**核安全综合保障**

**基于SPH方法的U材料力学性能探究**

|  |  |  |
| --- | --- | --- |
| 姓名 |  | 董毓龙 |
| 学号 |  | 201900501002 |

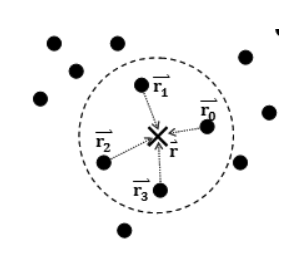
# 练习1：基于SPH方法的导数求解器

## 方法基本原理

SPH(Smoothed Particle Hydrodynamics)法，中文称其为光滑粒子动力学。最早是为了模拟银河系中天体之间的碰撞与形成等等宇宙物理学的现象而提出的算法；主要用于处理流体动力学中传统网格方法难以处理的困难，具有无网格性，自适应性，与拉氏描述结合较好的特性。

## SPH方法采用光滑核函数，对Dirac delta函数进行近似

和其他流体力学中的数学方法类似，SPH算法同样涉及到“光滑核”的概念，可以这样理解这个概念，粒子的属性都会“扩散”到周围，并且随着距离的增加影响逐渐变小，这种随着距离而衰减的函数被称为“光滑核”函数，最大影响半径为“光滑核半径”。反过来不难理解，尽管我们将流体视为一个个分散的粒子，但流体毕竟是连续充满整个空间的，流体中每个位置参与运算的值都是由周围一组粒子累加起来的。



场函数的光滑近似为：

狄拉克函数：，光滑函数，光滑函数在趋近狄拉克函数的同时保持物理意义。

光滑函数具有以下性质：

1. 光滑函数在h趋于0的极限为狄拉克函数
2. 正则性（归一性）
3. 紧支性
4. 非负性
5. 偶函数性
6. 单调递减性
7. 光滑性

应当足够光滑（连续多阶可微），通常改写为相对距离的形式

本次作业运用的光滑函数：钟型函数

其中，

光滑近似后的微分计算：

由散度定理：

高斯定理：

化简后：

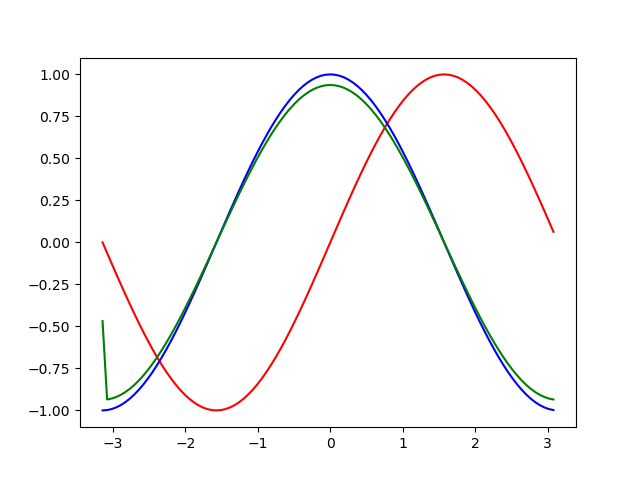
场函数导数的SPH积分表达式允许空间变量梯度由场函数本身的值和光滑函数W的导数来确定，而非函数本身的导数。

## 利用SPH方法进行导数求解

1. 将模型离散为一系列粒子，采用粒子近似计算积分：

无限自由度连续模型离散为有限个粒子所携带的自由度；各个离散粒子上携带有全部的物理量信息即确定场函数f。

1. 粒子描述下的积分计算：



## 4.可执行源代码

import math

import numpy as np

import matplotlib.pyplot as plt

a=[]

x=[]

y=[]

z=[]

for i in np.arange(-math.pi,math.pi,2\*math.pi/100):

a.append(i)

x.append(math.sin(i))

y.append(math.cos(i))

dx=2\*math.pi/100

for i in a:

h=2\*dx

ad=5/(4\*h)

s=0

for j in a:

k=(i-j)/abs(i-j)

if i==j:

k=0

if i!=j:

r=abs(i-j)/h

if r>1:

dw=0

else:

dw=ad\*(3\*(1-r)\*\*3-3\*(1-r)\*\*2\*(1+3\*r))\*(1/h)\*k

s=s+dx\*math.sin(j)\*dw

z.append(s)

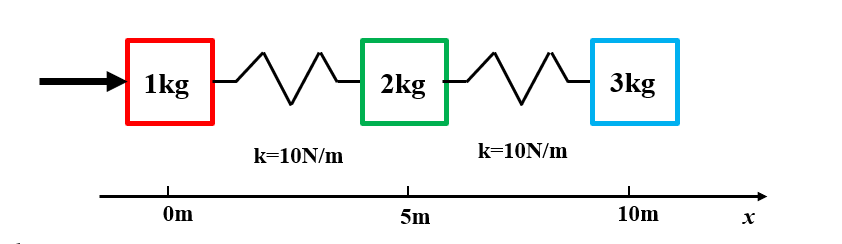
plt.plot(a, x,'r')

plt.plot(a, y,'b')

plt.plot(a, z,'g')

练习2：三物块弹簧链接动力学计算

## 基本物理模型



系统内每一个质量块仍然满足控制方程组：

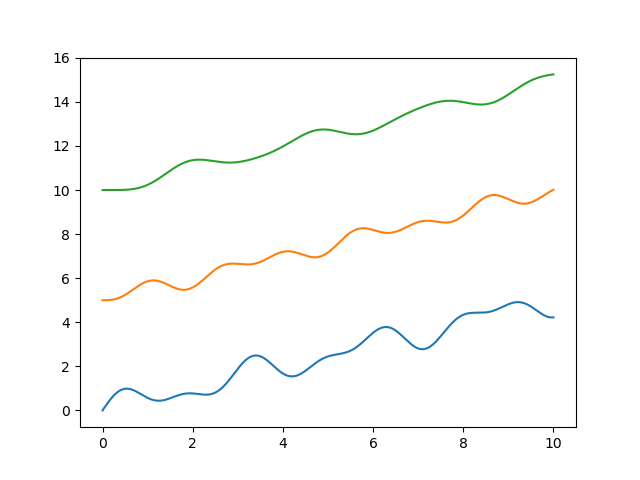
质量守恒：

动量守恒：

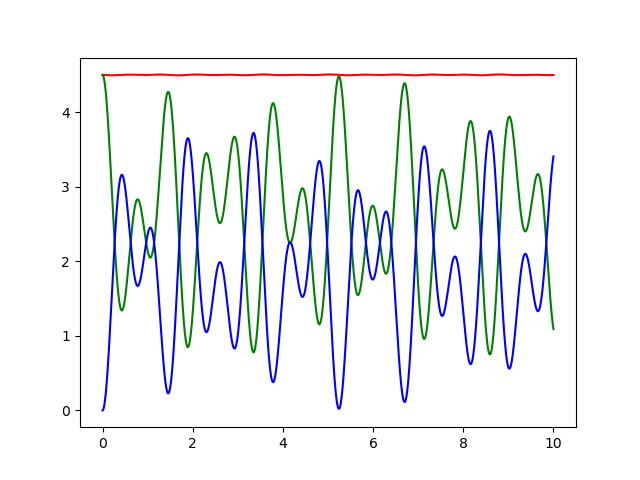
状态方程：

胡克定律：

三个物体的运动规律图像：



验证能量守恒：



## 2.可执行源代码

import numpy as np

import matplotlib.pyplot as plt

#数据初始化

k=10

t\_total=10

dt=0.001

x0\_a=0

m\_a=1

v0\_a=3

x0\_b=5

m\_b=2

v0\_b=0

x0\_c=10

m\_c=3

v0\_c=0

x\_a=[x0\_a]

x\_b=[x0\_b]

x\_c=[x0\_c]

v\_a=[v0\_a]

v\_b=[v0\_b]

v\_c=[v0\_c]

t1=[0]

i=0

ek=[4.5]

et=[0]

e=[4.5]

for t in np.arange(0,t\_total,dt):

xa=x\_a[i]+v\_a[i]\*dt

xb=x\_b[i]+v\_b[i]\*dt

xc=x\_c[i]+v\_c[i]\*dt

a\_reciptent\_a = k\*(xb-xa-x0\_b)/m\_a

a\_reciptent\_b = -k\*(xb-xa-x0\_b)/m\_b + k\*(xc-xb-(x0\_c-x0\_b))/m\_b

a\_reciptent\_c = -k\*(xc-xb-(x0\_c-x0\_b))/m\_c

va=v\_a[i]+a\_reciptent\_a\*dt

vb=v\_b[i]+a\_reciptent\_b\*dt

vc=v\_c[i]+a\_reciptent\_c\*dt

ek1=0.5\*m\_a\*va\*va+0.5\*m\_b\*vb\*vb+0.5\*m\_c\*vc\*vc

et1=0.5\*k\*(xa-xb+5)\*\*2+0.5\*k\*(xb-xc+5)\*\*2

e1=ek1+et1

ek.append(ek1)

et.append(et1)

e.append(e1)

x\_a.append(xa)

x\_b.append(xb)

x\_c.append(xc)

v\_a.append(va)

v\_b.append(vb)

v\_c.append(vc)

t1.append(t)

i+=1

#plt.plot(t1,e,'r') 能量守恒

#plt.plot(t1,ek,'g')

#plt.plot(t1,et,'b')

plt.plot(t1,x\_a)

plt.plot(t1,x\_b)

plt.plot(t1,x\_c)