Университет ИТМО

Факультет ПИиКТ

Вычислительная математика

Лабораторная работа 2

Вариант – 2 (Метод трапеций)

Выполнил: Давтян Давид Арменович

Группа: P3201

Преподаватель: Перл Ольга Вячеславовна

Санкт-Петербург, 2020 г.

Описание используемого метода

Путь есть функция $y = f(x)$, заданная на отрезке $[a, b]$. Разьём этот отрезок на n отрезков $[x_{i-1}, x_i]$, где $i \in [1, n]$ и $x_0 = a$, $x_n = b$.

На каждом из этих отрезков выберем произвольную точку ξ_i и для каждого отрезка посчитаем $s_i = f(\xi_i)(x_i - x_{i-1})$, тогда интегральной суммой будет сумма всех s: $\sum_{i=1}^{n} f(\xi_i)\,(x_i - x_{i-1})$.

Чтобы найти определённый интеграл, нужно найти предел суммы всех s при $\max\limits_{i\,=\,1,2,\dots,n}(x_i - x_{i-1}) \to 0$.

$$\int_a^b f(x)dx = \lim_{\substack{max \\ i\,=\,1,\dots,n}\Delta x \to 0} \sum_{i=1}^{n} f(\xi_i)\,\Delta x$$

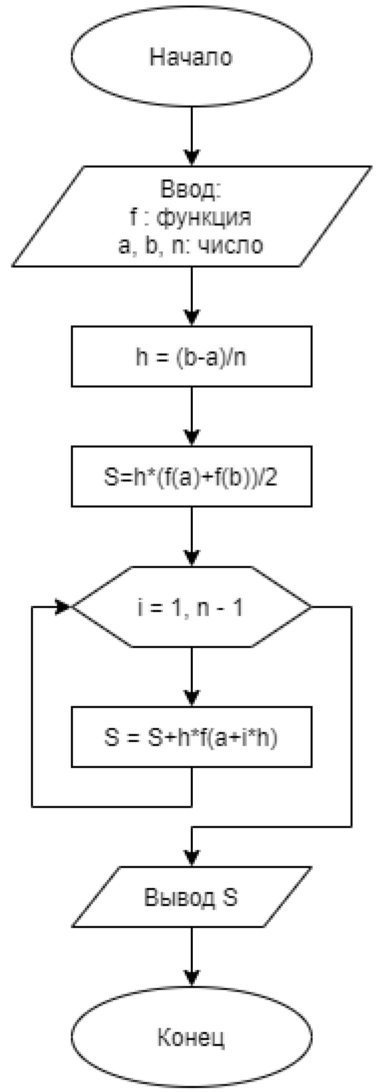
$$\Delta x = (x_i - x_{i-1})$$

Суть метода трапеций в том, что подынтегральную функцию на каждом отрезке $[x_{i-1}, x_i]$ заменяют интерполяционным многочленом первой степени: $f(x) = \varphi_i(x) = a_i x + b$.

Это значит, что график функции $y = f(x)$ будет представляться в виде ломаной, соединяющей точки (x_i, y_i). Если принять, что $\Delta x = \frac{b-a}{n} = const$, тогда площадь фигуры можно посчитать так:

$$S = \frac{1}{2}\sum_{i=1}^{n}(y_{i-1} + y_i)\Delta x$$

Блок-схема


Начало
Ввод:
f : функция
a, b, n: число
h = (b-a)/n
S=h*(f(a)+f(b))/2
i = 1, n - 1
S = S+h*f(a+i*h)
Вывод S
Конец

Листинг численного метода

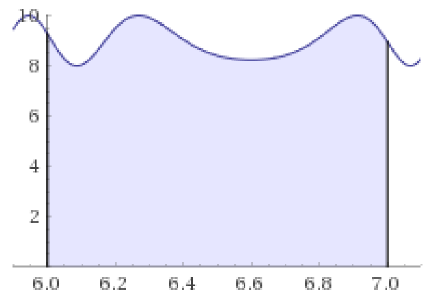
```java
package ru.david.compmath2.math.integration;

import ru.david.compmath2.math.expression.Expression;
import ru.david.compmath2.math.expression.VariableExpression;

public class Integrator {
    private static final double VERY_SMALL_NUMBER = 0.0000001;
    private static final int SPLITS_LIMIT = 100_000_000;

    public static IntegrationResult solve(
            Expression expression,
            VariableExpression variable,
            IntegrationLimits limits,
            double accuracy
    ) {
        int splits = 4;

        double previousResult;
        double result = solve(expression, variable, limits, 2);
        double error;

        do {
            previousResult = result;
            result = solve(expression, variable, limits, splits);
            error = Math.abs(previousResult - result)/3;
            splits *= 2;
        } while (
            Math.abs(error) > Math.abs(accuracy) &&
            Double.isFinite(result) &&
            splits < SPLITS_LIMIT
        );

        if (Double.isFinite(result) && splits < SPLITS_LIMIT)
            return new IntegrationResult(result, error, splits);
        return new IntegrationResult(IntegrationResult.Status.GAP);
    }
```

```java
    private static double solve(
            Expression expression,
            VariableExpression variable,
            IntegrationLimits limits,
            int splits) {
        double deltaXSize =
            (limits.getHighLimit() - limits.getLowLimit())/splits;
        double result = 0;

        for (int i = 0; i < splits; i++) {
            variable.setValue(
                limits.getLowLimit() + i*deltaXSize
            );
            double leftY = expression.value();
            if (!Double.isFinite(leftY)) {
                if (i == 0)
                    return Double.POSITIVE_INFINITY;

                variable.setValue(
                    limits.getLowLimit() +
                    i*deltaXSize +
                    VERY_SMALL_NUMBER
                );
                leftY = expression.value();
            }

            variable.setValue(limits.getLowLimit() + (i+1)*deltaXSize);
            double rightY = expression.value();
            if (!Double.isFinite(rightY)) {
                if (i == splits - 1)
                    return Double.POSITIVE_INFINITY;

                variable.setValue(
                    limits.getLowLimit() +
                    (i+1)*deltaXSize -
                    VERY_SMALL_NUMBER
                );
                rightY = expression.value();
            }

            double avgY = (leftY+rightY)/2;
            result += avgY*deltaXSize;
        }

        return result;
    }
}
```

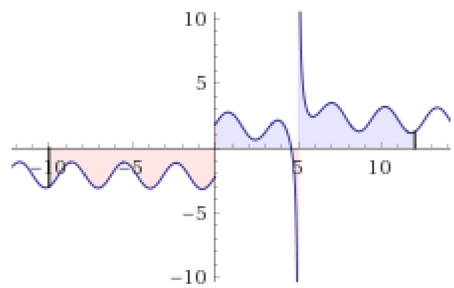
Примеры

Вывод программы / График

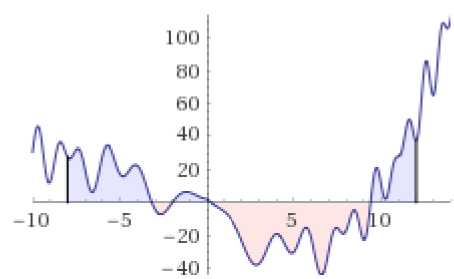
```
Введите любое выражение:
y = sin(x + x^2*cos(x)) + 9
Пределы интегрирования: 6 7
Точность: 0.00001
Результат: 8.9755387813312720
Погрешность: 0.0000057403610961
Сделано разбиений: 256
```



```
Введите любое выражение:
y = sin(x*2) + 2*abs(x)/x + 1/(x-5)
Пределы интегрирования: -10 12
Точность: 0.00001
Не получилось посчитать интеграл.
Возможно, он не сходится или
указана слишком высокая точность.
```



```
Введите любое выражение:
y = -5*abs(x) + 18*sin(x^2/4)*cosx + (x^2)/1.5 + (x^3)/20 - 6*x + 2
Пределы интегрирования: -8 12
Точность: 0.000000001
Результат: -28.1638511862336900
Погрешность: 0.0000000007469628
Сделано разбиений: 2097152
```



Вывод

В данной лабораторной работе я реализовал алгоритм вычисления интеграла с помощью метода трапеций и парсер математических выражений, позволяющий ввести любое выражение с переменной и посчитать его интеграл.

В методе трапеций подынтегральную функцию функцию на каждом $[x_{i-1}; x_i]$ заменяют интерполяционным многочленом первой степени. Интерполяция кусочно-линейная, поэтому график исходной функции представляется как ломаная, которая соединяет точки (x_i, y_i). Площадь всей фигуры состоит из суммы площадей всех трапеций. Погрешность метода трапеций выше, чем у метода средних прямоугольников, но ниже, чем у методов левых и правых прямоугольников. Это объясняется тем, что в методе средних прямоугольников используется значение в средней точке каждого отрезка, а в методе трапеций – полсумма двух крайних точек.

Метод прямоугольников в описанном выше виде неприменим в общем случае к функциям, значения которых мы знаем в конечном числе точек, так как, например, мы не всегда можем разбить отрезок интегрирования на подотрезки, серединами которых являются точки, в которых нам известно значение функции; в методе трапеций можно взять в качестве узлов интегрирования данные точки.

В методе Симпсона (парабол) разбиваем отрезок $[a;\ b]$ на четное число n равных частей с шагом, равным h. На каждом отрезке $[x_{i-1}; x_{i+1}]$ подынтегральную функцию заменяют интерполяционным многочленом второй степени. Тогда общая формула: $S = \frac{h}{3} \cdot (y_0 + 4\,(y_1 + y_3 + \ldots + y_{n-1}) + 2\,(y_2 +$

$+ y_4 + \ldots + y_{n-2}) + y_n)$. Точность метода Симпсона выше точности метода прямоугольников и трапеций.