实验报告二

李平治 PB19071501

2022年4月4日

I. 题目及其运行结果

1.1 题目

用复化Simpson自动控制误差方式计算积分 $\int_a^b f(x)dx$

输入：积分区间[a，b]，精度控制值e，定义函数f(x)

输出：积分值S

利用 $\int_1^2 \ln x dx$，$\varepsilon = 10^{-4}$ 验证结果

1.2 结果

```
Initial n: 2
Simpson: 0.3858346
Auto precision-control Simpson: 0.38624123
3 iterations totally

Initial n: 4
Simpson: 0.38625956
Auto precision-control Simpson: 0.38627797
1 iterations totally

Initial n: 6
Simpson: 0.38628716
Auto precision-control Simpson: 0.38629096
```

```
1 iterations totally

Initial n: 8
Simpson: 0.38629204
Auto precision-control Simpson: 0.38629327
1 iterations totally
```

II. 使用算法

本次实验使用自动控制误差的复化Simpson积分

$$S_n = \frac{h}{3}[f(a) + 4\Sigma_{i=1}^{m-1} f(x_{2i+1}) + 2\Sigma_{i=1}^{m-1} f(x_{2i}) + f(b)]$$

$$S_{2n} = \frac{1}{2}S_n(f) + \frac{1}{6}(4H_{2n}(f) - H_n(f))$$

III. 结果分析

Initial n	Result
2	0.38624123
4	0.38627797
6	0.38629096
8	0.38629327

该积分的准确数值(保留到小数点后10位)应当为:

$$\int_1^2 \ln(x)dx \approx 0.3862943611$$

实验结果表明，在n稍微大一些时($n > 2$)，数值就会在一开始就收敛到较为准确的结果，而非假收敛.

这是因为实验给定的精度较小($\epsilon = 10^{-4}$)，而第一次计算出的S_n在这个精度下就已经很接近准确数值.

附录

本次实验的Python代码如下

```python
from math import log


def sn_calc(a: float, b: float, m: int, f):
    n = 2 * m
    h = (b - a) / n
    sn = f(a) + f(b)
    for i in range(m):
        sn = sn + 4 * f(a + (2 * i + 1) * h)
    for i in range(1, m):
        sn = sn + 2 * f(a + (2 * i) * h)
    sn = sn * h / 3
    return sn


def s2n_from_sn(a: float, b: float, sn: float, n: int, f):
    h = (b - a) / n
    Hn = 0
    H2n = 0
    for i in range(n):
        Hn = Hn + f(a + h * (i + 1 / 2))
    for i in range(2 * n):
        H2n = H2n + f(a + (h / 2) * (i + 1 / 2))
    Hn = Hn * h
    H2n = H2n * h / 2
    s2n = 0.5 * sn + 1 / 6 * (4 * H2n - Hn)
    return s2n


def auto_precision_simpson(a: float, b: float, eps: float, f, initial_m: int):
    m = initial_m
    s2 = sn_calc(a=a, b=b, m=m, f=f)
    s1 = s2 + 1
    iter_times = 0
    while abs(s1 - s2) > eps:
        iter_times += 1
        s1 = s2
        s2 = s2n_from_sn(a=a, b=b, sn=s1, n=2 * m, f=f)
        m = 2 * m
    return s2, iter_times


start = 1
end = 2
e = 10 ** (-4)
for i in range(1, 5):
    integral, i_time = auto_precision_simpson(a=start, b=end, eps=e, f=log,
initial_m=i)
    print("Initial n:", 2 * i)
    print("Simpson:", round(sn_calc(a=start, b=end, m=i, f=log), 8))
```

```
50      print("Auto precision-control Simpson:", round(integral, 8), ".")
51      print(i_time, "iterations totally")
52
```