Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №3**

по «Вычислительной математике»

**Численное интегрирование**

Вариант 4

Выполнил:

Студент группы P3213

Ваховский П.А.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2022

**Цель работы**: найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

**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, исходя из варианта:

* Метод прямоугольников (3 модификации: левые, правые, средние)
* Метод трапеций
* Метод Симпсона

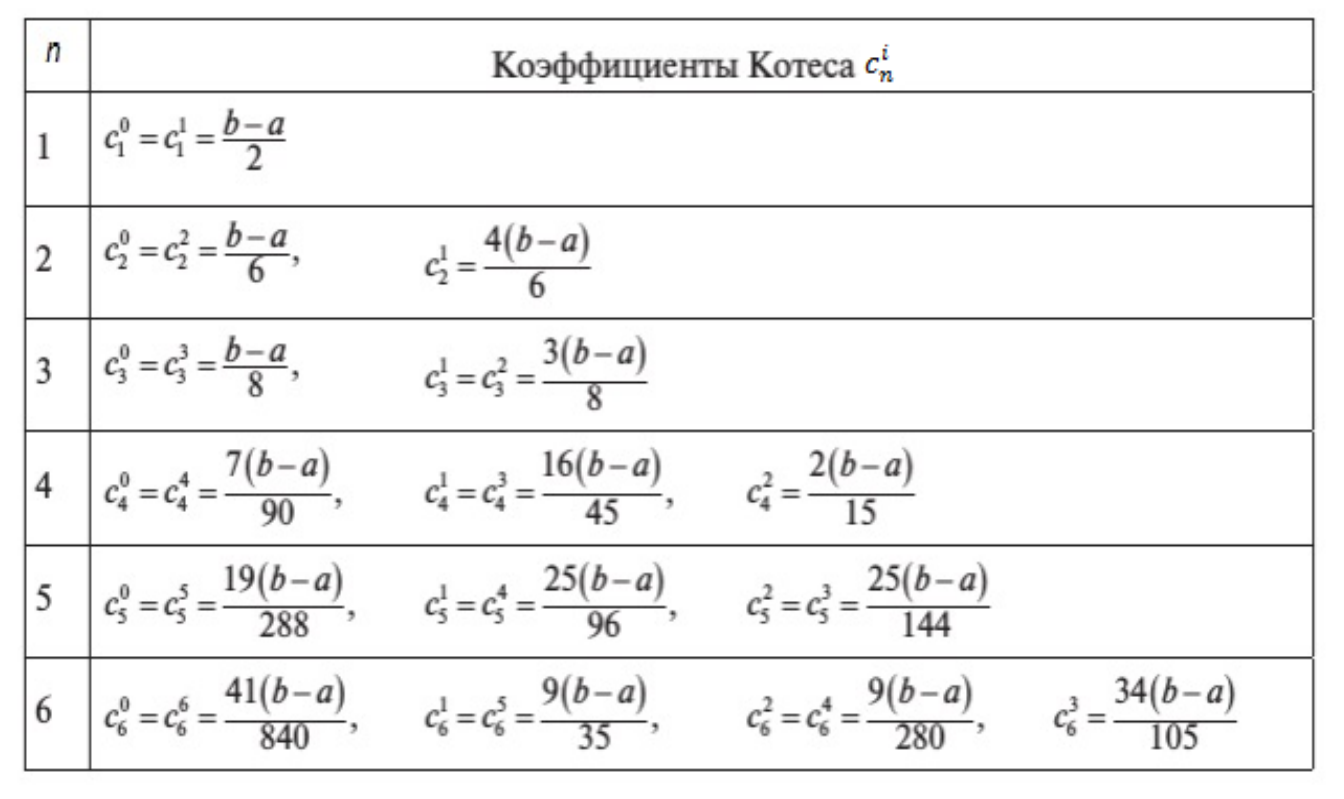
1. Методы должны быть оформлены в виде отдельной(ого) функции/класса.
2. Вычисление значений функции оформить в виде отдельной(ого) функции/класса.
3. Для оценки погрешности и завершения вычислительного процесса использовать правило Рунге.
4. Предусмотреть вывод результатов: значение интеграла, число разбиения интервала интегрирования для достижения требуемой точности.

**Вычислительная реализация задачи:**

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

**Вычислительная реализация**

Формула Ньютона-Котеса



N = 6 => h = (-1-(-3))/6 = 1/3

Ошибка составила 0.001

Относительная погрешность: 0.00002%

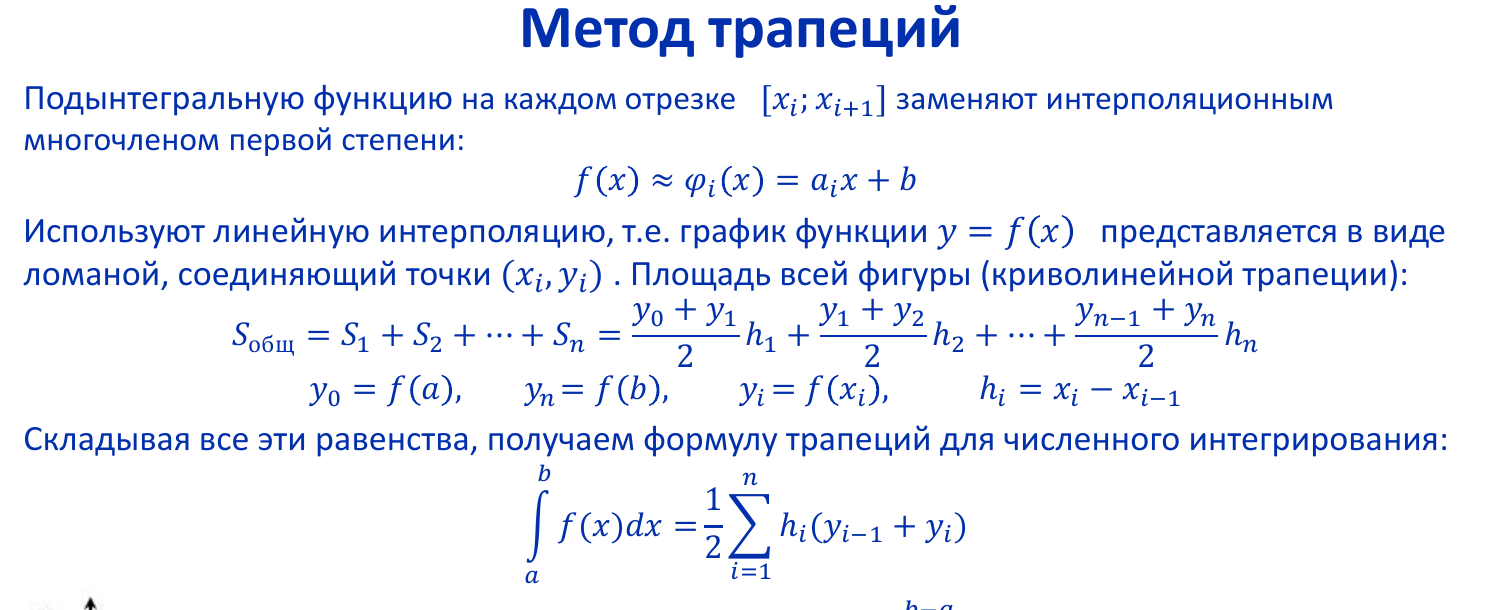
Формула средних прямоугольников:



Ошибка составила 0,148

Относительная погрешность: 0,43%

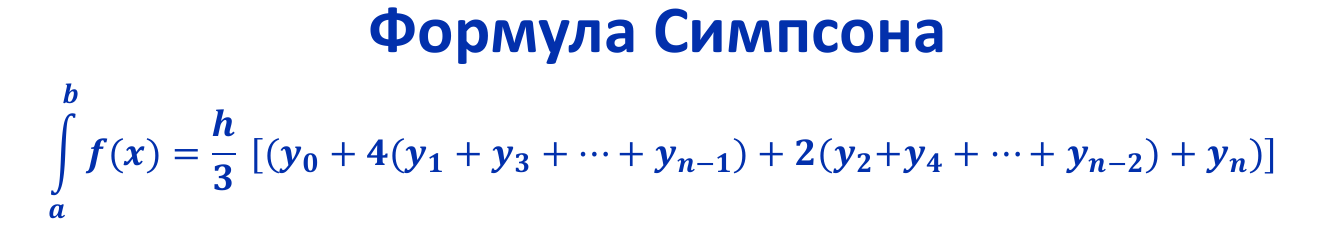
Формула трапеций:



Ошибка составила 5,47

Относительная погрешность: 15,78%

Формула Симпсона:



Ошибка составила 0.001

Относительная погрешность: 0.00001%

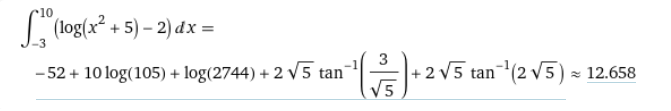
**Листинг класса RectangleMethod:**

public class RectangleMethod extends SolveMethod{  
 public RectangleMethod(Function function, double a, double b, double eps){  
 super(function, a, b, eps);  
 }  
  
 private double calcI(RectangleMethodType type){  
 switch (type){  
 case *LEFT*:  
 double lastFx = function.calcFunction(X.get(X.size()-1));  
 return h\*(X.stream().mapToDouble(x->function.calcFunction(x)).sum() - lastFx);  
 case *MID*:  
 double sum = 0d;  
 for(int i = 1; i < X.size(); i++){  
 sum += function.calcFunction((X.get(i) + X.get(i-1))/2);  
 }  
 return h\*sum;  
 case *RIGHT*:  
 double firstFx = function.calcFunction(X.get(0));  
 return h\*(X.stream().mapToDouble(x->function.calcFunction(x)).sum() - firstFx);  
 default:  
 return Double.*NaN*;  
 }  
  
 }  
 public double solution(RectangleMethodType type) {  
 double prevI = calcI(type), curI = Double.*NaN*;  
 while(n < Integer.*MAX\_VALUE* / 1000){  
 n \*= 2;  
 updateH();  
 updateX();  
 curI = calcI(type);  
 if(k\*Math.*abs*(curI-prevI) < eps)  
 break;  
 prevI = curI;  
 }  
 System.*out*.printf("Определенный интеграл вычислен при разбиении на n = %d равных частей\n", n);  
 System.*out*.println("Вычисленное значение: " + curI);  
 return curI;  
 }  
}

**Пример работы класса:**

Изображение выглядит как текст

Автоматически созданное описание



**Листинг класса TrapezoidMethod:**

import java.util.ArrayList;  
  
public class TrapezoidMethod extends SolveMethod{  
 ArrayList<Double> Y;  
  
 public TrapezoidMethod(Function function, double a, double b, double eps){  
 super(function, a, b, eps);  
 Y = new ArrayList<>();  
 updateY();  
 }  
  
 private void updateY(){  
 Y.clear();  
 X.forEach(x->Y.add(function.calcFunction(x)));  
 }  
  
 private double calcI(){  
 return h/2 \* (2\*Y.stream().mapToDouble(x->x).sum() - (Y.get(0) + Y.get(Y.size()-1)));  
 }  
  
 public double solution(){  
 double prevI = calcI(), curI = Double.*NaN*;  
  
 while(n < Integer.*MAX\_VALUE* / 1000){  
 n \*= 2;  
 updateH();  
 updateX();  
 updateY();  
 curI = calcI();  
 if(k\*Math.*abs*(curI-prevI) < eps)  
 break;  
 prevI = curI;  
 }  
 System.*out*.printf("Определенный интеграл вычислен при разбиении на n = %d равных частей\n", n);  
 System.*out*.println("Вычисленное значение: " + curI);  
 return curI;  
 }  
}

**Пример работы класса:**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Вывод**

В ходе данной лабораторной работы мной были изучены методы прямоугольников и трапеций для численного интегрирования. Программа пишется несложно и дает хорошую точность.