|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика и системы управления» |

|  |  |
| --- | --- |
| КАФЕДРА | «Теоретическая информатика и компьютерные технологии» |

**Отчет по лабораторной работе № 2**

**«Приближенное значение определенного интеграла»**

***по курсу***

***«Численные методы»***

Выполнил:

Студент группы ИУ9-62Б

Караник А.А.

Преподаватель:

Домрачева А.Б.

1. *г.*

## Цель работы

Целью данной работы является реализация программы для нахождения приближенного значения определенного интеграла, используя методы прямоугольников, трапеций и Симпсона.

## Постановка задачи

Пусть дана функция на отрезке , где , , .

Вычислить приближенное значение определенного интеграла методами прямоугольников, трапеций и Симпсона и, используя оценку точности при , вывести соответствующие значения n, I\*(h/2), R, I\*(h/2) + R.

## Теоретические сведения

Отрезок разбивается с постоянным шагом . Обозначим .

Метод прямоугольников:

Метод трапеций:

Метод Симпсона:

где , т.е. четное.

Оценка точности:

где формулы (для методов прямоугольников и трапеций равен 2, для метода Симпсона – 4).

Если , где – абсолютная погрешность, то вычисления прекращаются.

## Реализация

/\*\*

 \* Laboratory work: 2

 \* Discipline: Numerical methods

 \* Copyright (c) 2024, Andrey Karanik

\*/

#include <iostream>

#define \_USE\_MATH\_DEFINES

#include <cmath>

#include <iomanip>

float a = 0.25f;

float b = 4.0f;

float eps = 0.001f;

float f(float x) {

    return logf(4\*x) / x;

}

float rectangleMethod(float h, int n) {

    float sum = 0;

    for (int i = 0; i < n; i++) {

        float x = a + i \* h;

        sum += f(x + h / 2.0f);

    }

    return h \* sum;

}

float simpsonMethod(float h, int n) {

    float result = 0;

    result = f(a) + f(b);

    for (int i = 1; i < n; i += 2) {

        float x1 = a + i \* h;

        float x2 = a + (i + 1) \* h;

        result += 4 \* f(x1);

        if (i <= n - 2) {

            result += 2 \* f(x2);

        }

    }

    result \*= h / 3.0f;

    return result;

}

float trapezoidMethod(float h, int n) {

    float result = 0;

    result += (f(a) + f(b)) / 2.0f;

    for (int i = 1; i < n; i++) {

        result += f(a + i \* h);

    }

    result \*= h;

    return result;

}

int main(int argc, char \*argv[]) {

    int n = 2;

    int n1 = 0;

    int n2 = 0;

    int n3 = 0;

    float h1 = 0;

    float h2 = 0;

    float h3 = 0;

    float r1 = 0;

    float r2 = 0;

    float r3 = 0;

    float integral1 = 0;

    float integral2 = 0;

    float integral3 = 0;

    bool check1 = false;

    bool check2 = false;

    bool check3 = false;

    while (true) {

        float h = (b - a) / (float)n;

        float delta = 0;

        if (!check1) {

            integral1 = rectangleMethod(h / 2.0f, 2 \* n);

            r1 = fabs(rectangleMethod(h, n) - integral1) / (4.0f - 1.0f);

            check1 = r1 <= eps;

            if (check1) {

                h1 = h;

                n1 = n;

            }

        }

        if (n % 2 == 0) {

            if (!check2) {

                integral2 = simpsonMethod(h / 2.0f, 2 \* n);

                r2 = fabs(simpsonMethod(h, n) - integral2) / (16.0f - 1.0f);

                check2 = r2 <= eps;

                if (check2) {

                    h2 = h;

                    n2 = n;

                }

            }

        }

        if (!check3) {

            integral3 = trapezoidMethod(h / 2.0f, 2 \* n);

            r3 = fabs(trapezoidMethod(h, n) - integral3) / (4.0f - 1.0f);

            check3 = r3 <= eps;

            if (check3) {

                h3 = h;

                n3 = n;

            }

        }

        if (check1 && check2 && check3) {

            break;

        }

        n++;

    }

    std::cout << "n: " << n1 << std::endl;

    std::cout << "rectangleMethod(h/2): " << integral1 << std::endl;

    std::cout << "R: " << r1 << std::endl;

    std::cout << "rectangleMethod(h/2) + R: " << integral1 + r1 << std::endl;

    std::cout << std::endl;

    std::cout << "n: " << n2 << std::endl;

    std::cout << "simpsonMethod(h/2): " << integral2 << std::endl;

    std::cout << "R: " << r2 << std::endl;

    std::cout << "simpsonMethod(h/2) + R: " << integral2 + r2 << std::endl;

    std::cout << std::endl;

    std::cout << "n: " << n3 << std::endl;

    std::cout << "trapezoidMethod(h/2): " << integral3 << std::endl;

    std::cout << "R: " << r3 << std::endl;

    std::cout << "trapezoidMethod(h/2) + R: " << integral3 + r3 << std::endl;

    return 0;

}

## Тестирование

n: 2

rectangleMethod(h/2): 3.99631

R: 0.0006965

rectangleMethod(h/2) + R: 3.99701

n: 16

simpsonMethod(h/2): 3.8417

R: 0.000961526

simpsonMethod(h/2) + R: 3.84267

n: 69

trapezoidMethod(h/2): 3.84264

R: 0.000981251

trapezoidMethod(h/2) + R: 3.84362

## Вывод

В процессе выполнения лабораторной работы была разработана программа на языке C++, предназначенная для нахождения приближенного значения определенного интеграла, используя методы прямоугольников, трапеций и Симпсона. Заметим, что самым эффективным по скорости методом является метод прямоугольников, однако он и имеет самый грубый ответ по сравнению с результатами других методов. Метод Симпсона менее эффективен, но результат вычисления является более точным, как и метод трапеций.