计算物理作业二

于浩然 PB19020634 2021.10.07

1 作业题目

用 16807 产生器测试随机数序列中满足关系 $X_{n-1} > X_{n+1} > X_n$ 的比重。讨论 Fibonacci 延迟测试器中出现这种关系的比重。

2 算法简介

线性同余法是最简单的生成均匀随机数的方法，其优势在于速度快，但是这样产生的随机数在序列相关性上较差。例如，在 32 位计算机上 m 可能取的最大值 2^{31}，在 3 维空间中连续使用随机数的话，点将分布在空间中最多达 $[3!(2^{31})]^{1/3} = 2344$ 个平面上。因此，当我们只关注体积中一小部分的时候，平面的规则离散性对计算结果有很大影响。

Fibonacci 延迟产生器是另一种类型的随机数产生器，其中有些特例可以用来消除同余法中的关联问题。其思想是用序列中的两个整数进行操作得到后续的整数 (名称源于 Fibonacci 数列 $1, 2, 3, 5, 8, 13, 21, 34, \cdots$，其中一个整数为前两个数之和)，表达式为：

$$I_n = I_{n-p} \otimes I_{n-q} \mathrm{mod} m \tag{1}$$

操作符 $\otimes$ 可以是加、减、乘、除、XOR(与或)。整数对 $[p, q]$ 表示延迟 $(p > q)$，取值一般通过统计检验后确定。Fibonacci 延迟产生器与 LCG 相比其周期非常长，在 32 位机上的最大周期为 $(2^p - 1)2^{31}$.

一种 Fibonacci 延迟器的例子便是 **Marsaglia** 产生器 (组合产生器，由两个不同的随机数产生器生成另一个随机数序列) 中第一个随机数产生器，即一种减法操作 Fibonacci 延迟产生器 (**后文简称为减法生成器**)：

$$x_n = \begin{cases} x_{n-p} - x_{n-q} & , if \geq 0 \\ x_{n-p} - x_{n-q} + 1 & , otherwise \end{cases} \tag{2}$$

若用 (1) 来表达，等价于 $0 \geq I_n < 1$，且 $m = 1$. 其中 $[p, q]$ 整数对选取 $[97, 33]$，因此该算法要求存储所有前面的 97 个随机数值。

为方便起见，我们将要检验的条件列出：

$$X_{n-1} > X_{n+1} > X_n \tag{3}$$

3 编程实现

用 Fortran90 进行编程。共调用 3 个子程序，各自功能与源代码如下：

- `SUBROUTIHNE Minus(N)`

  减法产生器需要 97 个种子来生成随机数，考虑到之前编写过 Schrage 方法 LCG 的子程序，直接调用该子程序，用种子 `In(0) = m - 1` 生成 10^2 个随机数作为种子来生成随机数。传入参数 `N` 决定随机数总数目为 10^N，并将得到的随机数序列存入文件 `minrand.out`.

```
SUBROUTINE Minus(N) !用Marsaglia生成器中的“减法生成器”生成随机数序列
    INTEGER(KIND=4) :: p = 97, q = 33, N, i
    REAL(KIND=8) :: x(10**N), z(100)
    CALL Schrage(2) !用16807生成器生成10^2个随机数作为“减法生成器”的种子
    OPEN (99, file='lcgrand.out')
    READ (99, *) z !首先将种子读取到数组z中
    CLOSE (99)
    DO i = 1, 97 !由种子生成前97个随机数
        x(i) = z(97 + i - p) - z(97 + i - q) !x的第一个数据由z(1)和z(65)生成
        IF(x(i) < 0) THEN
            x(i) = x(i) + real(1)
        END IF
    END DO
    DO i = 98, 10**N  !由生成的前97个随机数生成总数目为10^N的随机数序列
        x(i) = x(i - p) - x(i - q)
        IF(x(i) < 0) THEN
            x(i) = x(i) + real(1)
        END IF
    END DO
    OPEN (99, file='minrand.out') !将随机数按行存入文件
    DO i = 1, 10**N
        WRITE (99, *) x(i)
    END DO
    CLOSE (99)

END SUBROUTINE Minus
```

- SUBROUTINE Schrage(P)
  即为前一题所用的 16807 产生器，产生 10^P 个随机数写入文件 lcgrand.out.

```
SUBROUTINE Schrage(P) !Schrage随机数生成器子程序
    IMPLICIT NONE
    INTEGER :: N = 1, P
    INTEGER :: m = 2147483647, a = 16807, q = 127773, r = 2836, In(10**P)
    REAL(KIND=8) z(10**P)
    In(1) = m - 1
    z(1) = REAL(In(1))/m
    DO N = 1, 10**P - 1
        In(N + 1) = a*MOD(In(N), q) - r*INT(In(N)/q)
        IF (In(N + 1) < 0) THEN !若值小于零，按Schrage方法加m
        In(N + 1) = In(N + 1) + m
        END IF
        z(N + 1) = REAL(In(N + 1))/m !得到第N+1个随机数
    END DO
```

```
    OPEN (1, file='lcgrand.out') !每次运行子程序将覆盖随机数
    DO N = 1, 10**P !将随机数按行存入文件
        WRITE (1, *) z(N)
    END DO
    CLOSE (1)
END SUBROUTINE Schrage
```

- SUBROUTINE Test(N, Filename)
  传入随机数序列大小 10^N，以及通过前面两个程序生成的随机数文件名 Filename，读取序列加载到实型数组 z(10**N) 中，随后遍历每一个元素，验证是否满足我们验证的关系 (3)，最后得到满足条件的元素数目 Number，与总数 10^N 作商求出比值，写入文件 ratio.out.

```
SUBROUTINE Test(N, Filename) !检验随机数序列中出现关联比重的子程序
    INTEGER(KIND=4) :: Number = 0, i = 1
    CHARACTER(LEN=11) :: Filename
    REAL(KIND=8) :: x(10**N), r !定义要检验的随机数序列
    OPEN (11, file=Filename ) !打开相应文件读取随机数
    READ (11,*) x
    CLOSE (11)
    IF (x(10**N) > x(2) .AND. x(2) > x(1)) THEN !用x(10**N)代替x(0)
        Number = Number + 1 !第1个随机数和最后一个随机数中最多只能有一个
            满足我们讨论的关系
    ELSE IF (x(10**N - 1) > x(1) .AND. x(1) > x(10**N)) THEN !用x(1)代替x
        (10**N + 1)
        Number = Number + 1
    END IF
    DO i = 2, 10**N - 1
        IF(x(i - 1) > x(i + 1) .AND. x(i + 1) > x(i)) THEN
            Number = Number + 1 !每当检验满足关系，累加Number
        END IF
    END DO
    r = real(Number) / 10**N !计算出现关联性的比重
    OPEN (1, ACCESS='append', file='ratio.out')
    WRITE (1, *) r
    CLOSE(1)
END SUBROUTINE Test
```

在主程序中使用 DO 循环结构，调用上述子程序生成不同大小的随机数序列并对其进行性质 (3) 的检验.

```
PROGRAM MAIN
    IMPLICIT NONE
    INTEGER :: N = 1 !用16807生成器生成100个随机数作为“减法生成器”的种子
    !PRINT *, "Input required quantity of random numbers(10^N)"
    !READ (*,*) N
    DO N = 2 , 7
        CALL Minus(N)
        CALL Schrage(N)
        CALL Test(N, 'minrand.out') !分别传入两个文件名作为实参来对其进行测试
        CALL Test(N, 'lcgrand.out')
    END DO
END PROGRAM MAIN
```

4 计算结果

两种产生器出现关系的比重：

分别读取 ratio.out 中偶数位次的数值 (即 16807 产生器得到的数值) 以及奇数位次的数值 (即减数产生器/Fibonacci 延迟产生器得到的数值)，整理列表展示如下：

N / RNG type	10^2	10^3	10^4	10^5	10^6	10^7
16807 RNG	0.30000	0.38300	0.41140	0.24127	0.19423	0.18636
Fibonacci RNG	0.15000	0.22300	0.24560	0.07597	0.02761	0.01977

表 1: 两种产生器出现关系的比重比较

从表中很容易看出：

- Fibonacci RNG 所产生的随机数序列中出现关系 (3) 的比重明显小于 16807 RNG.
- 随着生成随机数数目增多，出现关系的比重都会下降：但当数目从 10^4 变为 10^5 时，Fibonacci RNG 出现关系的比重显著下降了一个量级，并且在数目继续增大时其值与 16807 RNG 的差距进一步增大.

由此我们得出结论：Fibonacci 延迟产生器在序列相关性上的性质优于传统的 16807 线性同余产生器，当生成随机数序列较大时优势更加明显.

5 总结

本作业中我们编写了“减法生成器”——一种 Fibonacci 延迟器的子程序，可用于生成序列相关性比线性同余产生器更弱的随机数序列；对两种不同产生器我们分别测试了性质，并比较出了性质的优劣，这可以启发我们确定选取随机数产生器的标准. 本题有理论概率值，讨论结果是 Fibonacci 延迟器效果更差.