《计算机模拟》

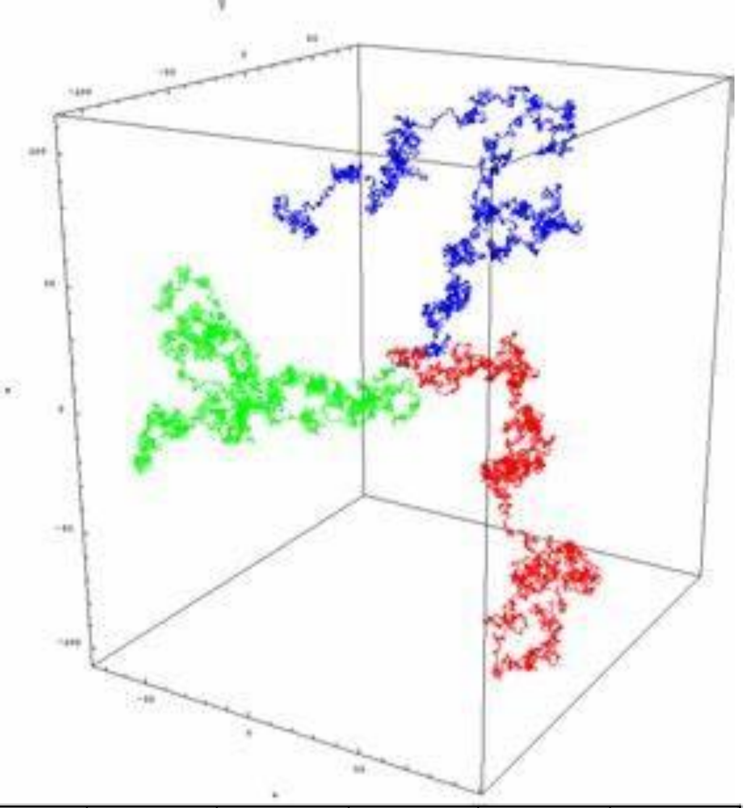
第8讲 – 方程求解

胡贤良
浙江大学数学科学学院

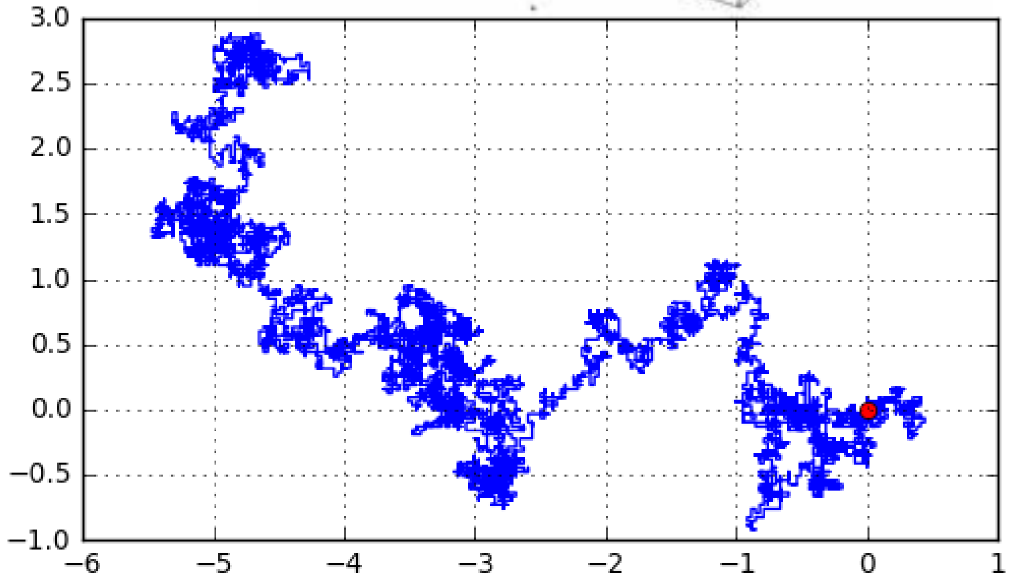
1. 随机游走

1. 布朗运动与扩散现象：Robert Brown(1828)

- Brown: ”对植物的花粉颗粒显微观察的纪要”
- 布朗运动 : 类似于随机游走的无规则移动，它是一种扩散现象！

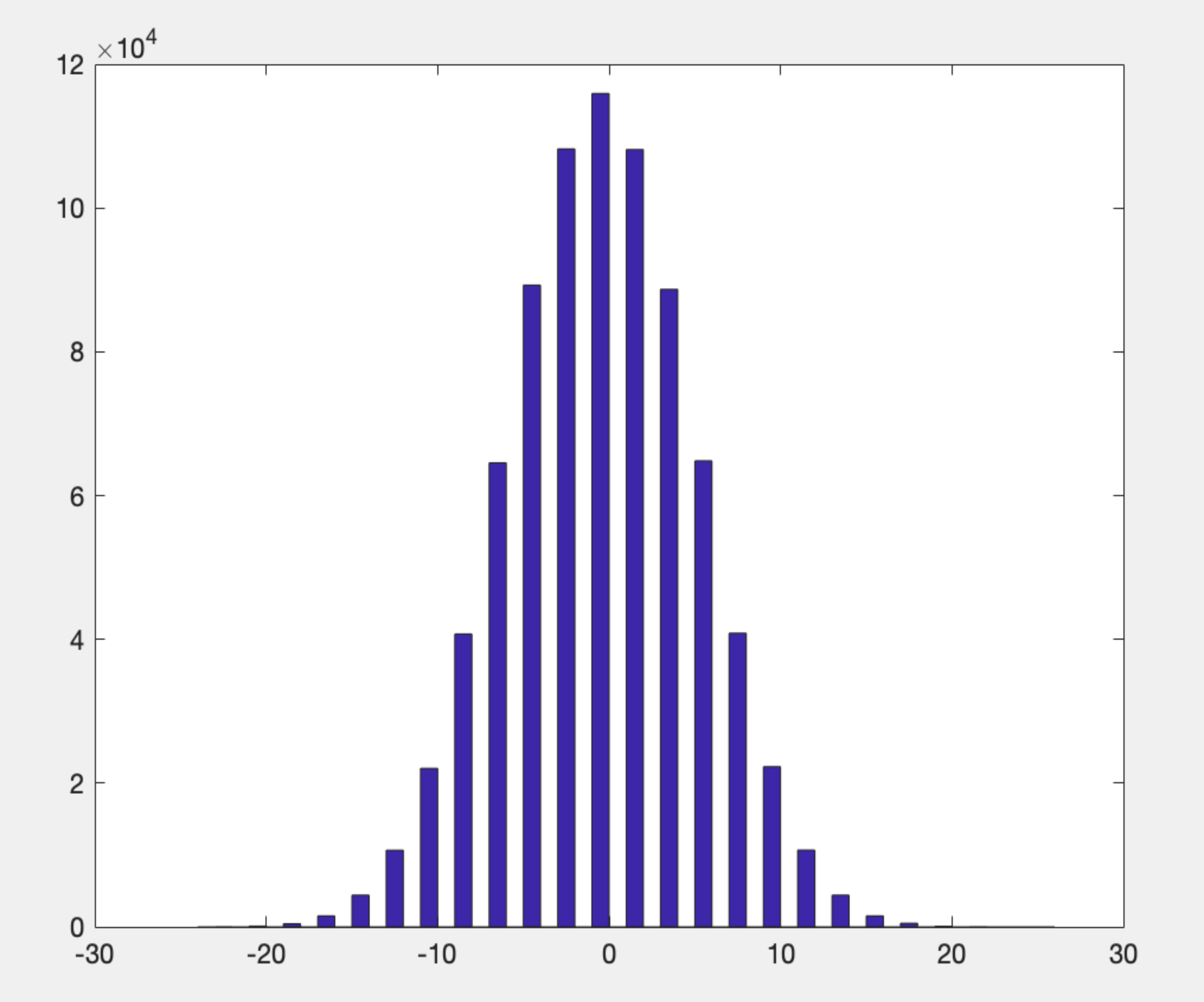


- **问题抽象：单个粒子在一条实线的整数格点上随机游走**。粒子开始于原点并以概率 p 向右走一步，以概率 $q = 1 - p$ 向左走一步，每一步走单位长度。当 $p = q = \frac{1}{2}$时，可以用掷一枚均匀硬币来模拟！



练习：模拟800000次随机游走了30步(ex1.m)

```
ntrials = 80000;
nsteps  = 30;
s = zeros(1,ntrials);
for j = 1:ntrials
    x = 2*(rand(1,nsteps)<0.5) - 1;
    s(j) = sum(x);
end
hist(s,50,-25:1:25)
```



若随机游走的步数(nsteps)很大,并且做了大量的重复实验(ntrials),由中心极限定理，所得粒子直方图将很接近正态分布

随机游走的连续化

- 考虑将时间和空间离散化。设：
  - 时间步Δt($t = n\Delta t$), 空间步位移Δx($x = m\Delta x$)，粒子从原点出发。
  - 在直线上随机游走，以概率 p向右一步$q = 1 - p$向左一步。
- 记$P(m,n)$为粒子经过 n个时间步后处在位置 m 的概率，由排列组合可知：n个时间步中有向右走r次的可能性为二项分布

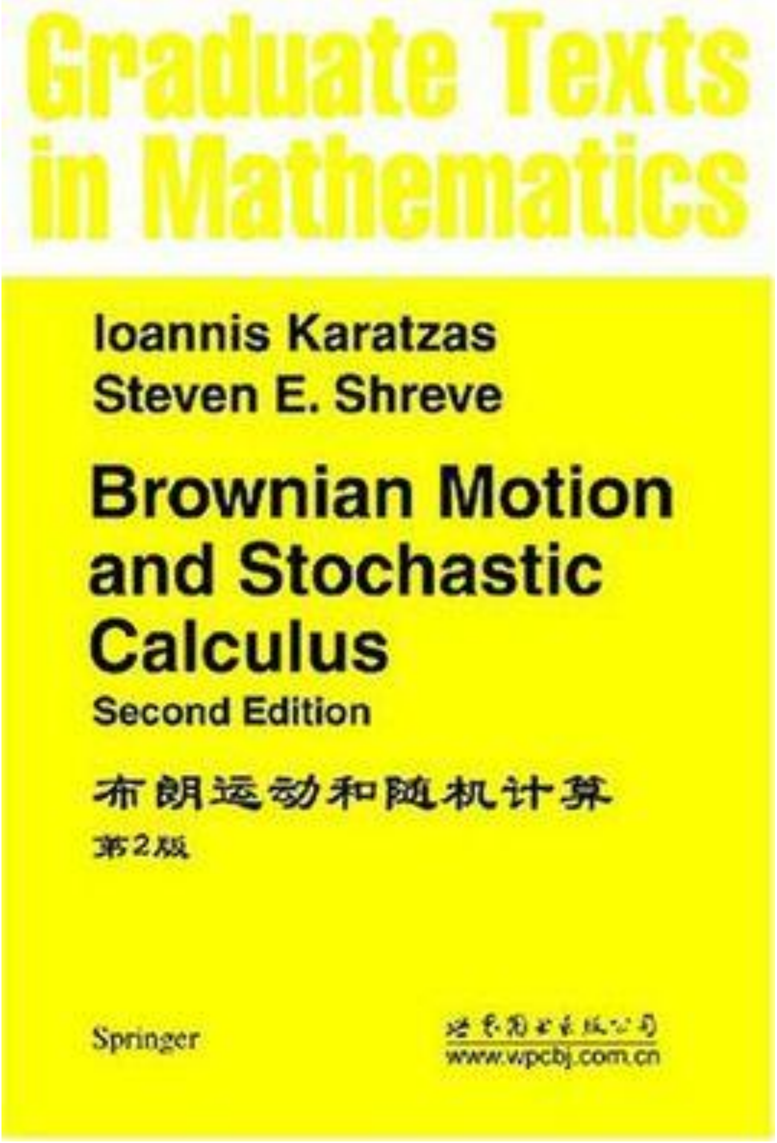
$$P(m,n) = C_n^r p^r q^{n-r}.$$

- 记 r为其中向右的步数， l为其中向左走的步数。则有

$$m = r - l, \qquad n = r + l$$

即

$$r = \frac{1}{2}(n + m), \qquad l = \frac{1}{2}(n - m)$$



随机游走的连续化($P(m,n)=C_n^r p^r q^{n-r}$)

$$\mu = E(m) = E(2r-n) = 2E(r) - n = (2p-1)n$$

$$var(m) = var(2r-n) = 4var(r) = 4npq$$

定义$\sigma = \sqrt{var(m)} = \sqrt{4npq}$。当$p=q=\frac{1}{2}$时，由中心极限定理

$$P(m,n) \approx \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(m-\mu)^2}{2\sigma^2}} m = \frac{1}{\sqrt{2\pi n}} e^{-\frac{m^2}{2n}} m = \frac{1}{\sqrt{2\pi t \frac{(\Delta x)^2}{\Delta t}}} e^{-\frac{x^2}{2t\frac{(\Delta x)^2}{\Delta t}}} \Delta x.$$

这里，$n = \frac{t}{\Delta t}, m = \frac{x}{\Delta x}$。接着，令$\Delta x, \Delta t \to 0$, $D = \frac{\Delta x^2}{\Delta t}$为**扩散系数**，是一个常量

$$p(x,t) = \frac{P(m,n)}{\Delta x} = \frac{1}{\sqrt{2\pi Dt}} e^{-\frac{x^2}{2Dt}}$$

- 函数$p(x,t)$满足某个一维**扩散方程**! (1905, Einstein)

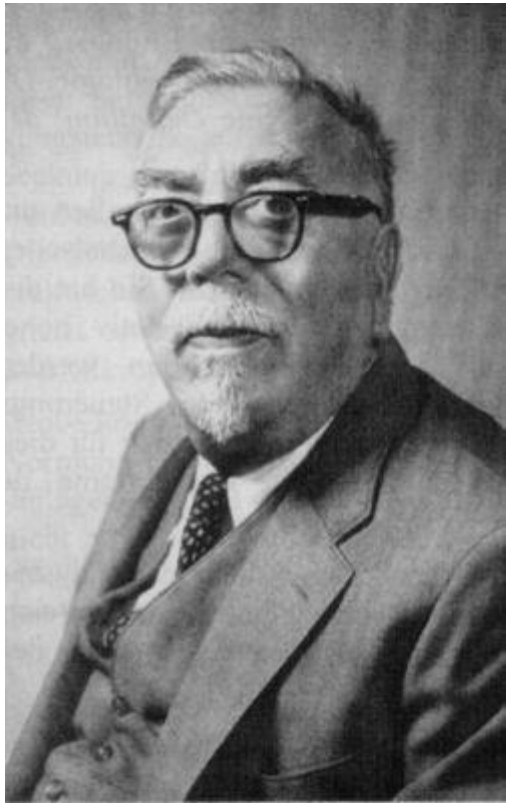
布朗运动的数学模型(Norbert Wiener ,1918)

设W_t是一个随时间t变化的随机变量（这里t为连续变量），如果它满足下面三个条件的话，则它是一个一维布朗运动

（1）$W_0 = 0$；

（2）对于$0 \le t_1 < t_2 < \cdots < t_n$，增量 $W_{t_2} - W_{t_1}, \cdots, W_{t_n} - W_{t_{n-1}}$ 是相互独立的

（3）对于$0 \le s < t$，$W_t - W_s$ 是均值为0，方差为c（t − s）的正态分布，其中$c > 0$是一个固定的常数。当$c = 1$时，它称为标准布朗运动。



Norbert Wiener

- 上述关于布朗运动的定义，可以推广到高维欧氏空间情形！

常返性（收敛性？$\mu=(2p-1)n, \sigma=\sqrt{4npq}$）

当$n\to\infty$,随机游走的粒子将以概率1最终会回到出发点!

一维形式证明：

- 记q_0为从原点出发沿无限长直线随机游走永不返回的概率
- 类似地，记q_i为从$x=i$点出发的随机游走永不返回的概率，故q_i=q

✓由于左右方向的对称性，粒子从原点出发，向右不返回概率是$\frac{q_0}{2}$. 故

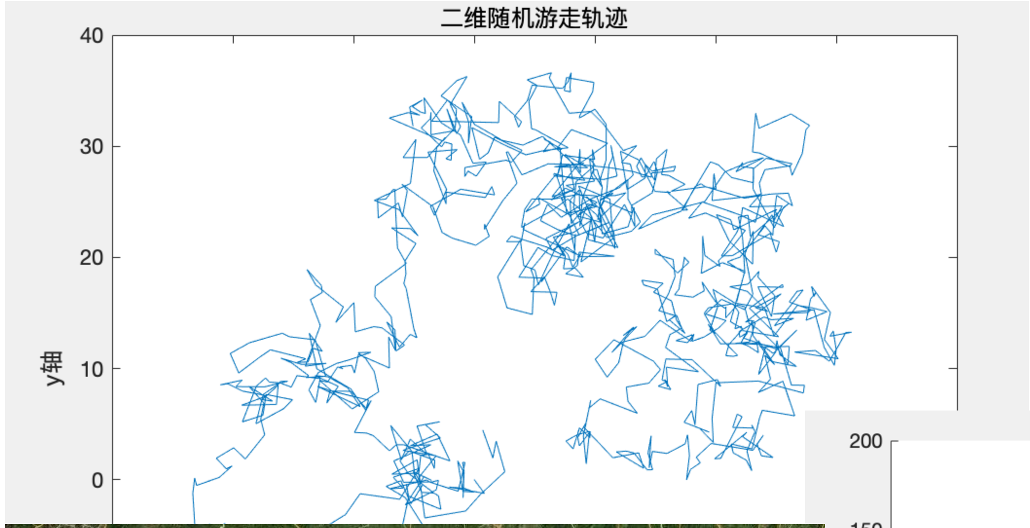
$$\frac{q_0}{2}=\frac{1}{2}\frac{q_1}{2}.$$

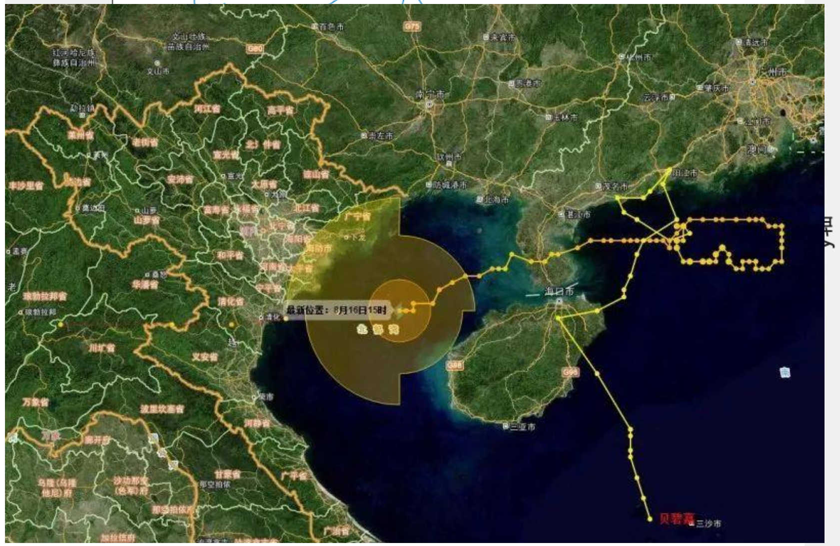
✓由此递推：

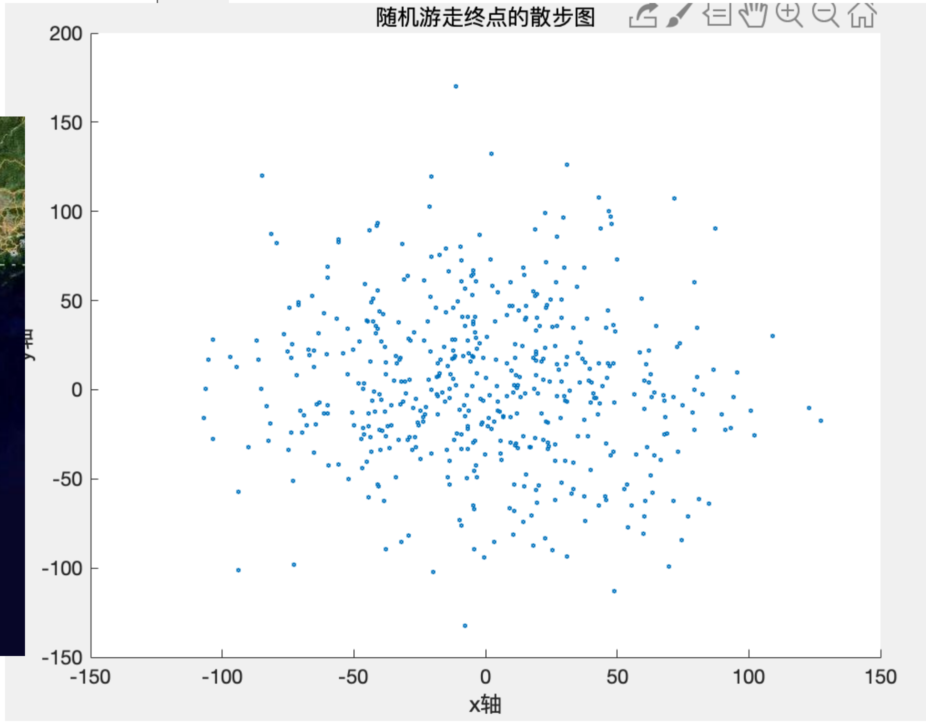
$$\frac{q_0}{2}=\frac{1}{2}\frac{q_1}{2}=\cdots\left(\frac{1}{2}\right)^n\left(\frac{q_n}{2}\right)\to 0.$$

✓注意到 $p_0=1-q_0$为返回原点的概率。从而 $p_0=1-q_0=1$.

模拟二维布朗运动(ex2.m)








```
sig=2;
nsteps=1000;
MU=[0 0];
SIGMA=[sig 0; 0 sig];
rng('shuffle')
R=mvnrnd(MU,SIGMA,nsteps);
x(1)=0;y(1)=0;
for i=2:nsteps
    x(i)=x(i-1)+R(i,1);
    y(i)=y(i-1)+R(i,2);
end
figure(1)
plot(x,y);
title('二维随机游走轨迹');
xlabel('x轴');ylabel('y轴');
nwalks=500;
endx=zeros(nwalks,1);
endy=zeros(nwalks,1);
for j=1:nwalks;
    R=mvnrnd(MU,SIGMA,nsteps);
    x(1)=0;y(1)=0;
    for i=2:nsteps
        x(i)=x(i-1)+R(i,1);
        y(i)=y(i-1)+R(i,2);
    end
    endx(j)=x(nsteps);
    endy(j)=y(nsteps);
end
figure(2);
scatter(endx,endy,3)
title('随机游走终点的散步图');
xlabel('x轴');ylabel('y轴');
```

规则网格上的随机游走，可参考：

https://jingyan.baidu.com/album/a17d5285c492378099c8f24c.html?picindex=3


买入?
卖出?

2. 金融学应用


国外实用金融统计丛书
CRC Press
Monte Carlo Methods and Models in Finance and Insurance
金融学和保险学中的
蒙特卡罗方法与模型
拉尔夫·科恩（Ralf Korn）
埃尔克·科恩（Elke Korn）
【德】 杰拉尔德·克罗桑特（Gerald Kroisandt） 著
陈栖龙 郑志勇 译
机械工业出版社

案例一：期权定价

期权：金融市场的衍生品，分为**看涨期权**和**看跌期权**。

- 看跌期权：期权购买方或持有人与出售方或出单方之间的一项合约，是一种选择权，它允许持有人有权**在未来某约定的时间内**以固定价格卖出一定数量的标的资产给期权的出售方.
- 看涨期权：期权购买方或持有人与出售方或出单方之间的一项合约，是一种选择权，它允许持有人拥有**在有效期内以敲定价**从出售方购买一定数量的标的资产的权利.

后面我们考虑的**标的资产**是**股票**。

- 1900年，法国数学家Louis Bachelier(巴施里耶)提出有效市场假设，用布朗运动描述证券价格的变化
- 1964年，美国经济学家Paul Samuelson (萨谬尔森) 做了修正与推广，期权定价问题得到重视
- **1973年**，美国人Fischer Black(布莱克)，Myron Scholes(肖尔斯) 发表了关于期权定价的论文。同年，期权正式在芝加哥交易所上市

（1）几何布朗运动模拟

假设（有效市场）：在一个有效市场上，股票的价格已经反映了所有已知的信息，因此在很短的时间间隔内股价的变动是完全随机的。即股票的相对变化率（回报）服从正态分布，即股票的回报是一个布朗运动。

➢ 股价的基本演化方程：

$$\frac{\Delta S_t}{S_t} = \boldsymbol{\mu}\Delta t + \boldsymbol{\sigma}\Delta W_t$$

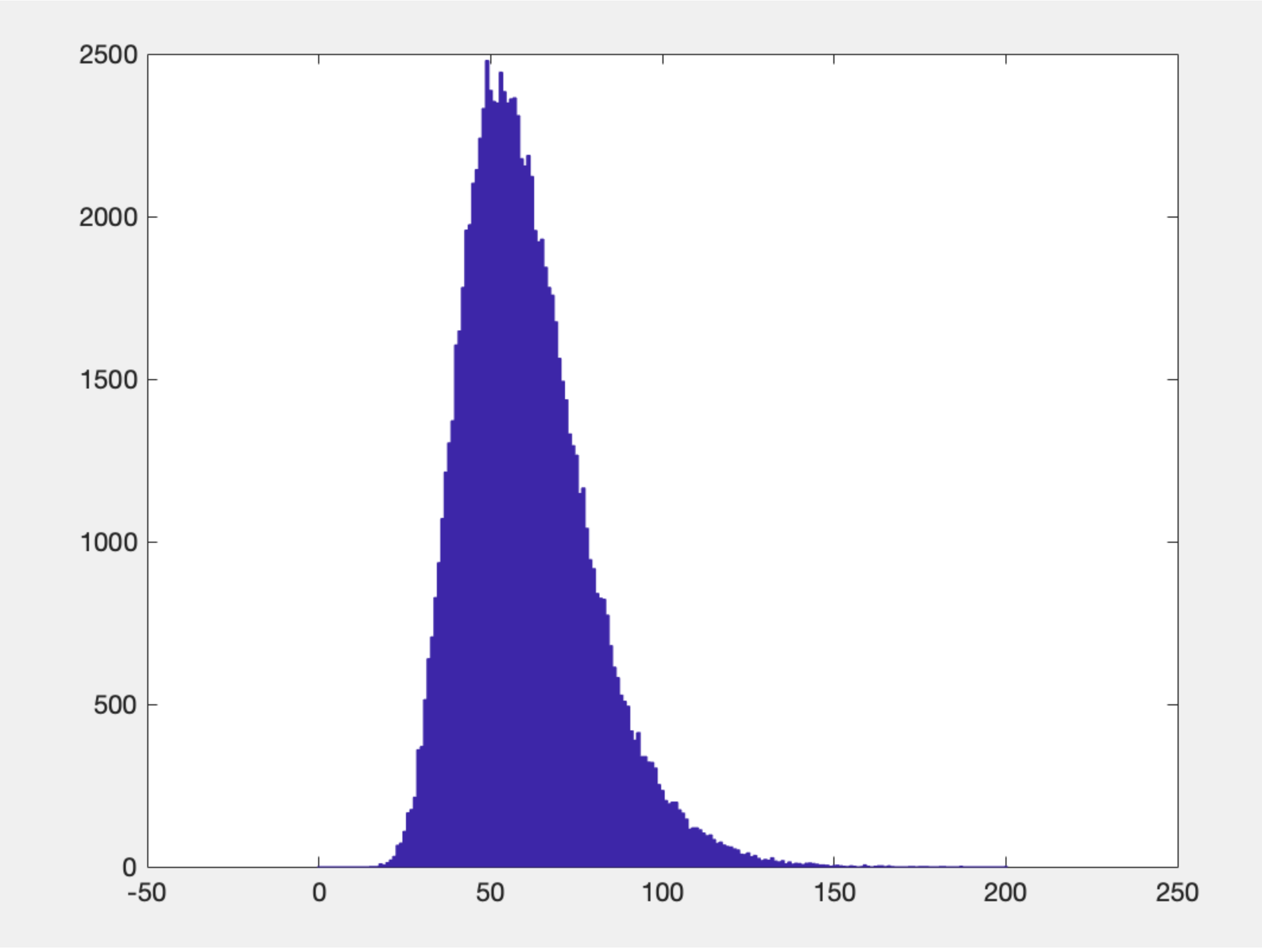
其中，S_t是股票在t时刻的市场价格，Δt是单位时间，ΔS_t为此期间价格的变动，ΔW_t是标准布朗运动的随机变量，μ，σ是股价变动的特征参数，分别被称为**漂移率**和**波动率**。

✓下一步股票价格$S_{t+1} = S_t + \Delta S_t$.

✓遵循上述方程的股价演化被称为几何布朗运动，也称为伊藤(Ito)过程。

（2）风险定价方式：Black-Scholes方程(略,参考$P_{190-191}$)

```
nDays=90;
dt=1/365.0;
T=nDays*dt
S0=20;%股票当前价格
K=25;%敲定价
r=0.031;%无风险利率
sigma=0.6;%波动率
expTerm=r*dt;
stddev=sigma*sqrt(dt);
ntrials=100000;
value=0;
for j=1:ntrials
    n=randn(1, nDays);
    S=60;
    for i=1:nDays
        dS=S*(expTerm + stddev*n(i));
        S=S+dS;
    end
    S90(j)=S;
    value=value + max(S-K, 0);
end
value=value/ntrials;
price=exp(-r*T)*value
hist(S90,0:1:200)
```



模拟该期权第90天的价格(ex3.m)

概率小知识：赌徒是如何破产的？

假设：赌徒下注1个单位的钱与庄家打赌，以概率p赢庄家，以概率q=1-p 输给庄家，并设赌徒开始有x单位的钱。

- X(t)是随机变量，表示赌徒在t时刻的财富，显然$X(0)=x$

$$X(1)=\begin{cases} x+1, & \text{以概率}p, \\ x-1, & \text{以概率}q. \end{cases}$$

与随机游走类似，不同之处是开始之处是x，不为0；

- 假设庄家开始有h单位的钱，双方钱数总和 a = x + h；
- 只要$0<X(t)<a$成立，游戏将继续进行下去，直到不满足终止游戏。

问：游戏能否无限期进行？

设W_x是赌徒以x单位的初始资金玩到破产的概率，Z_x是庄家破产的概率，固定资金总和为a。则有：

$$W_x = qW_{x-1} + pW_{x+1},$$

其中$W_0 = 1$，$W_a = 0$。将上述差分方程的形式解：$W_x = b^x$。

代入上式，可得：

$$b^{x-1}(-q + b - pb^2) = 0$$

从而差分方程的**通解**

$$W_x = \frac{(q/p)^a - (q/p)^x}{(q/p)^a - 1} \qquad p \neq \frac{1}{2}$$

$$W_x = 1 - \frac{x}{a} \qquad p = \frac{1}{2}$$

同理，有：

$$Z_x = \frac{(q/p)^x - 1}{(q/p)^a - 1} \qquad p \neq \frac{1}{2}$$

$$Z_x = \frac{x}{a} \qquad p = \frac{1}{2}$$

- $W_x + Z_x = 1$：意味着从任何数量赌本开始，**要么**赌徒破产，**要么**庄家破产！

游戏的预期持续时间分析

- 赌博的真相：原始资本小的破产概率几乎百分之百！（参考P197）
- 设T(x)表示从初始的x财富直到一方破产的期望时间
- 有递推关系

$$\mathrm{T}(x) = 1 + \frac{1}{2}T(x+1) - \frac{1}{2}T(x-1). \qquad 1 < x < a.$$

- 对于上述非其次差分方程

$$\mathrm{T}(x) = x(a-x).$$

- 易见:当x $= \frac{a}{2}$时，即双方赌注相等的时候游戏期望持续时间最久。

案例二：凯利准则(公式)

设掷硬币的赌局中每单位赌注的赢率为r，赢的概率是p，输的概率为q。记X_N是N次赌局后财富，X_0为初始资金，f为投注份额

➢ 经过一局：$X_1=\begin{cases}X_0+rfX_0=(1+rf)X_0, & 当赢时,\\ X_0-fX_0=(1-f)X_0, & 当输时,\end{cases}:=Q_1X_0,$

其中，$Q_1=\begin{cases}1+rf, & 以概率p,\\ 1-f, & 以概率q.\end{cases}$

➢ 假设凯利游戏无限玩下去，游戏的目的是最大限度提高期望的财富增长率

$$g=\frac{1}{N}\ln\left(\frac{X_N}{X_0}\right),$$

➢此外，该游戏的期望收益为

$$E=pr-q.$$

凯利公式(续)

$$\boldsymbol{Q}=\begin{cases}\boldsymbol{1+rf}, & 以概率\boldsymbol{p}\\ \boldsymbol{1-f}, & 以概率\boldsymbol{q}\end{cases}$$

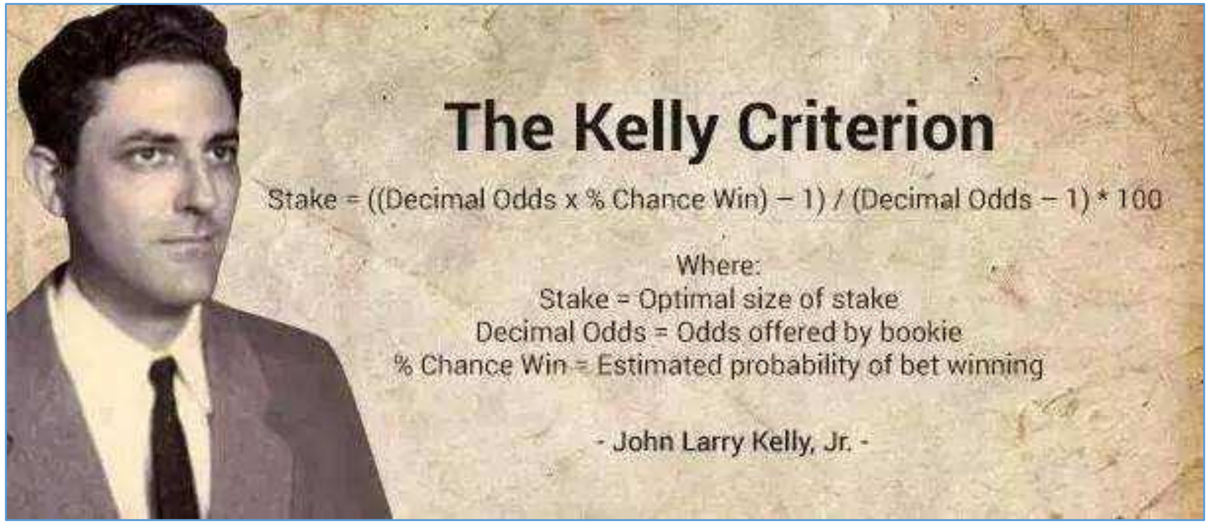
显然$Q_n=\cdots=Q_1=Q$, 从而$X_N=Q^N X_0$

$$g=\frac{1}{N}\ln(Q^N)=lnQ,\qquad E(g)=E(lnQ)=pln(1+rf)+qln(1-f)$$

关于f求上式最大值，求导并令其为0（极值必要条件），可得：

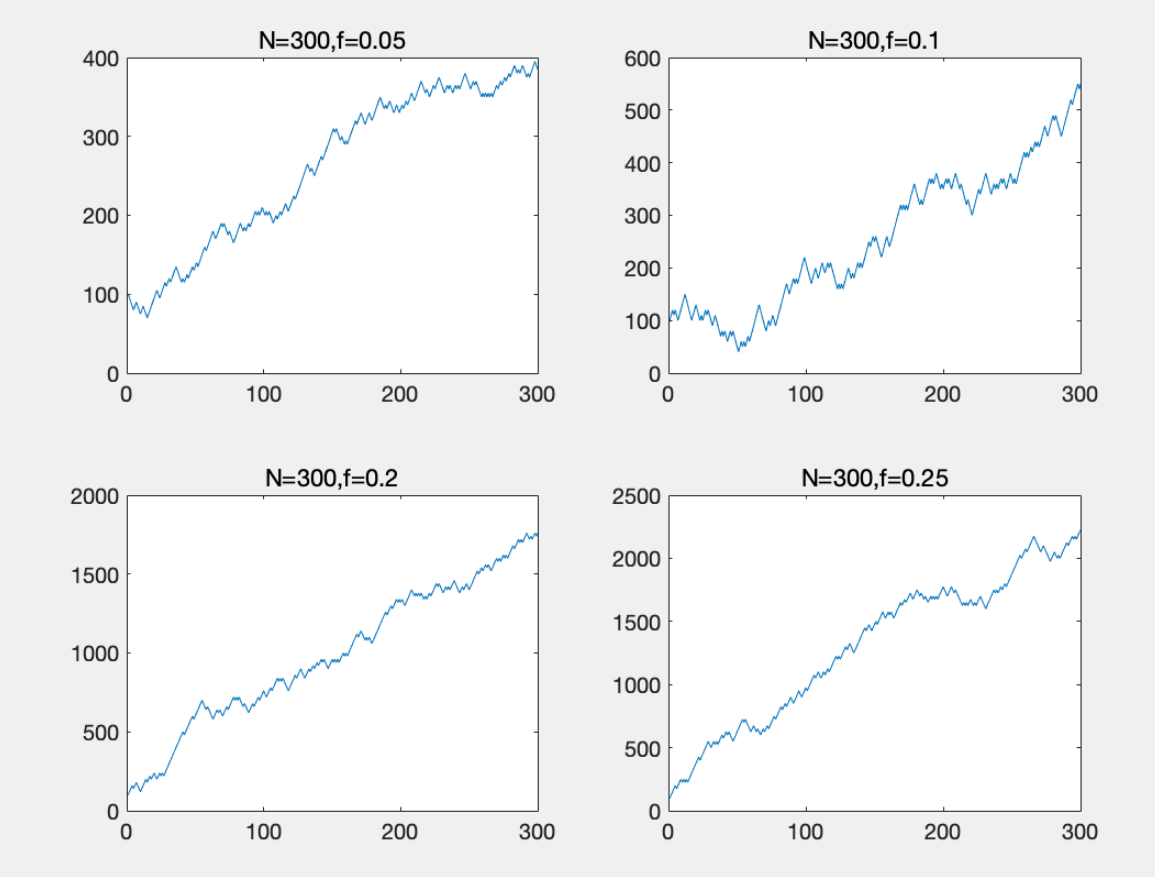
$$\frac{\mathrm{pr}}{1+rf}-\frac{q}{1-f}=0,\qquad f=\frac{pr-q}{r}:=\frac{\boldsymbol{E}}{\boldsymbol{r}}$$

✓此解的含义为：最佳投注比例就等于**期望收益**与**赢率**之比，这就**凯利准则**。



例9.4：不同投注比例的游戏财富模拟(ex4.m)

```
prob = 0.6;
f    = [0.05;0.1;0.2;0.25];
W0   = 100;
N    = 300;
W    = zeros(4,N);
W(:,1) = W0;
for i = 2:N
    r = (rand(4,1) < prob);
    r = W0*(2*r - 1).*f;
    W(:,i) = W(:,i-1) + r;
end
figure;
subplot(2,2,1); plot(W(1,:));  title('N=300,f=0.05');
subplot(2,2,2); plot(W(2,:));  title('N=300,f=0.1');
subplot(2,2,3); plot(W(3,:));  title('N=300,f=0.2');
subplot(2,2,4); plot(W(4,:));  title('N=300,f=0.25');
```



分析：简单游戏中潜在的风险

进一步地，为了让游戏更接近于金融市场的状况，我们修改上面的游戏使其包括出现大损失的可能。考虑投掷两个硬币赌胜负投注。设：

- O_1表示两个硬币出现不同面，对应$p_1=\frac{1}{2}$,
- O_2表示出现两个正面, $p_2=\frac{1}{4}$,
- O_3表示出现两个反面, $p_3=\frac{1}{4}$,
- 游戏的赔率是：O_1为1:3, O_3为3:1

➢ 记

- γ：赢得的金额是赌注的多少倍数，
- λ：输掉的金额是赌注的多少倍数，

显然，上述例子中：γ_1=4，$\lambda_2=1, \lambda_3=4$，

则该游戏的期望收益为：$E=p_1\gamma_1-p_2\lambda_2-p_3\lambda_3$

简单游戏中潜在的风险：考虑最优配置f

$$\boxed{\mathrm{E}-[\mathrm{p}_1\gamma_1(\lambda_2+\lambda_3)+p_2\lambda_2(\gamma_1-\lambda_3)+p_3\lambda_3(\gamma_1-\lambda_2)]f+\gamma_1\lambda_2\lambda_3 f^2=0}$$

分析过程：令f 表示每局游戏中投注资金的最优比例

$$\mathrm{X}_1=\begin{cases} X_0+fX_0\gamma_1=(1+\mathrm{f}\gamma_1)\mathrm{X}_0,\text{当}\mathrm{O}_1\text{时} \\ X_0-fX_0\lambda_2=(1-\mathrm{f}\lambda_2)\mathrm{X}_0,\text{当}\mathrm{O}_2\text{时} \\ X_0-fX_0\lambda_3=(1-\mathrm{f}\lambda_3)\mathrm{X}_0,\text{当}\mathrm{O}_3\text{时} \end{cases}:=\mathrm{Q}X_0$$

与前一游戏类似，则经过n局游戏后财富的期望增长率为：

$$\mathrm{E}(g)=E(lnQ)=\mathrm{p}_1\ln(1+f\gamma_1)+p_2\ln(1-f\lambda_2)+p_3\ln(1-f\lambda_3)$$

其最大值满足如下Euler-Lagrange方程给出

$$E'(g)=\frac{\mathrm{p}_1\gamma_1}{1+f\gamma_1}-\frac{p_2\lambda_2}{1-f\lambda_2}-\frac{p_3\lambda_3}{1-f\lambda_3}=0$$

期权交易应用1：买卖价差策略

这是一种持有两个期权产品构成一种组合的投资策略：卖出某种股票的看涨期权，同时又买入价格较低且到期日相同的同种股票看涨期权。

问题背景: 某投资者采用买卖价差策略进行期权投资。

交易的统计数据报告：这些投资在过去有过75次交易，其中

1. 66次赚钱，平均每笔交易的获益是659.12美元；
2. 有9次赔钱，最大的一次损失是2837美元
3. 除去最高损失那次，其他平均每笔损失是1669.32美元。

我们应用**凯利准则**来确定投资者的最优投资比例.

期权交易应用1：买卖价差策略（续）

上述投资情况也分为3个事件

1. O_1: $\gamma_1 = \frac{659.12}{1669.32} = 0.3948$
2. O_2: 不计最大损失的平均损失额作为单位“赌注”，即1669.32美元，$\lambda_2 = 1$
3. O_3: 对于最大损失那次，$\lambda_3 = \frac{2837}{1669.32} = 1.70$

$$p_1 = \frac{66}{75} = 0.880 \quad p_2 = \frac{8}{75} = 0.107 \quad p_3 = \frac{1}{75} = 0.013$$

投资的期望收益$\mathrm{E} = 0.2183$，代入下式：

$$\mathrm{E} - f[\mathrm{p}_1\gamma_1(\lambda_2 + \lambda_3) + p_2\lambda_2(\gamma_1 - \lambda_3) + p_3\lambda_3(\gamma_1 - \lambda_2)] + \gamma_1\lambda_2\lambda_3 f^2 = 0$$

可获得一个解：f=0.455，即：应将约46%的资金用于这种价差组合。

期权交易应用2：日期价差策略

这是一种持有两个期权产品构成一种组合的投资策略，卖出一个到期日短的某种股票的看涨期权，同时买入一个到期日长且敲定价相同的同一股票的看涨期权。

问题背景：市场上有关交易统计数据:

1. 12笔交易中10笔盈利，平均每笔盈利555.20美元（事件O_1）
2. 最大亏损124美元（事件O_3）
3. 不计最大损失那笔交易，其他平均每笔亏损90美元（事件O_2）

与上例相同，不计最大损失的平均损失额作为单位“赌注”. 可计算：

$$\gamma_1=6.17,\ \lambda_2=1,\ \lambda_3=1.38.$$

资产的期望收益E = 4.943，代入得：f=0.625

期权交易应用2：日期价差策略（续）

决定投资两份组合的最优资金比例。

项目A：投资买卖价差组合　项目B：投资日期价差策略组合。

- f_A, f_B分别为项目A和B的资金比例; p为赢利概率,q表示平均亏损概率
- r表示最大亏损概率，我们还假定每个项目的赢亏是相互独立的

情　形	概　率	随机变量 Q
A 盈利, B 盈利	$p_A p_B$	$1+f_A\gamma_A+f_B\lambda_B$
A 盈利, B 亏损	$p_A q_B$	$1+f_A\gamma_A-f_B$
A 盈利, B 大亏	$p_A r_B$	$1+f_A\gamma_A-f_B\lambda_B$
A 亏损, B 盈利	$q_A p_B$	$1-f_A+f_B\gamma_B$
A 亏损, B 亏损	$q_A q_B$	$1-f_A-f_B$
A 亏损, B 大亏	$q_A r_B$	$1-f_A-f_B\lambda_B$
A 大亏, B 盈利	$r_A p_B$	$1-f_A\lambda_A+f_B\gamma_B$
A 大亏, B 亏损	$r_A q_B$	$1-f_A\lambda_A-f_B$
A 大亏, B 大亏	$r_A r_B$	$1-f_A\lambda_A-f_B\lambda_B$

两份项目组合有9种情形

期权交易应用2：日期价差策略（续）

写出总资产期望增长率，求最优值，即求偏导，并令其为0。

✓最优投资的资金比例f_A的方程为

$$\frac{p_Ap_B\gamma_A}{1+\gamma_Af_A+\gamma_Bf_B}+\frac{p_Aq_B\gamma_A}{1+\gamma_Af_A-f_B}+\frac{p_Ar_B\gamma_A}{1+\gamma_Af_A+\lambda_Bf_B}-\frac{q_Ap_B}{1-f_A+\gamma_Bf_B}+\frac{q_Aq_B}{1-f_A-f_B}$$

$$+\frac{q_Ar_B}{1-f_A-\lambda_Bf_B}-\frac{r_Ap_B\lambda_A}{1-\lambda_Af_A+\gamma_Bf_B}+\frac{\gamma_Aq_B\lambda_A}{1-\lambda_Af_A-f_B}-\frac{r_Ar_B\lambda_A}{1-\gamma_Af_A-\lambda_Bf_B}=0$$

✓最优投资的资金比例f_B的方程为

$$\frac{p_Ap_B\gamma_B}{1+\gamma_Af_A+\gamma_Bf_B}-\frac{p_Aq_B}{1+\gamma_Af_A-f_B}-\frac{p_Ar_B\lambda_B}{1+\gamma_Af_A+\lambda_Bf_B}+\frac{q_Ap_B\gamma_B}{1-f_A+\gamma_Bf_B}-\frac{q_Aq_B}{1-f_A-f_B}$$

$$-\frac{q_Ar_B\lambda_B}{1-f_A-\lambda_Bf_B}+\frac{r_Ap_B\gamma_B}{1-\lambda_Af_A+\gamma_Bf_B}-\frac{\gamma_Aq_B}{1-\lambda_Af_A-f_B}-\frac{r_Ar_B\lambda_B}{1-\lambda_Af_A-\lambda_Bf_B}=0$$

➤传统方法难以求解上述非线性方程组，可用遗传算法有效求解 f_A=0.073， f_B=0.619

案例3：SIR模型及其传染病预测模拟

1. Threshold behavior in a stochastic SIR epidemic model with Logistic birth.pdf

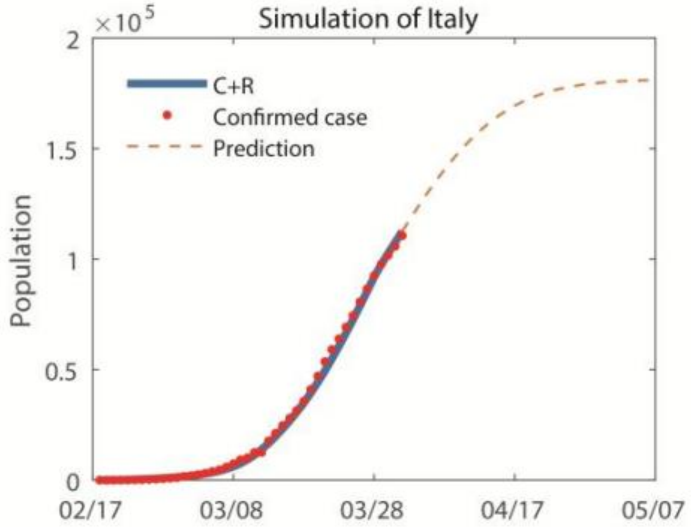
stable. Zhang et al. [3] proposed an SIR epidemic model with Logistic birth which takes on the following form

$$\begin{cases} \frac{dS}{dt} = \left(b - r\frac{N}{K}\right)N - \beta SI - \mu S, \\ \frac{dI}{dt} = \beta SI - \delta I, \\ \frac{dR}{dt} = \gamma I - \mu R, \end{cases} \tag{1.1}$$

where $N(t) = S(t)+I(t)+R(t)$ denotes the total population size at time t. $S(t)$, $I(t)$, $R(t)$ are the numbers of the susceptible

2. Suspected_Close_Contacts_as_the_Pilot_Indicator_of.99982.pdf

$$\begin{cases} \dot{S} = u_1 L(t) - \beta SI, \\ \dot{I} = (1-a)\beta SI + u_2 L(t) - \alpha I, \\ \dot{D} = a\beta SI - bD + (1-g)\alpha I + u_3 L(t), \\ \dot{C} = g\alpha I + bD - \gamma(t)C, \\ \dot{R} = \gamma(t)C, \end{cases} \tag{1}$$



3. 其它参考材料，详见: Covid19-TwoMonth/*.pdf

3. 线性代数方程组

- 求解线性代数方程组

$$Ax = c$$

A：已知m阶非奇异矩阵，$A = [a_{ij}]$;

x：未知m维列向量；$x = [x_i]^T$;

c：已知m维列向量，$c = [c_i]^T$。

迭代求解

通过一定变换，得到线性方程组的雅克比迭代形式：

$$x^{k+1} = Bx^k + d$$

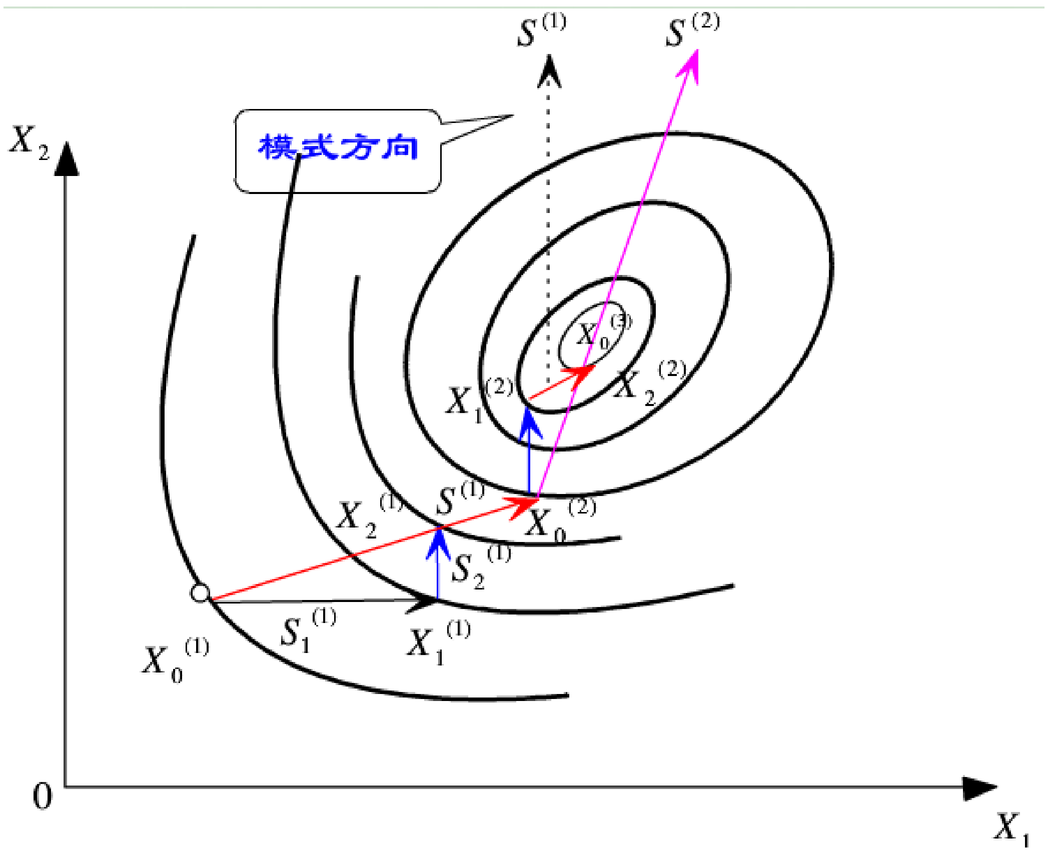
其中，$B = \{b_{ij}\}$, 并满足收敛条件：

$$\max_{1 \le i \le m} \sum_{j=1}^{m} |b_{ij}| < 1$$

迭代公式可以表示成诺依曼级数(x初值取0)：

$$x = \sum_{k=0}^{\infty} B^k d = d + Bd + \cdots + B^k d + \cdots$$

线性代数方程组第i个未知量x_i的诺依曼级数解为：



$$x_i = d_i + \sum_{i_1=1}^{m} b_{ii_1} d_{i_1} + \sum_{i_1=1}^{m} \sum_{i_2=1}^{m} b_{ii_1} b_{i_1 i_2} d_{i_2} + \cdots + \cdots + \sum_{i_1=1}^{m} \sum_{i_2=1}^{m} \cdots \sum_{i_k=1}^{m} b_{ii_1} b_{i_1 i_2} \ldots b_{i_{k-1} i_k} d_{i_k} + \cdots$$

迭代法的随机游动观点

假设有一个具有m个状态的马尔科夫链：

$$i \to i_1 \to i_2 \to \cdots \to i_m$$

其中m个状态之间的转移概率矩阵P满足

$$\sum_{j=1}^{m} p_{ij} = 1$$

这里取：
$$p_{ij} = \frac{|b_{ij}|}{\sum_{k=1}^{n} |b_{ik}|}$$

通过概率矩阵P和迭代矩阵B，我们可以得到矩阵W:

$$w_{ij} = \begin{cases} 0, & b_{ij} = 0, \\ \dfrac{b_{ij}}{p_{ij}}, & b_{ij} \neq 0. \end{cases}$$

基于随机游动的计算模型

定义随机变量e_k为：

$$e_k = w_{ii_1} w_{i_1 i_2} \dots w_{i_{k-1} i_k} \quad 并且 \quad e_0 = 1$$

$$w_{ij} = \begin{cases} 0, & b_{ij} = 0, \\ \dfrac{b_{ij}}{p_{ij}}, & b_{ij} \neq 0. \end{cases}$$

则随机变量θ定义为：

$$\theta = \sum_{k=0}^{\infty} e_k d_{i_k}$$

可以验证：θ的期望为x_i：

$$E(\theta) = d_i + \sum_{i_1=1}^{m} w_{ii_1} p_{ii_1} d_{i_1} + \sum_{i_1=1}^{m} \sum_{i_2=1}^{m} w_{ii_1} p_{ii_1} w_{i_1 i_2} p_{i_1 i_2} d_{i_2} + \cdots + \cdots$$

$$= \sum_{k=0}^{\infty} \sum_{i_1=1}^{m} \sum_{i_2=1}^{m} \cdots \sum_{i_k=1}^{m} w_{ii_1} p_{ii_2} \dots w_{i_{k-1} i_k} p_{i_{k-1} i_k} d_{i_k} = x_i$$

$$x_i = d_i + \sum_{i_1=1}^{m} b_{ii_1} d_{i_1} + \sum_{i_1=1}^{m} \sum_{i_2=1}^{m} b_{ii_1} b_{i_1 i_2} d_{i_2} + \cdots + \cdots + \sum_{i_1=1}^{m} \sum_{i_2=1}^{m} \dots \sum_{i_k=1}^{m} b_{ii_1} b_{i_1 i_2} \dots b_{i_{k-1} i_k} d_{i_k} + \cdots$$

随机算法流程：

1.设置随机游动初始状态为i。

2. 用状态i_k以及概率矩阵P计算下一步状态i_{k+1}

3. **计算统计量e_{k+1}**，以及$\theta_l = \theta_l + e_{k+1}$

4.　k=k+1：当k>r时随机游动停止；

否则，返回第2步。

5.　l=l+1：当l大于给定值n时循环结束；否则，返回第1步。

6. 最后，将统计量θ_l的平均值作为线性方程组解的估计值：

$$x_i \approx \frac{1}{n}\sum_{l=1}^{n}\theta_l$$

```
%% begin iteration
x = zeros(m,1);
for i = 1:m
    theta = 0;
    for h = 1:numn
      q = 1; a = i; ik = i;
      theta = theta + d(i);
      for k = 1:numr
        ik = discretize(rand,[-Inf f(ik,:) Inf]);
        q =q*W(a,ik);
        theta = theta + q*d(ik);
        a=ik;
      end
    end
    x(i) = theta/numn;
end
```

$$e_k = w_{ii_1}w_{i_1i_2}\dots w_{i_{k-1}i_k}$$

$$\theta = \sum_{k=0}^{\infty} e_k d_{i_k}$$

l　1:n

k　1:r

0~1

游走过程

文档 示例 函数 视频 MATLAB Answers

示例

将数据划分为 bin

使用 discretize 将数值分组到离散 bin。edges 定义五个 bin 边界，因此有四个 bin。

```
data = [1 1 2 3 6 5 8 10 4 4]
```

data = *1×10*

```
     1     1     2     3     6     5     8    10     4     4
```

```
edges = 2:2:10
```

edges = *1×5*

```
     2     4     6     8    10
```

```
Y = discretize(data,edges)
```

Y = *1×10*

```
   NaN   NaN     1     1     3     2     4     4     2     2
```

Y 表示数据的每个元素属于哪个 bin。由于值 1 超出了 bin 范围，因此 Y 会在这些元素位置包含 NaN 值。

```
%% begin iteration
x = zeros(m,1);
for i = 1:m
    theta = 0;
    for n = 1:numn
      q = 1; a = i; ik = i;
      theta = theta + d(i);
      for r = 1:numr
        ik = discretize(rand,[-Inf f(ik,:) Inf]);
        q =q*W(a,ik);
        theta = theta + q*d(ik);
        a=ik;
      end
     end
    x(i) = theta/numn;
end
```

算例1：

求解线性方程组：

$$Ax = b$$

$$A = \begin{bmatrix} 8 & -3 & 2 \\ 4 & 11 & -1 \\ 6 & 3 & 12 \end{bmatrix}, b = \begin{bmatrix} 20 \\ 33 \\ 36 \end{bmatrix}$$

方程的精确解为：

$$x = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

```
%mcmc 求解 线性方程组
A=[8 -3 2;4 11 -1;6 3 12];
b=[20;33;36];
numn = 1000;
numr=10;
m = size(A,1);
D = diag(diag(A));
L = -tril(A,-1);
U = -triu(A,1);
B = D\(L+U);
d = D\b; %%%%确定雅克比迭代形式
P=abs(A)./repmat(sum(abs(A),2),1,m); %%%%%%%%%确定概率矩阵
W=B./P;      %%%确定权重矩阵
f=cumsum(P,2);%%%%%%概率分布矩阵
x=zeros(m,1);
for i=1:m
    theta=0;
    for n = 1:numn
      q=1;a=i;ik=i;theta=theta+d(i);
       for r=1:numr
       ik=discretize(rand,[-Inf f(ik,:) Inf]);
       q=q*W(a,ik);
       theta=theta+q*d(ik);
       a=ik;
       end
     end
    x(i)=theta/numn;
end
```

$$x^{k+1} = Bx^k + d$$

r	n	x_1	x_2	x_3
10	100	2.9809026342	1.8693026859	0.8336682431
10	1000	2.9217169825	2.0147582644	0.9841580916
10	10000	2.9727338313	2.0785937093	1.0746126303
5	1000	3.0129762790	2.0468352272	1.1134566256
20	1000	3.1097911458	1.7783316086	0.5952683279

4. 椭圆型偏微分方程

$$a(P)\frac{\partial^2 u}{\partial x^2}+b(P)\frac{\partial^2 u}{\partial y^2}+c(P)\frac{\partial u}{\partial x}+d(P)\frac{\partial u}{\partial y}+e(P)u=\varphi(P), \qquad P\in D$$

$$u(Q)=f(Q), \qquad Q\in\Gamma$$

其中$a(P),b(P)\geq 0,e(P)\leq 0$

差分格式－离散

$$a(P)\frac{\partial^2 u}{\partial x^2}+b(P)\frac{\partial^2 u}{\partial y^2}+c(P)\frac{\partial u}{\partial x}+d(P)\frac{\partial u}{\partial y}+e(P)u=\varphi(P)$$

用五点差分格式，则原方程在规则网格上的差分格式为：

$$u_{i,j}=A_{ij}u_{i+1,j}+B_{ij}u_{i-1,j}+C_{ij}u_{i,j+1}+D_{ij}u_{i,j-1}+F_{ij}\quad P\in G^*$$

$$u(Q)=f(Q),\quad Q\in\Gamma^*$$

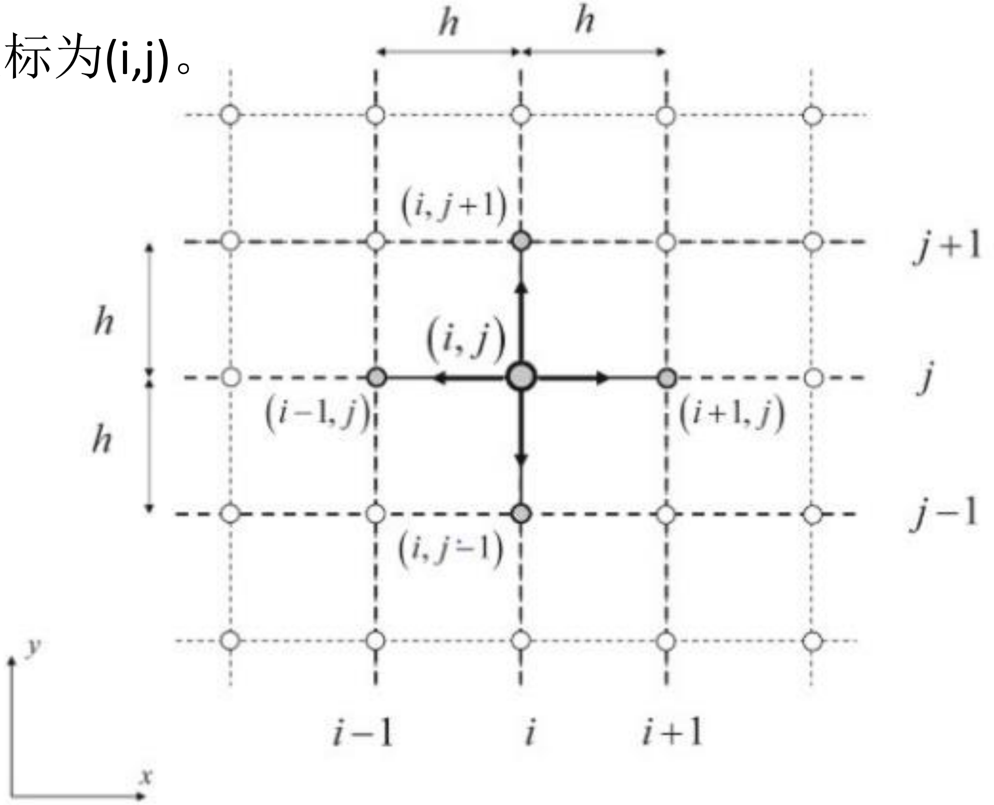
G^*,Γ^*分别是以h为步长的正方形网格的内点与边界点，P点坐标为(i,j)。

其中：

$$A_{ij}=K_{ij}^{-1}(\frac{a_{ij}}{h^2}+\frac{c_{ij}}{2h}),\ B_{ij}=K_{ij}^{-1}(\frac{a_{ij}}{h^2}-\frac{c_{ij}}{2h}),$$

$$C_{ij}=K_{ij}^{-1}(\frac{b_{ij}}{h^2}+\frac{d_{ij}}{2h}),\ D_{ij}=K_{ij}^{-1}(\frac{b_{ij}}{h^2}-\frac{d_{ij}}{2h}),$$

$$K_{ij}=\left(\frac{2a_{ij}+2b_{ij}}{h^2}-e_{ij}\right),F_{ij}=-K_{ij}^{-1}\varphi_{ij}$$



随机游动计算方式

$$u_{i,j} = A_{ij}u_{i+1,j} + B_{ij}u_{i-1,j} + C_{ij}u_{i,j+1} + D_{ij}u_{i,j-1} + \mathbf{F_{ij}}$$

- 当h足够小时，A_{ij}、B_{ij}、C_{ij}、D_{ij}为大于0小于1的正数，并且对任意$P \in G^*$有：

$$A(P) + B(P) + C(P) + D(P) \leq 1$$

- 因此，可以通过随机游动求解P点u_{ij}的值，向P点的四个邻点游动的概率为

$$A(P)、B(P)、C(P)、D(P)$$

相应地，停留在P点的概率为：

$$1 - (A(P) + B(P) + C(P) + D(P))$$

模拟方法：若随机游动路径为：$P_0 \to P_1 \to P_2 \ldots \to P_{k-1} \to Q_k$(随机游动至边界点则停止)。取随机变量：

$$\boldsymbol{\theta} = \boldsymbol{f}(\boldsymbol{P_0}) + \boldsymbol{f}(\boldsymbol{P_1}) + \cdots + \boldsymbol{f}(\boldsymbol{P_{k-1}}) + \boldsymbol{f}(\boldsymbol{Q_k})$$

在P点做n次随机游动，可以得到方程在P点的估计值：

$$u(P) = \frac{1}{n}\sum_{i=1}^{n}\theta_i$$

算例2：泊松方程

$$u_{xx} + u_{yy} = -2\pi^2 \sin(\pi x)\sin(\pi y) \qquad (\mathrm{x},\mathrm{y}) \in D$$

$$u = \sin(\pi x)\sin(\pi y) \qquad (x, y) \in \partial D$$

这里$D = [0,1] \times [0,1]$.

上述泊松方程的精确解为：

$$u = \sin(\pi x)\sin(\pi y)$$

```python
import random
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
```

```python
monte-carlo for poisson equation
"""
import ...

h = 1
points = 30  # Number of Points
d = h / points  # step size
N = 100  # Number of Random Walks

def u(*A):
    x = list(A)
    exact = np.sin(np.pi * x[0]) * np.sin(np.pi * x[1])
    return exact

def f(x):
    return -2*np.pi**2*np.sin(np.pi*x[0])*np.sin(np.pi*x[1])


def g(x):
    if x[0] <= 0 or x[0] >= h or x[1] <= 0 or x[1] >= h:
        return np.sin(np.pi*x[0])*np.sin(np.pi*x[1])
```

参考demo_PDE-simulation.html

代码：

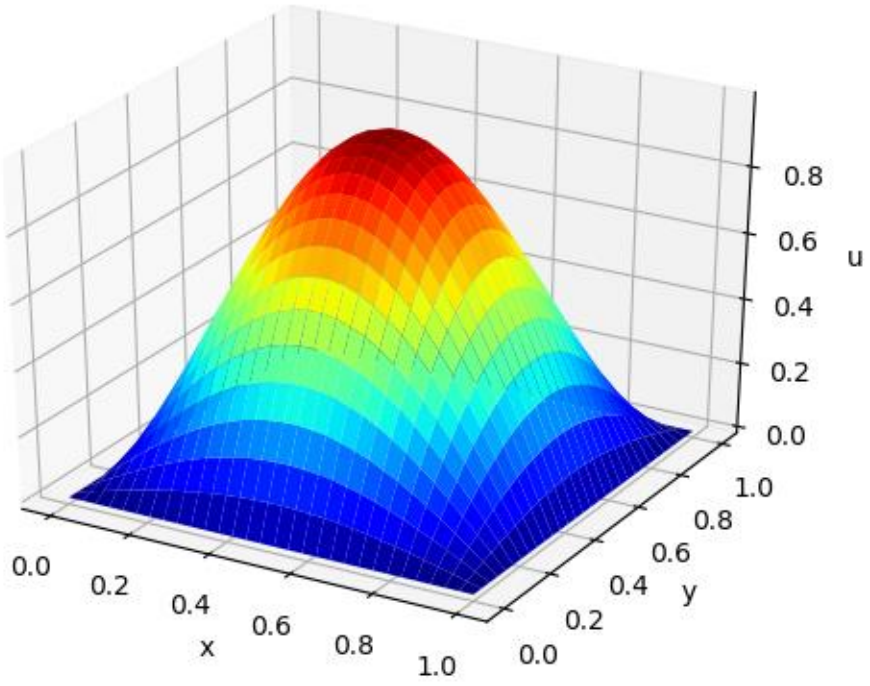
```python
@np.vectorize
def poisson_approximation_fixed_step(*A):
    result = 0
    F = 0
    for i in range(N):
        x = list(A)
        while True:
            if x[0] <= 0 or x[0] >= h or x[1] <= 0 or x[1] >= h:
                break
            random_number = random.randint(0, 3)
            if random_number == 0:
                x[0] += d
            elif random_number == 1:
                x[0] -= d
            elif random_number == 2:
                x[1] += d
            elif random_number == 3:
                x[1] -= d
            F += f(x) * d ** 2/4
        result += g(x)
    result = (result - F)/N
    return result
```

随机游动

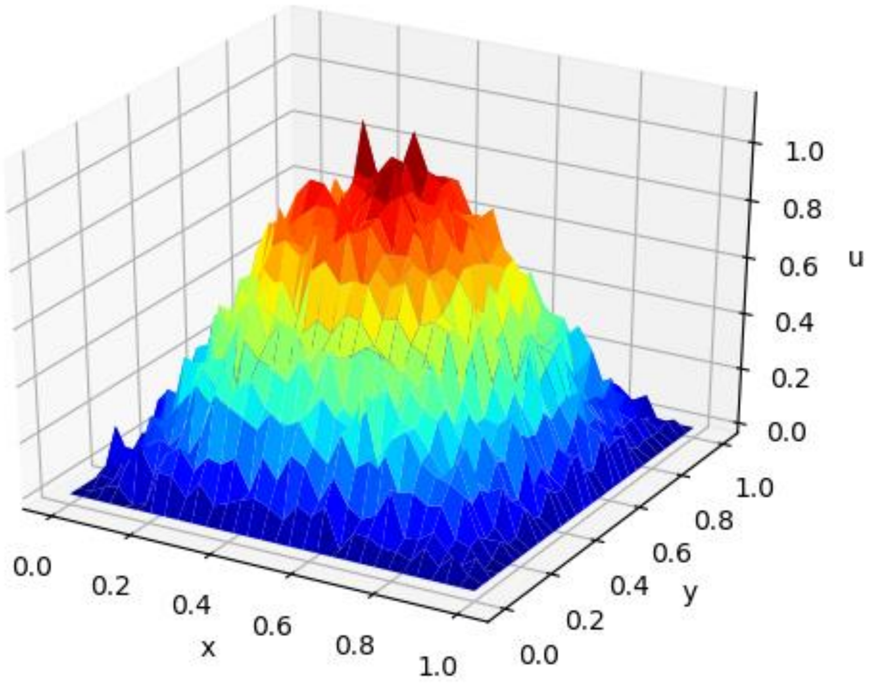
```python
def plot(x, y, z):
    # Function for plotting the value
    fig = plt.figure()
    ax = fig.add_subplot(111, projection="3d")

    ax.plot_surface(x, y, np.array(z), cmap=cm.jet, linewidth=0.1)
    plt.xlabel("x")
    plt.ylabel("y")
    ax.set_zlabel("u")
    plt.show()


if __name__ == "__main__":

    print(
        f"Calculating Monte Carlo with {points}x{points} lattice points and {N} random walks"
    )
    lattice_x, lattice_y = np.mgrid[
        0 : h : points * 1j, 0 : h : points * 1j
    ]
    z = poisson_approximation_fixed_step(lattice_x.ravel(), lattice_y.ravel()).reshape(
         lattice_x.shape
    )
    u = u(lattice_x.ravel(), lattice_y.ravel()).reshape(
        lattice_x.shape
    )

    #montecarlo solution
    plot(lattice_x, lattice_y, z)

    # #exact solution
    # plot(lattice_x, lattice_y, u)
```

求解及画图

结果：



精确解



100次随机游动时的蒙特卡洛解

5. 抛物型偏微分方程

$$\frac{\partial u}{\partial t}-a(P)\frac{\partial^2 u}{\partial x^2}-b(P)\frac{\partial^2 u}{\partial y^2}+c(P)\frac{\partial u}{\partial x}+d(P)\frac{\partial u}{\partial y}+r(P)u=s(P), \qquad P\in D$$

其中，$a(P)>0, b(P)>0, r(P)>0$. 初始条件和边界条件为：$\Phi(Q)=f(Q), Q\in\Gamma$

差分格式－离散

选用一阶迎风差分格式，将抛物方程离散化：

$$u_{i,j}^{n+1} = A_{ij}u_{ij}^{n} + B_{ij}u_{ij+1}^{n} + C_{ij}u_{ij-1}^{n} + D_{ij}u_{i+1j}^{n} + E_{ij}u_{i-1j}^{n} + F_{ij} \quad \mathrm{P} \in D^{*}$$

$$\Phi(Q) = f(Q), Q \in \Gamma^{*}$$

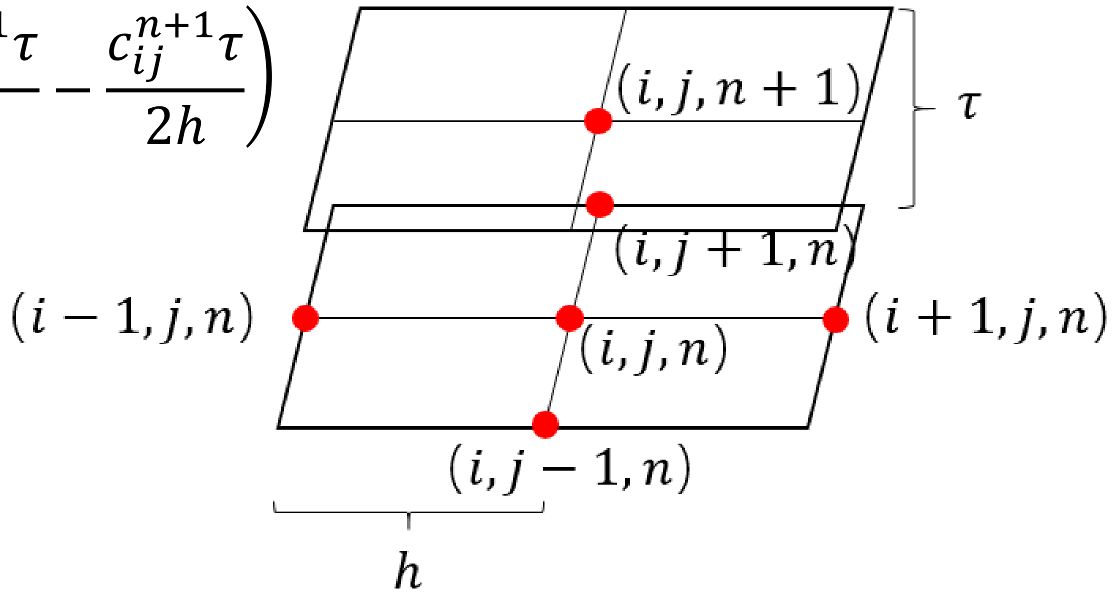
D^{*}, Γ^{*}分别是以h为空间步长，τ为时间步长的网格上的内点与边界点，P点坐标为(i,j,n+1)

其中：

$$A_{ij} = \left(1 - \frac{2a_{ij}^{n+1}\tau}{h^2} - \frac{2b_{ij}^{n+1}\tau}{h^2} - \tau r_{ij}^{n+1}\right), \mathrm{B_{ij}} = \left(\frac{a_{ij}^{n+1}\tau}{h^2} - \frac{c_{ij}^{n+1}\tau}{2h}\right)$$

$$\mathrm{C_{ij}} = \left(\frac{a_{ij}^{n+1}\tau}{h^2} + \frac{c_{ij}^{n+1}\tau}{2h}\right), \mathrm{D_{ij}} = \left(\frac{b_{ij}^{n+1}\tau}{h^2} - \frac{d_{ij}^{n+1}\tau}{2h}\right)$$

$$\mathrm{E_{ij}} = \left(\frac{b_{ij}^{n+1}\tau}{h^2} - \frac{d_{ij}^{n+1}\tau}{2h}\right), \mathrm{F_{ij}} = \tau s_{ij}^{n+1}$$



随机游动计算方式

选取适当的$\frac{\tau}{h^2}$，则A_{ij}、B_{ij}、C_{ij}、D_{ij}、E_{ij}为大于0小于1的正数，并且对任意$P \in D^*$有：

$$A(P) + B(P) + C(P) + D(P) + E(P) \leq 1$$

选取$A(P)$、$B(P)$、$C(P)$、$D(P)$、$E(P)$为P向5个目标点游动的概率，则

$$1 - (A(P) + B(P) + C(P) + D(P) + E(P))$$

为留在新P点的概率。

模拟方法：若随机游动路径为：$P_0 \to P_1 \to P_2 \ldots \to P_{k-1} \to Q_k$(随机游动至边界点则停止)

则取**随机变量**：

$$\boldsymbol{\theta} = F(P_0) + F(P_1) + \cdots + F(P_{k-1}) + \Phi(Q_k)$$

在P点做n次随机游动，则可以得到方程在P点的估计值：

$$u(P) = \frac{1}{n}\sum_{i=1}^{n} \theta_i$$

算例3：热传导方程

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

$$u(x, y, 0) = \sin \pi x + \sin \pi y$$

$$u(x, 0, t) = u(x, 1, t) = e^{-\pi^2 t} \sin \pi x$$

$$u(0, y, t) = u(1, y, t) = e^{-\pi^2 t} \sin \pi y$$

$u: [0,1] \times [0,1] \times [0, +\infty) \to R$

这个方程的精确解为：$u = e^{-\pi^2 t}(\sin \pi x + \sin \pi y)$

```python
"""
monte-carlo for heat equation
"""
import random
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np

t= 0.1
h = 1
points = 40  # Number of Points
d = h / points  # step size
N = 50  # Number of Random Walks
tau=d**2/5
tpoints=800

def u(*A):
    x = list(A)
    exact = np.exp(-np.pi**2*t)*(np.sin(np.pi*x[0])+np.sin(np.pi*x[1]))
    return exact

def f(x):
    return 0


def g(x):
    if x[2] <= 0:
        return np.sin(np.pi*x[0])+np.sin(np.pi*x[1])
    if x[0] <= 0 or x[0] >= h:
        return np.exp(-np.pi**2*x[2])*np.sin(np.pi*x[1])
    if x[1] <= 0 or x[1] >= h:
        return np.exp(-np.pi**2*x[2])*np.sin(np.pi*x[0])
```

参考demo_PDE-simulation.html

代码:

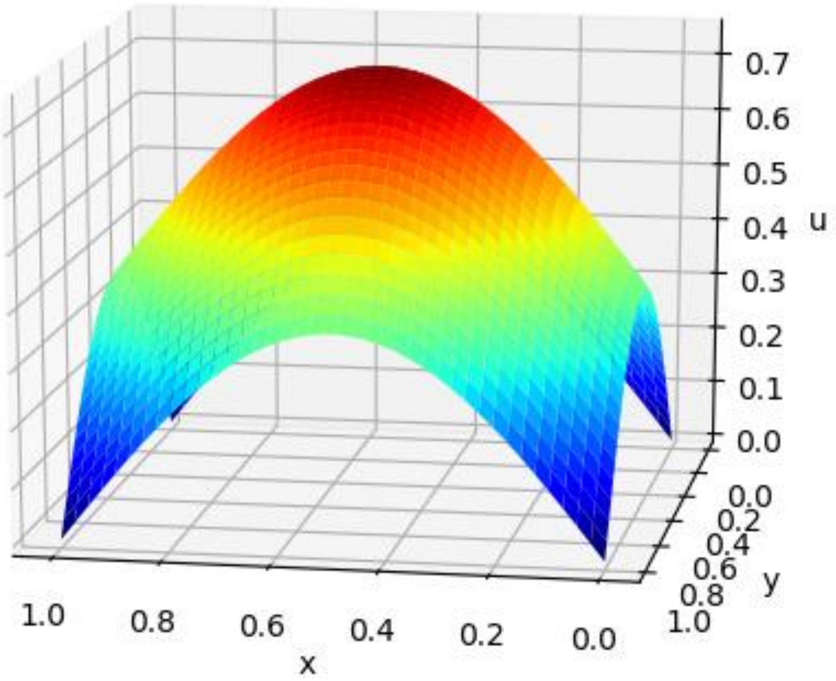
```python
@np.vectorize
def heat_approximation_fixed_step(*A):
    result = 0
    F = 0
    for i in range(N):
        x = list(A)
        while True:
            if x[0] <= 0 or x[0] >= h or x[1] <= 0 or x[1] >= h or x[2] <= 0:
                break
            random_number = random.randint(0, 4)
            if random_number == 0:
                x[0] += d
                x[2] += -tau
            elif random_number == 1:
                x[0] -= d
                x[2] += -tau
            elif random_number == 2:
                x[1] += d
                x[2] += -tau
            elif random_number == 3:
                x[1] -= d
                x[2] += -tau
            elif random_number == 4:
                x[2] += -tau
            F += f(x) * tau
        result += g(x)
    result = (result + F)/N
    return result
```

随机游动

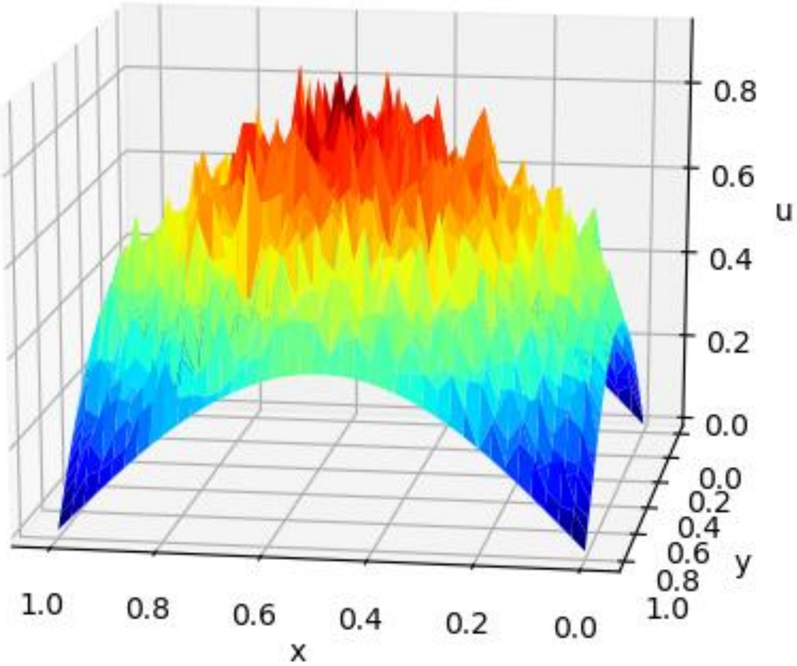
```python
def plot(x, y, z):
    # Function for plotting the value
    fig = plt.figure()
    ax = fig.add_subplot(111, projection="3d")

    ax.plot_surface(x, y, np.array(z), cmap=cm.jet, linewidth=0.1)
    plt.xlabel("x")
    plt.ylabel("y")
    ax.set_zlabel("u")
    plt.show()


if __name__ == "__main__":

    print(
        f"Calculating Monte Carlo with {points}x{points}x{tpoints} lattice points and {N} random walks"
    )
    lattice_x, lattice_y ,lattice_t= np.mgrid[
        0 : h : points * 1j, 0 : h : points * 1j , 0: t : tpoints *1j
    ]
    z = heat_approximation_fixed_step(lattice_x.ravel(), lattice_y.ravel(), lattice_t.ravel()).reshape(
         lattice_x.shape
    )
    z = z[:, :, tpoints - 1]
    x = lattice_x[:, :, tpoints - 1]
    y = lattice_y[:, :, tpoints - 1]
    # u = u(lattice_x.ravel(), lattice_y.ravel(),lattice_t.ravel()).reshape(
    #     lattice_x.shape
    # )
    # u = u[:, :, tpoints - 1]
    # x = lattice_x[:, :, tpoints - 1]
    # y = lattice_y[:, :, tpoints - 1]
    #montecarlo solution
    plot(x, y, z)

    # #exact solution
    # plot(x, y, u)
```

求解及画图

模拟结果



t=0.1时的精确解



t=0.1，50次随机游动时的蒙特卡洛解

Homework 08

1. 第九章课后作业：8，9，10

2. 分别用雅克比迭代法和monte-carlo方法求解线性方程组

- 尝试求解3、10阶， 100阶的能求解吗？
- 比较两类求解方法的速度与精度
- 记录你的结果并归纳你的结论