2.2 随机数与伪随机数

数列可以分为三种不同的类型：

真随机数列，准随机数列，伪随机数列

一、 真随机数

- 真随机数数列是不可预计的，因而也不可能重复产生两个相同的真随机数数列。
- 真随机数只能用某些随机物理过程来产生。例如：放射性衰变、电子设备的热噪音、宇宙射线的触发时间等等。
- 如果采用随机物理过程来产生蒙特卡洛计算用的随机数，理论上不存在什么问题。但在实际应用时，要做出速度很快（例如每秒产生上百个浮点数），而又准确的随机数物理过程产生器是非常困难的。

弗里吉雷欧(Frigerio)等人的真随机数获取：

用一个α粒子放射源和一个高分辨率的计数器做成的装置，在20毫秒时间内平均记录了24.315个α粒子。当计数为偶数时，便在磁带上记录二进制的“1”。

这个装置每小时可以产生大约6000个31比特(bits)的真随机数。这些数被存储在磁带上，并通过了一系列的“随机数”检验后用于蒙特卡洛计算当中。

消除奇数计数的几率并不精确等于1/2所引起的偏差的

处理方法：

利用上面介绍的装置得到的“0”或者“1”的真随机数序列中，0 和 1 出现的几率 P(0)和 P(1)可能并不精确等于 1/2。

我们从原始的真随机数序列出发，将序列中的二进制数依次成对组合；如果这组中的两个数相同，则舍去这两个数；如果这组中的两个数不相同，则保留第二个二进制数而丢弃第一个数。

这样构成的一个新序列可以保证：在原始序列中的数是相互独立的情况下，“0”和“1”出现的概率相等。

这一点可以从如下的计算中看出：“0”出现在新序列中的概率为 $P'(0)=P(1)P(0)$。这是因为新序列中的“0”只能在原始序列中“1”后面跟着“0”时才出现。同样“1”在新序列中出现的概率 $P'(1)=P(0)P(1)$。因而无论 $p(0)$ 和 $p(1)$ 等于什么值，$p'(0)$ 和 $p'(1)$ 都相等。由于在构成新序列时，舍去了一组数的几率为 $P^2(0)+P^2(1)$，因而 $P'(0)+P'(1)$ 不等于 1，而小于或等于 1/2。在这种方法中，对两个数不相同的一组数至少要丢掉一个二进制数。很明显，它的产生效率为 $P(0)P(1)=P(1-P)$，其中 p 为 $p(0)$ 或 $p(1)$。其产生效率的最大值为 25 %。

巴夫昂投针实验在真随机数产生器中由于物理偏差所引起的问题：

（1）在投针实验中平行线间间距必须保证为一个常数

值，并在所要求的误差范围内与针长相等。如果我们仅要求π值的一至二位有效数字，这个要求是不难满足做到的，但是如果要求更多位的有效数字，这就比较困难了。

（2）正确地判断临界状态下的针与平行线的相交也非易事。第三，我们还必须保证针的投掷位置和角度的分布是均匀分布的。为保证角度分布的均匀性，可以在投针的时候，让针迅速旋转，并采用非常平的、摩擦系数是各向同性的桌面。

（3）投针位置的分布决不是均匀分布的，而是在投掷目标点周围服从高斯分布。在实际应用中，我们必须由实验来决定这一分布宽度，并且要对它引起的偏差做类似于前面所述的由弗里吉雷欧等人所做的复杂修正。

二、准随机数

准随机数序列并不具有随机性质，仅仅是它用来处理问题时能够得到正确结果。

准随机数概念是来自如下的事实：对伪随机数来说，要实现其严格数学意义上的随机性，在理论上是不可能的，在实际应用中也没有这个必要。关键是要保证“随机”数数列具有能产生出所需要的结果的必要特性。例如，在多重积分和大多数模拟研究中，多维空间的每个点或模拟事例被认为是相互独立的，而这些点或事例的顺序则似乎并不重要。因

而我们可以在大多数运算中，放心地置随机性的概念于不顾。同样，我们也可以不考虑对某些分布均匀性的涨落程度。事实上在许多情况下，超均匀的分布比真随机数的均匀分布更合乎实际需要。

事实上，准蒙特卡洛是将蒙特卡洛方法处理问题的维数向高维扩展的方法。由此可见准蒙特卡洛方法的理论与真蒙特卡洛的理论很接近。

三、 伪随机数

实际应用的随机数通常都是通过某些数学公式计算而产生的伪随机数。这样的伪随机数从数学意义上讲已经一点不是随机的了。但是，只要伪随机数能够通过随机数的一系列的统计检验，我们就可以把它当作真随机数而放心地使用。这样我们就可以很经济地、重复地产生出随机数。

对物理问题的计算机模拟所需要的伪随机数应当满足如下的标准：

理论上，要求伪随机数产生器具备以下特征：良好的统计分布特性、高效率的伪随机数产生、伪随机数产生的循环周期长，产生程序可移植性好和伪随机数可以重复产生。其中满足良好的统计性质是最重要的。

然而实际使用的伪随机数产生程序还没有一个是十全十美的。因此我们要求产生出的伪随机数应当能通过尽可能

多的统计检验，以便人们放心地使用。

1. 伪随机数的产生方法

伪随机数产生器产生的实际上是伪随机数序列。

最基本的产生器是均匀分布的伪随机数产生器。

最早的伪随机数产生器可能是冯•诺曼平方取中法：该方法首先给出一个 2r 位的数，取它的中间的 r 位数码作为第一个伪随机数；然后将第一个伪随机数平方构成一个新的 2r 位数，再取中间的 r 位数作为第二个伪随机数...。如此循环便得到一个伪随机数序列。类似上述方法，利用十进制公式表示 2r 位数 x_n 的递推公式。

$$\begin{aligned} x_{n+1} &= \left[10^{-r} x_n^2\right] \left(\mathrm{mod} 10^{2r}\right) \\ \xi_n &= x_n / 10^{2r} \end{aligned},$$

这样得到的 $\{\xi_i\}$ 伪随机数序列是分布在[0，1]上的。相应的二进制递推公式为(x_n 为 2r 位二进制数)：

$$\begin{aligned} x_{n+1} &= \left[2^{-r} x_n^2\right] \left(\mathrm{mod}\, 2^{2r}\right) \\ \xi_n &= x_n / 2^{2r} \end{aligned},$$

但是这种方法不是很好，现在已很少使用。这主要是因为该方法产生的数列具有周期性，有些数(如零)甚至会紧接着重复出现。

实际使用的伪随机数产生器常常比平方取中法简单。如今比较流行，并用得最多的是同余产生器。我们通过如下的线性同余关系式来产生数列。

$$x_{n+1} = (ax_n + c)(\mathrm{mod}\, m)$$
$$\xi_n = x_n / m \quad ,$$

其中 x_0 称为种子，改变它的值就得到基本序列的不同区段。a, c, x_0, m 为大于零的整数，分别叫做**乘子、增量、初值**和**模**。使用时需要仔细地挑选模数 m 和乘子 a，使得产生出的伪随机数的循环周期要尽可能长。$c \neq 0$ 时能实现最大的周期，但是得到的伪随机数的特性不好。$c \neq 0$ 的这类情况称为**混合同余发生器**。通常选取 x_0 为任意非负整数，乘子 a 和增量 c 取如下形式

$$a = 4q + 1 \quad , \quad c = 2p + 1 \quad .$$

p 和 q 为正整数。这两种方法中的 p, q, x_0, m 值的选择一般是通过定性分析和计算机试验来选择，使得到的伪随机数列具有足够长的周期，而且独立性和均匀性都能通过一系列的检验。

对 $c = 0$ 的情况叫做**乘同余法**，由于减少了一个加法，因而可以使产生伪随机数的速度快些。这种方法产生的伪随机数递推公式为

$$x_{n+1} = ax_n (\mathrm{mod}\, m)$$
$$\xi_n = x_n / m \quad ,$$

x_0, a, m 也为正整数，并分别叫做**初值、乘子**和**模**。

其他的产生伪随机数的方法：混沌法伪随机数产生、反馈移位寄存器（RNG）等。

2．伪随机数的统计检验

统计检验：均匀性检验、独立性检验、组合规律检验、无连贯性检验、参数检验等等。

最基本的是它的均匀性和独立性的检验。

均匀性是指在[0，1]区域内等长度区间的随机数分布的个数应相等。

独立性是按先后顺序出现的若干个随机数中，每一个数的出现都和它前后的各个数无关。

一个好的伪随机数序列除了能通过这两种主要的统计检验外，还需要能通过别的多种检验。能通过的检验越多，则该产生器就越优良可靠。

（1） 均匀性检验—频率检验

均匀性检验是所有检验中最简单的一种。它的方法很多，如χ^2方法。

设有在区间[0，1]上的伪随机数序列为$\{\xi_1,\xi_2,\cdots\cdots,\xi_N\}$。如果该伪随机数是均匀分布的，则将[0，1]区间分成 k 个相等的子区间后，落在每个子区间的伪随机数个数N_i应当近似为N/k。此数也称频数。它的统计误差$\sigma_i=\sqrt{N_i}=\sqrt{N/k}$。统计量$\chi^2$按定义应为

$$\chi^2=\sum_{i=1}^{k}\frac{(N_i-N/k)^2}{N/k}=\frac{k}{N}\sum_{i-1}^{k}(N_i-N/k)^2 \quad .$$

χ^2 在此问题中应服从 $\chi^2(k-1)$ 的分布。据此可以假定一个显著性水平值来进行检验。我们可以从 χ^2 表查得 $(k-1)$ 个自由度的显著水平为 α 时的 t_α 值。计算出来的 χ^2 小于 t_α，则认为在 α 的显著水平下，原伪随机数在[0，1]区间是均匀分布的假定是正确的。

如果计算的得到的 χ^2 大于 t_α，则认为在 α 的显著水平下，伪随机数不满足均匀性的要求。

通常取显著水平为 0.01 或 0.05。为了反映均匀性分布的特性，k 的取值不宜太小，但也不能太大。一般选取的 k 值，要能使每个子区间有若干个伪随机数时就比较合适。

（2） 独立性检验—无重复联列检验

如果把[0，1]上的伪随机数序列 $\{\xi_1, \xi_2, \ldots\ldots, \xi_{2N}\}$ 分成两列

$$\xi_1, \xi_3, \ldots \xi_{2i-1}, \ldots, \xi_{2N-1}$$

$$\xi_2, \xi_4, \ldots \xi_{2i}, \ldots, \xi_{2N}$$

第一列作为随机变量 x 的取值，第二列作为随机变量 y 的取值。在 $x-y$ 平面内的单位正方形域 $[0 \le x \le 1, 0 \le y \le 1]$ 上，分别以平行于坐标轴的平行线，将正方形域分成 $k \times k$ 个相同面积的小正方形网格。落在每个网格内的随机数的频数 n_{ij} 应当近似等于 N/k^2。由此可以算出 χ^2 为

$$\chi^2 = \sum_{i,j=1}^{k} \frac{k^2}{N}\left(n_{ij} - \frac{N}{k^2}\right)^2 \quad .$$

χ^2 应满足 $\chi^2\left((k-1)^2\right)$ 的分布。据此可以采用与均匀性检验的 χ^2 方法，假定出显著性水平来进行检验。我们也可以把伪随机数序列分为三列、四列、...，用与上面所述相似的方法进行多维独立性检验。

3.独立于计算机机型的伪随机数产生器

在实际应用中，我们常常希望使用能够在各种型号的计算机上工作，并能重复产生相同伪随机数序列的产生器。

这种产生器的实现是基于如下的思想：如果要产生[0,1]区间的伪随机浮点数，选择精度最低的计算机作为标准精度，而对字长较长的计算机，用将较低的数位人为置零的方法，即在高精度的计算机上进行较低精度的运算。

这样的伪随机数产生器无论从伪随机数的重复周期和产生伪随机数的速度都不算理想，但它却可以在大多数的计算机上工作。

这些伪随机数产生器产生的前几个数近似为：R1=0.10791504...，R2=0.58747506...。

```
      FUNCTION RN32(IDUMMY)
C     CDC VERSION  F.JAMES,  1978
C     IY IS THE SEED.   CONS=2**-31
```

```
      DATA   IY/65539/
      DATA   CONS/1661400000000000000B/
      DATA   MASK31/17777777777B/
      IY=IY*69069
C       KEEP ONLY LOWER 31 BITS
      IY=IY.AND.MASK31
C       SET LOWER 8 BITS TO ZERO TO ASSURE EXACT FLOAT.
       IY=IY.AND.07777777777777777400B
       YFL=JY
       RN32=YFL*CONS
       RETURN
C          ENTRY TO INPUT SEED
       ENTRY RN32IN
       IY=IDUMMY
       RETURN
C          ENTRY TO OUTPUT SEED
       ENTRY RN32OT
       IDUMMY=IY
       RETURN
       END

       FUNCTION RN32(DUMMY)
```

```
C          IBM VERSION,   F.JAMES, 1978
C          IY IS THE SEED,  CONS=2**-31
         DATA  IY/65539/
         DATA  CONS/Z39200000/
         IY=IY*69069
C          ASSURE LEFTMOST BIT ZERO(POSITIVE INTEGER)
         IF(IY.GT.0) GOTO 6
         IY=IY+2147483647+1
6  CONTINUE
C          SET LOWER 8 BITS TO ZERO TO ASSURE EXACT FLOAT
         JY=(IY/256)*256
         YFL=JY
         RN32=YFL*CONS
         RETURN
C          ENTRY TO INPUT SEED
         ENTRY RN32IN(IX)
         IY=IX
         RETURN
C          ENTRY TO OUTPUT SEED
         ENTRY RN32OT(IX)
         IX=IY
         RETURN
```

END