Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и информатики

Лабораторная работа №8 по дисциплине

Численные методы

«Интерполяционные полиномы приближения табличных функций»

Выполнил

студент гр.5030102/20001 Дрекалов Н.С.

Преподаватель Козлов К.Н.

Санкт-Петербург

2023

Оглавление

[**Формулировка задачи** 3](#_Toc160617014)

[**Формализация** 3](#_Toc160617015)

[**Предварительный анализ задачи** 4](#_Toc160617016)

[**Тестовый пример к методам** 5](#_Toc160617017)

[**Контрольные тесты** 7](#_Toc160617018)

[**Модульная структура программы** 8](#_Toc160617019)

[C++ 8](#_Toc160617020)

[Python 10](#_Toc160617021)

[**Численный анализ методов** 11](#_Toc160617022)

[Равномерная сетка 11](#_Toc160617023)

[Чебышевская сетка 18](#_Toc160617024)

[**Вывод** 22](#_Toc160617025)

## **Формулировка задачи**

Необходимо для заданной табличной функции построить полином в форме Лагранжа и выполнить анализ графических результатов.

## **Формализация**

* Задана табличная функция
* Нужно построить интерполяционный полином таким образом, чтобы

**Интерполяционный полином в форме Лагранжа**

*Условия применимости:*

1. Полином всегда можно построить, если задана табличная функция для точек. Причём полином будет й степени.

*Алгоритм метода:*

1. Построим корневой полином по формуле
2. Построим полином в форме Лагранжа по формуле , для этого для слагаемого:
   1. Поделим полученный ранее на
   2. Домножим полином на , где

## **Предварительный анализ задачи**

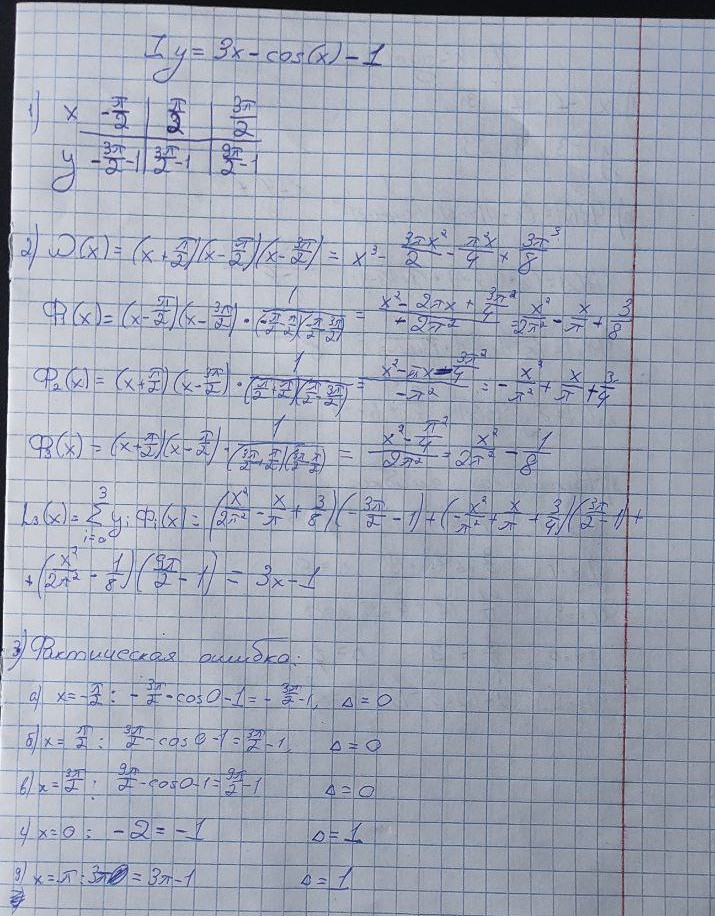
*Анализ функций на непрерывность:*

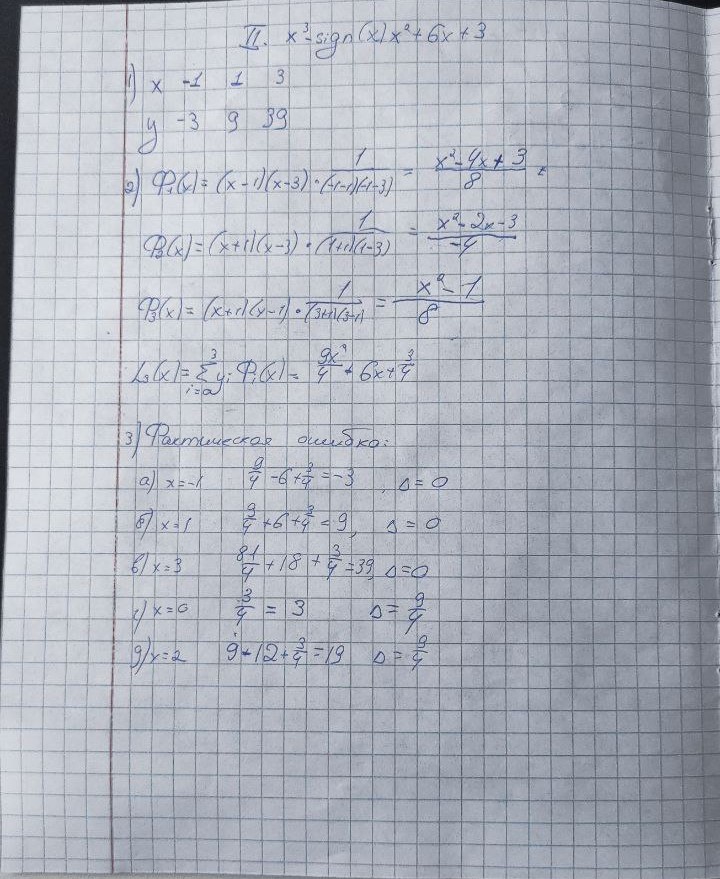
* непрерывна на  , поэтому можем рассматривать любой отрезок непрерывности (по умолчанию )
* также непрерывна на  , так как при . Однако есть некоторые сложности с расчётом производной в нуле (из-за sign(x), например, для 2-й производной справа: , а слева , разница между правосторонней и левосторонней производной не ноль). Поэтому будем рассматривать отрезок непрерывности

Анализ максимума модуля n-й производной:

* , начиная со второй производной, это либо , либо . Максимум этих функций – это 1
* , начиная с 4-й производной, это 0; т. е. для отрезков, не содержащих , полученный полином (с точностью до вычислительной ошибки) совпадает с

## **Тестовый пример к методам**





## **Контрольные тесты**

*Построим графики зависимостей для каждой из функций*

* Полиномов 5, 7 и 9-й степени, и функции от x на интервале [-1;1], а также линию теоретической ошибки для полинома 5-й степени
* Ошибки интерполяции от степени интерполяционного полинома. Степени будут в диапазоне 5...40
* Ошибки в выбранных точках от степени интерполяционного полинома. Будут выбраны две точки.

## **Модульная структура программы**

Выполнение нужных действий достигается с помощью 2-х программ: на языках python и C++. При этом C++ программа вычисляет коэффициенты по заданной табличной функции, в то время как python программа занимается построением этой табличной функции и анализом полученной C++ программой результата (расчёт различных ошибок, построение графиков и т. п.). Далее будут перечислены объявления в программах

### C++

poly.hpp

using PolyCoefs = std::vector<double>;  
  
// Класс полинома, упрощающий предавление полинома и преобразования над ним  
class Poly {  
 PolyCoefs coefficients;  
  
 static std::string formattedNumber(double number);  
public:  
 explicit Poly(PolyCoefs polyCoefs = **{**0.0**}**);  
 explicit Poly(size\_t degree);  
  
 [[nodiscard]] PolyCoefs const & getCoefs() const;  
 [[nodiscard]] size\_t degree() const;  
  
 // Различные переопределения операторов, позволяющие писать преобразования над полиномами в понятном виде и др.  
 // Ex. p = p1 + p2 вместо p = sumPoly(p1, p2)  
  
 Poly operator+(Poly const &poly) const;  
 Poly & operator+=(Poly const &poly);  
 Poly operator%(double root) const;  
 Poly operator\*(double number) const;  
 double & operator[](size\_t index);  
 double const &operator[](size\_t index) const;  
  
 void display() const;  
};

poly\_interpolator.hpp

using Grid = std::vector<double>;  
using GridFunc = std::vector<double>;  
  
// Класс для представления табличной функции  
class TabularFunc {  
public:  
 Grid grid;  
 GridFunc gridFunc;  
  
 void addPoint(double const x, double const y) {  
 grid.push\_back(x);  
 gridFunc.push\_back(y);  
 }  
};  
  
  
// Основной класс, ищущий коэффициенты полинома по заданной табличной функции  
class PolyInterpolator {  
 // Табличная функция  
 TabularFunc tabFunc;  
 // Путь к директории с файлами .tf .cf  
 const std::string dataDirPath = "../data";  
 // На случай, если в сетке будет 0  
 const double zeroReplacer = 0.000000000001;  
  
 /\*\*  
 \* \brief Функция, считающая корневой полином заданной табличной функции  
 \* \return Корневой полином  
 \*/  
 Poly findRootPoly();  
 /\*\*  
 \* \brief Функция, строящая полином по заданной табличной функции в форме Лагранжа  
 \* \return Интерполяционный полином  
 \*/  
 Poly findPolyLagrange();  
 /\*\*  
 \* \brief Функция, cчитающая w'(xk) (делитель в форме Лагранжа)  
 \* \return Делитель  
 \*/  
 double findDenomFor(size\_t index);  
public:  
 /\*\*  
 \* \brief Функция, cчитающая w'(xk) (делитель в форме Лагранжа)  
 \* \param postfix Уникальный постфикс файла in, содержащий в себе id функции и № параграфа  
 \* \return Делитель  
 \*/  
 explicit PolyInterpolator(std::string const &postfix);  
};

### Python

tab\_func.py

def func\_1(\_x: np.ndarray | float) -> np.ndarray | float

def func\_1\_str() -> str

def func\_2(\_x: np.ndarray | float) -> np.ndarray | float

def func\_2\_str() -> str

utility.py

def uni\_grid(left: float, right: float, point\_cnt: int) -> np.ndarray

def cheb\_grid(left: float, right: float, point\_cnt: int) -> np.ndarray

def write(func\_id: int, paragraph: int, exp\_cnt: int, min\_points\_cnt: int, step: int, left: float, right: float) -> None

def read\_written\_grid(func\_id: int, paragraph: int, exp\_num: int) -> list[float]

def read(func\_id: int, paragraph: int) -> list[list[float]]

p3.py

def get\_error\_y(x: np.ndarray | float, mn: float, n: int, grid: list[float]) -> np.ndarray | float

def print\_estimated\_error(func\_id: int, grid: list[float], left: float, right: float) -> None

def print\_func\_and\_polys(polys: list[list[float]], func: callable, left: float, right: float, func\_as\_str: str) -> None

def print\_error(polys: list[list[float]], func: callable, left: float, right: float, func\_as\_str: str) -> None

def analyze(func\_id: int, left: float, right: float) -> None

p4.y

def print\_max\_error(polys: list[list[float]], func: callable, left: float, right: float, func\_as\_str: str) -> None

def analyze(func\_id: int, left: float, right: float) -> None

p5.py

def print\_2\_dots\_error(polys: list[list[float]], func: callable, x1: float, x2: float, func\_as\_str: str) -> None

def analyze(func\_id: int, x1: float, x2: float) -> None

## **Численный анализ методов**

### Равномерная сетка

Для каждого графика искалось значение в 10000 точках на заданном интервале (, причем со сдвигом 0.000005 от краев, чтобы не было пересечения с искомой сеткой

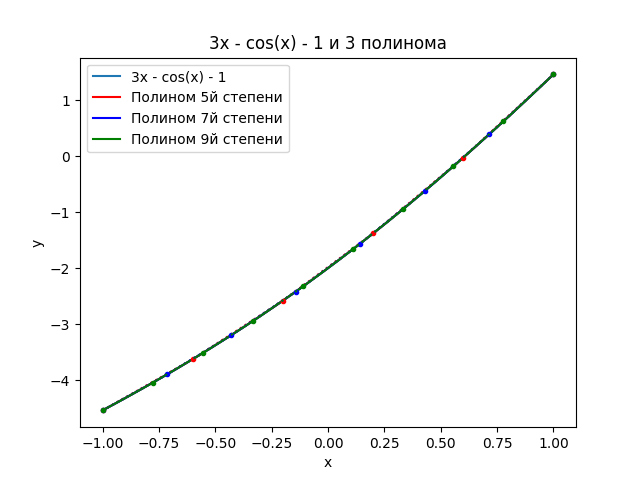


Рисунок 1. График зависимости 1-й функции и полиномов трех разных степеней

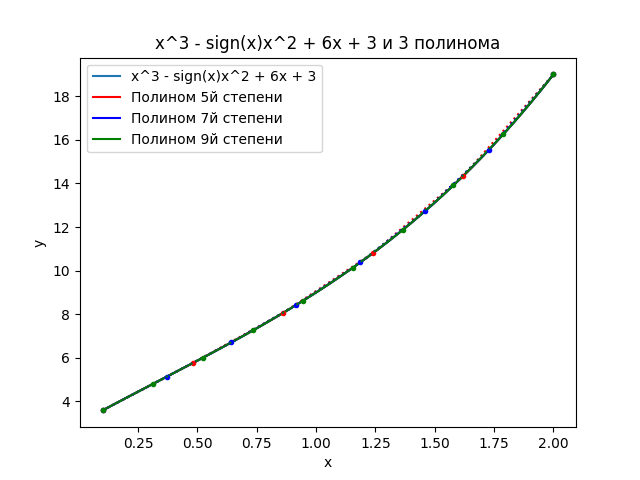


Рисунок 2. График зависимости 2-й функции и полиномов трех разных степеней

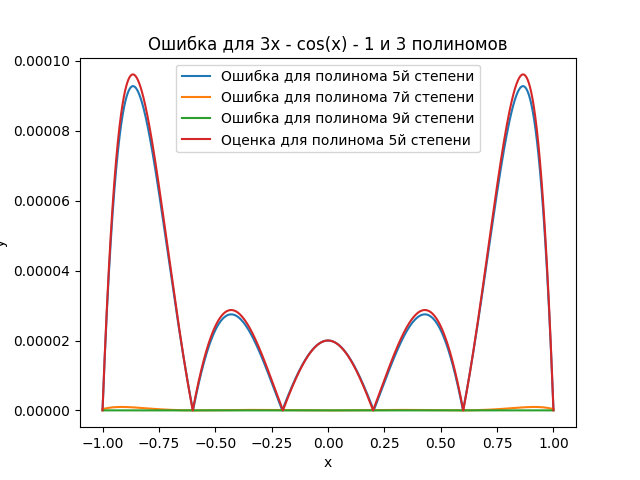


Рисунок 3. График зависимости ошибки значений 3-х полиномов и искомой функции, а также теоретической оценки ошибки для 1-й функции

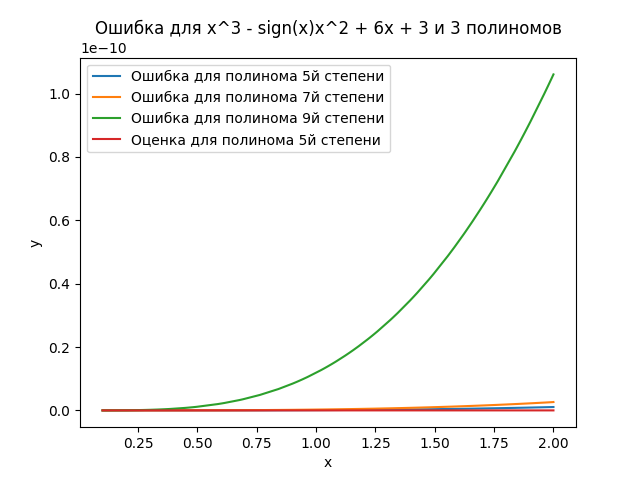


Рисунок 4. График зависимости ошибки значений 3-х полиномов и искомой функции, а также теоретической оценки ошибки для 1-й функции

Из полученных графиков 1–4 видно, что для относительно малого числа точек (6…10) ошибка интерполяции в форме Лагранжа незначительная. Также (из графиков 3 и 4) для полинома 5-й степени и 1-й функции линия теоретической ошибки совпадает с ожидаемой оценкой; для 2-й функции можно считать аналогично, так как ошибка порядка и обусловлена больше вычислительной ошибкой.

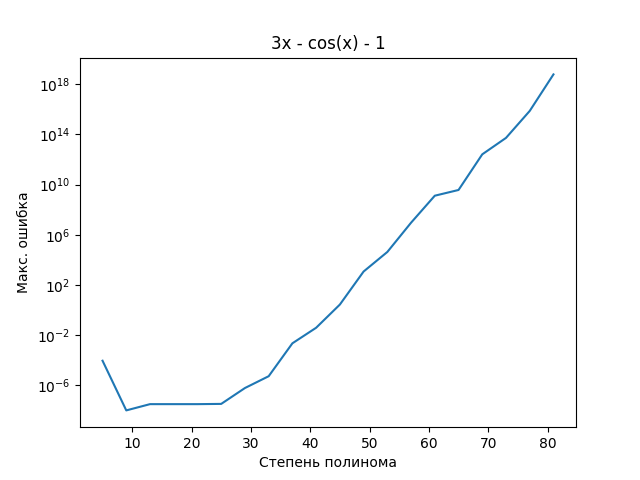


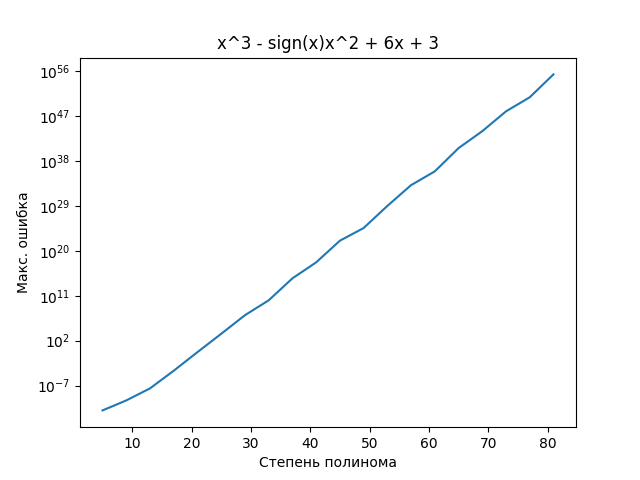
Рисунок 5. График зависимости максимальной ошибки полинома на отрезке от степени полинома для 1-й функции

Рисунок 6. График зависимости максимальной ошибки полинома на отрезке от степени полинома для 2-й функции

Из графиков 5 и 6 видно, что интерполяционный процесс расходится, причем у 1-й функции есть интервал сходимости до 15-й степени, а для 2-й функции максимальная ошибка растёт намного быстрее 1-й.

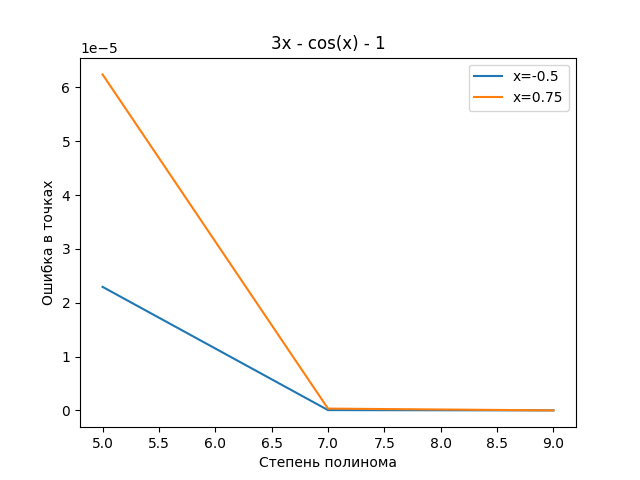


Рисунок 7. Зависимость ошибки в двух заданных точках от степени полинома для 1-й функции

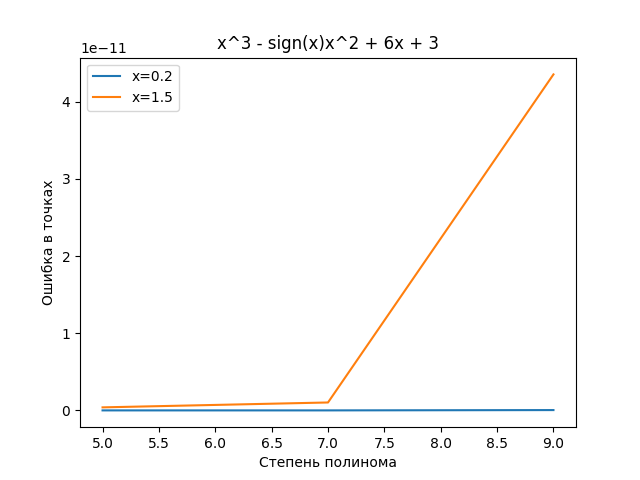


Рисунок 8. Зависимость ошибки в двух заданных точках от степени полинома для 2-й функции

Из графиков 7 и 8 видно, что для малых степеней полинома и 2-х отдельных точек процесс сходится и картина совпадает с общей картиной в графиках 3, 4, хотя 2-я функция уже для полинома 9-й степени выдает ошибку большую, чем у полинома 7-й степени

### Чебышевская сетка

Вместо равномерной сетки будем использовать Чебышевскую сетку (узлы Чебышева) – корни многочлена Чебышева первого рода. Общая формула для задания -го узла сетки из узлов на отрезке

Для такой сетки на тех же интервалах полученные многочлены давали такие графики:

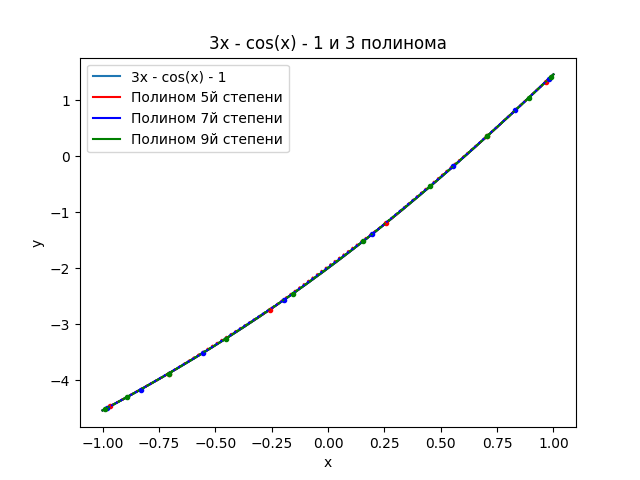


Рисунок 9. График зависимости 1-й функции и полиномов трех разных степеней при Чебышевской сетке

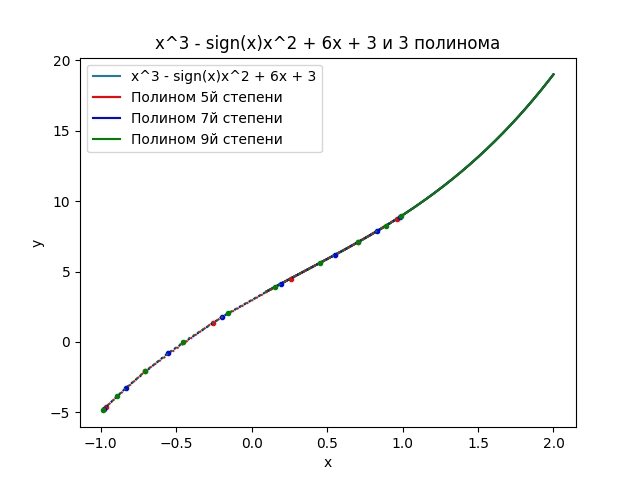


Рисунок 10. График зависимости 2-й функции и полиномов трех разных степеней при Чебышевской сетке

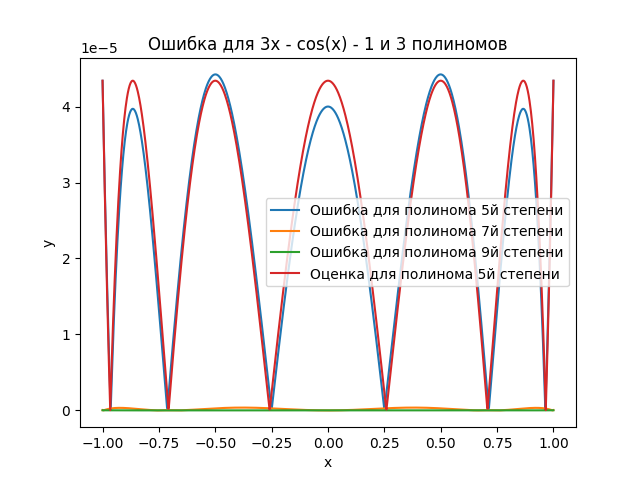


Рисунок 11. График зависимости ошибки значений 3-х полиномов и искомой функции, а также теоретической оценки ошибки для 1-й функции для сетки Чебышева

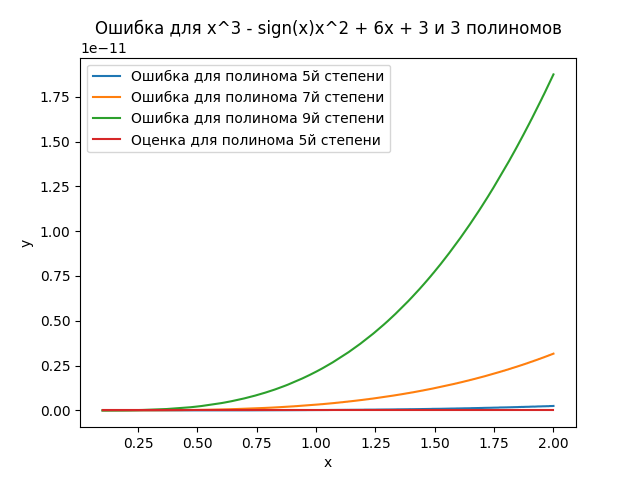


Рисунок 12. График зависимости ошибки значений 3-х полиномов и искомой функции, а также теоретической оценки ошибки для 2-й функции для сетки Чебышева

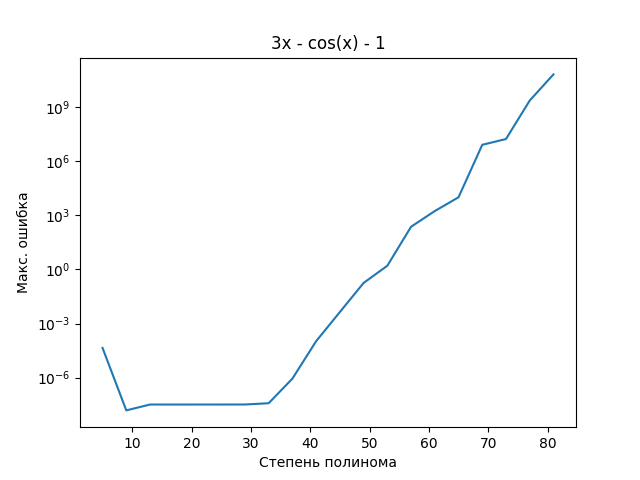


Рисунок 13. График зависимости максимальной ошибки полинома на отрезке от степени полинома для 1-й функции для сетки Чебышева

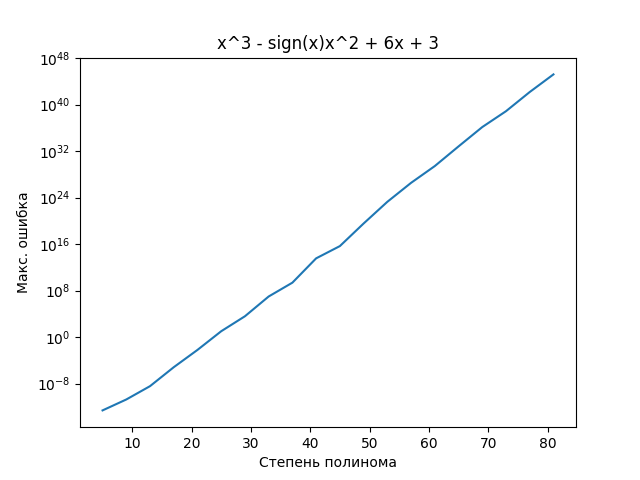


Рисунок 14. График зависимости максимальной ошибки полинома на отрезке от степени полинома для 2-й функции для сетки Чебышева

Сравнивая Рисунки 3, 4 с 11, 12, легко заметить, что ошибка уменьшилась в среднем на порядок для случая небольшого числа точек. А сравнивая Рисунки 5, 6 с 13, 14, можно также обнаружить, что уменьшение ошибки увеличивается с ростом самой ошибки (т. е. степени полинома), доходя до (т. е. уменьшение на 9 порядков)

## **Вывод**

В лабораторной работе мне удалось интерполировать табличные функции, основанных на данных 2-х функциях, методом построения интерполяционного полинома в форме Лагранжа.

В ходе исследования была проанализирована зависимость ошибок от степеней полинома, которая показала, что для обеих функций интерполяционный процесс расходится, так как ошибка увеличивается с увеличением степени полинома.

Также было установлено, что при сетке Чебышева наблюдается значительный прирост точности (от 1 до 9 порядков, в зависимости от «начальной» ошибки)