Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №2

Метод трапеций

Преподаватель: Перл Ольга Вячеславовна

Выполнили: Кульбако Артемий Юрьевич P3212

Санкт-Петербург
2020

Описание метода

Метод трапеций – модификация метода прямоугольников, дающая более точные результаты. Идея заключается в разбиении площади под графиком подынтегральной функции на равные по ширине трапеции, и суммировании их площадей.

$$S_{\text{общ}} = S_1 + S_2 + \cdots + S_n = \frac{y_o + y_1}{2} h_1 + \frac{y_1 + y_2}{2} h_2 + \cdots + \frac{y_{n-1} + y_n}{2} h_n =$$

$$= \int_a^b f(x)dx = \frac{1}{2} \sum_{i=1}^{n} h_i (y_{i-1} + y_i) = I_n$$

$$\text{где } y_o = f(a),\ \ y_n = f(b),\ \ y_i = f(x_i),\ \ h_i = x_i - x_{i-1}$$

После вычисления I_n проводится повторное интегрирование для $I_{2n} = \frac{1}{2}\sum_{i=1}^{2n} h_i(y_{i-1} + y_i)$ и вычисляется погрешность по правилу Рунге:

$$\Delta_{2n} = \frac{|I_{2n} - I_n|}{3}$$

Если $\Delta_{2n} < \varepsilon$ (ε – требуемая точность), то количество разбиений увеличивается в 2 раза, и Δ_{2n} оценивается ещё раз.

Вывод

Все три метода: прямоугольников, трапеций, парабол (Симпсона) являются модификациями метода Ньютона-Котеса, основанного на замене подынтегральной функции интерполяционным многочленом Лангража

$$\int_a^b ydx = (b-a) \sum_{i=0}^{n} H_i y_i$$

, где точность решения растёт с увеличением степени интерполяционного выражения. Погрешность же для каждого из методов определяется формулами:

1. Для средних прямоугольников: $|\Delta| \le max_{x\in[a,b]}|f''(x)| \cdot \frac{(b-a)^3}{24^{\ 2}}\ , k = 2$
2. Для трапеций: $|\Delta| \le max_{x\in[a,b]}|f''(x)| \cdot \frac{(b-a)^3}{12n^2}\ , k = 2$
3. Для парабол: $|\Delta| \le max_{x\in[a,b]}|f''(x)| \cdot \frac{(b-a)^5}{189n^4}\ , k = 4$

$$\text{где } k - \text{порядок точности}$$

Это говорит нам о том, что метод трапеций менее точные чем метод парабол, при равном количестве разбиений. На практике же применятся метод оценки погрешности Рунге:

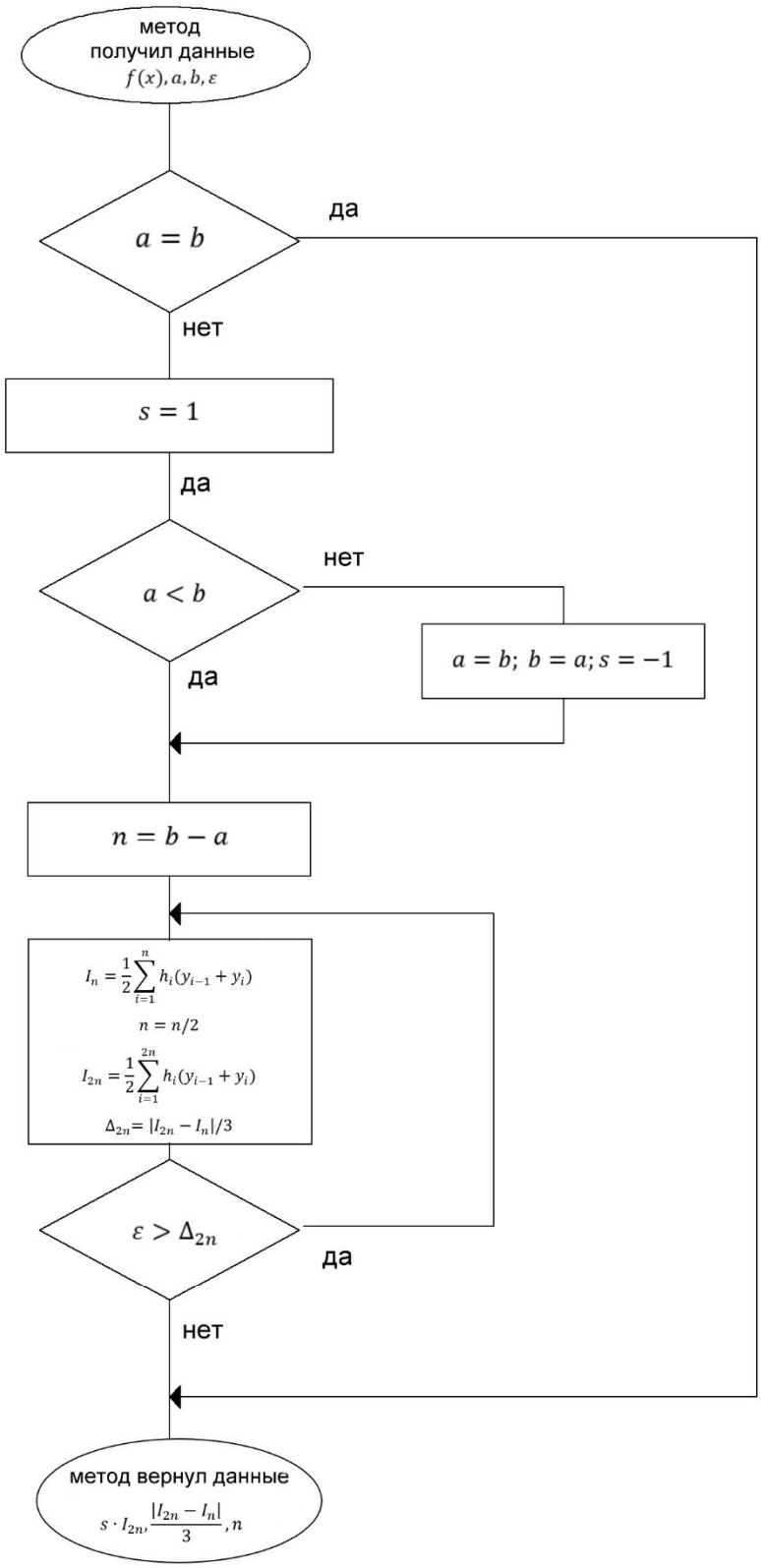
$$I - I_{\frac{h}{2}} \approx \frac{I_{\frac{h}{2}} - I}{2^k - 1}$$

Примеры

```
Введите номер желаемой функции:
0. x^2 dx
1. 1/ln(x) dx
2. cos(x)/(x+2) dx
3. sqrt(1 + 2x^2 - x^3) dx
0
Введите пределы интегрирования через пробел:
0 2
Введите точность:
0.01
Значение интеграла = 2.671875
Количество разбиений = 16
Погрешность = 0.005208333333333333
```

```
Введите номер желаемой функции:
0. x^2 dx
1. 1/ln(x) dx
2. cos(x)/(x+2) dx
3. sqrt(1 + 2x^2 - x^3) dx
4
Ошибка ввода: введите целое число в [1 ; 5].
3
Введите пределы интегрирования через пробел:
1.2 2
Введите точность:
0.000001
Значение интеграла = 1.090122705480749
Количество разбиений = 512
Погрешность = 4.4018402671023676E-7
```

Блок-схема


метод
получил данные
$f(x), a, b, \varepsilon$
$a = b$
да
нет
$s = 1$
да
$a < b$
нет
$a = b;\ b = a; s = -1$
да
$n = b - a$
$I_n = \frac{1}{2}\sum_{i=1}^{n} h_i(y_{i-1} + y_i)$
$n = n/2$
$I_{2n} = \frac{1}{2}\sum_{i=1}^{2n} h_i(y_{i-1} + y_i)$
$\Delta_{2n} = |I_{2n} - I_n|/3$
$\varepsilon > \Delta_{2n}$
да
нет
метод вернул данные
$s \cdot I_{2n}, \frac{|I_{2n} - I_n|}{3}, n$

```kotlin
import kotlin.math.abs

/**
 * Содержит результат решения интеграла.
 * @property resValue значение интеграла.
 * @property infelicity погрешность вычислений.
 * @property blocks количество разбиений.
 * @version 1.0
 */
data class IntegralAnswer(val resValue: Double, val infelicity: Double, val blocks: Int)

/**
 * Содержит результат пределы интегрирования.
 * @param limits верхний и нижний предел (не упорядоченные).
 * @property low нижний предел.
 * @property high верхний предел.
 * @property isSwitchedRange менялись ли пределы местами.
 * @version 1.0
 */
data class Limits(val limits: Pair<Double, Double>) {

    val low: Double
    val high: Double
    val isSwitchedRange: Boolean

    init {
        if (limits.first > limits.second) {
            low = limits.second
            high = limits.first
            isSwitchedRange = true
        } else {
            low = limits.first
            high = limits.second
            isSwitchedRange = false
        }
    }
}

/**
 * Находит численное значение интеграла.
 * @author Артемий Кульбако.
 * @version 1.0
 */
class IntegralSolver {

    companion object {
        /**
         * Находит значение интеграла методом трапеций.
         * @param mathFunc интеграл, который нужно посчитать.
         * @param limits пределы интеграла.
         * @param precision точность вычислений.
         * @return IntegralAnswer.
         * @version 1.0
         */
        fun integrateByTrapezoid(mathFunc: MathFunction, limits: Limits, precision: Double):
IntegralAnswer {

            fun approximate(mathFunc: MathFunction, limits: Limits, step: Double): Double {
                var area = 0.0
                for (i in 0 until ((limits.high - limits.low) / step).toInt()) {
                    area += step * 0.5 * (mathFunc.func(limits.low + i * step) + (mathFunc.func(limits.
low + (i + 1) * step)))
                    if (area.isNaN() || area.isInfinite()) throw Exception("Функция не определена на
заданном отрезке.")
                }
                return area
            }

            var step = limits.high - limits.low
            var error: Double
            var integralN: Double
            var integral2N = approximate(mathFunc, limits, step)
            do {
                integralN = integral2N
                step /= 2
                integral2N = approximate(mathFunc, limits, step)
                error = calcError(integral2N, integralN)
            } while (error > precision)
            if (limits.isSwitchedRange) integral2N = - integral2N
            return IntegralAnswer(integral2N, error, ((limits.high - limits.low) / step).toInt())
```

```
78          }
79
80          private fun calcError(integralN: Double, integral2N: Double) = abs(integral2N - integralN) / 3
81      }
82  }
```

```kotlin
/**
 * Представляет собой характеристику математической функции.
 * @property description описание функции.
 * @property func осуществляет расчёт неизвестного параметр x по переданному правилу.
 * @version 1.0
 */
interface MathFunction {

    fun func(xParam: Double): Double

    val description: String
}
```

```
import java.io.BufferedReader
import java.io.InputStreamReader
import java.lang.Exception
import java.lang.NumberFormatException
import kotlin.math.*

/**
 * Осуществляет взаимодействие с пользователем.
 * @param args аргументы командной строки.
 * @author Артемий Кульбако.
 * @version 1.0
 */
fun main(args: Array<String>) {

    fun buildFunctions(): MutableList<MathFunction> {
        return mutableListOf(
            object: MathFunction {
                override fun func(xParam: Double) = xParam.pow(2)
                override val description = "x^2"
            },
            object: MathFunction {
                override fun func(xParam: Double) = 1 / ln(xParam)
                override val description = "1/ln(x)"
            },
            object: MathFunction {
                override fun func(xParam: Double) = cos(xParam) / (xParam + 2)
                override val description = "cos(x)/(x+2)"
            },
            object: MathFunction {
                override fun func(xParam: Double) = sqrt(1 + 2 * xParam.pow(2) - xParam.pow(3))
                override val description = "sqrt(1 + 2x^2 - x^3)"
            }
        )
    }

    val keyReader = BufferedReader(InputStreamReader(System.`in`))
    val functions = buildFunctions()
    println("Введите номер желаемой функции:")
    functions.forEachIndexed { i, el -> println("${i}. ${el.description} dx") }
    var funcNumber: Int? = null
    var limits: List<Double>? = null
    var precision: Double? = null
    keyReader.use {
        var inputStep = 0
        while (inputStep < 3) {
            when (inputStep) {
                0 -> {
                    try {
                        funcNumber = it.readLine().trim().toInt()
                        if (funcNumber!! in functions.indices) {
                            inputStep++
                            println("Введите пределы интегрирования через пробел:")
                        } else throw NumberFormatException()
                    } catch (e: NumberFormatException) { printError("Ошибка ввода: введите целое число в [1 ; 5].") }
                }
                1 -> {
                    try {
                        limits = it.readLine().trim().split(" ").map { n -> n.toDouble() }
                        if (limits!!.size == 2) {
                            inputStep++
                            println("Введите точность:")
                        } else throw NumberFormatException()
                    } catch (e: NumberFormatException) { printError("Ошибка ввода: введите два числа через пробел.") }
                }
                2 -> {
                    try {
                        precision = it.readLine().trim().toDouble()
                        if (precision!! in 0.000001..1.0) inputStep++ else throw NumberFormatException()
                    } catch (e: NumberFormatException) { printError("Ошибка ввода: введите число в [0.000001 ; 1].") }
                }
            }
        }
    }
    try {
        val answer = IntegralSolver.integrateByTrapezoid(functions[funcNumber!!], Limits(Pair(limits!![0], limits!![1])), precision!!)
        println("""Значение интеграла = ${answer.resValue}
```

```
                ~Количество разбиений = ${answer.blocks}
                ~Погрешность = ${answer.infelicity}""".trimMargin("~"))
    } catch (e: Exception) { println(e.message) }
}

var counter = 0
fun printError(msg: String) {
    counter++
    if (counter == 100) println("Ты чо дурак!? Дифиченто, блин!")
    System.err.println(msg)
}
```