

**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**

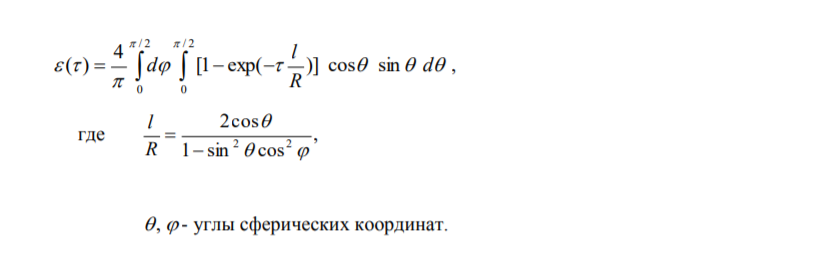
**Оценка (баллы)**

**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

****

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)

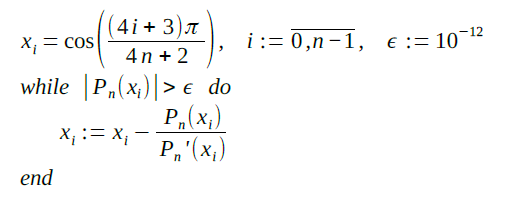
plt.grid()

plt.show()

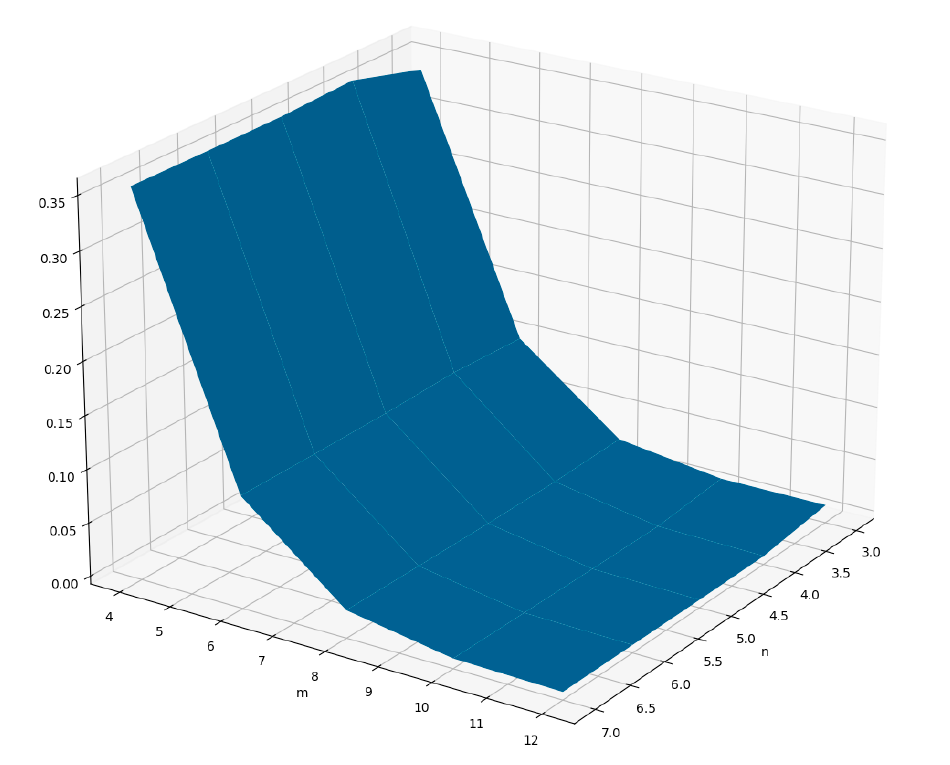
**Результаты работы:**

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

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



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.