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
по дисциплине "Вычислительные алгоритмы"

Тема Численное интегрирование.

Студент Романов А.В.

Группа ИУ7-43Б

Оценка (баллы) ________________

Преподаватель Градов В.М.

Москва — 2020 г.

1. Тема работы

Построение и программная реализация алгоритмов численного интегрирования.

2. Цель работы

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

3. Задание

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

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

где $\frac{1}{R} = \frac{2cos\theta}{1 - cos^2\theta sin^2\varphi}$

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

4. Описание алгоритма

Имеем $\int_{-1}^{1} f(t)dt = \sum_{i=1}^{n} A_i f(t_i)$, положим $\int_{-1}^{1} t^k dt = \sum_{i=1}^{n} A_i f(t_i^k)$, $k = 0, 1, 2, ..., 2n - 1$

Имеем систему:

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

Системая нелинейная, найти решение сложно. Для нахождения A_i и t_i можно воспользоваться полиномом Лежандра. Формула полинома:

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

Узлами формулы Гаусса являются нули полинома Лежандра $P_n(t)$, а A_i можно найти из вышеуказанной системы уравнений.

При вычислении интеграла на произвольном интервале $[a, b]$, для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной:

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

В таком случае, получаем конечную формулу для произвольного интервала $[a, b]$:

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

Так же, существует квадратнурная формула Симпсона:

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

Однако, эти методы можно применять и для приближенной оценки двукратных (и не только) интегралов. Рассмотрим интеграл по прямоугольной области:

$$I = \int_c^d \int_a^b f(x,y)dx\,dy = \int_a^b F(x)dx, \quad \text{где } F(x) = \int_c^d f(x,y)dy$$

По каждой координате введем сетку узлов. Каждый однократный интеграл вычисляют по квадратурным формулам. Для разных направлений можно использовать квадратурные формулы разных порядков точности, в т.ч. и Гаусса.

Конечная формула:

$$I = \int\int_G f(x,y)dx\,dy = \sum_{i=1}^{n}\sum_{j=1}^{m} A_i B_{ij} f(x_i, y_j)$$

где $A_i B_{ij}$ – известные постоянные.

5. Результаты

5.1. Алгоритм вычисления n корней полинома Лежандра n-ой степени.

Во-первых, стоит отметить что все корни полинома лежат на интервале [-1, 1]. При этом, интервалы [-1, 0] и [0, 1] – симметричны, так что при поиске достаточно рассмотреть интервал [0, 1]
Корни полинома можно вычислить итеративно по методу Ньютона:

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

причем начальное приближение для i-го корня берем по формуле:

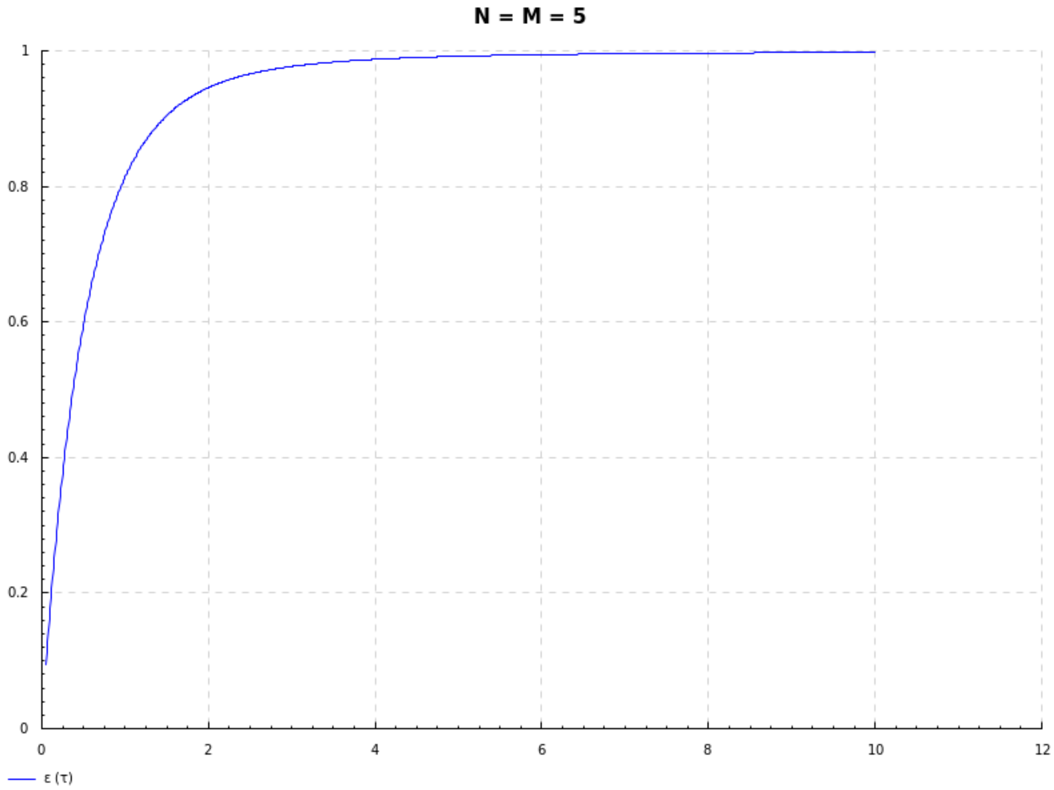
$$x_i^{(0)} = cos[\frac{\pi(4i-1)}{4n+2}]$$

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

N	M	Результат
2	2	0.775
2	3	0.766
2	4	0.768
2	5	0.778
3	2	0.801
4	2	0.805
5	2	0.807
5	5	0.814

Как можно заметить, при увеличении M (N = 2), результат меньше, чем при равном количестве узлов (N = M = 5). При увеличении N (M = 2) – видим аналогичную картину, но результат уже ближе к полученному при одинаковом количестве узлов. Таблица составлена при $\tau = 1$.

5.3. График зависимости $\epsilon(\tau)$.



Как видно на графике, с увеличением τ – увеличивается и $\epsilon(\tau)$, но и не превышает единицы, т.к. $\epsilon(\tau) < 1$.

6. Ответы контрольные вопросы

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

Если подынтегральная функция не имеет соответствующих производных. Например, если на отрезке интегрирования не существует 3-я и 4-я производные, то порядок тончности формула Симпсона будет только 2-ой.

2. Построить формулу Гаусса численного интегрирования при одном узле.

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

$A_0 = 2,\ t_0 = 0$

3. Построить формулу Гаусса численного интегрирования при двух узлах.

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

$A_0 = 1,\ A_1 = 1,\ t_0 = -\frac{1}{\sqrt{3}},\ t_1 = \frac{1}{\sqrt{3}}$

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

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

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

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

7. Код программы

Файл Main.hs:

```haskell
import System.IO
import Integration

main :: IO ()
main = do
    tau <- putStrLn "Enter tau: " >> fmap (\x -> read x :: Double) getLine
    n <- putStrLn "Enter N: " >> fmap (\x -> read x :: Int) getLine
    putStrLn "Enter M: " >> fmap (\x -> read x :: Int) getLine >>= print .
        gauss2 limits tau n
```

Файл Integration.hs:

```haskell
module Integration (
    limits,
    gauss2,
    simpson2,
) where

import Math.Polynomial.Legendre

type N = Int
type M = Int
type Tau = Double

type Matrix = [[Double]]
type Coeffs = [Double]
type Roots = [Double]

data Limits = Limits { a :: Double,
                       b :: Double,
                       c :: Double,
                       d :: Double
                     } deriving (Show)

limits :: Limits
limits = Limits 0.0 (pi / 2) 0.0 (pi / 2)

eps :: Double
eps = 0.001

f :: Double -> Double -> Tau -> Double
f phi theta tau = 4 / pi * ((1 - exp(-tau * ((2 * cos phi) /
    (1 - (sin phi)^2 * (cos theta)^2)))) * (cos phi) * (sin phi))

subtractRow :: [Double] -> [Double] -> [Double]
subtractRow subRow row = map (\x -> fst x - snd x * (head row / head subRow)) $
    zip row subRow

triangulation :: Matrix -> Matrix
triangulation matrix
    | length matrix == 0 = matrix
    | otherwise = head matrix :
        triangulation (map tail (map (subtractRow $ head matrix) $ tail matrix))

gaussSLE :: Matrix -> Coeffs
gaussSLE = coeffs . reverse . triangulation
    where coeffs = foldl (\x y -> (last y - (sum $ zipWith (*) (init $ tail y) x
        )) / (head y) : x) []

getKi :: Int -> Double
```

```haskell
getKi ind
    | ind `mod` 2 == 0 = 2 / (fromIntegral ind + 1)
    | otherwise = 0

getCoeffs :: Roots -> Coeffs
getCoeffs roots = gaussSLE $ foldr (\x acc -> ((map (^x) roots) ++ [getKi x]) :
   acc) [] [0..length roots - 1]

value :: Double -> Double -> Double -> Double
value a b root = (a + b) / 2 + root * (b - a) / 2

gauss :: Double -> M -> Tau -> Limits -> Double
gauss x m tau limits = (d limits - c limits) * sum' / 2
    where roots = legendreRoots (m + 1) eps
          ys = map (value (c limits) $ d limits) roots
          sum' = foldr (\y acc -> acc + (snd y * f x (fst y) tau)) 0 $ zip ys $
              getCoeffs roots

gauss2 :: Limits -> Tau -> N -> M -> Double
gauss2 limits tau n m = (b limits - a limits) * sum' / 2
    where roots = legendreRoots (n + 1) eps
          xs = map (value (c limits) (d limits)) roots
          sum' = foldr (\x acc -> acc + (snd x * (gauss (fst x) m tau limits)))
              0 $ zip xs $ getCoeffs roots

simpson2 :: Limits -> Tau -> N -> M -> Double
simpson2 limits tau n m = h / 3 * sum_of
    where h = (b limits - a limits) / (fromIntegral n)
          steps = take (n `div` 2) [a limits, a limits + 2 * h..10000]
          gauss' a = gauss a m tau limits
          sum_of = foldr (
              \a acc ->
                  acc + (gauss' a) + (4 * (gauss' $ a + h) + (gauss' $ a + 2 * h
                      ))) 0 steps
```