|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

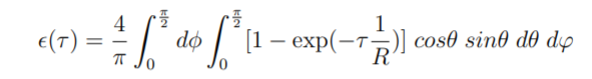
**Дисциплина**  Вычислительные алгоритмы

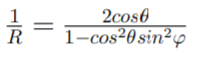
|  |  |
| --- | --- |
| **Тема** Численное интегрирование  **Студент** Воякин А.Я.  **Группа ИУ7-44Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_\_Градов В.М.\_\_** |  |

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

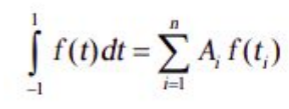
,

 где

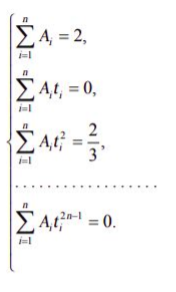
Применить метод последовательного интегрирования. По одному направлению использовать формулу Гаусса, а по-другому – формулу Симпсона.

1. **Алгоритм**

Предположим вычисление интеграла на стандартном интервале [-1,1]. Имеем квадратурную формулу:

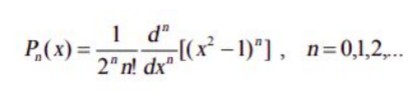


Коэффициенты Ai и ti можно найти из системы 2n уравнений:



Решение полученной системы достаточно трудно найти, так как она нелинейна.

Поэтому будем рассматривать полином Лежандра:

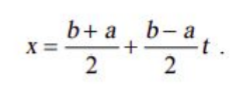


Узлами формулы Гаусса являются нули многочлена Лежандра

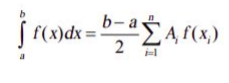
степени n.

Зная ti, из системы уравнений можно найти коэффициенты Ai.

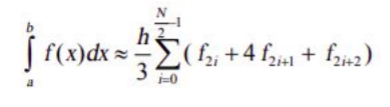
Для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной:



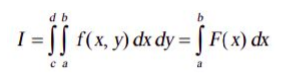
Далее получаем:

, где

При выполнении лабораторной работы нам также понадобится квадратная формула Симпсона:



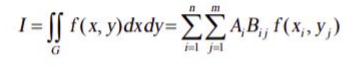
Рассмотрим интеграл по прямоугольной области:

, где

По каждой координате введем сетку узлов.

Каждый однократный интеграл вычисляют по квадратурным формулам.

Конечная формула:



где Ai, Bij - известные постоянные.

Все корни полинома лежат на интервале [-1, 1]. При этом интервалы

[-1,0] и [0,1] – симметричны, поэтому при поиске достаточно рассмотреть один интервал [0,1].

1. **Влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов**

|  |  |  |
| --- | --- | --- |
| **N** | **M** | **Результат** |
| 2 | 2 | 0.777 |
| 2 | 3 | 0.767 |
| 2 | 4 | 0.767 |
| 2 | 5 | 0.779 |
| 3 | 2 | 0.800 |
| 4 | 2 | 0.806 |
| 5 | 2 | 0.809 |
| 5 | 5 | 0.815 |

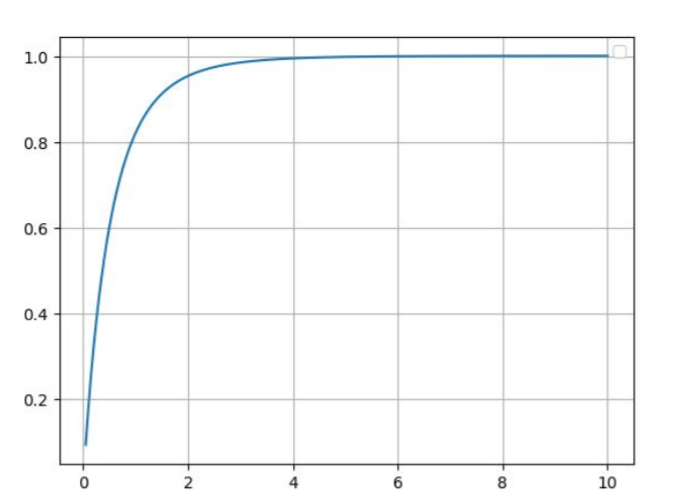
Из таблицы видно, что при увеличении M (N=2), результат меньше,

чем при равном количестве узлов (N = M = 5). При увеличении

N (M=2) можно увидеть похожу картину, но результат уже ближе к полученному при одинаковом количестве узлов. Таблица заполнена при 𝜏 = 1

1. **График зависимости**

**N=M=5**



Как видно на графике, с увеличением 𝜏 – увеличивается и

 но и не превышает единицы, так как < 1

**Ответы на контрольные вопросы:**

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

Теоретический порядок квадратурных формул численного интегрирования не достигается, если подынтегральная функция не имеет соответствующих производных.

1. **Построить формулу Гаусса численного интегрирования при одном узле.**

A0 = 2, t0 = 0

1. **Построить формулу Гаусса численного интегрирования при двух узлах.**

A0 = 1, A1= 1, t0 = - , t1 =

1. **Получить обобщенную кубатурную формулу, на основе методе трапеций, с тремя узлами на каждом направлении**

***Код программы***

def converts(func2, value):

return lambda y: func2(value, y)

def variable\_conversion(a, b, t):

return (b + a) / 2 + (b - a) \* t / 2

def func(parameter):

return lambda x, y: (4 / pi) \* (1 - \

exp(-parameter \* 2 \* cos(x) / (1 - \

(sin(x) \*\* 2) \* (cos(y) \*\* 2)))) \* cos(x) \* sin(x)

def gauss(func, a, b, num\_of\_nodes):

args, coeffs = leggauss(num\_of\_nodes)

res = 0

for i in range(num\_of\_nodes):

res += (b - a) / 2 \* coeffs[i] \* \

func(variable\_conversion(a, b, args[i]))

return res

def simpson(func, a, b, num\_of\_nodes):

if (num\_of\_nodes < 3 or num\_of\_nodes & 1 == 0):

raise ValueError

h = (b - a) / (num\_of\_nodes - 1)

x = a

res = 0

for i in range((num\_of\_nodes - 1) // 2):

res += func(x) + 4 \* func(x + h) + func(x + 2 \* h)

x += 2 \* h

return res \* (h / 3)

def result(func, n, m, tao):

return simpson(lambda x: gauss(converts(func, x), \

LIMITS[1][0], LIMITS[1][1], m), LIMITS[0][0], LIMITS[0][1], n)

N = int(input("\033[36m» Введите N: "))

M = int(input("» Введите M: "))

tao = float(input("» Введите τ: "))

print("Результат: ", result(function(tao),N, M, tao))