[文章编号] 1003－4684(2008)05-0065-05

伪随机数生成算法及比较

郑 列, 宋正义
(湖北工业大学理学院, 湖北武汉 430068)

[摘 要] 讨论了用数学方法快速产生随机数的方法. 运用 Monte-Carlo 方法设计数学实验, 理论分析和实验结果表明:比较好的算法是同余法, 而其中最好的算法是混合同余法和乘同余法.

[关键词] Monte-Carlo 方法; 伪随机数; 同余
[中图分类号] TP391 [文献标识码]:A

在计算机上用数学的方法产生随机数列是目前通用的方法, 它的特点是占用的内存少, 速度快.

用数学方法产生的随机数列是根据确定的算法推算出来的, 严格说来并不是随机的, 因此一般称用数学方法产生的随机数列为伪随机数列. 不过只要用数学公式产生出来的伪随机数列通过统计检验符合一些统计要求, 如均匀性、抽样的随机性等, 也就是说只要具有真正随机数列的一些统计特征, 就可以把伪随机数列当作真正的随机数列使用[1,2].

由于是用算法产生的, 因而本质上是决定性的, 再加上计算机字长有限, 所以无论用什么算法产生的数列, 在统计特征上都不可能完全与从均匀分布中抽样所得的子样完全相同, 因而只能要求尽可能地近似.

这种在计算机上用算法得到的统计性质上近似于[0, 1] 上均匀分布的数, 一般就称为伪随机数, 以区别于真正从[0, 1] 上均匀分布中抽样所得到的随机数. 为了快速产生统计特征良好的伪随机数, 数学上一般采用递推公式

$$x_{n+1} = f(x_n, x_{n-1}, \cdots, x_{n-k}).$$

在给定一组初值 $x_0, x_{-1}, x_{-2}, \cdots, x_{-k}$ 后, 即可逐步求出 $x_1, x_2, x_3, \cdots$. 在此基础上产生随机数.

1 伪随机数生成算法

1.1 取中法

产生伪随机数列最早的方法是平方取中法, 即将一个 $2s$ 位十进制随机数平方后得到的一个 $4s$ 位数, 去头截尾取中间 $2s$ 位数作为一个新的随机数, 重复上述过程便得到一伪随机数列. 平方取中法的递推公式

$$x_{n+1} = [x_n^2 / 10^s] \pmod{10^{2s}}, \quad (1)$$

产生伪随机数列

$$r_n = x_n / 10^{2s} = x_n 10^{-2s}. \quad (2)$$

平方取中法的优点为在计算机上易于实现, 内存占用少, 但仍存在对小数目偏倚的现象, 均匀性不好, 数列的长度和周期难以确定, 对初始数据的依赖很大.

对平方取中法进行改进, 产生了乘积取中法, 如果要产生具有 10 进制 $2s$ 位的伪随机数列, 任取两个初始随机数 x_0, x_1, 用递推公式

$$x_{n+2} = [x_{n+1} x_n / 10^s] \pmod{10^{2s}}, \quad (3)$$

产生伪随机数列

$$r_n = x_n 10^{-2s}. \quad (4)$$

乘积取中法与平方取中法比较, 从产生的伪随机数列长度及均匀性方面都有改善, 但是数列长度还是不够, 而且对初始值的依赖性很大.

1.2 移位法

电子计算机善于进行移位等逻辑运算, 应用机器的这个特点有一类产生伪随机数列的方法, 该方法称为移位法.

如果字长为 32 位的计算机, 取一初始值 x_0, 将 x_0 左移 7 位得 x_{01}, 右移 7 位得 x_{02}. 将 x_{01} 和 x_{02} 进行指令相加得到 x_1, 再对 x_1 进行上述运算过程得到

[收稿日期] 2007－09－04
[基金项目] 湖北省统计局科研项目(HB062-11), 湖北省教育厅科研项目(2006y119).
[作者简介] 郑 列(1963－), 男, 湖北黄石人, 湖北工业大学教授, 研究方向:应用数学.

x_2,这样多次重复可得正整数序列x_n, 取$r_n = x_n 2^{-32}$为[0,1] 上均匀分布的伪随机数列通项.这一算法递推公式为

$$x_{n+1} = x_n 2^7 + x_n 2^{-7} \pmod{2^{32}},$$

产生伪随机数列

$$r_{n+1} = x_{n+1} 2^{-32}. \tag{5}$$

移位法运算速度快, 但是对初始值的依赖性也很大, 一般地初始值不能取得太小, 选得不好会使伪随机数列长度较短.

1.3 同余法

目前产生伪随机数列比较好的方法为同余法, 其中有混合同余法、加同余法和乘同余法[3] .采用线性递推公式

$$x_{n+1} = \lambda x_n + c \pmod{M},$$
$$0 \leqslant x_{n+1} < M,$$

产生伪随机数列

$$r_n = x_n / M. \tag{6}$$

其中 λ, c, M 以及初值x_0 都是正整数 .容易看出r_n 满足

$$0 \leqslant r_n < 1.$$

若 $c \neq 0$ 且 $\lambda \neq 1$, 则称这种方法为混合同余法.

当 $c = 0$ 时,

$$x_{n+1} = \lambda x_n \pmod{M},$$
$$0 \leqslant x_{n+1} < M,$$
$$r_n = x_n / M, \tag{7}$$

称为乘同余法.

当 $\lambda = 1, c \neq 0$时,

$$x_{n+1} = x_n + c \pmod{M}, 0 \leqslant x_{n+1} < M,$$
$$r_n = x_n / M, \tag{8}$$

称为加同余法.

虽然加同余法产生的序列周期长, 电子计算机实现也很方便, 只要进行加法及移位运算即可完成, 但从理论上看, 所得到随机数列的性质一般不如乘同余法和混合同余法.

若 $\lambda = c = 1$,则有$x_{n+1} = x_n + 1 \pmod{M}$.这样构成的序列虽然周期可以达到 M, 但不是随机的.因此乘子λ, 增量c 的选取是非常重要的, 下面给出λ和 c 的所有使得周期为最大的选取方法, 以便从中挑选符合统计性质的伪随机数列.

本文不加证明地引用有关周期的一些结果[4] :

一个混合同余数列, 达到周期 M 的充要条件是 :1)c 与M 互素 ;2)对每一个M 的素因子 p,$\lambda - 1$ 为 p 的倍数 ;3)若M 是 4 的倍数,那么$\lambda - 1$ 是 4 的倍数.

由上面的结论易知, 混合同余法达到最大周期的条件为 :在 2 进制计算机上, 若 $M = 2^s$, 应取$\lambda = 4q_1 + 1, c = 2a_1 + 1$,$x_0$ 为任意非负整数, 其中q_1, a_1 为正整数.

乘同余法达到最大周期的条件为 :当 $M = 2^s$, 若取x_0 与M互素, λ为M的本原元素, 应取$x_0 = 2a_2 + 1, \lambda = 8q_q \pm 3, a_2, q_2$ 为正整数, 此时乘同余法的最大周期为 2^{s-2} .

2 Monte-Carlo 方法设计思想

Monte-Carlo 方法的要点是 :对要解决的数值计算问题, 构造适当的概率模型, 使要得到的解正好重合于概率模型中随机变量的概率分布或数字特征, 其后在计算机上用伪随机数列对随机变量进行模拟得到一个大子样的观测数据, 进行统计整理以后, 给出问题的一个近似估计. 因此, Monte-Carlo 方法是双重近似, 一是将数值计算问题用概率模型作近似, 二是在计算机上用伪随机数作近似抽样值进行统计整理作出一些估计.

为了应用 Monte-Carlo 方法对上述的 3 类(6 种)生成伪随机数的方法进行比较, 设计如下的数学实验 :

设 $g(x)$ 是[0,1] 上的连续函数, 且 $0 \leqslant g(x) \leqslant 1$.考虑定积分

$$s = \int_0^1 g(x)\mathrm{d}x \tag{9}$$

的值.如图 1 所示, 在单位正方形内, 曲线 $y = g(x)$ 下面的阴影 A 的面积就是积分值s .

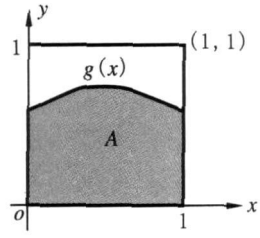

图 1 积分值 s 的几何表示图

如果在图 1 所示的单位正方形内均匀地投点(x, y), 则该随机点落入曲线 $y = g(x)$ 下阴影的概率为

$$p = P\{y \leqslant g(x)\} = \int_0^1 \int_0^{g(x)} \mathrm{d}y\mathrm{d}x = \int_0^1 g(x)\mathrm{d}s = s. \tag{10}$$

由此想法构造计算定积分的投点模型.

向正方形 $0 \leqslant x \leqslant 1, 0 \leqslant y \leqslant 1$ 内均匀投点 (ξ_i, η_i), ξ_i, η_i 是相互独立的均匀随机数列, 第 i 次试验成功, 即(ξ_i, η_i) 落入A 中, 也就是满足$\eta_i \leqslant g(\xi_i)$, 若每次成功的概率为 p, 进行n 次试验成功了 k 次, 则由大数定理知

$$\lim_{n\to\infty}\frac{k}{n}=p\ (\text{a.s}).$$

即 n 充分大时,k/n 依概率收敛到 p. 由于 $p=s$, 因此常取 $s\approx k/n$.

3　数学实验结果及分析

下面应用上述 Monte-Carlo 方法对第一部分所述的 3 类(6 种)生成伪随机数方法进行比较, 其步骤是 :首先用算法产生随机数, 然后用所生成的随机数进行简单的定积分计算. 以计算积分

$$I_2=\int_0^1 x^2\,\mathrm{d}x,\ I_3=\int_0^1 x^3\,\mathrm{d}x,$$
$$I_4=\int_0^1 x^4\,\mathrm{d}x,\ I_9=\int_0^1 x^9\,\mathrm{d}x$$

为例.

显然由直接积分得 :

$$I_2=\int_0^1 x^2\,\mathrm{d}x=\frac{1}{3}=0.3333,$$
$$I_3=\int_0^1 x^3\,\mathrm{d}x=\frac{1}{4}=0.2500,$$
$$I_4=\int_0^1 x^4\,\mathrm{d}x=\frac{1}{5}=0.2000,$$
$$I_9=\int_0^1 x^9\,\mathrm{d}x=\frac{1}{10}=0.1000.$$

现在用投点方法来计算, 以 k/n 作为 I_i 的近似值, 这里 k 是 n 次投点试验中, 使不等式

$$\xi_m^i\geqslant\eta_m\quad(i=2,3,4,9)$$

成立的个数.

下面将用混合同余法、乘同余法、加同余法、平方取中法、乘积取中法、移位法这 6 种方法分别产生随机数列, 并用投点法来计算上述 4 个积分.

以下程序在 VC++6.0 上编译通过.

```
//求积分 Integrate[ x^n,{x, 0, 1}]
#include<iostream>
#include<iomanip>
#include<math.h>
using namespace std;
const long M=2097152;//2^21
const long N=1600000;
const int n=2;
void rand1(long &x1, long &x2);//混合同余法
void rand2(long &x1, long &x2);//乘同余法
void rand3(long &x1, long &x2);//加同余法
void rand4(long &x);//平方取中法
void rand5(long &x1, long &x2);//乘积取中法
void rand6(long &x);//移位法
float f(float x, int k);//y=x^k
void main()
{
  int i;
  long k[6]={0,0,0,0,0,0}, A[8][2], j;
  float I[6];
  A[0][0]=19;A[0][1]=37;
  A[1][0]=19;A[1][1]=37;
  A[2][0]=19;A[2][1]=515;
  A[3][0]=37;A[3][1]=129;
  A[4][0]=1234;A[4][1]=5829;
  A[5][0]=6513;A[5][1]=3245;
  A[6][0]=4158;A[6][1]=3023;
  A[7][0]=797152;A[7][1]=315023;//初值
  for(j=1;j<N;j++)
  {
    rand1(A[0][0], A[0][1]);//混合同余法
    rand2(A[1][0], A[1][1]);//乘同余法
    rand3(A[2][0], A[2][1]);//加同余法
    rand3(A[2][0], A[2][1]);//为了增加独立性, 取其偶数项
    rand3(A[3][0], A[3][1]);//加同余法
    rand3(A[3][0], A[3][1]);//为了增加独立性, 取其偶数项
    rand4(A[4][0]);//平方取中法
    rand4(A[4][1]);//平方取中法
    rand5(A[5][0], A[5][1]);//乘积取中法
    rand5(A[6][0], A[6][1]);//乘积取中法
    rand6(A[7][0]);//移位法
    rand6(A[7][1]);//移位法
    for(i=0;i<2;i++)
    {
      if((1.0*A[i][1])/M<f((1.0*A[i][0])/M, n))
        k[i]++;
    }
    if((1.0*A[2][0])/M<f((1.0*A[3][0])/M, n))
      k[2]++;
    if((1.0*A[4][1])/10000<f((1.0*A[4][0])/10000, n))
      k[3]++;
    if((1.0*A[5][0])/10000<f((1.0*A[6][0])/10000, n))
      k[4]++;
    if((1.0*A[7][1])/M<f((1.0*A[7][0])/M, n))
      k[5]++;
  }
  for(i=0;i<6;i++)
    I[i]=(1.0*k[i])/N;
  cout<<showpoint<<fixed<<left<<setprecision(4);
  cout<<"试验次数 N="<<N<<", 不同随机数产生算法的实验结果:"<<endl;
  cout<<"混合同余法"<<"\t"<<"乘同余法"<<"\t"<<"加同余法"<<endl;
  cout<<"I_"<<n<<"="<<setw(12)<<I[0];
  cout<<"I_"<<n<<"="<<setw(12)<<I[1];
  cout<<"I_"<<n<<"="<<setw(12)<<I[2]<<endl;
  cout<<"平方取中法"<<"\t"<<"乘积取中法"<<"\t"<<"移位法"<<endl;
  cout<<"I_"<<n<<"="<<setw(12)<<I[3];
  cout<<"I_"<<n<<"="<<setw(12)<<I[4];
  cout<<"I_"<<n<<"="<<setw(12)<<I[5]<<endl;
}
void rand1(long &x1, long &x2)//混合同余法
{
  int a[2]={129, 515}, c[2]={17, 53};//a[2], c[2] 分别为乘子和增量
  x1=(a[0]*x1+c[0])%M;x2=(a[1]*x2+c[1])%M;
}
void rand2(long &x1, long &x2)//乘同余法
{
  int a[2]={83, 197};//a[2] 为乘子
  x1=(a[0]*x1)%M;x2=(a[1]*x2)%M;
}
void rand3(long &x1, long &x2)//加同余法
{
  long t;
  t=x2;x2=(x1+x2)%M;x1=t;
}
void rand4(long &x)//平方取中法
{
```

```
    x=((x *x)/100)%10000;
狅
 void rand5(long & x1, long & x2) //乘积取中法
狅
    long t;
    t=x2; x2=((x1 *x2)/100)%10000; x1=t;
狅
 void rand6(long & x) //移位法
狅
    x=(x *128+x/128)%M;
狅
 float f(float x, int k) //y=x^k
狅
    int i;
    float y=1;
    if(k==0)
       return 1;
    for(i=0;i<k;i++)
       y=y *x;
    return y;
狅
```

程序运行结果:分别设定程序中的 n 为 2, 3, 4, 9 即为分别求

$$I_2=\int_0^1 x^2\,\mathrm{d}x,\quad I_3=\int_0^1 x^3\,\mathrm{d}x,$$

$$I_4=\int_0^1 x^4\,\mathrm{d}x,\quad I_9=\int_0^1 x^9\,\mathrm{d}x.$$

1) $n=2$.

试验次数 $N=1\,600\,000$, 不同随机数产生算法的实验结果:

混合同余法	乘同余法	加同余法
I_2=0.3330	I_2=0.3323	I_2=0.3342
平方取中法	乘积取中法	移位法
I_2=0.0000	I_2=0.0000	I_2=0.3333

2) $n=3$.

试验次数 $N=1\,600\,000$, 不同随机数产生算法的实验结果:

混合同余法	乘同余法	加同余法
I_3=0.2499	I_3=0.2496	I_3=0.2506
平方取中法	乘积取中法	移位法
I_3=0.0000	I_3=0.0000	I_3=0.2667

3) $n=4$.

试验次数 $N=1\,600\,000$, 不同随机数产生算法的实验结果:

混合同余法	乘同余法	加同余法
I_4=0.1999	I_4=0.2000	I_4=0.2005
平方取中法	乘积取中法	移位法
I_4=0.0000	I_4=0.0000	I_4=0.0667

4) $n=9$.

试验次数 $N=1\,600\,000$, 不同随机数产生算法的实验结果:

混合同余法	乘同余法	加同余法
I_9=0.1001	I_9=0.1000	I_9=0.1007
平方取中法	乘积取中法	移位法
I_9=0.0000	I_9=0.0000	I_9=0.0667

由上面的几组试验可以看出, 3 种同余法都有比较好的性质, 算出的结果精度高. 而取中法(平方取中法、乘积取中法)和移位法的计算结果明显是错误的.

在程序运行时通过变量跟踪会发现, 取中法(平方取中法、乘积取中法等)很快退化为 0 或产生周期较短的循环. 使用平方取中法, 当 $2s=4$ 初值 $x_0=9835$ 时, $x_{27}=0$ 初值 $x_0=6406$ 时, $x_{19}=8100$, $x_{20}=6100$, $x_{21}=2100$, $x_{22}=4100$, $x_{23}=8100$, 这时进入了一个循环 初值 $x_0=5829$ 时, $x_{36}=8100$, 由上面的结果可以看出此时也进入了循环 ;初值 $x_0=1234$ 时, $x_{56}=0$. 由此可以看出, 平方取中法有很大的缺陷. 而采用改进了的乘积取中法, 当 $2s=4$ 初值 $x_0=6513$, $x_1=3245$ 时, $x_{287}=0$;初值 $x_0=4158$, $x_1=3023$ 时, $x_{93}=0$. 由此可以看出, 乘积取中法对平方取中法的改进还是有一些效果.

而采用移位法也很快退化为 0 或产生周期较短的循环, 并且对初值的依赖性很大, 产生的随机数列长度不能满足一般的实验要求. 在上面的程序中, 移位为 7 位, $M=2^{21}$;初值 $x_0=797152$ 时, $x_1=1378387$, $x_{16}=1378387$ 初值 $x_0=315023$ 时, $x_1=479517$, $x_{16}=479517$;初值 $x_0=73559$ 时, $x_1=1027518$, $x_{16}=1027518$. 由此可以看出移位法对初值的依赖性很大.

但是几种同余法只要选择了合适的参数, 对初值的依赖较小, 产生的随机数列的性质也较好. 由上面的实验还可以看出, 乘同余法和混合同余法的稳定性要比加同余法好. 这对产生随机数有一定的参考价值.

[参 考 文 献]

[1] Von Neumann J. Various technique used in connection with random digits[J]. U. S. Nat. Bur. Stand. Appl. Math. Ser., 1951, 12:36—38.

[2] Pang W K, Yang Z H, Hou S H. et al. Non-uniform random variate generation by vertical strip method with gived density[J]. Euroupean Joural of Operation Research, 2002, 35:63—77.

[3] 杨振海, 张国志. 随机数生成[J]. 数理统计与管理, 2006, 25(2):224—252.

[4] 方再限. 计算机模拟和蒙特卡洛方法[M]. 北京:北京工业大学出版社, 1988:98—137.

(下转第 88 页)

[参 考 文 献]

[1] 柯惠新, 丁立宏. 市场调查与分析[M]. 北京:中国统计出版社, 2000.

[2] 上海质量管理研究院. 顾客满意度测评[M]. 上海:上海科技技术出版社, 2002.

[3] 马国庆. 管理统计[M]. 北京:科学出版社, 2002.

[4] 余建英, 何旭宏. 数据统计分析与 SPSS 应用[M]. 北京:人民邮电出版社, .

A Statistic Analysis on the Factors of Influencing Student Statisfaction

LIU Lei

(*School of Science, Hubei Univ. of Technology, Wuhan* 430068, *China*)

Abstract: 18 indxes which may have impacts on student statisfaction have been chosen, Based on the reseach to the papers of question and answers in this aspect from the students of Hubei University of Technology, Then 8 key fators have been concluded out. Moveover, we analyze how the 8 factors influence student statisfaction by using multivariable regression approach, and hence provide a scientific base for university to improve stutedt statisfaction.

Keywords: student satisfaction; factor analysis; multivariable regression

[责任编校:张培炼]

(上接第 68 页)

Algorithms to Generate Pseudo Random Numbers and Comparison

ZHENG Lie, SONG Zheng-yi

(*School of Sciece, Hubei Univ. Technology, Wuhan* 430068, *China*)

Abstract: It is discussed how to generate a great deal of available random numbers with the method of the mathematics. Six methods to generate pseudo-random numbers are used for the simple experiments of the Monte-Carlo Method. It is found that the good algorithms are congruent algorithms. The best is blend congruent algorithms and multiplication congruent algorithms.

Keywords: Monte-Carlo Method; pseudo random number; congruence

[责任编校:张培炼]

(上接第 74 页)

The Analysis of the ESD Current Waveform Stability Based on the Cosine of the Angle between the Vectors

LI Gang[1,2], GUI Yu-feng[2], XU Xiao-ying[2]

(1 *School of Science, Hubei Univ. of Technology, Wuhan* 430068, *China*;
2 *School of Science, Wuhan Univ. of Technology, Wuhan* 430063, *China*)

Abstract: With the method of the cosine of the angle between the vectors similar coefficient, the data of two different experimental conditions to obtain the electrostatic discharge (ESD) are analyzed and compared, which draws the conclusion that a current waveform stability under the first condition is better than that under the second condition if with the same discharge voltage.

Keywords: the cosine of the angle; ESD; similar coefficient; current waveform

[责任编校:张培炼]