

计算物理作业一

于浩然 PB19020634 2021.10.04

1 作业题目

用 Schrage 方法编写随机数子程序，用指定间隔（非连续 $l > 1$ ）两个随机数作为点的坐标值绘出若干点的平面分布图。再用 $\langle x^k \rangle$ 测试均匀性（取不同量级的 N 值，讨论偏差与 N 的关系）、 $C(l)$ 测试其二维独立性（总点数 $N > 10^7$ ）。

2 算法简介

(a) Schrage 方法简介

最简单的均匀随机数产生器是线性同余法产生器 (LCG)，随机数序列 x_n 按线性关系得到，其原理如下：

$$I_{n+1} = (aI_n + b) \bmod m \quad (1)$$

$$x_n = I_n / m \quad (2)$$

具体举例，有一种通过了许多理论测试和实用性考验的 LCG，它通常被叫做“16807”产生器，其参数满足：

$$a = 7^5 = 16807, \quad b = 0, \quad m = 2^{31} - 1 = 2147483647 \quad (3)$$

但在具体运算中， aI_n 很可能在取模前就超过计算机能够表示的最大整数值，故需要设计一种让计算机能够在其运算范围内取模的方法，亦即我们所采用的 Schrage 方法：

设 m 可表为

$$m = aq + r, \quad q = [m/a], \quad r = m \bmod a \quad (4)$$

则 $aI_n \bmod m$ 可用如下等式表达

$$aI_n \bmod m = \begin{cases} I_n(z \bmod q) - r[z/q] & , if \geq 0 \\ I_n(z \bmod q) - r[z/q] + m, & otherwise \end{cases} \quad (5)$$

(b) k 阶矩检验均匀性

均匀性是指在 $[0, 1]$ 区间内等长度子区间中随机数的数量相同。对于均匀分布的

随机变量 x , 其 k 阶矩满足如下关系:

$$\langle x^k \rangle = \frac{1}{N} \sum_{i=1}^N x_i^k \Rightarrow \int_0^1 x^k p(x) dx = \frac{1}{k+1} \quad (6)$$

上面 (6) 式中 $p(x)$ 为均匀分布的概率密度函数, 其值为 1.

$$\left| \langle x^k \rangle - \frac{1}{k+1} \right| \approx \mathcal{O} \left(\frac{1}{\sqrt{N}} \right) \quad (7)$$

由上述关系, 可通过计算随机数的 k 阶矩来检验其均匀性, $\left| \langle x^k \rangle - \frac{1}{k+1} \right|$ 越小说明其均匀性越好。

(c) 自相关系数检验 2 维独立性

独立性是指按先后顺序出现的随机数中, 每个随机数的取值与其相距一定间隔的随机数取值之间无关。我们使用的讨论随机数序列独立性的方法是顺序相关法, 它用相距一定距离的两个随机数的自相关函数 (或相关系数) 来标识伪随机数序列的独立性情况, 表达式如下:

$$C(l) = \frac{\langle x_n x_{n+l} \rangle - \langle x_n \rangle^2}{\langle x_n^2 \rangle - \langle x_n \rangle^2} \quad (8)$$

其中 $\langle x_n \rangle$ 表示平均值 $\sum_{n=1}^N x_n / N$. 相关系数小只能保证 x_n 与 x_{n+l} 之间线性关系弱, 但无法保证不存在其他类型的函数关系。

$$|C(l)| \approx \mathcal{O} \left(\frac{1}{\sqrt{N}} \right) \quad (9)$$

通过上述关系, 我们可计算 $C(l)$ 的值来考察随机数的独立性。

3 编程实现

采用最简单的 16807 生成器进行编程实现, 由 (3)(4) 可得

$$a = 16807, \quad q = 127773, \quad r = 2836, \quad m = 2147483647 \quad (10)$$

使用 Fortran90 进行编程, 共包含三个子程序, 功能简介与源代码如下:

- SUBROUTINE Schrage(P)

通过传入的参数 P 确定要生成的随机数数目 (10^P), 为简单可重复起见我们采用的种子为 $In(1) = m - 1$. 按 Schrage 方法生成随机数并将其存入实型数组 z, 最后将数组 z 写入文件 rand.out.

```

1 SUBROUTINE Schrage(P) !Schrage随机数生成器子程序
2   IMPLICIT NONE
3   INTEGER :: N = 1, P
4   INTEGER :: m = 2147483647, a = 16807, q = 127773, r =
5     2836, In(10**P)
6   REAL(KIND=8) z(10**P)
7   In(1) = m - 1
8   z(1) = REAL(In(1))/m
9   DO N = 1, 10**P - 1
10    In(N + 1) = a*MOD(In(N), q) - r*INT(In(N)/q)
11    IF (In(N + 1) < 0) THEN !若值小于零，按Schrage方法加m
12      In(N + 1) = In(N + 1) + m
13    END IF
14    z(N + 1) = REAL(In(N + 1))/m !得到第N+1个随机数
15  END DO
16  OPEN (1, file='rand.out') !每次运行子程序将覆盖随机数
17  DO N = 1, 10**P !将随机数按行存入文件
18    WRITE (1, *) z(N)
19  END DO
20  CLOSE (1)
21 END SUBROUTINE Schrage

```

- SUBROUTINE Moment(P)

由传入参数 P 确定随机数组大小，读取先前生成的包含 10^P 个随机数的文件 rand.out 写入实型数组 z，根据 (6) 计算出 k 阶矩的值 S(k)，由 (7) 计算出 k 阶矩的偏差 D(k)，分别写入文件 moments.out 和 difference.out.

```

1 SUBROUTINE Moment(P) !k阶矩均匀性检验子程序(N一定)
2   INTEGER(KIND=4) :: i = 1, k = 1, P
3   REAL(KIND=8), DIMENSION(10) :: S = 0, D = 0
4   REAL(KIND=8) :: z(10**P)
5   OPEN (1, file='rand.out') !从前面产生的'rand.out'文件中读
6     取随机数表
7   READ (1, *) z
8   CLOSE (1)

```

```

8      DO k = 1, 10 !求随机数的1阶矩至10阶矩
9          S(k) = SUM(z**k) / SIZE(z)
10         D(k) = ABS(S(k) - real(1) / (1 + k)) !求出各k值对应的
11             偏差
12     END DO
13     OPEN (99, ACCESS='append', file='moments.out') !将k阶均
14         值存入文件
15     WRITE (99, *) S
16     CLOSE(99)
17     OPEN (3, ACCESS='append', file='difference.out') !将偏差
18         存入文件
19     WRITE (3, *) D
20     CLOSE(3)
21 END SUBROUTINE Moment

```

- SUBROUTINE Independence(P)

由传入参数 P 确定随机数组大小, 读取文件写入实型数组 (同上). 根据公式 (8)(9) 计算 $C(l)$, 将结果写入文件 indep.out.

```

1 SUBROUTINE Independence(P) !2维独立性检验子程序
2     INTEGER(KIND=4) :: P, l = 1, i = 1
3     REAL(KIND=8) :: ave = 0, coave = 0, sqrave = 0 !分别表示<
4         x>,<x_i*x_{i+1}>,<x^2>
5     REAL(KIND=8), DIMENSION(1:4) :: C
6     REAL(KIND=8), DIMENSION(1:10**P) :: z
7     OPEN (1, file='rand.out')
8     READ (1, *) z
9     CLOSE(1)
10    ave = SUM(z) / SIZE(z) !求均值
11    sqrave = SUM(z**2) / SIZE(z)
12    DO l = 1, 4
13        DO i = 1, 10**P - 1
14            coave = coave + z(i) * z(i + 1)
15        END DO
16        DO i = 1, l !在这里将越界的z(i+1)替换为z(N+i-1)

```

```

16         coave = coave + z(i) * z(10**P + i - 1)
17      END DO
18      coave = coave / SIZE(z)
19      C(l) = ABS((coave - ave)**2)/(sqgrave - ave**2) ! 相关
          系数公式
20    END DO
21    OPEN (99, ACCESS='append', file='indep.out')
22    WRITE (99, *) C ! 将相关系数值存入文件
23    CLOSE (99)
24 END SUBROUTINE Independence

```

在主程序中只需声明一个变量 P，使用一个 DO 循环结构，在随机数序列大小为 10^2 到 10^7 的各次幂时，分别调用上述子程序。

```

1 PROGRAM MAIN
2   IMPLICIT NONE
3   INTEGER :: P=1
4   !PRINT *, 'How many random numbers are required?(10^P)'
5   !READ *, P !从键盘读取随机数数量大小
6   DO P = 2, 7
7     CALL Schrage(P)
8     CALL Moment(P)
9     CALL Independence(P)
10  END DO
11 END PROGRAM MAIN

```

为实现绘制散点图功能，使用了 Python 脚本，通过 `matplotlib.pyplot.scatter` 实现，源代码展示如下：

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import matplotlib as mpl
4 plt.rcParams['savefig.dpi'] = 600

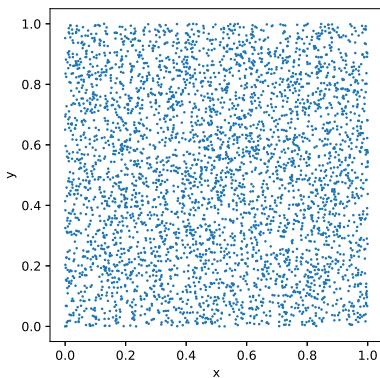
```

```
5 plt.rcParams['figure.dpi'] = 600
6
7 fig, ax = plt.subplots()
8 ax.set_aspect(1)
9 data = np.loadtxt("rand6.out")
10 LENGTH = 4000 # 绘制4000个点的散点图
11 #LENGTH = 20000 # 绘制20000个点的散点图
12 #LENGTH = 40000
13 x = np.zeros(LENGTH)
14 y = np.zeros(LENGTH)
15 for num in range(1, LENGTH):
16     x[num] = data[num]
17     y[num] = data[num + 3] # 取1值为3
18 plt.scatter(x, y, s=2)
19 ax.set_xlabel('x')
20 ax.set_ylabel('y')
21 plt.savefig("rand.png")
```

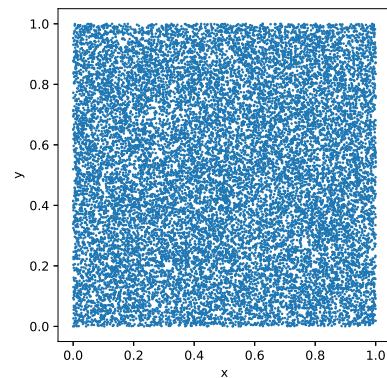
4 计算结果

(a) 随机数平面分布图

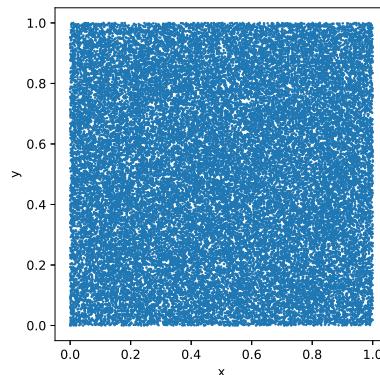
运行 Python 脚本可得到给定种子一定数目随机数的平面分布图。种子一定时，生成不同数目随机数的前 N 个数都是相同的，故为了读取数据速度快直接读取 10^6 个随机数的文件 `rand6.out`. 取 l 值为 3，即 y 坐标的随机数为对应 x 坐标的随机数后面的第三个数. 取 LENGTH 分别为 4000,20000,40000 绘图，展示如下. 可看出，16807 产生的随机数具有很好的性质，在平面图上无法看出随机数间的关联性。



(a) 4000 个随机数平面分布图



(b) 20000 个随机数平面分布图



(c) 40000 个随机数平面分布图

图 1: 随机数平面分布图

(b) k 阶矩检验均匀性

将 `moments.out` 中的数据整理列表如下, 保留小数点后四位, 将 $\langle x^k \rangle$ 与 $1/(k+1)$ 比较, 可以观察出, 随着 N 增大, $\langle x^k \rangle$ 的值越来越趋近 $1/(k+1)$.

$\langle x^k \rangle \backslash k$	1	2	3	4	5	6	7	8	9	10
N	10 ²	10 ³	10 ⁴	10 ⁵	10 ⁶	10 ⁷				
10 ²	0.4857	0.3245	0.2474	0.2028	0.1739	0.1536	0.1386	0.1269	0.1176	0.1098
10 ³	0.5022	0.3312	0.2461	0.1959	0.1628	0.1395	0.1222	0.1087	0.0981	0.0893
10 ⁴	0.4982	0.3318	0.2488	0.1990	0.1658	0.1421	0.1243	0.1105	0.0995	0.0904
10 ⁵	0.4997	0.3329	0.2495	0.1996	0.1664	0.1426	0.1248	0.1109	0.0999	0.0908
10 ⁶	0.4999	0.3332	0.2498	0.1998	0.1665	0.1427	0.1248	0.1109	0.0998	0.0907
10 ⁷	0.4999	0.3333	0.2499	0.1999	0.1666	0.1428	0.1249	0.1110	0.0999	0.0908
$1/(k+1)$	0.5000	0.3333	0.2500	0.2000	0.1667	0.1428	0.1250	0.1111	0.0100	0.0909

表 1: 随机数序列 k 阶矩数值

再将 `difference.out` 中的数据整理列表, 表中数据为 $|\langle x^k \rangle - 1/(k+1)|$ 的值。观察可得: 偏差的数值总是小于 $1/\sqrt{N}$ 的数值, 这表明生成随机数的均匀性优良.

$N \backslash k$	1	2	3	4	5	6	7	8	9	10	$1/\sqrt{N}$
10 ²	1.427e-2	8.820e-3	2.557e-3	2.836e-3	7.240e-3	1.078e-2	1.360e-2	1.584e-2	1.760e-2	1.898e-2	1.000e-1
10 ³	2.281e-3	2.114e-3	3.823e-3	4.095e-3	3.788e-3	3.296e-3	2.783e-3	2.311e-3	1.898e-3	1.544e-3	3.162e-2
10 ⁴	1.778e-3	1.439e-3	1.199e-3	9.965e-4	8.312e-4	7.044e-4	6.111e-4	5.437e-4	4.949e-4	4.589e-4	1.000e-2
10 ⁵	2.840e-4	4.224e-4	4.013e-4	3.305e-4	2.561e-4	1.941e-4	1.474e-4	1.146e-4	9.270e-5	7.868e-5	3.162e-3
10 ⁶	2.948e-5	1.151e-4	1.442e-4	1.488e-4	1.453e-4	1.400e-4	1.348e-4	1.301e-4	1.260e-4	1.224e-4	1.000e-3
10 ⁷	1.856e-5	3.198e-5	3.946e-5	4.534e-5	5.016e-5	5.395e-5	5.678e-5	5.876e-5	6.000e-5	6.062e-5	3.162e-4

表 2: k 阶矩与 $1/(k+1)$ 偏差数值

(c) 自相关系数检验 2 维独立性

将输出文件 `indep.out` 中的数据整理如下，可观察出 $C(l)$ 的值随着数目 N 增大而逐渐减小，其值基本保持在预估值 $1/\sqrt{N}$ 的量级甚至更小，这表明在自相关系数检验下，16807 产生器产生的随机数独立性较好。

$C(l) \setminus N$	10^2	10^3	10^4	10^5	10^6	10^7
l						
1	5.914e-2	4.035e-2	3.132e-7	2.426e-3	2.709e-4	3.444e-4
2	9.297e-2	3.047e-2	1.175e-2	2.705e-3	1.720e-3	3.892e-5
3	4.510e-2	4.889e-3	5.929e-3	3.452e-3	8.768e-4	3.613e-4
4	4.958e-3	3.291e-2	7.180e-5	3.826e-3	3.790e-4	4.476e-5
$1/\sqrt{N}$	1.000e-1	3.162e-2	3.162e-2	3.162e-3	1.000e-3	3.162e-4

表 3: 自相关系数 $C(l)$ 数值

5 结论

本作业使用 Schrage 方法编写了随机数产生器程序，并对产生的随机数进行了均匀性和独立性检验，通过自己的程序实现证明了线性同余法随机数产生器 (LCG) 基本性质的优良，并可在日后的其他场合下随时调用。但不存在完美的随机数产生器，LCG 仍存在一定的缺点，在某些情况下仍需要使用更加复杂精巧的随机数产生器。