几点跟进说明

- 作业比例需做调整30%-34%，期末为70%
- 平时作业需要有写代码的能力
- 期末不考写代码，但会有题目考察算法的实施，会是比较简单可以手算的情况。
- 期末会需要各位同学可以读代码，matlab或者python的代码。

蒙特卡洛算法：几个游戏

计算物理b
高阳

经典问题：计算π

基本数学常数：π. 此为超越数，所谓超越数是无法作为代数方程根的数字。如何估算？

级数展开的方法：

基于黎曼ζ函数：

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots$$

拉马努金的冥想公式

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{99^2} \sum_{k=0}^{\infty} \frac{(4k)!}{(k!)^4} \frac{26390k + 1103}{396^{4k}}$$

结果

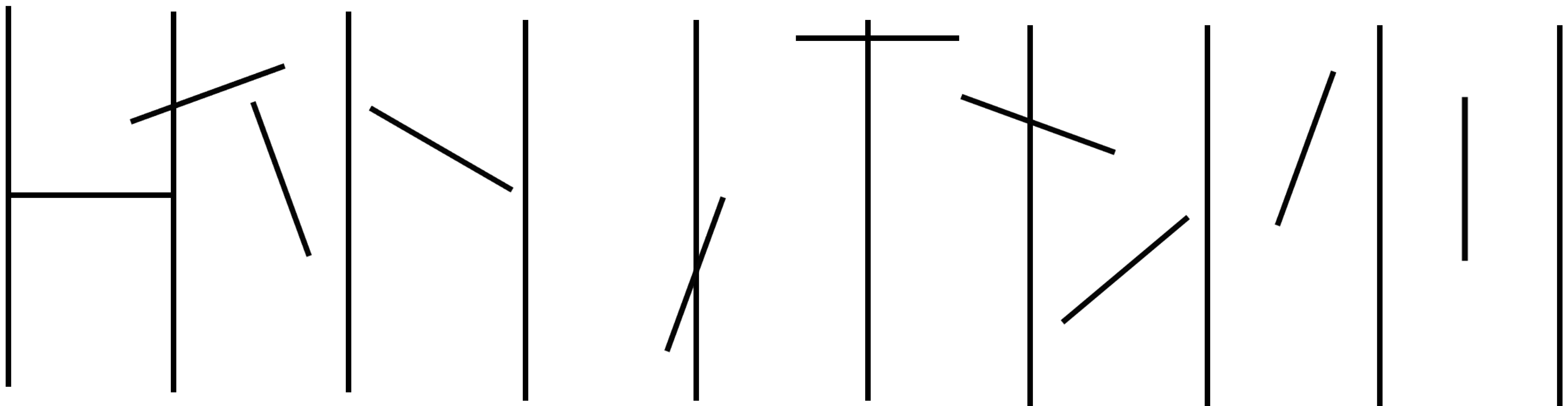
我们当然有好的确定性算法去计算π

项数	1	2	3	4	5	6	1000
方法1	2.449	2.738	2.857	2.922	2.963	2.991	3.1406
方法2	$\pi + 7.64 * 10^{-8}$	$\pi + 4.44 * 10^{-16}$	π	π	π	π	π

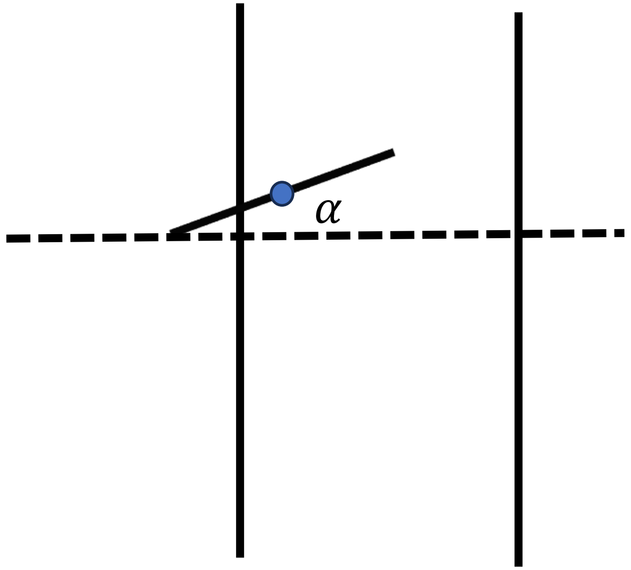
历史起源：布冯投针

π与如下概率问题相关

基本问题：考虑二维平面的一列等距平行线（距离为ℓ），以及一根长为ℓ的针。将此针随机的投于平面之上，问：此针与平行线相交的概率为多少？



分析



当夹角为α时，为相交细针中心须在离平行线如下距离处：$|d| < \frac{\ell}{2}\cos\alpha$, 故概率为$\frac{\ell}{2}\cos\alpha * \frac{2}{\ell} = \cos\alpha$

故总概率为$\frac{1}{\pi}\int_{-\frac{\pi}{2}}^{\pi/2} d\alpha\ cos\alpha = \frac{2}{\pi}$

计算思路

- 根据概率学原理（之后会讲），当投针的次数足够多时，我们可统计针与平行线相交的次数，并用此除以总投针数，此比值应为概率$\frac{2}{\pi}$
- 我们所需的是用计算机模拟随机投针这样的过程，然后统计与平行线相交的次数
- 由于每次投针，针的中心应在两跟平行线之间，我们只需在一个区间投针即可。
- 需要用到随机数生成器，代表即为[0,1]之间的随机数

算法步骤

- 初始化变量：$N_1 = 0, N_2 = 0;$
- 产生[0,1]之间的随机数x,这作为中心点的坐标。此为步骤0.
- 产生$[-\frac{\pi}{2}, \frac{\pi}{2}]$之间的随机角度 α
- 计算左端点$x_1 = x - \frac{1}{2} * \cos\alpha$
- 计算右端点$x_2 = x + \frac{1}{2} * \cos\alpha$
- $N_2 \leftarrow N_2 + 1.$ 若$(x_1 < 0 \,\&\&\, x_2 > 0)$或者 $(x_1 < 1 \&\& \, x_2 > 1)$ 则 $N_1 \leftarrow N_1 + 1$
- 回到步骤0，重复之前步骤，直到重复N 次。
- 根据公式 $\pi \approx \frac{2}{N_1} * N_2$来估算$\pi$

代码

```
function res = buffon(N)

x0=rand([1,N]);
ang=rand([1,N])*pi-pi/2;
N1=0;
for i=1:N
x1=x0(i)-0.5*cos(ang(i));
x2=x0(i)+0.5*cos(ang(i));
if x1<0 && x2>0
N1=N1+1;
else if x1<1 && x2>1
N1=N1+1;
end
end
end
res=2/N1*N;

end
```

运行结果

- 投针100次：　　3.076923，误差 -0.0647
- 投针 10000次：3.1392246, 误差 -0.0024
- 投针1000000次：3.1400822，误差 -0.0015

概率算法的表现

- 不可重复性！ （当然，是理论上的）

代码运行次数	估计值
1	3.0722
2	3.0948
3	3.1646
4	3.1360
5	3.0936
6	3.0900
7	3.1128

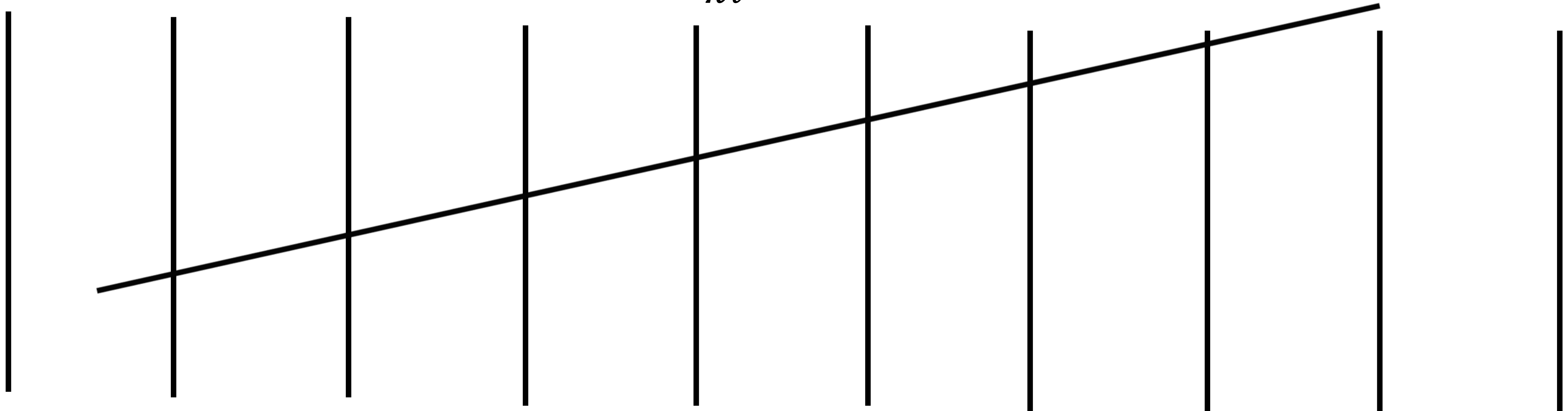
固定循环次数为4000

算法的问题

- 主要在这一步：产生$[-\frac{\pi}{2},\frac{\pi}{2}]$之间的随机角度 α
- 目的是计算pi但算法里需要pi
- 只适合实验，不是合理的算法
- 改进方法见后一个例子。

布冯投针实验的扩展

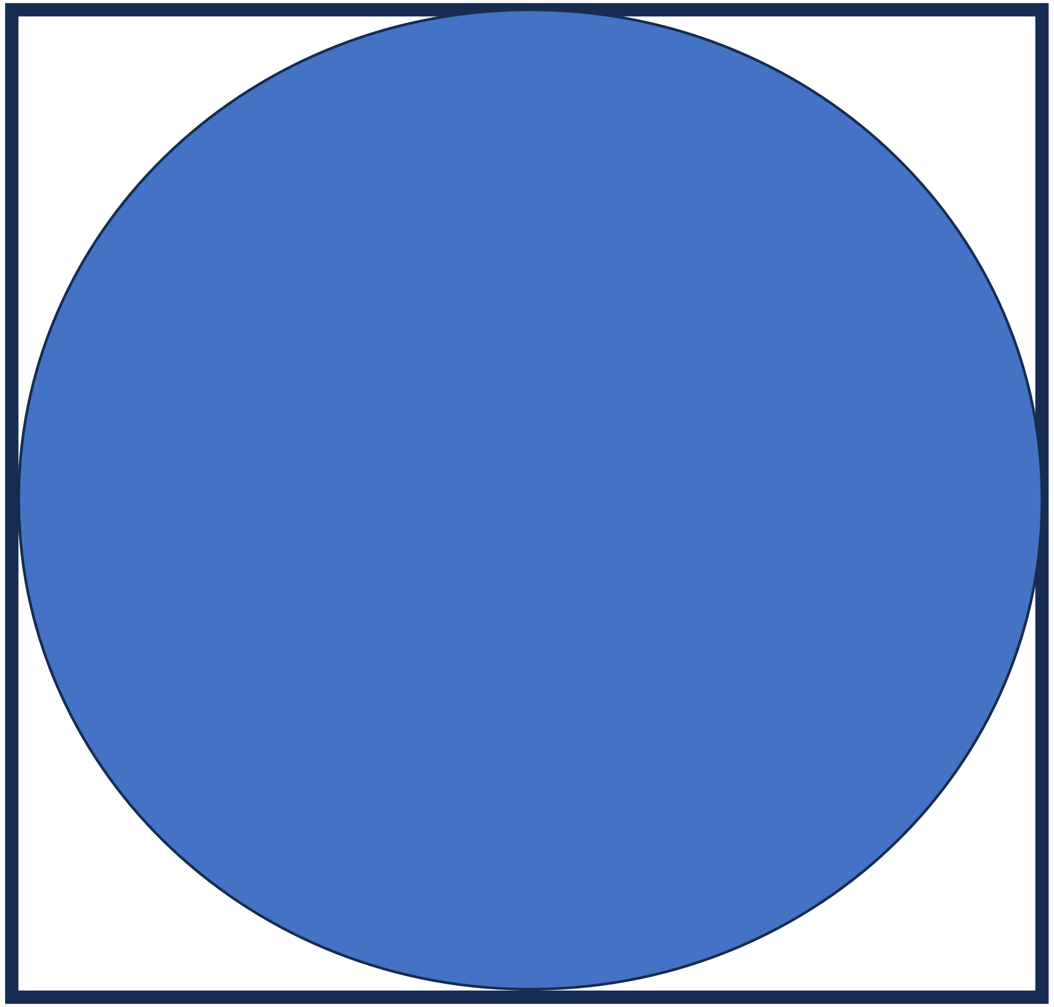
- 针上每个点与线相交的概率其实相等，考虑无穷大的针即可。
- 此结论对于怪异形状的针亦成立，只需将其连在直针之上。
- 每个点相交的概率可由上述推论简单获得：假定直线距离为2ℓ，考虑一个圆形针，半径为ℓ,投此针与直线总相交，而且相交次数总为2,故$p * 2\pi\ell = 2 \Rightarrow p = \frac{1}{\pi\ell}$。



布冯投针实验的扩展

- 考虑如下情况：
  （1）半径为ℓ的圆形针
  （2）长度为$2\pi\ell$的直针
- 二者对于间距为2ℓ的平行线而言给出的平均相交次数相同，都是2
- 但是二者的分布完全不同。（1）是均匀的，（2）最多可与平行线相交4次。
- 这是缩减方差的范例之一。

概率算法1：直接抽样法



在正方形中的单位圆，正方形边长为2.
随机撒点，
点在圆中的概率为$\frac{\pi}{4}$

概率算法1：算法思路

- 设定总撒点次数N.
- 初始化记录数Nh=0.
- 在（-1,1）之间分别获得两个随机数，记为x与y。此步标记为（3）.
- 若x^2+y^2<1, $Nh \leftarrow Nh + 1$.
- 回到步骤（3）往下重新进行，直到总共执行N次。
- 输出$\pi = 4 * \frac{Nh}{N}$

概率算法1：算法引申

- 观测量$O(x,y)$，标记在圆内或外，当点落在(x,y)上，若在圆内则$O(x,y)=1$；否则$O(x,y)=0$.
- 点有一个确定的概率分布$p(x,y)$

这个函数是通过随机抽样获得的，而不是先验给出。这是概率算法与确定性算法的本质不同。

- 我们需要观测量的期望值

$$\frac{N_{hit}}{N_{tot}} = \frac{1}{N_{tot}}\sum_i O_i \approx \langle O \rangle = \frac{\int_{-1}^{1}\int_{-1}^{1} dxdy\, p(x,y) O(x,y)}{\int_{-1}^{1}\int_{-1}^{1} dxdy\, p(x,y)}$$

- 为何分母是需要的？

概率算法1：对布冯投针算法的优化

- 利用上述算法的步骤，当x^2+y^2<1时，这些点(x,y)实际上是单位圆内均匀分布的。
- 故我们不再需要对角度进行撒点
- 而且因为是在圆内的均匀分布，我们也不需直接计算三角函数。
- 当产生出圆内的随机点(x,y)之后，$\frac{x}{(x^2+y^2)^{0.5}}$ 即为cos角度的值。

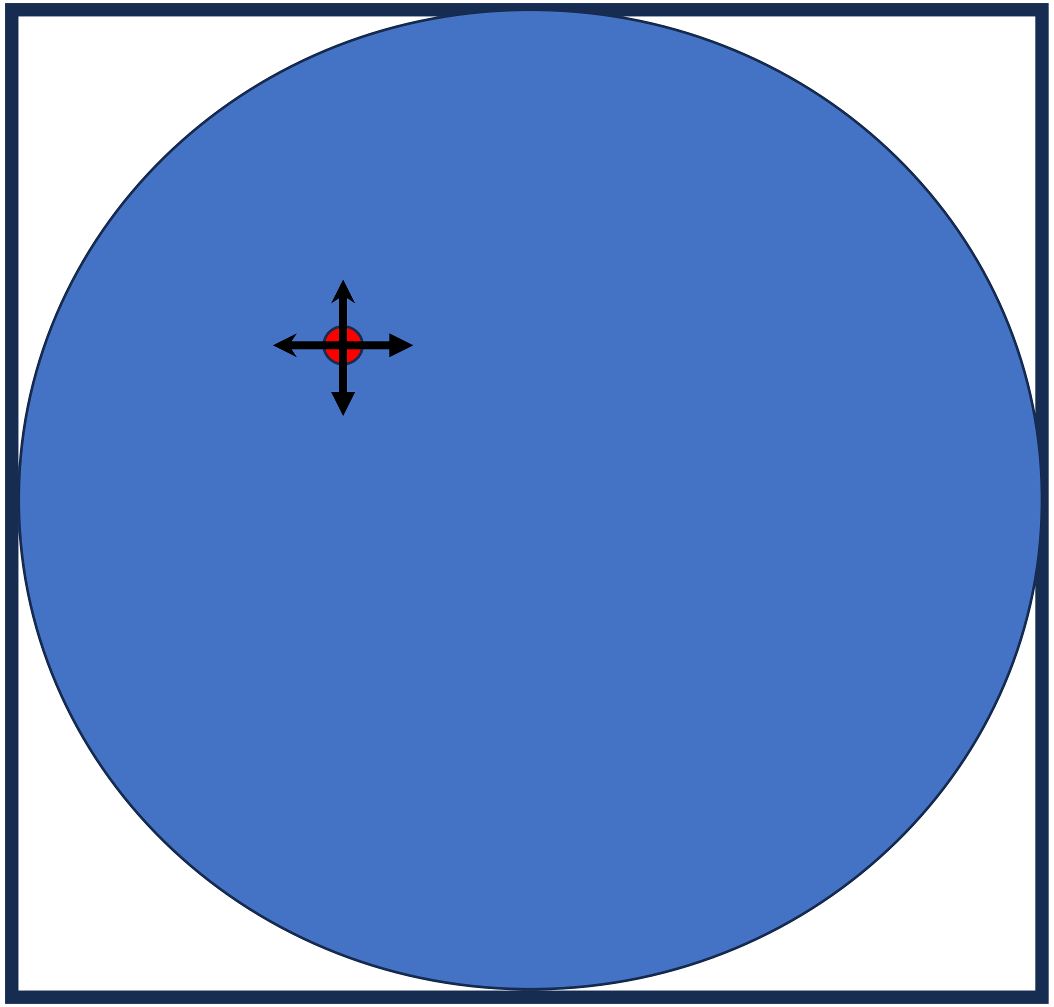
优化的布冯投针算法思路

- 初始化变量：$N_1 = 0, N_2 = 0$;
- 产生[0,1]之间的随机数x,这作为中心点的坐标。此为步骤0.
- 在[0,1]之间分别产生两个随机数x与y，此为步骤1.
- 计算$r = \left(x^2 + y^2\right)^{0.5}$;
- 如果$r > 1$, 回到步骤1重新执行。
- 计算左端点$x_1 = x - \frac{1}{2} * \frac{x}{r}$
- 计算右端点$x_2 = x + \frac{1}{2} * \frac{x}{r}$
- $N_2 \leftarrow N_2 + 1$. 若$(x_1 < 0 \,\&\&\, x_2 > 0)$或者 $(x_1 < 1 \&\& \, x_2 > 1)$ 则 $N_1 \leftarrow N_1 + 1$
- 回到步骤0，重复之前步骤，直到重复N 次。
- 根据公式 $\pi \approx \frac{2}{N_1} * N_2$来估算$\pi$

代码与结果

步数	结果
100	3.125
1000	3.2
10000	3.1496
100000	3.1350

概率算法2：随机游走（马尔可夫链抽样）



假如圆的面积过大（圆形球场！）或者你的力气太小

采取投石（投点）法

在某个点的位置时，前后投的距离随机分布在$(-\epsilon, \epsilon)$，并且左右投的距离也随机分布在$(-\epsilon, \epsilon)$。前后与左右投是不相关的。此距离比圆和方形的尺寸要小很多。

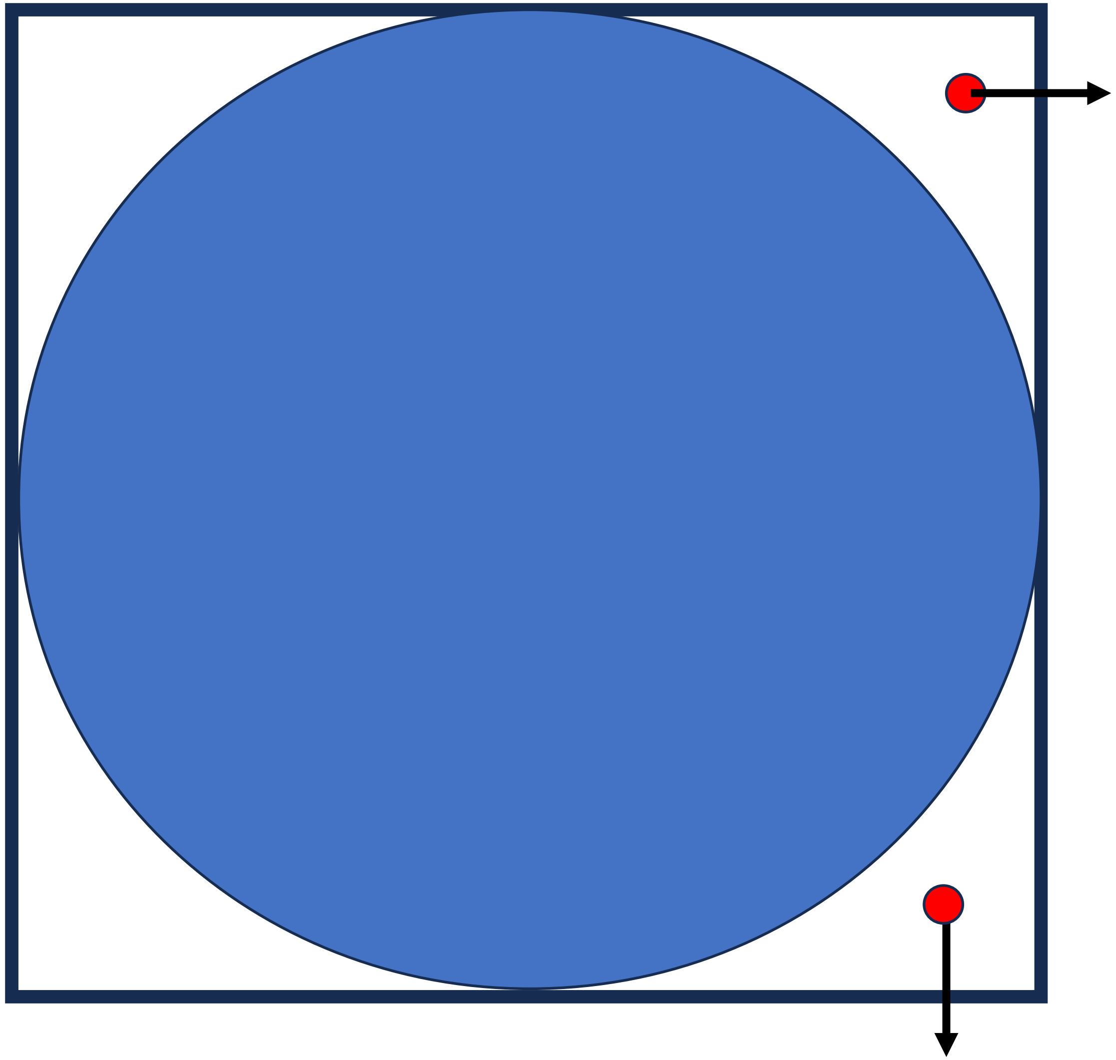
投好点之后移动到那个点，然后接着投点。

这是一个马尔可夫过程，也即下一步的位置只与当前步相关（虽然当前步与前一步相关）。

概率算法2：算法思路

- 从区域内随机某个点出发。
- 按照随机规则往左右投，再往前后投。
- 走到新的位置。如果此位置在圆内则计数器加1.
- 重复N步。
- 用计数器的值除以N，此即为点在圆内的概率。

概率算法2：关键问题—边界处理



在靠近边界的地方随机投石，发现石头新位置出了边界，怎么办？？？

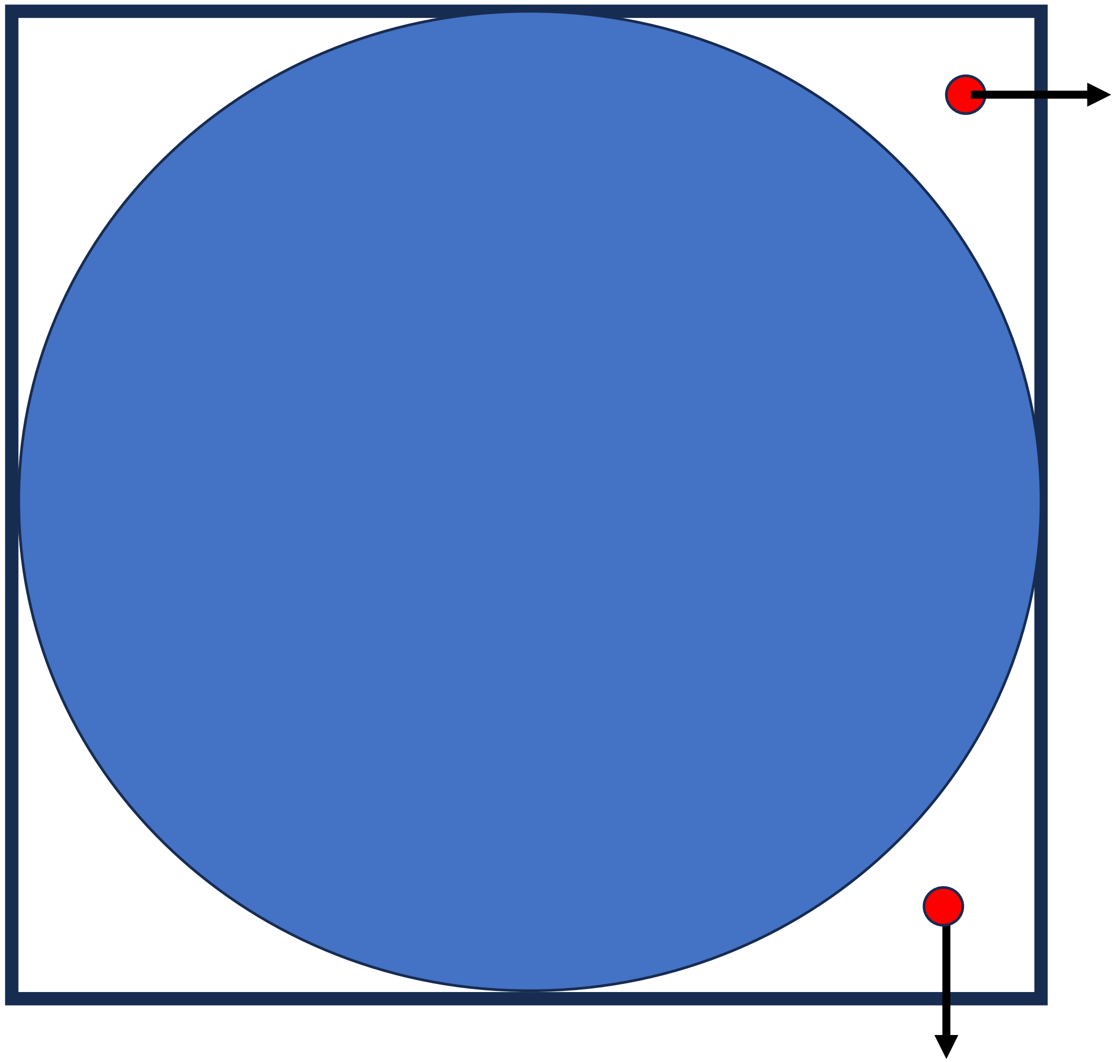
策略1：走到新位置再投，和前面一样。

策略2：将此次投点记为无效，不计入总投点数，然后接着投，直至不超出边界。

策略3：此投点仍有效，仍记入总投点数（堆石法），但位置不变继续投，直至位置可以变化。

哪个是正确的？？

概率算法2：关键问题—边界处理



在靠近边界的地方随机投石，发现石头新位置出了边界，怎么办？？？

策略1：走到新位置再投，和前面一样。

策略2：将此次投点记为无效，不计入总投点数，然后接着投，直至不超出边界。

策略3：此投点仍有效，仍记入总投点数（堆石法），但位置不变继续投，直至位置可以变化。

这是Metropolis算法，其本质是细致平衡条件。

概率算法2：算法步骤

- 初始化循环次数N。初始化投石最大距离ϵ. 初始化计数器$Nh = 0$; 获得初始位置(x, y), $x, y = (-1,1)$内的两个不相关随机数。

- 左右方向投$dx = (-\epsilon, \epsilon)$内的随机数；前后方向投$dy = (-\epsilon, \epsilon)$内的随机数.此步即为步骤1.

- 判断新的位置：若$|x + dx| < 1\&\&|y + dy| < 1$, 则$(x, y) \leftarrow (x + dx, y + dy)$。否则跳至步骤1.
- 判断是否在圆内：若$x^2 + y^2 < 1$则Nh $\leftarrow Nh + 1$.跳至步骤1.
- 循环执行N次。
- 用计数器Nh的值除以N，此即为点在圆内的概率。

代码与结果

randomwalk.m

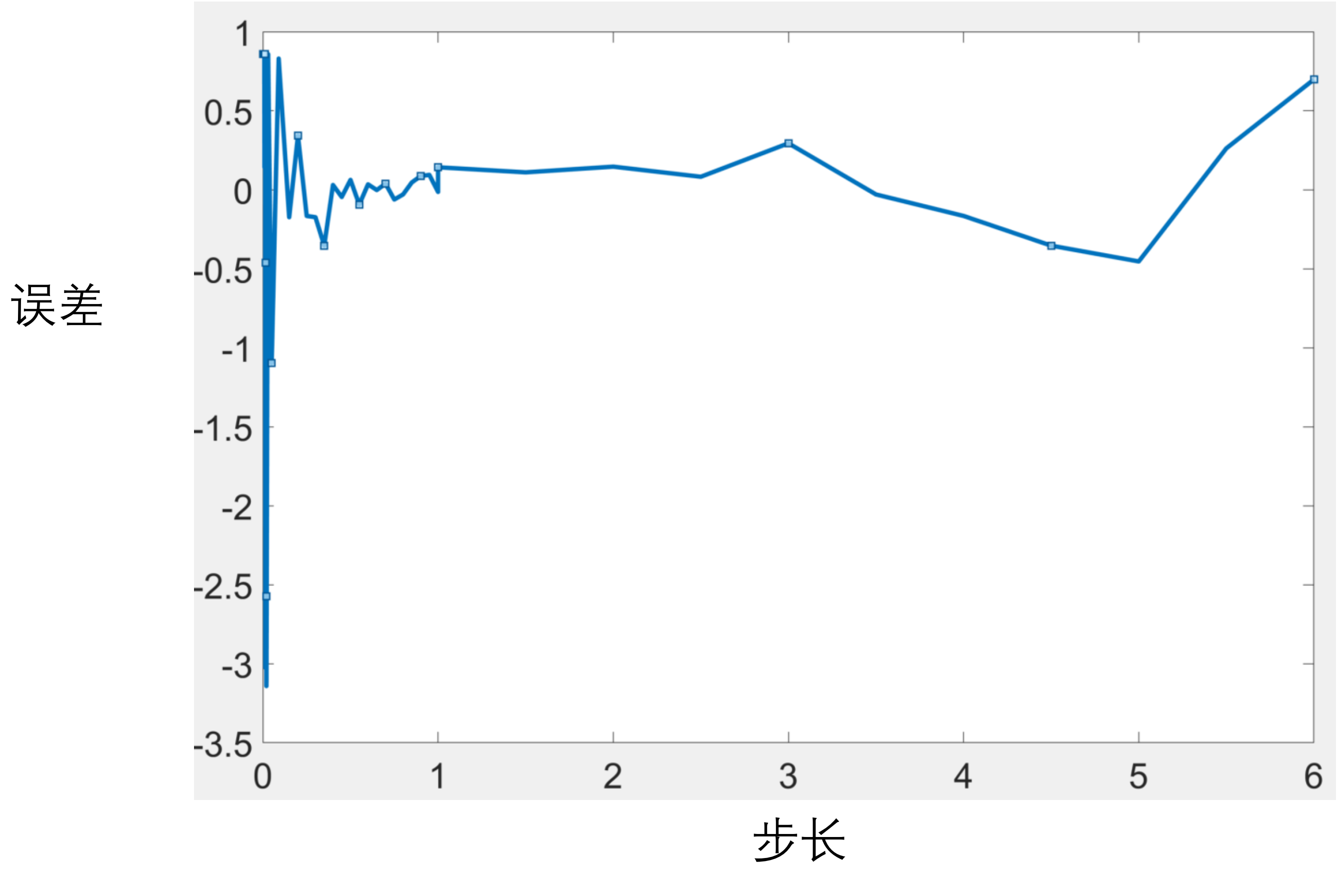
```
function [px,py,res] = randomwalk(N,epsl)
%RANDOMWALK 此处显示有关此函数的摘要
%   此处显示详细说明

Nh=0;
x=rand*2-1;
y=2*rand-1;
px=zeros(1,N+1);
py=px;
px(1)=x;
py(1)=y;

for i=1:N
     dx=2*epsl*rand-epsl;
     dy=2*epsl*rand-epsl;
     px(i+1)=x;
     py(i+1)=y;
     if abs(x+dx)<1 && abs(y+dy)<1
         x=x+dx;
         y=y+dy;
         px(i+1)=x;
         py(i+1)=y;
     end
     if x^2+y^2<1
             Nh=Nh+1;
     end
end

res=4*Nh/N;

end
```

投石距离为$\epsilon = 0.3$

步数	结果
1000	3.4320
10000	3.0968
100000	3.0903
1000000	3.1336

概率算法2：步长（投石距离）选择

- 步长不可太大，否则投点很容易出界，没有有效的进圆点。
- 步长不可太小，否则投点会局域在初始点附近，无法在整个区域均匀分布。



N=1000

概率算法2：初始值问题

- 算法里，初始值是在正方形里随机生成的。
- 这种方法和随机游走的内核是不相符的：如果可以在正方形里随机生成，那用直接抽样法即可，无须随机游走。随机游走应处理不能直接抽样的情形。
- 策略1：人为定规则选某个初始值，其唯一的目的只是这个点是合理的（在正方形区域内）
- 策略2：蒙卡算法一般是嵌套进更大的问题中的，我们可以用前面代码给出的某个位置做初始位点。

随机游走与直接抽样数学内涵相同

- 观测量$O(x,y)$，标记在圆内或外，当点落在(x,y)上，若在圆内则$O(x,y)=1$；否则$O(x,y)=0$.
- 点有一个确定的概率分布$p(x,y)$
- 我们需要观测量的期望值

$$\frac{N_{hit}}{N_{tot}} = \frac{1}{N_{tot}}\sum_i O_i \approx \langle O\rangle = \frac{\int_{-1}^{1}\int_{-1}^{1} dxdy\, p(x,y)O(x,y)}{\int_{-1}^{1}\int_{-1}^{1} dxdy\, p(x,y)}$$

概率方法计算e

另一个基本数学常数：e. 超越数.
是否有概率算法进行估算?

如下方法可以:
考虑满足如下条件的最小的n，$\sum_i^n r_i > 1$,其中r_i是n个在[0,1]区间上的随机数，则n的期望值为e

算法思路

1 设定步数N。生成长度为N的数组a，并将其所有元素初始化为零。

2 设置计数器Sout=1;

3 设置S=0；设置计数器Nh=0;

4 生成[0,1]区间的随机数r1。$S \leftarrow S + r1$

5 当S<=1时，$\mathrm{Nh} \leftarrow Nh + 1$，并回到步骤4继续执行；否则$a(Sout) \leftarrow Nh, \mathrm{Sout} \leftarrow Sout + 1$；$\mathrm{Sout} > \mathrm{N}$,则跳出循环进行第6步，否则回到步骤3继续执行。

6 输出结果:将a数组所有元素加和并除以N。

代码与结果

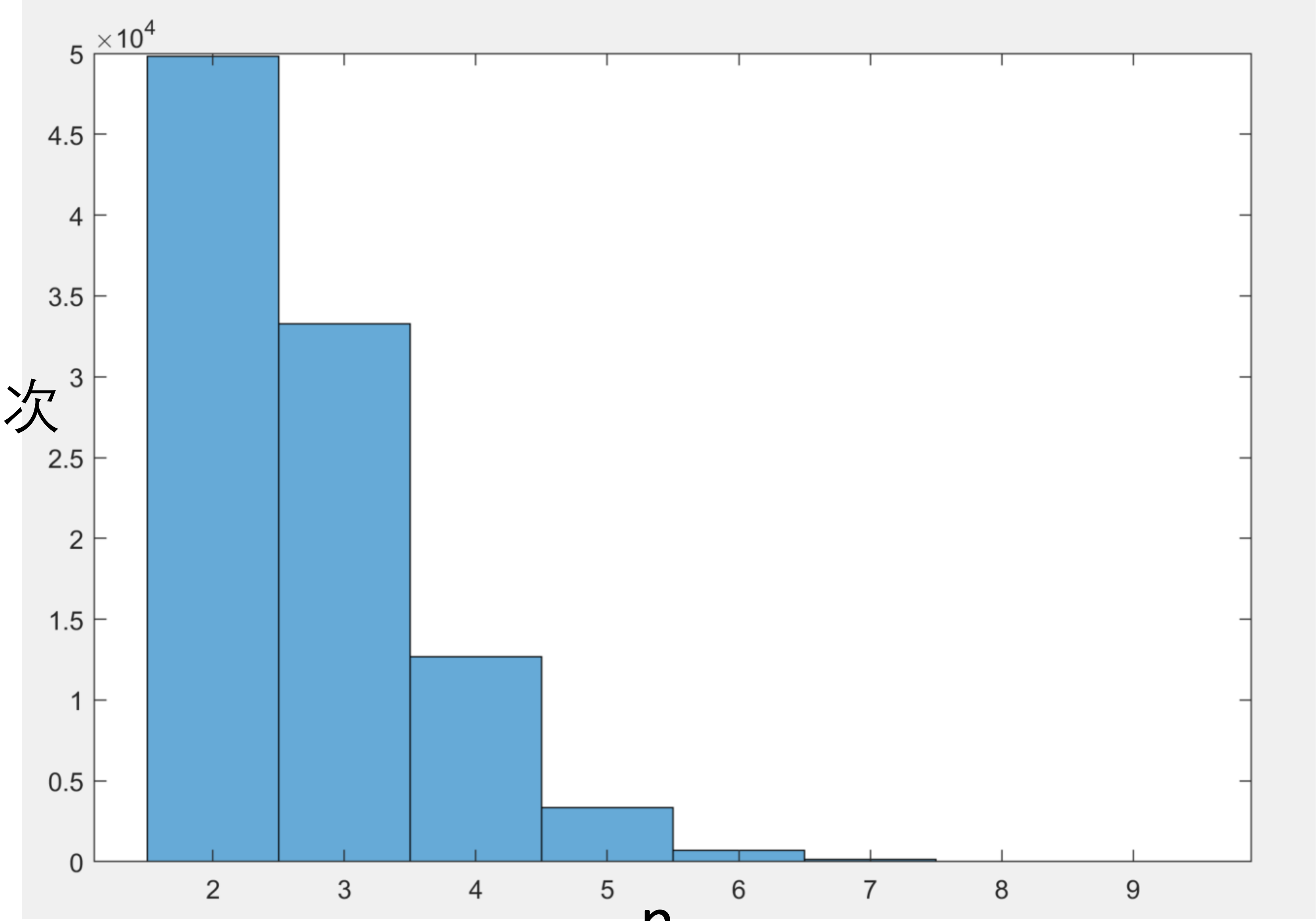
estimatee.m

```
function [a,res] = estimatee(N)
%ESTIMATEE 此处显示有关此函数的摘要
%   此处显示详细说明

a=zeros(1,N);
for i=1:N
    S=0;
    Nh=0;
    while S<1
        r1=rand;
        S=S+r1;
        Nh=Nh+1;
    end
    a(i)=Nh;
end
res=sum(a)/N;


end
```

步数	结果
1000	2.7260
10000	2.7262
100000	2.7163
1000000	2.7187

分布直方图

频次



n

样本总量为100000

作业

1. 请利用软件里的随机数生成代码，设定N=3，按照第16页的算法思路执行三步，并手动计算出π的值。

2. 请将16页的算法写成代码，并分别设定N=100，N=1000，N=10000获得π的值。

3. （附加）证明课件中给出的估算e的算法是正确的，也即证明此方法定义的n的期望值确实是e。