УНИВЕРСИТЕТ ИТМО

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.

$$S_{\text{общ}} = S_1 + S_2 + \dots + S_n = \frac{y_o + y_1}{2} h_1 + \frac{y_1 + y_2}{2} h_2 + \dots + \frac{y_{n-1} + y_n}{2} h_n =$$
$$= \int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n} h_i(y_{i-1} + y_i) = I_n$$

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

## Вывод

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

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

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

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

где $k -$ 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:

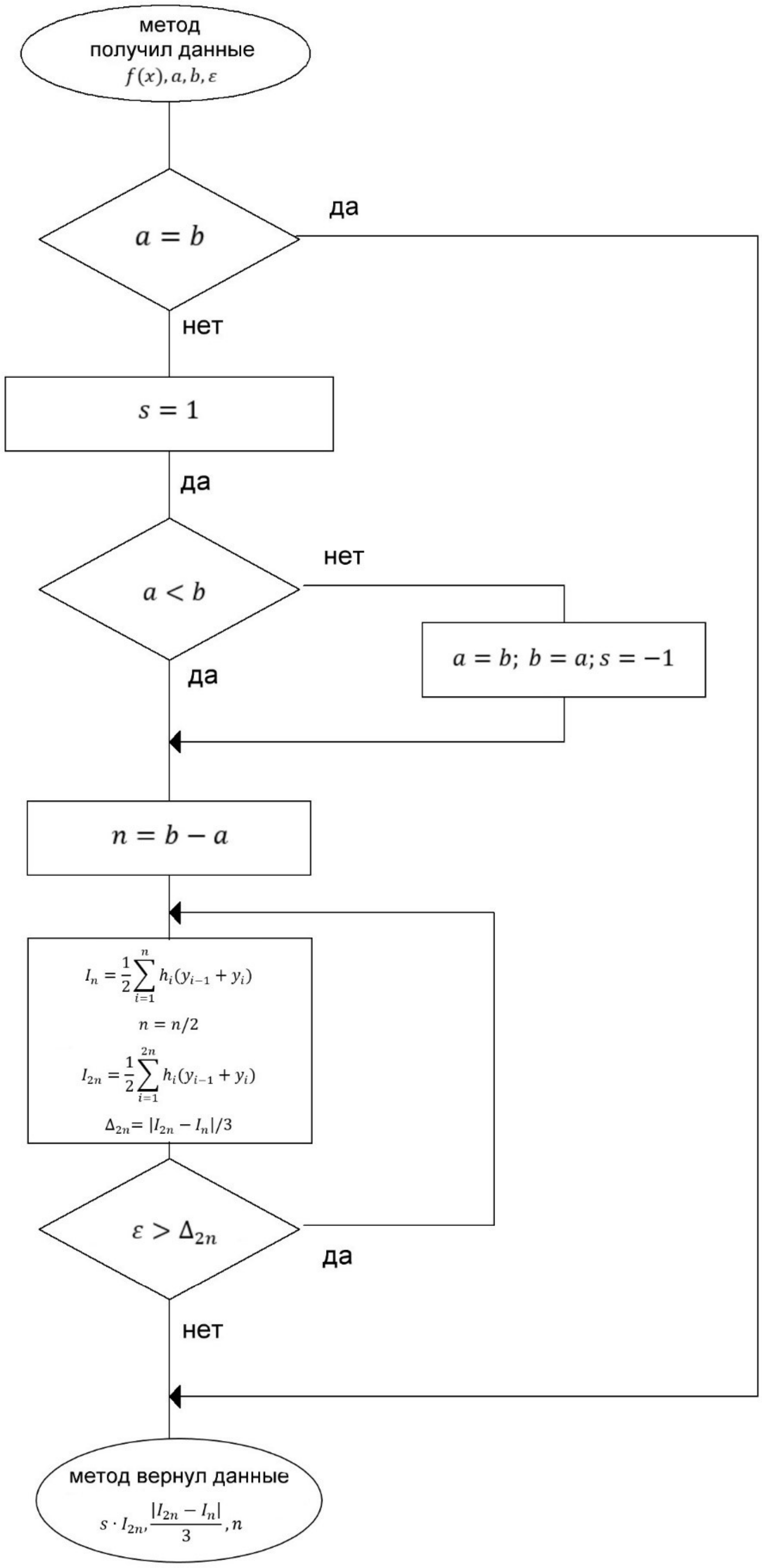
$$I - I_{\frac{h}{2}} \approx \frac{I_{\frac{h}{2}} - I}{2^k - 1}$$

# 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)

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

    /**
     * Константы, определяющие варианты решения методом прямоугольников.
     * @property LEFT метод левых прямоугольников.
     * @property CENTER метод средних прямоугольников.
     * @property RIGHT метод правых прямоугольников.
     */
    enum class RectangleMethodType { LEFT, CENTER, RIGHT }

    private interface ApproximationRule { fun findValue(step: Double, i: Int): Double }

    companion object {

        /**
         * Находит значение интеграла методом трапеций.
         * @param integral интеграл, который нужно посчитать.
         * @param precision точность вычислений.
         * @return IntegralAnswer.
         * @version 1.1
         */
        fun integrateByTrapezoid(integral: Integral, precision: Double): IntegralAnswer {
            val rule = object: ApproximationRule {
                override fun findValue(step: Double, i: Int) =
                    step * 0.5 * (integral.f.func(integral.limits.low + i * step) +
                            (integral.f.func(integral.limits.low + (i + 1) * step)))
            }
            return approximate(integral, precision, rule)
        }

        /**
         * Находит значение интеграла методом прямоугольников.
         * @param integral интеграл, который нужно посчитать.
         * @param precision точность вычислений.
         * @return IntegralAnswer.
         * @version 1.0
         */
        fun integrateByRectangle(integral: Integral, precision: Double, type: RectangleMethodType):
IntegralAnswer {
            val rule = when (type) {
                RectangleMethodType.LEFT -> object: ApproximationRule {
                    override fun findValue(step: Double, i: Int) =
                        step * integral.f.func(integral.limits.low + i * step)
                }
                RectangleMethodType.CENTER -> object: ApproximationRule {
                    override fun findValue(step: Double, i: Int) =
                        (step * integral.f.func(integral.limits.low + i * step) +
                            step * integral.f.func(integral.limits.low + (i + 1) * step)) / 2
                }
                RectangleMethodType.RIGHT -> object: ApproximationRule {
                    override fun findValue(step: Double, i: Int) =
                        step * integral.f.func(integral.limits.low + (i + 1) * step)
                }
            }
            return approximate(integral, precision, rule)
        }

        private fun approximate(integral: Integral, precision: Double, rule: ApproximationRule):
IntegralAnswer {
            fun findArea(integral: Integral, step: Double): Double {
                var area = 0.0
                for (i in 0 until ((integral.limits.high - integral.limits.low) / step).toInt()) {
                    area += rule.findValue(step, i)
                    if (area.isNaN() || area.isInfinite()) throw Exception("Функция не определена на
```

```
заданном отрезке.")
                }
                return area
            }

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

        private fun calcError(integralN: Double, integral2N: Double) = abs(integral2N - integralN) / 3
    }
}
```