Национальная научно-образовательная корпорация ИТМО
Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине
«Вычислительная математика»

Вариант № 8

Выполнил:
Студент группы P3209
Ляшенко Никита Андреевич
Преподаватель:
Наумова Надежда Александровна

Санкт-Петербург, 2024

Цель работы

Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

Задание

<u>Обязательное задание (до 80 баллов)</u>

Исходные данные:

1. Пользователь выбирает функцию, интеграл которой требуется вычислить (3-5 функций), из тех, которые предлагает программа.
2. Пределы интегрирования задаются пользователем.
3. Точность вычисления задается пользователем.
4. Начальное значение числа разбиения интервала интегрирования: n=4.
5. Ввод исходных данных осуществляется с клавиатуры.

Программная реализация задачи:

1. Реализовать в программе методы по выбору пользователя:
   - Метод прямоугольников (3 модификации: левые, правые, средние)
   - Метод трапеций
   - Метод Симпсона
2. Методы должны быть оформлены в виде отдельной(ого) функции/класса.
3. Вычисление значений функции оформить в виде отдельной(ого) функции/класса.
4. Для оценки погрешности и завершения вычислительного процесса использовать правило Рунге.
5. Предусмотреть вывод результатов: значение интеграла, число разбиения интервала интегрирования для достижения требуемой точности.

Вычислительная реализация задачи:

1. Вычислить интеграл, приведенный в таблице 1, точно.
2. Вычислить интеграл по формуле Ньютона – Котеса при $n = 6$.
3. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при $n = 10$.
4. Сравнить результаты с точным значением интеграла.
5. Определить относительную погрешность вычислений для каждого метода.
6. В отчете ***отразить последовательные вычисления***.

<u>Необязательное задание (до 20 баллов)</u>

1. Установить сходимость рассматриваемых несобственных интегралов 2 рода (2-3 функции). Если интеграл - расходящийся, выводить сообщение: «Интеграл не существует».
2. Если интеграл сходящийся, реализовать в программе вычисление несобственных интегралов 2 рода (заданными численными методами).
3. Рассмотреть случаи, когда подынтегральная функция терпит бесконечный разрыв: 1) в точке a, 2) в точке b, 3) на отрезке интегрирования

Выполнение первой части

№3

$$\int_2^3 (3x^3-2x^2-7x-8)\,dx = \frac{3x^4}{4}-\frac{2x^3}{3}-\frac{7x^2}{2}-8x\Bigg|_2^3 = \frac{127}{12} = 10{,}5833$$

Ньютона-Котеса

$n=6 \quad a=2 \quad b=3$

$\boxed{b-a=1}$

$h=0{,}14$

$$\int_2^3 (3x^3-2x^2-7x-8) = \frac{41}{840}F(2)+\frac{216}{840}\cdot F(2{,}166)+$$

$$+\frac{27}{840}F(2{,}332)+\frac{272}{840}F(2{,}498)+\frac{27}{840}\cdot F(2{,}664)+\frac{216}{840}F(2{,}83)$$

$$+\frac{41}{840}F(3) = \ldots = -19{,}26777$$

$$= \frac{41\cdot(-6)+216\cdot(-2{,}05)+27\cdot(2{,}845)+272\cdot(8{,}796)+27\cdot(15{,}88)+}{840}$$

$$\frac{+216\cdot(24{,}1678)+41\cdot(34)}{} = \underline{10{,}504}$$

$n = 10$

Метод прямоугольников

$h = \frac{b-a}{n} = 0,1$

$F = h \cdot \sum_{i=1}^{n} f(x_{i-1}) = 0,1 \cdot (-4,9096 - 2,47987 + 0,296875 +$

$+ 3,4386 + 6,96337 + 10,889 + 15,2339 + 20,016 +$

$+ 25,252 + 30,9621) = 0,1 \cdot (-7,092635 + 56,54087 + 56,2141) =$

$= \underline{10,5662335}.$

Метод Симпсона

$F = \frac{0,1}{3} \cdot (-6 + 4 \cdot (\cancel{} - 3,737 \cancel{} + 1,821 + 8,875 +$

$+ 17,569 + 28,047) + 2 \cdot (-1,136 + 5,152 + 13,008 + 22,576) + 34) =$

$= \cancel{}\ \underline{10,58\overline{3}}$

Метод трапеции

$F = 0,1 \cdot \left(\frac{34 + (-6)}{2} + (-3,737) - 1,136 + 1,821 + 5,152 + 8,875 +\right.$

$\left. + 13,008 + 17,569 + 22,576 + 28,047\right) = \underline{10,6175}$

Погрешности:

1. $\frac{|10,5833 - 10,5833|}{10,5833} \cdot 100 = 0\%$ (Метод Симпсона)

2. $\frac{|10,5833 - 10,6175|}{10,5833} \cdot 100 \approx 0,323151\%$ (Метод Трапеций)

3. $\frac{|10,5833 - 10,566233|}{10,5833} \cdot 100 \approx 0,161269\%$ (Метод Прямоуг.)

$\frac{|10,5833 - 10,504|}{10,5833} \cdot 100 = 0,749294$ (Метод Ньютона-Котеса)

Выполнение второй части

https://github.com/Miqvet/4SemMath/tree/main/Lab3

Метод левых прямоугольников:

```
public class MethodLeftRect extends Method{
    @Override
    public Result compute(Function<Double, Double> function, double a, double b, double
accuracy, String modify) throws StringIndexOutOfBoundsException {
        long n = START_PARTITION * 2;

        double res1, res2;
        res1 = computeRes(function, a, b, START_PARTITION, modify);
        while (true) {
            res2 = computeRes(function, a, b, n, modify);
            if (Math.abs(res2 - res1) < accuracy)
                break;
            n *= 2;
```

```
            res1 = res2;
        }

        return new Result(res2, n);
    }

    @Override
    double computeRes(Function<Double, Double> function, double a, double b, long n,
String modify) throws StringIndexOutOfBoundsException {
        double x, h, res;
        res = 0;
        h = (b - a) / n;
        for (int i = 0; i < n; i++) {
            x = a + h * i;
            res += h * function.apply(x);
        }
        if(Math.abs(res)>60000){
            throw new StringIndexOutOfBoundsException();
        }
        return res;
    }
    @Override
    public String toString() {
        return "Метод левых прямоугольник";
    }
}
```

Метод центральных прямоугольников:

```
public class MethodCenterRect extends Method{
    @Override
    public Result compute(Function<Double, Double> function, double a, double b, double
accuracy, String modify) throws StringIndexOutOfBoundsException {
        long n = START_PARTITION * 2;

        double res1, res2;
        res1 = computeRes(function, a, b, START_PARTITION, modify);
        while (true) {
            res2 = computeRes(function, a, b, n, modify);
            if (Math.abs(res2 - res1) < accuracy)
                break;
            n *= 2;
            res1 = res2;
        }

        return new Result(res2, n);
    }

    @Override
    double computeRes(Function<Double, Double> function, double a, double b, long n,
String modify) throws StringIndexOutOfBoundsException {
        double x, h, res;
        res = 0;
        h = (b - a) / n;
        for (int i = 0; i < n; i++) {
            x = a + h * i;
            res += h * function.apply(x + h / 2);
        }
        if(Math.abs(res)>60000){
            throw new StringIndexOutOfBoundsException();
        }
        return res;
    }
    @Override
    public String toString() {
        return "Метод центральных прямоугольников";
    }
}
```

Метод правых прямоугольников:

```java
public class MethodRightRect extends Method{
    @Override
    public Result compute(Function<Double, Double> function, double a, double b, double
accuracy, String modify)throws StringIndexOutOfBoundsException  {
        long n = START_PARTITION * 2;

        double res1, res2;
        res1 = computeRes(function, a, b, START_PARTITION, modify);
        while (true) {
            res2 = computeRes(function, a, b, n, modify);
            if (Math.abs(res2 - res1) < accuracy)
                break;
            n *= 2;
            res1 = res2;
        }

        return new Result(res2, n);
    }

    @Override
    double computeRes(Function<Double, Double> function, double a, double b, long n,
String modify) throws StringIndexOutOfBoundsException {
        double x, h, res;
        res = 0;
        h = (b - a) / n;
        for (int i = 0; i < n; i++) {
            x = a + h * i;
            res += h * function.apply(x + h);
            }
        if(Math.abs(res)>60000){
            throw new StringIndexOutOfBoundsException();
        }
        return res;
    }
    @Override
    public String toString() {
        return "Метод правых прямоугольников";
    }
}
```

Метод трапеций:

```java
public class MethodTrapezoids extends Method{
    @Override
    public Result compute(Function<Double, Double> function, double a, double b, double
accuracy, String modify) throws StringIndexOutOfBoundsException {
        long partition2 = START_PARTITION * 2;

        double res1, res2;
        res1 = computeRes(function, a, b, START_PARTITION, modify);
        while (true) {
            res2 = computeRes(function, a, b, partition2, modify);
            if (Math.abs(res2 - res1) < accuracy)
                break;
            partition2 *= 2;
            res1 = res2;
        }

        return new Result(res2, partition2);
    }

    @Override
    double computeRes(Function<Double, Double> function, double a, double b, long n,
String modify) throws StringIndexOutOfBoundsException {
        double x, h, res;
        res = 0;
        h = (b - a) / n;
        for (int i = 0; i < n; i++) {
            x = a + h * i;
```

```
            res += h * (function.apply(x) + function.apply(x + h));
        }
        if(Math.abs(0.5 * res)>60000){
            throw new StringIndexOutOfBoundsException();
        }
        return 0.5 * res;
    }
    @Override
    public String toString(){
        return "Метод трапеций";
    }
}
```

Метод Симпсона:

```
public class MethodSimpson extends Method{
    @Override
    public Result compute(Function<Double, Double> function, double a, double b, double
accuracy, String modify) throws StringIndexOutOfBoundsException {
        long partition2 = START_PARTITION * 2;

        double res1 = computeRes(function, a, b, START_PARTITION, modify);
        double res2;
        while (true) {
            res2 = computeRes(function, a, b, partition2, modify);
            if (Math.abs(res2 - res1) < accuracy)
                break;
            partition2 *= 2;
            res1 = res2;
        }

        return new Result(res2, partition2);
    }

    @Override
    double computeRes(Function<Double, Double> function, double a, double b, long n,
String modify) throws StringIndexOutOfBoundsException {
        double x, res, h;
        res = 0;
        h = (b - a) / n;
        x = a;
        res += function.apply(x);
        for (int i = 1; i < n; i += 2) {
            x = a + h * i;
            res += 4 * function.apply(x);
        }
        for (int i = 2; i < n; i += 2) {
            x = a + h * i;
            res += 2 * function.apply(x);
        }
        x = b;
        res += function.apply(x);
        if(Math.abs(h / 3 * res)>60000){
            throw new StringIndexOutOfBoundsException();
        }
        return h / 3 * res;
    }
    @Override
    public String toString(){
        return "Метод Симпсона";
    }
}
```

Вывод программы:

Список доступных функций:

1 --> x^2

2 --> (x^4)/10 + (x^2)/5 - 7

3 --> 1/(1-2x)^(1/3)
4 --> (Math.cos(x))/(Math.cbrt(Math.pow(x,2)))
5 --> 1/(x-1)^2
6 --> 1/x
7 --> x+2

Введите номер функции: 6
Введите левую границу интервала: -1
Введите правую границу интервала: 1.5
Введите точность: 0.01

[Метод левых прямоугольник, 0,41, 16]
[Метод центральных прямоугольников, 0,41, 8]
[Метод правых прямоугольников, 0,40, 8]
[Метод Симпсона, 0,41, 8]
[Метод трапеций, 0,41, 8]

Рабочие формулы
Для левых и правых прямоугольников:

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} y_{i-1}$$

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} y_{i}$$

Для центральных прямоугольников:

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

Для метода Симпсона:

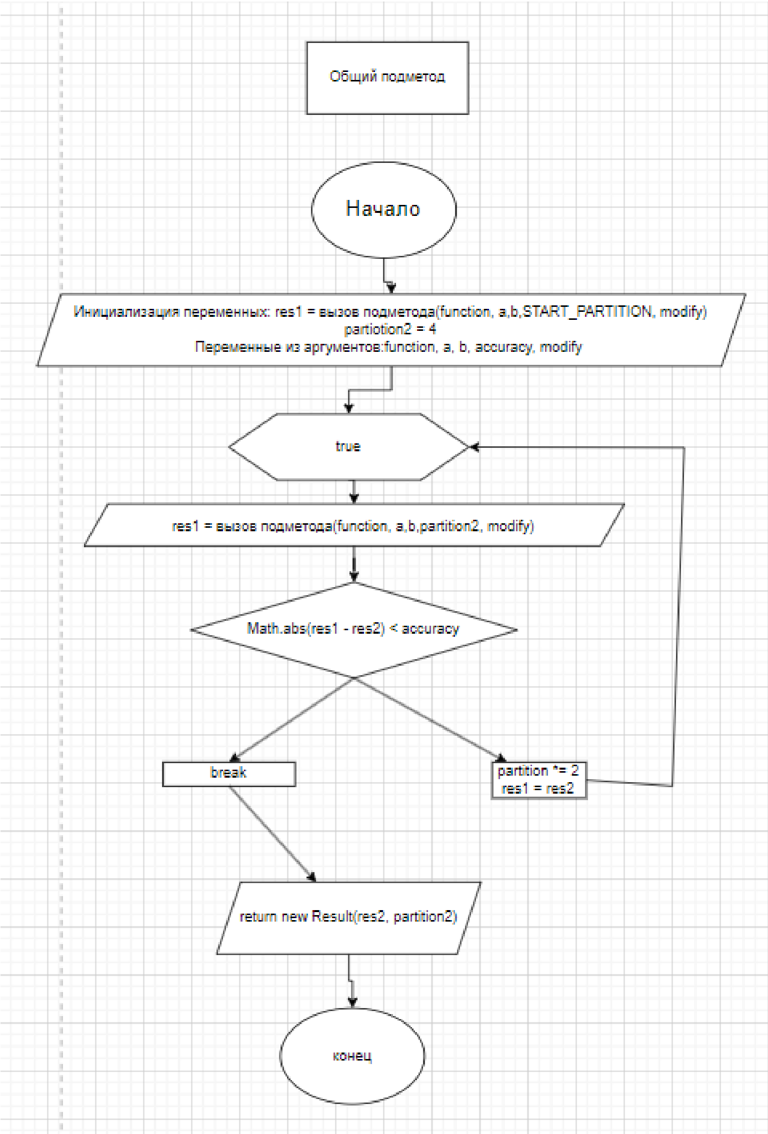
$$\int_a^b f(x) = \frac{h}{3}\left[(y_0 + 4(y_1 + y_3 + \cdots + y_{n-1}) + 2(y_2 + y_4 + \cdots + y_{n-2}) + y_n)\right]$$

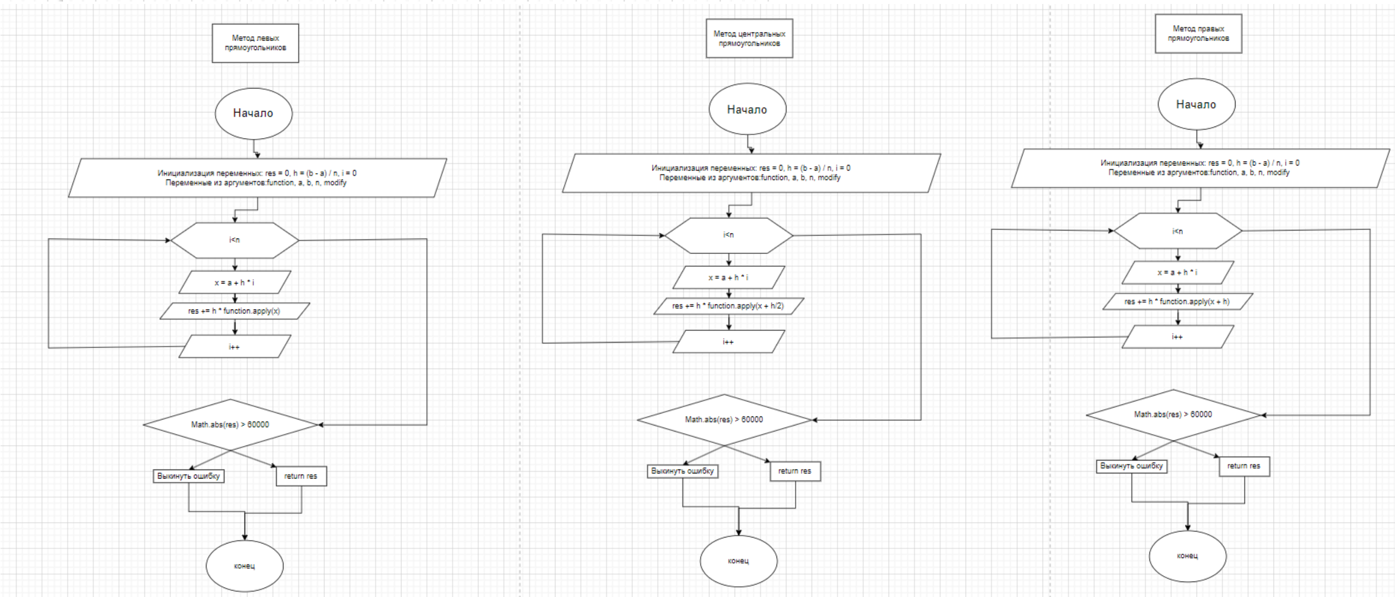
Для метода трапеций:

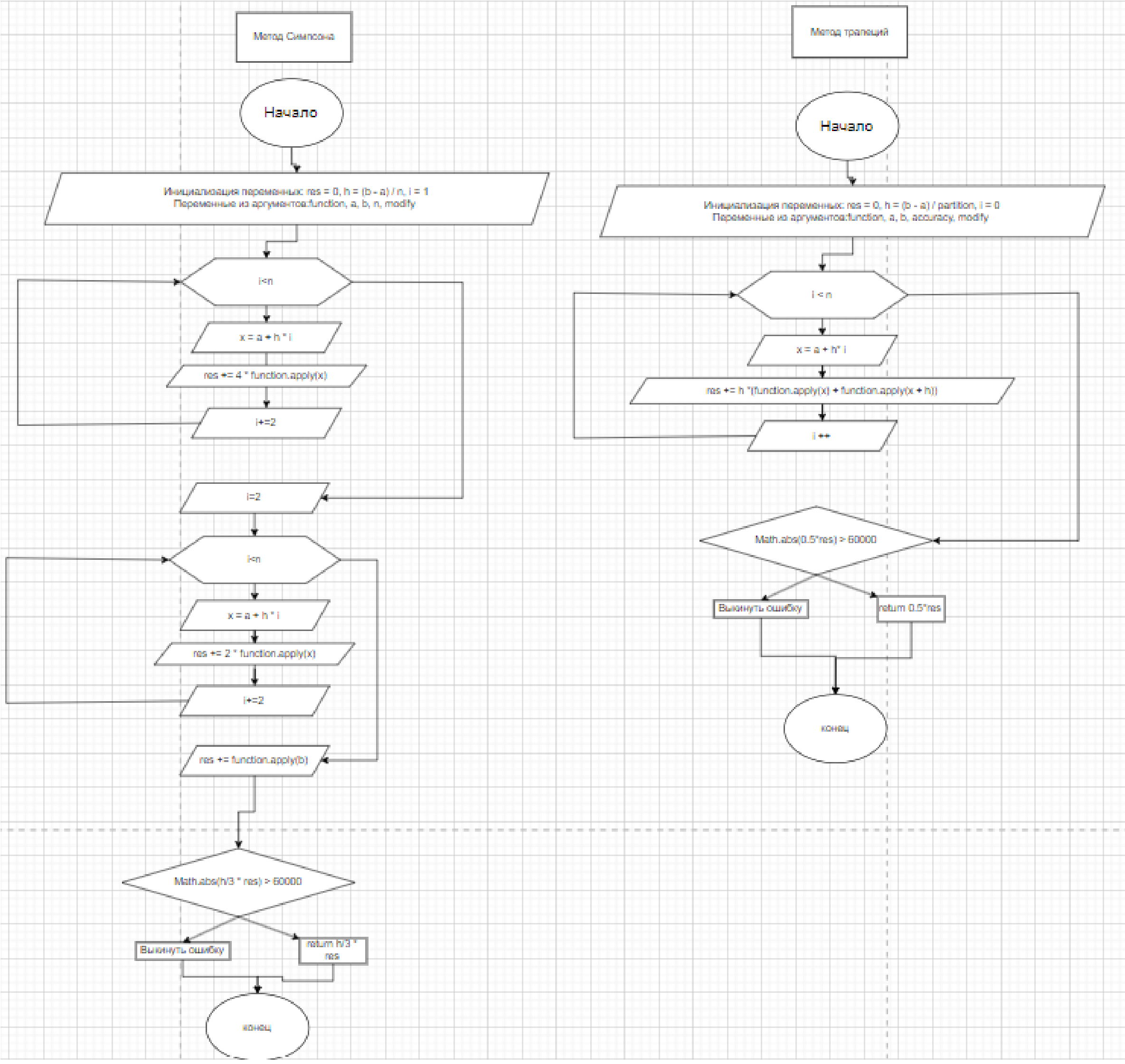
$$\int_a^b f(x)dx = h\cdot\left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i\right)$$

или $$\int_a^b f(x)dx = \frac{h}{2}\cdot\left(y_0 + y_n + 2\sum_{i=1}^{n-1} y_i\right)$$

Блок схемы для Методов:


Общий подметод
Начало
Инициализация переменных: res1 = вызов подметода(function, a,b,START_PARTITION, modify)
partiotion2 = 4
Переменные из аргументов:function, a, b, accuracy, modify
true
res1 = вызов подметода(function, a,b,partition2, modify)
Math.abs(res1 - res2) < accuracy
break
partition *= 2
res1 = res2
return new Result(res2, partition2)
конец


Метод левых прямоугольников
Начало
Инициализация переменных: res = 0, h = (b - a) / n, i = 0
Переменные из аргументов:function, a, b, n, modify
i<n
x = a + h * i
res += h * function.apply(x)
i++
Math.abs(res) > 60000
Выкинуть ошибку
return res
конец
Метод центральных прямоугольников
Начало
Инициализация переменных: res = 0, h = (b - a) / n, i = 0
Переменные из аргументов:function, a, b, n, modify
i<n
x = a + h * i
res += h * function.apply(x + h/2)
i++
Math.abs(res) > 60000
Выкинуть ошибку
return res
конец
Метод правых прямоугольников
Начало
Инициализация переменных: res = 0, h = (b - a) / n, i = 0
Переменные из аргументов:function, a, b, n, modify
i<n
x = a + h * i
res += h * function.apply(x + h)
i++
Math.abs(res) > 60000
Выкинуть ошибку
return res
конец



Вывод

В ходе лабораторной работы я познакомился с численными метода для вычисления интегралов, реализовал их на языке JAVA.