第八次作业

赖显松 2021214726

1 书面作业题目

P273 T1

解 (1) 梯形公式

$\int_0^1 e^{-x}dx \approx \frac{1-0}{2}(f(0)+f(1)) = \frac{1}{2}(1+\frac{1}{e}) = 0.684$

余项为 $-\frac{h^3}{12}f''(\xi) = -\frac{h^3}{12}\cdot e^{-\xi}$, $\xi \in [0,1]$ $h=b-a=1$

$\therefore$ 误差界 $\varepsilon = \max|-\frac{h^3}{12}e^{-\xi}| = \frac{1}{12} = 0.8333$

Simpson 公式

$\int_0^1 e^{-x}dx \approx \frac{1-0}{6}(f(0)+4f(\frac{0+1}{2})+f(1)) = \frac{1}{6}(1+4e^{-\frac{1}{2}}+e^{-1}) = 0.632$

余项为 $-\frac{(b-a)^5}{2880}f^{(4)}(\xi) = -\frac{(b-a)^5}{2880}e^{-\xi}$, $\xi \in [0,1]$ $b-a=1$

$\therefore$ 误差界 $\varepsilon = \max|-\frac{(b-a)^5}{2880}e^{-\xi}| = \frac{1}{2880} = 0.3472\times10^{-3}$

(2) 梯形公式

$\int_1^{1.5} x^2\ln x\,dx \approx \frac{1.5-1}{2}(f(1)+f(1.5)) = \frac{1}{4}(0+1.5^2\ln 1.5) = 0.228$

余项为 $-\frac{h^3}{12}f''(\xi) = -\frac{h^3}{12}(2\ln\xi+3)$, $\xi\in[1,1.5]$ $h=(b-a)=0.5$

$\therefore$ 误差界 $\varepsilon = \max|-\frac{h^3}{12}(2\ln x+3)| = \frac{0.5^3}{12}(2\ln 1.5+3) = 0.397\times10^{-1}$

Simpson公式

$\int_1^{1.5} x^2\ln x\,dx \approx \frac{1.5-1}{6}(f(1)+4f(\frac{1+1.5}{2})+f(1.5)) = \frac{1}{12}(0+4\times1.25^2\ln1.25+1.5^2\ln1.5) = 0.192$

余项为 $-\frac{(b-a)^5}{2880}f^{(4)}(\xi) = -\frac{(b-a)^5}{2880}(-\frac{2}{\xi^2})$, $\xi\in[1,1.5]$ $h=(b-a)=0.5$

$\therefore$ 误差界 $\varepsilon = \max|-\frac{(b-a)^5}{2880}(-\frac{2}{\xi^2})| = \frac{0.5^5}{2880}\cdot 2 = 2.1701\times10^{-5}$

$\int_0^2 f(x)dx = C_0 f \quad C_1 \quad C_2.$

P274 T3 $\quad \int_0^1 \cdots \quad C_0 f_0 \quad C_1 f_1$

解: $n+1$个等距节点的代数精度至多为 $n+1$

(1) 设求积公式对 $f(x)=1, x, x^2, x^3$ 准确成立, 得.

$\int_0^2 dx = 2 = C_0+C_1+C_2$ ① 三个未知数, 四个等式

$\int_0^2 x\,dx = 2 = C_1+2C_2$ ②

$\int_0^2 x^2 dx = \frac{8}{3} = C_1+4C_2$ ③

$\int_0^2 x^3 dx = 4 = C_1 + 8C_2$ ④

联立②③，得：$C_2 = \frac{1}{3}$ ；联立②④，得：$C_2 = \frac{1}{3}$ 成立

解得：$C_0 = \frac{1}{3}$ $C_1 = \frac{4}{3}$ $C_2 = \frac{1}{3}$ ，具有 3 次代数精度

(2) 设 $f(x) = 1, x, x^2, \cdots$

$\int_0^1 dx = 1 = C_0 + C_1$ $\quad C_0 = \frac{1}{4}$ 代入

$$\left.\begin{aligned} \int_0^1 x dx &= \frac{1}{2} = C_1 x_1 \\ \int_0^1 x^2 dx &= \frac{1}{3} = C_1 x_1^2 \end{aligned}\right\} \text{解得：} x_1 = \frac{2}{3},\ C_1 = \frac{3}{4}$$

$\int_0^1 x^3 dx = \frac{1}{4} \neq \frac{3}{4}\cdot(\frac{2}{3})^3 = \frac{2}{9}$ 故具有 2 次代数精度

$C_0 = \frac{1}{4}$ $C_1 = \frac{3}{4}$ $x_1 = \frac{2}{3}$

P274 T10

解：两个节点的 Gauss-Legendre 求积公式为，

$$\int_{-1}^1 f(x)dx \approx f(-\tfrac{\sqrt{3}}{3}) + f(\tfrac{\sqrt{3}}{3})$$

(1) $a=0$ $b=\frac{\pi}{2}$

变量代换 $x = \frac{a+b}{2} + \frac{b-a}{2}t$ $\quad t_0 = -\frac{\sqrt{3}}{3},\ t_1 = \frac{\sqrt{3}}{3}$

对应 $x_0 = \frac{\pi}{4} - \frac{\sqrt{3}}{12}\pi$, $x_1 = \frac{\pi}{4} + \frac{\sqrt{3}}{12}\pi$

则 $\int_0^{\frac{\pi}{2}} \sin x dx \approx \frac{\pi}{4}\left(\sin(\frac{\pi}{4} - \frac{\sqrt{3}}{12}\pi) + \sin(\frac{\pi}{4} + \frac{\sqrt{3}}{12}\pi)\right) = 0.998$

(2) $a=0, b=1$

变量代换 $x = \frac{a+b}{2} + \frac{b-a}{2}t$ $\quad t_0 = -\frac{\sqrt{3}}{3},\ t_1 = \frac{\sqrt{3}}{3}$

对应 $x_0 = \frac{1}{2} - \frac{\sqrt{3}}{6}$ $x_1 = \frac{1}{2} + \frac{\sqrt{3}}{6}$

则 $\int_0^1 x^2 e^x dx \approx \frac{1}{2}\left((\frac{1}{2} - \frac{\sqrt{3}}{6})^2 e^{\frac{1}{2} - \frac{\sqrt{3}}{6}} + (\frac{1}{2} + \frac{\sqrt{3}}{6})^2 e^{\frac{1}{2} + \frac{\sqrt{3}}{6}}\right) = 0.71194$

2 编程题目

2.1 1203 课堂任务 1

生成任意阶次 Cotes 系数表的函数：

按照 Cotes 系数公式，得到对应的 Cotes 系数的函数（输入：n 阶次， 第 k 个系数；输出：对应的 Cotes 系数 $C^{(n)}{}_k$）。

求解步骤为：根据公式，其中较难求的是多项式连乘加上积分，这个其实也很好处理，用上一次作业中提到的，多项式展开其实就是一维卷积；另外，展开后的多项式积分也很容易，n 次方积分就是 n+1 次方除以 n。

代码如下：

```
def cotesInteg(n, k):
    ori_papa = np.zeros((n+3,)) # 经过 padding
    temp_para = ori_papa.copy()
    inte_para = np.zeros((n+2,))
    ori_papa[1] = 1
    for j in range(0, n+1):
        if j != k:
            kenel = np.array([1, -j])
            for i in range(n+2):
                temp_para[i+1] = np.sum(ori_papa[i:i+2] * kenel)
            ori_papa = temp_para.copy()
    inte_para[0] = 0
    for i in range(1, n+2):
        inte_para[i] = ori_papa[i]/i
    return s(n, inte_para)
```

求解不同阶次的 Cotes 求积：

首先初始化，生成 Cotes 系数表，通过循环采用不同阶次（1 至 8）的 Cotes 求积公式计算对应的积分。

代码如下：

```
# 生成 cotes 系数表
m, n = 9, 9
table = np.zeros((m, n))
for i in range(1, m):
    for j in range(0, i+1):
        table[i, j] = integ.cotes(i, j)

a, b = 1, 2

```

```python
def func(x):
    f = np.log(x)
    # f = (10/x)**2 * np.sin(10/x)
    return f

for n in range(1, 9):
    x = np.linspace(a, b, n+1, endpoint=True)
    fx = x.copy()
    for i in range(len(x)):
        fx[i] = func(x[i])
    inteCotes = 0
    for k in range(n+1):
        inteCotes += table[n, k] * fx[k]
    print(n,"阶积分结果：",inteCotes)
```

结果：

第一个积分公式的积分结果（图 2. 1）：

```
1 阶积分结果：  0.34657359027997264
2 阶积分结果：  0.3858346021654338
3 阶积分结果：  0.3860837836516575
4 阶积分结果：  0.38628789352450854
5 阶积分结果：  0.3862906470527488
6 阶积分结果：  0.3862942047858907
7 阶积分结果：  0.386294263534237
8 阶积分结果：  0.3862943562924036
```

图 2. 1 第一个积分式不同阶次的 Cotes 求积公式积分结果

积分的精确结果为 0.38629，可以看出，积分的结果精度随阶数增加升高，并且 5 阶时已实现小数点后 5 位相同。

同理对第二个积分式子改动，得到结果（图 2. 2）：

```
1 阶积分结果：  -28.259766449332297
2 阶积分结果：  -25.401993394163075
3 阶积分结果：  -17.10729764832243
4 阶积分结果：  -5.985229952189645
5 阶积分结果：  -3.258585759206183
6 阶积分结果：  0.11850792974579034
7 阶积分结果：  0.06553113990822518
8 阶积分结果：  0.0013795734898784107
```

图 2. 2 第二个积分式不同阶次的 Cotes 求积公式积分结果

积分的精确结果为 1.426，Cotes 积分结果很不理想，没有接近精确解的结果。是因为积分原函数的特殊性，以及高阶的 cotes 系数表出现了负值的系数，导致

数值不稳定的原因。

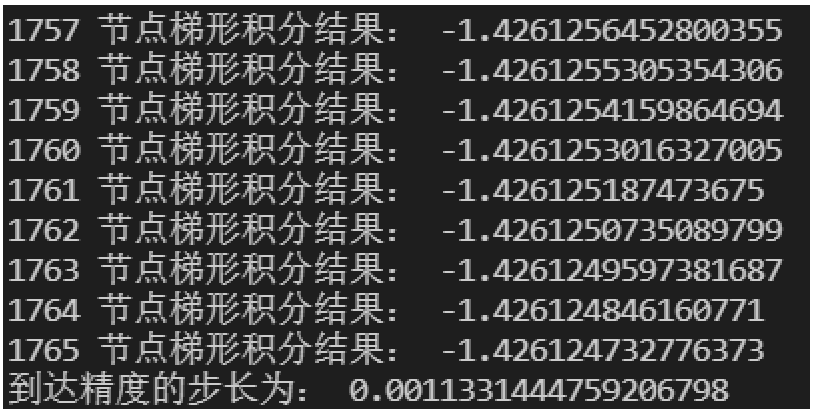
2.2 1203 课堂任务 2

复合梯形求积公式：复合梯形公式通过在相邻节点处使用梯形积分公式，随着节点数的增多，精度提高。当节点趋于无穷，误差理论为 0。

```
for n in range(2, 5000):
    x = np.linspace(a, b, n+1, endpoint=True)
    fx = x.copy()
    for i in range(len(x)):
        fx[i] = func(x[i])

    h = (b - a) / n
    # 复合梯形求积公式
    cs = func(a) + func(b)
    for k in range(1, n):
        cs += 2 * func(x[k])
    cs *= h/2
    print(n,"节点梯形积分结果：", cs)
    if np.abs(cs + 1.42602475634) < 1e-4:
        print("到达精度的步长为：", h)
        break
```

第二个积分式复合梯形公式积分结果如图 2. 3 所示。



```
1757 节点梯形积分结果：  -1.4261256452800355
1758 节点梯形积分结果：  -1.4261255305354306
1759 节点梯形积分结果：  -1.4261254159864694
1760 节点梯形积分结果：  -1.4261253016327005
1761 节点梯形积分结果：  -1.426125187473675
1762 节点梯形积分结果：  -1.4261250735089799
1763 节点梯形积分结果：  -1.4261249597381687
1764 节点梯形积分结果：  -1.426124846160771
1765 节点梯形积分结果：  -1.426124732776373
到达精度的步长为：  0.0011331444759206798
```

图 2. 3 第二个积分式复合梯形公式积分结果

复合梯形公式到达 10^{-4} 精度时，节点的个数为 1765，步长 h 为 0.001133144…。

同理采用复合 Simpson 公式：

```
sinp = func(a) + func(b)
for k in range(1, n):
    sinp += 4 * func((x[k-1] + x[k])/2) + 2 * func(x[k])
sinp += 4 * func((x[k-1] + x[k])/2)
sinp *= h/6
```

```python
print(n,"节点复合 Simpson 积分结果: ", sinp)
if np.abs(sinp + 1.42602475634) < 1e-4:
    print("到达精度的步长为: ", h)
    break
```

结果（图 2. 4）：

```
595 节点复合Simpson积分结果:  -1.4261272571281953
596 节点复合Simpson积分结果:  -1.4261269124926308
597 节点复合Simpson积分结果:  -1.426126569592267
598 节点复合Simpson积分结果:  -1.4261262284154594
599 节点复合Simpson积分结果:  -1.4261258889506958
600 节点复合Simpson积分结果:  -1.4261255511865107
601 节点复合Simpson积分结果:  -1.4261252151115853
602 节点复合Simpson积分结果:  -1.4261248807146405
603 节点复合Simpson积分结果:  -1.4261245479845532
到达精度的步长为:  0.0033167495854063
```

图 2. 4 第二个积分式复合 Simpson 公式积分结果

复合 Simpson 公式在 603 个节点的时候就达到了 10^{-4} 精度，步长 h 为 0.331675…。显然在相同节点个数的情况下，复合 Simpson 公式的精度要比复合梯形公式高。

2.3 计算实习题——Gauss 复合求积：

Gauss-Chebyshev 求积函数：

（输入：n 阶次，区间左 a，区间右 b，积分原函数 func；输出：积分结果）

根据公式，没有特别的地方，代码如下：

```python
def gaussCheb(n, a, b, func):
    t = np.zeros((n+1, ))
    A = np.ones((n+1,)) * np.pi/(n+1)
    for k in range(n+1):
        t[k] = np.cos((2*k+1)*np.pi/2/(n+1))
    I = 0
    for k in range(n+1):
        I += np.sqrt(1-t[k]**2) * func((a+b)/2 + (b-a)/2*t[k])
    return I * (b-a)/2*np.pi/(n+1)
```

复合 Gauss-Chebyshev 求积函数：

（输入：m 个区间，n 阶次，区间左 a，区间右 b，积分原函数 func；输出：积分结果）

复合 Gauss-Chebyshev 求积函数与原来相比并没有很明显的不同，思路只是

将原来的积分区间分割为若干等分，对每一等分分别进行 Gauss-Chebyshev 求积，最后累加起来，代码如下：

```
def combGaussCheb(m, n, a, b, func):
    x = np.linspace(a, b, m+1, endpoint=True)
    I_plus = 0
    for k in range(m):
        I_plus += gaussCheb(n, x[k], x[k+1], func)
    return I_plus
```

选取求积公式的阶次为 10，节点数增加的同时，积分结果的变化很小，1000 个节点后仍未达到 10^{-4} 的精度；选取更高的阶次：100，在 18 个节点的时候就达到了精度要求（图 2.5）。可见复合积分公式阶次和区间数选取恰当，可以提高求解的精度。换不同的阶次，找到对应的节点个数，画出节点个数随阶次的变化曲线（图 2.6）：

```
9 节点复合 100 阶Chebyshev积分: -1.4262514719878716
10 节点复合 100 阶Chebyshev积分: -1.4262174472222238
11 节点复合 100 阶Chebyshev积分: -1.4261927500072242
12 节点复合 100 阶Chebyshev积分: -1.4261742723748885
13 节点复合 100 阶Chebyshev积分: -1.426160090979103
14 节点复合 100 阶Chebyshev积分: -1.426148969136511
15 节点复合 100 阶Chebyshev积分: -1.4261400842425223
16 节点复合 100 阶Chebyshev积分: -1.426132872554763
17 节点复合 100 阶Chebyshev积分: -1.426126937474433
18 节点复合 100 阶Chebyshev积分: -1.4261219934796205
```

图 2.5 正交多项式对应的线性关多项式系数对比

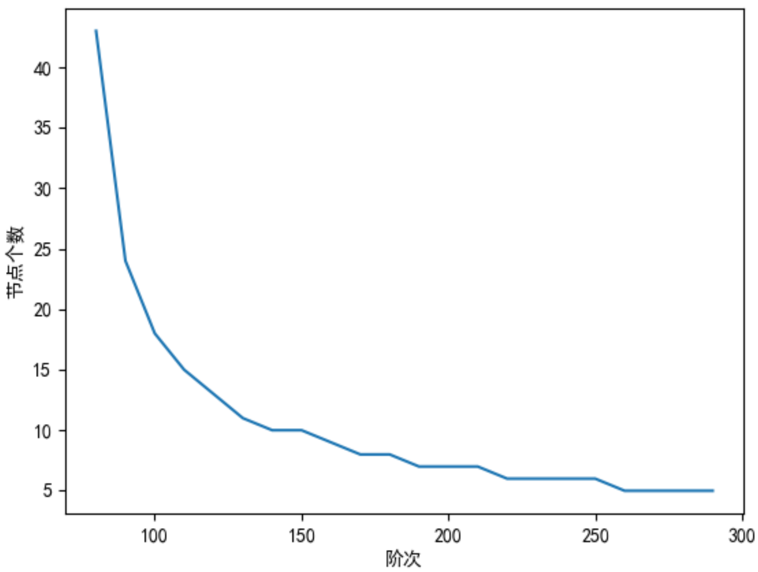


图 2.6 复合 Gauss-Chepshev 求积公式 10^{-4} 精度节点随阶次变化曲线

可以发现，在阶次为 140 左右，节点个数的变化趋于平缓，因此，节点个数为 10，阶次为 140 左右的复合 Gauss-Chepshev 求积是较合适的选择。

2.3 计算实习题——Romberg 求积：

Romberg 求积需要用到复合梯形公式，将上述复梯形公式的代码集成为函数（输入：n 段，左区间 a，右区间 b，积分原函数 func；输出：求积结果 cs）：

```python
def combTrap(n, a, b, func):
    h = (b - a) / n
    x = np.linspace(a, b, n+1, endpoint=True)
    cs = func(a) + func(b)
    for k in range(1, n):
        cs += 2 * func(x[k])
    return cs * h/2
```

Romberg 求积函数（输入：p 层，左区间 a，右区间 b，积分原函数 func；输出：求积结果表格 T（p, p））：

求解积分，得到在外推公式 7 层达到 10^{-4} 精度（图 2. 7）。

```
7 层达到精度要求，结果： -1.4260367474096967
[[-56.5195329    0.           0.           0.           0.
    0.           0.        ]
 [-52.23287332 -50.80398679   0.           0.           0.
    0.           0.        ]
 [-23.85638483 -14.39755533 -11.9704599    0.           0.
    0.           0.        ]
 [ -6.82780081  -1.15160614  -0.26854286  -0.08279815   0.
    0.           0.        ]
 [ -2.68153053  -1.29944044  -1.30929606  -1.32581595  -1.33069053
    0.           0.        ]
 [ -1.73265585  -1.41636429  -1.42415921  -1.42598244  -1.42637525
   -1.42646878   0.        ]
 [ -1.50221765  -1.42540492  -1.42600763  -1.42603697  -1.42603718
```

图 2. 7 Romberg 外推求积结果

参考文献

[1] 关治．数值方法[M]．北京：清华大学出版社，2006：66-92．

附录

见附件 python 源代码。