Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 5

Тема: Построение и программная реализация алгоритмов численного интегрирования.

Студент Тартыков Л.Е.

Группа ИУ7-44Б

Оценка (баллы) _______________

Преподаватель Градов В.М.

2021 г.

Содержание

Цель работы: получение навыков построения алгоритма вычисления двукратного интеграла с использованием квадратурных формул Гаусса и Симпсона.

Задание

Построить алгоритм и программу для вычисления двукратного интеграла при фиксированном значении параметра τ

$$\varepsilon(\tau) = \frac{4}{\pi} \int_0^{\pi/2} d\varphi \int_0^{\pi/2} [1 - \exp(-\tau \frac{l}{R})] \ \cos\theta \ \sin\theta \ d\theta \ ,$$

где $$\frac{l}{R} = \frac{2\cos\theta}{1 - \sin^2\theta\cos^2\varphi},$$

θ, φ - углы сферических координат.

Применить метод последовательного интегрирования. По одному направлению использовать формулу Гаусса, а по другому - формулу Симпсона.

Код программы

Замечание: данная программа написана на языке Python 3.8

Листинг 1; main.py

```
from math import pi, cos, sin, pow, exp, fabs
import numpy as np
import matplotlib.pyplot as plt

EPS = 1e-6
A = C = 0
B = D = pi / 2

def convert_to_radian(degree):
    """
    Перевести градусы в радианы
    """
    return degree * pi / 180

def calculate_ratio_l_R(theta, phi):
    """
    Вычислить отношение l/R
    """
    theta_radian = convert_to_radian(theta)
    phi_radian = convert_to_radian(phi)
    return 2 * cos(theta) / (1 - pow(sin(theta_radian), 2) * pow(cos(phi_radian), 2))

def func(tau, theta, phi):
    """
    Подынтегральное выражение
    """
    return (1 - exp(-tau * calculate_ratio_l_R(theta, phi))) * cos(theta) * sin(theta)

def init_phi(n):
    phi = [0 for i in range(n)]
    part = pi / 2 / n
    for i in range(n):
        phi[i] = part * i

    return phi

def integrate_by_gauss_quadrature_formula(n, m, tau, phi):
    """
    Интегрирование квадратурной формулой Гаусса
    """
    x = []
    len_x = 0
    step = 2.0 / m
    while (len_x < m):
        step /= 2.0
        len_x = 0
```

```
        a = -1; b = a + step
        while (a < 1):
            if calculate_legendre_polynomial(m, a) * calculate_legendre_polynomial(m, b) < 0:
                len_x += 1
            a = b; b += step

    a = -1
    b = a + step
    i = 0
    while (a < 1 and i < m):
        if calculate_legendre_polynomial(m, a) * calculate_legendre_polynomial(m, b) < 0:
            x.append(calculate_value_by_bisection(a, b, m))
            i += 1
        a = b
        b += step

    right_slau = []
    for i in range(0, m):
        if i % 2 == 0:
            right_slau.append(2.0 / (i + 1))
        else:
            right_slau.append(0)

    help_slau = [1 for i in range(m)]
    left_slau = [[] for i in range(m)]

    for i in range(m):
        for j in range(m):
            left_slau[i].append(help_slau[j])
            help_slau[j] *= x[j]

    r_slau = np.asarray(right_slau)
    l_slau = np.asarray(left_slau)

    weights = np.linalg.solve(l_slau, r_slau)

    for i in range(m):
        x[i] = pi / 4 * (1 + x[i])

    integrals = [0 for i in range(n)]
    for i in range(n):
        for j in range(m):
            integrals[i] += weights[j] * func(tau, x[j], phi[i])
        integrals[i] *= pi/ 4
    return integrals

def calculate_legendre_polynomial(count_nodes, x):
    """
    Вычисление полинома Лежандра
    """
    if count_nodes == 0:
```

```
        return 1
    elif count_nodes == 1:
        return x
    else:
        p_before_last = 1 #p_0
        p_last = x #p_1
        p_current = 0
        for i in range(2, count_nodes + 1):
            p_current = ((2 * i - 1) * p_last * x - (i - 1) * p_before_last) / i
            p_before_last = p_last
            p_last = p_current
        return p_current

def function(count_nodes, x):
    """
    Вычисляемая функция (в данном случае полином Лежандра)
    """
    return calculate_legendre_polynomial(count_nodes, x)

def calculate_value_by_bisection(left, right, count_nodes):
    """
    Метод половинного деления
    """
    middle = (left + right) / 2
    while fabs(left - right) > EPS:
        d = function(count_nodes, middle) * function(count_nodes, left)
        if d > 0:
            left = middle
        else:
            right = middle
        middle = (left + right) / 2

    return middle

def calculate_simpson_integral(integrals, left, right, n):
    """
    Вычисление интеграла Симпсона
    """
    result = 0
    h = (right - left) / (n - 1)

    for i in range(0, int(n / 2 - 1)):
        result += integrals[2 * i] + 4 * integrals[2 * i + 1] + integrals[2 * i + 2]
    return result * (h / 3) * (4 / pi) # 4/pi - коэф заданной функции

def create_graph():
    """
    Построение графиков интеграла функции
    """
    tau = np.linspace(0, 10, 200)
    count_nodes_n = 5 #количество узлов по внешнему направлению
```

```
count_nodes_m = 2 #количество узлов по внутреннему направлению
phi = init_phi(count_nodes_n)

result_1 = []
for i in range(len(tau)):
    current_tau = i
    integrals = integrate_by_gauss_quadrature_formula(count_nodes_n, count_nodes_m, current_tau, phi)
    result_1.append(calculate_simpson_integral(integrals, C, D, count_nodes_n))
plt.plot(tau, result_1, color = "blue", label = "N = 5, M = 2, Simpson - Gauss")

count_nodes_n = 5
count_nodes_m = 3
phi = init_phi(count_nodes_n)

result_2 = []
for i in range(len(tau)):
    current_tau = i
    integrals = integrate_by_gauss_quadrature_formula(count_nodes_n, count_nodes_m, current_tau, phi)
    result_2.append(calculate_simpson_integral(integrals, C, D, count_nodes_n))
plt.plot(tau, result_1, color = "orange", label = "N = 5, M = 3, Simpson - Gauss")

count_nodes_n = 5
count_nodes_m = 4
phi = init_phi(count_nodes_n)

result_3 = []
for i in range(len(tau)):
    current_tau = i
    integrals = integrate_by_gauss_quadrature_formula(count_nodes_n, count_nodes_m, current_tau, phi)
    result_3.append(calculate_simpson_integral(integrals, C, D, count_nodes_n))
plt.plot(tau, result_1, color = "green", label = "N = 5, M = 4, Simpson - Gauss")

count_nodes_n = 5
count_nodes_m = 5
phi = init_phi(count_nodes_n)

result_4 = []
for i in range(len(tau)):
    current_tau = i
    integrals = integrate_by_gauss_quadrature_formula(count_nodes_n, count_nodes_m, current_tau, phi)
    result_4.append(calculate_simpson_integral(integrals, C, D, count_nodes_n))
plt.plot(tau, result_1, color = "red", label = "N = 5, M = 5, Simpson - Gauss")

plt.xlabel("Tau value")
plt.ylabel("Result")
plt.title("Численное интегрирование")
plt.legend()
plt.show()
```

```
def main():
    create_graph()

if __name__ == "__main__":
    main()
```

Результаты работы

1. Описать алгоритм вычисления n корней полинома Лежандра n-ой степени Pn(x) при реализации формулы Гаусса.

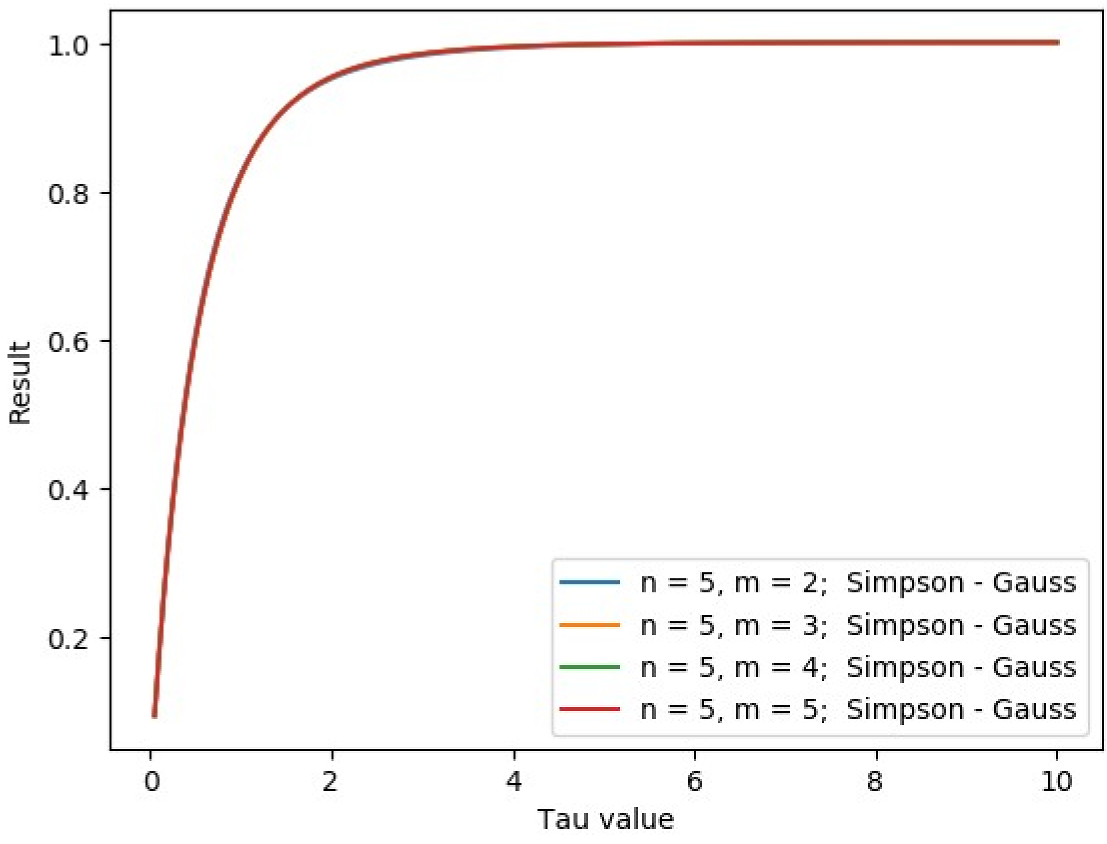
Все корни полинома лежат на отрезке [-1; 1], причем два отрезка [-1; 0] и [0; 1] симметричны. Таким образом, можно рассматривать для нахождения корней только отрезок [0; 1] и отображать полученные корни относительно оси Ox (для отрезка [1; 0]).

Делим первоначальный отрезок на более мелкие отрезки. Для каждого из них проверяем знак произведения концов отрезка. Если это произведение отрицательно, то методом половинного деления ищем корень на этом отрезке.

2. Исследовать влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов.

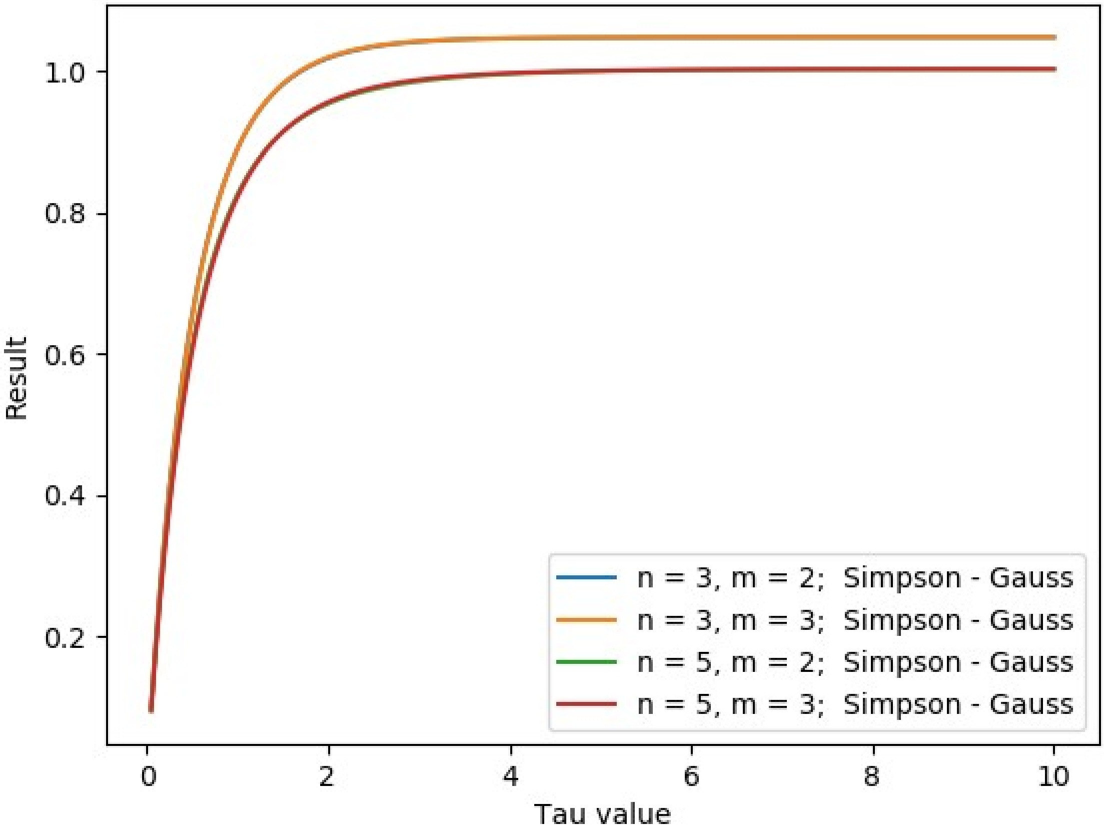
Замечание: в качестве внутреннего направления интегрирования использовался метод Гаусса, для внешнего — метод Симпсона

При задании большего числа узлов при интегрировании методом Симсона, метод Гаусса выдает одинаковый результат



При задании меньшей степени полинома для внешнего интегрирования будет происходит расхождение с физическим смыслом (степень черноты ε не превышает единицу)

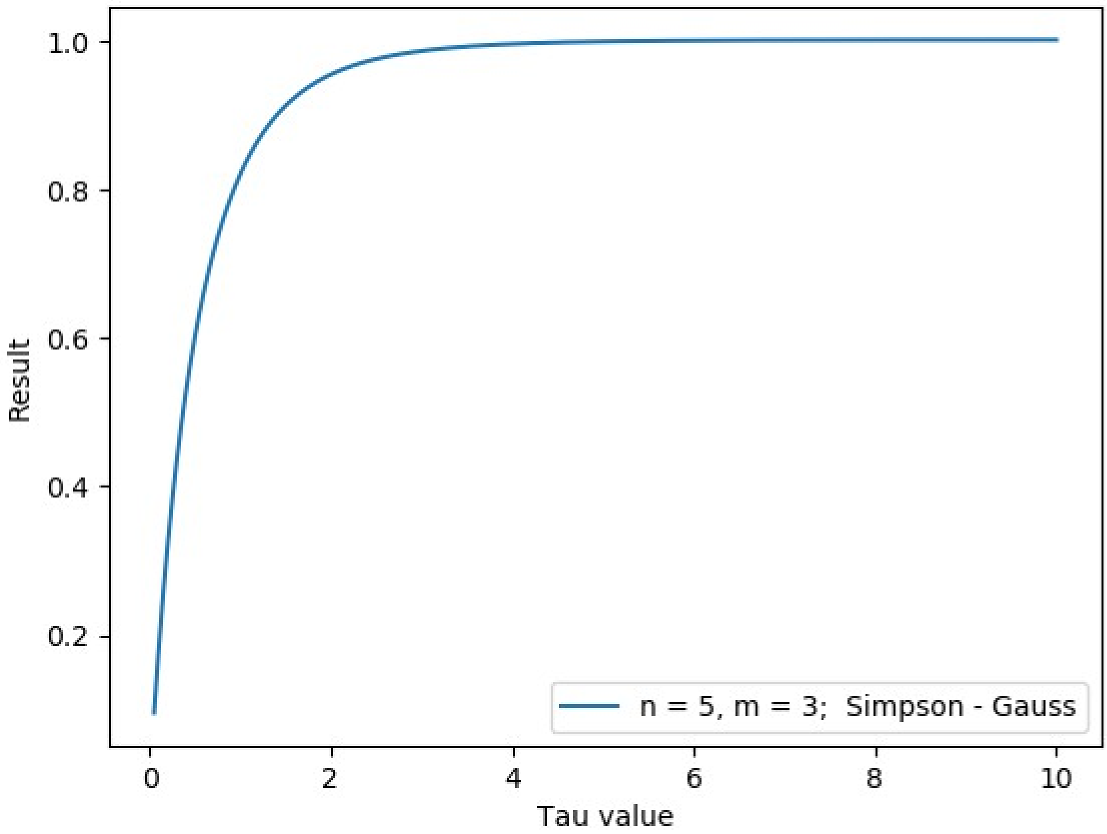
3.



Построить график зависимости ε(τ) в диапазоне изменения τ =0.05-10. Указать при каком количестве узлов получены результаты.

Число узлов для внешнего направления — 5

Число узлов для внутреннего направления - 3


1.0
0.8
0.6
0.4
0.2
Result
0
2
4
6
8
10
Tau value
n = 5, m = 3; Simpson - Gauss

Ответы на контрольные вопросы

1. В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается.

Такой случай возникает, когда подынтегральная функция не имеет производных. Либо же это может возникать, когда значения производных интегрируемой функции является очень большими (когда исходная функция имеет резкие скачки).

2. Построить формулу Гаусса численного интегрирования при одном узле.

$$P(x)=x \Rightarrow x=0$$

$$\sum_{(i=1)}^{n} A_1=2$$

$$\int_a^b f(x)dx=\frac{b-a}{2}*2f*(\frac{b+a}{2}+0*(\frac{b-a}{2}))=(b-a)*f(\frac{a+b}{2})$$

3. Построить формулу Гаусса численного интегрирования при двух узлах.

$$P(x)=\frac{1}{2}*(3x^2-1)\Rightarrow x=\pm\frac{\sqrt{(3)}}{3}$$

$$\begin{cases} A_1+A_2=2 \\ -\frac{\sqrt{(3)}}{3}A_1+\frac{\sqrt{(3)}}{3}A_2=0 \end{cases} \Rightarrow \begin{cases} A_1=1 \\ A_2=1 \end{cases}$$

$$\textit{На промежутке}[-1;1]$$

$$\int_{-1}^{1} f(f)df=f(-\frac{\sqrt{(3)}}{3})+f(\frac{\sqrt{(3)}}{3})$$

$$\textit{Перейдем к}[a;b]$$

$$\int_a^b f(x)dx=\frac{b-a}{2}(f(\frac{b+a}{2}-\frac{\sqrt{(3)}b-a}{6})+f(\frac{b+a}{2}+\frac{\sqrt{(3)}b-a}{6}))$$

4. Получить обобщенную кубатурную формулу, аналогичную (6.6) из лекции No6, для вычисления двойного интеграла методом последовательного интегрирования на основе формулы трапеций с тремя узлами по каждому направлению.

$$\int_c^d \int_a^b f(x,y)dx\,dy = h_x(\frac{1}{2}(F_0+F_2)+F_1) =$$

$$= h_x h_y[\frac{1}{4}(f(x_0,y_0)+f(x_0,y_2)+f(x_2,y_0)+f(x_2,y_2))+$$

$$+\frac{1}{2}(f(x_0,y_1)+f(x_2,y_1)+f(x_1,y_0)+f(x_1,y_2))+f(x_1,y_1)]$$