МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине

“ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА”

Вариант «Метод прямоугольников»

Студент:
Чернова Анна Ивановна

Группа P32301

Преподаватель:

Перл Ольга Вячеславовна

Санкт-Петербург, 2023

1. Описание реализованного метода, расчетные формулы

Определенный интеграл заменяется на интегральную сумму. На каждом шаге интегрирования функция аппроксимируется полиномом нулевой степени. Площадь криволинейной трапеции приближенно заменяется площадью многоугольника, составленного из n – многоугольников. Таким образом, вычисление определённого интеграла сводится к нахождению суммы n-элементарных прямоугольников $\int_a^b f(x)dx \approx S_n = \sum_{i=1}^{n} f(\varepsilon_i)\Delta x_i$, где в качестве точек ε_i могут выбираться левые ($\varepsilon_i = x_{i-1}$) или правые ($\varepsilon_i = x_i$) границы отрезков.

Принимая за $h = \frac{b-a}{n} = const$, получаем следующие формулы для метода прямоугольника:

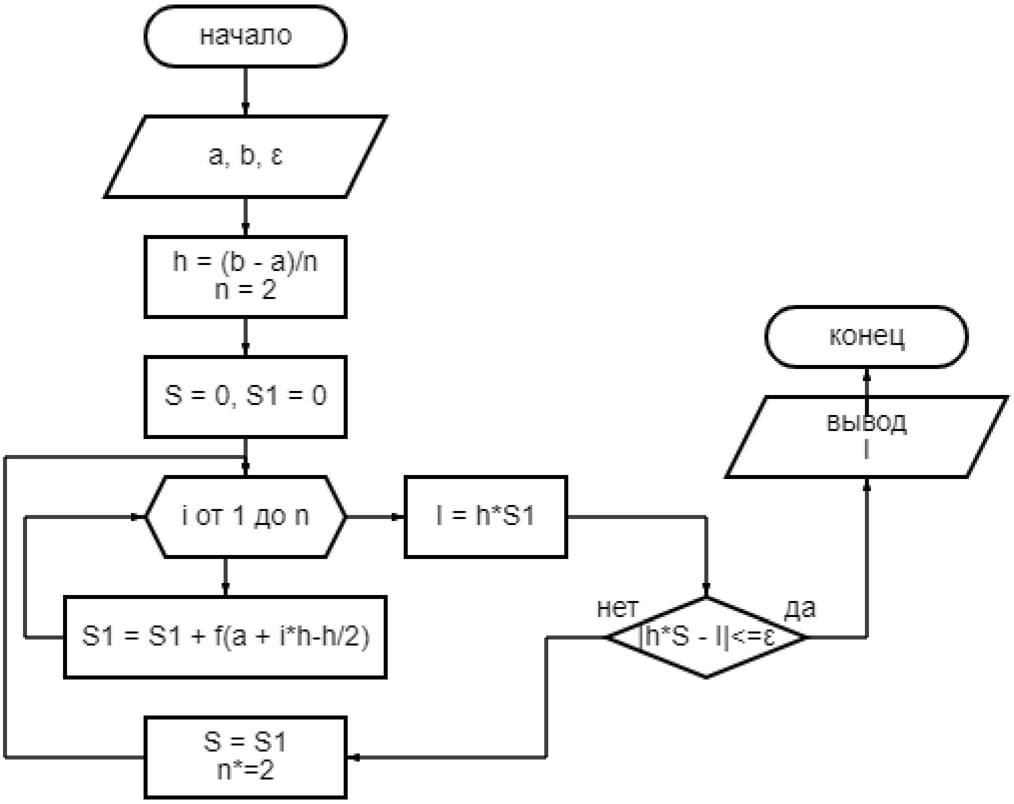
$\int_a^b f(x)dx \approx h_1 y_0 + h_2 y_1 + \cdots + h_n y_{n-1} = h \sum_{i=1}^{n} y_{i-1}$ - метод левых прямоугольников,

$\int_a^b f(x)dx \approx h_1 y_1 + h_2 y_2 + \cdots + h_n y_n = h \sum_{i=1}^{n} y_i$ - метод правых прямоугольников,

$\int_a^b f(x)dx = h \sum_{i=1}^{n} f(x_{i-1/2})$ - метод средних прямоугольники,

где $f(x_i) = y_i,\ f(a) = y_0,\ f(b) = y_n, \Delta x_i = x_i - x_{i-1} = h_i, \qquad x_{i-1/2} = \frac{x_{i-1}+x_i}{2} = x_{i-1} + \frac{h_i}{2}, i = 1, 2, \dots n.$

2. Блок-схема численного метода



3. Листинг реализованного метода

```
4. public double doMethod() {
       int n = 2;
       double previousAnswer = 0;
       double answer = Double.MAX_VALUE;
       if (data.findDiscontinuities() == null) {
           while (Math.abs(answer - previousAnswer) > data.getEpsilon()) {
               previousAnswer = answer;
               if (data.getMethodNumber() == 1) {
                   answer = getSquareLeft(data.getLowerLimit(),
   data.getUpperLimit(), n);
               }
               if (data.getMethodNumber() == 2) {
                   answer = getSquareRight(data.getLowerLimit(),
   data.getUpperLimit(), n);
               }
               if (data.getMethodNumber() == 3) {
                   answer = getSquareMiddle(data.getLowerLimit(),
   data.getUpperLimit(), n);
               }
               n *= 2;
           }
       } else {
           System.out.println("В интервал интегрирования входит точка
   разрыва первого рода. Рассчитаем интеграл слева и справа от точки
   разрыва.");
           while (Math.abs(answer - previousAnswer) > data.getEpsilon()) {
```

```
                previousAnswer = answer;
                double leftPart, rightPart;
                if (data.getMethodNumber() == 1) {
                    leftPart = getSquareLeft(data.getLowerLimit(),
data.findDiscontinuities(), n);
                    rightPart = getSquareLeft(data.findDiscontinuities(),
data.getUpperLimit(), n);

                } else if (data.getMethodNumber() == 2) {
                    leftPart = getSquareRight(data.getLowerLimit(),
data.findDiscontinuities(), n);
                    rightPart = getSquareRight(data.findDiscontinuities(),
data.getUpperLimit(), n);
                } else {
                    leftPart = getSquareMiddle(data.getLowerLimit(),
data.findDiscontinuities(), n);
                    rightPart = getSquareMiddle(data.findDiscontinuities(),
data.getUpperLimit(), n);
                }
                answer = leftPart + rightPart;
                n *= 2;
            }
        }
        return answer;
    }
public double getSquareLeft(double lowerLimit, double upperLimit, int n)
{
        double h = (upperLimit - lowerLimit) / n;
        double square = 0;
        for (int i = 0; i < n; i++) {
            double xi = lowerLimit + i * h;
            square += data.getFunction(xi);
        }
        return h * square;
}
public double getSquareRight(double lowerLimit, double upperLimit, int
n) {
        double h = (upperLimit - lowerLimit) / n;
        double square = 0;
        for (int i = 1; i <= n; i++) {
            double xi = lowerLimit + i * h;
            square += data.getFunction(xi);
        }
        return h * square;
}
public double getSquareMiddle(double lowerLimit, double upperLimit, int
n) {
        double h = (upperLimit - lowerLimit) / n;
        double square = 0;
        for (int i = 0; i < n; i++) {
            double xi = lowerLimit + i * h + h / 2;
            square += data.getFunction(xi);
        }
        return h * square;
}
```

4. Примеры результата работы программы

Выберите функцию из списка. Введите число от 1 до 4
1. y = x^3 - 2x + 3
2. y = sinx / x
3. y = 3lnx / (x - 1)
4. y = xcosx
1

Выберите метод из списка. Введите число от 1 до 3
1. метод левых прямоугольников
2. метод правых прямоугольников
3. метод средних прямоугольников
3

Введите нижний предел интегрирования
1
Введите верхний предел интегрирования
4

Введите точность
0.001

Ответ: 57.74974250793457

Выберите функцию из списка. Введите число от 1 до 4
1. y = x^3 - 2x + 3
2. y = sinx / x
3. y = 3lnx / (x - 1)
4. y = xcosx
3

Выберите метод из списка. Введите число от 1 до 3
1. метод левых прямоугольников
2. метод правых прямоугольников
3. метод средних прямоугольников
2

Введите нижний предел интегрирования
0.1
Введите верхний предел интегрирования
2

Введите точность
0.05

В интервал интегрирования входит точка разрыва первого рода. Рассчитаем интеграл слева и справа от точки разрыва.
Ответ: 6.32682344570323

Выберите функцию из списка. Введите число от 1 до 4

1. y = x^3 - 2x + 3
2. y = sinx / x
3. y = 3lnx / (x - 1)
4. y = xcosx

2

Выберите метод из списка. Введите число от 1 до 3

1. метод левых прямоугольников
2. метод правых прямоугольников
3. метод средних прямоугольников

1

Введите нижний предел интегрирования
0
Введите верхний предел интегрирования
1

Введите точность
0.00001

Одно из введенных границ не входит в область допустимых значений функции

‘

5. Вывод

Метод прямоугольников, метод трапеций и метод Симпсона схожи между собой тем, что в каждом из них подынтегральная функция заменяется приближенной функцией, однако для метода прямоугольников – это полином нулевой степени, для метода трапеций – полином первой степени, а для метода Симпсона – полином второй степени. Метод прямоугольников используется, если функция достаточно простая и имеет малые изменения. Если функция имеет более сложную форму и/или имеет большие изменения, то лучше использовать метод трапеций или метод Симпсона. Если нужна высокая точность, то лучше использовать метод Симпсона.