Py 语言初步——随机游走

for i in range(0,nstep):

theta = random.random()*2.0*np.pi

dx = np.cos(theta); dy = np.sin(theta)

x=x+dx; y=y+dy; xlist.append(x); ylist.append(y)

二． 数值计算——差值，数值微分，数值积分

2.1. 差值法

拉格朗日差值 $y(x)=\sum_{j=0}^{n}A_j(x)y_j$，$A_j(x)=\prod_{i=0, i\neq j}^{n}\frac{x-x_i}{x_j-x_i}$

import scipy.interpolate as sp

_=sp.lagrange(x[ ],y[ ])

2.2. 三次样条差值

tck=sp.splrep(x, y, k=3, s=1.2)

y=sp.splev(x,tck,der=0)

2.3. 数值微分 $y'(x_i)=\frac{y_{i+1}-y_{i-1}}{2h}$，$y''(x_i)=\frac{y_{i-1}-2y_i+y_{i+1}}{h^2}$

中心差分 Ex = -1.0*(U(x0+dx,y0)-U(x0-dx,y0))/(2.0*dx)

2.4. 数值积分

矩形公式：$I=\sum_{i=1}^{n-1}\Delta S_i=\sum_{i=1}^{n-1}f(x_i)\Delta x$

梯形公式：$I=\sum_{i=1}^{n-1}\Delta S_i=\sum_{i=1}^{n-1}[f(x_i)+f(x_{i+1})]\frac{\Delta x}{2}$

高斯积分：$\int_a^b f(x)dx \approx \sum_{k=1}^{N}w_k f(x_k)$

from scipy.special import roots_legendre

x,w=roots_legendre(N); xk= x*(b-a)/2+(b+a)/2; wk= w*(b-a)/2

for i in range(0,N-1,2):

U = U + wk[i]*f(xk[i])

抛物线(辛普森)积分：$V=\sum_{i=2,4}^{n}\frac{1}{3}[y_{i-1}+4y_i+y_{i+1}]\Delta x$

三． 常微分方程 （给定初始条件向后演化）

欧拉法：向前差分$y_{n+1}=y_n+\Delta t y'(y_n,t_n)$

改进欧拉法：

for i in range(nstep): t=t+dt; k1=f(y,t); k2=f(y+k1*dt/2,t)

y=y+k2*dt #存储到一个 y[i]数组中

二阶 RK: $y_{n+1}=y_n+\Delta t f(y_{n+1/2},t_{n+1/2})$, $y_{n+1/2}=y_n+\Delta t/2*f(y_n,t_n)$

四阶 RK : $y_{n+1}=y_n+\frac{\Delta t}{6}(k_1+2k_2+2k_3+k_4)$

$$\begin{cases} k_1=f(y_n,t_n)\\ k_2=f(y_n+k_1\Delta t/2.,t_{n+1/2})\\ k_3=f(y_n+k_2\Delta t/2.,t_{n+1/2})\\ k_4=f(y_n+k_3\Delta t,t_{n+1})\end{cases}$$

变量数为 1 示例，n 变量即括号中有 n 个变量 dx/dt=f(x)

def fx(t,x) : return dx/dt (f(x))

def RK4(t,x): k1=fx(t,x); k2=fx(t+dt/2,x+dt/2*k1); k3=... k4=... ;

x=x+dt/6.*(k1+2*k2+2*k3+k4); return x

for i in range(n): t=t+dt; x=RK4(t,x); T.apend(t); X.append(x) #or T[i]=t,t=t+dt

四．偏微分方程

一维热传导：$\frac{\partial u}{\partial t}=\lambda\frac{\partial^2 u}{\partial x^2}+q(t,x)$ 初始，边界条件后，

for k in range(1,NT): for i in range(1,NX): U[i,k+1]=A*U[i+1,k]+(1-2*A)*U[i,k]+A*U[i-1,k] + tau*q(i*h)

A=λ*τ/h^2，

$$u_{i,k+1}=\frac{\lambda\tau}{h^2}u_{i+1,k}+\left(1-2\frac{\lambda\tau}{h^2}\right)u_{i,k}+\frac{\lambda\tau}{h^2}u_{i-1,k}+\tau q|_{i,k}$$

二维热传导-有限差分：$\frac{\partial u}{\partial t}=\lambda\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+q(t,x,y)$,

$\frac{u_{i,j,k+1}-u_{i,j,k}}{\tau}=\lambda\frac{u_{i-1,j,k}-2u_{i,j,k}+u_{i+1,j,k}}{h^2}+\lambda\frac{u_{i,j-1,k}-2u_{i,j,k}+u_{i,j+1,k}}{h^2}+q|_{i,j,k}$，$u_{i,j,k}=\cdots$

反应扩散：$\frac{\partial U}{\partial t}=D_u\nabla^2U-UV^2+F(1-U)$，$\frac{\partial V}{\partial t}=D_v\nabla^2V+UV^2-(F+r)V$ ； 在↑加($-UV^2+F(1-U)$)⋯

弦振动：$\frac{\partial^2 y}{\partial t^2}=v^2\frac{\partial^2 y}{\partial x^2}+P(x,t)$ ➔$\frac{\partial^2 y}{\partial x^2}=\frac{y_{i+1,k}-2y_{i,k}+y_{i-1,k}}{h^2}$, $\frac{\partial^2 y}{\partial t^2}=\cdots$

$$y_{i,k+1}=2\left(1-\frac{\tau^2v^2}{h^2}\right)y_{i,k}+\frac{\tau^2v^2}{h^2}\left(y_{i+1,k}+y_{i-1,k}\right)-y_{i,k-1}+\tau^2P_{i,k}$$

二维泊松方程：$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=-\frac{\rho}{\epsilon_0}$ →$\phi_{i,j}=\frac{1}{4}(\phi_{around})+\frac{h^2}{4\epsilon_0}\rho_{i,j}$ 对 ij 循环

收敛条件：$\left|\phi_{i,j}^k-\phi_{i,j}^{k+1}\right|<\epsilon$

二维膜振动：$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}=\Delta u$ → $u_{i,j}^{k+1}=2u_{i,j}^k-u_{i,j}^{k-1}+\left(\frac{c\tau}{h}\right)^2(u_{around}^k-4u_{i,j}^k)$

五．线性计算

线性方程组 from numpy import array, linalg; A=array([[ ],[ ],[ ]])， b=([ ]); x=linalg.solve(A,b)

矩阵本正在 a=array([...]), eigenValues,eigenVectors = linalg.eigh(a) ;

idx=e...Values.argsort() #开序; Evalue=e...V...[idx]; Evector=e...V...[:,dx]

六．非线性计算

搜索求方程根：f(x),x_0,dx,eps（精度）

while abs(fold)>eps: x=x+dx; fnew=f(x);

if fnew*fold<0: x=x-dx; dx=dx/2; fold=f(x)

二分法求根：for i in range(0, N): x=(a+b)/2;（搜索区间中间）

if (f(a)*f(x) > 0.): a=x; else: b=x; (更新边界)

if (abs(f(x))<eps): break

牛顿法：for i in range(0, N): xold=x; x=xold-f(xold)/fp(xold); $x^{i+1}=x^i-\frac{f(x^i)}{f'(x^i)}$

if (abs(f(x)) < eps): 找到 x break; if (i == N-1): 没找到 break

弦割法：用$f'(x^i)=\frac{f(x^i)-f(x^{i-1})}{x^i-x^{i-1}}$代替求导 设置初始 x1,x2, x3=⋯

for i in range(0, N): if (abs(f(x3)) < eps): break; if (i == N-1):没找到 break;

x1=x2; x2=x3; x3=x2-f(x2)*(x2-x1)/(f(x2)-f(x1))

打靶法求微分方程本征值 （按照常微分方程演化对比终止条件）无限深势阱为例

while abs(dk)>eps:

k=k+dk; psiold=psi; psi=RK4new(x, psi, phi) #是 RK4 找到的端点值

if psiold*psi > 0 : continue

k = k-dk; dk=dk/2

最速下降法：求梯度，按梯度方向下降步长→U(x',y')<U(x,y)继续；U(x',y')>U(x,y)缩小一半步长；步长<精度停止

for i in np.arange(0,1000):

$fx=\frac{\partial U}{\partial x}$，$fy=\frac{\partial U}{\partial x}$ #偏导；$norm=(fx^2+fy^2)^{0.5}$ #归一化

dx = fx/norm; dy = fy/norm #梯度方向

$\delta x=\delta r*dx$；$\delta y=\delta r*dy$ #搜索步长

$x=x+\delta x; y=y+\delta y$ #更新坐标

$fnew=U(x,y)$ #新函数值

if (fnew>fold): $x=x-\delta x$；$y=y-\delta y; \delta r=\frac{\delta r}{2.0}; fold=fnew$

xx.append(x); yy.append(y);

if(deltar<1.0e-6): print (x,y)#最低点; break

共轭梯度优化：$\Delta(x_n,y_n)=G(x_n,y_n)+\lambda\Delta(x_{n-1},y_{n-1}),\lambda=\frac{|G(x_n,y_n)|^2}{|G(x_{n-1},y_{n-1})|^2}$

if i > 0: $\beta=\frac{fx^2+fy^2}{fx_{old}^2+fy_{old}^2}$; else: $\beta=0$;

$fx_0=\frac{fx}{(fx^2+fy^2)^{0.5}}$; $fy_0=\frac{fy}{(fx^2+fy^2)^{0.5}}$ #当前梯度方向

$dx=fx_0+\beta*dx_{old}$; $dy=fy_0+\beta*dy_{old}$ #当前搜索方向

$dx=\frac{dx}{(dx^2+dy^2)^{0.5}}$; $dy=\frac{dy}{(dx^2+dy^2)^{0.5}}$;

$dx_{old}=dx$; $dy_{old}=dy; fx_{old}=fx$; $fy_{old}=fy$;

插入在原代码 fx,fy 后，ΔxΔy 前

七．计算机模拟

产生确定分布函数的随机数：

反函数：$\eta=F^{-1}(\xi)$ ξ为均匀分布

舍选法：for i in range(nmax):

r1 = a+(b-a)*random.random(); r2 = random.random(); if(r2<f(r1)/fmax): row.append(r1)

布朗运动-朗之万方程$m\frac{d^2x}{dx^2}=F(x)-\frac{\xi dx}{dt}+F_R$；雷诺数小可略二阶导

for it in range(nt-1):

FR=np.sqrt(2*Temp/ksai/dt)*random.gauss(0,1); xt[it+1]=xt[it]+1.0/ksai*F(xt(it))*dt+FR*dt

重要性抽样积分：$I = \int_0^\infty A(x)dx = \int_0^\infty \frac{A(x)}{g(x)} g(x)dx = \int_0^\infty A^*(x)g(x)dx = \frac{1}{N}\sum_{i=1}^{N} A^*(\xi_i)$

for i in range(N): kesi = 某分布; integral+=1.0/N*A*(kesi)

蒙特卡洛判据$T_{S\to S'} = \min\left(1, \frac{P(S')}{P(S)}\right)$ $P_a = \begin{cases} 1 & if\ H(S') \le H(S) \\ e^{-\beta\Delta H} & if\ H(S') < H(S) \end{cases}$

涉及温度，β=1/kT，判断从 S 态随机游走到 S'态被接受的概率

Randon-Snew→Enew→if dE<0: S=Snew; else: w=np.exp(-dh/T), ksi=np.random.random()

If ksi<w: S=Snew #(S 表示态)