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 «Программная инженерия»

Лабораторная работа № 5

Тема Построение и программная реализация алгоритмов численного интегрирования.

Студент Бугаенко Андрей Павлович

Группа ИУ7-45Б

Оценка (баллы)

Преподаватель Градов В.М.

Москва.
2020 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

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

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.

Результаты работы программы:

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

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

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

Код программы:

```
from typing import List, Callable as func

from math import exp, sin, cos, pi

from numpy import linspace, array

from numpy.linalg import solve

from matplotlib import pyplot as plt

from math import cos, pi


def leg_polym(n, x):

    if n < 2: return [1, x][n]

    P1, P2 = leg_polym(n - 1, x), leg_polym(n - 2, x)
```

```
    return ((2 * n - 1) * x * P1 - (n - 1) * P2) / n

def leg_polym_proz(n, x):
    P1, P2 = leg_polym(n - 1, x), leg_polym(n, x)
    return n / (1 - x * x) * (P1 - x * P2)

def leg_roots(n: int, eps: float = 1e-12) -> List[float]:
    roots = [cos(pi * (4 * i + 3) / (4 * n + 2)) for i in range(n)]
    for i, root in enumerate(roots):
        root_val = leg_polym(n, root)
        while abs(root_val) > eps:
            root -= root_val / leg_polym_proz(n, root)
            root_val = leg_polym(n, root)
        roots[i] = root
    return roots

def norm_gauss_integ(f, n):
    t = leg_roots(n)
    T = array([[t_i**k for t_i in t] for k in range(n)])
    int_tk = lambda k: 2 / (k + 1) if k % 2 == 0 else 0
    b = array([int_tk(k) for k in range(n)])
    A = solve(T, b)
    return sum(A_i * f(t_i) for A_i, t_i in zip(A, t))

def gauss_integ(f, a, b, n):
    mean, diff = (a + b) / 2, (b - a) / 2
    g = lambda t: f(mean + diff * t)
    return diff * norm_gauss_integ(g, n)

def simp_integ(f, a, b, n):
    h, res = (b - a) / n, 0
```

```
    for i in range(0, n, 2):
        x1, x2, x3 = i * h, (i + 1) * h, (i + 2) * h
        f1, f2, f3 = f(x1), f(x2), f(x3)
        res += f1 + 4 * f2 + f3
    return h / 3 * res


def compose_integ(f, a1, b1, a2, b2, method_1, method_2, n1, n2):
    F = lambda y: method_1(lambda x: f(x, y), a1, b1, n1)
    return method_2(F, a2, b2, n2)


def function_integ(f, a, b, c, d, n, m):
    return compose_integ(f, a, b, c, d, gauss_integ, simp_integ, n, m)


def function(t, n, m):
    L_R = lambda theta, phi: 2 * cos(theta) / (1 - sin(theta)**2 * cos(phi)**2)
    f = lambda theta, phi: (1 - exp(-t * L_R(theta, phi))) * cos(theta) * sin(theta)
    return 4 / pi * function_integ(f, 0, pi / 2, 0, pi / 2, n, m)


tao = linspace(0.05, 10, 100)
eps = [function(t, 4, 5) for t in tao]


plt.plot(tao, eps)
plt.grid()
plt.show()
```

Результаты работы:

1. Алгоритм вычисления n корней полинома Лежандра n-ной степени:

Для алгоритма вычисления n корней полинома Лежандра используем простой итеративный метод Ньютона:

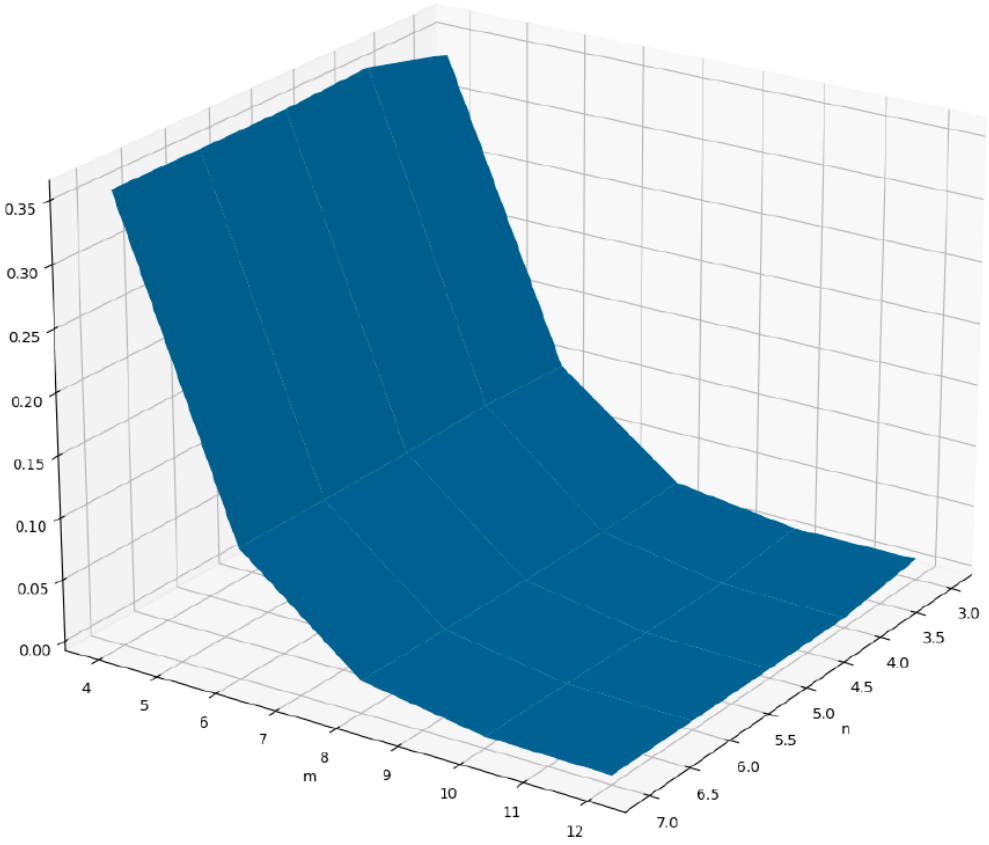
$$x_i = \cos\left(\frac{(4i+3)\pi}{4n+2}\right), \quad i := \overline{0,n-1}, \quad \epsilon := 10^{-12}$$

while $|P_n(x_i)| > \epsilon$ *do*

$$x_i := x_i - \frac{P_n(x_i)}{P_n'(x_i)}$$

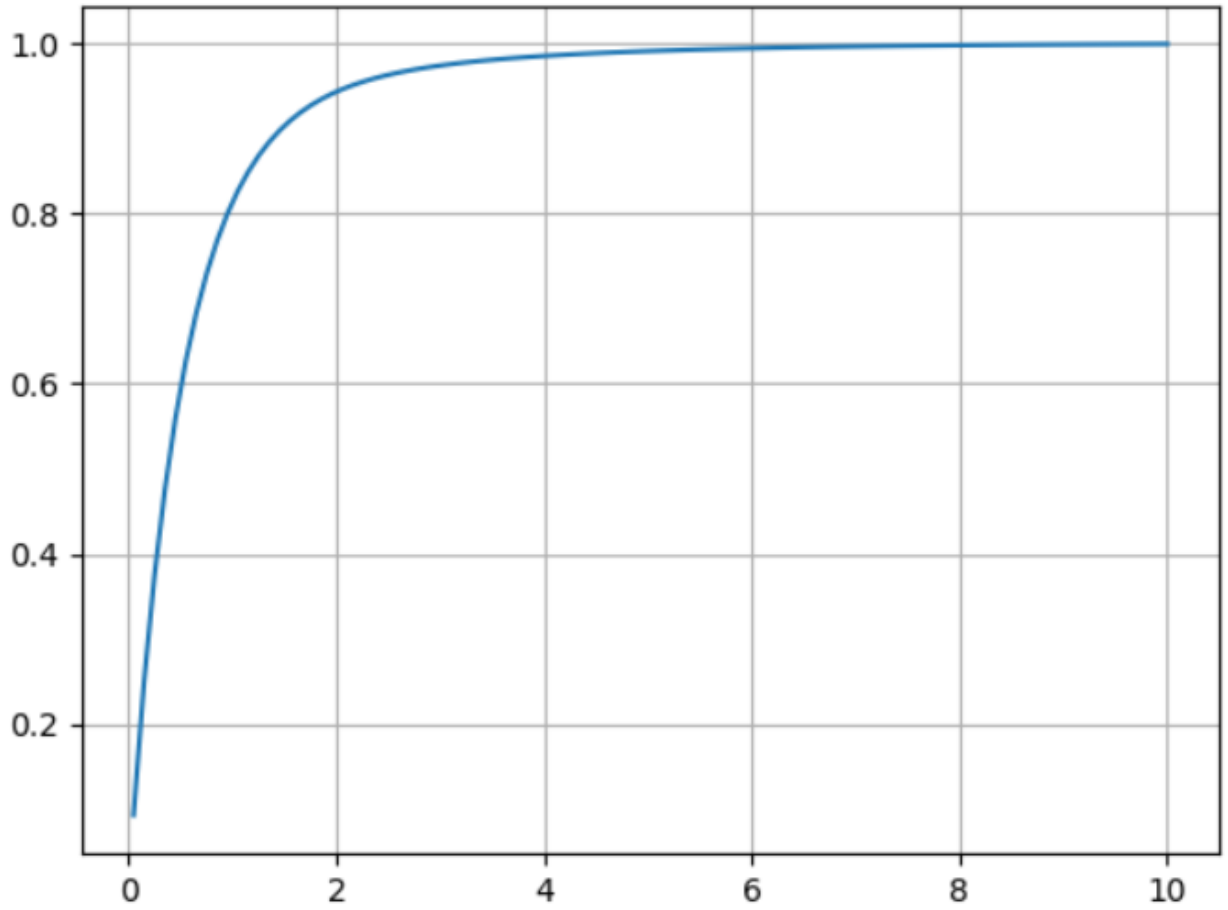
end

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



Данный график соответствует функции зависимости среднеквадратичной ошибки вычисления исходной функции. За эталон выбрана исходная функция при максимально возможных параметрах: n = 7 для метода Гаусса и m = 12 для формулы Симпсона. По графику можно сказать, что зависимость ошибки от параметра n несущественна, что означает более высокую точность метода Гаусса в сравнении с формулой Симпсона. Поэтому для направления интегрирования по методу Гаусса можно брать небольшие количества узлов сетки – примерно 3, тогда как по второму направлению около 10.

3. График зависимости ε(τ) при n = 3, m = 12:



По оси y – значение эпсилон.
По оси x – значение тау.

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

В случае, когда интегрируемая функция делает резкие скачки при больших значениях производной, теоретический порядок точности квадратурных формул численного интегрирования может не достигаться.

2. Построить формулу Гаусса численного интегрирования при одном узле.

$$N = 1: \quad \int_a^b f(x)dx \approx \frac{b-a}{2} A_1 f(x_1)$$

$$P_1(t) = t \;\rightarrow\; t_1 = 0$$

$$x_1 = \frac{a+b}{2} + \frac{b-a}{2} t_1 = \frac{a+b}{2}$$

$$A_1 = 2$$

$$\int_a^b f(x)dx \approx (b-a)\, f\left(\frac{a+b}{2}\right)$$

3. Построить формулу Гаусса численного интегрирования при двух узлах.

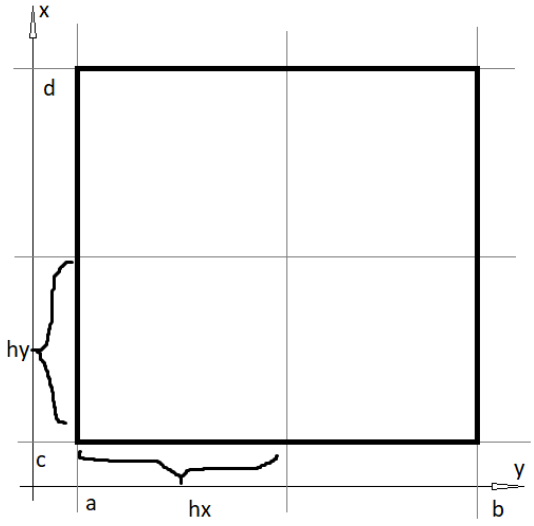
$$N=2:\quad \int_a^b f(x)dx \approx \frac{b-a}{2}\left[\, A_1 f(x_1) + A_2 f(x_2)\,\right]$$

$$P_2(t) = 3t^2 - 1 \;\rightarrow\; t_1 = -\frac{1}{\sqrt{3}},\; t_2 = \frac{1}{\sqrt{3}}$$

$$x_1 = \frac{a+b}{2} + \frac{b-a}{2}t_1 = \frac{a+b}{2} - \frac{b-a}{2\sqrt{3}}$$
$$x_2 = \frac{a+b}{2} + \frac{b-a}{2}t_2 = \frac{a+b}{2} + \frac{b-a}{2\sqrt{3}}$$
$$\begin{cases} A_1 + A_2 = 2 \\ A_1 t_1 + A_2 t_2 = 0 \end{cases} \;\rightarrow\; A_1 = A_2 = 1$$

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\left[f\left(\frac{a+b}{2} - \frac{b-a}{2\sqrt{3}}\right) + f\left(\frac{a+b}{2} + \frac{b-a}{2\sqrt{3}}\right)\right]$$

4. Получить обобщенную кубатурную формулу, аналогичную (6.6) из лекции №6, для вычисления двойного интеграла методом последовательного интегрирования на основе формулы трапеций с тремя узлами по каждому направлению.



$$h_x = \frac{b-a}{2},\; h_y = \frac{d-c}{2}$$
$$I = \int_c^d \int_a^b f(x,y)dx\,dy = \int_c^d F(y)dy,\;\; F(y) = \int_a^b f(x,y)dx$$

$$F(y) = \frac{h_x}{2}\left(f(a,y) + 2f(\frac{a+b}{2},y) + f(b,y)\right)$$
$$\int_c^d F(y)dy = \frac{h_y}{2}\left(F(c) + 2F(\frac{c+d}{2}) + F(d)\right)$$

$$I = \frac{h_x h_y}{4}\begin{pmatrix} f(a,c) & + 2f(\frac{a+b}{2},c) & + f(b,c) + \\ 2f(a,\frac{c+d}{2}) & + 4f(\frac{a+b}{2},\frac{c+d}{2}) & + 2f(b,\frac{c+d}{2}) + \\ f(a,d) & + 2f(\frac{a+b}{2},d) & + f(b,d) \end{pmatrix}$$