|  |  |  |
| --- | --- | --- |
| Министерство образования и науки Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
| path817.png | | |
| Кафедра прикладной математики | | |
|  | | |
| Расчетно-графическое задание | | |
| по дисциплине «Компьютерная графика» | | |
|  | | |
|  | | |
|  | | |
| эмблема_светлая.png | Факультет: | ПМИ |
| Группа: | ПМИ-62 |
| Вариант: | 2 |
| Студент: | Ершов П. К. |
| Преподаватель: | Задорожный А. Г. |
|  | | |
| Новосибирск | | |
| 2019 | | |

1. Цель работы

Реализовать программу, отображающую график функции, получаемой в результате использования соответствующего сплайна.

1. **Задание**

Интерполяционный сплайн на основе полиномов Лагранжа степени 3.

Входными данными является набор точек на экране, задаваемый в произвольном порядке с помощью мыши. При добавлении новой точки сплайн должен автоматически перестраиваться. Программа должна предоставить возможность движения по сцене и масштабирования с помощью клавиатуры/мыши. Координатные оси и координатная сетка должны быть отображены и подписаны, причем при масштабировании размер ячеек сетки не должен меняться – меняются только числовые подписи и масштабируется сам график. Для заданий с дифференцированием (интегрированием) на экране должен отображаться график не только самого сплайна, но и его производной (первообразной). В том случае, когда недостаточно контрольных точек для построения сплайна, следует выдавать соответствующее предупреждение, а сам сплайн, например, отображать ломаной.

1. **Анализ задачи**

Для того, чтобы построить кривую, необходимо решить сплайн. Сплайн задаётся формулой:

Здесь это граници отрезка, на котором расположены две соседние точки кривой и для которого будет расчитываться сплайн.

Чтобы решить сплайн необходимо найти коэффициенты .

Для этого из системы уравнений:

извлекаются формулы, необходимые для получения коэффициентов. Из указаннйо системы можно сделать вывод, что: сплайн от = , от = . А превые и вторые производные текущего сплайна равны первым и вторым производным предыдущего сплайна.

Исходя из этого пожно получить формулы:

Однако, эти формулы применимы только для точек, начиная со второй и дальше.

Первая точка теребует отделного решения: начального приближения.

тут это рзность .

1. **Описание управления**

Добавление новых точек производится с помощью левой кнопки мыши.

Управление:

w – перемещение вверх.

a – перемещение влево.

s – перемещение вниз.

d – перемещение вправо.

+ – приблизить кривую.

- – отдалить кривую.

z – удалить кривую.

1. Текст программы

#include <windows.h>

#include "gl\glut.h"

#include <string>

#include <vector>

using namespace std;

class Point//структура ТОЧКА

{

public: double x, y;

Point()

{

x = 0;

y = 0;

}

~Point() {}

public: void SetCoordinates(double x2, double y2)

{

x = x2;

y = y2;

}

};

vector <Point> points;

vector <Point> l\_coord; // точки полинома Лангранжа

Point shift;

bool newPoint = false;

int width = 1200;

int height = 800;

double scale = 2;

void drawBitmapText(string str, double x, double y, double z)

{

glRasterPos3f(x, y, z); // Определяет положение растра для операций с пикселями

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

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, str[i]);

}

}

void drawDot(int x, int y)//отрисовка точки

{

glColor3f(0.3, 0.5, 0.7);

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

void drawLine(Point p1, Point p2)//отрисовка линии

{

glColor3f(0.7, 0.4, 0.3);

glBegin(GL\_LINES);

glVertex2f(p1.x, p1.y);

glVertex2f(p2.x, p2.y);

glEnd();

}

void drawGrid()//отрисовка сетки

{

glLineWidth(1);

glColor3d(0, 0, 0);

{

glBegin(GL\_LINES);

for (int i = 0; i < 10000; i += 100 / scale)

{

glVertex2f(i, -10000);

glVertex2f(i, 10000);

glVertex2f(-10000, i);

glVertex2f(10000, i);

}

glEnd();

glBegin(GL\_LINES);

for (int i = 0; i > -10000; i -= 100 / scale)

{

glVertex2f(i, -10000);

glVertex2f(i, 10000);

glVertex2f(-10000, i);

glVertex2f(10000, i);

}

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glVertex2f(0, -10000);

glVertex2f(0, 10000);

glEnd();

glBegin(GL\_LINES);

glVertex2f(-10000, 0);

glVertex2f(10000, 0);

glEnd();

}

char number[4];

glLineWidth(3);

drawBitmapText("0", 10, -10, 0);

for (double i = 100 / scale; i < 10000; i += 100 / scale)

{

glBegin(GL\_LINES);

glVertex2f(i, 5);

glVertex2f(i, -5);

glEnd();

drawBitmapText(to\_string(i / 100.0), i + 3, -20 / scale, 0);

}

for (double i = -100 / scale; i > -10000; i -= 100 / scale)

{

glBegin(GL\_LINES);

glVertex2f(i, 5);

glVertex2f(i, -5);

glEnd();

drawBitmapText(to\_string(i / 100.0), i + 3, -20 / scale, 0);

}

glEnd();

for (double i = 100 / scale; i < 10000; i += 100 / scale)

{

glBegin(GL\_LINES);

glVertex2f(-5, i);

glVertex2f(5, i);

glEnd();

drawBitmapText(to\_string(i / 100.0), 15 / scale, i + 3, 0);

}

for (double i = -100 / scale; i > -10000; i -= 100 / scale)

{

glBegin(GL\_LINES);

glVertex2f(-5, i);

glVertex2f(5, i);

glEnd();

drawBitmapText(to\_string(i / 100.0), 15 / scale, i + 3, 0);

}

}

void Lagrange()//Расчёт сплайна

{

l\_coord.clear();

double a = 0, b = 0, c = 0, d = 0, y = 0, ab = 0, bb = 0, cb = 0, h = 0;

double beta = 0, gamma = 0;

double c\_x = 0;

if (points.size() > 1)

for (c\_x = points[0].x; c\_x != points[1].x; c\_x += 1 / 8.000)

{

h = points[1].x - points[0].x;

a = 0;

b = 0;

c = (points[1].y - points[0].y) / h;

d = points[0].y - c \* points[0].x;

y = c \* c\_x + d;

Point cur;

cur.x = c\_x;

cur.y = y;

l\_coord.push\_back(cur);

}

ab = a;

bb = b;

cb = c;

if (points.size() > 1)

{

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

{

ab /= 1.5;

bb /= 1.5;

cb /= 1.5;

for (c\_x = points[i].x; c\_x != points[i + 1].x;c\_x += 1 / 8.000)

{

//\*

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

beta = 3 \* ab \* pow(points[i].x, 2) + 2\* bb \* points[i].x + cb;

gamma = 3 \* ab \* points[i].x + bb;

a = ((points[i + 1].y - points[i].y) / pow(h, 3)) - (beta / pow(h, 2)) - gamma / h;

b = gamma - 3 \* a \* points[i].x;

c = beta - 2 \* b \* points[i].x - 3 \* a \* pow(points[i].x, 2);

d = points[i].y - c \* points[i].x - b \* pow(points[i].x, 2) - a \* pow(points[i].x, 3);

y = a \* pow(c\_x, 3) + b \* pow(c\_x, 2) + c \* c\_x + d;

ab = a;

bb = b;

cb = c;

Point cur;

cur.x = c\_x;

cur.y = y;

l\_coord.push\_back(cur);

}

}

}

}

void del()//очистить вектор точек

{

points.clear();

}

void repos\_point()//сортировка точек

{

for (int i = points.size() - 1; i > 0; i--)

{

if (points[i].x < points[i - 1].x)

{

Point b;

b.x = points[i].x;

b.y = points[i].y;

points[i] = points[i - 1];

points[i - 1] = b;

}

}

}

bool x\_not\_eq(int x)//закпрет на добавление точки на уже занятую координату Х

{

int i = 0;

bool f = true;

while (i < points.size())

{

if (points[i].x == x)

{

f = false;

break;

}

i++;

}

return f;

}

void MouseClick(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

Point p;

p.SetCoordinates((double)(x - width / 2 - shift.x) / scale, (double)(height / 2 - y - shift.y) / scale);

if (x\_not\_eq(p.x) != false)

{

points.push\_back(p);

repos\_point();

Lagrange();

}

}

}

void Keyboard(GLubyte key, int x, int y)

{

switch (key)

{

case 'w':

{

shift.y -= 25; break;

}

case 'a':

{

shift.x += 25; break;

}

case 's':

{

shift.y += 25; break;

}

case 'd':

{

shift.x -= 25; break;

}

case '=':

{

scale += 0.1; break;

}

case '-':

{

scale -= 0.1; break;

}

case 'z'://удалить кривую

{

del();

}

}

glutPostRedisplay();

}

void Reshape(GLint w, GLint h)

{

width = w; height = h;

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-w / 2, w / 2, -h / 2, h / 2);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glutPostRedisplay();

}

void Display()

{

glClearColor(1, 1, 1, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glEnable(GL\_POINT\_SMOOTH);

glShadeModel(GL\_SMOOTH);

glLoadIdentity();

glTranslated(shift.x, shift.y, 0);

glScalef(scale, scale, scale);

glLineWidth(1);

drawGrid();

glLineWidth(3);

glPointSize(6);

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

{

drawDot(points[i].x, points[i].y);

}

if (points.size() > 1)

{

Point from, to; // Инициализируем начальную и конечную точку для отрисовки отрезка

from.x = 0;

from.y = 0;

to.x = 0;

to.y = 0;

for (int i = 1; i < l\_coord.size(); i++)

{

from.x = l\_coord[i - 1].x;

from.y = l\_coord[i - 1].y;

to.x = l\_coord[i].x;

to.y = l\_coord[i].y;

drawLine(from, to);

}

from.x = l\_coord[l\_coord.size() - 1].x;

from.y = l\_coord[l\_coord.size() - 1].y;

drawLine(from, points[points.size() - 1]);

}

glFinish();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB);

glutInitWindowSize(width, height);

glutInitWindowPosition(width / 10, height / 6);

glutCreateWindow("Интерполяционный сплайн на основе полиномов Лагранжа степени 3");

glutMouseFunc(MouseClick);

glutKeyboardFunc(Keyboard);

glutReshapeFunc(Reshape);

glutDisplayFunc(Display);

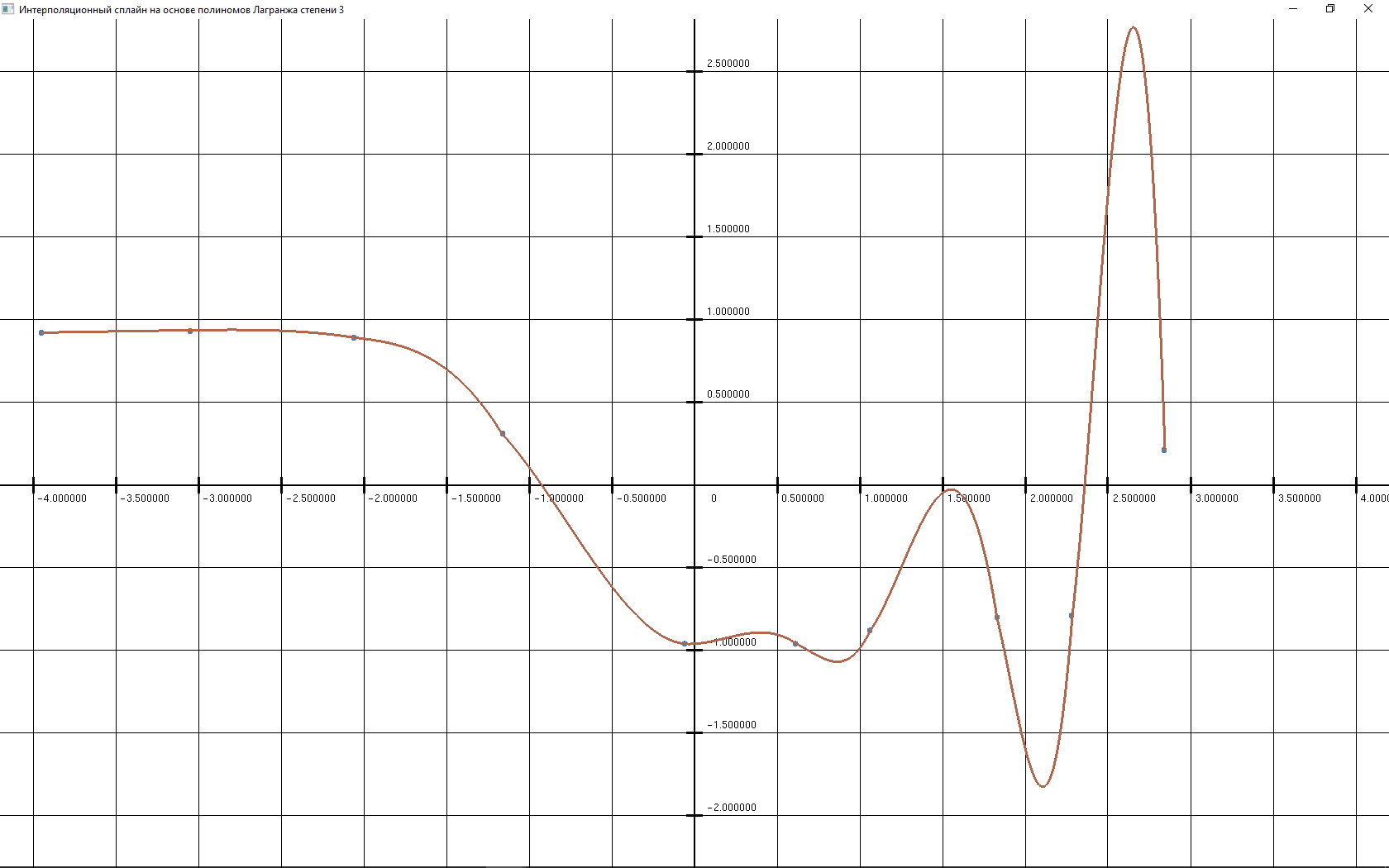
glutIdleFunc(Display);

glutMainLoop();

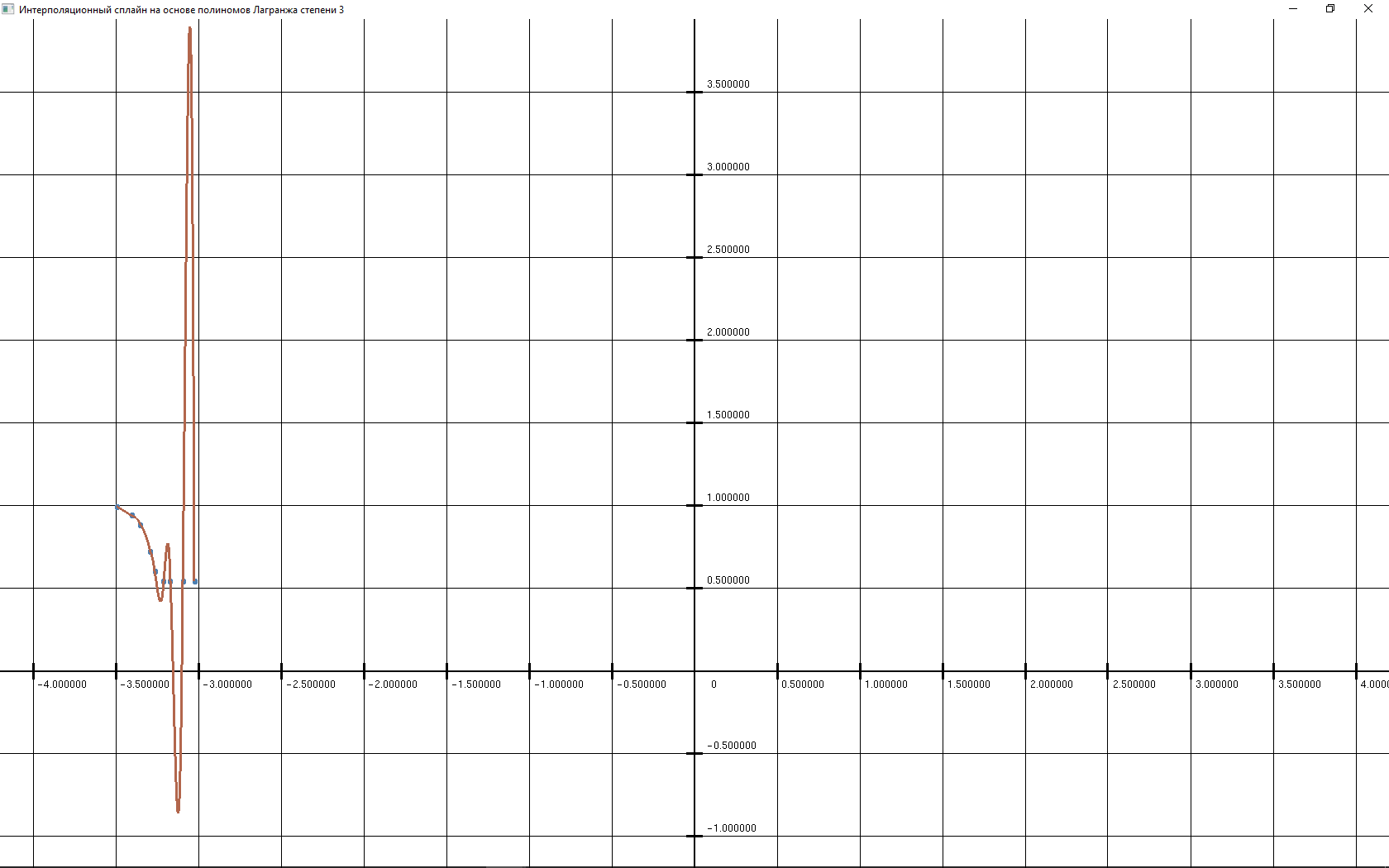
}

1. Тесты

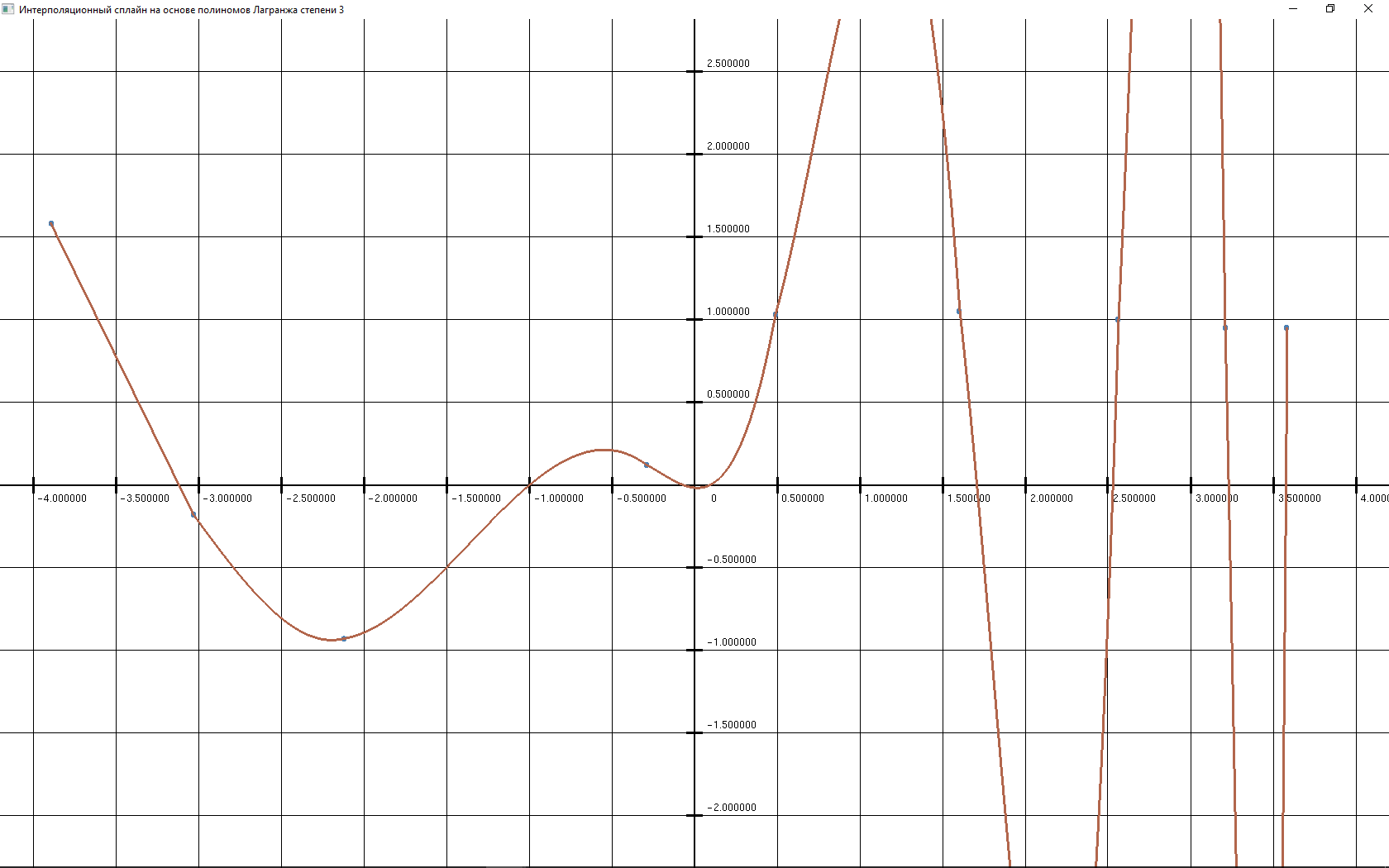
Тест 1. Отрисовка кривой.



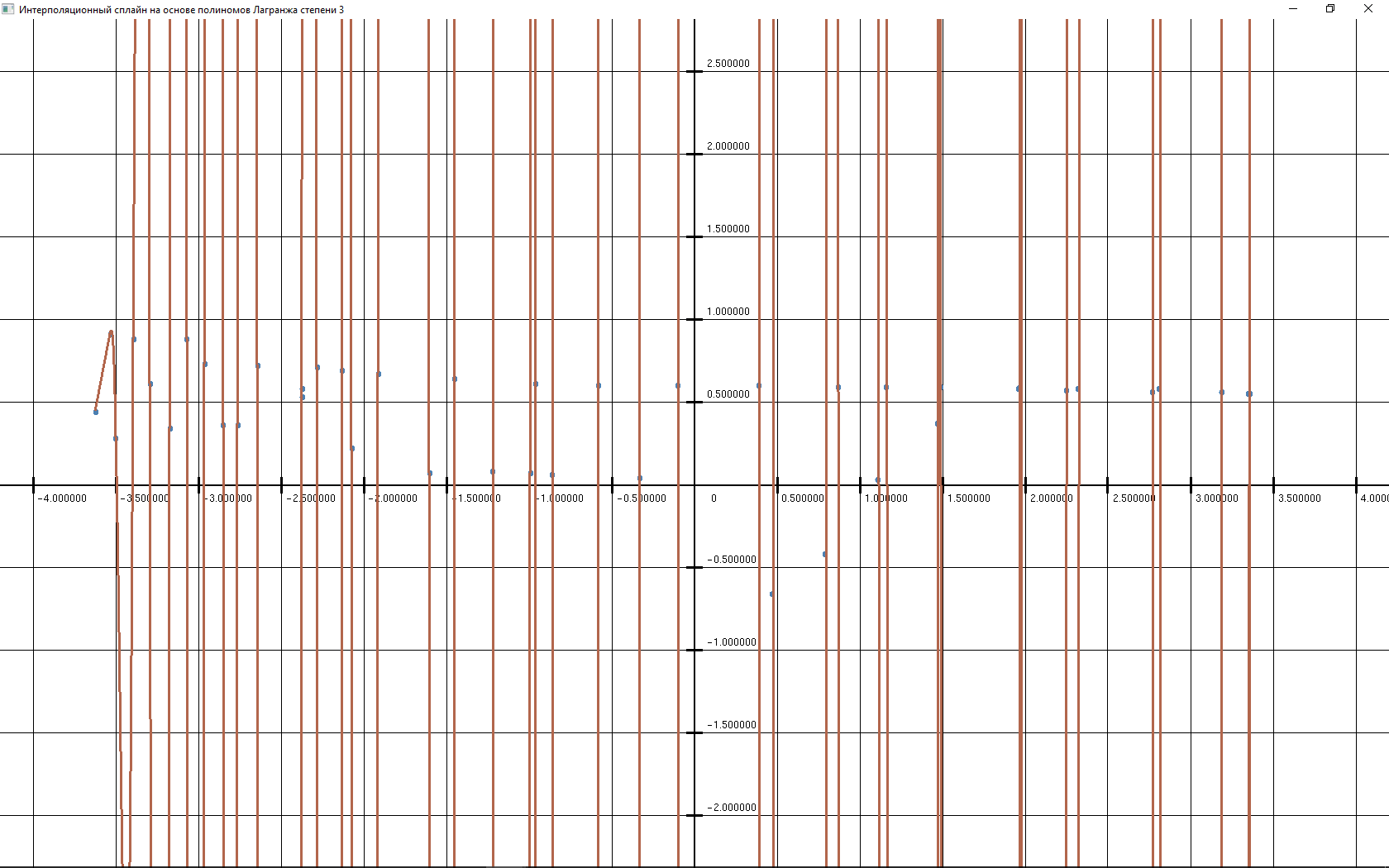
Тест 2. Кривая на малом отрезке.



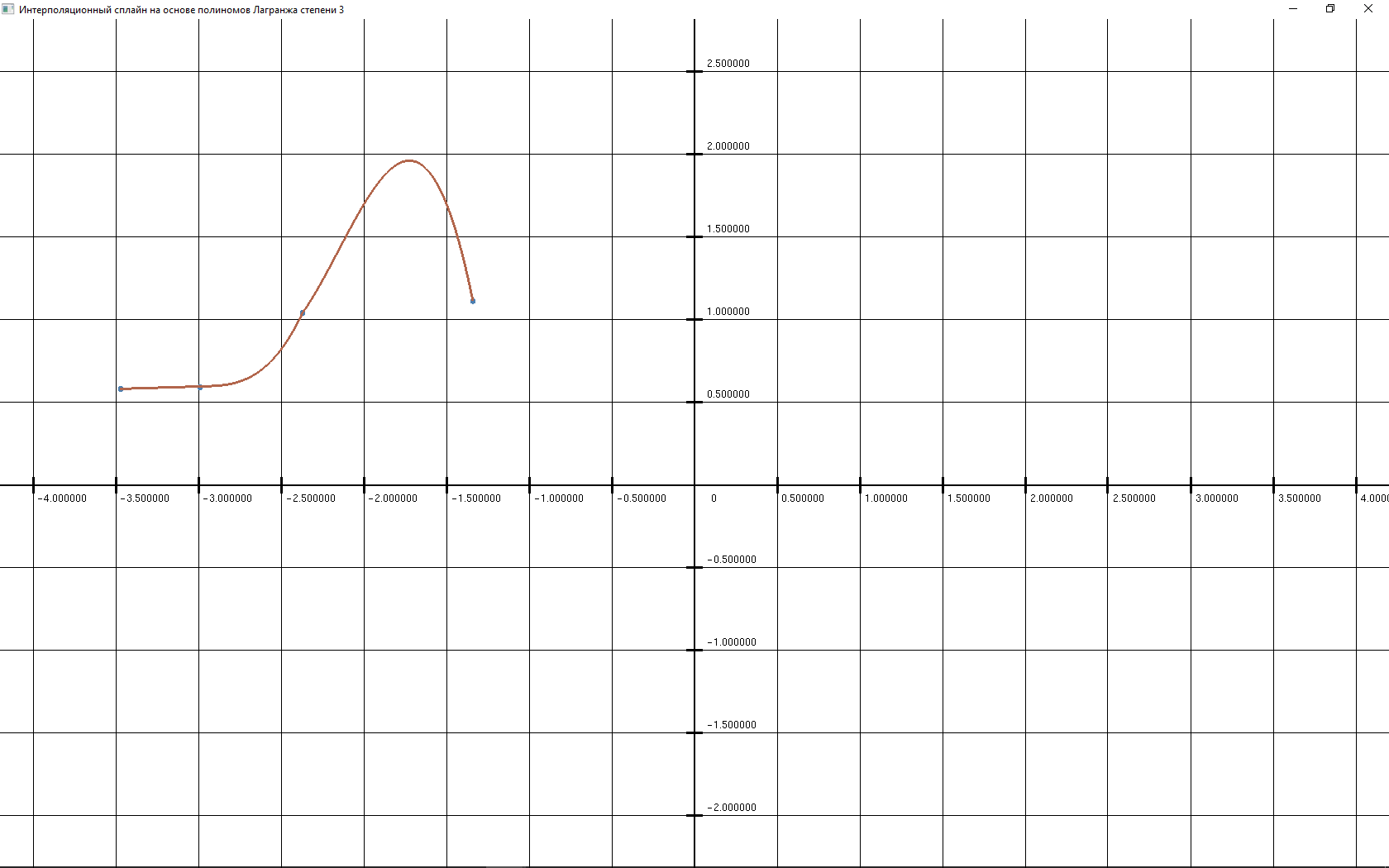
Тест 3. Кривая на большом отрезке.

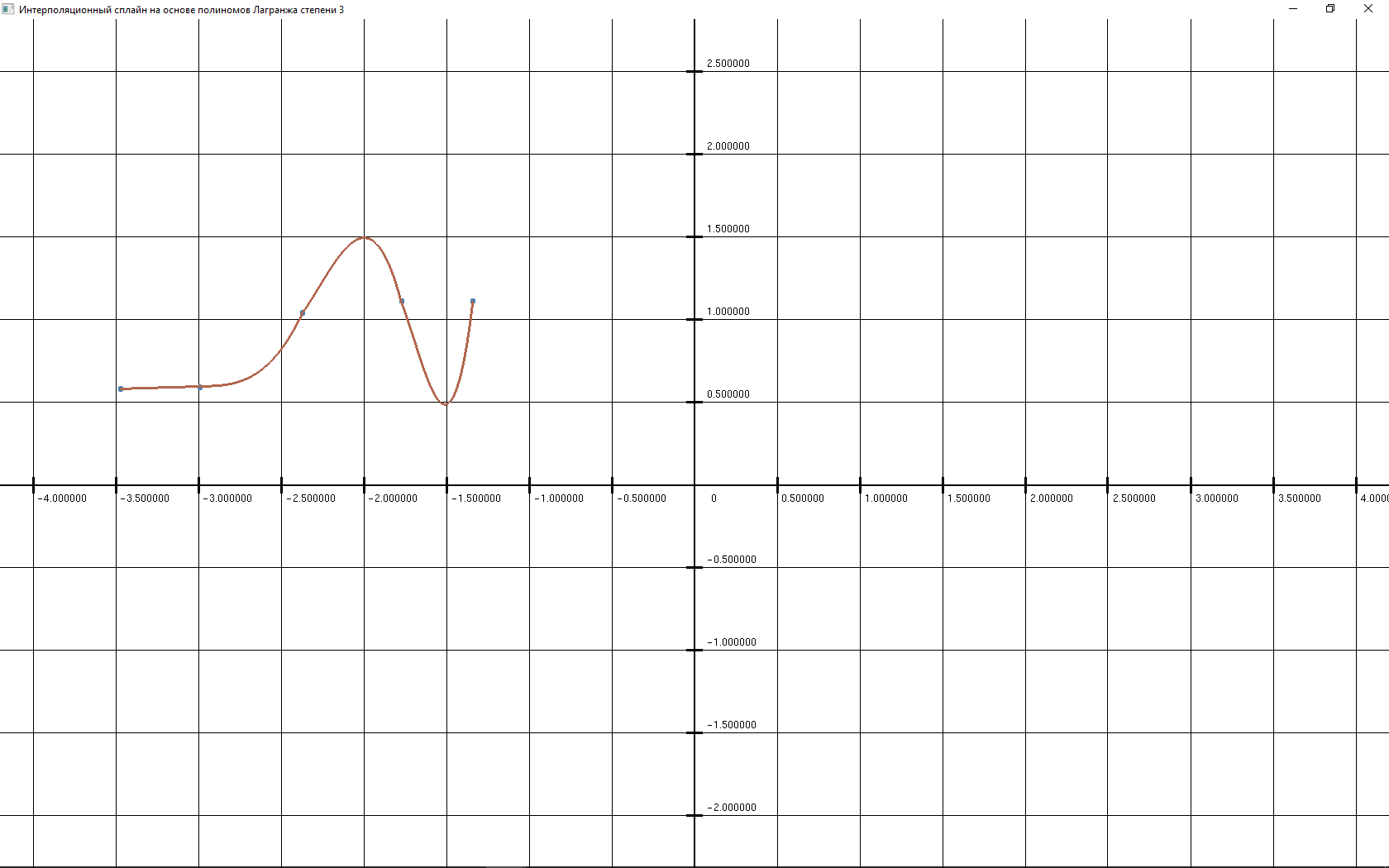


Тест 4. Кривая из множества точек.



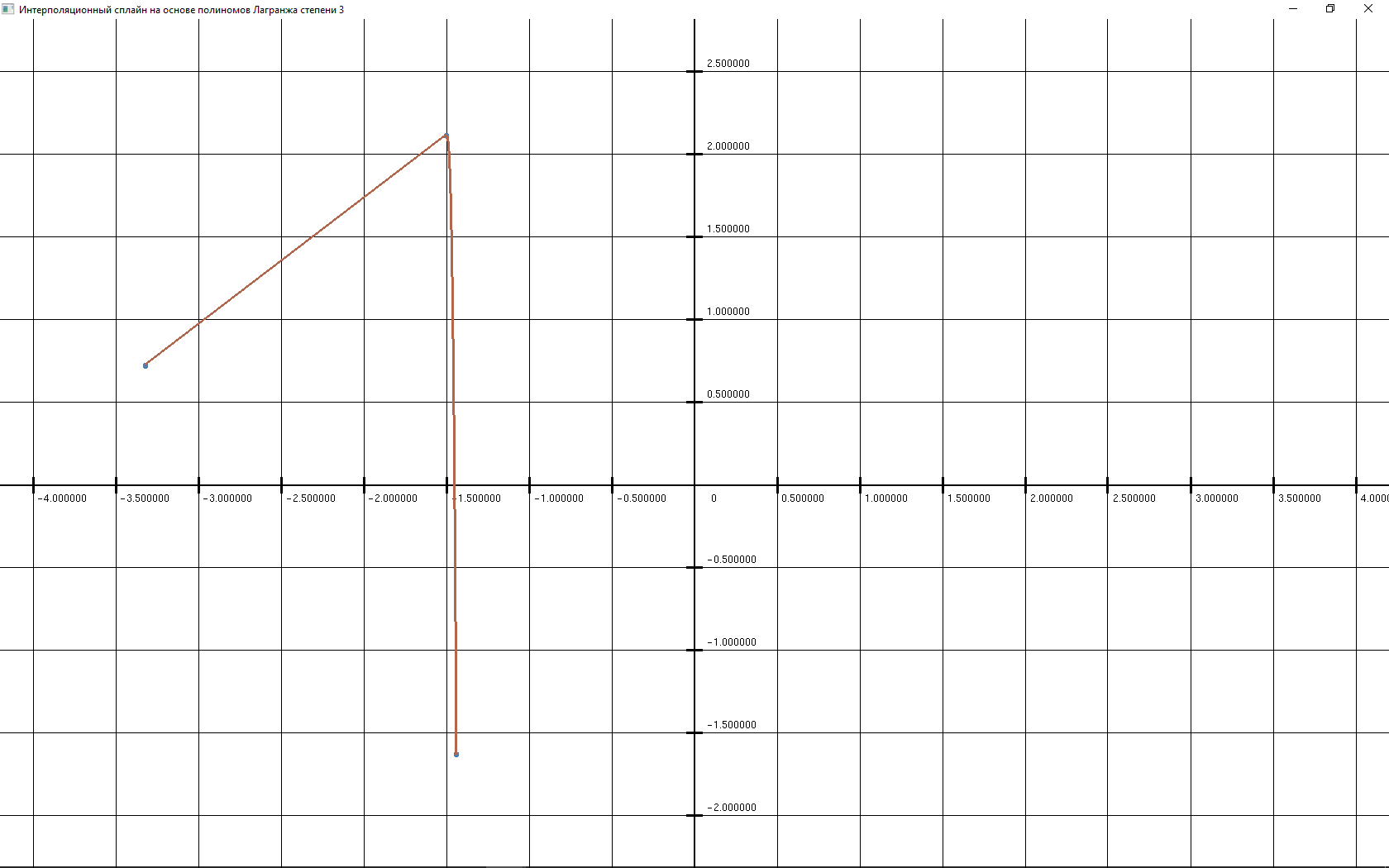
Тест 5. Установка точки левее предыдущей.





Тест 6. Демонстрация сглаженности кривой.

Вид в обычном размере.



Вид с увеличением.

