

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

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине
«Вычислительная математика»

Вариант № 8

Выполнил:
Студент группы Р3210
Мальков Павел Александрович
Преподаватель:
Машина Екатерина Алексеевна

Санкт-Петербург, 2024

Цель работы

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

Задание

Обязательное задание (до 80 баллов)

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*.

Необязательное задание (до 20 баллов)

1. Установить сходимость рассматриваемых несобственных интегралов 2 рода (2-3 функции). Если интеграл - расходящийся, вывести 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

$$\int_2^3 (3x^3 - 2x^2 - 7x - 8) dx = \left. \frac{3x^4}{4} - \frac{2x^3}{3} - \frac{7x^2}{2} - 8x \right|_2^3 = \frac{127}{12} = 10,58\bar{3}$$

Ньютона-Котеса
 $n=6$ $a=2$ $b=3$
 $h=0,14$ $b-a=1$

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

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

$$+ \frac{41}{840} f(3) = \frac{41 \cdot 19,58\bar{3}}{840} + \frac{216 \cdot 19,801}{840} + \frac{27 \cdot 19,885}{840} + \frac{272 \cdot 19,977}{840} + \frac{27 \cdot 20,069}{840} + \frac{216 \cdot 20,167}{840} + \frac{41 \cdot 20,269}{840}$$

$$= \frac{41 \cdot (19,58\bar{3} + 20,269)}{840} + \frac{216 \cdot (19,801 + 20,167)}{840} + \frac{27 \cdot (19,885 + 20,069)}{840} + \frac{272 \cdot (19,977 + 20,269)}{840}$$

$$= \frac{41 \cdot 39,852}{840} + \frac{216 \cdot 39,968}{840} + \frac{27 \cdot 39,954}{840} + \frac{272 \cdot 40,246}{840}$$

$$= \frac{1633,932}{840} + \frac{8633,088}{840} + \frac{1078,218}{840} + \frac{11046,912}{840} = \frac{22592,15}{840} = 26,9073$$

$n = 10$
Метод левых прямоугольников

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

$$F = h \cdot \sum_{i=1}^n f(x_{i-1}) = 0,1 \cdot (-4,9096 + -2,47987 + 0,296825 +$$

$$+ 3,4386 + 6,46337 + 10,889 + 15,2339 + 20,016 +$$

$$+ 25,252 + 30,9621) = 0,1 \cdot (-7,092635 + 56,54087 + 56,2191) =$$

$$= \underline{10,5662335}$$

Метод Симпсона

$$F = \frac{0,1}{3} \cdot (-6 + 4 \cdot (-2,47987 + 0,296825 + 3,4386 + 6,46337 + 10,889 + 15,2339 + 20,016 + 25,252 + 30,9621) +$$

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

$$= \underline{10,583}$$

Метод трапеции

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

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

Точности:

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

$$\frac{|10,5833 - 10,504|}{10,5833} \cdot 100 = 0,749294\%$$
 (Метод Косинуса-Бисекции)

Выполнение второй части

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:

```

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:

```

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 --> $(\text{Math.cos}(x))/(\text{Math.cbrt}(\text{Math.pow}(x,2)))$
 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:

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

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

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

Вывод

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