哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

实验成绩

实验报告

课程名称: 数值逼近

实验项目: 数值积分

所在院系: 信息与计算科学

学生姓名: 葛煜龙

学生学号: 1201200206

授课学期: 22 秋

完成时间: 2022.11.09

1 习题一 Simpson 求积公式

计算代码如下：

```
import math
from scipy import integrate
import numpy as np


def gauss(x):
    return (200 / (0.1 * math.pi)) * math.exp(-math.pow(x - 1.7, 2) /
                                        (2 * 0.01))


def simpson(function, a, b):
    return ((b - a) / 6) * (function(a) + 4 * function((a + b) / 2) +
                                        function(b))


if __name__ == "__main__":
    n = 10
    h = 0.01
    x = np.arange(1.8, 1.905, 0.01)
    compound_simpson = 0
    '''
        for i in range(n):
        compound_simpson += 4 * gauss((x[i] + x[i + 1]) / 2)
    for i in range(n):
        compound_simpson += 2 * gauss(x[i])
    compound_simpson += gauss(1.8)
    compound_simpson += gauss(1.9)
    compound_simpson *= 0.01/6
    '''
    for i in range(n):
        compound_simpson += simpson(gauss, x[i], x[i + 1])

    print("复合simpson方法求得的积分为" + str(compound_simpson),
          "误差为" + str(compound_simpson - integrate.quad(gauss, 1.8
                                            , 1.9)[0]))
    print("simpson方法求得的积分为" + str(simpson(gauss, 1.8, 1.9)),
          "误差为" + str(simpson(gauss, 1.8, 1.9) - integrate.quad(
                                            gauss, 1.8, 1.9)[0]))
    print("标准数值积分为" + str(integrate.quad(gauss, 1.8, 1.9)[0]))
```

最终计算结果如下：

1. 复合 simpson 方法求得的积分为 21.687316429129037 ,误差为-3.2837683896502767e-06。

2. simpson 方法求得的积分为 21.650120786676524 ,误差为-0.037198926220902706。

3. 标准数值积分为 21.687319712897427。

2 习题二 Romberg 求积方法

计算代码如下：

```
import math

import numpy as np
from scipy import integrate


def trapezium_integral(f, a, b, n):
    h = (b - a) / n
    x = np.arange(a, b + h, h)
    integral_sum = 0
    for i in range(n):
        integral_sum += (h / 2) * (f(x[i]) + f(x[i + 1]))
    return integral_sum


def romberg(f, a, b, tolenrance):
    t = []
    k = 1
    # t.append(((b - a) / 2) * (f(a) + f(b)))
    t.append(trapezium_integral(f, a, b, 1))
    t.append(
        [trapezium_integral(f, a, b, 2), (4 * trapezium_integral(f, a
                                          , b, 2) -
                                          trapezium_integral(f, a, b
                                          , 1)) / 3])
    while abs(t[k][k] - t[k][k - 1]) > tolenrance:
        k += 1
        tk = [trapezium_integral(f, a, b, 2 ** k)]
        for j in range(1, k + 1):
            tk.append((1 / (4 ** j - 1)) * (4 ** j * tk[j - 1] - t[k
                                            - 1][j - 1]))
        t.append(tk)

    return t[k][k]


def f(x):
    return math.log(x)


if __name__ == "__main__":
```

```
    print(trapezium_integral(math.cos, 0, 1, 1))
    epsilon = 10 ** (-6)
    solution = romberg(f, 1, 2, epsilon)
    solution2 = integrate.quad(f, 1, 2)
```

最终计算结果如下:

1. romberg 方法求得的积分为 0.38629430908624807, 误差为-5.203364256134435e-08

3 习题三 复合积分方法和 Gauss 积分方法的比较

计算代码如下:

```
import math
from scipy import integrate
import numpy as np


def f(x):
    return math.sqrt(1 + math.exp(x))


def simpson(function, a, b):
    return ((b - a) / 6) * (function(a) + 4 * function((a + b) / 2) +
                                        function(b))


# 复合梯形求积
def trapezium_integral(f, a, b, n):
    h = (b - a) / n
    x = np.arange(a, b + h, h)
    integral_sum = 0
    for i in range(n):
        integral_sum += (h / 2) * (f(x[i]) + f(x[i + 1]))
    return integral_sum


def func1(x):
    return 2 * (1 + np.exp(2 * x + 2)) ** (1 / 2)


def Guass_Integral(n):
    weight = np.array([0.417959184,
                       0.381830051,
```

```
                    0.381830051,
                    0.279705391,
                    0.279705391,
                    0.129484966,
                    0.129484966, ])
    xxxx = np.array([
        0,
        - 0.405845151,
        0.405845151,
        - 0.741531186,
        0.741531186,
        - 0.949107912,
        0.949107912,
    ])
    fxxxx = func1(xxxx)

    if n == 3:
        return 0.88888888 * func1(0) + 0.55555555 * func1(-0.
                                          775666924) + 0.55555555 *
                                          func1(0.774596669)
    elif n == 5:
        return 0.56888888 * func1(0) + 0.47862867 * func1(-0.
                                          538469310) + 0.47862867 *
                                          func1(
            0.538469130) + 0.23692688 * func1(-0.90617984) + 0.
                                              23692688 * func1(0.
                                              90617984)
    elif n == 7:
        return np.matmul(weight, np.transpose(fxxxx))


if __name__ == "__main__":
    compound_simpson = 0
    # 区间等分数
    n = 2
    x = np.linspace(0, 4, n + 1)
    for i in range(n):
        compound_simpson += simpson(f, x[i], x[i + 1])
    print('7点的高斯积分为' + str(Guass_Integral(7)))
    print('5点的高斯积分为' + str(Guass_Integral(5)))
    print('3点的高斯积分为' + str(Guass_Integral(3)))
    print("复合梯形积分为" + str(trapezium_integral(f, 0, 4, n)))
```

最终计算结果如下:

1. 7 点的高斯积分为 13.57730239773161
2. 5 点的高斯积分为 13.577301646358315
3. 3 点的高斯积分为 13.575820922114811
4. 复合梯形积分为 14.663403727504809
5. 复合 simpson 积分为 13.581379562032348
6. 标准数值积分为 13.577302400789666