蒙特卡洛算法：积分

计算物理b
高阳

一维积分问题

- 定积分

$$I = \int_a^b dy\, f_1(y)\,, \qquad 0 \le L \le f_1(y) \le M$$

- “归一化”变换

$$f(y) = \frac{1}{M-L}[f_1(y) - L], \quad 0 \le f(y) \le 1$$

$$x = \frac{y-a}{b-a}, \quad I = (b-a)(M-L)\int_0^1 dx\, f(x) + L(b-a)$$

- 概率算法：x是（0,1）内的均匀分布，则

$$I_0 = E\big(f(x)\big)$$

- 若x按照某个概率g(x)分布，则

$$f^* = \frac{f(x)}{g(x)}, \qquad I_0 = E\big(f^*(x)\big)$$

离散化

- 方差

$$V(f^*) = \int_0^1 (f^\star - I_0)^2 g(x)dx$$

- 离散撒点之后

$$I_0 \approx \frac{1}{N}\sum_i f^*(x_i)$$

- 方差

$$V = \frac{1}{N}\sum_i [f^*(x_i)]^2 - I_0^2 \approx \frac{1}{N}$$

- 根据中心极限定理，方差为0的话算的才是完全准确的。
- 减小方差是核心！

掷点法

- I_0实际上也是在正方形内，点在曲线下的概率。
- 产生两个（0,1）上的均匀随机数ξ_1, ξ_2
- 若$\xi_1 \le f^*(\xi_2)$，则给计数器Nu加1
- 总共产生N对数字，从中获得Nu的值。
- $I_0 \approx \frac{Nu}{N}$

优劣

- 掷点法方差

$$V_2 = p(1-p) = I(1-I)$$

- 对比方差

$$V_2 - V_1 = I(1-I) - \int_0^1 [f(x) - I]^2 dx$$

$$= I - I^2 - \int_0^1 f(x)^2 dx + I^2$$

$$= \int_0^1 f(x)\big(1 - f(x)\big) dx \geq 0$$

平均值法更优

特例

- 蒙卡算法的复杂性可能远超想象，其代码本身的脆弱性也需仔细考量
- 考察下面这个很有代表性的例子：

  定义$I(\gamma)=\int_0^1 dx\, x^\gamma=\frac{1}{\gamma+1}, \gamma>-1$

- 平均值法：产生(0,1)内的均匀分布的随机数x, 计算x^γ的均值与方差：

$$\frac{1}{N}\sum_i x_i^\gamma=\langle x^\gamma\rangle$$

$$\frac{1}{N}\sum_i x_i^{2\gamma}=\langle x^{2\gamma}\rangle$$

$$std=\frac{\sqrt{\langle x^{2\gamma}\rangle-\langle x^\gamma\rangle^2}}{\sqrt{N}}$$

代码与结果

intgamma.m

```
function res = intgamma(gam,N)
%INTGAMMA 此处显示有关此函数的摘要
%   此处显示详细说明
temp=rand(1,N);
res(1)=sum(temp.^gam)/N;
temp=sum(temp.^(2*gam))/N;
res(2)=(temp-res(1)^2)^0.5/N^0.5;

end
```

N=10000

γ	$\langle x^{\gamma}\rangle \pm std$	$1/(\gamma+1)$
2	0.3342±0.0030	0.333
1	0.4963±0.0029	0.5
0	1±0	1
-0.2	1.2557±0.0033	1.25
-0.4	1.6769±0.0159	1.667
-0.5	1.9639±0.0239	2
-0.6	2.4890±0.0640	2.5
-0.8	3.9503±0.1296	5
-0.9	5.8357±0.3416	10

跑了三次里的最差结果

由于误差的形式，N提高100倍，小数点更精确一位，故我们确实应该获得精确到小数点后第二位的结果。但是后两个明显有问题：即使跑多次，中心极限定理告诉我们，每次的结果不能偏离正确值太远。问题在哪里？？？

实时平均

这实际是收敛性检测

intgamma2.m

```
function res = intgamma2(gam,N)
%INTGAMMA 此处显示有关此函数的摘要
%   此处显示详细说明
res=zeros(1,N);

for i=1:N
    temp=rand;
    res(i)=((i-1)*res(i-(i>1.5))+temp^gam)/i;
end

end
```