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)

Дисциплина «Вычислительные алгоритмы»

Лабораторная работа №5

по теме:

«Построение и программная реализация алгоритмов численного интегрирования.»

Работу выполнил:

студент группы ИУ7-43Б

Сукочева А.

Работу проверил:

Градов В.М

2020 г.

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:

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.

Описание алгоритма:

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

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

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

$$\begin{cases} \sum_{i=1}^{n} A_i = 2, \\ \sum_{i=1}^{n} A_i t_i = 0, \\ \sum_{i=1}^{n} A_i t_i^2 = \frac{2}{3}, \\ \dots\dots\dots\dots\dots\dots \\ \sum_{i=1}^{n} A_i t_i^{2n-1} = 0. \end{cases}$$

Данная система нелинейная, и ее решение найти довольно трудно. Поэтому рассмотрим полиномом Лежандра. Формула:

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

Узлами формулы Гаусса являются нули многочлена Лежандра степени n. Зная ti, из системы (выше) уравнений легко найти коэффициенты Ai.

Для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной:

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

Получим:

$$\int_a^b f(x)dx = \frac{b-a}{2}\sum_{i=1}^{n} A_i f(x_i)$$

Где:

$$x_i = \frac{b+a}{2} + \frac{b-a}{2}t_i$$

В данной лабораторной работе нам также понадобится квадратурная формула Симпсона:

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

Рассмотрим интеграл по прямоугольной области

$$I = \int_c^d\int_a^b f(x, y)\, dx\, dy = \int_a^b F(x)\, dx$$

где

$$F(x) = \int_{c}^{d} f(x, y)dy.$$

По каждой координате введем сетку узлов. Каждый однократный интеграл вычисляют по квадратурным формулам. Конечная формула:

$$I = \iint_{G} f(x, y)dxdy = \sum_{i=1}^{n}\sum_{j=1}^{m} A_i B_{ij}\, f(x_i, y_j)$$

где Ai, Bij - известные постоянные.

Нам нужно искать корни только на отрезке (0; 1] (Из-за симметрии)

Далее представлена программа.

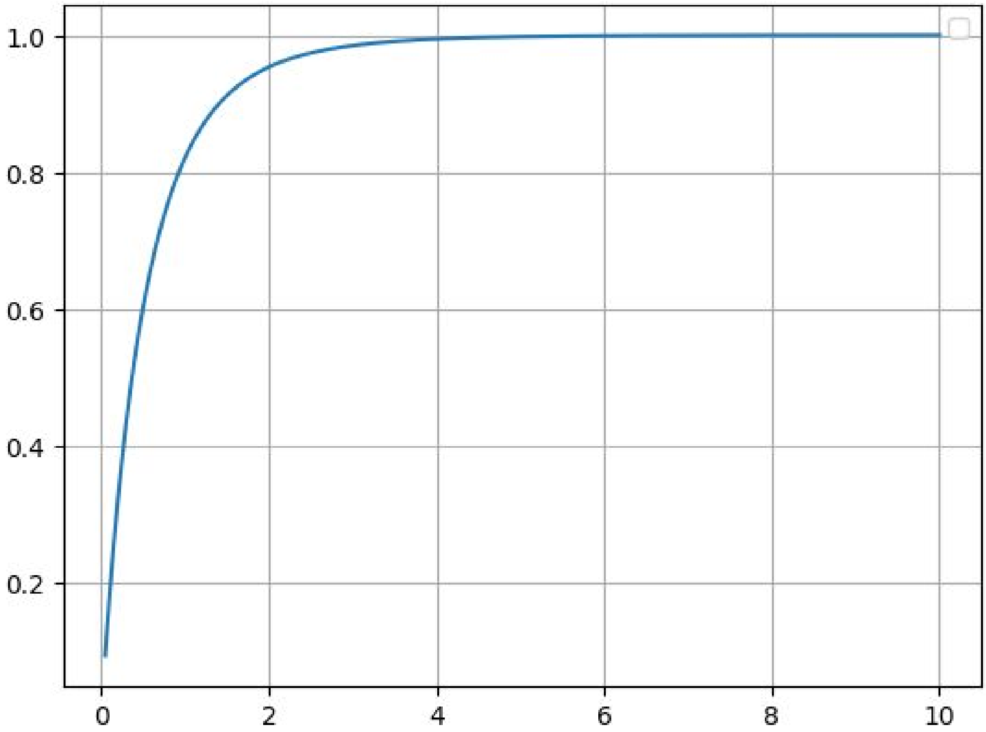
```python
def converts(func2, value):
    return lambda y: func2(value, y)


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


def function(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)


def main():
    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))
```

```
☁ lab_05 [lab_05] ⚡ python3 main.py
» Введите N: 3
» Введите M: 1
» Введите τ: 1
Результат:  0.8883007737422615
☁ lab_05 [lab_05] ⚡ python3 main.py
» Введите N: 3
» Введите M: 2
» Введите τ: 1
Результат:  0.8890043702642582
☁ lab_05 [lab_05] ⚡ python3 main.py
» Введите N: 3
» Введите M: 3
» Введите τ: 1
Результат:  0.8886948534129957
☁ lab_05 [lab_05] ⚡ python3 main.py
» Введите N: 3
» Введите M: 4
» Введите τ: 1
Результат:  0.8888050660353763
☁ lab_05 [lab_05] ⚡ python3 main.py
» Введите N: 3
» Введите M: 5
» Введите τ: 1
Результат:  0.8887830243938745
```

При увеличении M результат почти совпадает, что говорит нам о том, что большее влияние оказывает точность внешнего интегрирования.

```
☁ lab_05 [lab_05] ⚡ python3 main.py
» Введите N: 3
» Введите M: 3
» Введите τ: 1
Результат:  0.8886948534129957
☁ lab_05 [lab_05] ⚡ python3 main.py
» Введите N: 5
» Введите M: 3
» Введите τ: 1
Результат:  0.8208551739111585
☁ lab_05 [lab_05] ⚡ python3 main.py
» Введите N: 7
» Введите M: 3
» Введите τ: 1
Результат:  0.8152779841066683
```

График зависимости:



N = M = 5. Как и было сказано в физическом содержание задачи степень черноты не может быть больше 1.

Ответы контрольные вопросы:

1. Если подынтегральная функция не имеет соответствующих производных. К примеру, если на отрезке интегрирования не существует 3-я и 4-я производные, то порядок точности формулы Симпсона будет только 2-ой. O(h^2).
2. $\int_a^b f(x)dx = \frac{b-a}{2}\, 2f(\frac{b+a}{2})$
3. $\int_a^b f(x)dx = \frac{b-a}{2}(f(\frac{b+a}{2} - \frac{b-a}{2}(\frac{1}{\sqrt{3}})) + f(\frac{b+a}{2} + \frac{b-a}{2}(\frac{1}{\sqrt{3}}))$