机数序列。

2.1.2 Fibonacci 延迟产生器: Marsaglia 1 号产生器

其思想是用序列中的两个整数进行操作得到后续的整数, 较线性同余法的优势在于它的周期非常长:

$$I_n = I_{n-p} \otimes I_{n-q} \bmod m \quad (4)$$

其中操作符 \otimes 可以是: “+”, “-”, “ \times ”, “XOR”。整数对 $[p, q]$ 表示延迟, 取值并非按 Fibonacci 数序列, 而是根据统计验证后确认。

在此程序中, 我们使用的是 Fibonacci 延迟产生器的一个特例——Marsaglia 1 号产生器:

$$I_n = \begin{cases} I_{n-p} - I_{n-q}, & if \geq 0 \\ I_{n-p} - I_{n-q} + 1, & otherwise \end{cases} \quad (5)$$

其中的 $[p, q]$ 整数对的值选为 $[97, 33]$, 因此其算法要求存储所有前面的 97 个随机数值。在本程序中, 前 97 个随机数值由在 Linux 系统下访问 “/dev/random”一次性来得到¹, ²并存储为文件以便后期读取使用 (此文件的原始数据见附录 B 和 C)。

2.1.3 直接抽样法

对于任意已归一化的几率密度分布函数 $p(x)$ ($x \in [a, b]$), 其累积函数 $\xi(x)$ 的计算式为:

$$\xi(x) = \int_a^x p(x') dx' \quad (6)$$

其有性质 $\xi(a) = 0, \xi(b) = 1$ 。则根据上式求出 $\xi(x)$ 的反函数 $x(\xi)$, 则当 ξ 为 $[0, 1]$ 上均匀分布时, 得到的 $x(\xi)$ 按照概率密度函数 $p(x)$ 分布。此即直接抽样法。

¹对于 Linux 下的“/dev/random”文件在 Linux manual page 上有如下解释: The random number generator gathers environmental noise from device drivers and other sources into an entropy pool. The generator also keeps an estimate of the number of bits of noise in the entropy pool. From this entropy pool, random numbers are created. 可以看出此种方法产生的随机数并非来自于算法, 而来自于热噪声。

²便于结果的可重复性, 方便调试程序

在本问题中，需要得到在球面上均匀分布的随机点，即在球面上任意位置，单位面积上点出现的概率为一定值。在某一球面上，半径 ρ 值是固定的，自变量仅为角度 θ 和 ϕ 。设概率密度函数为 $p(\theta, \phi)$ ，则对于单位面积元上点出现的概率 dP 有：

$$const = dP = p(\theta, \phi) d\theta d\phi dS^2 \quad (7)$$

当 $dS^2 = \sin(\theta)d\theta d\phi$ 为定值时，我们有 $d\theta \propto \frac{1}{\sin(\theta)}$ 。则为了满足上式，我们有 $p(\theta, \phi) \propto \sin(\theta)$ 。设 $p(\theta, \phi) = k \sin(\theta)$ ，根据密度函数的归一性有

$$1 = \int_0^{2\pi} \int_0^\pi k \sin(\theta) d\theta d\phi \quad (8)$$

得到 $k = \frac{1}{4\pi}$ 。即有 $p(\theta, \phi) = p_\theta(\theta)p_\phi(\phi) = \frac{1}{4\pi} \sin(\theta)$ ，再根据 $p_\theta(\theta)$ 和 $p_\phi(\phi)$ 的归一性得到：

$$\begin{cases} p_\theta(\theta) = \frac{1}{2} \sin(\theta) \\ p_\phi(\phi) = \frac{1}{2\pi} \end{cases} \quad (9)$$

则 $p_\theta(\theta)$ 和 $p_\phi(\phi)$ 的累积函数 $\xi(\theta)$, $\eta(\phi)$ 分别为：

$$\begin{cases} \xi(\theta) = \frac{1}{2} - \frac{1}{2} \cos(\theta) \\ \eta(\phi) = \frac{\phi}{2\pi} \end{cases} \quad (10)$$

则其反函数为：

$$\begin{cases} \theta(\xi) = \arccos(2\xi - 1) \\ \phi(\eta) = 2\pi\eta \end{cases} \quad (11)$$

则若生成两组互不相关的 $[0, 1]$ 间均匀分布的随机点 ξ, η ，即可利用上式得到球面上均匀分布点的 θ 和 ϕ 坐标。

2.2 程序思路

对于 16807 方法，程序需要一个种子，在本程序中需要产生两组互不相关的随机数，则需要 2 个“互不相关”的初始种子值。这两个种子值通过一次性读取“/dev/random”文件得到，分别为 34028207 和 1677078722³。

³在这里我们这么做是为了结果的可重复性

产生随机种子后，各自利用递推公式：

$$I_{n+1} = aI_n \bmod m = \begin{cases} a(I_n \bmod q) - r[I_n/q], & if \geq 0 \\ a(I_n \bmod q) - r[I_n/q] + m, & otherwise \\ a = 7^5 = 16807 \\ q = m/a = 127773 \\ r = m \bmod a = 2836 \end{cases} \quad (12)$$

计算随机数列。

对于 Marsaglia 1 号产生器，所需要 97 个初始随机数，得到方式在上文已经提及。

产生 $[0, 1]$ 的随机数后即可利用上文提到的直接抽样法得到球面均匀分布点的坐标 (θ, ϕ) 。

详情请见代码。

3 程序使用方法

在运行程序后，会看到请求输入所需总随机点数的提示，按照提示在后面输入所需要的总随机点数，摁回车继续。然后按照程序提示，输入 Marsaglia 1 号产生器所需延迟整数值 $[p, q]$ （最大值不可超过 97，Marsaglia 1 号产生器简易使用默认延迟整数值为 $[97, 33]$ ，建议按此输入），按回车继续程序。然后经过计算给出 16807 方法和 Marsaglia 方法产生的随机坐标 (θ, ϕ) 分别存为两个文件。程序输出完这些后会自动退出。

```
请输入您所需要的总点数: 1000
请输入Marsaglia方法的延迟整数对[p,q](建议输入97,33)(英文逗号隔开): 97,33
```

```
进程已结束, 退出代码 0
```

图 1：一个典型程序的运行过程示例

4 程序结果与讨论

当输入一些不同的点数时，得到如下结果：

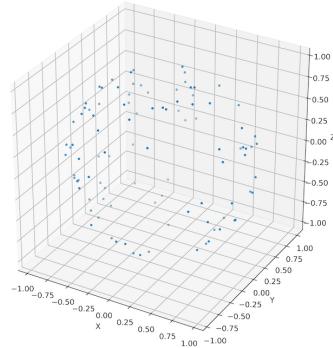
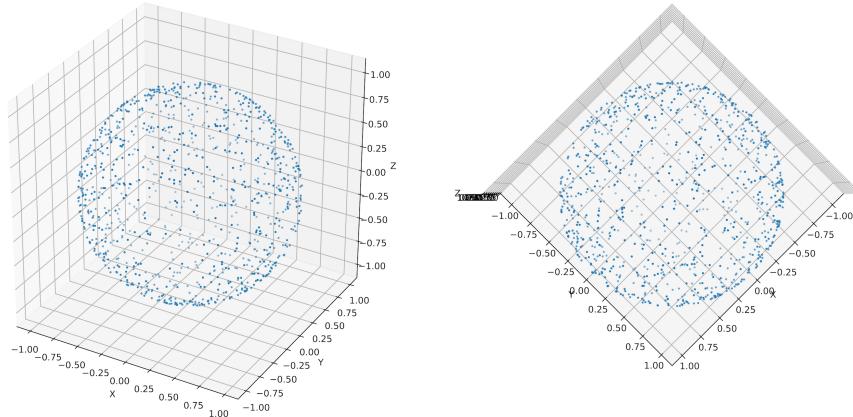


图 2：球面上均匀分布的 100 随机点



(a) 默认视图

(b) z 方向视图

图 3：球面上均匀分布的 1000 随机点

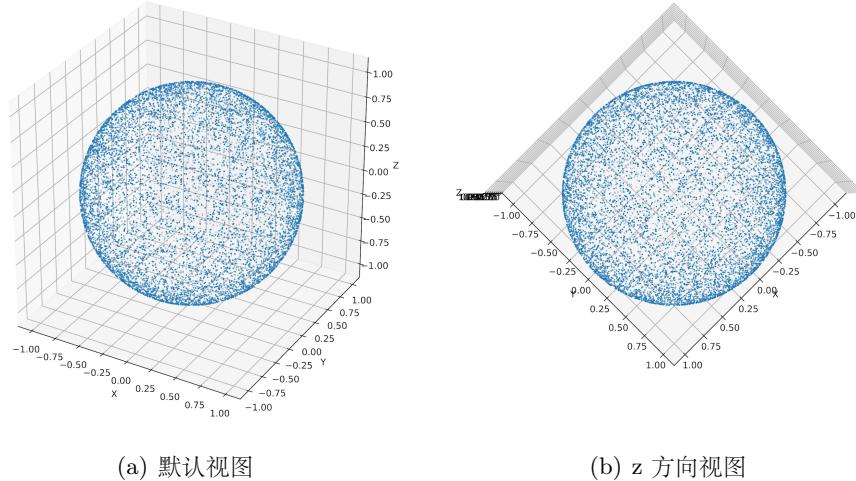


图 4: 球面上均匀分布的 10^4 随机点

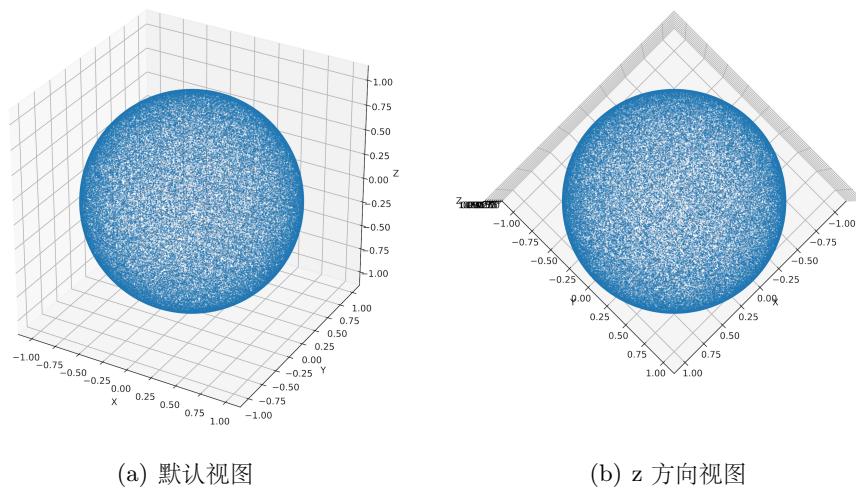
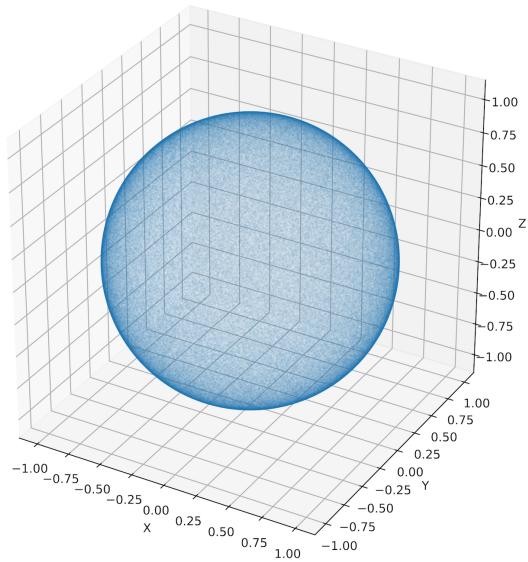
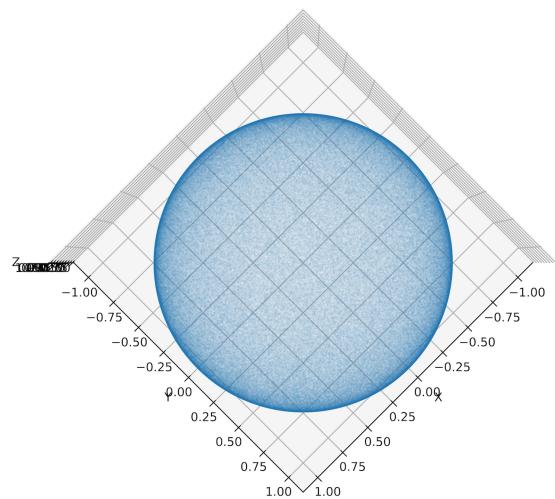


图 5: 球面上均匀分布的 10^5 随机点



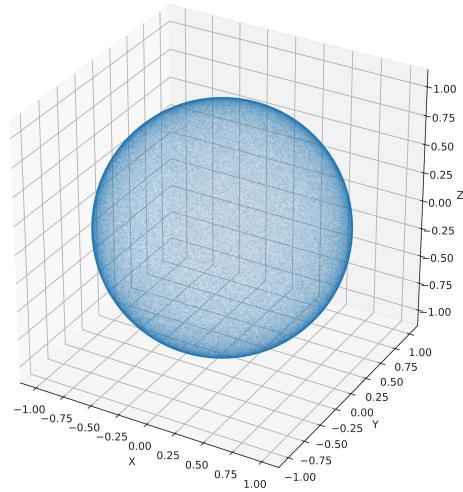
(a) 默认视图



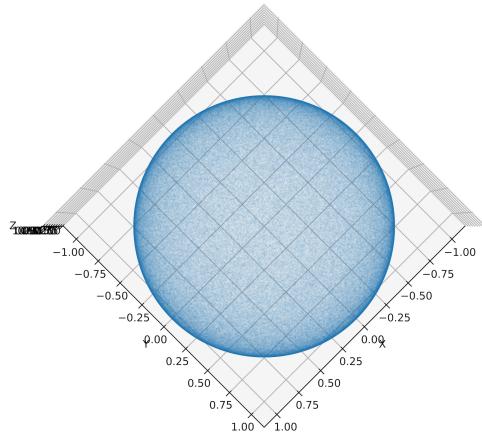
(b) z 方向视图

图 6: 球面上均匀分布的 10^6 随机点

由上述结果可以看出利用 16807 产生器, 当 $N \leq 10^6$ 时, 随机点在球面上的分布还算均匀, 几乎看不出什么条带结构和规则网格结构等有明显规律性的结构。更高次方的点数由于计算机计算能力有限, 画图比较困难, 故并没有在这里给出图像和分析。当我们换取 Marsaglia 1 号产生器进行试验, 得到如下结果:



(a) 默认视图



(b) z 方向视图

图 7: Marsaglia 1 号产生器产生的球面上均匀分布的 10^6 随机点

可以看出利用 marsaglia 1 号产生器产生的球面上均匀分布的随机点在 $N = 10^6$ 时看不出什么有规律性的结构，仍可认为是比较均匀的。

5 心得与体会

通过此次作业结果，发现 16807 产生器和 Marsaglia 1 号产生器产生的随机数的均匀性在 $N \leq 10^6$ 时还是比较好的⁴。通过编程作业，也更加熟悉了一些 C 语言和 LATEX。

⁴至少在肉眼看上去

6 附录

A 产生随机数种子的 C 语言程序

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 //利用/dev/random 产生随机种子
5
6 int my_randomseed(int seed[],int n){
7
8 //seed为存储随机种子的数据, n为所需要种子的个数
9
10 FILE * fp1 = fopen("/dev/random","r");
11
12 for(int i=0;i<n;i++){
13     fread(&seed[i], 1, sizeof(int), fp1);
14 }
15
16 fclose(fp1);
17
18 return 0;
19 }
20
21 int main(){
22 int ranI[97];
23
24 my_randomseed(ranI,97);
25
26 FILE * fp;
27
28 fp = fopen("ranI.txt","w+");
29
30 for(int i=0;i<96;i++){
31     fprintf(fp,"%d,",ranI[i]);
32 }
```

```
33
34     fprintf(fp, "%d", ranI[96]); //最后一个数据后不加 ","
35
36     fclose(fp);
37
38     return 0;
39
40 }
```

B 由上述程序产生的随机数文件 1

809576131,-1025892587,558213681,2056216731,-1652403892,-849346475,
-947398527,-2082766203,131013461,753913035,-634671236,995067675,-1025241712,
-57745909,187349022,-1487194468,174292363,-1523076266,-1381700026,1751317799,
812085654,-384904015,935241924,-2120102602,-1176252493,2084189454,-1308162651,
-721701972,1776417975,-1953284964,-1385173859,-1156931024,-272454405,-1527712783,
1040918716,966408491,-899150905,-1102248190,363327930,1940215723,924796768,861925965,
1030548881,1694050903,-1841641721,-5770120,448493076,1650846131,1012895949,
-2135767741,1924774890,-885687700,-219340479,-1417922044,-1101149761,1198019956,
-1522975896,842839847,2023490906,-1533950707,1267038558,1823349522,-1476843016,
786347740,1101867560,110498268,-1018391091,-1677263454,2117579346,-306068839,
569112568,-1031628950,1399939025,1102660725,-526224915,212410442,-968658785,
1808251768,166339465,-347114238,-1130541052,450232354,-684115741,-1313290066,
140916175,-1221648810,-1079940150,2079743593,414264974,-1639098679,142289346,
-2080694919,-345845707,-1201999899,1632261567,98605000,-1843368371

C 由上述程序产生的随机数文件 2

-753424879,1872953453,608893860,597180613,2049110889,2140413464,1070133919,
-256067828,-1191457009,-574707053,-1410977199,1133706033,-404358309,1141209606,
-1604908044,2123698944,1542004389,894557832,-1972430293,-1544149044,-155439154,
2108522693,1220970092,697871479,-1502727159,1427890000,-1629207648,-132267497,
-749675637,1645721388,-1858854966,-2111533027,-151887203,-1412609570,-876296650,
-1227304997,701135804,1224292810,1805574955,-955336658,619132236,1948713847,

1737807285,845890383,-1491155579,-650075247,-1329866879,1910267507,-1879619968,
289013714,-24743716,1328343531,656220109,2045662973,-137884045,-1218521402,
1439903068,1203175289,-2098367843,1830857371,106846260,-1413985256,584445562,
-1141203261,-1584804986,-2109822000,-1573941759,1728136137,-54161293,-1017405251,
1652078157,1750018552,1594371093,1725375480,1445249495,-1114348371,-237895730,
1481242551,-1581295506,-970746651,397491742,462545489,-436989135,-1245824880,
-945627835,773430348,429365809,-108076855,620101426,1362478058,-923198353,
-1447637852,-1914066341,-1674823883,824439501,-461869274,-634359412

D C 语言程序源码

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <math.h>
5 #define a 16807
6 #define b 0
7 #define m 2147483647
8 #define r (m%a)
9 #define q (m/a)
10 #define Pi 3.1415926
11
12
13
14 int my_filereader_int(char str[],int num[],int n){
15     FILE * fp;
16     fp = fopen(str,"r");
17
18     for(int i=0;i<n-1;i++)
19     {
20         fscanf(fp,"%d,",&num[i]);
21
22     }
23     fscanf(fp,"%d",&num[n-1]); //最后一个数据后不加 ","
24     fclose(fp);
```

```

25     return 0;
26 }
27
28 int my_filewriter(char str[],double num[],int n){
29     FILE * fp;
30     fp = fopen(str,"w+");
31
32     for(int i=0;i<n;i++){
33     {
34         if (i == (n-1)){
35             fprintf(fp,"%lf",num[i]);
36             break;
37         }
38         fprintf(fp,"%lf,",num[i]);
39     }
40     fclose(fp);
41     return 0;
42 }
43
44
45
46 int my_schrage_sphere(int seed[],double rantheta[],double
47 ranphi[],int n){
48     for (int i = 0; i < n; i++) {
49         if(seed[0] == m-1){
50             if(a >= b){ //由于Schrage方法只对z in
51                 (0,m-1)成立，故这里要讨论z == m-1的情况
52                 seed[0] = m + (b-a) % m;
53             }
54             else seed[0] = (b-a) % m;
55         }
56         if(seed[1] == m-1){
57             if(a >= b){ //由于Schrage方法只对z in
58                 (0,m-1)成立，故这里要讨论z == m-1的情况
59                 seed[1] = m + (b-a) % m;
}
59         else seed[1] = (b-a) % m;

```

```

60
61     }
62     seed[0] = ((a * (seed[0] % q) - r * (seed[0] / q)) + b % m )
63         % m;
64     seed[1] =((a * (seed[1] % q) - r * (seed[1] / q)) + b % m )
65         % m;
66     if (seed[0] >= 0) {
67         rantheta[i] = acos(2 * (seed[0] / (double)m) - 1);
68     }
69     else{
70         rantheta[i] = acos(2 * ((seed[0] + m) / (double)m) - 1);
71     }
72     if (seed[1] >= 0) {
73         ranphi[i] = 2 * Pi * seed[1] / (double)m;
74     }
75     else{
76         ranphi[i] = 2 * Pi * (seed[1] + m) / (double)m;
77     }
78
79 }
80 }
81 }
82
83
84
85 //Fibonacci延迟器
86 int my_fibonacci_theta(double ran[],int ranI[],int n,int o,int p){
87     int temp;
88
89     for (int i = p; i < n; i++) { //ranI[i % p]存放的为第i项
90
91         temp = (ranI[i % p] - ranI[(i % p+(p-o)) % p]) ; //递推式
92
93         if(temp >= 0) ranI[i % p] = temp ;
94         else ranI[i % p] = temp + 1;
95         //递推得到的新结果放入ranI中i%p一项

```

```

95
96     if (ranI[i % p] >= 0) { //计算ran
97         ran[i-p] = ranI[i % p] / (double) m;
98
99     } else {
100        ran[i-p] = (ranI[i % p] + m) / (double) m;
101    }
102    ran[i-p] = acos(2*ran[i-p]-1);
103
104 }
105 return 0;
106 }

107 //Fibonacci延迟器
108 int my_fibonacci_phi(double ran[],int ranI[],int n,int o,int p){
109     int temp;
110
111     for (int i = p; i < n; i++) { //ranI[i % p]存放的为第i项
112
113         temp = ranI[i % p] - ranI[(i % p+(p-o)) % p]; //递推式
114
115         if(temp >= 0) ranI[i % p] = temp ;
116         else ranI[i % p] = temp + 1;
117         //递推得到的新结果放入ranI中i%p一项
118
119         if (ranI[i % p] >= 0) { //计算ran
120             ran[i-p] = ranI[i % p] / (double) m;
121         } else {
122             ran[i-p] = (ranI[i % p] + m) / (double) m;
123         }
124         ran[i-p] = 2*Pi*ran[i-p];
125
126     }
127     return 0;
128 }

129 int main() {
130
131

```

```

132     int N;
133     char str[50];
134     printf("请输入您所需要的总点数: ");
135     while (!scanf("%d",&N)){ //简单的输入检查
136         gets(str);
137         printf("\nInput error,please try again\n");
138         printf("请输入您所需要的总点数: ");
139     }
140
141
142
143     int seed[2] = {34028207,1677078722};
144     //设定初始种子值, 来源于读取"/dev/random"
145
146     double * rantheta = malloc(sizeof(double)*N);
147     double * ranphi = malloc(sizeof(double)*N);
148
149
150
151
152     int o,p;
153     inputpq:printf("请输入Marsaglia方法的延迟整数对[p,q](建议输入97,33)(英文逗号隔开): ");
154     while (!scanf("%d,%d",&o,&p)) { //相对简单的输入检查
155         gets(str);
156         printf("\nInput error,please try again\n");
157         printf("请输入Marsaglia方法的延迟整数对[p,q](建议值97,33)(英文逗号隔开): ");
158     }
159
160
161     int temp;
162     if(o > p){ //使 p > o
163         temp = o;
164         o = p;
165         p=temp;
166     }
167     if( N-p < 3 ) { //排除N相对于p,q太小的情况发生
168         printf("输入的p,q值相对于总点数太大了, 请重试!\n");

```

```
169     goto inputpq;
170 }
171
172
173 if(N > 1000000) printf("您输入的参数已接受，正在计算请稍后~\n");
174
175 my_schrage_sphere(seed,rantheta,ranphi,N);
176 //16807方法生成theta,phi
177
178 my_filewriter("theta_16807.dat",rantheta,N);
179 my_filewriter("phi_16807.dat",ranphi,N);
180
181 free(rantheta);
182 free(ranphi);
183
184 rantheta = malloc(sizeof(double)*(N+p));
185 ranphi = malloc(sizeof(double)*(N+p));
186
187 int *ranI1 = malloc(sizeof(int) * p); //用来存放递推数列
188 int *ranI2 = malloc(sizeof(int) * p);
189 my_filereader_int("ranI-1.txt",ranI1,p);
190 //读取之前利用"/dev/random"产生的初始随机数组
191 my_filereader_int("ranI-2.txt",ranI2,p);
192
193
194 my_fibonacci_theta(rantheta,ranI1,N+p,o,p);
195 my_fibonacci_phi(ranphi,ranI2,N+p,o,p);
196
197 my_filewriter("theta_fibo.dat",rantheta,N);
198 my_filewriter("phi_fibo.dat",ranphi,N);
199
200 return 0;
201 }
```

E 可视化绘图 python 程序源码

```
1
2 import matplotlib.pyplot as plt
3 from mpl_toolkits.mplot3d import Axes3D
4 import numpy as np
5 from IPython.core.pylabtools import figsize # import figsize
6 figsize(7, 7) # 设置 figsize
7 plt.rcParams['savefig.dpi'] = 1000 #图片像素
8 plt.rcParams['figure.dpi'] = 1000 #分辨率
9 # 默认的像素: [6.0,4.0], 分辨率为100, 图片尺寸为 600&400
10
11 fig = plt.figure()
12 ax = Axes3D(fig)
13 theta = []
14 phi = []
15
16 with open('cmake-build-debug/theta_16807.dat', 'r') as f:
17     while True:
18         lines = f.readline() # 整行读取数据
19         if not lines:
20             break
21         pass
22         tmp = [float(i) for i in lines.split(',')]] #
23             # 将整行数据分割处理。 theta.append(tmp) # 添加新读取的数据
24         pass
25     theta = np.array(theta) # 将数据从list类型转换为array类型。
26
27
28 with open('cmake-build-debug/phi_16807.dat', 'r') as f:
29     while True:
30         lines = f.readline() # 整行读取数据
31         if not lines:
32             break
33         pass
34         tmp = [float(i) for i in lines.split(',')]] #
```

将整行数据分割处理。

```
35     phi.append(tmp) # 添加新读取的数据
36     pass
37     phi = np.array(phi) # 将数据从list类型转换为array类型。
38     pass
39
40
41
42
43 X = np.sin(theta) * np.cos(phi)
44 Y= np.sin(theta) * np.sin(phi)
45 Z = np.cos(theta)
46
47
48 # 绘制散点图
49 ax.scatter(X, Y, Z,s=0.0001) #画散点图
50 ax.view_init(elev=90,azim=45) #设置观察角度
51 ax.set_aspect('equal') #设置三个坐标轴比例尺相同
52 ax.set_xlabel('X')
53 ax.set_ylabel('Y')
54 ax.set_zlabel('Z')
55 plt.savefig("1.png")
```