**核安全综合保障**

**基于SPH方法的U材料力学性能探究**

|  |  |  |
| --- | --- | --- |
| 姓名 |  | 杜星举 |
| 学号 |  | 201900501003 |

# 练习1：基于SPH方法的导数求解器

## 1.SPH方法基本原理

1. SPH基本思想

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。

1. 主要求解的问题

数学上：将偏微分方程组在问题域内离散化；获取任意一点的变量函数及其导数的近似值；将近似函数应用于偏微分方程组来获得一系列离散化的、只与时间相关的常微分方程。

物理上：对NS方程进行空间域离散化，推导出适用于广义流体动力学的SPH方程，得到一系列与时间相关的常微分方程。这一系列常微分方程可以通过对时间的积分进行求解。

1. 优势

SPH方法是一种纯拉格朗日性质的无网格粒子自适应算法，在处理大变形、跟踪运动界面或自由表面，以及获取变量的时间历程等方面有着很大优势。

1. 缺点

相较网格数值法，尽管SPH体现出了不俗的能力，但是其在数学严谨性上的短板使得SPH并未被CFD社区所广泛容纳。从细节方面讲，SPH对于流体动态压力的计算并不准确，这也导致了SPH在模拟局部现象时精度较低，例如流体边界层，涡旋等等。

1. 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。

## 2.可执行源代码

**Matlab代码**

syms x1 x2 N h length dilf;

x1=input('x1=');

x2=input('x2=');

N=input('N=');

hanshu=input('求导的原函数=',"s");

length=x2-x1;

h=2.7\*length/N;

a=5/(4\*h);

dd=[];

xx=[];

x=0;

xj=0;

x0=0;

for i=x1:length/N:x2

dilf=0;

for j=i-length/2:length/N:i+length/2

if j~=i

d=length/N;

R=abs(i-j)/h;

dr=(i-j)/(abs(i-j)\*h);

if R>1

dilw=0;

else

dilw=-a\*12\*R\*(1-R)^2\*dr;

end

x=j;

dilf=dilf+d\*eval(hanshu)\*dilw;

else

continue

end

end

dd=[dd;dilf];

xx=[xx;i];

end

y2=max(dd);

y1=min(dd);

figure

hold on;

h1=plot(xx,dd,'rx-','linewidth',2,'MarkerSize',2);

xlim([x1 x2]);

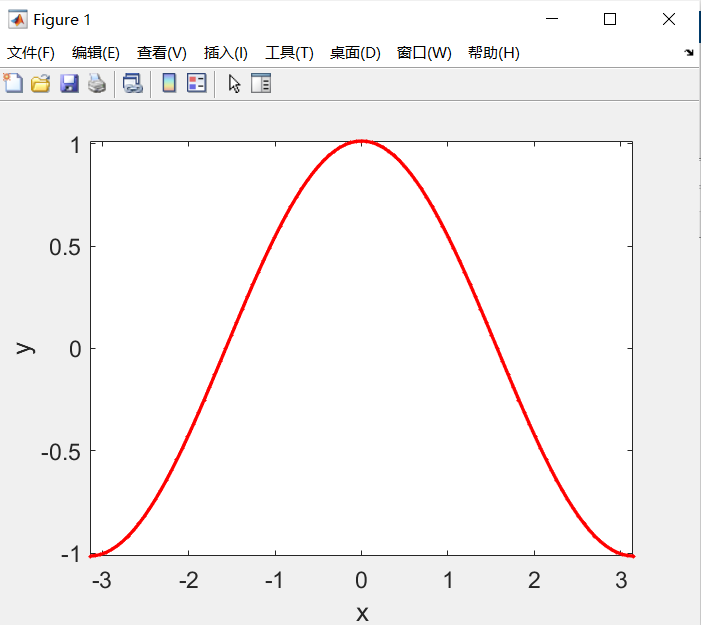
ylim([y1 y2]);

set(gca,'fontsize',14);

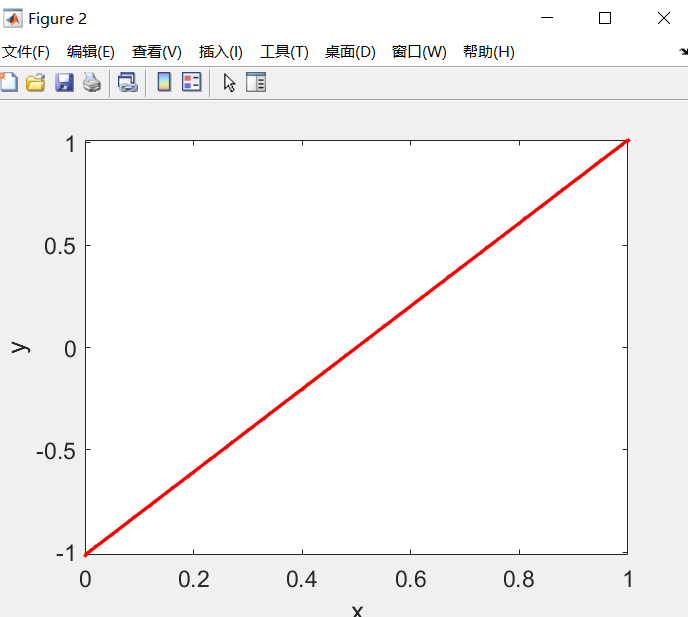
xlabel('x')

ylabel('y');

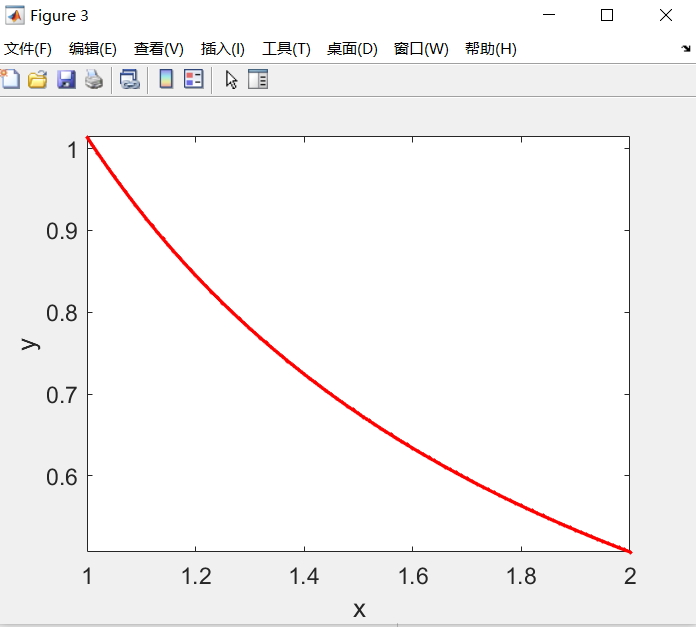
box on;

****

**f(x)=sinx**

****

**F（x）=x^2-x+1**

****

**F（x）=lnx**

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：

## 2. 可执行源代码

**Matlab代码**

%初始条件

%质量

m1=1;

m2=2;

m3=3;

%位置

x1=1;

x2=5;

x3=10;

%弹簧弹性系数

k1=10;

k2=10;

%弹簧原长

yc1=5;

yc2=5;

%速度

v1=0;

v2=0;

v3=0;

%时间步长

t2=10;

t1=0;

length=t2-t1;

N=10000;

%初始化

x11=[];

x22=[];

x33=[];

tt=[];

epp=[];

ekk=[];

ee=[];

for t=t1:length/N:t2

dt=length/N;%时间步长

%位置数组

x11=[x11;x1];

x22=[x22;x2];

x33=[x33;x3];

tt=[tt;t];

%受力计算

dx1=x2-x1;

dx2=x3-x2;

F1=k1\*(dx1-yc1);

F2=k2\*(dx2-yc2);

%加速度计算

a1=F1/m1;

a2=(F2-F1)/m2;

a3=-F2/m3;

%速度更新

v1=v1+a1\*dt;

v2=v2+a2\*dt;

v3=v3+a3\*dt;

%能量计算

ep=1/2\*k1\*(dx1-yc1)^2+1/2\*k2\*(dx2-yc2)^2;

ek=1/2\*m1\*v1^2+1/2\*m2\*v2^2+1/2\*m3\*v3^2;

e=ep+ek;

epp=[epp;ep];

ekk=[ekk;ek];

ee=[ee;e];

%位置更新

x1=x1+v1\*dt;

x2=x2+v2\*dt;

x3=x3+v3\*dt;

end

%x-t图像

figure

plot(tt,x11,'linewidth',2,'MarkerSize',2)

hold on

plot(tt,x22,'linewidth',2,'MarkerSize',2)

hold on

plot(tt,x33,'linewidth',2,'MarkerSize',2)

hold on

legend('m1','m2','m3')

xlabel('时间/s')

ylabel('位置/m')

set(gca,'fontsize',14);

%e-t图像

figure

plot(tt,epp,'linewidth',2,'MarkerSize',2)

hold on

plot(tt,ekk,'linewidth',2,'MarkerSize',2)

hold on

plot(tt,ee,'linewidth',2,'MarkerSize',2)

hold on

legend('ep','ek','e')

xlabel('时间/s')

ylabel('能量/J')

set(gca,'fontsize',14)

