Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Отчет

по лабораторной работе № 5

"Интерполяционный многочлен Лагранжа"

по дисциплине

«Вычислительные алгоритмы»

Студент гр. ПИ-92 Дьяков Д.В.

Борисов А.Е.

Преподаватель, к.т.н. Проскурин А.В.

Барнаул 2022

# Задание

1. Придумайте функцию и составьте таблицу ее значений.
2. Напишите программу, которая интерполирует значения функции в произвольных точках по ее таблице при помощи интерполяционного многочлена Лагранжа.
3. Постройте сравнительные графики: исходной функции, множества точек интерполяции, многочлена Лагранжа по этим точкам.
4. Исследуйте зависимость уклонения интерполяционного многочлена от количества и расположения узлов интерполяции.
5. Повторите пункты 3,4 для функций: , ,

# Тесты

**Функция**

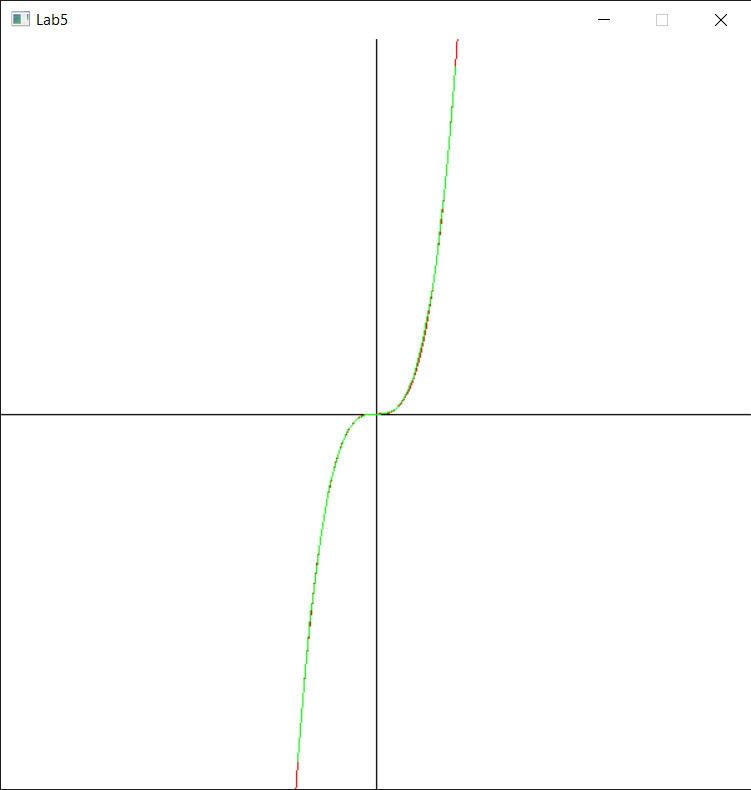


Рисунок 1 Тест 1

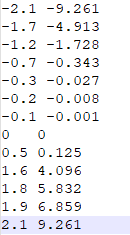


Рисунок 2 Таблица значений к 1 тесту

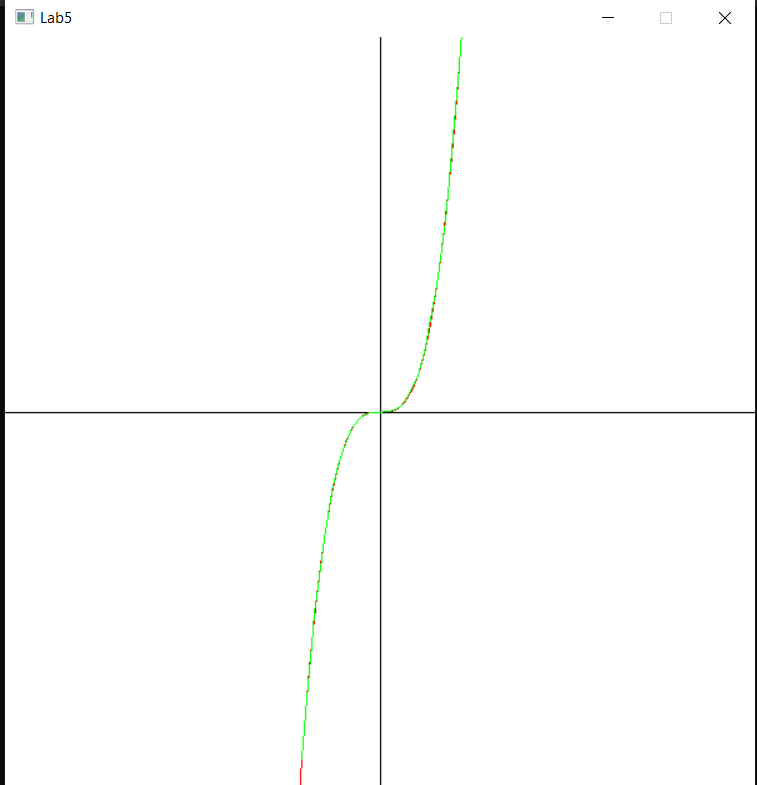


Рисунок 3 Тест 2

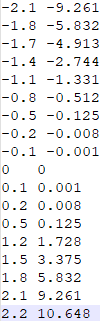


Рисунок 4 Входные данные к тесту 2

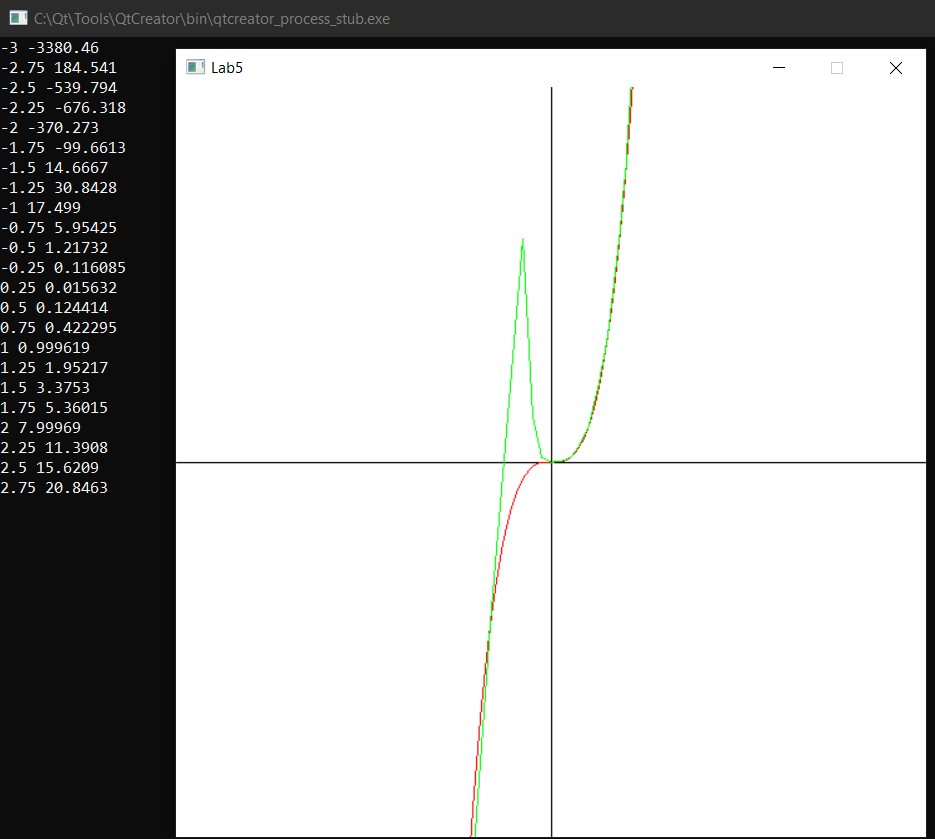


Рисунок 5 Тест 3

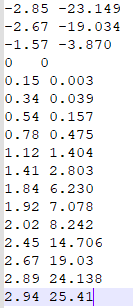


Рисунок 6 Входные данные к тесту 3

**Функция**

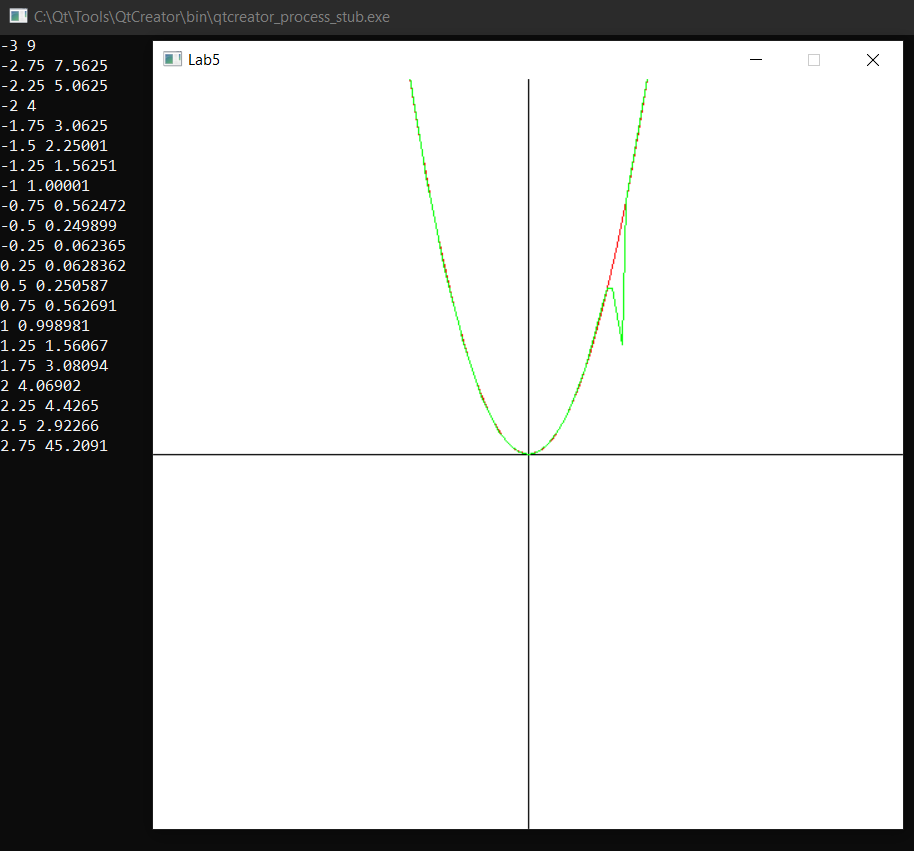


Рисунок 7 Тест 1

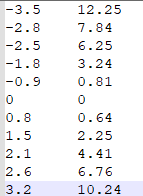


Рисунок 8 Входные данные к тесту 1

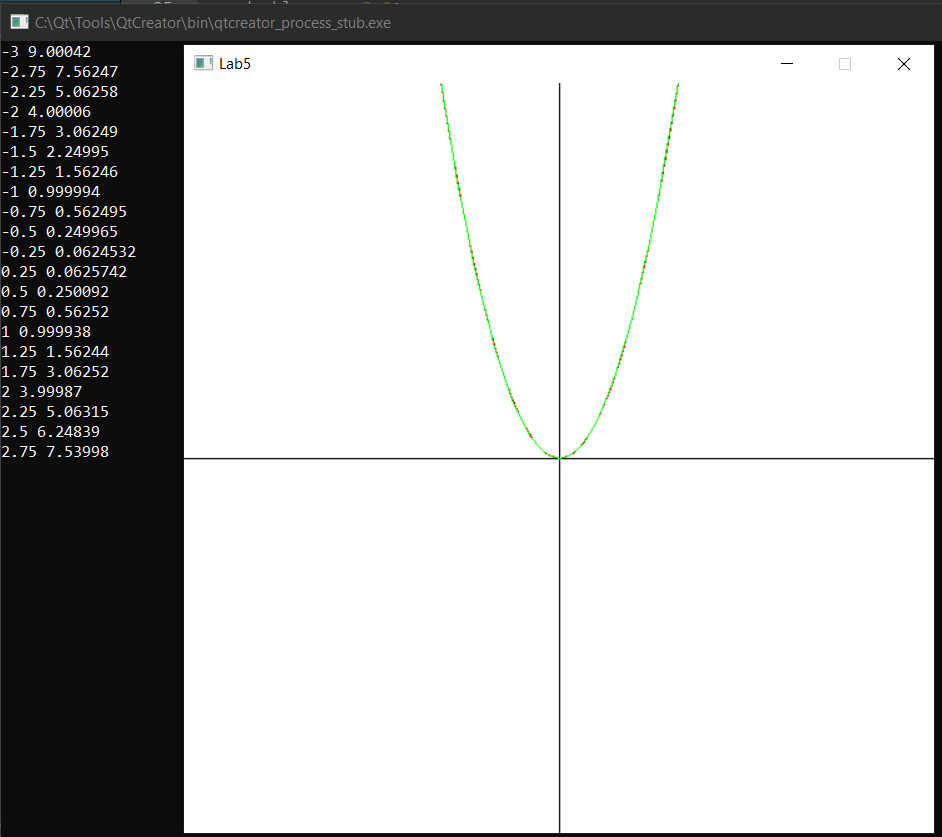


Рисунок 9 Тест 2

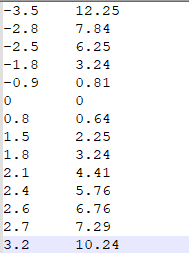


Рисунок 10 Входные данные к тесту 2

Как видно по графику после добавления нескольких значений в таблицу в том месте, где в предыдущем тесте наблюдался скачек, расхождение значений стало практически нулевым.

**Функция**

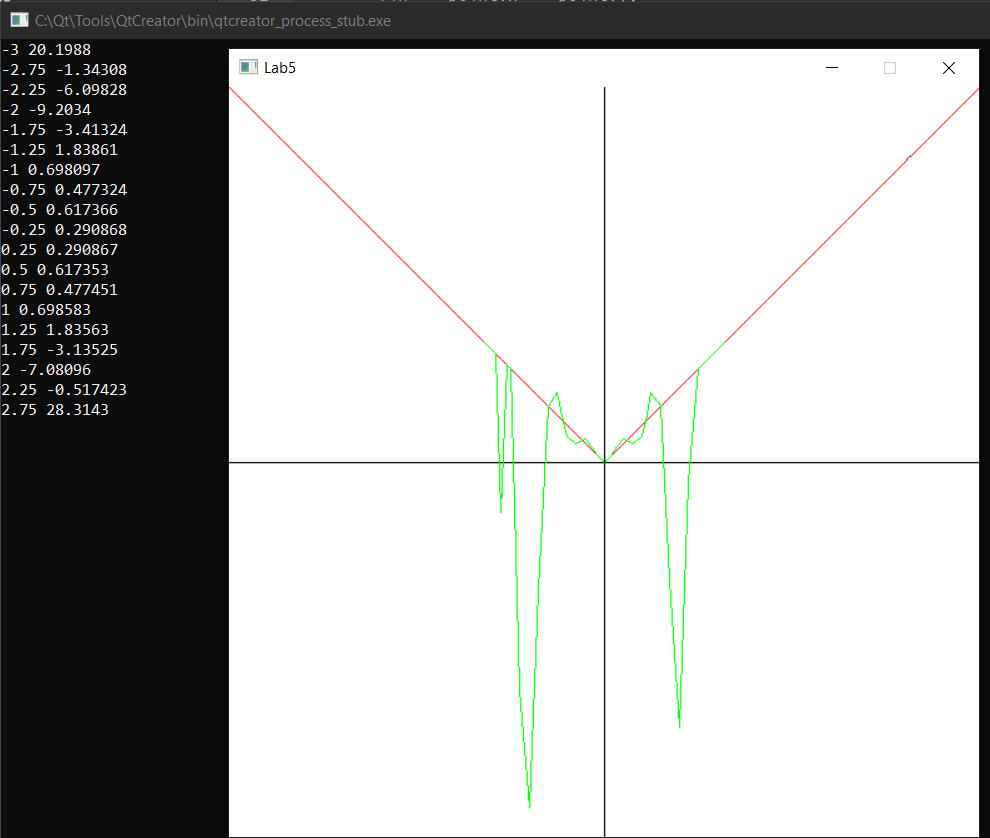


Рисунок 11 Тест 1

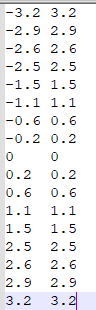


Рисунок 12 Входные данные к тесту 1



Рисунок 13 Тест 2

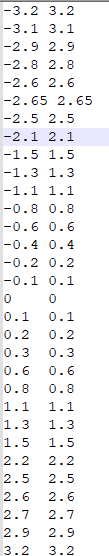


Рисунок 14 Входные данные к тесту 2

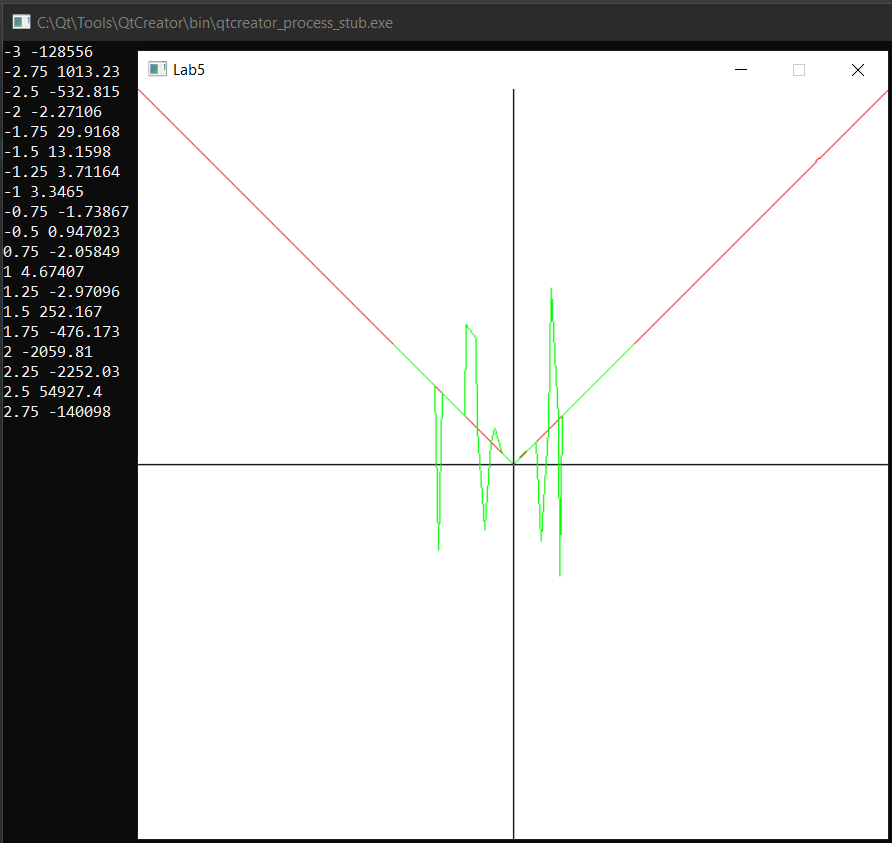


Рисунок 15 Тест 3

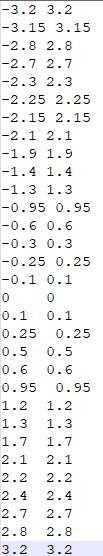


Рисунок 16 Входные данные к тесту 3

По графику видно, что не смотря на изменение количества узлов интерполяции и их расположение, расхождение велико, это можно объяснить тем, что функция не дифференцируема.

Функция

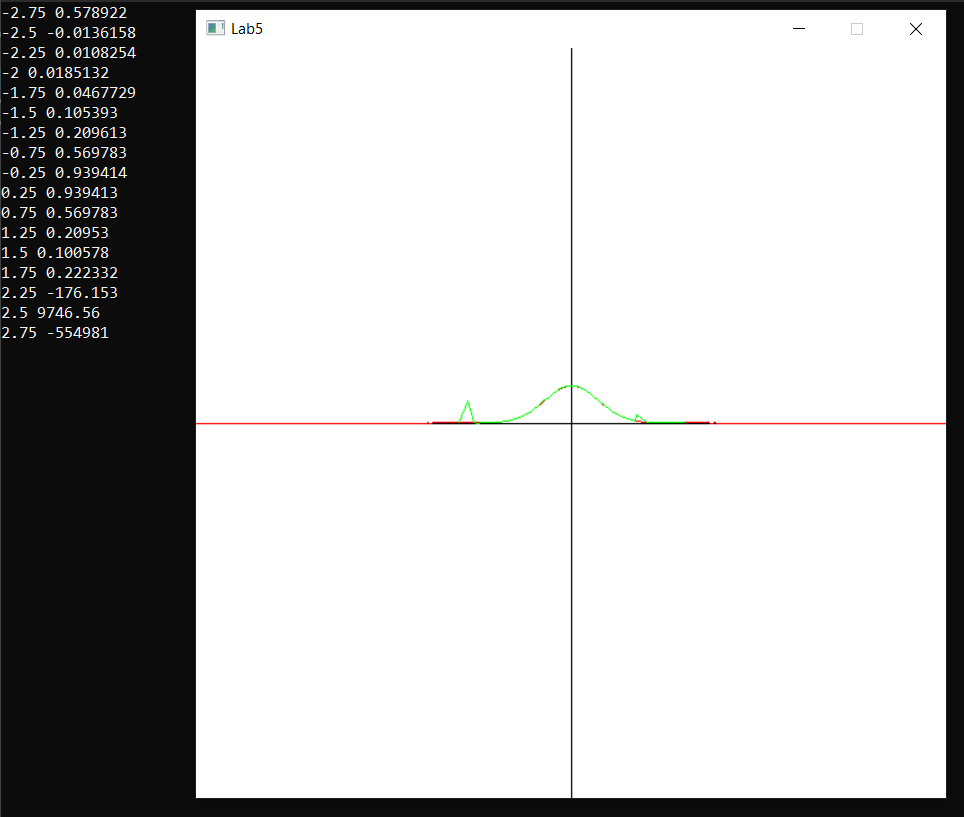


Рисунок 17 Тест 1

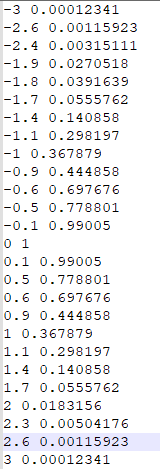


Рисунок 18 Входные данные к тесту 1

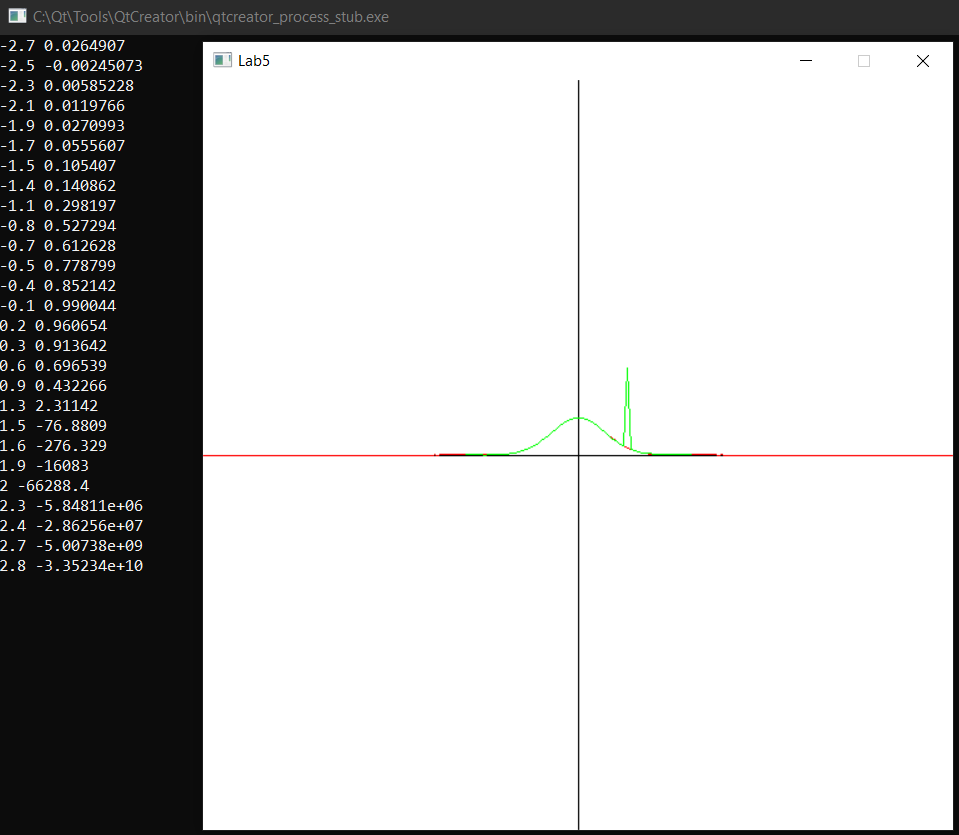


Рисунок 19 Тест 2

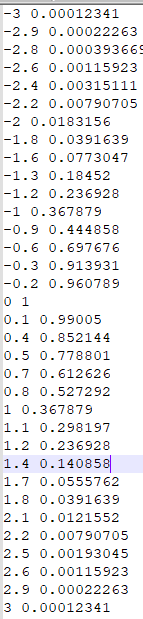


Рисунок 20 Входные данные к тесту 2

Как видно из графиков, что отклонение уменьшается с увеличением количества узлов интерполяции, а также большое значение имеет расположение этих узлов, чем ближе друг к другу эти узлы, тем точнее будут значения многочлена на этом интервале. Кроме того, большое значение имеет дифференцируемость функции, если функция не дифференцируема, то точность интерполирования будет низкой.

# Текст программы

#include <iostream>

#include <vector>

#include <SDL.h>

#include <functional>

#include <fstream>

#include <algorithm>

#undef main

constexpr int winHeight = 600;

constexpr int winWidth = 600;

SDL\_FPoint **ToWindowCoordinate**(const SDL\_FPoint& point);

double **Polinomial**(double x, const std::vector<SDL\_FPoint>& table);

// Создает таблицу точек для заданной функции

std::vector<SDL\_FPoint> **GetPoints**(const std::function<double(double value)>& fn, double eps);

void **PollEvent**();

int **main**(int argc, char\*\* argv)

{

std::ifstream fin("input.dat");

if (!fin.is\_open())

return -3;

std::vector<SDL\_FPoint> tablePoint;

while (!fin.eof()) {

SDL\_FPoint point;

fin >> point.x >> point.y;

tablePoint.push\_back(point);

}

double x = -3.0;

while(x < 3.0) {

auto it = std::find\_if(tablePoint.begin(), tablePoint.end(), [x](const SDL\_FPoint& point) {

return std::abs(point.x - (float)x) < 0.001;

});

if (it == tablePoint.end()) {

double y = Polinomial(x, tablePoint);

SDL\_FPoint point;

point.x = x;

point.y = y;

if (std::abs(point.y) <= 10.)

tablePoint.push\_back(point);

std::cout << x << " " << y << std::endl;

}

x += 0.1;

}

if(SDL\_Init(SDL\_INIT\_EVERYTHING) < 0)

return -1;

SDL\_Window\* pwindow = SDL\_CreateWindow("Lab5",

SDL\_WINDOWPOS\_UNDEFINED,

SDL\_WINDOWPOS\_UNDEFINED ,

winWidth, winHeight, 0);

SDL\_Renderer\* prenderer = SDL\_CreateRenderer(*pwindow*, -1, 0);

auto fun = [](double x) {

return sin(x);

};

std::cout << "Sin(x): " << std::endl;

std::vector<SDL\_FPoint> points = GetPoints(fun, 0.1);

std::for\_each(tablePoint.begin(), tablePoint.end(), [](SDL\_FPoint& point) {

point = ToWindowCoordinate(point);

});

std::sort(tablePoint.begin(), tablePoint.end(), [](const SDL\_FPoint& p1, const SDL\_FPoint& p2)

{

return p1.x > p2.x;

});

while (true) {

SDL\_RenderPresent(*prenderer*);

SDL\_SetRenderDrawColor(*prenderer*, 255, 255, 255, 255);

SDL\_RenderClear(*prenderer*);

// Отрисовка осей координат

auto p1 = ToWindowCoordinate({ - 10., 0});

auto p2 = ToWindowCoordinate({10, 0});

SDL\_SetRenderDrawColor(*prenderer*, 0, 0, 0, 255);

SDL\_RenderDrawLineF(*prenderer*, p1.x, p1.y, p2.x, p2.y);

auto p3 = ToWindowCoordinate({0, 10});

auto p4 = ToWindowCoordinate({0, -10});

SDL\_SetRenderDrawColor(*prenderer*, 0, 0, 0, 255);

SDL\_RenderDrawLineF(*prenderer*, p3.x, p3.y, p4.x, p4.y);

SDL\_SetRenderDrawColor(*prenderer*, 255, 0, 0, 255);

SDL\_RenderDrawLinesF(*prenderer*, points.data(), (int)points.size());

SDL\_SetRenderDrawColor(*prenderer*, 0, 255, 0, 255);

SDL\_RenderDrawLinesF(*prenderer*, tablePoint.data(), (int)tablePoint.size());

PollEvent();

}

SDL\_DestroyRenderer(*prenderer*);

SDL\_DestroyWindow(*pwindow*);

return 0;

}

void **PollEvent**()

{

SDL\_Event event;

while (SDL\_PollEvent(*&event*))

{

if (event.type == SDL\_QUIT)

exit(0);

}

}

SDL\_FPoint **ToWindowCoordinate**(const SDL\_FPoint& point)

{

SDL\_FPoint winPoint;

winPoint.x = winWidth / 2.f + ( (winWidth / 2.f) / 10.f \* point.x);

winPoint.y = winHeight / 2.f - ( (winHeight / 2.f) / 10.f \* point.y);

return winPoint;

}

std::vector<SDL\_FPoint> **GetPoints**(const std::function<double(double value)>& fn, double eps)

{

std::vector<SDL\_FPoint> points;

double x = -10.;

while (x < 10.)

{

SDL\_FPoint point;

point.x = static\_cast<float>(x);

point.y = static\_cast<float>(fn(x));

points.push\_back(ToWindowCoordinate(point));

x += eps;

}

return points;

}

double **Polinomial**(double x, const std::vector<SDL\_FPoint>& table)

{

double sum = 0;

for (size\_t k = 0; k < table.size(); ++k) {

// П(j = 0, n) (x - xj)

double p1 = 1;

for (size\_t j = 0; j < table.size(); ++j) {

if (j != k)

p1 \*= x - table[j].x;

}

// П(j = 0, n) (xk - xj)

double p2 = 1;

for (size\_t j = 0; j < table.size(); ++j) {

if (j != k)

p2 \*= table[k].x - table[j].x;

}

sum += table[k].y \* p1 / p2 ;

}

return sum;

}