Университет ИТМО

Факультет ФПИ и КТ

Лабораторная работа №3
“Численное интегрирование”

По вычислительной математике

Вариант 8

Выполнил: Рогачев М. С.

Группа: P32082

Преподаватель: Машина Е. А.

Санкт-Петербург

2023

Цель:

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

Задание:

Вычислительная реализация задачи:

Вычислительная реализация задачи:

1. Вычислить интеграл, приведенный в таблице 1, точно.
2. Вычислить интеграл по формуле Ньютона – Котеса при $n = 5$.
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***.

1. Точное значение:

Вычислительная часть лабораторной работы №3

$$\int_2^3 (3x^3-2x^2-7x-8)\,dx = \left(\frac{3x^4}{4}-\frac{2x^3}{3}-\frac{7x^2}{2}-8x\right)\Big|_2^3 = -0{,}75+15\frac{1}{3} =$$

$= 10{,}58(3)$

Используем формулу Ньютона - Котеса ($n=5$)

$$\int_a^b f(x)\,dx \approx \sum_{j=1}^{N} c_j f(x_j) = \frac{n\cdot h}{C_n}\sum_{j=1}\sum_{i=0} c_{in} f(x_i)$$

n	C_n	C_{0n}	C_{1n}	C_{2n}	C_{3n}	C_{4n}	C_{5n}
0	0,5	0,5					
1							
2							
3							
4							
5							

C_5^0	C_5^1	C_5^2	C_5^3	C_5^4	C_5^5
$\frac{19}{288}$	$\frac{75}{288}$	$\frac{50}{288}$	$\frac{50}{288}$	$\frac{75}{288}$	$\frac{19}{288}$

$f(2)$	$f(2,2)$	$f(2,4)$	$f(2,6)$	$f(2,8)$	$f(3)$
-6	-1,136	5,152	13,008	22,576	34

$$\int_2^3 (3x^3-2x^2-7x-8)\,dx \approx C_5^0\cdot f(2) + C_5^1\cdot f(2,2) + C_5^2\cdot f(2,4) + C_5^3\cdot f(2,6) +$$
$$+ C_5^4\cdot f(2,8) + C_5^5\cdot f(3) = 10{,}583$$

Метод средних прямоугольников

$h=0{,}1$

x	2,05	2,15	2,25	2,35	2,45	2,55	2,65	2,75	2,85	2,95
y	-4,91	-2,1799	0,297	3,439	6,963	10,889	15,234	20,016	25,252	30,962

$$\int_2^3 f(x)dx = 10,5662$$

Метод трапеций и метод Симпсона

$h = 0.1$

x	2	2.1	2.2	2,3	2.4	2,5	2. 6	2,7	2,8	2,9	3
y	-6	-3,737	-1,136	1,821	5.152	8,875	13,008	17,569	22,576	28,047	34

М. трапеций: $\int_2^3 f(x)dx \approx \frac{h}{2}\left(y_0 + y_{10} + 2\sum_{i=1}^{9} y_i\right) = 10,6175$

М. Симпсона: $\int_2^3 f(x)dx \approx \frac{h}{3}\left(y_0 + y_{10} + 4(y_1 + y_3 + y_5 + y_7 + y_9) + 2(y_2 + y_4 + y_6 + y_8)\right)$

$= 10,9268$

Относительные погрешности

Ф-ла Ньютона-Котеса 0%

Метод ср. прямоугольников ≈ 0,158%

Метод трапеций: ≈ 0,326 %

Метод Симпсона ≈ 3,24 %

Программная реализация задачи:

Метод левых прямоугольников

```java
public class LeftRectangleMethod {

    public void method1(double sa, double sb, double eps, int num) {
        int n = 4;
        double x, x1 = solve(sa, sb, n, num);

        do {
            x = x1;
            n *= 2;
            x1 = solve(sa, sb, n, num);
        } while (Math.abs(x1 - x) / 3.0 > eps);

        System.out.println("I = " + x1 + "\nn = " + n + "\nПогрешность:
" + Math.abs(x1 - x) / 3.0);
    }

    public double solve(double sa, double sb, int n, int num) {
        double h = (sb - sa) / n, ans = 0;

        for (double i = sa; i < sb; i += h) {
            ans += Function.getFunction(i, num) * h;
        }

        return ans;
    }
}
```

Метод средних прямоугольников

```java
public class MidRectangleMethod {
    public void method2(double sa, double sb, double eps, int num) {
        int n = 4;
        double x, x1 = solve(sa, sb, n, num);

        do {
            x = x1;
            n *= 2;
            x1 = solve(sa, sb, n, num);
        } while (Math.abs(x1 - x) / 3.0 > eps);

        System.out.println("I = " + x1 + "\nn = " + n + "\nПогрешность:
" + Math.abs(x1 - x) / 3.0);
    }

    public double solve(double sa, double sb, int n, int num) {
        double h = (sb - sa) / n, ans = 0;

        for (double i = sa; i < sb; i += h) {
            ans += Function.getFunction(i + h / 2.0, num) * h;
        }

        return ans;
    }
}
```

Метод правых прямоугольников

```java
public class RightRectangleMethod {
    public void method3(double sa, double sb, double eps, int num) {
        int n = 4;
        double x, x1 = solve(sa, sb, n, num);

        do {
            x = x1;
            n *= 2;
            x1 = solve(sa, sb, n, num);
        } while (Math.abs(x1 - x) / 3.0 > eps);

        System.out.println("I = " + x1 + "\nn = " + n + "\nПогрешность: 
" + Math.abs(x1 - x) / 3.0);
    }

    public double solve(double sa, double sb, int n, int num) {
        double h = (sb - sa) / n, ans = 0;

        for (double i = sa; i < sb; i += h) {
            ans += Function.getFunction(i + h, num) * h;
        }

        return ans;
    }
}
```

Метод трапеций

```java
public class TrapesyMethod {

    public void method4(double sa, double sb, double eps, int num) {
        int n = 4;
        double x, x1 = solve(sa, sb, n, num);

        do {
            x = x1;
            n *= 2;
            x1 = solve(sa, sb, n, num);
        } while (Math.abs(x1 - x) / 3.0 > eps);

        System.out.println("I = " + x1 + "\nn = " + n + "\nПогрешность: 
" + Math.abs(x1 - x) / 3.0);
    }

    public double solve(double sa, double sb, int n, int num) {
        double h = (sb - sa) / n, ans = Function.getFunction(sa, num) + 
Function.getFunction(sb, num);

        for (double i = sa + h; i < sb; i += h) {
            ans += Function.getFunction(i, num) * 2;
        }

        return ans * h / 2.0;
```

```
    }
}
```

Метод Симпсона

```
public class SimpsonMethod {

    public void method5(double sa, double sb, double eps, int num) {
        int n = 4;
        double x, x1 = solve(sa, sb, n, num);

        do {
            x = x1;
            n *= 2;
            x1 = solve(sa, sb, n, num);
        } while (Math.abs(x1 - x) / 15.0 > eps);

        System.out.println("I = " + x1 + "\nn = " + n + "\nПогрешность:
" + Math.abs(x1 - x) / 15.0);
    }

    public double solve(double sa, double sb, int n, int num) {
        double h = (sb - sa) / n, ans = Function.getFunction(sa, num) +
Function.getFunction(sb, num);
        boolean flag = true;

        for (double i = sa + h; i < sb; i += h) {
            if (flag) {
                ans += Function.getFunction(i, num) * 4.0;
            } else {
                ans += Function.getFunction(i, num) * 2.0;
            }
            flag = !flag;
        }

        return ans * h / 3.0;
    }
}
```

Пример работы:

```
Введите номер метода:
1 - Метод прямоугольников (левый)
2 - Метод прямоугольников (средний)
3 - Метод прямоугольников (правый)
4 - Метод трапеций
5 - Метод Симпсона
5
Введите номер функции:
1 - y = -x^3 - 2 * x^2 + 3 * x + 23
2 - y = x^2 + x + 1
3 - y = 3 * x
4 - y = 4
1
Введите a:
2
Введите b:
4
Введите погрешность [0,0001; 1]:
0,0001
I = -33.333333333333336
n = 8
Погрешность: 0.0
```

Вывод:

в ходе лабораторной работы я научился реализовывать в программе методы для вычисления интегралов. Выполняя вычислительную часть лабораторной работы, я понял, что метод Симпсона наиболее точен и при этом не так сложен в вычислениях, как формула Ньютона – Котеса.