伪代码：

* **排斥力的修改**

设置初始参数

for t=0:dt:T

计算每个粒子感受到的密度（j是粒子i前方的粒子）

计算临界密度rho\_p2p, rho\_p2w

for 行人粒子j in 粒子i的作用域

if:

else:

找到粒子i作用域内最近的障碍粒子u

if:

else:

计算加速度模长：

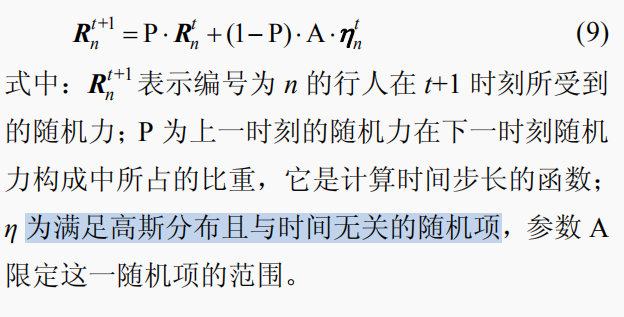
计算*ar\_x, ar\_y*

.

.

.

* **朗之万随机力项**



在时间步循环外定义全局变量 来存储t时刻各粒子的朗之万随机力产生的加速度*x，y*分量

新参数P\_r = 0.93

theta = [0, 2π)随机抽样

e\_x = cos(theta)

e\_y = sin(theta)

a\_x[j] = a\_x[j] \* P\_r + (1-P\_r) \* A \* randn(0,1) \* e\_x % randn是想产生一个服从均值0方差1的高斯分布的随机数，我不确定是不是这个式子了QAQ

% 参数A根据加速度的量级来确定，比排斥力的加速度小一个量级即可

a\_y[j] = a\_y[j] \* P\_r + (1-P\_r) \* A \* randn(0,1) \* e\_y

* **跟随行为**

在全局变量中添加熟悉程度familiarity = zeros(1, N)

对familiarity进行初始化，随机生成在[0, 1]之间的数。可以设置familiarity在[0.8, 1]之间随机生成的粒子较多(如占比80%)，剩下的familiarity在[0, 0.8]之间随机生成

每个粒子的跟随程度P\_f = exp(-2\*familiarity)

|  |
| --- |
| %% 考虑跟随行为计算行人主动力产生的加速度  e\_x=zeros(1,n); %初始化方向向量的x坐标  e\_y=zeros(1,n); %初始化方向向量的y坐标  e\_x\_familiar=zeros(1,n); %初始化主动力方向向量的x坐标  e\_y\_familiar=zeros(1,n); %初始化主动力方向向量的y坐标  e\_x\_follow=zeros(1,n);%初始化跟随行为方向向量的x坐标  e\_y\_follow=zeros(1,n);%初始化跟随行为方向向量的y坐标  %计算行人对逃生路线熟悉所产生的方向  r=sqrt((person\_x(i)-exit\_x)^2+(person\_y(i)-exit\_y)^2); %熟悉逃生路线的行人粒子与出口之间的距离  e\_x\_familiar(i)=(exit\_x-person\_x(i))/r;  e\_y\_familiar(i)=(exit\_y-person\_y(i))/r;  %计算行人跟随行为所产生的方向  对i的作用域内的粒子j：  r\_2=(person\_x(i)-person\_x(j))^2+(person\_y(i)-person\_y(j))^2; %两行人粒子距离的平方  if r\_2<=h1^2  e\_x\_follow(i)=(m\_person\*e\_x(j)/Rho\_person(j))\*(4/(pi\*h1^8))\*(h1^2-r\_2)^3;  e\_y\_follow(i)=(m\_person\*e\_y(j)/Rho\_person(j))\*(4/(pi\*h1^8))\*(h1^2-r\_2)^3;  end  e\_x(i)=P\_f(i)\*e\_x\_follow(i) + (1-P\_f(i))\*e\_x\_familiar(i);  e\_y(i)=P\_f(i)\*e\_y\_follow(i) + (1-P\_f(i))\*e\_y\_familiar(i);  r=sqrt(e\_x(i)^2+e\_y(i)^2);%方向向量的模  e\_x(i)=e\_x(i)/r;%将方向向量转换为单位方向向量  e\_y(i)=e\_y(i)/r;%将方向向量转换为单位方向向量  r=sqrt((person\_x(i)-exit\_x)^2+(person\_y(i)-exit\_y)^2);  if r<=5 %当不熟悉疏散路线的行人接近出口时，将其视为熟悉疏散路线的行人  if person\_x(i)<=exit\_x  e\_x(i)=(exit\_x-person\_x(i))/r;  e\_y(i)=(exit\_y-person\_y(i))/r;  end  end |