

信息隐藏课程作业报告

伪随机数发生器



姓名： 张天然

学号： 1751237

任课老师： 钟计东

日期： 2019 年 10 月 27 日

一、实验背景

● 根据密码学原理，随机数的随机性检验可以分为三个标准：

- 1、统计学伪随机性：在给定的随机比特流样本中，1 的数量大致等于 0 的数量。同理，00，11，01，10 四者数量大致相等；
- 2、密码学安全伪随机性：给定随机样本的一部分和随机算法，不能有效地演算出随机样本的剩余部分；
- 3、真随机性：随机样本不可重现。

● 相应的，随机数也分为三类：

- 1、伪随机数：满足第一个条件的随机数；
- 2、密码学安全的伪随机数：同时满足前两个条件的随机数，可以通过密码学安全伪随机数生成器计算得出；
- 3、真随机数：同时满足三个条件的随机数。

真随机数通过物理实验得出，要满足**随机性，不可预测性和不可重现性**。

伪随机数通过一定的算法和种子得出。本次实验实现的正是伪随机数。

只要随机数是由确定算法生成的，那就是伪随机数。**不断优化算法只能使随机数接近随机。现代计算机中**，目前无法通过一个纯算法来生成真正的随机数。无论是哪种语言，单纯的算法生成的数字都是伪随机数，都是由可确定的函数通过一个种子产生的伪随机数。这也就意味着如果知道了种子，就可以推断接下来的随机数序列的信息。

伪随机数发生器采用特定的算法，将随机数种子 seed 转换成一系列的伪随机数。伪随机数依赖于 seed 的值，给定相同的 seed 值总是生成相同的随机数。伪随机数的生成过程只依赖 CPU，不依赖任何外部设备，生成速度快，不会阻塞。

本次实验将通过时间种子生成符合一定分布规律的伪随机数。

二、 实验人员和开发平台介绍

- 成员：

- (1) 吴依玲：
产生符合高斯分布的伪随机数，并用参数估计法估计相关参数。
- (2) 张天然：
产生符合尺度为 β 的指数分布的伪随机数并估计 β 的值；产生符合 GGD 分布的伪随机数 ($c=1.0$ & $c=0.5$)。

- 开发平台：

Visual studio 2019，使用 C++ 语言进行开发。

三、实验原理

(1) 生成符合指数分布的伪随机数：

指数分布本质上是 gamma 分布的一种特殊形式，可以通过均匀函数的变化实现。

生成符合 0-1 分布的均匀函数则通过 Messene Twister 算法。

MT19937 算法是利用线性反馈移位寄存器产生随机数的。整个算法主要分为三个阶段：

- 1) 获得基础的梅森旋转链；
- 2) 对于旋转链进行旋转算法；
- 3) 对于旋转算法得到的结果进行处理。

从 C++ 11 开始，可以直接使用此算法。本实验中将直接使用 random.h 文件中自带的 MT19937 算法。

```
/*梅森旋转算法生成0-1之间的随机数(double)*/
long double MTRand()
{
    /*-----引用外部代码-----*/
    random_device rd;
    mt19937 mt(rd());
    uniform_real_distribution<double> dis(0.0, 1.0);

    int count = 0;
    double res = dis(rd);
    while (count < 1000) {
        res = dis(rd);
        ++count;
    }
    return res;
}
```

随机数则根据公式生成：

```
/*生成符合指数分布的随机数*/
long double ExponentialRand(double b)
{
    double rand01 = MTRand();
    return (-1.0 / b) * log(1 - rand01);
}
```

尺度参数的估计：

由于随机数符合指数分布，那么尺度参数 beta 即为随机数的期望。

```

/*计算指数分布的数学期望*/
long double ExponentialCalculate(long double* num, int amount)
{
    int n;
    long double sum = 0;
    for (n = 0; n < amount; n++)
    {
        sum += num[n];
    }
    return sum / amount;
}

```

(2) 生成符合广义高斯分布的伪随机数

1) $c=1.0$ 时:

此时广义高斯分布即为 Laplace 分布。

```

/*c=1.0时实际上就是laplace分布*/
long double ILaplace(long double exp, long double beta, long double rand01)
{
    return exp - beta * signal(rand01 - 0.5) * log(1 - 2 * fabs(rand01 - 0.5));
}

```

2) $c=0.5$ 时:

E 的 gamma 分布的形状参数为 2.0, 尺度函数为 $1/(\sqrt{\beta})$ 。

先生成符合伯努利分布的伪随机数 w ,

```

/*生成符合伯努利分布的随机数*/
bool BernoulliRand()
{
    /*-----引用外部代码-----*/
    long double prob = 0.5;
    default_random_engine generator;
    bernoulli_distribution b(prob);
    bool Z = 0;
    for (int i = 0; i < 1000; i++)
        Z = b(generator);
    return Z;
}

```

再生成符合 gamma 分布的伪随机数 E 。

```

/*生成符合gamma分布的随机数*/
long double GammaRand(long double alpha, long double lambda)

```

若 $w=1$, 符合广义高斯分布的伪随机数 $z=E^c$; 反之 $z=-E^c$ 。

```
/*生成广义高斯分布Z*/  
long double GGDRand(long double c)  
{  
    bool w = BernoulliRand();  
    double E = GammaRand(MTRand(), MTRand());  
    double Z;  
    if (w == 1)    Z = pow(E, c);  
    else    Z = -1 * pow(E, c);  
    return Z;  
}
```

四、实验结果比较和分析讨论

- (1) 指数分布测试：
以 $\beta=2$ 进行测试。

- 输出十个值时，计算得出的 β 值误差很大：

```
please input scale parameter beta:
scale parameter beta=2
please input the amount of numbers matching exponential distribution:
10
5.47441
1.02665
1.03612
0.814414
6.02447
0.651025
4.02413
1.51705
4.74603
5.72218

scale parameter(obtained from samples)=3.10365
```

- 当样本数量变大时（如 1000 个），误差减小：

```
please input scale parameter beta:
scale parameter beta=2
please input the amount of numbers matching exponential distribution:
1000
2.48744
1.50479
5.23716
1.02249
0.0155379
0.366923
1.14169
0.664792
0.23969
1.08067
1.14616
1.7456
1.69607
0.245104
0.880267
1.45206
0.706414
4.7983

scale parameter(obtained from samples)=1.92917
```

- (2) 广义高斯分布测试 $c=1.0$

```
1.19143
please input the standard deviation(c=1.0):
1
please input the amount of numbers matching GGD:
10
0.614199
-1.19143
-2.8458
0.216151
-1.85076
-0.0206327
-0.226286
-0.0135767
0.294775
-0.863248
```


(3) 广义高斯分布测试 $c=0.5$

```
please input the scale parameter:
2
please input the amount of numbers matching GGD(c=0.5):
10
-3.3259
-9.47639e-05
-0.0182112
-2.91514
-6.70724e-05
-0.00225199
-0.000229529
-0.000121624
-0.610731
-0.00279313
```


五、实验感想

- 实验中出现的 bug 有：

(1) 调用 random 头文件中的模板报错：

 C2338	invalid template argument for uniform_real_distribution: N4659 29.6.1.1 [rand.req.genl]/1d requires one of float, double, or long double	Project1	random	2708
---	---	----------	--------	------

排查后发现在 MTRandInt 函数中实例化错误，应为 int 型。

(2) 缓冲区溢出

溢出原因为数组申请的空间不足。

- 小结：

本次实验一个月前就布置了，但一直没有着手开始完成，因为觉得实验要求看起来很难。但开始实现之后，发现实验思路很清晰，实现起来也没有什么难以解决的技术问题。

本次实验最大的挫折就是 uniform_real_distribution 模块实例化错误，浪费了大量时间，而错误原因仅仅是实例化时数据类型写错。

六、 参考

- (1) 梅森旋转算法: <https://blog.csdn.net/dianshu1593/article/details/101524003>
- (2) 伯努利随机数: https://blog.csdn.net/caroline_wendy/article/details/17335871
- (3) 求指数分布的期望:
<https://baike.baidu.com/item/%E6%8C%87%E6%95%B0%E5%88%86%E5%B8%83/776702?fr=aladdin>