姓名	学号	班级	选题	论述	结论	总分
陈忠	2010301020155	天眷班				

标题：封闭区域的碰撞问题

作者：陈忠 2013301020155 天眷班

摘要：本文主要讨论可以视为质点的刚性球在不同的封闭边界下发生弹性碰撞的运动轨迹，此时质点运动规律为牛顿力学。并通过分析质点的运动轨迹给出混沌出现的条件，利用 Vpython 动态展示质点的运动。最后以环形边界为例，比较在不同的物理情境下质点的运动情形，包括粒子带一个单位正电荷的情形。

正文：

碰撞原理：质点与边界碰撞是速度改变：

$$\vec{v}_{i,\perp} = (\vec{v}_i \cdot \hat{n})n, \vec{v}_{i,\parallel} = \vec{v}_i - \vec{v}_{i,\perp} \quad (1\text{ 式})$$

$$\vec{v}_{f,\perp} = -\vec{v}_{i,\perp}, \vec{v}_{f,\parallel} = \vec{v}_{i,\parallel} \quad (2\text{ 式})$$

设边界在碰撞点的法向量为$\vec{n} = a\hat{\imath} + b\hat{\jmath}$

$$\vec{v}_i = v_{i,x}\hat{\imath} + v_{i,y}\hat{\jmath} \quad (3\text{ 式})$$

$$\vec{v}_f = v_{f,x}\hat{\imath} + v_{f,y}\hat{\jmath} \quad (4\text{ 式})$$

将 4 式 3 式代入 1 式 2 式可得

$$v_{f,x} = (1-2a^2)v_{i,x} - 2abv_{i,y}$$
$$v_{f,y} = (1-2b^2)v_{i,y} - 2abv_{i,x}$$

在经典情形下质点的运动：

$$\frac{dx}{dt} = v_x, \frac{dy}{dt} = v_y \quad (5\text{ 式})$$

首先讨论在矩形边界中的碰撞。

整个程序的主要难点在于碰撞的边界如何确定，显然在程序中边界是无法用两个方程求解的出来的，那么我们怎么用手里的两个方程确定呢？我在这里用到的是二分法。我们用 IF，

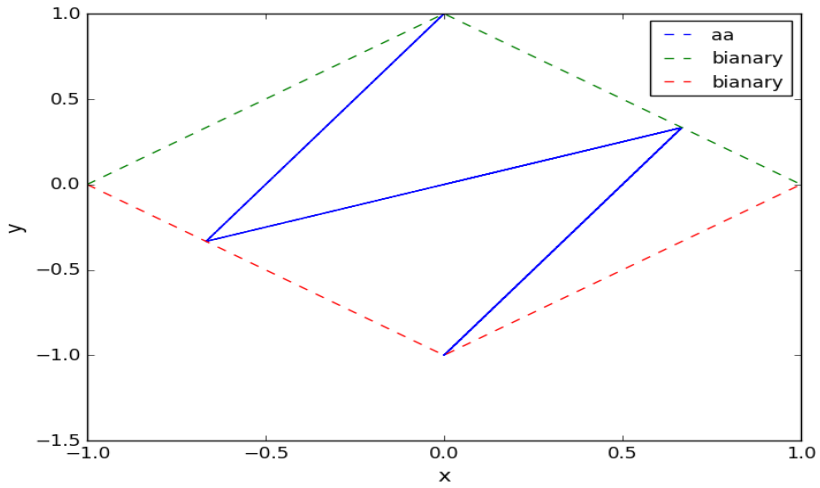
while 语句 if x[i+1]**2.0+(y[i+1]+alpha/2.0)**2.0>1:
            while abs(x[i+1]**2+(y[i+1]+alpha/2.0)**2-1)>0.0001
当然这里的 0.0001 是精度，越小表示越接近边界点，但是太小造成的计算量过大也是个问题。
其次就是话边界问题，对于这个封闭区间我们只需要分成 4 段边界函数即可
来写这个碰撞问题。
如下是完整程序：

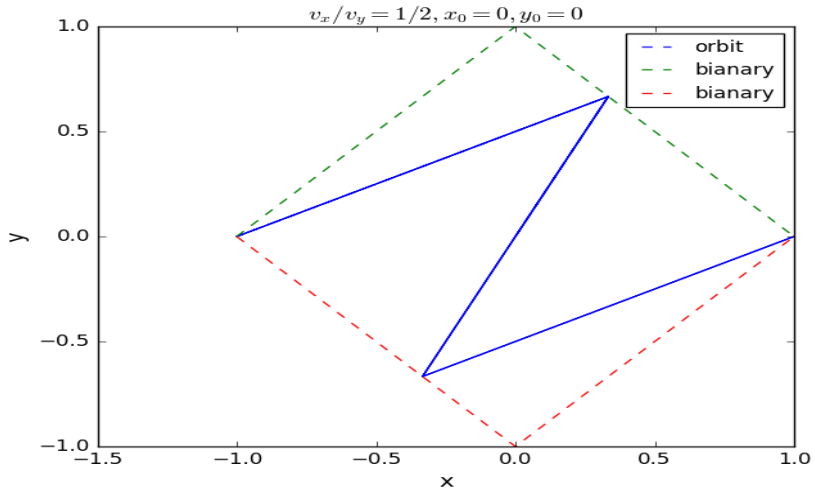
```
import math
x_0=0
y_0=0
v_x_0=1
v_y_0=0.002
dt=0.01
end_t=7000
x=[]
y=[]
v_x=[]
v_y=[]
t=[]
x.append(x_0)
y.append(y_0)
v_x.append(v_x_0)
v_y.append(v_y_0)
t.append(0)
alpha=0.001
v_x_1=[]
x_1_1=[]
for i in range(int(end_t/dt)):
    x.append(x[i]+v_x[i]*dt)
    y.append(y[i]+v_y[i]*dt)
    v_x.append(v_x[i])
    v_y.append(v_y[i])
    t.append(t[i]+dt)
    if abs(x[i+1]-1)<0.0001:
        print x[i+1]
        a=-x[i+1]
        b=0
        v_x[i+1]=(1-2*a**2)*v_x[i]-2*a*b*v_y[i]
        v_y[i+1]=(1-2*b**2)*v_y[i]-2*a*b*v_x[i]
    else:
        if y[i+1]<0:
            if  x[i+1]**2.0+(y[i+1]+alpha/2.0)**2.0>1:
              while abs(x[i+1]**2+(y[i+1]+alpha/2.0)**2-1)>0.0001:
                  #print abs(x[i+1]**2+y[i+1]**2-1)
```

```
                    X=(x[i]+x[i+1])/2
                    Y=(y[i]+y[i+1])/2
                    a=X/((X**2+(Y+alpha/2.0)**2)**0.5)
                    b=(Y+alpha/2.0)/((X**2+(Y+alpha/2.0)**2)**0.5)
                    v_x[i+1]=(1-2*a**2)*v_x[i]-2*a*b*v_y[i]
                    v_y[i+1]=(1-2*b**2)*v_y[i]-2*a*b*v_x[i]
                    #v_y[i+1]=-((2.0*a**3-2.0*a)/b)*v_x[i+1]-(2*a**2-1)*v_y[i+1]
                    if X**2+(Y+alpha/2.0)**2>1:
                        x[i+1]=X
                        y[i+1]=Y
                        continue
                    else:
                        x[i]=X
                        y[i]=Y
                        continue
            else:
                if    x[i+1]**2.0+(y[i+1]-alpha/2.0)**2.0>1:
                  while abs(x[i+1]**2+(y[i+1]-alpha/2.0)**2-1)>0.0001:
                    #print abs(x[i+1]**2+y[i+1]**2-1)
                    X=(x[i]+x[i+1])/2
                    Y=(y[i]+y[i+1])/2
                    a=X/((X**2+(Y-alpha/2.0)**2)**0.5)
                    b=(Y-alpha/2.0)/((X**2+(Y-alpha/2.0)**2)**0.5)
                    v_x[i+1]=(1-2*a**2)*v_x[i]-2*a*b*v_y[i]
                    v_y[i+1]=(1-2*b**2)*v_y[i]-2*a*b*v_x[i]
                    #v_y[i+1]=-((2.0*a**3-2.0*a)/b)*v_x[i+1]-(2*a**2-1)*v_y[i+1]
                    if X**2+(Y-alpha/2.0)**2>1:
                        x[i+1]=X
                        y[i+1]=Y
                        continue
                    else:
                        x[i]=X
                        y[i]=Y
                        continue
        if abs(y[i]-0)<0.0001:
            p=x[i]
            q=v_x[i]
            v_x_1.append(q)
            x_1_1.append(p)
print v_x[30]
x_1=[-1]
y_1=[0]
x_2=[-1]
y_2=[0]
```

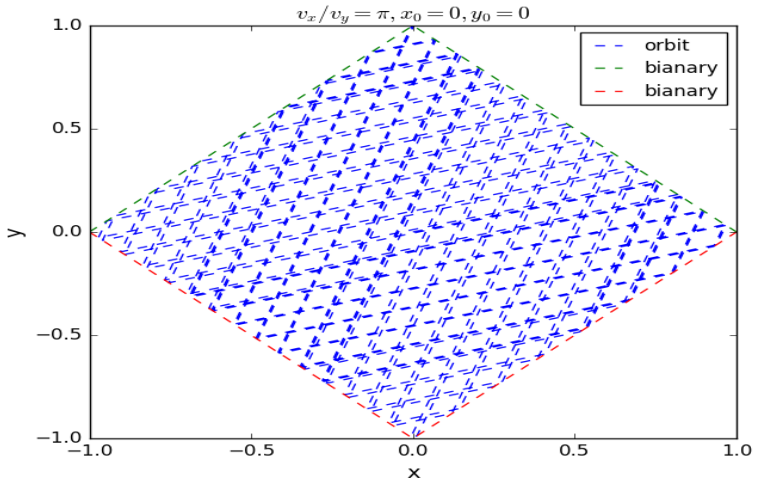
```
dx=0.01
for j in range(199):
    x_1.append(x_1[j]+dx)
    y_1.append(alpha/2.0+(1-x_1[j+1]**2)**0.5)
    x_2.append(x_2[j]+dx)
    y_2.append(-alpha/2.0-(1-x_2[j+1]**2)**0.5)
import matplotlib.pyplot as plt
import numpy as np
import math
fig= plt.figure(figsize=(7,6))
#fig= plt.figure(figsize=(7,6))
plt.scatter(x_1_1,v_x_1,marker='+',color='b',label='phase space plot')
#plt.plot(x,y,'--' ,label='orbit')
#plt.plot(x_1,y_1,'--',label='bianary')
#plt.plot(x_2,y_2,'--',label='bianary')
plt.title(u'$v_x/v_y=1/0.002,y_0=0,x_0=0,a=0.001,scan-point-y=0$',fontsize=16)
plt.xlabel(u'x',fontsize=14)
plt.ylabel(u'y',fontsize=14)
plt.legend(fontsize=12,loc='best')
plt.show(fig)
```

得到的图形

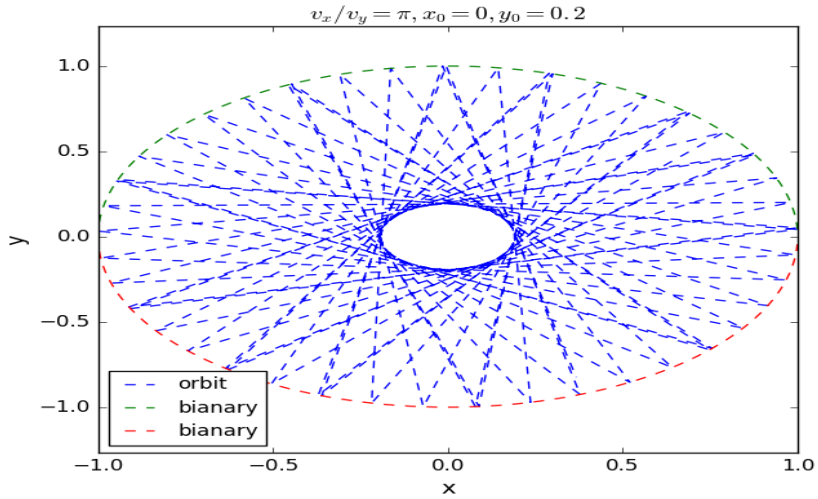




修改取点总时间 endt 得到碰撞多次的图形



修改边界条件得到椭圆边界的情况：



其次我们做出相空间的点的位置：

其程序为

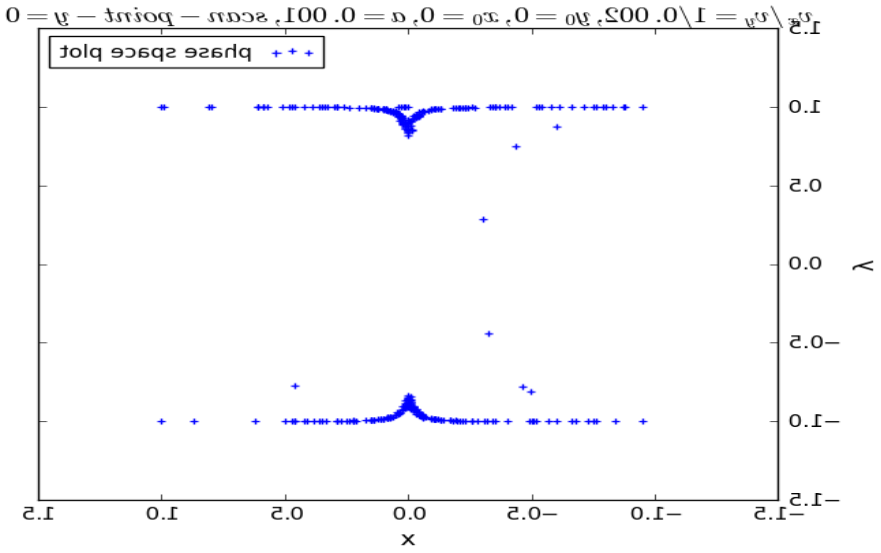
```
import math
```

```
x_0=0
y_0=0
v_x_0=1
v_y_0=0.002
dt=0.01
end_t=7000
x=[]
y=[]
v_x=[]
v_y=[]
t=[]
x.append(x_0)
y.append(y_0)
v_x.append(v_x_0)
v_y.append(v_y_0)
t.append(0)
alpha=0.001
v_x_1=[]
x_1_1=[]
for i in range(int(end_t/dt)):
    x.append(x[i]+v_x[i]*dt)
    y.append(y[i]+v_y[i]*dt)
    v_x.append(v_x[i])
    v_y.append(v_y[i])
    t.append(t[i]+dt)
    if abs(x[i+1]-1)<0.0001:
        print x[i+1]
        a=-x[i+1]
        b=0
        v_x[i+1]=(1-2*a**2)*v_x[i]-2*a*b*v_y[i]
        v_y[i+1]=(1-2*b**2)*v_y[i]-2*a*b*v_x[i]
    else:
        if y[i+1]<0:
            if    x[i+1]**2.0+(y[i+1]+alpha/2.0)**2.0>1:
              while abs(x[i+1]**2+(y[i+1]+alpha/2.0)**2-1)>0.0001:
                  #print abs(x[i+1]**2+y[i+1]**2-1)
                  X=(x[i]+x[i+1])/2
                  Y=(y[i]+y[i+1])/2
                  a=X/((X**2+(Y+alpha/2.0)**2)**0.5)
                  b=(Y+alpha/2.0)/((X**2+(Y+alpha/2.0)**2)**0.5)
                  v_x[i+1]=(1-2*a**2)*v_x[i]-2*a*b*v_y[i]
                  v_y[i+1]=(1-2*b**2)*v_y[i]-2*a*b*v_x[i]
                  #v_y[i+1]=-((2.0*a**3-2.0*a)/b)*v_x[i+1]-(2*a**2-1)*v_y[i+1]
                  if X**2+(Y+alpha/2.0)**2>1:
```

```
                    x[i+1]=X
                    y[i+1]=Y
                    continue
                else:
                    x[i]=X
                    y[i]=Y
                    continue
        else:
            if    x[i+1]**2.0+(y[i+1]-alpha/2.0)**2.0>1:
              while abs(x[i+1]**2+(y[i+1]-alpha/2.0)**2-1)>0.0001:
                #print abs(x[i+1]**2+y[i+1]**2-1)
                X=(x[i]+x[i+1])/2
                Y=(y[i]+y[i+1])/2
                a=X/((X**2+(Y-alpha/2.0)**2)**0.5)
                b=(Y-alpha/2.0)/((X**2+(Y-alpha/2.0)**2)**0.5)
                v_x[i+1]=(1-2*a**2)*v_x[i]-2*a*b*v_y[i]
                v_y[i+1]=(1-2*b**2)*v_y[i]-2*a*b*v_x[i]
                #v_y[i+1]=-((2.0*a**3-2.0*a)/b)*v_x[i+1]-(2*a**2-1)*v_y[i+1]
                if X**2+(Y-alpha/2.0)**2>1:
                    x[i+1]=X
                    y[i+1]=Y
                    continue
                else:
                    x[i]=X
                    y[i]=Y
                    continue
    if abs(y[i]-0)<0.0001:
        p=x[i]
        q=v_x[i]
        v_x_1.append(q)
        x_1_1.append(p)
print v_x[30]
x_1=[-1]
y_1=[0]
x_2=[-1]
y_2=[0]
dx=0.01
for j in range(199):
    x_1.append(x_1[j]+dx)
    y_1.append(alpha/2.0+(1-x_1[j+1]**2)**0.5)
    x_2.append(x_2[j]+dx)
    y_2.append(-alpha/2.0-(1-x_2[j+1]**2)**0.5)
import matplotlib.pyplot as plt
import numpy as np
```

```
import math
fig= plt.figure(figsize=(7,6))
#fig= plt.figure(figsize=(7,6))
plt.scatter(x_1_1,v_x_1,marker='+',color='b',label='phase space plot')
#plt.plot(x,y,'--' ,label='orbit')
#plt.plot(x_1,y_1,'--',label='bianary')
#plt.plot(x_2,y_2,'--',label='bianary')
plt.title(u'$v_x/v_y=1/0.002,y_0=0,x_0=0,a=0.001,scan-point-y=0$',fontsize=16)
plt.xlabel(u'x',fontsize=14)
plt.ylabel(u'y',fontsize=14)
plt.legend(fontsize=12,loc='best')
plt.show(fig)
```

做出来的图形如图为：



中间的点很稀少具体原因我也不太明确。

我们在中间设置一个带电场的情形，这样我们以前的速度设置也必须改变，

我们舍在圆环中存在电场，

具体程序

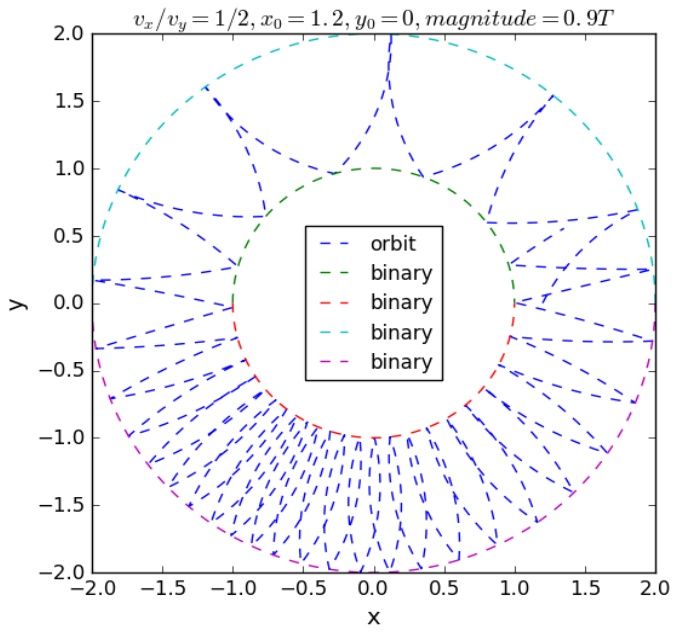
```
import math
x_0=1.2
y_0=0
v_x_0=1
v_y_0=2
dt=0.01
end_t=25
x=[]
y=[]
v_x=[]
v_y=[]
t=[]
```

```
x.append(x_0)
y.append(y_0)
v_x.append(v_x_0)
v_y.append(v_y_0)
t.append(0)
#v_x_1=[]
#x_1_1=[]
q=1.6*10**-19
m=1*10**-19
b=0.9
for i in range(int(end_t/dt)):
    x.append(x[i]+v_x[i]*dt)
    y.append(y[i]+v_y[i]*dt)
    #v_x.append(v_x[i])
    #v_y.append(v_y[i])
    v_x.append(v_x[i]+q*b*v_y[i]*dt/m)
    v_y.append(v_y[i]-q*b*v_x[i]*dt/m)
    t.append(t[i]+dt)
    #print x[i]**2+y[i]**2
    if  x[i+1]**2+y[i+1]**2>4:
        print  x[i+1]**2+y[i+1]**2
        while abs(x[i+1]**2+y[i+1]**2-4)>0.001:
            #print abs(x[i+1]**2+y[i+1]**2-4)
            X=(x[i]+x[i+1])/2
            Y=(y[i]+y[i+1])/2
            a=X/(X**2+Y**2)**0.5
            b=Y/(X**2+Y**2)**0.5
            v_x[i+1]=(1-2*a**2)*v_x[i]-2*a*b*v_y[i]
            v_y[i+1]=(1-2*b**2)*v_y[i]-2*a*b*v_x[i]
            #v_y[i+1]=-((2.0*a**3-2.0*a)/b)*v_x[i+1]-(2*a**2-1)*v_y[i+1]
            if X**2+Y**2>4:
                x[i+1]=X
                y[i+1]=Y
                continue
            else:
                x[i]=X
                y[i]=Y
                continue
        #if abs(abs(X)+abs(Y)-1)<0.1:
           # break
    if  x[i+1]**2+y[i+1]**2<1:
        while abs(x[i+1]**2+y[i+1]**2-1)>0.001:
            X=(x[i]+x[i+1])/2
            Y=(y[i]+y[i+1])/2
```

```
                    a=X/(X**2+Y**2)**0.5
                    b=Y/(X**2+Y**2)**0.5
                    v_x[i+1]=(1-2*a**2)*v_x[i]-2*a*b*v_y[i]
                    v_y[i+1]=(1-2*b**2)*v_y[i]-2*a*b*v_x[i]
                    #v_y[i+1]=-((2.0*a**3-2.0*a)/b)*v_x[i+1]-(2*a**2-1)*v_y[i+1]
                    if X**2+Y**2<1:
                        x[i+1]=X
                        y[i+1]=Y
                        continue
                    else:
                        x[i]=X
                        y[i]=Y
                        continue
        #if abs(y[i])<0.0001:
          #    p=x[i]
          #    q=v_x[i]
          #    v_x_1.append(q)
          #    x_1_1.append(p)
x_1=[-1]
y_1=[0]
x_2=[-1]
y_2=[0]
x_3=[-2]
y_3=[0]
x_4=[-2]
y_4=[0]
dx=0.0001
for j in range(20000):
        x_1.append(x_1[j]+dx)
        y_1.append((1-x_1[j+1]**2)**0.5)
        x_2.append(x_1[j]+dx)
        y_2.append(-(1-x_1[j+1]**2)**0.5)
for k in range(40000):
        x_3.append(x_3[k]+dx)
        y_3.append((4-x_3[k+1]**2)**0.5)
        x_4.append(x_4[k]+dx)
        y_4.append(-(4-x_4[k+1]**2)**0.5)
#x_4=x_4[1:39998]
#y_4=y_4[1:39998]
import matplotlib.pyplot as plt
import numpy as np
import math
fig= plt.figure(figsize=(6,6))
#plt.scatter(x_1_1,v_x_1,marker='+',color='b',label='phase space plot')
```

```
plt.plot(x,y,'--',label='orbit')
plt.plot(x_1,y_1,'--',label='binary')
plt.plot(x_2,y_2,'--',label='binary')
plt.plot(x_3,y_3,'--',label='binary')
plt.plot(x_4,y_4,'--',label='binary')
plt.title(u'$v_x/v_y=1/2,x_0=1.2,y_0=0,magnitude=0.9T$',fontsize=14)
plt.xlabel(u'x',fontsize=14)
plt.ylabel(u'y',fontsize=14)
plt.legend(fontsize=12,loc='best')
plt.show(fig)
```

如图



结论：1：在不规则圆形中很容易造成混沌现象

2：细微的初始状态很容易改变碰撞的整体路径

3：非对称的比例在百分之五以上是就可以出现大规模混沌

4：带电场将影响各地方轨迹的疏密程度，并且出现混沌。

引用和致谢：1。latex 数学符号表

2：Computational Physics, Nicholas J. Giordano & Hisao Nakanishi