高等数值算法与应用(8)

– 随机数 (chap9) –

喻文健

Outline

- 伪随机数
- 均匀分布随机数生成算法
  - 相乘取模法
  - Marsaglia算法
  - Mersenne Twister算法
- 正态分布随机数生成算法
  - 基于中心极限定理的算法
  - 金字形神塔算法
- rand, randn命令

伪随机数 (pseudorandom)

- rand命令 -- 产生0~1之间均匀分布的随机数
  - 每次启动Matlab, 第一次运行rand都得到同样的数
  - 0.814723686393179
  - rand命令真的能生成随机数吗?
  - 计算机是确定性的机器, 只能生成伪随机数!
  - 即使访问外部时钟, 也只是避免每次生成出相同的序列
  - 伪随机数的定义: (by D. Lebmer, 1951)
  - 随机序列是一个模糊的说法……其中每一项对于外行都是不可预测的，并且这些数字通过了一定数量的统计学传统测试
  - 如何生成一个看似随机的数的序列? (同时计算量要小)

均匀分布随机数生成算法

- 相乘取模法
  Marsaglia算法
  Mersenne Twister算法

均匀分布随机数

- 相乘取模法 (multiplicative congruential)
  - 计算公式: $x_{k+1} = ax_k + c \bmod m$ 除以m后取余数
  - 整数参数: a, c, m; 初始值x_0 (整数) 种子
  - 例: $\boxed{x_{k+1} = 13x_k \mod 31}$, $x_0 = 1$
  - 1, 13, 14, 27, 10, 6, 16, 2, 7, 29, 5, 3, ⋯
  - 下一个值是? 8
  - 前30项是1~30的一个排列，然后重复自身，周期T=m−1
  - 通过计算x_k/m, 得到(0, 1)区间上均匀分布随机数
  - 注意：在这种方法中，一般设置c=0 不能有0
  - 因此, 适当设置a, m的值，得到$\{x_k\}$为整数1~m−1的一种排列，然后以周期T=m−1重复自身

均匀分布随机数

- 相乘取模法
  - 计算公式$x_{k+1} = a \cdot x_k \mod m$ (假设c=0)
  - 两条性质：
  - 1. 序列$\{x_k\}$的周期不可能大于m-1 x_{k+1}只由x_k决定
  - 2. a与m不能有共同的质因子 否则$\{x_k\}$都有这个因子
  - 优点：计算简单，长长的序列中每个值的可能性相同
  - 缺点：统计性质不太好, 生成的数有限制
  - 例：1960年代IBM大型机中用的RANDU算法

$2^{16}+3$, 质数 $x_{k+1} = 65539 \cdot x_k \mod 2^{31}$ 20亿多一点

  - 存在问题: 连续三个数之间关系密切 $x_{k+2} = 6x_{k+1} - 9x_k$

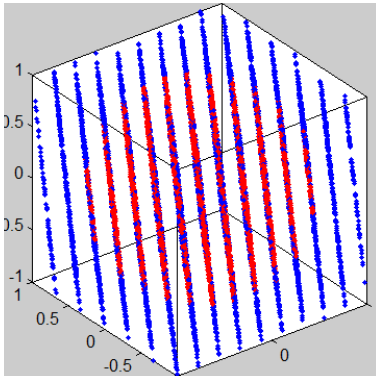
(取模之后)

均匀分布随机数

- 相乘取模法
  - 演示程序randgui近似计算π，也评价随机数的统计性质
  - 在立方体内生成大量的均匀分布随机点(N个)，统计其中落入内切球内的个数(n个), 则 $\lim\limits_{N\to\infty}\frac{n}{N}=\frac{4\pi r^3/3}{8r^3}=\frac{\pi}{6}$
  - $\frac{6n}{N}\approx\pi$, 只要随机点足够多，可得到π值的很好近似
  - 生成随机点：若$\{x_k\}$为均匀分布随机数, 用连续生成的三个数得一个点，得到的点是均匀分布的
  - 测试RANDU的算法, 它的程序为randssp

```
>> randgui randssp
```

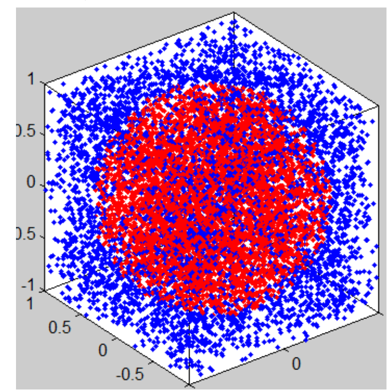
  - 由于 $x_{k+2}=6x_{k+1}-9x_k$ (取模), 生成的点在一些平面上, 无法充满立方体



虽然可近似算π

均匀分布随机数

- 相乘取模法
  - 早期Matlab中的rand命令 (by Park and Miller, 1988)
  - $x_{k+1} = a \cdot x_k \mod m$
  - $a = 7^5 = 16807$, $m = 2^{31} - 1$ (20亿多一点, 质数)
  - 实现在程序randmcg中(Multiplicative Congruential)
  - randgui的测试结果
  - 生成的随机点之间不再有强的联系，在立方体中形成更好的"随机云"
  - 缺点：生成约20亿个数后重复自身(目前的计算机完成这个周期只需几分钟)； 只生成一部分的浮点数



周期不够长！

0~1之间浮点数个数：$2^{52} \times 1022 \approx 2^{62}$

均匀分布随机数

- Marsaglia生成器
  - 1995至2006年Matlab中的rand (by G. Marsaglia, 1991)
  - 不再是相乘取模法; 没有乘、除法，也非两个整数的比值
  - 不采用单一的“种子”，而维护长35个字的“状态向量”
  - 其中32个为0~1之间的浮点数：$z_0, z_1, \cdots, z_{31}$
  - 索引值i (0~31), 随机整数j, 借位标记b (初值0)各一个
  - 生成随机数x_k的算法:　　　决定初始的浮点数组 $\{z\}$
  - 1. $z_{k \bmod 32} := z_{(k+20) \bmod 32} - z_{(k+5) \bmod 32} - b$　　借位减法
  - 2. If $z_{k \bmod 32} > 0$, then $b := 0$;　　ulp, 1与左侧浮点数的间隔
  - Else $z_{k \bmod 32} := z_{k \bmod 32} + 1$; $b := 2^{-53}$;
  - 3. $x_k := z_{k \bmod 32}$;

均匀分布随机数

- Marsaglia生成器
  - 数组{z}初始值的确定：由j作为种子的随机整数发生器，得到一组随机整数，将它们乘以2^{-53}
  - 优点：产生的值在2^{-53}~$1-2^{-53}$之间，是2^{-53}的倍数，共约$2^{53}-1$个值；x_{k+1}不只依赖于x_k，周期很长，几乎2^{1430}
  - 缺点：值是2^{-53}的倍数，仍有很多浮点数不能产生出来
  - 改进措施：利用随机整数j，让z_k的小数部分与它做异或操作(XOR)

z_k (小数、指数分解)log2

f (0.5~1)　　e −52~1

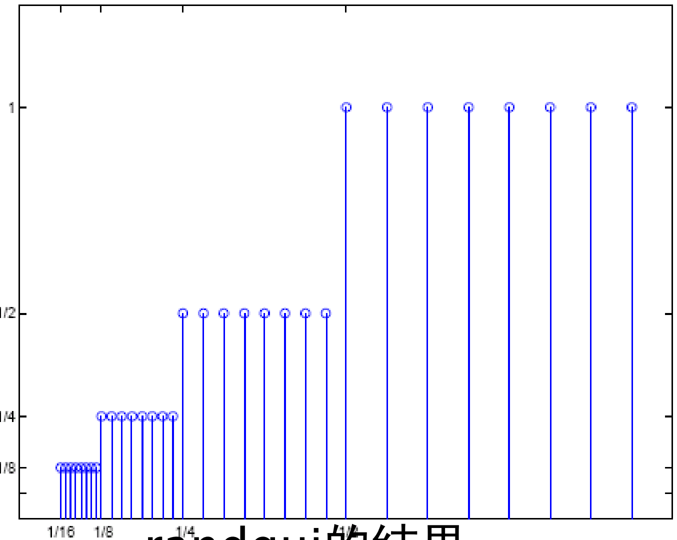
→ (f×2^{53} XOR 随机整数) × 2^{e-53}

可表示2^{-53}~$1-2^{-53}$之间所有浮点数

个数为$2^{52}\times 53$

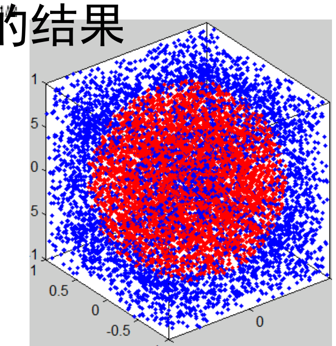
  - 周期更长，达2^{1492}

均匀分布随机数



randgui的结果

- Marsaglia生成器
  - 程序源代码已实现在randtx.m中
  - 生成浮点数的概率分布图 (相对量)
    (作为示意，假设ulp$=2^{-4}$, 而非2^{-53})
- Mersenne Twister算法
  - Marsaglia生成器无法通过run-length测试
  - 自Matlab v2007采用Mersenne Twister算法
  - 采用624个字的“状态向量”
  - 巨大的周期$2^{19937}-1$, 产生的也是$2^{-53}\sim1-2^{-53}$之间所有浮点数
  - M. Matsumoto and T. Nishimura, Mersenne Twister Home Page, http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html



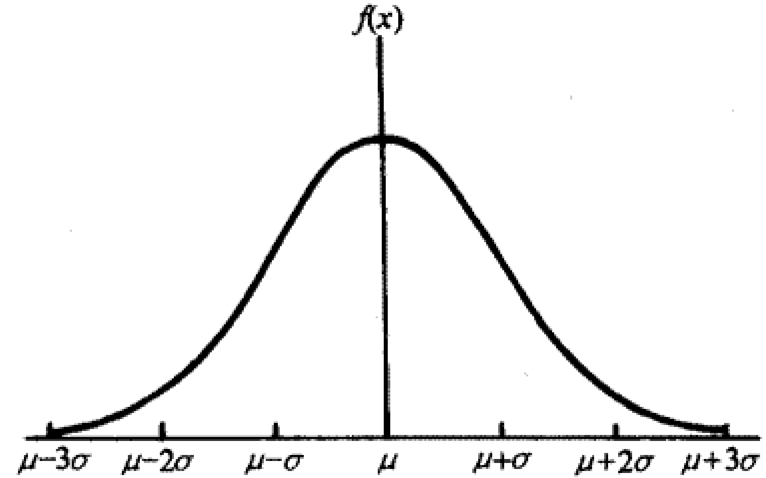
程序run_length_test.m

正态分布随机数生成算法

- 基于中心极限定理的算法
- 金字形神塔算法

正态分布随机数

- 正态分布　　　(也叫高斯分布)
  - 概率密度函数f(x)的曲线
  - 均值μ, 标准差σ, 记为$N(\mu, \sigma^2)$
  - Matlab中randn命令产生标准正态分布随机数, 即上述$\mu=0$, $\sigma=1$



- 基于中心极限定理的算法
  - 中心极限定理: 若一组随机数$\{x_k\}$中的每项独立同分布, 均值为μ, 方差为σ^2, 则$\frac{1}{n}\sum_{i=1}^{n} x_i \rightarrow N(\mu, \sigma^2/n)$
  - 可用均匀分布随机数得到正态分布随机数
  - 均匀分布U[a,b], 均值为(a+b)/2, 方差为$(b-a)^2/12$

分布密度函数
$f(x)=1/(b-a)$

x为mxn数组,近似$N(0,1)$分布

```
x= sum(rand(m, n, 12), 3)-6;
```

按第3维求和

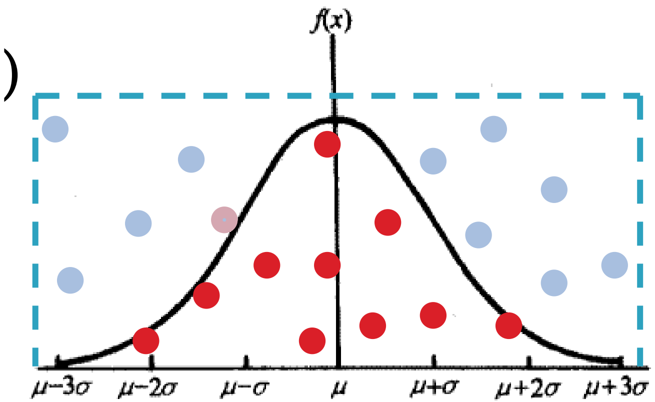
得到U[0, 1]数构成的三维数组

见normalrand1.m

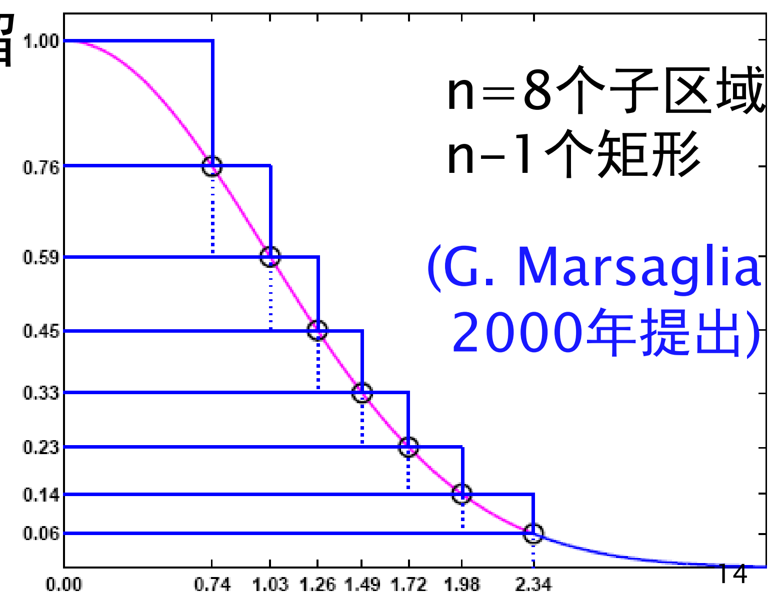
Ziggurat算法

- “金字形神塔”算法 (randn的内部算法)
  - 基于中心极限定理的算法的缺点:
    1.计算量大 (12个随机数→1个数);
    2.无法准确刻画分布的末端效应

    $\boxed{\texttt{x= sum(rand(m, n, 12), 3)-6;}}$

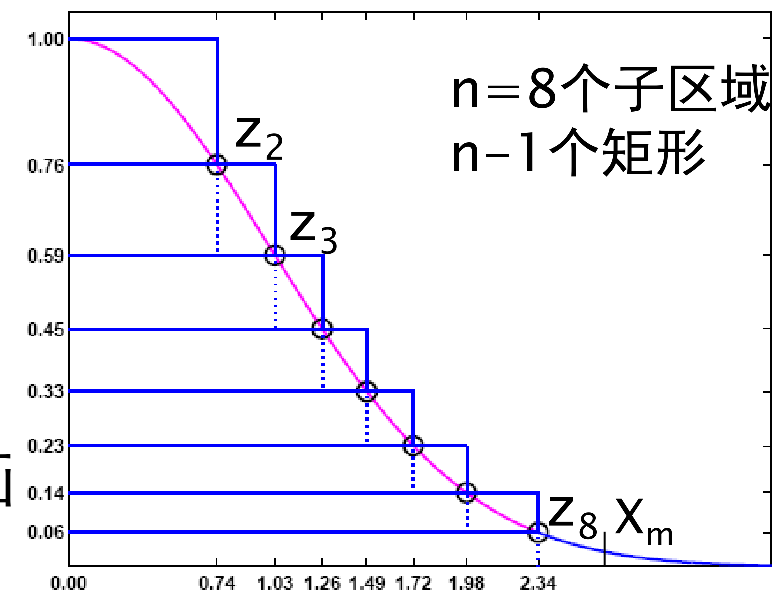


  - 另一思路(拒绝采样): 根据概率密度函数$f(x)=ce^{-x^2/2}$, 在覆盖其曲线的矩形内均匀投点, 保留曲线下点, 其x坐标为正态分布
  - 缺点：计算复杂(exp函数)、舍弃点多使得代价大
  - 不一定用矩形区域覆盖!
  - 在右图蓝色外包络线内均匀取点



Ziggurat算法

- “金字形神塔”算法
  - 预先计算横坐标值$z_2, z_3, \ldots, z_8$，使得8个子区域面积相等
  - 计算$\sigma_j = z_j / z_{j+1}$，表示第j个子区域的左侧矩形(核心区)所占比例
  - 第1个无核心区，第8个不用σ刻画
  - 算法的主要步骤



落入核心区时，只做一次乘法运算

```
j= ceil(n*rand);
u= 2*rand-1; %(-1,1)
if j>1 && j<n,
  if abs(u) <sigma(j)
    r= u* z(j+1);
  else
```

```
    … %判断是否在曲线下
  end
elseif j==n && Xm*abs(u)<z(n),
  r= u*Xm;
else   %子区域1, 子区域n非核心
  …      %判断是否在曲线下
end
```

点不在核心区的情况，需判断是否在函数曲线内

实际n=128，在核心区内的概率>97%

rand, randn命令

rand与randn命令

- rand
  - rand, rand(n), rand(m,n), rand(m,n,p), rand(..., 'single')
  - 采用Mersenne Twister算法, 用RandStream类控制状态
  - randtx是对早期rand版本的实现(Marsaglia算法)
  - randtx('state', 0)设置初始状态, 保证产生同样一组数
  - 为了向后兼容, rand也允许这种设置, 并使用同样的算法
- randn
  - 生成标准正态分布随机数, 语法同rand
  - 算法基于新的rand; 为了向后兼容, 也支持state设置
  - randntx实现了早期rand版本+金字形神塔算法
- 相关命令: randi生成某个范围内均匀分布的随机整数