割圆术与蒙特卡洛模拟

数学科学学院

PB18010496 杨乐园

序言

“在一切平面图形中，圆是最美的。”而谈到圆必然离不开圆周率π。秦汉以前，人们以“径一周三”作为圆周率，也即“古率”；但随着更多人的独自测量，改善为“圆径一周三有余”。而更为精确的圆周率的值在古代却难以求得。众所周知，中国古代数学家刘徽（公元 223~295 年）在《九章算术》中早在三国时期便率先提出了割圆术，并得到了近似值$\pi \approx 3.14159$，并且提出可以用两个分数作为圆周率的近似值$\frac{157}{50} \approx 3.14$和$\frac{3927}{1250} \approx 3.1416$，而且主张在实用计算中用 3.14 作为圆周率，后人称之为“徽率”。祖冲之（公元 429~500 年）在割圆术基础上计算得到$3.1415926 < \pi < 3.1415927$，这一结果是我国古代数学辉煌成就之一，并且该计算精度记录保持了 100 多年才被中亚数学家阿尔·卡西所突破。

诱于本人最近学习概率论的蒙特卡洛模拟逼近圆周率π，以及中国古代数学史有关圆周率的辉煌成就，本人通过参考文献学习了割圆术，将通过 Mathematica 进行模拟割圆术以及蒙特卡洛模拟过程求解π的近似值。

实验内容

首先我们了解一下什么是割圆术。

刘徽割圆术思想是用圆内接正多边形的面积来逼近圆的面积。首先在半径为1的圆中作圆的内接正六边形，其边长为$a_0=1$，再在该正六边形的基础上作圆内接正12边形，其边长为a_1，依次下，并设a_n为圆内接正6×2^n边形的边长，S_n为圆内接正6×2^n边形的面积，即得一系列内接正多边形的面积。直觉上，当n越大，内接正多边形与圆的差别便越小，从而S_n作圆的面积的近似值也就越精确；当n无限增大时，即得$\lim\limits_{n\to\infty} S_n=\pi$。刘徽精彩的总结到“割之弥细，所失弥少，割之又割，以至于不可割，而无所失矣。”

通过计算可以得到

$$a_n=\sqrt{2-2\sqrt{1-(\frac{a_{n-1}}{2})^2}}$$

$$S_n=3\cdot 2^{n-1}\cdot a_{n-1}$$

设S为单位圆的面积，则有

$$S_n<S=\pi<S_n+(S_n-S_{n-1})$$

First Part：用割圆术迭代公式计算π的近似值。

程序代码如下：

```
a[n_]:=a[n]=Sqrt[2-2 Sqrt[1-(a[n-1]/2)^2]]
a[0]=1;
s[n_]:=3 2^(n-1)*a[n-1];
t=Table[{n,6*2^n,N[s[n],20],N[2 s[n+1]-s[n],20]},{n,1,20}];
t1=Prepend[t,{"分割数","边数","下界","上界"}];
GridBox[t1,ColumnAlignments->Left,GridBoxDividers->{"Rows"->{{True}},"Columns"->{{True}}}]//DisplayForm
```

运行程序得到如下输出表格：

分割数	边数	下界	上界
1	12	3.0000000000000000000	3.2116570824604982964
2	24	3.1058285412302491482	3.1594286853322272461
3	48	3.1326286132812381972	3.1460717928124962171
4	96	3.1393502030468672071	3.1427136987341520691
5	192	3.1410319508905096381	3.1418729936804145128
6	384	3.1414524722854620755	3.1416627435382532156
7	768	3.1415576079118576455	3.1416101763847791718
8	1536	3.1415838921483184087	3.1415970343077817828
9	3072	3.1415904632280500957	3.1415937487704930054
10	6144	3.1415921059992715505	3.1415929273850433446
11	12288	3.1415925166921574476	3.1415927220386104628
12	24576	3.1415926193653839552	3.1415926707019978382
13	49152	3.1415926450336908967	3.1415926578678444067
14	98304	3.1415926514507676517	3.1415926546593060317
15	196608	3.1415926530550368417	3.1415926538571714368
16	393216	3.1415926534561041393	3.1415926536566377881
17	786432	3.1415926535563709637	3.1415926536065043759
18	1572864	3.1415926535814376698	3.1415926535939710228
19	3145728	3.1415926535877043463	3.1415926535908376846
20	6291456	3.1415926535892710154	3.1415926535900543500

而当运行该程序，当 n=48 时，得到π的值介于
3.14159265358979323846264338327和3.14159265358979323846264338328
之间。

Second Part：改进割圆术迭代，提高计算精度。

由面积计算公式有

$$S_n = 3 \cdot 2^{n-1} \cdot a_{n-1} = 3 \cdot 2^{n-1} \cdot \sin\frac{\pi}{3 \cdot 2^{n-1}}$$

并根据重要极限$\lim\limits_{x\to 0}\frac{\sin x}{x} = 1$知，$\lim\limits_{n\to\infty} S_n = \pi$。记$b_0 = S_0, b_n = S_n - S_{n-1}$(n=1, 2, …)
则有

$$\pi = \sum_{n=0}^{\infty} b_n$$

其中S_n是该级数的部分和。

下面分析该级数通项的渐进态。令$x_n = 3 \cdot 2^{n-1}$，则

$$b_n = S_n - S_{n-1} = x_n\left(\sin\frac{\pi}{x_n} - \frac{1}{2}\sin\frac{2\pi}{x_n}\right) = x_n \sin\frac{\pi}{x_n}\left(1 - \cos\frac{\pi}{x_n}\right)$$
$$\sim \frac{\pi^3}{2}\frac{1}{x_n^2} = \frac{2\pi^3}{9}(\frac{1}{4})^n, \quad as\ n \to \infty$$

这说明级数$\sum_{n=0}^{\infty} b_n$的渐进态相当于几何级数。

下面再分析级数的余项。由$\lim\limits_{x\to 0}\frac{x-\sin x}{x^3} = 1$知，所以当$n \to \infty$时，有

$$r_n = \mathrm{S} - S_n = \pi\left(1 - \frac{\sin\frac{\pi}{x_n}}{\frac{\pi}{x_n}}\right) \sim \frac{\pi^3}{6}\frac{1}{x_n^2} = \sim\frac{2\pi^3}{27}(\frac{1}{4})^n$$

从而即知，$r_{n+1} \approx \frac{1}{4}r_n$，且$r_n \sim \frac{1}{3}b_n$。于是，即有

$$\pi = S_n + r_n \approx S_n + \frac{1}{3}b_n = S_n + \frac{1}{3}(S_n - S_{n-1}) = S_{n-1} + \frac{4}{3}(S_n - S_{n-1})$$

从而可以利用上述公式提高计算精度与速度，给出π的近似迭代计算值。

程序代码如下：

```
a[n_]:=a[n]=Sqrt[2-2 Sqrt[1-(a[n-1]/2)^2]]
a[0]=1;
s[n_]:=3 2^(n-1)*a[n-1];
tt=Table[{n,6*2^n,N[4/3 s[n+1]-1/3 s[n],18]},{n,1,12}];
时tt1=Prepend[tt,{"分割数","边数","近似值"}];
GridBox[tt1,ColumnAlignments->Left,GridBoxDividers->{"Rows"->{{True}},"Columns"->{{True}}}]//DisplayForm
```

运行结果如下输出表格图：

分割数	边数	近似值
1	12	3.14110472164033220
2	24	3.14156197063156788
3	48	3.14159073296874354
4	96	3.14159253350505712
5	192	3.14159264608377955
6	384	3.14159265312065617
7	768	3.14159265356047200
8	1536	3.14159265358796066
9	3072	3.14159265358967870
10	6144	3.14159265358978608
11	12 288	3.14159265358979279
12	24 576	3.14159265358979321

可以看到，当 n=3 时，就得到了刘徽的结果，这大大减少了运算量。

Third Part：Borwein 二阶迭代算法近似计算π值。

查阅文献过程中，我发现了另一种迭代求解π近似值得计算方法，即 1984 年，Borwein 提出了一个π的近似值的二阶迭代算法，如下：

$$y_0 = \frac{1}{\sqrt{2}},\ \alpha_0 = \frac{1}{2},\ y_n = \frac{1-\sqrt{1-y_n^2}}{1+\sqrt{1-y_n^2}},\ \alpha_{n+1} = (1+y_{n+1})^2\alpha_n - 2^{n+1}y_{n+1},$$

当$n \to \infty$时，有$\alpha_n \to \pi$。

程序代码如下：

```
y[0]=1/Sqrt[2];x[0]=1/2;
y[n_]:=y[n]=(1-Sqrt[1-y[n-1]^2])/(1+Sqrt[1-y[n-1]^2]);
x[n_]:=x[n]=(1+y[n])^2*x[n-1]-2^n*y[n];
Table[N[1/x[n],45],{n,1,5}]
```

运行结果如下图：

```
Out[54]= {2.91421356237309504880168872420969807856967188,
          3.14057925052216824831133126897582331177344024,
          3.14159264621354228214934443198269577431443722,
          3.14159265358979323827951277480186397438122550,
          3.14159265358979323846264338327950288419711468}
```

可以看到，值迭代了 5 次便得到了π的 40 位近似值，速度惊人！

除此之外，Browein 于 1985 年又提出了一个四阶迭代算法，在此，出于实验内容并不偏重于此，并不做详细介绍。

Forth Part：蒙特卡洛模拟计算π的近似值。

1777 年，法国数学家 Bufon 提出用投针实验的方法求圆周率：在平面上画满了相互距离是 d 的平行线，将以长度是 L（L<d）的针任意的投在这个二平面上，问针与这些平行线相交的概率是多少？

设针中心和最近的一条平行线的距离是$x(0 \le x \le d)$，其倾斜角为$\theta(0 \le \theta \le d)$，而针与直线相交的充要条件为$0 \le x \le \frac{L}{2}\sin\theta$。以θ以及x为坐标构成直角坐标

系，得到(θ，x)平面的一个矩形区域，其中的任意一点(θ，x)在投掷过程中都有相同的机会被达到，从而矩形的面积与所有可能发生的事件的总和相对应，再画出曲线$x=\frac{L}{2}\sin\theta$，只有当点落在此曲线下方部分区域或该区域关于直线$x=\frac{d}{2}$的对称区域内时，针与直线相交，而这个区域的面积$2\int_0^{\pi}\frac{L}{2}\sin\theta\,d\theta=2L$，它所对应于所有可能的针与直线相交的事件的总和。于是要求的投掷概率 P 为这两个面积之比，即$P=\frac{2L}{\pi d}$。

所以当投掷次数足够多时，可以通过$\pi=\frac{2L}{Pd}$得到其近似值。

程序代码如下：

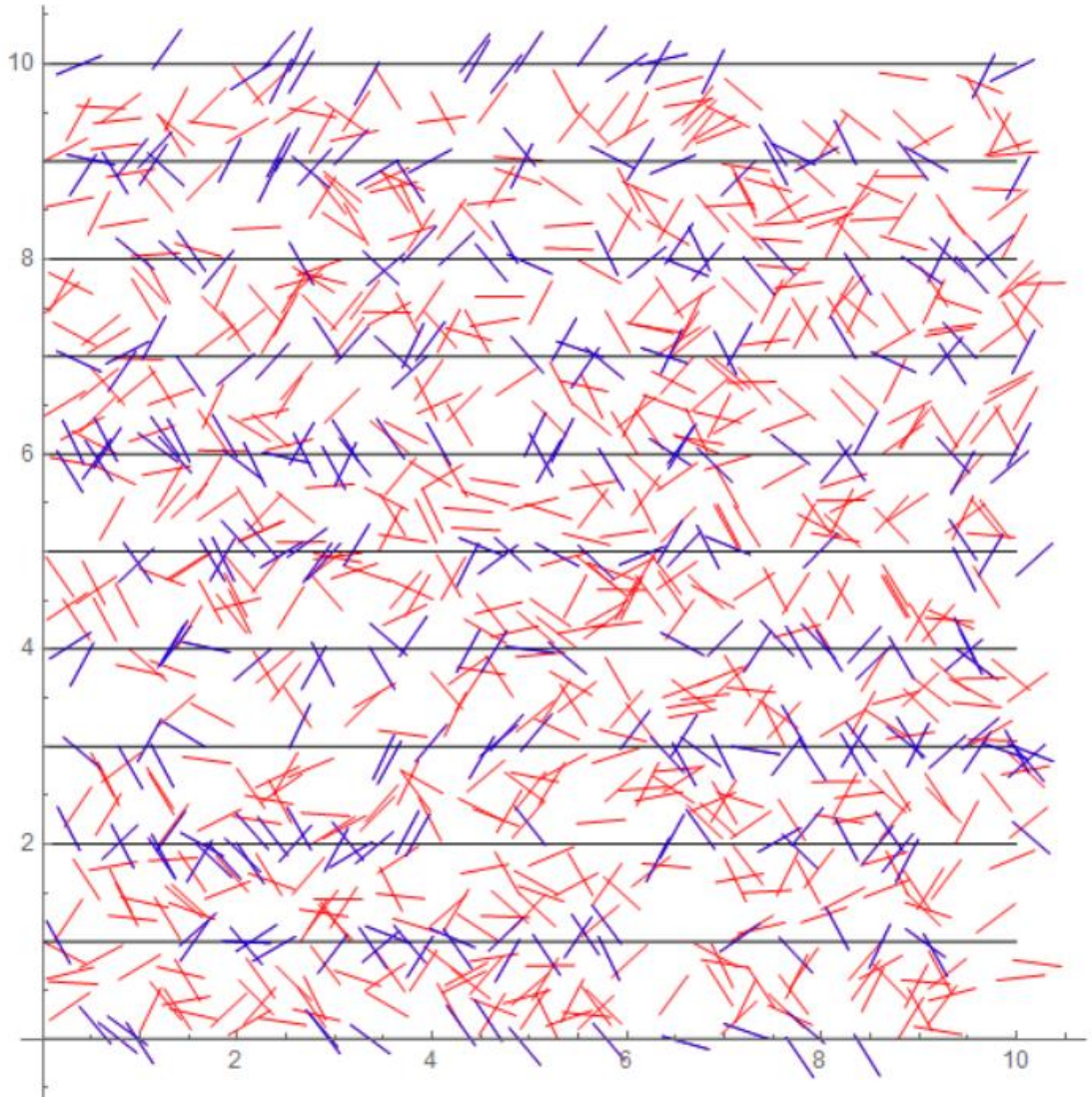
```
sol=Solve[{y-b==k*(x-a),(x-a)^2+(y-b)^2==h^2},{x,y}]//Simplify;
sol1=Flatten[sol];
x[a_,b_,h_,k_]:=x/.sol1[[4]];
y[a_,b_,h_,k_]:=y/.sol1[[3]];
h=1/2;
m=1000;
a=Table[Random[Real,{0,10},6],{n,m}];
b=Table[Random[Real,{0,10},6],{n,m}];
k=Table[Random[Real,{-2.1,2.1},10],{n,m}];
g=Table[Graphics[{RGBColor[1,0,0],Line[{{a[[n]],b[[n]]},{x[a[[n]]
,b[[n]],h,k[[n]]],y[a[[n]],b[[n]],h,k[[n]]]}}]},{n,m}];
r1=Table[Graphics[Line[{{0,k},{10,k}}]],{k,0,10}];(*画线*)
x1[a_,b_,k_,l_]:=-((b-a*k-2*h*l)/k);
r=0;
ls={};
For[n=1,n<m+1,n++,For[l=0,l<11,l++,If[a[[n]]<=x1[a[[n]],b[[n]],k[
[n]],l]<=x[a[[n]],b[[n]],h,k[[n]]],r=r+1;ls=Append[ls,{{a[[n]],b[[n]]
},{x[a[[n]],b[[n]],h,k[[n]]],y[a[[n]],b[[n]],h,k[[n]]]}}]]]];
lss=Flatten[ls];
s=Length[lss];
lp=Table[Graphics[{RGBColor[0,0,1],Line[{{lss[[4 k+1]],lss[[4
k+2]]},{lss[[4 k+3]],lss[[4 k+4]]}}]}],{k,0,s/4-1}];
Print["m=",m]
Print["r=",r]
Print["P=",m/r//N]
Show[g,r1,lp,Axes->True,AspectRatio->Automatic]
```

运行结果如下图：

m=1000

r=318

P=3.14465



首先画出一系列相距为 d=1 的平行线，再随机投掷长为 L=0.5 的线段，记投掷总次数为 n，线段与平行直线相交的次数为 m，则由上可知，扔一定次数后$\frac{n}{m}$即为圆周率π的近似值。

我们可以看到，输出某 1000 次投掷的到的π的近似值为 3.14465。

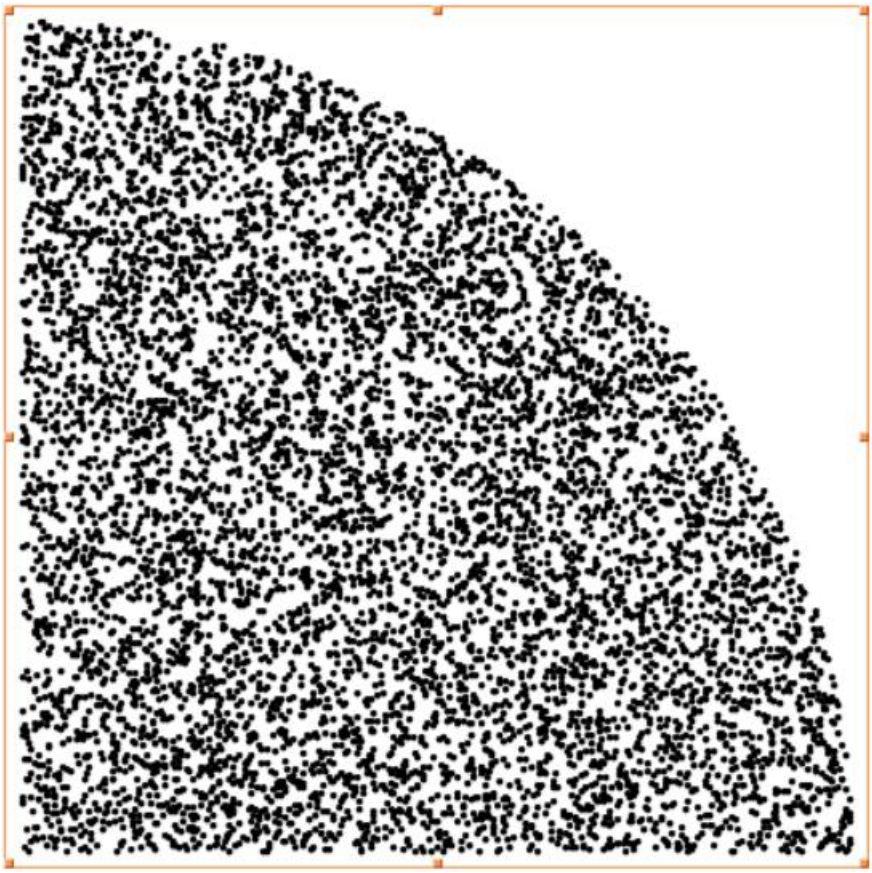
Fifth Part：类似给出推广的蒙特卡洛模拟。

已知$\frac{1}{4}$单位圆的面积为$\frac{\pi}{4}$，据此可向边长为 1 的正方形中随机投点，那么落在$\frac{1}{4}$的圆的概率即为$\frac{\pi}{4}$。则在[0,1]区间随机取两个随机数 x,y 组成点（x,y），重复 n 次，设落在$\frac{1}{4}$圆内的次数为 m 次，则得到π的近似值为$\frac{4m}{n}$。

程序代码如下：

```
a={};
n=10000;
Table[a1={Random[Real,{0,1}],Random[Real,{0,1}]};If[a1[[1]]^2+a1[
[2]]^2<=1,a=Append[a,a1]],{i,1,n}];
4 Length[a]/n//N
Show[Graphics[Point[a]]]
```

运行输出结果为：



该运行输出结果为 3.1496。

总结

通过五个实验的对比，从最初的古代割圆术，到适当的外推进行提高估算速度与精度，再到 Browein 提出的二阶与四阶近似估算，这种计算的速度是十分惊人的。而再从蒙特卡洛模拟看估算，又提供了一种新的计算方式，虽然在概率上面具有随机性，而且是基于大数据的基础下，虽然计算机给出的随机模拟是伪随机，但我们仍见识到了该方法的奇特之处。