#Exercise 4.2

#15.10.21

#author: chuanlu

import numpy as np

import scipy as sp

import math

e = math.e

def \_composite\_simpson\_formula(a, b, n, func):

x\_series1 = [a + (b - a)/n \* i for i in range(n + 1)]

x\_series2 = [x + (b - a)/(2 \* n) for x in x\_series1[:-1]]

f\_series1 = [func(x) for x in x\_series1]

f\_series2 = [func(x) for x in x\_series2]

return (b - a)/n \*(sum(f\_series1) \* 2 + sum(f\_series2) \* 4 - f\_series1[0] - f\_series1[-1]) / 6

def simpson\_double\_integrate(x\_range, func, func1, func2, n):

x\_start = x\_range[0]

x\_end = x\_range[1]

k = lambda x: func2(x) - func1(x)

def funcAUX(i, n, func1, func2):

def listfunc(x):

return func1(x) + k(x) \* i / n

return listfunc

funclist1 = [funcAUX(i,n,func1,func2) for i in range(n+1)]

def funcAUX2(n, func):

def listfunc2(x):

return func(x) + k(x) / (2 \* n)

return listfunc2

funclist2 = [funcAUX2(n, f) for f in funclist1[:-1]]

f\_values1 = \_composite\_simpson\_formula(x\_start, x\_end, n, lambda x: func(x, funclist1[0](x)) \* k(x))

f\_values2 = [\_composite\_simpson\_formula(x\_start, x\_end, n, lambda x: func(x, funclist2[i](x)) \* k(x)) for i in range(n)]

f\_values3 = [\_composite\_simpson\_formula(x\_start, x\_end, n, lambda x: func(x, funclist1[i](x)) \* k(x)) for i in range(1, n)]

f\_values4 = \_composite\_simpson\_formula(x\_start, x\_end, n, lambda x: func(x, funclist1[-1](x)) \* k(x))

return (f\_values1 + 4 \* sum(f\_values2) + 2 \* sum(f\_values3) + f\_values4) / (6 \* n)

def q1(func):

x\_range = [0, 1]

func1 = lambda x: 0

func2 = lambda x: 1

n = 4

result = simpson\_double\_integrate(x\_range, func, func1, func2, n)

print(result)

def q2(func):

x\_range = [0, 1]

func1 = lambda x: 0

func2 = lambda x: math.sqrt(1 - x \*\* 2)

n = 4

result = simpson\_double\_integrate(x\_range, func, func1, func2, n)

print(result)

def db\_gauss\_legendre\_formula(func, n = 4):

x\_points = [-0.9061798, -0.5384693, 0, 0.5384693, 0.9061798]

A\_list = [0.2369269, 0.4786287, 0.5688889, 0.4786287, 0.2369269]

print(sum([A\_list[i] \* A\_list[j] \* func(x\_points[i], x\_points[j]) for i in range(len(x\_points)) for j in range(len(x\_points))])/4)

def main():

func0 = lambda x, y: e \*\* (- x \* y)

q1(func0)

func1 = lambda x, y: e \*\* (-(x + 1) \* (y + 1) / 4)

db\_gauss\_legendre\_formula(func1)

q2(func0)

if \_\_name\_\_ == '\_\_main\_\_':

main()

运行结果如下：

第一小题：

0.7965999679462029

0.7965996777384788

分别为复合辛普森公式和高斯求积公式的结果

第二小题：

0.6701136333590952

采用复合辛普森公式的结果