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

Изображение выглядит как текст, Шрифт, снимок экрана, письмо

Автоматически созданное описание

## Выполнение первой части

Изображение выглядит как текст, рукописный текст, бумага, документ

Автоматически созданное описание

Изображение выглядит как текст, бумага, рукописный текст, книга

Автоматически созданное описание

Изображение выглядит как текст, рукописный текст, рукописный, документ

Автоматически созданное описание

# 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:

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 --> (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:

Изображение выглядит как Шрифт, текст, рукописный текст, диаграмма

Автоматически созданное описание

Для центральных прямоугольников:

Изображение выглядит как Шрифт, текст, рукописный текст, линия

Автоматически созданное описание

Для метода Симпсона:

Изображение выглядит как Шрифт, рукописный текст, текст, линия

Автоматически созданное описание

Для метода трапеций:

Изображение выглядит как текст, рукописный текст, Шрифт, диаграмма

Автоматически созданное описание

Блок схемы для Методов:

Изображение выглядит как текст, диаграмма, линия, чек

Автоматически созданное описание

Изображение выглядит как текст, диаграмма, линия, План

Автоматически созданное описание

Изображение выглядит как диаграмма, зарисовка, текст, План

Автоматически созданное описание

# Вывод

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