



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: **Ивахненко Д. А**

Группа: **ИУ7-46Б**

Оценка (баллы): _____

Преподаватель: **Градов В.М.**

Москва

2021 г.

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

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

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

θ, φ - углы сферических координат.

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

Исходные данные

- количество узлов сетки N, M
- значение параметра τ
- методы для направлений при последовательном интегрировании.

Выходные данные

Значение интеграла при заданном параметре, график зависимости $\varepsilon(\tau)$ в диапазоне $\tau \in [0.05; 10]$.

Описание алгоритма

1. Квадратурная формула Гаусса.

Пусть интеграл вычисляется на стандартном интервале $[-1; 1]$. Задача состоит в том, чтобы подобрать точки t_1, t_2, \dots, t_n и коэффициенты A_1, A_2, \dots, A_n , так чтобы квадратурная формула была точной для всех полиномов наивысшей возможной степени:

$$\int_{-1}^1 f(t) dt = \sum_{i=1}^n A_i f(t_i)$$

Получаем систему нелинейных уравнений:

$$\left\{ \begin{array}{l} \sum_{i=1}^n A_i = 2 \\ \sum_{i=1}^n A_i t_i = 0 \\ \sum_{i=1}^n A_i t_i^2 = \frac{2}{3} \\ \dots \\ \sum_{i=1}^n A_i t_i^{2n-1} = 0 \end{array} \right.$$

Для решения данной систему будем использовать полиномы Лежандра:

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n, n = 0, 1, 2, \dots$$

Можно показать, что узлами формулы Гаусса являются нули многочлена Лежандра степени n . Таким образом, нужно найти все корни многочлена Лежандра, после чего решить систему линейных алгебраических уравнений (решаем методом Гаусса).

Для вычисления интеграла на произвольном интервале $[a; b]$, для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной:

$$x = \frac{b+a}{2} + \frac{b-a}{2}$$

2. Формула Симпсона.

Суть метода заключается в приближении подынтегральной функции на отрезке $[a; b]$ интерполяционным многочленом второй степени $P_2(x)$, то есть приближение графика функции на отрезке параболой.

$$\int_a^b f(x)dx \approx \frac{h}{3} \sum_{i=0}^{\frac{N}{2}-1} (f_{2i} + 4f_{2i+1} + f_{2i+2}),$$

$$R \approx -\frac{1}{180} h^4 \int_a^b f^{(4)}(x)dx = O(h^4).$$

Следует отметить, что **если подынтегральная функция не имеет соответствующих производных, то указанный порядок точности не достигается**. Так, если на отрезке интегрирования не существуют 3-я и 4-я производные, то порядок точности формулы Симпсона будет только второй, $O(h^2)$.

Исходный код программы

main.py

```
from math import pi

from plot import plot
import solve

def main():
    all_n, all_m = [], []
    methods1, methods2 = [], []
    values = []

    go_on = 'y'
    while go_on == 'y':
        all_n.append(int(input("Введите N: ")))
        all_m.append(int(input("Введите M: ")))

        p = float(input("Введите параметр (tau): "))

        methods1.append(solve.Method(
            int(input('Выберите "внешний" метод интегрирования: (0 - Гаусс, 1 - Симпсон): '))))
        methods2.append(solve.Method(
            int(input('Выберите "внутренний" метод интегрирования: (0 - Гаусс, 1 - Симпсон): '))))

        lm = [[0, pi / 2], [0, pi / 2]]

        values.append(solve.Calculator(
            lm, [all_n[-1], all_m[-1]], [methods1[-1], methods2[-1]]))

        print(f'Результат (tau = {p:.2f}): {values[-1](p):.7f}')

        go_on = input("Продолжить работу? [y/n]: ")

    plot(values, all_n, all_m, methods1, methods2)
```

solve.py

```
from enum import Enum
from math import cos, sin, exp, pi
from scipy.special import roots_legendre
from typing import List, Callable

class Method(Enum):
    GAUSS = 0
    SIMPSON = 1

    def get_func(self) → Callable:
        interp = {
            Method.GAUSS: Calculator.gauss,
            Method.SIMPSON: Calculator.simpson
        }
        return interp[self]

    def __str__(self) → str:
        interp = {
            Method.GAUSS: 'Гаусс',
            Method.SIMPSON: 'Симпсон'
        }
        return interp[self]

class Calculator(object):
    def __init__(self, boundaries: List[List[float]], n: int, m: int, methods:
List[Method]):
        self.boundaries = boundaries
        self.n = n
        self.m = m
        self.f1 = methods[0].get_func()
        self.f2 = methods[1].get_func()
```

```

def __call__(self, p: float) → float:
    f = Calculator.__integrated(p)

    def inner(x): return self.f2(
        lambda val1: f(x, val1),
        self.boundaries[1][0],
        self.boundaries[1][1],
        self.n[1])

    def integ(): return self.f1(
        inner,
        self.boundaries[0][0],
        self.boundaries[0][1],
        self.n[0])

    return integ()

@staticmethod
def __integrated(p: float) → Callable[[float, float], float]:
    def t(x, y): return 2 * cos(x) / (1 - sin(x) ** 2 * cos(y) ** 2)
    return lambda x, y: 4 / pi * (1 - exp(-p * t(x, y))) * cos(x) * sin(x)

@staticmethod
def simpson(f: Callable[[float], float], a: float, b: float, n: int) → float:
    if n < 3 or n % 2 == 0:
        raise Exception('Некорректное значение N')

    h = (b - a) / (n - 1.0)
    x = a
    res = 0.0

    for _ in range((n - 1) // 2):
        res += f(x) + 4 * f(x + h) + f(x + 2 * h)
        x += 2 * h

    return res * h / 3

@staticmethod
def gauss(f: Callable[[float], float], a: float, b: float, n: int) → float:
    def p2v(p: float, c: float, d: float) → float:
        return (d + c) / 2 + (d - c) * p / 2

    x, w = roots_legendre(n)
    return sum([(b - a) / 2 * w[i] * f(p2v(x[i], a, b)) for i in range(n)])

```

plot.py

```
import matplotlib.pyplot as plt
import settings

def get_label(n, m, md1, md2) → str:
    return f'N = {n}, M = {m}, Методы: {md1}-{md2}'

def plot(values, all_n, all_m, methods1, methods2):

    plt.clf()

    plt.xlabel("Tao")
    plt.ylabel("Eps(Tao)")
    plt.grid(which='minor', color='k', linestyle=':')
    plt.grid(which='major', color='k')

    for i in range(len(values)):
        x, y = [], []
        j = settings.TAU_START
        while j < settings.TAU_END:
            x.append(j)
            y.append(values[i](j))
            j += settings.TAU_STEP

        plt.plot(x, y, label=get_label(all_n[i], all_m[i], methods1[i],
methods2[i]))

    plt.legend()
    plt.savefig('points.png')
    plt.show()
```


Результаты

Алгоритм вычисления n корней полинома Лежандра n -ой степени $P_n(x)$ при реализации формулы Гаусса.

1. Разобьем отрезок $[-1; 1]$ на $2n$ равных частей, и проверим наличие отрезков со сменой знаков на концах.
2. Если таких отрезков найдется менее, чем n , то увеличим число разбиений исходного отрезка в два раза.
3. Повторять 1-2.
4. Находим корни полинома Лежандра на найденных отрезках методом половинного деления.

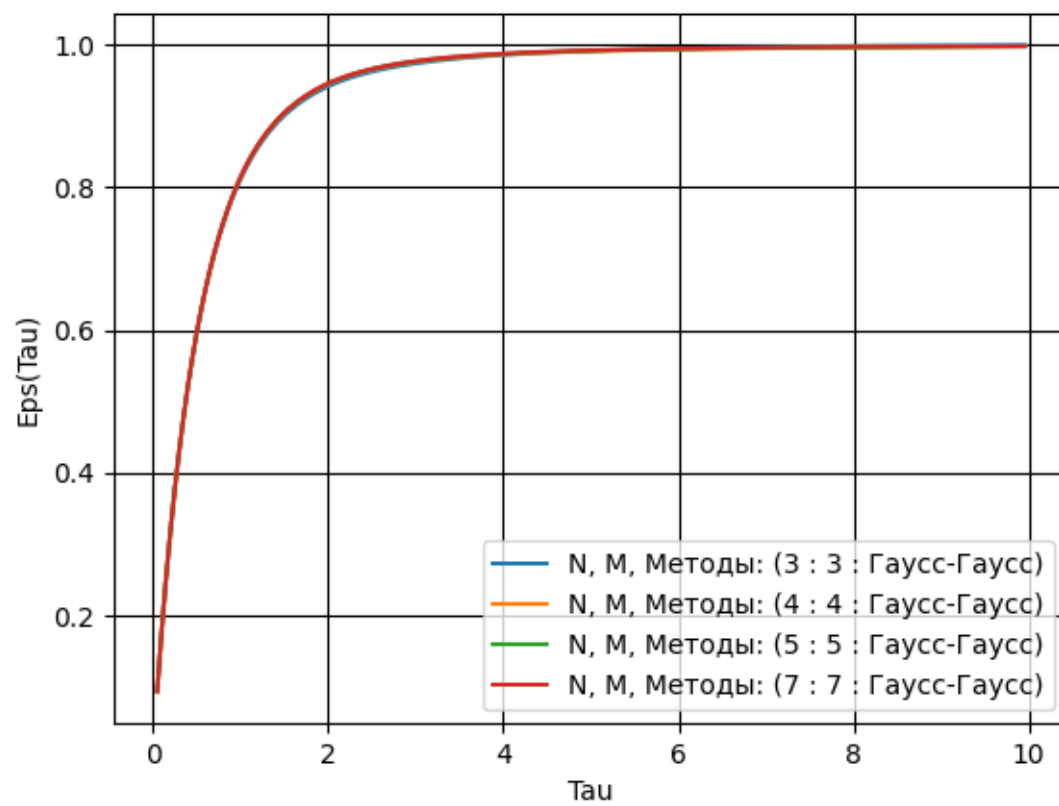
Влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов

Гаусс-Симпсон ($\tau = 0.75$).

$$\varepsilon(\tau) = 0.7299$$

		<i>Гаусс</i>			
		3	5	7	9
<i>Симпсон</i>	3	0.7279	0.7281	0.7280	0.7280
	5	0.7310	0.7300	0.7298	0.7298
	7	0.7295	0.7299	0.7299	0.7299
	9	0.7295	0.7299	0.7299	0.7299

График зависимости $\varepsilon(\tau)$ ($\tau = 1$)



Ответы на контрольные вопросы

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

В случае если подынтегральная функция не имеет соответствующих производных. Порядок точности равен номеру последней существующей производной.

- 2. Построить формулу Гаусса численного интегрирования при одном узле.**

$$\sum_{i=1}^n A_i = 2, P_1(x) = x \rightarrow x = 0$$
$$\int_a^b f(x)dx = \frac{b-a}{2} 2f\left(\frac{b+a}{2} + \frac{b-a}{2} * 0\right) = (b-a)f\left(\frac{b+a}{2}\right)$$

- 3. Построить формулу Гаусса численного интегрирования при двух узлах.**

$$P_2(x) = \frac{1}{2}(3x^2 - 1) \rightarrow x = \pm \frac{1}{\sqrt{3}}$$

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

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

$$\int_a^b f(x)dx = \frac{b-a}{2} \left(f\left(\frac{b+a}{2} - \frac{b-a}{2} * \frac{1}{\sqrt{3}}\right) + f\left(\frac{b+a}{2} + \frac{b-a}{2} * \frac{1}{\sqrt{3}}\right) \right)$$

- 4. Получить обобщенную кубатурную формулу, на основе метода трапеций, с тремя узлами на каждом направлении.**

$$\begin{aligned}
\int_c^d \int_a^b f(x, y) dx dy &= h_x \left(\frac{1}{2} F(x_0) + F(x_1) + \frac{1}{2} F(x_2) \right) = \\
&= h_x h_y \left[\frac{1}{2} \left(\frac{1}{2} f(x_0, y_0) + f(x_0, y_1) + \frac{1}{2} f(x_0, y_2) \right) + \right. \\
&\quad \left. + \left(\frac{1}{2} f(x_1, y_0) + f(x_1, y_1) + \frac{1}{2} f(x_1, y_2) \right) + \right. \\
&\quad \left. + \frac{1}{2} \left(\frac{1}{2} f(x_2, y_0) + f(x_2, y_1) + \frac{1}{2} f(x_2, y_2) \right) \right]
\end{aligned}$$