Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Учебная практика**

Реализация процедуры интерполяции дискретным сплайном 3 степени

Выполнил

студент гр. 3530901/10005

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тучков Д.А.

(подпись)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Макаренко А.А.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Санкт-Петербург   
2023

Содержание

[Техническое задание 3](#_Toc138460783)

[**1.** **Теория** 4](#_Toc138460784)

[**2.** **Реализация технического задания** 5](#_Toc138460785)

[**3.** **Код программы** 8](#_Toc138460786)

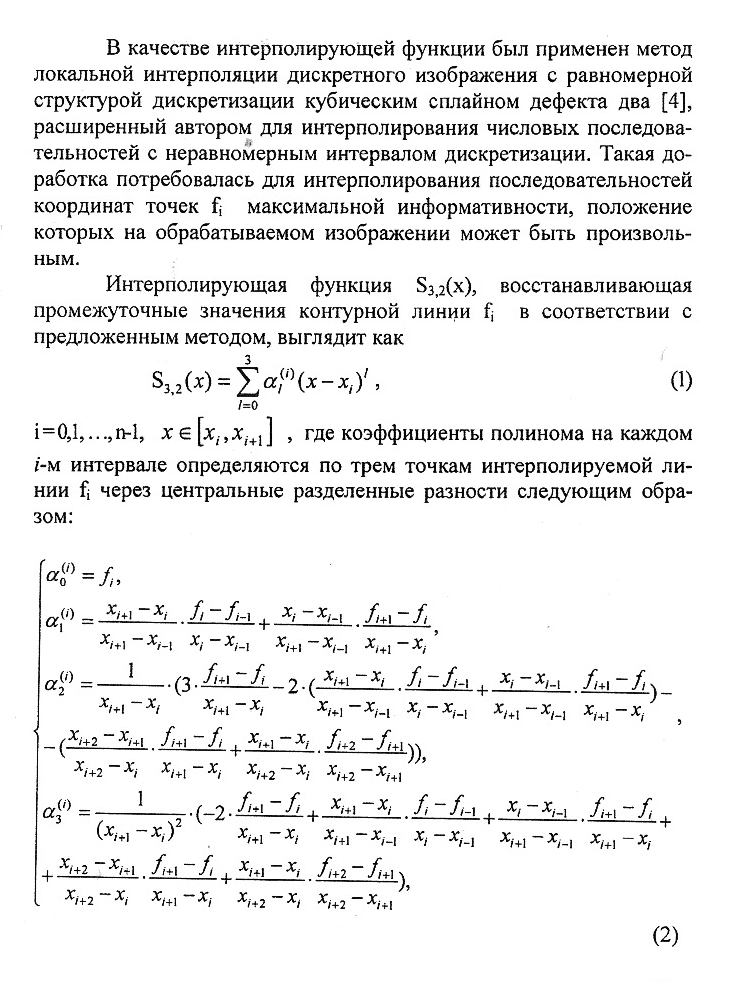
[Вывод данных в терминал 8](#_Toc138460787)

[Вывод графика через gnuplot 11](#_Toc138460788)

[**4.** **Руководство пользователя** 14](#_Toc138460789)

[Вывод 17](#_Toc138460790)

# Техническое задание



1. **Теория**

* Интерполяция - это процесс восстановления промежуточных значений функции на основе известных значений в заданных точках. Она используется для аппроксимации функции или данных между известными узлами или точками.

В основе интерполяции лежит предположение о том, что между двумя близкими точками можно найти функцию, которая проходит через эти точки. Интерполяционная функция может быть полиномиальной, сплайновой, тригонометрической или иной формы, в зависимости от метода интерполяции и природы данных.

Основная цель интерполяции - получить значения функции в промежуточных точках, чтобы заполнить пробелы между имеющимися данными или создать гладкую аппроксимацию функции.

Интерполяция широко применяется в различных областях, таких как наука, инженерия, компьютерная графика, статистика и другие. Некоторые известные методы интерполяции включают полиномиальную интерполяцию Лагранжа, интерполяцию Ньютона, сплайн-интерполяцию, кубическую интерполяцию и другие.

* Дискретный сплайн третьей степени (также известный как кубический сплайн) - это метод интерполяции, при котором функция аппроксимируется кусочно-полиномиальными сегментами третьей степени. Он используется для восстановления непрерывной функции по её дискретным значениям (узлам) с использованием кубических полиномов на каждом интервале между соседними узлами.

Характеристики дискретного сплайна третьей степени:

1. Кусочно-полиномиальная функция: Для каждого интервала между двумя соседними узлами строится отдельный полином третьей степени, который аппроксимирует функцию на этом интервале. Таким образом, функция разбивается на сегменты, и на каждом сегменте используется кубический полином.

2. Гладкость: Дискретный сплайн третьей степени обеспечивает гладкость функции внутри каждого сегмента, то есть первая и вторая производные функции непрерывны на всей области определения.

3. Интерполяция: Дискретный сплайн третьей степени проходит через все заданные узлы (дискретные значения функции), что обеспечивает точность интерполяции в этих точках.

4. Граничные условия: Для построения дискретного сплайна третьей степени необходимо задать граничные условия, такие как естественные условия (нулевые кривизны на краях), условия с фиксированными производными на концах или периодические условия.

Дискретные сплайны третьей степени широко используются в численных методах и аппроксимации функций, особенно когда требуется гладкость и точность интерполяции между узлами. Они являются одним из наиболее распространенных типов сплайнов.

1. **Реализация технического задания**

В качестве интерполирующей функции был применен метод локальной интерполяции дискретного изображения с равномерной структурой дискретизации кубическим сплайном дефекта два [4], расширенный автором для интерполирования числовых последова тельностей с неравномерным интервалом дискретизации. Такая до работка потребовалась для интерполирования последовательностей координат точек f максимальной информативности, положение которых на обрабатываемом изображении может быть произволь ным.

    Интерполирующая функция Ѕ3,2(x), восстанавливающая промежуточные значения контурной линии в соответствии с предложенным методом, выглядит как

    S(3,2)(x) = интервал от 3 до i=0 а1^(i)(x-xi)^i

    i= 0,1,...,n-1, x принадлежит [xi, xi+1], где коэффициенты полинома на каждом i-м интервале определяются по трем точкам интерполируемой линии Fi через централтные разделенные разности следующим образом:

    a0^(i) = fi,

    a1^(i) = (((xi+1) - (xi))/((xi+1)-(xi-1))) \* ((f1 - (fi-1))/(xi-(xi-1))) + (xi-(xi-1)) / ((xi+1) -(xi-1)) \* (((fi+1) - fi) / (xi+1 - xi)),

    a2^(i) = ((1) / ((xi+1) - (xi))) \* 3\*((fi+1 - fi) / ((xi+1) - (xi)) - 2\*((xi+1 - xi)/((xi+1) - (xi-1)) \* \*((fi) - (fi-1)) / ((xi) - (xi-1)) / ((xi+1) - (xi-1)) \* (((fi+1) - (fi)) / ((xi+1) - (xi)))) - ((((xi+2) - (xi+1)) / ((xi+2)-(xi))) \* ((fi+1 - fi) / ((xi+1) - (xi))) + ((xi+1 - xi) / ((xi+2) - (xi))) \* (((fi+2) - (fi+1)) / ((xi+2) - (xi+1))))),

    a3^(i) = ((1)/((xi+1) - (xi))^2) / ( -2\* (((fi+1) - (fi)) / ((xi+1) - (xi))) + ((xi+1 - (xi)) / ((xi+1) - (xi-1))) \* ((fi) - (fi-1)) / ((xi) -(xi-1)) + ((xi) - (xi-1)) + (((xi)-(xi-1)) / (xi+1) - (xi-1)) \* (((fi+1)  - (fi)) / ((xi+1)  - (xi))) + ((xi+2) - (xi+1)) / (((xi+2)  - (xi)) \* ((xi+2)  - (xi)) ) \* (((fi+1) - (fi)) / (((xi+1) - (xi))) + ((xi+1) - (xi)) / ((xi+2) - (xi))) \* ((fi+2) - (fi+1)) / ((xi+2) - (xi+1)))

    Файл tzVersion.c

    Заданные точки xi являются упорядоченными по возрастанию.

    Заданные точки xi и соответствующие значения fi.

    Интерполируем значения в numInterpolatedPoints равномерно распределенных точках между минимальным и максимальным значением xi.

    Для каждой интерполированной точки используем функцию cubicSplineInterpolation для вычисления интерполированного значения.

    Результат выводится на экран.

    Вывод графика при помощи gnuplot файл tzVersionGrafic.c

        Код открывает пайп к gnuplot с помощью функции popen и отправляет команды для построения графика.

        График строится с использованием линий (with lines).

        Построенные точки интерполируются с помощью функции cubicSplineInterpolation.

        После отправки данных для построения графика, команда e указывает gnuplot на завершение ввода данных.

1. **Код программы**

Были выполнены различные варианты реализации программы.

* **Вывод результатов в консоль (файл tzVersion.c)**

#include <stdio.h>

// Функция для вычисления интерполированного значения

double cubicSplineInterpolation(double x, double xi[], double fi[], int n) {

    int i;

    for (i = 0; i < n - 1; i++) {

        if (x >= xi[i] && x <= xi[i + 1]) {

            break;

        }

    }

    double h1 = xi[i + 1] - xi[i];

    double h2 = xi[i + 2] - xi[i + 1];

    double delta1 = (fi[i + 1] - fi[i]) / h1;

    double delta2 = (fi[i + 2] - fi[i + 1]) / h2;

    double a0 = fi[i];

    double a1 = (((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i + 1] - fi[i - 1]) / (xi[i] - xi[i - 1]))) +

                (((xi[i] - xi[i - 1]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])));

    double a2 = (1 / (xi[i + 1] - xi[i])) \*

                (3 \* ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) -

                 2 \* ((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])) -

                 (((xi[i + 2] - xi[i + 1]) / (xi[i + 2] - xi[i])) \*

                  ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1]))));

    double a3 = (1 / ((xi[i + 1] - xi[i]) \* (xi[i + 1] - xi[i]))) /

                (-2 \* ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                 ((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])) +

                 (xi[i] - xi[i - 1]) +

                 (((xi[i] - xi[i - 1]) / (xi[i + 1] - xi[i - 1])) \*

                  ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1])) \*

                      ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1])) \*

                      ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i]))));

    double result = 0;

    double xPow = 1;

    for (int j = 0; j < 4; j++) {

        result += (a0 + a1 \* xPow + a2 \* xPow \* xPow + a3 \* xPow \* xPow \* xPow);

        xPow \*= (x - xi[i]);

    }

    return result;

}

int main() {

    // Задание количества точек

    int n;

    printf("vvedite kollichestvo tochek: ");

    scanf("%d", &n);

    // Ввод точек xi

    double xi[n];

    printf("vvedite znacchenie tochek xi:\n");

    for (int i = 0; i < n; i++) {

        scanf("%lf", &xi[i]);

    }

    // Ввод значений fi

    double fi[n];

    printf("vvedite znachenie in zadannix tochkax fi:\n");

    for (int i = 0; i < n; i++) {

        scanf("%lf", &fi[i]);

    }

    // Интерполирование

    int numInterpolatedPoints = 10;            // Количество интерполированных точек

    double stepSize = (xi[n - 1] - xi[0]) / (numInterpolatedPoints - 1);

    printf("Interpolated values:\n");

    for (int i = 0; i < numInterpolatedPoints; i++) {

        double interpolatePoint = xi[0] + i \* stepSize;

        double interpolatedValue = cubicSplineInterpolation(interpolatePoint, xi, fi, n);

        printf("x=%.2f, f=%.2f\n", interpolatePoint, interpolatedValue);

    }

    system ("pause");

    return 0;

}

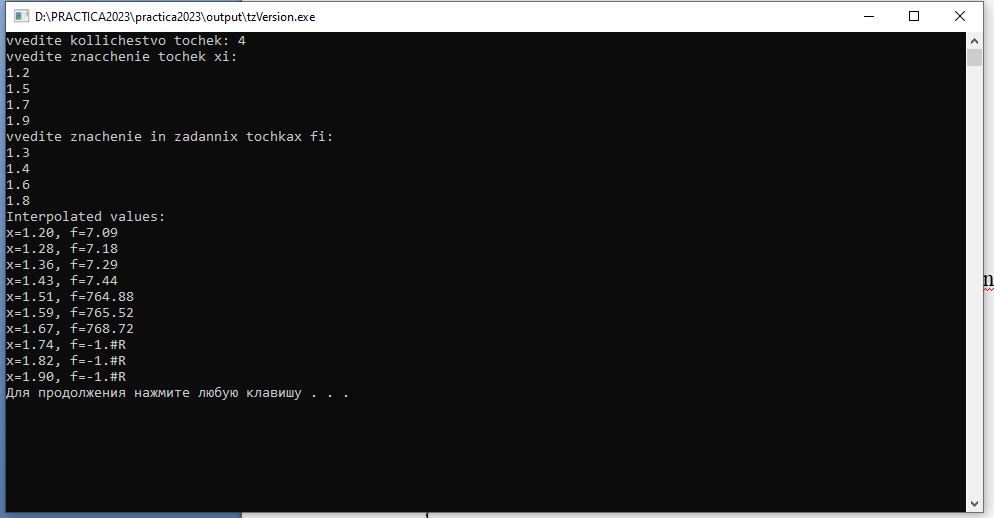


Рис.1 Запуск программы с exe файла

* **Вывод результатов при помощи графика, необходима установка gnuplot (файл tzVersionGraficGnuplot.c)**

#include <stdio.h>

#include <stdlib.h>

// Функция для вычисления интерполированного значения

double cubicSplineInterpolation(double x, double xi[], double fi[], int n) {

    int i;

    for (i = 0; i < n - 1; i++) {

        if (x >= xi[i] && x <= xi[i + 1]) {

            break;

        }

    }

    double h1 = xi[i + 1] - xi[i];

    double h2 = xi[i + 2] - xi[i + 1];

    double delta1 = (fi[i + 1] - fi[i]) / h1;

    double delta2 = (fi[i + 2] - fi[i + 1]) / h2;

    double a0 = fi[i];

    double a1 = (((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i + 1] - fi[i - 1]) / (xi[i] - xi[i - 1]))) +

                (((xi[i] - xi[i - 1]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])));

    double a2 = (1 / (xi[i + 1] - xi[i])) \*

                (3 \* ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) -

                 2 \* ((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])) -

                 (((xi[i + 2] - xi[i + 1]) / (xi[i + 2] - xi[i])) \*

                  ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1]))));

    double a3 = (1 / ((xi[i + 1] - xi[i]) \* (xi[i + 1] - xi[i]))) /

                (-2 \* ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                 ((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])) +

                 (xi[i] - xi[i - 1]) +

                 (((xi[i] - xi[i - 1]) / (xi[i + 1] - xi[i - 1])) \*

                  ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1])) \*

                      ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1])) \*

                      ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i]))));

    double result = 0;

    double xPow = 1;

    for (int j = 0; j < 4; j++) {

        result += (a0 + a1 \* xPow + a2 \* xPow \* xPow + a3 \* xPow \* xPow \* xPow);

        xPow \*= (x - xi[i]);

    }

    return result;

}

int main() {

    // Ввод количества точек

    int n;

    printf("vvedite collichestvo tochek: ");

    scanf("%d", &n);

    // Выделение памяти для массивов

    double\* xi = malloc(n \* sizeof(double));

    double\* fi = malloc(n \* sizeof(double));

    // Ввод точек и значений

    printf("vvedite znachenie tochek xi and  znachenie fi:\n");

    for (int i = 0; i < n; i++) {

        scanf("%lf %lf", &xi[i], &fi[i]);

    }

    // Интерполирование

    int numInterpolatedPoints = 10;            // Количество интерполированных точек

    double stepSize = (xi[n - 1] - xi[0]) / (numInterpolatedPoints - 1);

    FILE\* gnuplotPipe = popen("gnuplot -persistent", "w");

    fprintf(gnuplotPipe, "plot '-' with lines\n");

    for (int i = 0; i < numInterpolatedPoints; i++) {

        double interpolatePoint = xi[0] + i \* stepSize;

        double interpolatedValue = cubicSplineInterpolation(interpolatePoint, xi, fi, n);

        fprintf(gnuplotPipe, "%f %f\n", interpolatePoint, interpolatedValue);

    }

    fprintf(gnuplotPipe, "e\n");

    pclose(gnuplotPipe);

    // Освобождение памяти

    free(xi);

    free(fi);

    return 0;

}

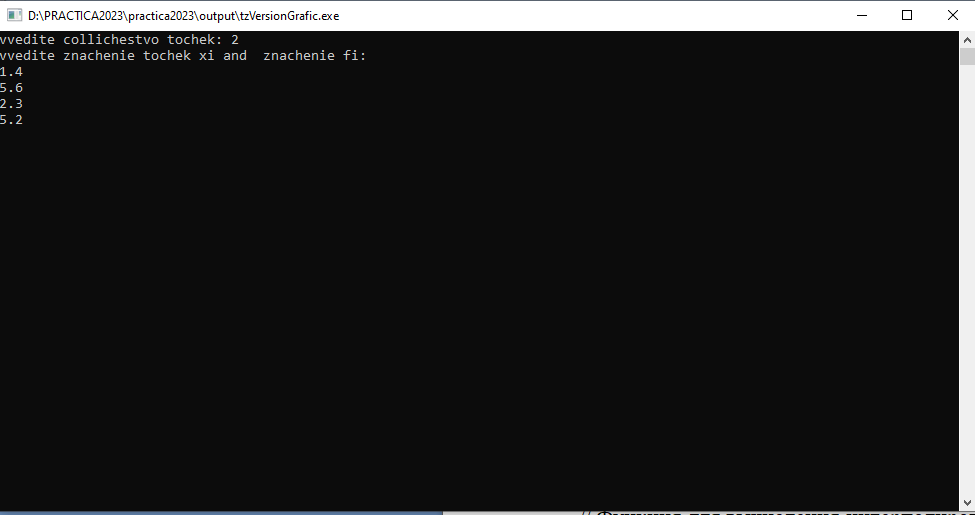
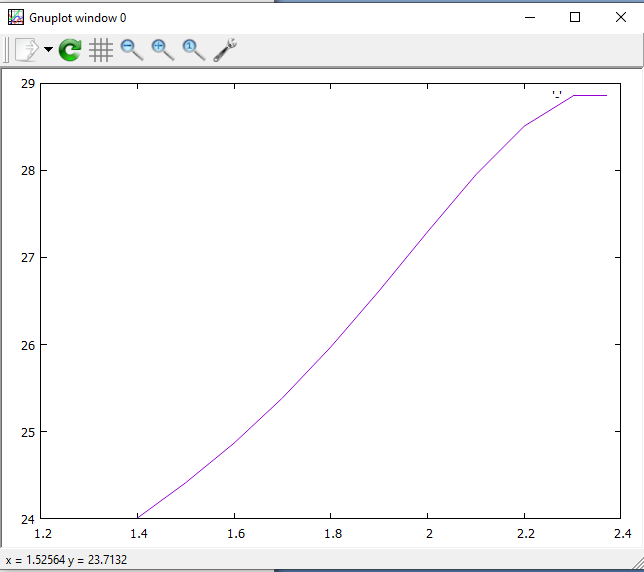
 

Рис.2 Запуск программы с exe файла

* **Вывод результатов в текстовый файл (язык С), используя библиотеку python для рисования графиков matplotlib (каталог usinPython (grafic.c и plot\_graph.py)).**

### Код grafic.c

#include <stdio.h>

#include <stdlib.h>

// Функция для вычисления интерполированного значения

double cubicSplineInterpolation(double x, double xi[], double fi[], int n) {

     int i;

    for (i = 0; i < n - 1; i++) {

        if (x >= xi[i] && x <= xi[i + 1]) {

            break;

        }

    }

    double h1 = xi[i + 1] - xi[i];

    double h2 = xi[i + 2] - xi[i + 1];

    double delta1 = (fi[i + 1] - fi[i]) / h1;

    double delta2 = (fi[i + 2] - fi[i + 1]) / h2;

    double a0 = fi[i];

    double a1 = (((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i + 1] - fi[i - 1]) / (xi[i] - xi[i - 1]))) +

                (((xi[i] - xi[i - 1]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])));

    double a2 = (1 / (xi[i + 1] - xi[i])) \*

                (3 \* ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) -

                 2 \* ((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])) -

                 (((xi[i + 2] - xi[i + 1]) / (xi[i + 2] - xi[i])) \*

                  ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1]))));

    double a3 = (1 / ((xi[i + 1] - xi[i]) \* (xi[i + 1] - xi[i]))) /

                (-2 \* ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                 ((xi[i + 1] - xi[i]) / (xi[i + 1] - xi[i - 1])) \* ((fi[i] - fi[i - 1]) / (xi[i] - xi[i - 1])) +

                 (xi[i] - xi[i - 1]) +

                 (((xi[i] - xi[i - 1]) / (xi[i + 1] - xi[i - 1])) \*

                  ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1])) \*

                      ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i])) +

                  ((xi[i + 1] - xi[i]) / (xi[i + 2] - xi[i])) \* ((fi[i + 2] - fi[i + 1]) / (xi[i + 2] - xi[i + 1])) \*

                      ((fi[i + 1] - fi[i]) / (xi[i + 1] - xi[i]))));

    double result = 0;

    double xPow = 1;

    for (int j = 0; j < 4; j++) {

        result += (a0 + a1 \* xPow + a2 \* xPow \* xPow + a3 \* xPow \* xPow \* xPow);

        xPow \*= (x - xi[i]);

    }

    return result;

}

int main() {

    // Ввод количества точек и значений

    int n;

    printf("vvedite collichestvo tochek:: ");

    scanf("%d", &n);

    // Выделение памяти для массивов

    double\* xi = malloc(n \* sizeof(double));

    double\* fi = malloc(n \* sizeof(double));

    // Ввод точек и значений

    printf("vvedite znachenie tochek xi and  znachenie fi: \n");

    for (int i = 0; i < n; i++) {

        scanf("%lf %lf", &xi[i], &fi[i]);

    }

    // Интерполирование

    int numInterpolatedPoints = 10;            // Количество интерполированных точек

    double stepSize = (xi[n - 1] - xi[0]) / (numInterpolatedPoints - 1);

    // Создание файла для сохранения данных

    FILE\* dataFile = fopen("data.txt", "w");

    if (dataFile == NULL) {

        printf("error open file.\n");

        return 1;

    }

    // Сохранение данных в файл

    for (int i = 0; i < numInterpolatedPoints; i++) {

        double interpolatePoint = xi[0] + i \* stepSize;

        double interpolatedValue = cubicSplineInterpolation(interpolatePoint, xi, fi, n);

        fprintf(dataFile, "%f %f\n", interpolatePoint, interpolatedValue);

    }

    fclose(dataFile);

    // Освобождение памяти

    free(xi);

    free(fi);

    // Выполнение команды Python для построения графика

    system("python D:\\PRACTICA2023\\practica2023\\usinPyton\\plot\_graph.py");

    return 0;

}

### Код plot\_graph.py

import matplotlib.pyplot as plt

# Чтение данных из файла

data = []

with open("data.txt", "r") as file:

    for line in file:

        x, y = line.split()

        data.append((float(x), float(y)))

# Извлечение координат X и Y

x\_coords, y\_coords = zip(\*data)

# Построение графика

plt.plot(x\_coords, y\_coords)

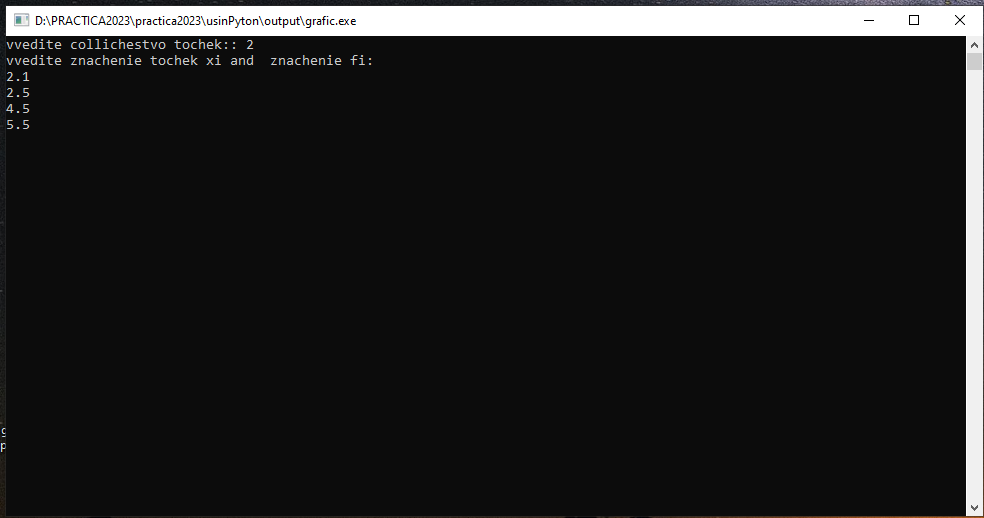
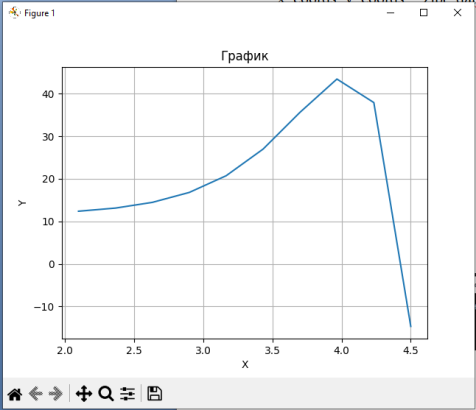
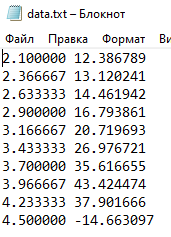
plt.xlabel("X")

plt.ylabel("Y")

plt.title("График")

plt.grid(True)

plt.show()

1. **Руководство пользователя**

Понадобится несколько программ:

1. Visual Studio Code (или другой редактор)
2. MinGW
3. Gnuplot
4. Python

Далее приступаем к установке:

Для начала подготовим VS Code

Скачиваем его с официального сайта. Он бесплатный это главный критерий в нашем случае.

Выбираем нашу платформу и устанавливаем

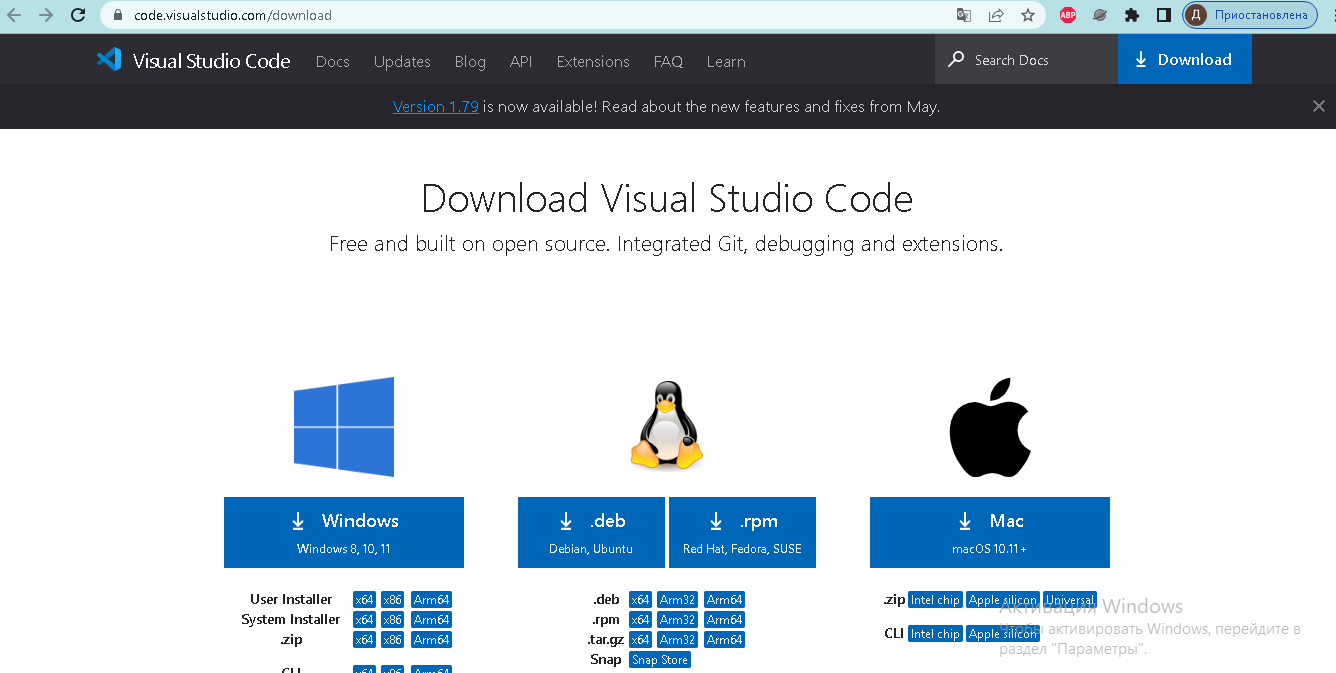


Рис. 3 Скачивание VS Code

Заходим в VS Code

Необходимо установить Extention, так как мы реализуем код на языке С, то устанавливаем соответствующее приложение.

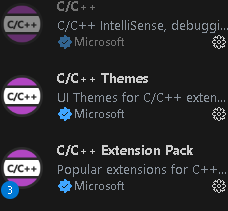


Рис. 4 Установка Extention

Переходим к среде разработки ПО MinGW

Скачиваем

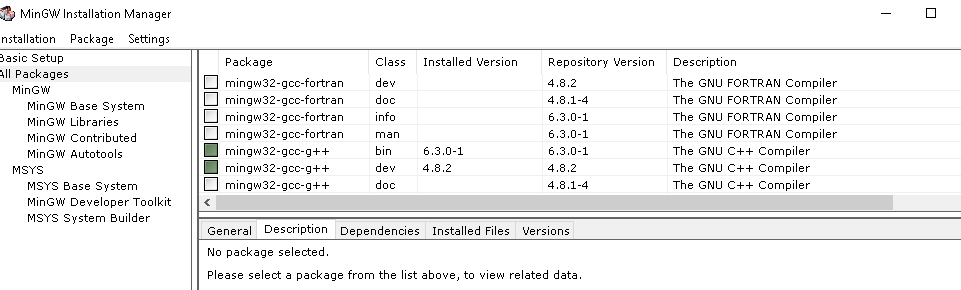


Рис.5 MinGW

Заходим в переменные среды и кидаем на него ссылку

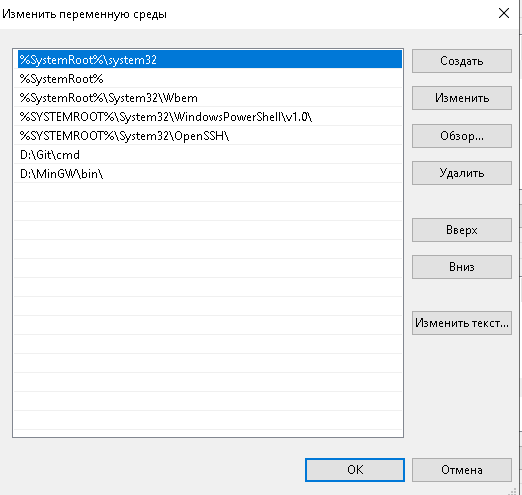


Рис.6 Переменные среды

Устанавливаем gnuplot скачиваем с официального сайта



Рис.7 gnuplot

Все готово для работы с программой. Так же можно запустить через файлы exe.

Python устанавливаем с официального сайта.

# Вывод

В ходе работы была написаны программы на языке С реализация процедуры интерполяции дискретным сплайном 3 степени. Получены практические навыки работы на языке Си.