Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и информатики

Лабораторная работа №9 по дисциплине

Численные методы

«Приближение табличных функций сплайнами и МНК»

Выполнил
студент гр.5030102/20001 Дрекалов Н.С.

Преподаватель Козлов К.Н.

Санкт-Петербург

2023

Оглавление

Формулировка задачи

Необходимо для заданной табличной функции построить квадратичный сплайн с пропадающим узлом и выполнить анализ графических результатов.

Формализация

- Задана табличная функция $(x_i, y_i), i = 0, \ldots, n$.
- Нужно построить квадратичный интерполяционный сплайн $S_2^1(x)$, который на каждом элементарном промежутке является полиномом второй степени: $S_2^1|_{x_{i-1},x_i} = g_i(x) = a_i x^2 + b_i x + c_i$. Для этого требуются условия:

  $$\forall i = 1..n \begin{cases} g_i(x_{i-1}) = y_{i-1} \\ g_i(x_i) = y_i \end{cases}, \forall i = 1..n-1 \; g_i'(x_i) = g_{i+1}'(x_i), \text{(1)}$$

  а также $\begin{cases} g_1(x_0) = y_0 \\ g_1(x_1) = y_1 \\ g_1(x_2) = y_2 \end{cases}$ (2)

Квадратичный сплайн с пропадающим узлом

Условия применимости:

1. Сплайн всегда можно построить по табличной функции, если она непрерывная и есть решение систем для каждой пары точек (а оно есть всегда, когда $x_i \neq x_{i-1}$, так как в общем решении есть деление на их разность).

Алгоритм метода:

1. Решаем систему (2), получая оттуда $g_1(x) = g_2(x)$
2. Затем для каждого $x_i, i = 2..n$, решаем систему уравнений (1), получая $g_i(x)$
3. В итоге получаем $n - 1$ полиномов второй степени, которые вместе составляют сплайн $S_2^1(x)$ (по необходимости из этого полинома также получается лишь значение в конкретной точке вне искомой сетки)

Предварительный анализ задачи

Анализ функций на непрерывность:

- $y = 3x - cos(x) - 1$ непрерывна на R, поэтому можем рассматривать любой отрезок непрерывности (по умолчанию $[-1; 1]$)
- $y = x^3 - sign(x)x^2 + 6x + 3$ также непрерывна на R, так как при $x \to \pm 0$ $y \to \pm 0^3 \mp 0^2 \pm 6 * 0 + 3 = 3 = y(0)$. Однако есть некоторые сложности с расчётом производной в нуле (из-за sign(x), например, для 2-й производной справа: $6x - 2$, а слева $6x + 2$, разница между правосторонней и левосторонней производной не ноль). Поэтому будем рассматривать отрезок непрерывности $[0.1; 2]$

Тестовый пример к методам

Тестовый пример.

- $y = x^3 - \operatorname{sign}x \cdot x^2 + 6x + 3 = f(x)$

Табличная функция:

x	1	2	3	4
y	9	19	39	75

1) $\begin{cases} a + b + c = 9 \\ 4a + 2b + c = 19 \\ 9a + 3b + c = 39 \end{cases} \Leftrightarrow \begin{cases} a = \dfrac{-19 + 39 + 2\cdot 9 - 2\cdot 39 - 3\cdot 9 + 3\cdot 19}{1\cdot 2\cdot(-1)} \\ b = \dfrac{\cancel{39-19}}{1} - \cancel{2}3\cdot a \\ c = 9 - b - a \end{cases} \Leftrightarrow$

$\Leftrightarrow \begin{cases} a = 5 \\ b = \cancel{105} -5 \\ c = \cancel{-61}\ 9 \end{cases}$ $\Rightarrow g_1(x) = g_2(x) = 5x^2 - 5x + 9$

$g_1'(x) = 10x - 5$

2) $\begin{cases} 9a + 3b + c = 39 \\ 16a + 4b + c = 75 \\ 6a + b = 10\cdot 3 - 5 \end{cases} \Leftrightarrow \begin{cases} a = \dfrac{75-39}{1\cdot 1} - \dfrac{25}{1} \\ b = \dfrac{75-39}{1} - 7a \\ c = 39 - 3b - 9a \end{cases} \Leftrightarrow \begin{cases} a = 11 \\ b = -41 \\ c = 63 \end{cases}$

$g_3(x) = 11x^2 - 41x + 63$

3) Проверка:

$x=1$: $S_2^1(x) = \cancel{5+10-1=9=y_0} = 5-5+9=9=y_0$

$x=\frac{3}{2}$: $f\left(\frac{3}{2}\right) - S_2^1\left(\frac{3}{2}\right) = \frac{3}{8} = 0{,}375$

$x=2$: $S_2^1(x) = 20-10+9 = 19 = y_1$

$x=\frac{5}{2}$: $f\left(\frac{5}{2}\right) - S_2^1\left(\frac{5}{2}\right) = -\frac{3}{8} = -0{,}375$

$x=3$: $S_2^1(x) = 45-15+9 = 39 = y_2^*$

$x=\frac{7}{2}$: $f\left(\frac{7}{2}\right) - S_2^1\left(\frac{7}{2}\right) = \frac{3}{8} = 0{,}375$

$x=4$: $S_2^1(x) = \cancel{80-20+9=69} = 11\cdot16-41\cdot4+63=75=y_3$

Контрольные тесты

Построим графики зависимостей для каждой из функций

- Сплайнов от 6, 8 и 10 узлов и функции от x на выбранном интервале, а также ошибки этих же сплайнов.
- Ошибки интерполяции от числа узлов изначальной сетки.
- Максимальной ошибки от значения в пропадающем узле в некотором диапазоне от его точного значения.
- Ошибки интерполяции при возмущении данных.

Модульная структура программы

Выполнение нужных действий достигается с помощью 2-х программ: на языках python и C++. При этом C++ программа строит сплайн и вычисляет его значения в проверочной сетке, в то время как python программа занимается анализом результата (расчёт различных ошибок, построение графиков и т. п.). Далее будут перечислены объявления функций.

C++

quadratic_spline.hpp

```cpp
/// \class точка
class Point {
public:
    double x, y;

    explicit Point(double x, double y) : x(x), y(y) {}
};

/// \class Квадратичный полином
class QuadPoly {
public:
    /// Коэффициенты полинома
    double a = 0, b = 0, c = 0;
    /// Отрезок, на котором действует полином
    double x0 = 0, x1 = 0;

    QuadPoly() = default;
    QuadPoly(double a, double b, double c, double x0, double x1) : a(a), b(b), c(c),
x0(x0), x1(x1) {}
};


#define str(a) std::to_string(a)
/// \brief Функция преобразует полином в одну строку
/// \param quadPoly полином
/// \return Полином как строка
std::string quadPolyToStr(QuadPoly const& quadPoly);

/// \brief Функция преобразует точку в одну строку
/// \param point точка
/// \return Точка как строка
std::string pointToStr(Point const& point);

// Класс, строящий квадратичный сплайн и вычисляющий его значения в нужных точках
class QuadraticSpline {
    std::vector<Point> tabFunc;

    /// \brief Функция считает значние сплайна в заданной точке
    /// \param spline сплайн
    /// \param x заданная точка
    /// \return Значение сплайна в заданной точке
    static double splineY(std::vector<QuadPoly> const& spline, double x);

    /// \brief Функция считает значние полинома в заданной точке
    /// \param quadPoly полином
    /// \param x заданная точка
    /// \return Значение полинома в заданной точке
    static double quadPolyY(QuadPoly const& quadPoly, double x);
```

```
    /// \brief Функция считает значние производной полинома в заданной точке
    /// \param a коэффициент при x^2
    /// \param b коэффициент при x
    /// \return Значение производной полинома в заданной точке
    static double derFromCoefs(double a, double b, double x);

    /// \brief Функция считает коэффициенты квадратичного полинома в заданном диапазоне
    /// \param \a коэффициент при x^2
    /// \param x1, x0 диапазон
    /// \param y1, y0 значения в данных x1, x0
    /// \param d0 производная в x0
    /// \return Коэффициенты полинома в заданном диапазоне
    static QuadPoly evalLocalCoefficients(double x1, double x0, double y0, double y1,
double d0);

    /// \brief Функция считает коэффициенты квадратичного полинома по трём точкам
    /// \param \a коэффициент при x^2
    /// \param x2, x1, x0 значения по x
    /// \param y2, y1, y0 значения в данных x2, x1, x0
    /// \return Коэффициенты полинома
    static QuadPoly evalFirstCoefficients(double x2, double x1, double x0, double y2,
double y1, double y0);
public:
    /// \brief Функция возвращает искомую табличную функцию
    /// \return Табличная функция как массив точек
    std::vector<Point> const & getPoints();

    /// \brief Функция добавляет точку в массив искомых точек
    /// \param x,y - координаты точки
    void addPointToTabFunc(double x, double y);

    /// \brief Основная функция программы - считает по добавленным искомым точкам
коэффициенты полиномов сплайна
    /// \return Сплайн как массив коэффиццентов квадратичных полиномов
    std::vector<QuadPoly> evaluateCoefficients();

    /// \brief Функция считает для переданного сплайна значения в конкретных внутренних
точках
    /// \param pointsCnt число точек, в которых надо посчитать значения
    /// \return Массив точек сплайна размером в pointsCnt
    static std::vector<Point> getSplineValues(std::vector<QuadPoly> const& spline, long
pointsCnt);
};
```

main.cpp

```
/// \brief Функция, принимающая табличные фукнции и возвращающая значение сплайнов
/// \param context[out] контекст запроса
/// \param request[in] табличные фукнции
/// \param response[out] переменная для отправки значений сплайнов
grpc::Status Evaluate(
        grpc::ServerContext* context,
        nm::TabFuncs const *request,
        grpc::ServerWriter<nm::Spline>* response)
```

Численный анализ методов

Для каждого графика искалось значение в 10000 точках на заданном интервале ($[-1; 1]$и $[0.1; 2]$).

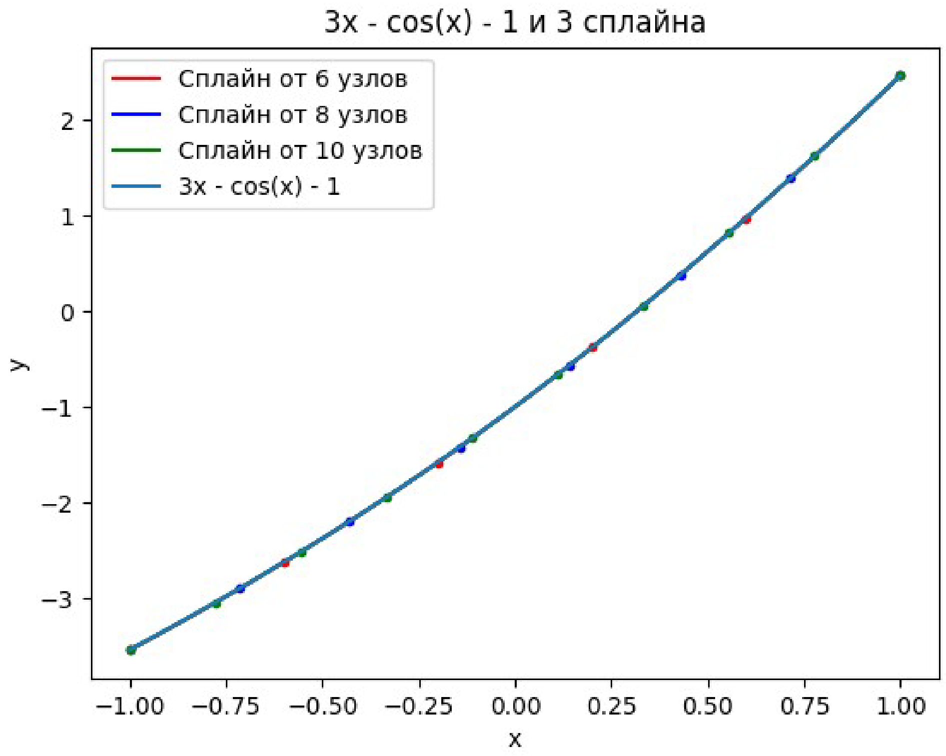

Рисунок 1. График зависимости 1-й функции и сплайнов

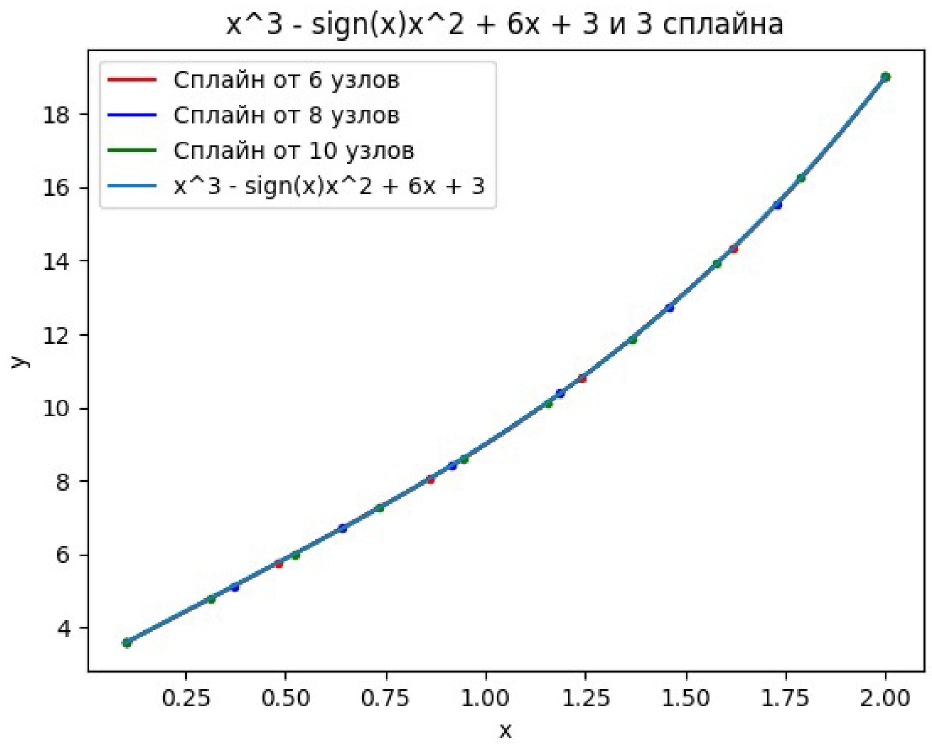

Рисунок 2. График зависимости 2-й функции и сплайнов

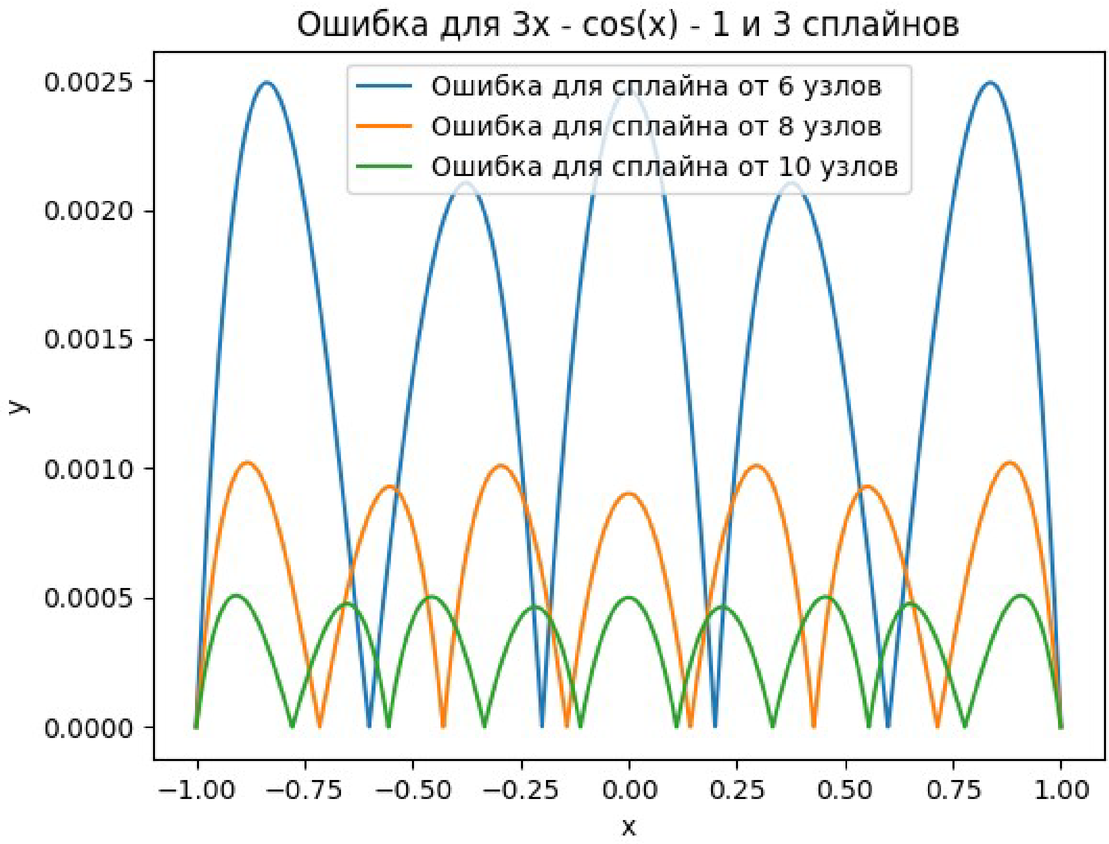

Рисунок 3. График зависимости ошибки значений 3-х сплайнов и искомой функции

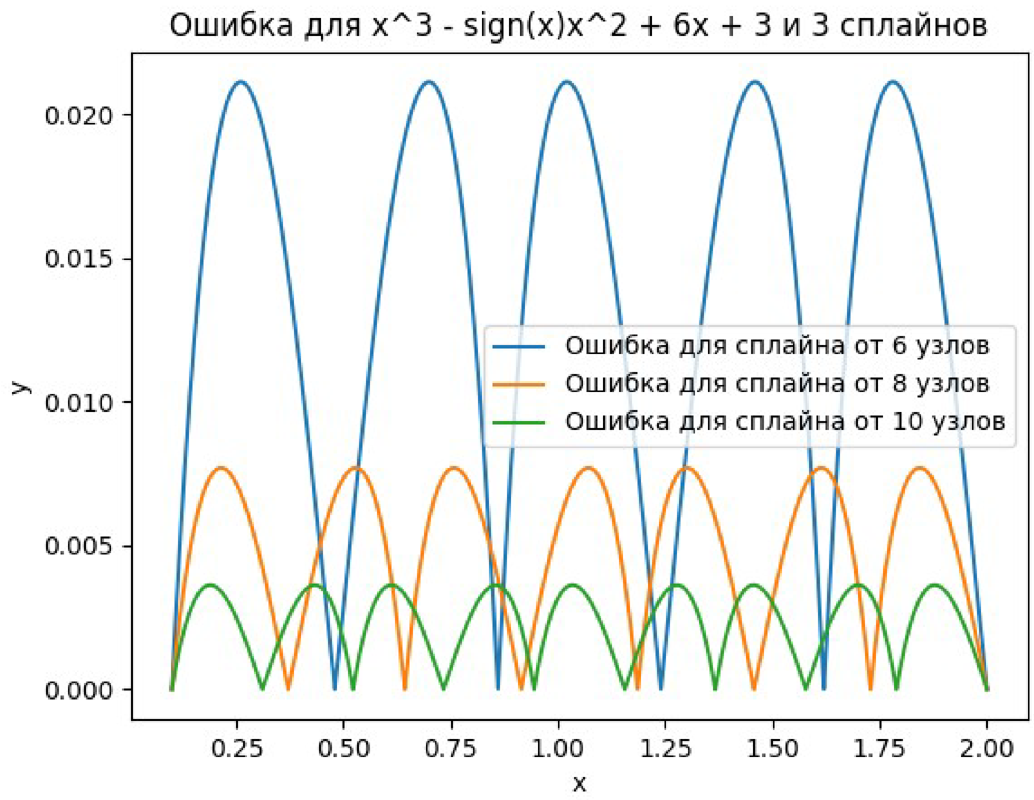

Рисунок 4. График зависимости ошибки значений 3-х сплайнов и искомой функции

Из полученных графиков 1–4 видно, что ошибка нулевая в узлах искомой сетки и достигает максимума в серединах между точками, а также не зависит от значения функции в точках и уменьшается с увеличением узлов в искомой сетке.

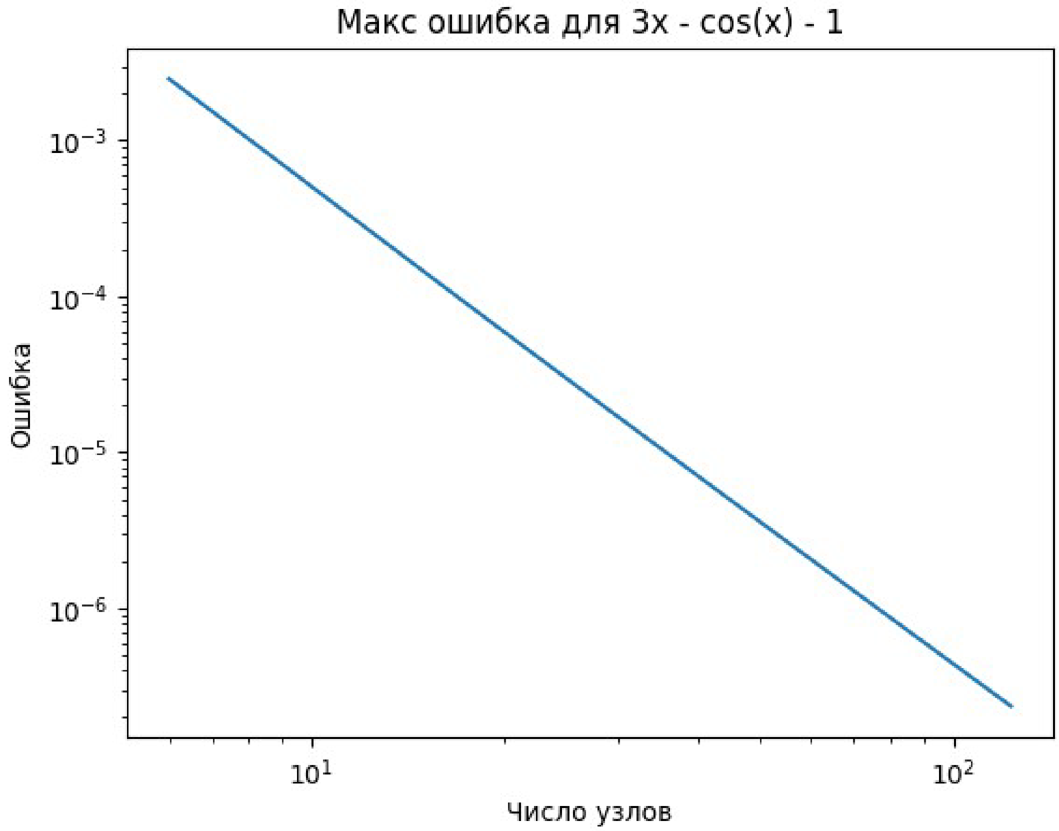

Рисунок 5. График зависимости максимальной ошибки сплайна на отрезке от числа узлов для 1-й функции

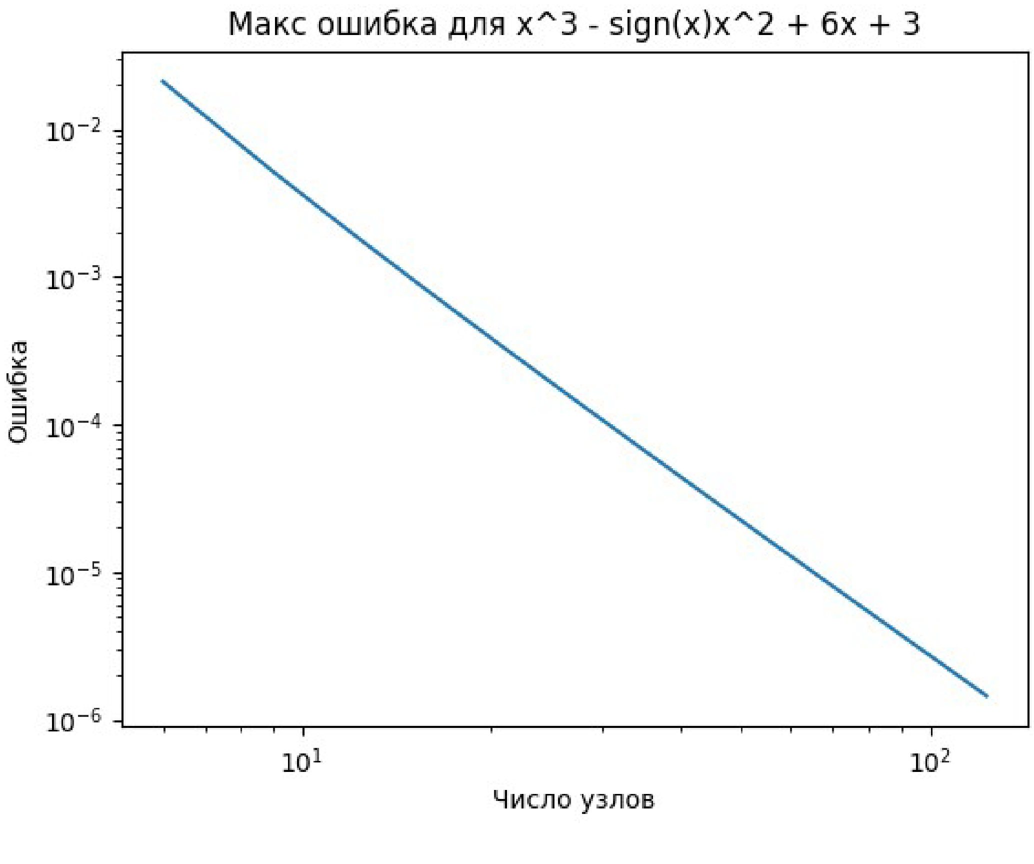

Рисунок 6. График зависимости максимальной ошибки сплайна на отрезке от числа узлов для 2-й функции

Из графиков 5 и 6 видно, ошибка очень быстро уменьшается с увеличением числа узлов (увеличение числа узлов на 10^2 уменьшает ошибку примерно на 10^{-4}).

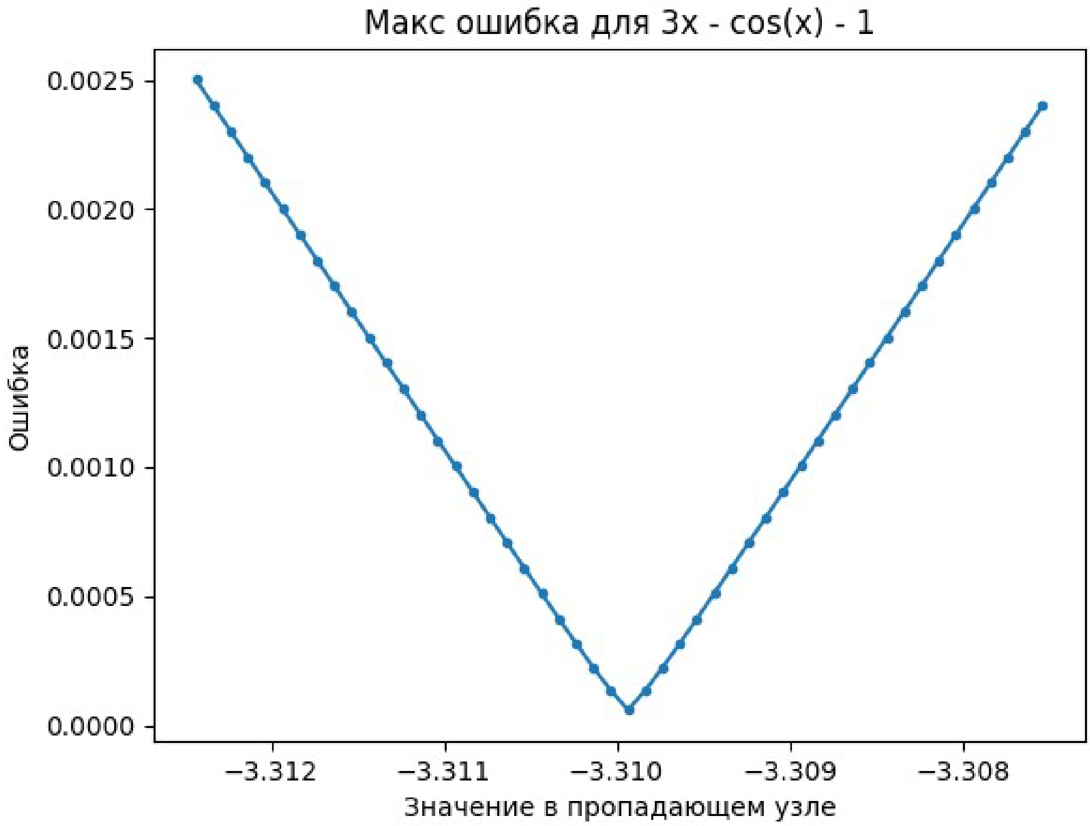

Рисунок 7. График зависимости максимальной ошибки сплайна на отрезке от ошибки в пропадающем узле для 1-й функции

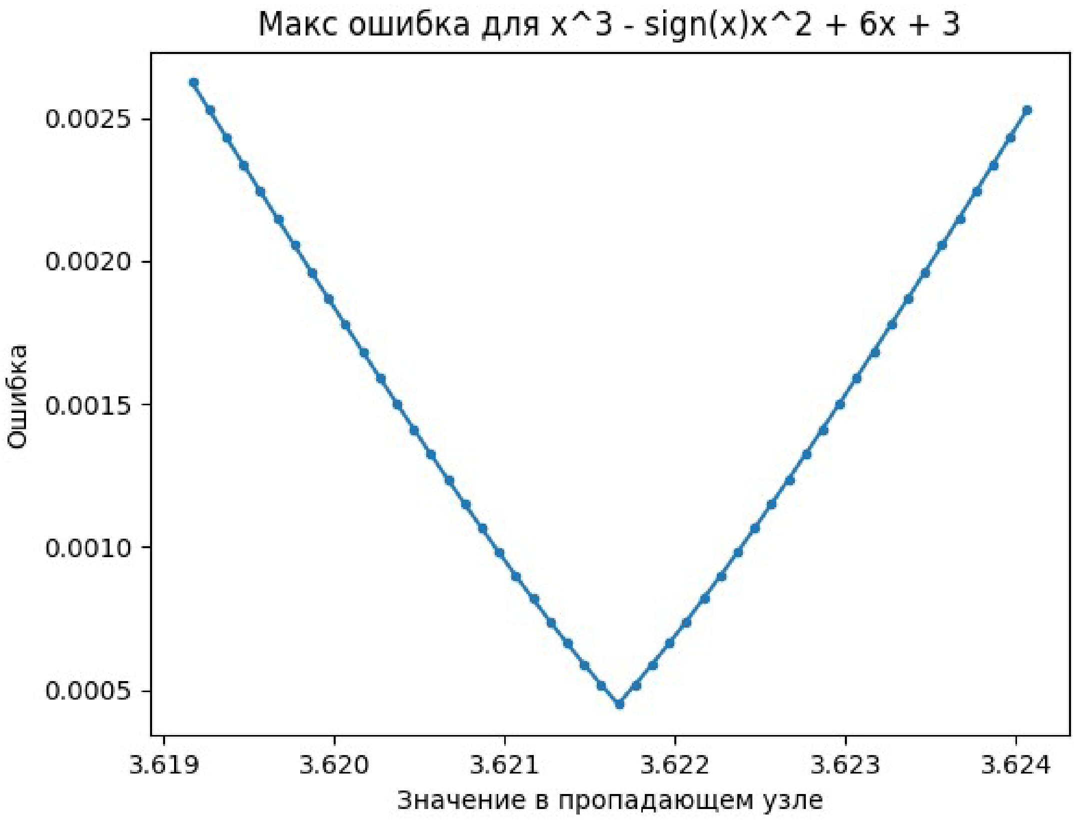

Рисунок 8. График зависимости максимальной ошибки сплайна на отрезке от ошибки в пропадающем узле для 2-й функции

Из графиков 7 и 8 видно, максимальная ошибка зависит линейно от модуля ошибки в пропадающем узле.

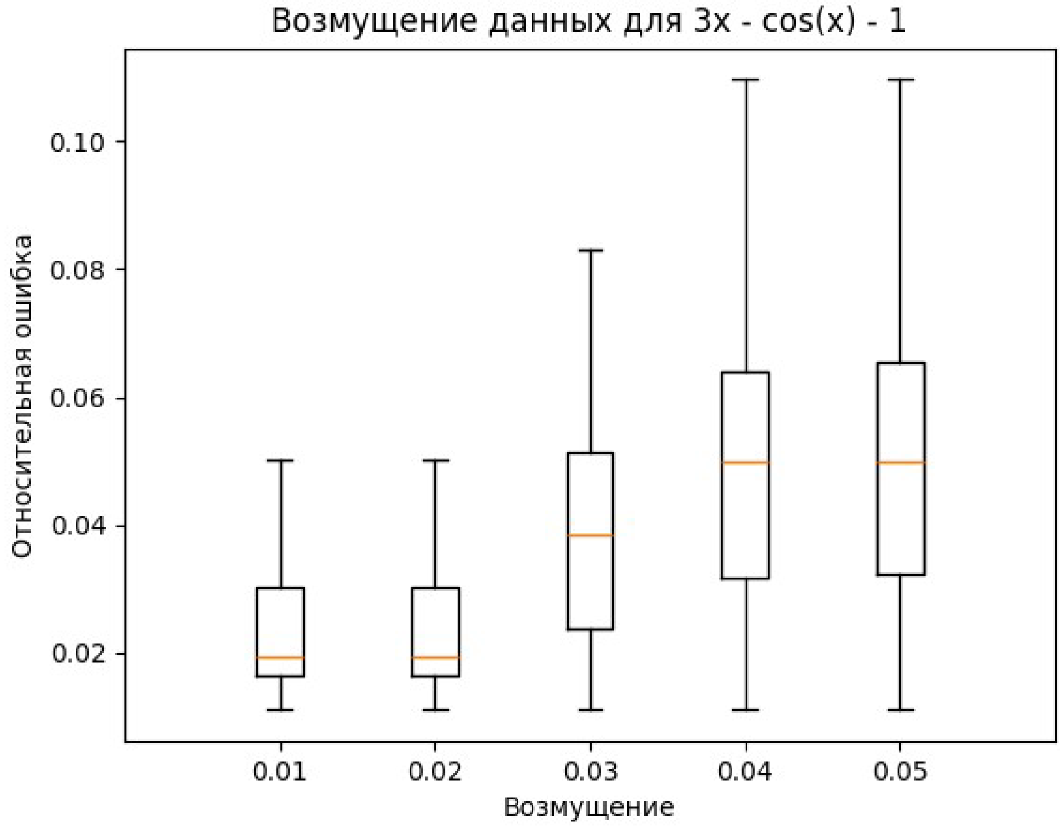

Рисунок 9. График зависимости возмущения значения функции (в долях) от максимальной относительной ошибки типа boxplot для 1й функции

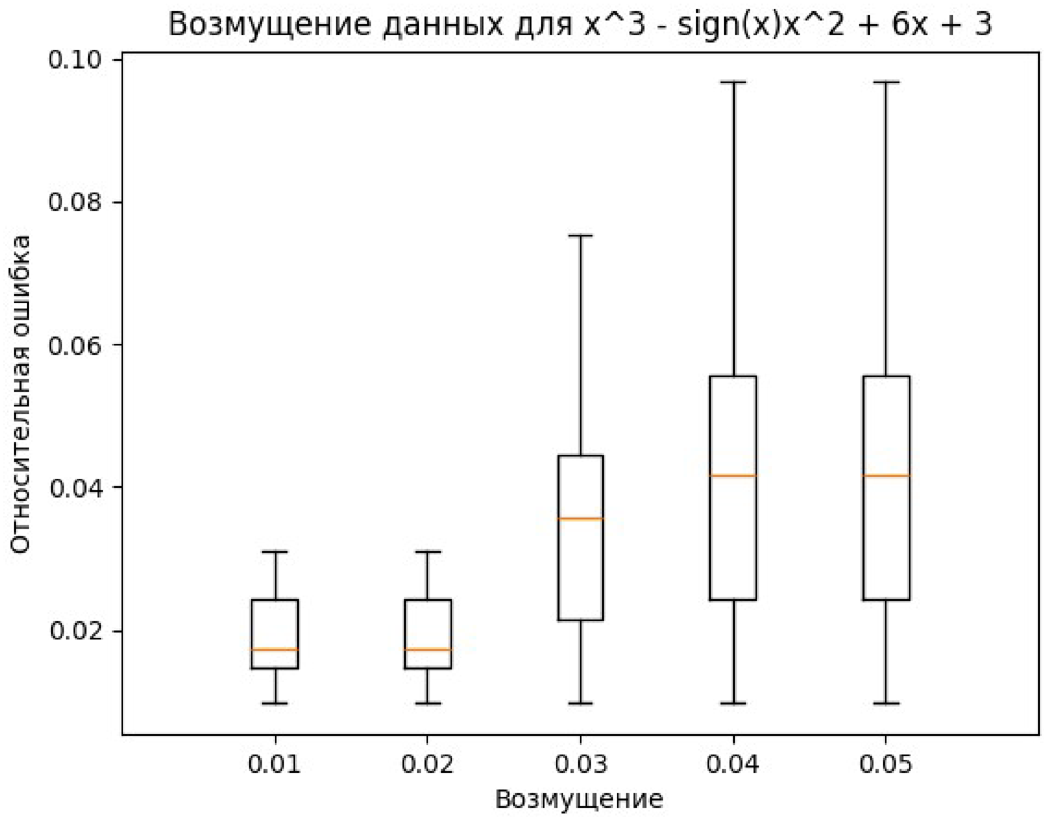

Рисунок 10. График зависимости возмущения значения функции (в долях) от максимальной относительной ошибки типа boxplot для 2й функции

На графиках видно, что с увеличением ошибки в значении функции возрастает как максимальная ошибка, так и средние значения ошибки. Причём, если смотреть на средние значения, то средняя относительная ошибка того же порядка, что и среднее возмущение, иногда даже меньше, что говорит нам о том, что использованный метод построения сплайна является устойчивым.

Вывод

В лабораторной работе мне удалось построить сплайн по табличным функциям, основанных на данных 2-х функциях, используя квадратичный сплайн с пропадающим узлом.

В ходе исследования была проанализирована зависимость ошибок от степеней полинома, которая показала, что уменьшение ошибки происходит в квадрат раз быстрее, чем увеличение числа узлов.

Также было установлено, что ошибка в вычислениях зависит примерно линейно от ошибки в значении в пропадающем узле. В том числе установлено, что при внесении ошибки во все значения табличной функции ошибка также увеличивается не сильно. Эти факты говорят о том, что метод построения сплайна является устойчивым.