# 数值计算方法：原理、算法和应用
# Numerical Methods: Principles, Algorithms and Applications

授课教师：周铁

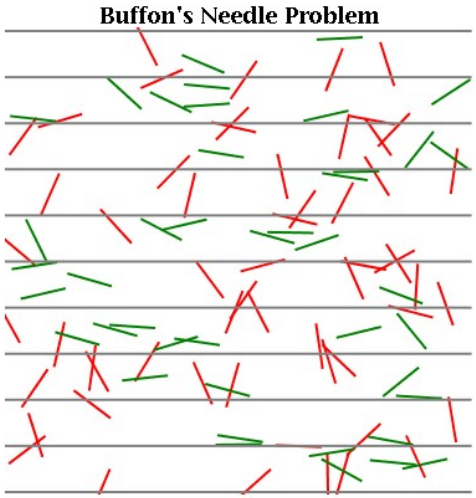
北京大学数学科学学院

2021 年 11 月 3 日

1. 随机模拟
   - Buffon's needle problem
   - 随机模拟：$\pi$ 的计算
   - Monte Carlo 数值积分
   - 随机数的产生

# Buffon's needle problem

Buffon(Georges-Louis Leclerc, Comte de Buffon)，1777 年发表，微积分，几何概率.




设针长为 $l$，线距为 $d$，并且设 $l < d$，随机地投一根针.

针与某一条平行直线相交的概率 $p = \dfrac{2l}{\pi d} \quad \Longrightarrow \quad \pi = \dfrac{2l}{pd}$

重复做 $n$ 次随机投针试验，记录到 $m$ 次针与平行直线相交，

$$p \approx \frac{m}{n} \quad \Longrightarrow \quad \pi \approx \frac{2nl}{md}$$

- 随机模拟方法：对一个问题进行**概率建模**，然后对概率模型进行**随机抽样**，从而得到原问题数值解的方法.
- 最早有文献记载的随机模拟方法大概就是 Buffon's needle problem.
- N. Metropolis, S. Ulam，J. von Neumann 等人把这种方法称为"**Monte Carlo 方法**".
- Monte Carlo 方法真正用于科学计算开始于 1940 年代末年，因为从那时开始才有电子计算机.

`https://en.wikipedia.org/wiki/Monte_Carlo_method:`

*The modern version of the Monte Carlo method was invented in the late 1940s by Stanislaw Ulam, while he was working on nuclear weapons projects at the Los Alamos National Laboratory. Immediately after Ulam's breakthrough, John von Neumann understood its importance and programmed the ENIAC computer to carry out Monte Carlo calculations.*

Monte Carlo 方法大概分成两类：

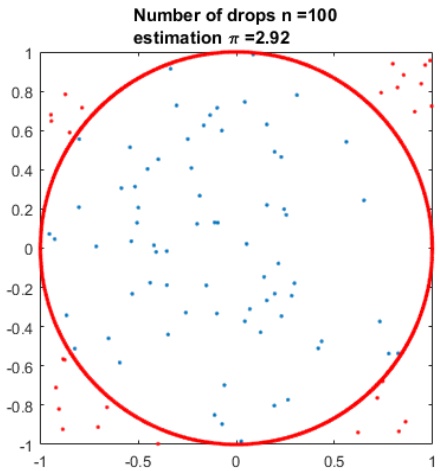
- 一类是借助计算机的运算能力直接模拟问题本身的随机现象. 例如核反应堆里的中子传输问题，中子与原子核相互作用是微观现象，只知道它们相互作用发生的概率，无法获得中子与原子核作用时的位置以及裂变产生的新中子的速度. Monte Carlo 方法使用随机抽样得到裂变位置和速度来模拟大量中子的行为，并采用统计方法获得中子传输的规律.
- 另一类是把要求解的确定型问题转化为某个随机变量的数字特征，比如随机事件出现的概率，或者随机变量的期望. 通过随机抽样的方法，以随机事件出现的频率估计其概率，或者以抽样的数字特征近似随机变量的数字特征，并将其作为问题的解. 这种方法多用于求解自由度很大的问题或者高维空间中的积分问题.

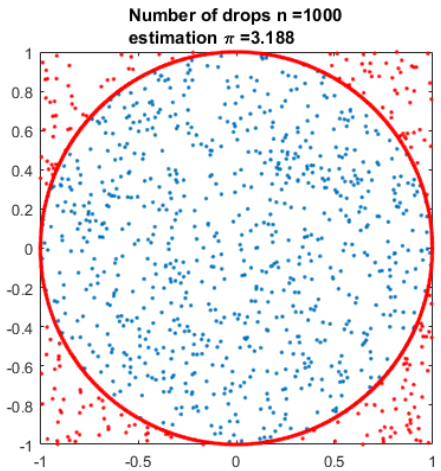
例如：要计算一个复杂图形的面积，图形的不规则程度和分析性计算 (比如积分) 的复杂程度是成正比的.

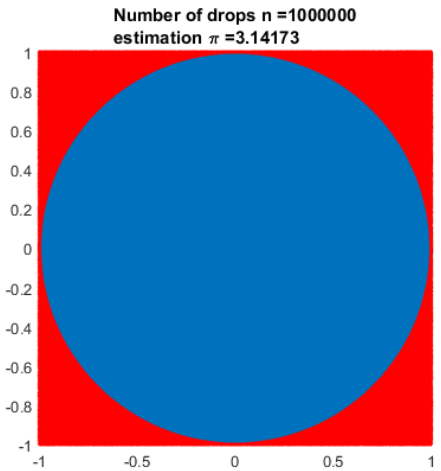
Monte Carlo 方法基于这样的思想：先把图形看成一个规则区域的子集，假想你有一袋已知数目的豆子，把袋中每个豆子随机地抛掷到包含复杂图形的规则区域上，并统计落在这个复杂图形之中有多少颗豆子，这个豆子的数目就对应了图形的面积. 当你的豆子尺寸越小，数量越多的时候，结果就越精确.

借助计算机程序可以快速产生大量规则区域中的均匀分布坐标点，统计出落在复杂图形内的那些点的个数，通过这个数占总点数的比例和规则区域的面积就可以近似求出复杂图形面积.

# 随机模拟：$\pi$ 的计算




- 一次实验：在 $\mathbb{R}^2$ 的正方形区域 $[-1,1]\times[-1,1]$ 内 (均匀地) 随机取一个点 $(x,y)$，它落在单位圆内的概率为：

$$p=\frac{\text{圆面积}}{\text{正方形面积}}=\frac{\pi}{4}$$

- 多次重复实验：(均匀地) 随机撒 $n$ 个点，其中 $m$ 个点落在圆内，

$$p\approx\frac{m}{n}\quad\Longrightarrow\quad\pi\approx\frac{4m}{n}$$

- 判断一个点 $(x, y)$ 是否在闭单位圆内只要看 $x^2 + y^2 \leq 1$ 是否成立.
- 这种方法的理论基础是 “**大数定律**” .
  引进 $n$ 个随机变量 $X_i,\ i = 1, 2, \ldots, n$，用它们描述取点情况，

$$X_i = \begin{cases} 1, & \text{点落圆内}, \\ 0, & \text{点落圆外}, \end{cases} \quad X_i \sim \begin{pmatrix} 0 & 1 \\ 1 - \pi/4 & \pi/4 \end{pmatrix}, \quad E(X_i) = \pi/4$$

显然，重复实验 $n$ 次后，落在圆内的点数 $m$ 为

$$m = X_1 + X_2 + \cdots + X_n$$

由大数定律，对任意 $\epsilon > 0$，

$$\lim_{n\to\infty} P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \frac{\pi}{4}\right| \geq \epsilon\right) = \lim_{n\to\infty} P\left(\left|\frac{m}{n} - \frac{\pi}{4}\right| \geq \epsilon\right) = 0$$

$$\lim_{n\to\infty} P\left(\left|\frac{m}{n} - \frac{\pi}{4}\right| < \epsilon\right) = 1, \quad \text{当 } n \text{ 很大时，} \frac{m}{n} \approx \frac{\pi}{4} \text{ 是大概率事件}$$

还可以用“**中心极限定理**”分析上述方法的可靠性.
设 $X_1, X_2, \ldots, X_{2n}$ 是相互独立同分布的随机变量，$X_i \sim U[-1, 1]$.
对 $k = 1, 2, \ldots, n$，再定义随机变量 $Y_k$，

$$Y_k = \begin{cases} 4, & X_{2k-1}^2 + X_{2k}^2 \le 1 \\ 0, & \text{otherwise} \end{cases}, \quad Y_k \sim \begin{pmatrix} 4 & 0 \\ \pi/4 & 1 - \pi/4 \end{pmatrix}, \quad E(Y_k) = \pi$$

记 $\overline{Y} = \dfrac{Y_1 + Y_2 + \cdots + Y_n}{n}$，于是有

$$E(\overline{Y}) = E\left(\frac{Y_1 + Y_2 + \cdots + Y_n}{n}\right) = \frac{E(Y_1) + E(Y_2) + \cdots + E(Y_n)}{n} = \pi$$

再看方差 $Var(\overline{Y})$，首先

$$Var(Y_k) = E(Y_k^2) - E(Y_k)^2 = \pi(4 - \pi), \quad k = 1, 2, \ldots, n$$

由于 $Y_k$ 相互独立，于是就有

$$Var(\overline{Y}) = \frac{\sum_{k=1}^{n} Var(Y_k)}{n^2} = \frac{\pi(4 - \pi)}{n}$$

由中心极限定理，用符号 $\dot{\sim}$ 表示近似服从某种分布，则有

$$\overline{Y} \dot{\sim} N(\mu, \sigma^2) = N\left(\pi, \frac{\pi(4-\pi)}{n}\right).$$

可以估计当 $n = 1000$ 时，概率 $P(|\overline{Y} - \pi| < 0.1)$ 有多大.

$$\frac{\overline{Y} - \pi}{\sqrt{\frac{\pi(4-\pi)}{1000}}} = \frac{10\sqrt{10}(\overline{Y} - \pi)}{\sqrt{\pi(4-\pi)}} \text{ 近似} \sim N(0,1)$$

$$\begin{aligned} P(|\overline{Y} - \pi| < 0.1) &= P\left(\left|\frac{10\sqrt{10}(\overline{Y} - \pi)}{\sqrt{\pi(4-\pi)}}\right| < \frac{\sqrt{10}}{\sqrt{\pi(4-\pi)}}\right) \\ &\approx \Phi\left(\frac{\sqrt{10}}{\sqrt{\pi(4-\pi)}}\right) - \Phi\left(\frac{-\sqrt{10}}{\sqrt{\pi(4-\pi)}}\right) \\ &= 2\Phi\left(\frac{\sqrt{10}}{\sqrt{\pi(4-\pi)}}\right) - 1 \approx 0.945852461229737 \end{aligned}$$

## Chebyshev 不等式

如果随机变量 $X$ 满足：$E(X) < \infty$, $Var(X) < \infty$，则对 $\forall \epsilon > 0$，都有

$$P(|X - E(X)| \geq \epsilon) \leq \frac{Var(X)}{\epsilon^2}$$

回到前面的问题，问当 $n$ 多大时可以保证

$$P(|\overline{Y} - \pi| < 0.1) \geq 0.95$$

由 Chebyshev 不等式，

$$P(|\overline{Y} - \pi| < 0.1) = 1 - P(|\overline{Y} - E(\overline{Y})| \geq 0.1) \geq 1 - \frac{Var(\overline{Y})}{0.01}$$

$$1 - \frac{Var(\overline{Y})}{0.01} = 1 - \frac{100\pi(4-\pi)}{n} \geq 0.95$$

$$n \geq \frac{100\pi(4-\pi)}{0.05} \approx 5394$$

用随机变量 $\overline{Y}$ 逼近 $\pi$ 的误差怎么估计？一种自然的想法是用标准差

$$\sqrt{E(|\overline{Y}-\pi|^2)}=\sqrt{Var(\overline{Y})}=\sqrt{\frac{\pi(4-\pi)}{n}}$$

来度量这个误差. 在这个意义下，可以说 Monte Carlo 方法的误差阶是 $\mathcal{O}(1/\sqrt{n})$，其中 $n$ 是随机采样的次数.

而且不管 $n$ 多大，因为 $\overline{Y}$ 是一个随机变量，都不能保证

$$|\overline{Y}-\pi|\leq\frac{C}{\sqrt{n}}$$

一定成立，只能说“这个不等式成立是一个大概率事件”.

跟我们前面见过的数值计算方法相比，可以说：

**Monte Carlo 方法收敛很慢！**

# Monte Carlo 数值积分

以一元定积分

$$I(f) = \int_a^b f(x)\,dx, \quad \text{其中 } -\infty < a < b < +\infty \tag{3.1}$$

为例，前面已经介绍了很多数值求积方法. 那里的所有求积公式都表示为被积函数在某些点的函数值的加权平均. 例如，复合梯形求积公式为

$$I(f) \approx T(h) = \frac{h}{2}f(x_0) + \sum_{i=1}^{n-1} hf(x_i) + \frac{h}{2}f(x_n),$$

其中 $h = (b-a)/n$, $x_i = a + ih$ $(i = 0, 1, 2, \ldots, n)$.

Monte Carlo 数值求积方法的基本思想是：把被积函数的自变量 $x$ 看成 $[a, b]$ 区间上的均匀分布随机变量 $X$，即

$$X \sim U[a, b], \quad \text{其 pdf 为 } p(x) = \begin{cases} 1/(b-a), & x \in [a, b] \\ 0, & x \in (-\infty, a) \cup (b, +\infty) \end{cases}$$

于是 $f(X)$ 也是随机变量，而且**定积分**$(3.1)$**就是** $f(X)$ **的数学期望**

$$I(f)/(b-a) = \int_{-\infty}^{\infty} f(x)p(x)\,dx = E(f(X))$$

为了近似计算 $E(f(X))$，取相互独立的随机变量序列：

$$X_1, X_2, \ldots, X_n, \quad X_i \sim U[a,b].$$

通过随机变量的函数 $f(X)$ 又得到相互独立且同分布的随机变量序列：

$$f(X_1), f(X_2), \ldots, f(X_n).$$

而且有

$$E(f(X_i)) = E(f(X)) = \frac{1}{b-a}\int_a^b f(x)\,dx = I(f)/(b-a), \quad 1 \le i \le n.$$

根据**大数定律**，对 $\forall\,\epsilon > 0$，有“依概率收敛”结果：

$$\lim_{n\to\infty} P\left(\left|\frac{(b-a)}{n}\sum_{i=1}^{n} f(X_i) - I(f)\right| > \epsilon\right) = 0.$$

从而得到数值积分公式

$$I_n(f) \triangleq \frac{(b-a)}{n} \sum_{i=1}^{n} f(X_i) \approx I(f).$$

只要对的随机变量 $X \sim U[a,b]$ 做 $n$ 次独立抽样，得到 $[a,b]$ 区间中的具体数值 $x_1, x_2, \ldots, x_n$，就可得到积分 $I(f)$ 的一个近似值

$$\frac{(b-a)}{n} \sum_{i=1}^{n} f(x_i) \approx I(f).$$

上述方法完全可以推广到高维重积分的计算.

设 $d \in \mathbb{N}_+$，$x \in D = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$，$f(x)$ 为 $D$ 上可积函数,

$$J(f) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_d}^{b_d} f(x_1, x_2, \ldots, x_d)\, dx_1\, dx_2 \cdots dx_d = \int_D f(x)\, dx$$

引进 $d$ 个独立的，均匀分布随机变量 $X_j \sim U[a_j, b_j], j = 1, 2, \ldots, d$，则

$$J(f)/|D| = \int_{\mathbb{R}^d} f(x) p_1(x_1) \cdots p_d(x_d)\, dx = E(f(X_1, \ldots, X_d))$$

其中 $p_j(x_j)$ 为 $X_j$ 的概率密度函数 (pdf)

$$p_j(x) = \begin{cases} 1/(b_j - a_j), & x_j \in [a_j, b_j] \\ 0, & x_j \in \mathbb{R} \setminus [a_j, b_j] \end{cases},$$

对随机向量 $X = (X_1, X_2, \ldots, X_d)$ 进行 $n$ 次独立抽样，得到

$$X^{(i)} = (X_1^{(i)}, X_2^{(i)}, \ldots, X_d^{(i)}), \quad i = 1, 2, \ldots, n, \quad X_j^{(i)} \sim U[a, b].$$

于是 $d$ 重积分 $J(f)$ 的近似值为

$$J(f) \approx J_n(f) \triangleq \frac{|D|}{n} \sum_{i=1}^{n} f\left(X^{(i)}\right), \quad |D| = \prod_{j=1}^{d} (b_j - a_j).$$

由于 $X^{(i)}$ 是 i.i.d. 的随机向量，$J_n(f)$ 为一个随机变量，其均值为

$$E(J_n(f)) = \frac{|D|}{n} \sum_{i=1}^{n} E\left(f\left(X^{(i)}\right)\right) = J(f),$$

方差为

$$Var(J_n(f)) = \frac{|D|^2}{n^2} \sum_{i=1}^{n} Var\left(f\left(X^{(i)}\right)\right) = \frac{|D|^2}{n} Var(f(X))$$

即收敛速率为

$$|J_n(f) - J(f)| = \mathcal{O}(1/\sqrt{n})$$

注意采样点个数 $n$ 与维数 $d$ 无关!

一个五重积分的例子：

$$I=\int_0^5\int_0^4\int_0^1\int_0^2\int_0^3 \sqrt[3]{v}\sqrt{w}x^2y^3z\ \mathrm{d}z\,\mathrm{d}y\,\mathrm{d}x\,\mathrm{d}w\,\mathrm{d}v.$$

取自《Scientific Computing with MATLAB》, page: 112,
作者：Dingyu Xue, Yangquan Chen,
CRC Press, 2016.

```
1  syms x y z w v;
2  F=v^(1/3)*sqrt(w)*x^2*y^3*z;
3  I=int(int(int(int(int(F,z,0,3),y,0,2),x,0,1),w,0,4),v,0,5)
4  vpa(I)
5  120*nthroot(5,3)
6  MonteCarlo_interal_5D
```

正如前面提到的，Monte Carlo 积分方法的 $1/2$ 阶收敛速度太慢了！在维数较低 $(d \leq 4)$ 的时候，Monte Carlo 积分方法没有实际意义.

假设用复合中点公式计算一个在 $D = [0,1]^d$ 区域内的 $d$ 重积分，在积分区间 $D$ 中总共取 $n$ 个点作均匀剖分，则每个坐标轴上的网格步长为

$$h = \mathcal{O}(n^{-\frac{1}{d}}).$$

此时复合中点公式的计算精度为

$$|T(h) - J(f)| = \mathcal{O}(n^{-\frac{2}{d}}).$$

如果用 Monte Carlo 方法，则计算精度为 $\mathcal{O}(n^{-\frac{1}{2}})$. 当维数 $d > 4$ 时，Monte Carlo 方法就比复合中点公式收敛的快，如果维数 $d \leq 3$，一般还是应该使用确定型数值积分方法.

还需要强调的是，Monte Carlo 方法的计算效果往往依赖于一个随机数生成算法.

# 随机数的产生

设随机变量 $X$ 的分布函数为 $F(x)$，且 $\{X_i,\ i=1,2,\dots,\}$ 是 i.i.d. 的，则 $\{X_i,\ i=1,2,\dots,\}$ 的一次观测值 $\{x_i,\ i=1,2,\dots,\}$ 称为概率分布 $F(x)$ 的随机数序列，简称随机数.

可以用物理方法得到真正的均匀分布随机数，比如抛硬币，掷骰子，抽签，摇号等等. 这些方法产生的随机数太少，速度太慢，不能满足随机模拟的需要.
也可以预先产生大量真实随机数并存入计算机硬盘，进行随机模拟时再读到内存里使用，这种方法因为读取速度太慢，现在也很少使用.

需要产生某种特定分布的随机数时，一般先产生均匀分布的随机数，然后再经过一些数学变换转化成其它分布的随机数.

# 伪随机数

现在的主流方法是使用计算机**实时产生**的“伪随机数 (pseudo-random number)”代替真正的随机数，它们由确定型算法 (递推公式) 产生，并且看上去与真正服从概率分布 $F(x)$ 的随机数没有区别，同时最好满足下列条件：

1. 随机性：能通过随机性的统计学检验
2. 长周期：循环长度越大越好
3. 生成效率高：计算复杂度和存储复杂度低
4. 可重复：同样初始条件生成相同序列
5. 可移植：在不同计算机系统上可工作并产生同样序列

# 平方取中 (midsquare) 法

在电子计算机刚刚问世时，John von Neumann 等人为进行随机模拟提出了“平方取中 (midsquare)”法.
例如，首先取一给定的四位数 3333，将其平方，得到 11108889，取出中间的一个数 1088，再将其平方得到 1183744，循环这个过程即得数列：

$$\begin{aligned} x_0 = 3333 &\quad \to \quad 11108889 \\ x_1 = 1088 &\quad \to \quad 1183744 \\ x_2 = 8374 &\quad \to \quad 70123876 \\ x_3 = 1238 &\quad \to \quad \cdots\cdots \end{aligned}$$

显然，这一方法最大循环长度仅为 $10^4$，而且其均匀性不理想，很快就收敛到零，但是这一算法在当时进行的原子核反应数值模拟中起到了关键作用.

# 线性同余生成器 (Linear Congruential Generator)

线性同余生成器的一般形式如下

$$x_{k+1} = (ax_k + c) \pmod{m}$$

其中 $a, m$ 为正整数，$c$ 非负整数. 显然还需要事先给定一个非负整数 $x_0$ 作为初值（“种子”）.
这种方法产生的 $x_k \in \{0, 1, \ldots, m-1\}$，其最大循环长度不会超过 $m$.
例 1:

$$x_{k+1} = (7x_k + 7) \pmod{10}$$

初值 $x_0 = 7$，产生的序列为 $\{7, 6, 9, 0, 7, 6, 9, 0, 7, \ldots\}$，周期为 $4 < 10$.
例 2:

$$x_{k+1} = (5x_k + 1) \pmod{10}$$

初值 $x_0 = 1$，产生的序列为 $\{1, 6, 1, 6, 1, \ldots\}$，周期为 $2 < 10$.
例 3:

$$x_{k+1} = (5x_k + 1) \pmod{8}$$

初值 $x_0 = 1$，产生的序列为 $\{1, 6, 7, 4, 7, 5, 2, 3, 0, 1, 6, 7 \ldots\}$，周期为 $8$.

线性同余生成器只有适当选择参数 $a, b, m$ 才能具有满周期和比较高的随机性.

**定理**

设 $c > 0$ 并且满足

1. $c$ 与 $m$ 互素
2. 对 $m$ 的任一个素因子 $p$，$a - 1$ 被 $p$ 整除
3. 如果 $4$ 是 $m$ 的因子，则 $a - 1$ 被 $4$ 整除

则线性同余生成器达到满周期.

根据这个定理，取 $m = 2^L, a = 4u + 1, c = 2v + 1$，其中 $L, u, v \in \mathbb{N}_+$，初值 $x_0 \in \mathbb{N}_+$，则周期为 $2^L - 1$.

1970 年代的一些被广泛使用的线性同余生成器在生成多维随机数时相关性很高，分布不均匀，造成当时很多随机模拟结果不可靠.

线性同余生成器产生介于 $0$ 和 $m-1$ 之间的均匀分布随机非负整数. 为了产生服从 $U[0,1)$ 的浮点随机数，只要把数列再除以 $m$(浮点除法). 可靠的随机数列应该服从均匀分布并且互相独立. 即数列 $\{x_k\}$ 应该是 i.i.d. 随机变量序列 $\{X_k\}$ 的采样. 在满周期时，应该有

$$P\{X_k = i\} = \frac{1}{m}, \quad i = 0, 1, \ldots, m-1.$$

此时

$$E(X_k) = \sum_{i=0}^{m-1} \frac{i}{m} = \frac{m-1}{2}$$

$$Var(X_k) = E(X_k^2) - (E(X_k))^2 = \sum_{i=0}^{m-1} \frac{i^2}{m} - \frac{(m-1)^2}{4} = \frac{m^2-1}{12}$$

记 $R_k = X_k/m$，则 $m$ 很大时有

$$E(R_k) = \frac{1}{2} - \frac{1}{2m} \approx \frac{1}{2}, \quad Var(R_k) = \frac{1}{12} - \frac{1}{12m^2} \approx \frac{1}{12}$$

伪随机数序列需要有很好的随机性和独立性，数字的出现不应有任何规律，并且任何两项不应该相关.
由于产生自确定型的公式，所以真正的独立是做不到的，只能要求它们相关性比较弱. 在满周期的情况下，序列前后两项的自相关系数有近似公式

$$\rho(1) \approx \frac{1}{a} - \frac{6c}{am}\left(1 - \frac{c}{m}\right)$$

由此看出只有 $a$ 比较大时相关性才能弱.

例 4:

$$x_{k+1} = (314159269x_k + 453806245) \pmod{2^{31}}$$

周期为 $2^{31}$(满周期)，统计性质也比较好.

# 线性反馈移位寄存器 (linear feedback shift register, LFSR)

如果用前面介绍的方法产生多维随机数，往往相关性大而且分布不均匀，并且周期受到计算机字长的限制. LFSR 才是现在的主流方法.
LFSR 本来是一种电路，其数学模型就是利用下面多重递推公式产生二进制数列 $\{a_k\}$.

$$a_k = (c_p a_{k-p} + c_{p-1} a_{k-p+1} + \cdots + c_1 a_{k-1}) (\mathrm{mod}\, 2), \quad k = 1, 2, \ldots$$

其中 $p$ 是给定正整数，系数 $c_1, c_2, \ldots, c_p \in \{0, 1\}$ 且只有两个取 1 其它全取 0，还需要事先给定初值 $a_{-p+1}, \ldots a_0$.
如果把系数为 0 的都去掉，设 $c_p = c_{p-q} = 1$，则有

$$a_k = (a_{k-p} + a_{k-p+q}) (\mathrm{mod}\, 2) = \begin{cases} 0, & \text{if } a_{k-p} = a_{k-p+q} \\ 1, & \text{if } a_{k-p} \neq a_{k-p+q} \end{cases}$$

这就是异或运算

$$a_k = a_{k-p} \oplus a_{k-p+q}, \quad k = 1, 2, \ldots$$

用上述方法产生二进制数列 $\{a_k\}$ 后，截取其中连续的 $L$ 位构成一个 $L$ 位二进制正整数

$$x_1 = a_1 a_2 \cdots a_L$$

再截取连续的 $L$ 位构成下一个 $L$ 位二进制正整数

$$x_2 = a_{L+1} a_{L+2} \cdots a_{2L}$$

依此类推产生 $\{x_k\}$.
记 $r_k = x_k/2^L$，则 $\{r_k\}$ 就是 LFSR 方法产生的 $[0,1]$ 区间均匀随机数.
可以证明如果 $L$ 与 $2^p - 1$ 互素，则 $\{r_k\}$ 就满足

1. 均值近似为 $1/2$,
2. 方差近似为 $1/12$,
3. 自相关系数近似为 $0$,
4. 当 $dL \le p$ 且 $dL$ 与 $2^p - 1$ 互素时，可产生 $d$ 维均匀随机数.

# 旋转 LFSR 生成器

旋转反馈移位寄存器法是 $1992$ 年提出的方法. 其数学形式为

$$b_{k+p} = b_{k+q} \,.\oplus\, b_k\, A$$

其中 $b_k$ 为 $d$ 维二进制行向量，$.\oplus$ 表示两个向量对应元素的异或运算，$A$ 是一个元素为 $0, 1$ 的 $d$ 阶方阵.

# Mersenne 旋转 (Mersenne twister) 算法

Mersenne 数是指形如 $2^n - 1$ 的数，如果一个 Mersenne 数是素数那么它称为 Mersenne 素数. 目前已知最大的梅森素数是 $2^{77232917} - 1$.

Mersenne 旋转算法是 1997 年提出的. 它是基于旋转 LFSR 生成器的方法，其周期是一个 Mersenne 素数，故而得名. 它可以快速产生高质量的伪随机数，克服了原有随机数发生器的很多缺陷.
Mersenne 旋转算法目前是 C++, R, Python, Matlab 等编程语言的默认伪随机数产生器.
最为广泛使用的 Mersenne 旋转算法叫 MT19937，它可以产生 32 位整数序列，并具有以下的特点:

1. 有 $2^{19937} - 1$ 长的最大循环周期.
2. 对 $1 \leq d \leq 623$ 都服从 $U[0,1]^d$.
3. 在所有伪随机数生成器中是最快的.
4. 占用内存比较多.

# 参考文献


1. Michael T. Heath
   Scientific Computing An Introductory Survey (2nd Edition)
   清华大学出版社, 2001.
2. 肖柳青，周石朋
   随机模拟方法与应用
   北京大学出版社, 上海交通大学出版社, 2019.
3. Jun S. Liu
   Monte Carlo Strategies in Scientific Computing
   Springer, 2001.
4. 李东风
   统计计算
   高等教育出版社, 2017.
5. 杨雪
   关于伪随机数发生器的综述
   吉林大学硕士学位论文，2006.