李燕琴 20195633

第五章 数值积分

5-12

（第五章 数值积分 李燕琴 20195633） 数值计算 Hw3

12. 解：(1) $f(x)=e^x$

$f^{(k)}(x)=e^x$

当 $0\leq x\leq 1$ 时，$f^{(k)}(x)\leq e$

$|E[f]| = \left|-\frac{1-0}{12}h^2 f''(\xi)\right| \leq \left|-\frac{h^2 e}{12}\right| \leq 10^{-6}$，$0\leq\xi\leq 1$

$h\leq 2.1\times 10^{-3}$

$n\geq\frac{1}{h}\geq\frac{1000}{2.1}$ 即 $n\geq 477$

$h=\frac{1}{477}$

$T_{477}=\frac{h}{2}\times\left[f(0)+f(1)+2\sum_{i=1}^{n-1}f(0+h\cdot i)\right]$ 编程计算得 1.718282.

(2) $|E[f]| = \left|-\frac{(1-0)h^4}{2880}f^{(4)}(\xi)\right| \leq \frac{eh^4}{2880}\leq 10^{-6}$

$h\leq 0.18$

$n\geq\frac{1}{h}=\frac{1}{0.18}$ 即 $n\geq 6$，$h=\frac{1}{6}$

$S_6=\frac{h}{6}\sum_{k=0}^{n-1}[f(x_k)+4f(x_{k+\frac{1}{2}})+f(x_{k+1})]$

编程计算得 $S_6=1.718282$

(3) step1：$T_1=\frac{f(0)+f(1)}{2}=1.859141$

step2：$T_2=[T_1+f(\frac{1}{2})]\times\frac{1}{2}=1.753931$

$S_1=\frac{4T_2-T_1}{3}=1.718861$

step3：$T_4=(T_2+\frac{1}{2}\times[f(\frac{1}{4})+f(\frac{3}{4})])\times\frac{1}{2}=1.727222$

$S_2=\frac{4T_4-T_2}{3}=1.718319$

$C_1=\frac{16S_2-S_1}{15}=1.718283$

step4：$T_8=(T_4+\frac{1}{4}\times[f(\frac{1}{8})+f(\frac{3}{8})+f(\frac{5}{8})+f(\frac{7}{8})])\times\frac{1}{2}=1.720519$

$S_4=\frac{4T_8-T_4}{3}=1.718284$

$C_2=\frac{16S_4-S_2}{15}=1.718282$

$R_1=\frac{64C_2-C_1}{63}=1.7182818$

step5：$T_{16}=(T_8+\frac{1}{8}\times[f(\frac{1}{16})+f(\frac{3}{16})+f(\frac{5}{16})+f(\frac{7}{16})+f(\frac{9}{16})+f(\frac{11}{16})+f(\frac{13}{16})+f(\frac{15}{16})])\times\frac{1}{2}$

$=1.718841$

$S_8=1.718282$

$C_4=1.718282$

$R_2=1.7182818$

代码：

```python
import numpy as np
import math
def fun(x):
    return math.exp(x)

def fuhua_tixing(n):
    h = 1/n
    sum_res = 0
    for i in range(1,n):
        sum_res = sum_res + 2*fun(h*i)
    res = h/2*(fun(0)+fun(1)+sum_res)
    return res

def fuhua_Simpson(n):
    h = 1/n
    sum_res = 0
    for i in range(0,n):
        sum_res = sum_res + fun(0+h*i) + 4*fun(0+h*(i+0.5)) + fun(h*(i+1))
    res = h/6*sum_res
    return res

def H2n(a,b,h): # 这里的h是指T1阶段的h
    x = a + h/2
    s = fun(x)
    while True:
        x = x + h # 注意，这里是h
        if x>b:
            return s*h
        s = s + fun(x)

def Romberg(a,b,e,outflag=True):
    if outflag:
        print("="*20+"Romberg中间过程"+"="*20)
    T1 = 0; S1 = 0; C1 = 0; R1 = 0
    T2 = 0; S2 = 0; C2 = 0; R2 = 0
    T = []; S = []; C = []; R = []
    h = b-a
    T1 = (fun(a)+fun(b))*h/2
    T.append(T1)
    k = 1
    while True:
        T2 = (T1+H2n(a,b,h))/2 # 这里的h是指T1阶段的h
        T.append(T2)
        S2 = (4*T2-T1)/3
        S.append(S2)
        if k>1:
            C2 = (16*S2-S1)/15
            C.append(C2)
        if k>2:
            R2 = (64*C2-C1)/63
            R.append(R2)
        if k>3:
            if np.abs(R2-R1)<e:    # 判断
                if outflag:
                    print(f"T = {T},\nS = {S},\nC = {C},\nR = {R}")
                return R2
        k = k+1
```

```python
        h = h/2
        T1 = T2; S1 = S2; C1 = C2; R1 = R2

print(f"复化梯形法：\t{fuhua_tixing(477)}")
print(f"复化Simpson法：\t{fuhua_Simpson(6)}")
print(f"Romberg算法：\t{Romberg(0,1,10**(-6),True)}")
```

5-18

题 18 解：① 根据 Gauss-Legendre 公式，取 Legendre 多项式 $L_n(x)$ 的零点作为高斯节点

<数值计算 Hw3>

$$L_n(x) = \frac{n!}{(2n)!}\frac{d^n(x^2-1)^n}{dx^n}$$ ，即 $L_3(x)=0$

则有 $\int_{-1}^{1} f(x)\,dx = \frac{5}{9}f(-\sqrt{\tfrac{3}{5}}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{\tfrac{3}{5}})$

当代入 $f(x)=x^2\cos x$ 得 $\int_{-1}^{1} x^2\cos x\,dx = 0.476469$

② Simpson 公式：$\int_{-1}^{1} f(x)\,dx = \frac{b-a}{6}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right]$

得 $\int_{-1}^{1} x^2\cos x\,dx = 0.360202$

③ 准确值计算

$$\int_{-1}^{1} x^2\cos x\,dx = (x^2\sin x + 2x\cos x - 2\sin x)\Big|_{-1}^{1} = 0.478267$$

计算代码：

```python
import numpy as np
def f(x):
    return x**2*np.cos(x)
def ff(x):
    return x**2*np.sin(x)+2*x*np.cos(x)-2*np.sin(x)
print(5/9*f(-1*np.sqrt(3/5))+8/9*f(0)+5/9*f(np.sqrt(3/5)))
print(2/6*(f(-1)+4*f(0)+f(1)))
print(ff(1)-ff(-1))
```

第六章 线性代数方程组的解法

6-9

为使Jacobi和Gauss_Seidel迭代公式收敛，需要使增广矩阵严格占优。

严格占优原理：

0、严格占优

矩阵中每个主对角元素的模都大于与它同行的其他元素的模的总和.这种矩阵就叫'严格对角占优的'；对列同样成立

1、最开始的A,b

$$\begin{cases} 11x_1 - 3x_2 - 33x_3 = 1 \\ -22x_1 + 11x_2 + x_3 = 0 \\ x_1 - 4x_2 + 2x_3 = 1 \end{cases}$$

2、为保证其严格占优，交换一下行

```python
A = np.array([[-22, 11, 1], \
              [1, -4, 2], \
              [11, -3, -33]])
print(A)
b = np.array([0, 1, 1])
```

代码实现：

```python
import numpy as np
import copy
import math
def Jacobi(n,A,b,x0,N,e):
    x1 = x0
    x2 = np.zeros(n)
    for k in range(0,N): # 迭代次数
        for i in range(0,n):
            _sum = 0
            for j in range(0,n):
                if i != j:
                    _sum += A[i][j]*x1[j]
            x2[i] = (b[i]-_sum)/A[i][i]
        print(f"第{k}次迭代：{np.max(np.abs(x2-x1))}\n\tx1={x1}\n\tx2={x2}")

        if np.max(np.abs(x2-x1))<e:
            return x2
        x1 = copy.deepcopy(x2) # 简单赋值，不传递引用
```

```python

def Gauss_Seidel(n,A,b,x0,N,e):
    x = x0
    for k in range(0,N): # 迭代次数
        max_abs = 0
        for i in range(0,n):
            t = x[i]
            sum1 = 0
            for j in range(0,n):
                if i != j:
                    sum1 += A[i][j]*x[j]
            x[i] = (b[i]-sum1)/A[i][i]
            if np.abs(x[i]-t)>max_abs:
                max_abs = np.abs(x[i]-t)
        if max_abs<e:
            return x
        print(f"第{k}次迭代：{max_abs}\n\tx={x}")

n = 3
A = np.array([[-22,11,1], \
             [1,-4,2], \
             [11,-3,-33]])
print(A)
b = np.array([0,1,1])
# x0 = np.random.random(n)
x0 = np.zeros(n)
print(x0)
e = 1e-3
N = int(1e3)
print("="*20+"Jacobi"+"="*20)
Jacobi(n,A,b,x0,N,e)

print("="*20+"Gauss_Seidel"+"="*20)
Gauss_Seidel(n,A,b,x0,N,e)
```

迭代结果：

```
====================Jacobi====================
第0次迭代：0.25
    x1=[0. 0. 0.]
    x2=[-0.         -0.25       -0.03030303]
第1次迭代：0.12637741046831955
    x1=[-0.         -0.25       -0.03030303]
    x2=[-0.12637741 -0.26515152 -0.00757576]
第2次迭代：0.04074839302112029
    x1=[-0.12637741 -0.26515152 -0.00757576]
    x2=[-0.13292011 -0.28538223 -0.04832415]
第3次迭代：0.022009871441689643
    x1=[-0.13292011 -0.28538223 -0.04832415]
    x2=[-0.14488767 -0.3073921  -0.04866589]
第4次迭代：0.011020469533805394
    x1=[-0.14488767 -0.3073921  -0.04866589]
    x2=[-0.15590814 -0.31055486 -0.05065418]
第5次迭代：0.003749261649806323б
    x1=[-0.15590814 -0.31055486 -0.05065418]
    x2=[-0.15757989 -0.31430413 -0.05404015]
第6次迭代：0.0021109224029520２
```

```
    x1=[-0.15757989 -0.31430413 -0.05404015]
    x2=[-0.15960843 -0.31641505 -0.05425656]
第7次迭代：0.0010652980430587988
    x1=[-0.15960843 -0.31641505 -0.05425656]
    x2=[-0.16067373 -0.31703039 -0.05474084]
第8次迭代：0.0005084632214585327
    x1=[-0.16067373 -0.31703039 -0.05474084]
    x2=[-0.16100341 -0.31753885 -0.05504   ]
===================Gauss_Seidel===================
第0次迭代：0.25
    x=[-0.         -0.25       -0.00757576]
第1次迭代：0.1253443526170799
    x=[-0.12534435 -0.28512397 -0.04616412]
第2次迭代：0.024123181476675082
    x=[-0.14466035 -0.30924715 -0.05040977]
第3次迭代：0.012254574836458837
    x=[-0.15691493 -0.31443362 -0.05402313]
第4次迭代：0.0027574780813259814
    x=[-0.15967241 -0.31692967 -0.05471538]
第5次迭代：0.0012794907536307631
    x=[-0.1609519  -0.31759566 -0.05508133]
```

6-10

见.md文件

~~10.解:~~

10、证明：充分性：若 $\lim_{k\to\infty}\|X^{(k)}-X^*\|_2=0$，则 $\forall\varepsilon>0$，∃整数$N(\varepsilon)$，使得当 $k\geq N(\varepsilon)$ 有

$$\sqrt{\sum_{i=1}^{n}(x_i^{(k)}-x_i^*)^2}=\|X^{(k)}-X^*\|_2<\varepsilon$$

则有 $|x_i^{(k)}-x_i^*|<\sqrt{\varepsilon}$，$i=1,2,\dots,n$

$$\therefore \lim_{k\to\infty}x_i^{(k)}=x_i^*,\ i=1,2,\dots,n$$

必要性：若 $\lim_{k\to\infty}x_i^{(k)}=x_i^*$，$i=1,2,\dots,n$，则 $\forall\varepsilon>0$，∃整数$N_i(\varepsilon)$，使得当 $k>N_i(\varepsilon)$ 有

$$|x_i^{(k)}-x_i^*|<\varepsilon,\ i=1,2,\dots,n \quad 且\ (x_i^{(k)}-x_i^*)^2<\varepsilon^2$$

令 $N(\varepsilon)=\max_{i=1,2\dots n}N_i(\varepsilon)$，当 $k\geq N(\varepsilon)$ 时，

$$\sqrt{\sum_{i=1}^{n}(x_i^{(k)}-x_i^*)^2}<\sqrt{n\varepsilon^2}=\sqrt{n}\,\varepsilon$$

$$\therefore \lim_{k\to\infty}\|X^{(k)}-X^*\|_2=0.$$

结论即证.