1.数值积分

1.1 复合梯形公式

定义：将区间$[a,b]$划分为n等份，分点 $x_k=a+kh, h=\frac{b-a}{n}, k=0,1,\cdots,n$，在每个子区间 $[x_k,x_{k+1}](k=0,1,\cdots,n-1)$ 上采用梯形公式，即

$$I=\int_a^b f(x)dx=\sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)dx=\frac{h}{2}\sum_{k=0}^{n-1}[f(x_k)+f(x_{k+1})]+R_n(f)$$

记

$$T_n=\frac{h}{2}\sum_{k=0}^{n-1}[f(x_k)+f(x_{k+1})]=\frac{h}{2}[f(a)+\sum_{k=1}^{n-1}f(x_k)+f(b)]$$

即

$$I=T_n+R_n(f)$$

翻译上述代码的方式非常简单，优化的角度上思考，将累加变成向量化操作。在此不赘述，代码如下：

```
def CompoundTrapezoidal(f, a, b, n):
    h = (b - a) / n
    s = 0.5 * (f(a) + f(b))
    xk = np.arange(a + h, b, h) #向量化操作
    s += np.sum(f(xk))
    return s * h
```

1.2 复合辛普森公式

定义：将区间$[a,b]$划分为n等份，在每个子区间 $[x_k,x_{k+1}]$ 上采用辛普森公式，即 $x_{k+1/2}=x_k+\frac{1}{2}h$, 即

$$I=\int_a^b f(x)dx=\sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)dx=\frac{h}{6}\sum_{k=0}^{n-1}[f(x_k)+4f(x_{k+1/2})+f(x_{k+1})]+R_n(f)$$

积分项为

$$S_n=\frac{h}{6}[f(a)+4\sum_{k=0}^{n-1}f(x_{k+1/2})+2\sum_{k=1}^{n-1}f(x_k)+f(b)]$$

即

$$I=S_n+R_n(f)$$

翻译上述代码的方式非常简单，优化的角度上思考，将累加变成向量化操作。在此不赘述，代码如下：

```
def CompoundSimpson(f, a, b, n):
    h = (b - a) / n
    s = f(a) + f(b)
    xk = np.arange(a + h, b, h)
    s += 2 * np.sum(f(xk))
    xk = np.arange(0.5*h, b, h)
    s += 4 * np.sum(f(xk))
    return s * h / 6
```

1.3 Romberg法

通过复合求积方法的思想，当计算精度不足时，将积分区间 $[a,b]$ 逐级二分，并将二分前后的两个积分值联合考察，可以得到每次二分后的积分递推公式为：

$$T_{2n}=\frac{1}{2}T_n+\frac{b-a}{2^n}\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})$$

逐次二分可以提高求积公式的精度，可以表示为

$$T(h)=I+\frac{b-a}{12}h^2f''(\eta),\lim_{h\to 0}T(h)=T(0)=I$$

其余项可以展开为级数形式，有定理描述为

$$T(h)=I+\alpha_1h^2+\alpha_2h^4+\cdots+\alpha_lh^{2l}+\cdots$$

其中，系数α_l与h无关

对其进行外推，将h数次减半，可得到Romberg求积算法：

$$T_m^{(k)}=\frac{4^m}{4^m-1}T_{m-1}^{(k+1)}-\frac{1}{4^m-1}T_{m-1}^{(k)}$$

计算过程为：

1. 取$k=0,h=b-a$.求
   $T_0^0=\frac{b-a}{2}[f(a)+f(b)]$
2. 求梯形值$T_0(\frac{b-a}{2^k})$,依据递推公式
3. 依据Romberg法求加速值
4. 判断精度，否则k+1

衍生的T表为：

k	h	$T_0^{(k)}$	$T_1^{(k)}$	$T_2^{(k)}$	$T_3^{(k)}$	$T_4^{(k)}$	$\cdots$
0	$b-a$	$T_0^{(0)}$					
1	$\frac{b-a}{2}$	$T_0^{(1)}$ ↓①	$T_1^{(0)}$				
2	$\frac{b-a}{4}$	$T_0^{(2)}$ ↓②	$T_1^{(1)}$ ↓③	$T_2^{(0)}$			
3	$\frac{b-a}{8}$	$T_0^{(3)}$ ↓④	$T_1^{(2)}$ ↓⑤	$T_2^{(1)}$ ↓⑥	$T_3^{(0)}$		
4	$\frac{b-a}{16}$	$T_0^{(4)}$ ↓⑦	$T_1^{(3)}$ ↓⑧	$T_2^{(2)}$ ↓⑨	$T_3^{(1)}$ ↓⑩	$T_4^{(0)}$	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

1.4 GaussLegendre法

Gauss求积公式的一半原理为

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{k=0}^{n} A_k f\left(x_k\right)$$

即通过求解2n+2个参数的待定系数方程得到插值求积公式。当 $[a,b]$ 为 $[-1,1]$ 时，注意到勒让德多项式在此区间上是正交的。因此可以将勒让德多项式的零点作为高斯求积公式的高斯点。称为 $Gauss-Legendre$ 求积公式。

下表给出了 $Gauss-Legendre$ 求积公式的n+1点求积公式系数。

n	x_k	A_k	n	x_k	A_k
0	0.000 000 0	2.000 000 0	3	±0.861 136 3 ±0.339 981 0	0.347 854 8 0.652 145 2
1	±0.577 350 3	1.000 000 0	4	±0.906 179 8 ±0.538 469 3 0.000 000 0	0.236 926 9 0.478 628 7 0.568 888 9
2	±0.774 596 7 0.000 000 0	0.555 555 6 0.888 888 9	5	±0.932 469 5 ±0.661 209 4 ±0.238 619 2	0.171 324 5 0.360 761 6 0.467 913 9

对于三点高斯勒让德求积公式（n=2），为

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{5}{9} f\left(-\frac{\sqrt{15}}{5}\right)+\frac{8}{9} f(0)+\frac{5}{9} f\left(\frac{\sqrt{15}}{5}\right)$$

若被积函数积分区间处于 $[a,b]$，则通过变换

$$x=\frac{b-a}{2} t+\frac{a+b}{2}$$

改写为

$$\int_a^b f(x)\mathrm{d}x=\frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2} t+\frac{a+b}{2}\right)\mathrm{d}t$$

2 数值微分

2.1 三次样条微分

自然边界条件，通过分段定义可以得到三次样条插值的表达式为：

$$S(x) = M_j\frac{(x_{j+1}-x)^3}{6h_j} + M_{j+1}\frac{(x-x_j)^3}{6h_j} + \left(y_j - \frac{M_j h_j^2}{6}\right)\frac{x_{j+1}-x}{h_j} + \left(y_{j+1} - \frac{M_{j+1}h_j^2}{6}\right)\frac{x-x_j}{h_j}, \quad j = 0, 1, \cdots, n-1$$

其导数格式为：

$$S(x) = -3M_j\frac{(x_{j+1}-x)^2}{6h_j} + 3M_{j+1}\frac{(x-x_j)^2}{6h_j} + \left(y_j - \frac{M_j h_j^2}{6}\right)\frac{-1}{h_j} + \left(y_{j+1} - \frac{M_{j+1}h_j^2}{6}\right)\frac{1}{h_j}, \quad j = 0, 1, \cdots, n-1$$

即获得了三次样条微分的格式。

3 题目回答

Q1 使用组合梯形公式和组合辛普森公式求表面积

	$f(x)=x^3, 0\le x\le 1$	$f(x)=sin(x), 0\le x\le \pi/4$	$f(x)=e^{-x}, 0\le x\le 1$
组合梯形	3.642446639175053	2.4197237019545064	4.85802251835429
组合辛普森	3.5637281587877507	2.4224337903921787	4.849242825376741

代码如下：

```
import numpy as np
import matplotlib.pyplot as plt
from CompoundTrapezoidal import CompoundTrapezoidal as CT
from CompoundSimpson import CompoundSimpson as CS
"根据题目内容定义三个函数f以及其导函数df"
def f1(x):
    return x**3
def df1(x):
    return 3*x**2

def f2(x):
    return np.sin(x)
def df2(x):
    return np.cos(x)
```

```
def f3(x):
    return np.exp(-x)
def df3(x):
    return -np.exp(-x)

def S(x,f,df):
    return 2*np.pi*f(x)*np.sqrt(1+df(x)**2)

S1 = lambda x: S(x,f1,df1)
S2 = lambda x: S(x,f2,df2)
S3 = lambda x: S(x,f3,df3)


"计算复合梯形积分值"
num1 = CT(S1,0,1,10)
num2 = CT(S2,0,np.pi/4,10)
num3 = CT(S3,0,1,10)
print(num1,num2,num3)

"计算复合辛普森积分"
num1 = CS(S1,0,1,5)
num2 = CS(S2,0,np.pi/4,5)
num3 = CS(S3,0,1,5)
print(num1,num2,num3)
```

Q2 试编程求解以下定积分的近似值

	$\int_0^3 \frac{sin(2x)}{1+x^5}dx$	$\int_0^2(\sqrt{4x}-x^2)dx$
复合梯形	0.6563770521962247	1.0564050764636244
复合辛普森	0.6709757573886819	1.090047919738746
三点高斯	0.5360674977872105	1.118784989031315
Romberg	0.6717578026299921	1.1045360231991097

代码：

```
import numpy as np
import matplotlib.pyplot as plt
from CompoundTrapezoidal import CompoundTrapezoidal as CT
from CompoundSimpson import CompoundSimpson as CS
from GaussLegendre import GaussLegendre3 as GL3
from Romberg import Romberg as R

def f1(x):
    return np.sin(2*x)/(1+x**5)
```

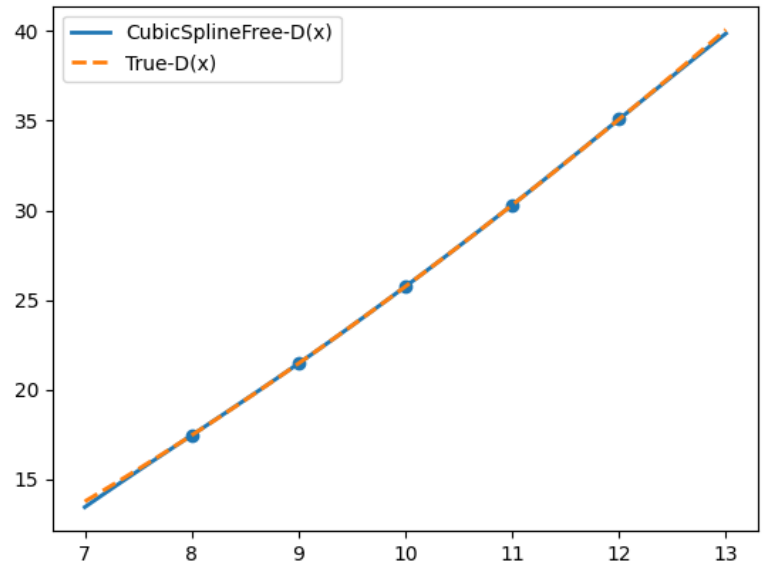
```
def f2(x):
    return np.sqrt(4*x)-x**2

"计算复合梯形积分值"
num1 = CT(f1,0,3,10)
num2 = CT(f2,0,2,10)
print(num1,num2)

"计算复合辛普森积分"
num1 = CS(f1,0,3,5)
num2 = CS(f2,0,2,5)
print(num1,num2)

"计算三点高斯积分"
num1 = GL3(f1,0,3)
num2 = GL3(f2,0,2)
print(num1.res(),num2.res())

"计算Romberg积分"
num1 = R(f1,0,3,0.0001)
num2 = R(f2,0,2,0.0001)
print(num1.res(),num2.res())
```

Q3: 根据物体运动距离表求解数值微分并比较

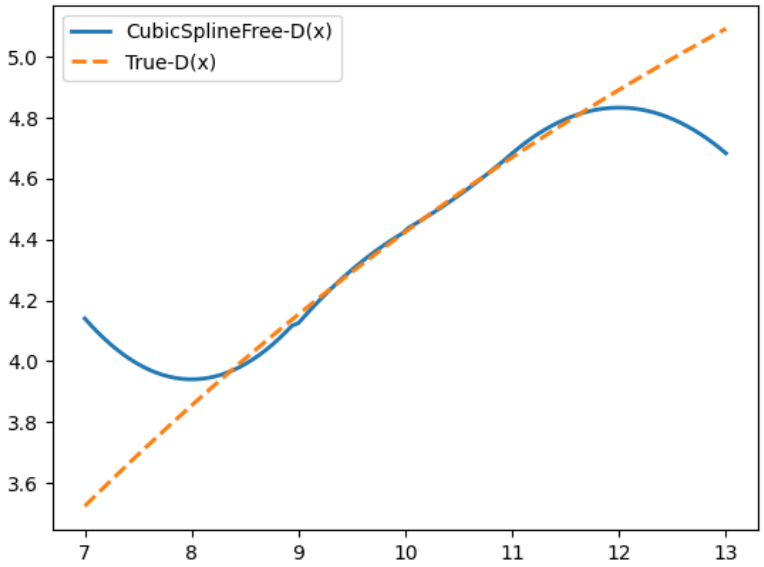
通过对三点自由样条插值的求导，经过计算

	V(10)
数值微分值	4.432333333333332
真实值	4.424843911799904

原函数的对比为：



导数的对比为：



代码为：

```
import numpy as np
import matplotlib.pyplot as plt
from CubicSplineFree import CubicSplineFree as CSF

def True_D(x):
    return -70+7*x+70*np.exp(-x/10)
def True_dD(x):
    return 7-7*np.exp(-x/10)

t = np.array([8,9,10,11,12])
```

```
D = np.array([17.453,21.460,25.752,30.301,35.084])
arr = np.c_[t,D]

csf = CSF(arr)
print(csf.dnum(10))
print(True_dD(10))

x = np.linspace(7,13,100)
y1 = np.array([])
y3 = np.array([])
y2 = True_dD(x)
y4 = True_D(x)
for i in x:
    y1 = np.append(y1,csf.dnum(i))
    y3 = np.append(y3,csf.num(i))
plt.figure()
plt.scatter(t,D)
plt.plot(x,y3,lw = 2,label='CubicSplineFree-D(x)')
plt.plot(x,y4,lw = 2,linestyle='--',label='True-D(x)')
plt.legend()
plt.show()

plt.figure()
plt.plot(x,y1,lw = 2,label='CubicSplineFree-D(x)')
plt.plot(x,y2,lw = 2,linestyle='--',label='True-D(x)')
plt.legend()
plt.show()
```

Q4: 试分别用分段三点公式和分段五点公式编程求节点处的一阶导数近似值

	0	0.2	0.4	0.6	0.8	1.0
三次样条	0.92583	0.79167	0.56617	0.40983	0.25950	0.19950
三点公式	1.1775	0.9165	0.6807	0.4896	0.3336	0.1932
五点公式	0.3233	0.2365	-0.67325	-0.48375	0.16565	0.3763

可以发现，随着点的增高，可能会由于龙格现象产生高次插值的病态现象。

代码如下：

```
import numpy as np
from CubicSplineFree import CubicSplineFree as CSF

x = np.array([0,0.2,0.4,0.6,0.8,1.0])
y = np.array([0.3927,0.5672,0.6982,0.7941,0.8614,0.9053])
arr = np.c_[x,y]
csf = CSF(arr)
```

```
for i in x:
    print(csf.dnum(i))

def point3(x,y):
    h = np.abs(x[-1]-x[0])/len(x)
    df = np.array([])
    df = np.append(df,(-3*y[0]+4*y[1]-y[2])/(2*h))
    for i in range(1,len(x)-1):
        df = np.append(df,(y[i+1]-y[i-1])/(2*h))
    df = np.append(df,(3*y[-1]-4*y[-2]+y[-3])/(2*h))
    return df

def point5(x,y):
    h = np.abs(x[-1]-x[0])/len(x)
    df = np.array([])
    df = np.append(df,(y[0]-8*y[1]+8*y[2]-y[3])/(12*h))
    df = np.append(df,(y[1]-8*y[2]+8*y[3]-y[4])/(12*h))
    for i in range(2,len(x)-2):
        df = np.append(df,(y[i+2]-8*y[i+1]+8*y[i-1]-y[i-2])/(12*h))
    df = np.append(df,(y[-4]-8*y[-3]+8*y[-2]-y[-1])/(12*h))
    df = np.append(df,(y[-3]-8*y[-2]+8*y[-1]-y[-0])/(12*h))
    return df

print(point3(x,y))
print(point5(x,y))
```