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

Дисциплина Вычислительные алгоритмы

Тема Построение и программная реализация алгоритмов численного интегрирования.

Студент Хетагуров П.К.

Группа ИУ7-45

Оценка (баллы) _____

Преподаватель Градов В. Г

Москва.
2020 г.

Задание

Тема: Построение и программная реализация алгоритмов численного интегрирования

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

Задание.

Работа основывается на материалах лекций №5 и 6.

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

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

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.

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

Сам полином Лежандра:

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

Корни полинома вычисляются итеративно по методу Ньютона .

$$x_i^{(k+1)} = x_i^{(k)} - \frac{P_n(x_i^{(k)})}{P_n'(x_i^{(k)})},$$

А начальное приближение для i-го корня берется по формуле:

$$x_i^{(0)} = \cos[\pi(4i - 1)/(4n + 2)].$$

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

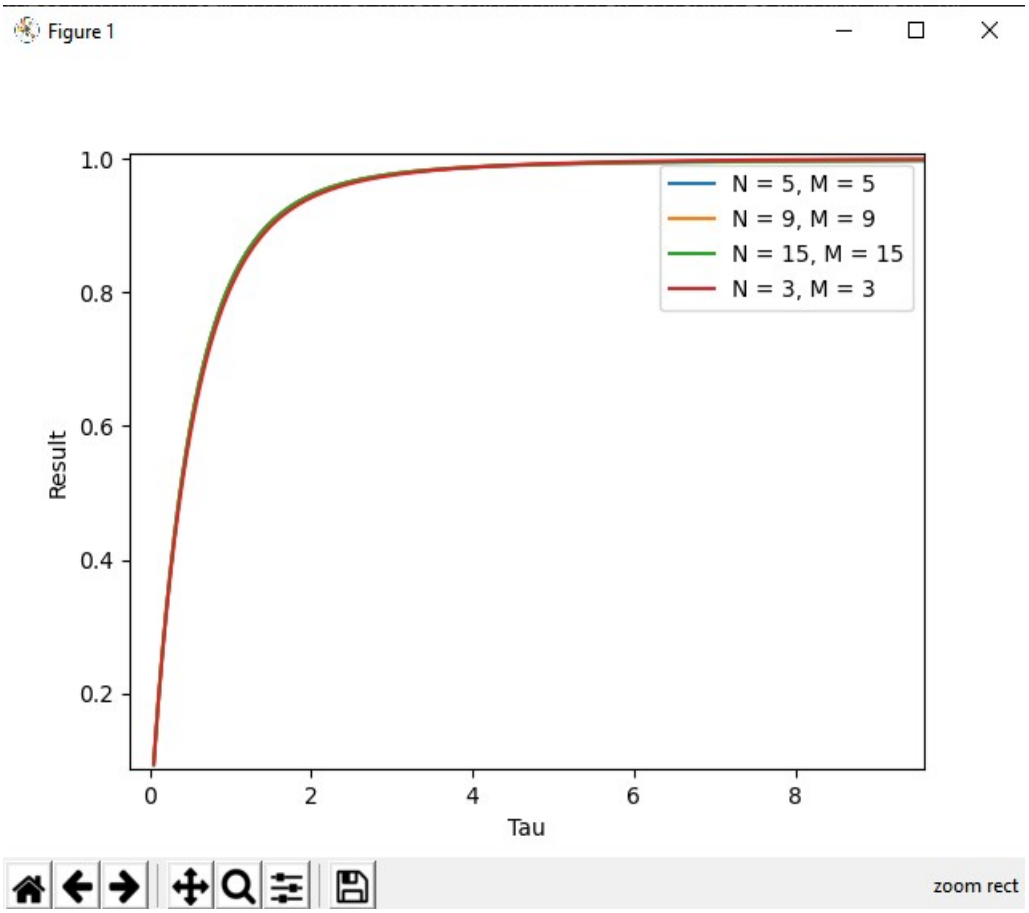
Значения получены при tau = 1.

Ожидаемый результат ~ 0.814

N(внутренний)	M(внешний)	Result
3	1	1.332
3	3	0.805
3	5	0.809
3	7	0.809
5	3	0.813
7	3	0.812
9	3	0.812
9	9	0.814

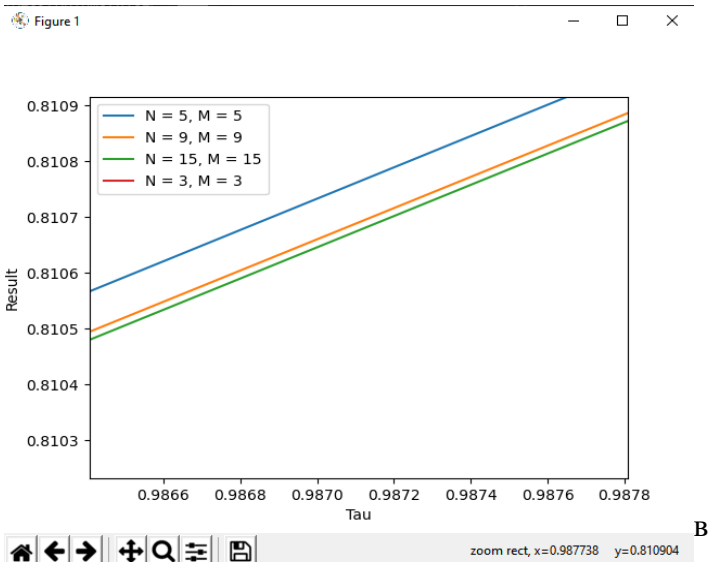
Как видно из таблицы, увеличение точности внутреннего интеграла более существенно влияет на результат.

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



Как видно из графика, наименьшее отклонение от ожидаемого результата достигается при tau ~ 1.

Количество узлов же не влияет на поведение графика, а при количистве узлов > 5 графики почти совпадают



в

Вопросы при защите лабораторной работы.

1. В каких ситуациях теоретический порядок квадратурных формул численного

интегрирования не достигается.

Если подынтегральная функция не имеет соответствующих производных, то указанный теоретический порядок точности не достигается. Так, если на отрезке интегрирования не существуют 3-я и 4-я производные, то порядок точности формулы Симпсона будет только 2-ой.

2. Построить формулу Гаусса численного интегрирования при одном узле.

$$\int_a^b = \frac{b-a}{2}\, 2f(\frac{b+a}{2})$$

3. Построить формулу Гаусса численного интегрирования при двух узлах.

$$\int_a^b = \frac{b-a}{2}\, (f(\frac{b+a}{2} - \frac{b-a}{2}\frac{1}{\sqrt{3}}) + f(\frac{b+a}{2} + \frac{b-a}{2}\frac{1}{\sqrt{3}}))$$

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.

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

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

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

Код:

```
def func_form(param):
    subfunc = lambda x, y: 2 * cos(x) / (1 - (sin(x) ** 2) * (cos(y) ** 2))
    func =    lambda x, y: (4 / pi) * (1 - exp(-param * subfunc(x, y))) * cos(x) * sin(x)
    return func
```

```
def pin_x(func, x):
    return lambda y: func(x, y)

# limits = ([a, b], [c, d])
def two_integrate(N, M, limits, func):
    first = lambda x: simpson(pin_x(func, x), limits[0][0], limits[0][1], N)
    return gauss_quadrature(first, limits[1][0], limits[1][1], M)

def simpson(func, a, b, nodes_count):

    h = (b - a) / (nodes_count - 1)
    x = a
    res = 0

    for i in range((nodes_count - 1) // 2):
        res += func(x) + 4 * func(x + h) + func(x + 2 * h)
        x += 2 * h

    return (h / 3) * res



def gauss_x_transform(t, a, b):
    return (b + a) / 2 + (b - a) / 2 * t



def gauss_quadrature(func, a, b, nodes_count):
      args, coeffs = leggauss(nodes_count)  # Computes the sample points and weights for Gauss-Legendre quadrature
    res = 0

    for i in range(nodes_count):
        res += (b - a) / 2 * coeffs[i] * func(gauss_x_transform(args[i], a, b))

    return res
```