МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА



Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Лабораторная работа №4

«Численное интегрирование функций»

по дисциплине

Вычислительная математика

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Суркова А.С.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В.А.

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

Оглавление

[Цель 3](#_Toc69138063)

[Постановка задачи 4](#_Toc69138064)

[Теоретические сведения 5](#_Toc69138065)

[Метод средних (центральных) прямоугольников. 5](#_Toc69138066)

[Метод трапеций 6](#_Toc69138067)

[Метод Симсона 7](#_Toc69138068)

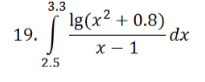
[Расчетные данные 8](#_Toc69138069)

# Цель

Закрепление знаний и умений по численному интегрированию функций.

# Постановка задачи

Вычислить интеграл по формулам центральных (средних) прямоугольников, трапеций и формуле Симпсона, при n=8 и n=20; оценить погрешность результата.



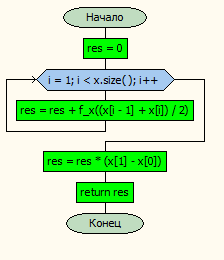
# Теоретические сведения

## Метод средних (центральных) прямоугольников.

**Метод прямоугольников-** метод интегрирования функции одной переменной, заключающийся в замене подынтегральной функции на многочлен нулевой степени, то есть константу, на каждом элементарном отрезке. Если рассмотреть график подынтегральной функции, то метод будет заключаться в приближённом вычислении площади под графиком суммированием площадей конечного числа прямоугольников, ширина которых будет определяться расстоянием между соответствующими соседними узлами интегрирования, а высота — значением подынтегральной функции в этих узлах.

Составная квадратурная формула для метода средних (центральных) прямоугольников.

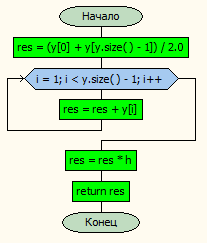
Погрешность формулы интегрирования метода средних (центральных) прямоугольников:



## Метод трапеций

**Метод трапеций** — метод численного интегрирования функции одной переменной, заключающийся в замене на каждом элементарном отрезке подынтегральной функции на многочлен первой степени, то есть линейную функцию. Площадь под графиком функции аппроксимируется прямоугольными трапециями.

Погрешность формулы интегрирования метода средних (центральных) прямоугольников:



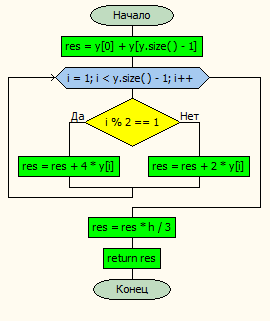
## Метод Симсона

**Формула Симпсона** (также **Ньютона-Симпсона**[) относится к приёмам численного интегрирования. Получила название в честь британского математика Томаса Симпсона (1710—1761).](https://ru.wikipedia.org/wiki/Формула_Симпсона#cite_note-1)

Суть метода заключается в приближении подынтегральной функции на отрезке {\displaystyle [a,b]} интерполяционным многочленом второй степени {\displaystyle p\_{2}(x)}, то есть приближение графика функции на отрезке параболой.

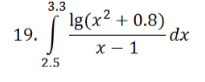
Погрешность формулы интегрирования метода Симпсона:

Погрешность формулы интегрирования метода Симпсона при невозможности ввода производной четвертого порядка:

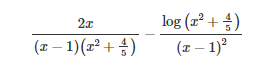


# Расчетные данные

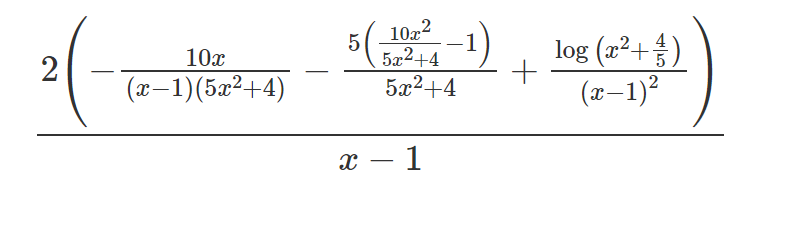
Исходная функция:



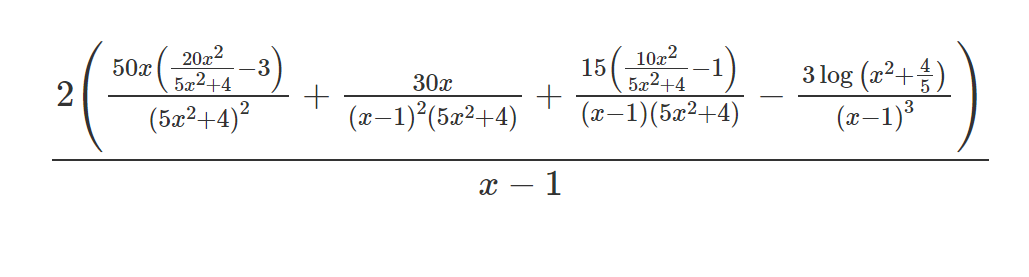
Первая производная:



Вторая производная:



Третья производная:



**При n = 8:**

|  |  |  |
| --- | --- | --- |
|  | Значение интеграла | Значение погрешности |
| Метод средних прямоугольников | 0.407905 | 0.0103256 |
| Метод трапеций | 0.408001 | 0.0206513 |
| Метод Симпсона | 0.407937 | 0.000001 |

**При n = 20:**

|  |  |  |
| --- | --- | --- |
|  | Значение интеграла | Значение погрешности |
| Метод средних прямоугольников | 0.407932 | 0.001652 |
| Метод трапеций | 0.407947 | 0.003304 |
| Метод Симпсона | 0.407937 | 0.0000001 |

Код программы

## function.h

#pragma once

#include<iostream>

#include<vector>

#include<cmath>

#include"colors.h"

using namespace std;

/\*Библиотека для работы с исходной функцией и её производными \*/

/\*Функция считывания пределов интегрирования\*/

void get\_limits(double\* lower\_limit, double\* upper\_limit) {

cout << Green << "Введите пределы интегрирования \n"

<< Yellow << "\tНижний предел ";

cin >> \*lower\_limit;

cout << "\tВерхний предел ";

cin >> \*upper\_limit;

}

/\*Функция считывания n\*/

int get\_n() {

int n;

cout << Green << "Введите n - число отрезков разбиения ";

cin >> n;

return n;

}

/\*Функция вывода меню\*/

char menu\_item\_selection() {

char c;

cout << Green << "\nВыберите, что нужно сделать :\n" << Yellow

<< "\t{1} - Найти интеграл методом центральных прямоугольников\n"

<< "\t{2} - Найти интеграл методом трапеций\n"

<< "\t{3} - Найти интеграл по формуле Симпсона\n"

<< "\t{n} - Сменить n\n"

<< "\t{q} - Завершить программу\n";

cin >> c;

return c;

}

/\*Функция нахождения значений х на каждом отрезке

\*Параметры:

\* 1)lower\_limit-нижний предел

\* 2)upper\_limit-верхний предел

\* 3)n-число отрезков

\*/

vector <double> get\_x(double lower\_limit, double upper\_limit, int n) {

vector <double> x(n + 1);

double h = (upper\_limit - lower\_limit) / n; //Шаг

x[0] = lower\_limit;

for (int i = 1; i < (n + 1); i++) {

x[i] = x[i - 1] + h;

}

return x;

}

/\*Функция нахождения значения функции в заданной точке\*/

double f\_x(double x) {

double y = (log10(x \* x + 0.8)) / (x - 1);

return y;

}

/\*Функция нахождения у на каждом отрезке

\*Параметры:

\* 1)x - вектор значений х

\*/

vector <double> get\_y(vector<double>x) {

vector <double> y(x.size());

for (int i = 0; i < (x.size()); i++) {

y[i] = f\_x(x[i]);

}

return y;

}

/\*Функция нахождения значения второй производной в заданной точке

\*/

double second\_derivative(double x) {

double temp1, temp2, temp3, res;

temp1 = -2\*x / (x - 1) \* (x \* x + 0.8);

temp2 = -((2 \* x \* x) / (x \* x + 0.8) - 1) / (x \* x + 0.8);

temp3 = log10(x \* x + 0.8) / ((x - 1) \* (x - 1));

res = 2 \* (temp1 + temp2 + temp3) / (x - 1);

return res;

}

/\*Функция нахождения значения третьей производной в заданной точке

\*/

double third\_derivative(double x) {

double temp1, temp2, temp3,temp4, res;

temp1 = 3 \* (2 \* x \* x / (x \* x + 0.8) - 1) /

((x - 1) \* (x \* x + 0.8));

temp2 = 6 \* x / ((x - 1) \* (x - 1) \* (x \* x + 0.8));

temp3 = 2 \* x \* (4 \* x / (x \* x + 0.8) - 3) /

((x \* x + 0.8) \* (x \* x + 0.8));

temp4 = -3 \* log10(x \* x + 0.8) /

((x - 1) \* (x - 1) \* (x - 1));

res = 2 \* (temp1 + temp2 + temp3+temp4) / (x - 1);

return res;

}

## Solutions.h

#pragma once

#include<iostream>

#include<vector>

#include"colors.h"

using namespace std;

/\*Библиотека, содержащая методы нахождения интеграла и погрешности этих методов\*/

/\*Функция поиска интеграла методом центральных прямоугольников

\*Параметры

\* 1)x - вектор значений х

\*/

double central\_rectangles(vector<double>x) {

double res=0;

for (int i = 1; i < x.size(); i++){

res = res + f\_x((x[i - 1] + x[i]) / 2);

}

res = res \* (x[1] - x[0]);

cout<<Blue << "Значение интеграла, вычисленное методом"

<<"центральных прямоугольников при n="

<< x.size() - 1 << " равно " << res << "\n";

return res;

}

/\*Функция поиска интеграла методом трапеций

\*Параметры

\* 1)h - длина отрезка (шаг)

\* 2)у - вектор значений у

\*/

double trapeze(double h, vector <double> y) {

double res = (y[0]+y[y.size()-1])/2.0;

for (int i = 1; i < y.size()-1; i++){

res = res + y[i];

}

res = res \* h;

cout<<Blue << "Значение интеграла, вычисленное методом"

<<" трапеций при n="<< y.size() - 1 << " равно "

<< res << "\n";

return res;

}

/\*Функция поиска интеграла методом Симпосна

\*Параметры

\* 1)h - длина отрезка (шаг)

\* 2)у - вектор значений у

\*/

double Simpson(double h, vector <double> y) {

double res=y[0]+y[y.size()-1];

for (int i = 1; i < y.size()-1; i++){

if (i % 2 == 1) {res = res + 4 \* y[i];}

else { res = res + 2 \* y[i];}

}

res = res \* h / 3;

cout<<Blue << "Значение интеграла, вычисленное методом"

<<" Симпсона при n="<< y.size() - 1 << " равно "

<< res << "\n";

return res;

}

/\*Функция поиска погрешности метода центральных прямоугольников

\*Параметры

\* 1)lower\_limit - нижний предел

\* 2)upper\_limit - верхний предел

\* 3)x - вектор значений х

\*/

double central\_rectangles\_errror(double lower\_limit, double upper\_limit, vector<double>x) {

double res=1,temp, max\_d;

max\_d=abs(second\_derivative(x[0]));

for (int i = 1; i < x.size(); i++){

temp = abs(second\_derivative(x[i]));

if (temp > max\_d) { max\_d = temp; }

}

res = (upper\_limit - lower\_limit) \* (x[1]-x[0])

\* (x[1] - x[0]) \* max\_d / 24;

cout << Blue << "Значение погрешности при вычислении”

<<“ методом центральных прямоугольников при n="

<< x.size() - 1 << " равно " << res << "\n";

return res;

}

/\*Функция поиска погрешности метода трапеций

\*Параметры

\* 1)lower\_limit - нижний предел

\* 2)upper\_limit - верхний предел

\* 3)x - вектор значений х

\*/

double trapeze\_errror(double lower\_limit, double upper\_limit, vector<double>x) {

double res = 1, temp, max\_d;

max\_d = abs(second\_derivative(x[0]));

for (int i = 1; i < x.size(); i++) {

temp = abs(second\_derivative(x[i]));

if (temp > max\_d) { max\_d = temp; }

}

res = (upper\_limit - lower\_limit) \* (x[1] - x[0])

\* (x[1] - x[0]) \* max\_d / 12;

cout << Blue << "Значение погрешности при вычислении”

<<“ трапеций при n="

<< x.size() - 1 << " равно " << res << "\n";

return res;

}

/\*Функция поиска погрешности метода Симсона

\*Параметры

\* 1)lower\_limit - нижний предел

\* 2)upper\_limit - верхний предел

\* 3)x - вектор значений х

\*/

double Simpson\_errror(double lower\_limit, double upper\_limit, vector<double>x) {

double res = 1, temp, max\_d;

max\_d = abs(third\_derivative(x[0]));

for (int i = 1; i < x.size(); i++) {

temp = abs(third\_derivative(x[i]));

if (temp > max\_d) { max\_d = temp; }

}

res = (upper\_limit - lower\_limit) \* (x[1] - x[0])

\* (x[1] - x[0]) \* (x[1] - x[0]) \* max\_d / 288;

cout << Blue << "Значение погрешности при вычислении”

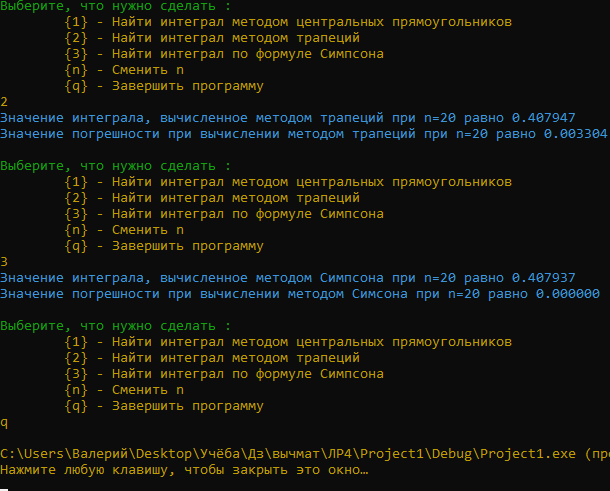
<<“ Симпсона при n="

<< x.size() - 1 << " равно " << res << "\n";

return res;

}

# Результаты работы программы



# Вывод

В ходе данной работы были закреплены знания и умения по вычислению интеграла при помощи методов средних прямоугольников, трапеция и методу Симпсона.

Самым точным является метод Симпсона, а самым не точным метод трапеций. При увеличении кол-во промежутков разбиения n точность результатов возрастает.