**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Отчёт по лабораторной работе

**Построение кубического сплайна**

Выполнил:

студент ИИТММ гр. 381906-2

Челебаев А.А.

Проверил:

Эгамов А.И.

.

Нижний Новгород

2021 г.

Содержание

[Введение 3](#_Toc89110441)

[Постановка задачи 4](#_Toc89110442)

[Руководство пользователя 5](#_Toc89110443)

[Руководство программиста 7](#_Toc89110445)

[Описание структуры программы 7](#_Toc89110446)

[Описание структур данных 7](#_Toc89110447)

[Математическое описание алгоритмов 7](#_Toc89110448)

[Эксперименты 10](#_Toc89110453)

[Заключение 12](#_Toc89110454)

[Литература 13](#_Toc89110455)

[Код программы 14](#_Toc89110456)

# Введение

Сплайн — функция в математике, область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым алгебраическим многочленом (полиномом). Максимальная из степеней использованных полиномов называется степеньюсплайна. Разность между степенью сплайна и получившейся гладкостью называется дефектом сплайна. Например, непрерывная ломаная есть сплайн степени 1 и дефекта 1. В современном понимании сплайны — это решения многоточечных краевых задач сеточными методами.

Другими словами, сплайн — это кусочно заданная функция, то есть совокупность нескольких функций, каждая из которых задана на каком-то множестве значений аргумента, причём эти множества попарно непересекающиеся.

Сплайны имеют многочисленные применения как в математической теории, так и в прикладной математике (в частности, в разнообразных вычислительных программах). В частности, сплайны двух переменных интенсивно используются для задания поверхностей в различных системах компьютерного моделирования. Сплайны двух аргументов называют би-сплайнами (например, бикубический сплайн), которые являются двумерными сплайнами, моделирующими поверхности. Их часто путают с B-сплайнами (базисными сплайнами), которые являются одномерными и в линейной комбинации составляют кривые — каркас для «натягивания» поверхностей. Также из базисных сплайнов возможно составить трёхмерную конструкцию для моделирования объёмных тел.

# Постановка задачи

Реализовать программу, строящую график кубического интерполяционного сплайна.

Входные данные: координаты точек.

Выходные данные: координаты точек и график кубического сплайна.

# Руководство пользователя

Использование программы пользователем:

1. Скачать файл «CubicSpline.exe».
2. Запустить файл «CubicSpline.exe»
3. Пользователю будет предложено ввести желаемое количество точек.

# 

Рисунок 1. Скриншот запуска программы

1. Пользователю будет предложено ввести через клавишу Enter значения координаты X, затем, после ввода последней координаты X и нажатия Enter, значения координаты Y (их количество должно совпадать с количеством значений по x). После ввода значений по Y нужно нажать Enter.

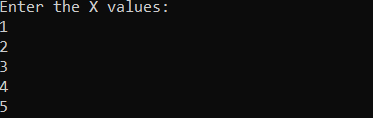


Рисунок 2. Скриншот интерфейса программы

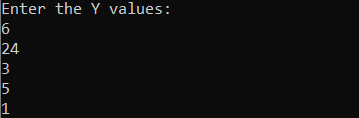


Рисунок 3. Скриншот интерфейса программы

1. В появившемся окне будет представлен график кубического сплайна.

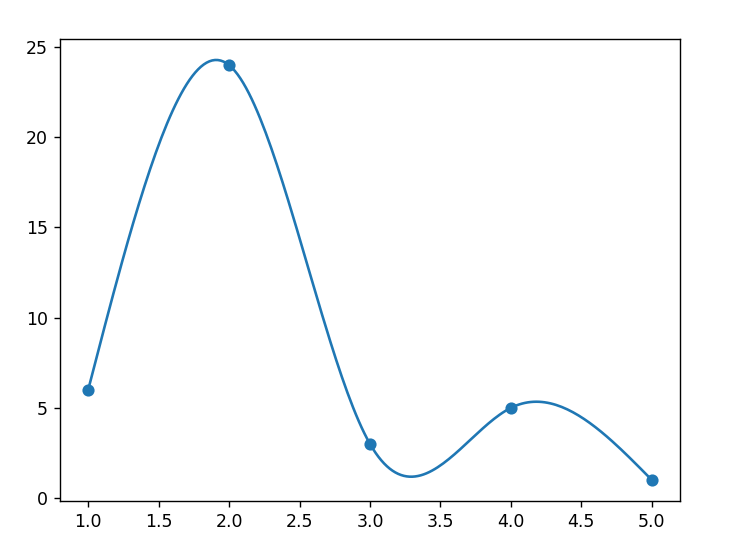


Рисунок 4. Скриншот результата выполнения программы

# Руководство программиста

## Описание структуры программы

Программа состоит из одного Python-файла. В программе используются ряд вспомогательных функций:

* BuildSpline(x, y, n) – функция, возвращающая функции сплайнов.
* Interpolate(splines, x) – функция, возвращающая значение интерполированной функции splines в точке x.

В программе используются следующие импортированные модули:

* NumPy (библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами);
* scipy.linalg (предоставляет множество функций для разрешения различных задач линейной алгебры);
* matplotlib.pyplot (библиотека на языке программирования Python для визуализации данных при помощи 2D-графики).

## Описание структур данных

Основными структурами данных, использованными в этой программе, являются списки - структуры данных, содержащие упорядоченный набор элементов.

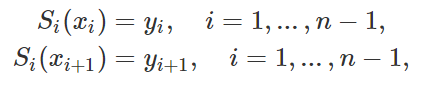
Ниже приведено описание функций, используемых мной в программе:

* plt.plot(new\_x, yi) – функция для рисования линий между точками на графике, где new\_x, yi – координаты следующей точки.
* plt.show() – функция, показывающая получившийся график.
* np.linspace(x[0], x[-1], 201) – функция, расставляющая равноудалённые точки на отрезке от x[0] до x[-1]. В данном случае на данном отрезке будет 201 точка.

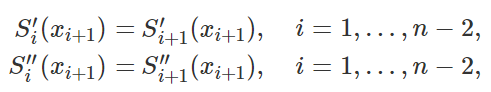
## Математическое описание алгоритмов

Предполагается, что точки *(xi, yi)* и *(xi+1, yi+1)* соединены кубическим полиномом *Si = aix3 + bix2 + cix + di* , где *xi ≤ x ≤ xi+1* при *i = 1, … , n – 1*. Чтобы найти интерполирующую функцию, нужно сначала определить коэффициенты *ai* , *bi* , *ci*,*di*  для каждой из кубических функций. Для *n* точек существует *n – 1* кубических функций, и для каждой кубической функции требуется 4 коэффициента, то есть в общей сложности у нас *4(n−1)* неизвестных, и поэтому нам нужно *4(n−1)* независимых уравнений для нахождения всех коэффициентов.

Во-первых, мы знаем, что кубические функции должны пересекать данные точки слева и справа:

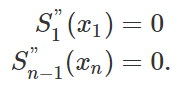


это дает нам *2(n−1)* уравнения. Далее, мы хотим, чтобы каждая кубическая функция как можно плавнее соединялась со своими «соседями», поэтому мы ограничиваем сплайны непрерывными первой и второй производными в точках при *i = 2 ,..., n−1*.



это дает нам *2(n−2)* уравнения.

Еще два уравнения требуются для вычисления коэффициентов *Si(x)*. Эти последние два ограничения являются произвольными, и их можно выбрать в соответствии с условиями выполняемой интерполяции. Общий набор конечных ограничений предполагает, что вторые производные равны нулю в конечных точках.



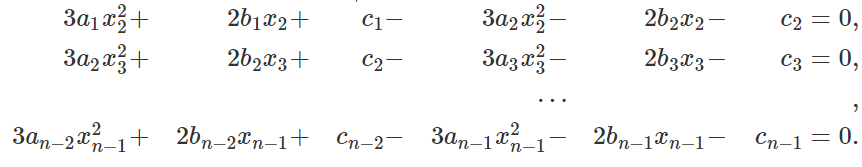
Это означает, что кривая представляет собой “прямую линию” в конечных точках.

# Чтобы определить коэффициенты каждой кубической функции, мы записываем ограничения явно в виде системы линейных уравнений с *4(n−1)* неизвестными. Для *n* точек неизвестными являются коэффициенты кубического сплайна *ai* , *bi* , *ci* , *di* , Si-соединения для точек *xi* и *xi+1*.

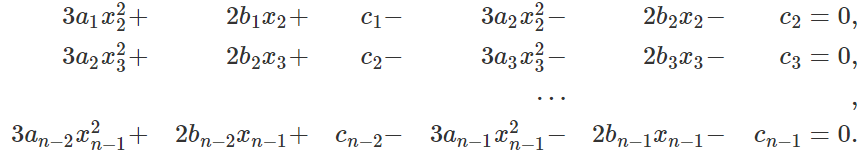
# Для ограничений *Si(xi) = yi* имеется:

# 

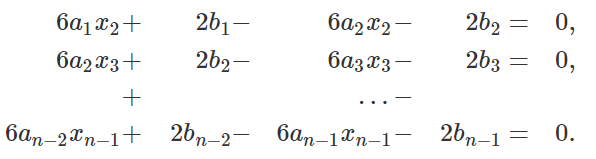
# Для ограничений *Si(xi+1) = yi+1* имеется:



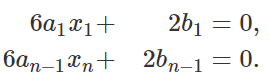
Для ограничений имеется:



 Для ограничений имеется:



 Для ограничений конечной точки и имеется:



Эти уравнения линейны относительно неизвестных коэффициентов *ai* , *bi* , *ci* и *di*.

Записываем их в матричном виде и решаем для коэффициентов каждого сплайна методом прогонки, при этом сама матрица должна быть квадратной и обратимой.

# Эксперименты

Проведём 3 эксперимента.

1. Возьмём следующие значения для координат x и y:

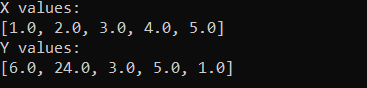


Рисунок 5. Скриншот пользовательского ввода для эксперимента №1

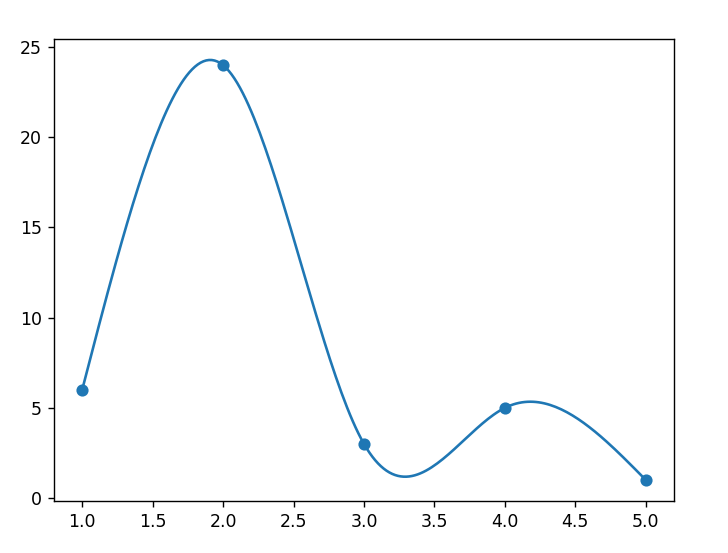


Рисунок 6. Скриншот графика, полученного в результате эксперимента №1

1. Возьмём следующие значения для координат x и y:

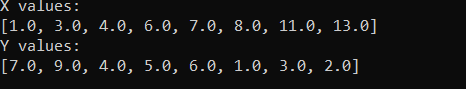


Рисунок 7. Скриншот пользовательского ввода для эксперимента №2

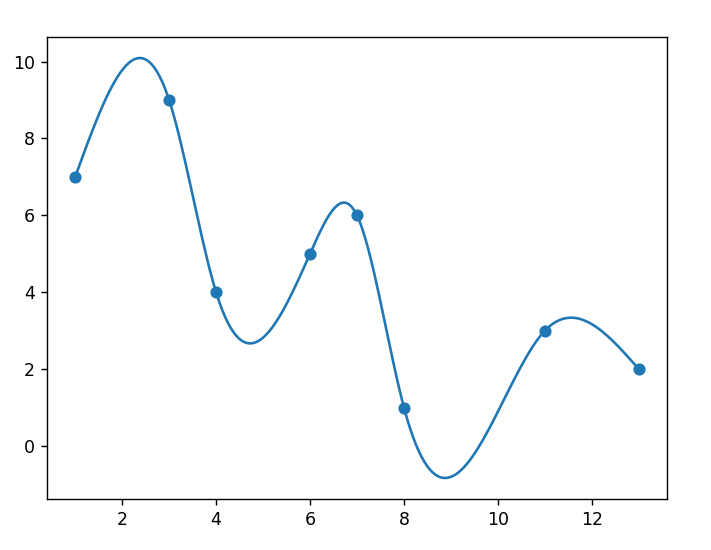


Рисунок 8. Скриншот графика, полученного в результате эксперимента №2

1. Возьмём следующие значения для координат x и y:

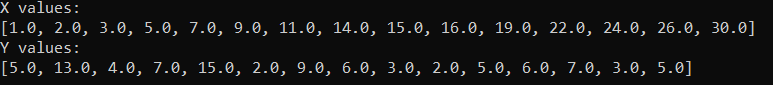


Рисунок 9. Скриншот пользовательского ввода для эксперимента №3

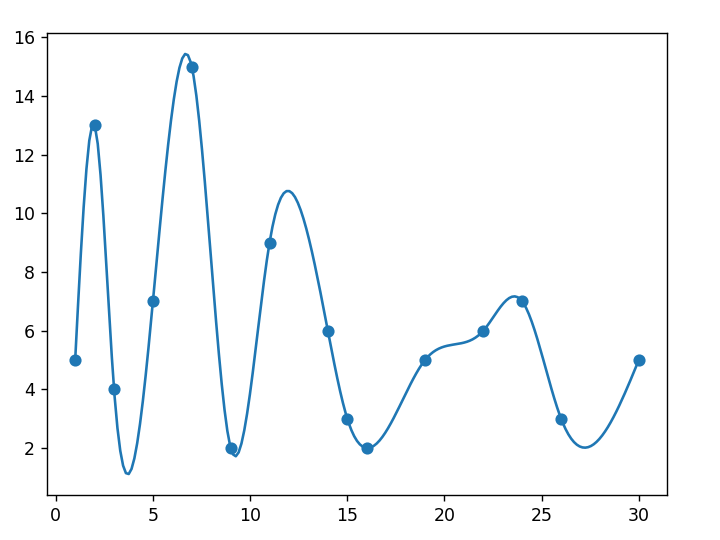


Рисунок 10. Скриншот графика, полученного в результате эксперимента №3

# Заключение

Таким образом, в данной лабораторной работе реализована кубическая-сплайн-интерполяция.

# Литература

1. Амосов А.А., Дубинский Ю.А., Копченова Н.В. «Вычислительные методы для инженеров» (учебное пособие) 1994г. – стр. 159(161) – 161(163) https://scask.ru/i\_book\_clm.php
2. Стронгина Н. Р. «Интерполяция кубическими сплайнами», 2021г - <http://www.lib.unn.ru/students/src/NumMeth%20%20Modul%2012.1.pdf>
3. Официальный сайт PythonWorld - <https://pythonworld.ru/novosti-mira-python/scientific-graphics-in-python.html>
4. Официальный сайт CoderLessons.com - <https://coderlessons.com/tutorials/python-technologies/uchitsia-stsipi/scipy-kratkoe-rukovodstvo>
5. Официальное руководство по NumPy – <https://numpy.org/doc/>
6. Официальное руководство по SciPy – <https://docs.scipy.org/doc/scipy/reference/>

**Приложение**

## Код программы

import matplotlib.pyplot as plt

import numpy as np

class SplineTuple: # Класс для сплайнов

    def \_\_init\_\_(self, a, b, c, d, x):

        self.a = a

        self.b = b

        self.c = c

        self.d = d

        self.x = x

# Функция построения сплайна

# x - точки, должны быть упорядочены по возрастанию

# y - значения функции в точках

# n - количество точек

def BuildSpline(x, y, n):

    # Создание массива сплайнов

    splines = [SplineTuple(0, 0, 0, 0, 0) for \_ in range(0, n)]

    for i in range(0, n):

        splines[i].x = x[i]

        splines[i].a = y[i]

    splines[0].c = splines[n - 1].c = 0.0

    # Решение СЛАУ относительно коэффициентов сплайнов c[i] методом прогонки для трехдиагональных матриц

    # Вычисление прогоночных коэффициентов прямым ходом метода прогонки

    alpha = [0.0 for \_ in range(0, n - 1)]

    beta  = [0.0 for \_ in range(0, n - 1)]

    for i in range(1, n - 1):

        hi  = x[i] - x[i - 1]

        hi1 = x[i + 1] - x[i]

        A = hi

        C = 2.0 \* (hi + hi1)

        B = hi1

        F = 6.0 \* ((y[i + 1] - y[i]) / hi1 - (y[i] - y[i - 1]) / hi)

        z = (A \* alpha[i - 1] + C)

        alpha[i] = -B / z

        beta[i] = (F - A \* beta[i - 1]) / z

    # Нахождение решения СЛАУ обратным ходом метода прогонки

    for i in range(n - 2, 0, -1):

        splines[i].c = alpha[i] \* splines[i + 1].c + beta[i]

    # По известным коэффициентам c[i] находим значения b[i] и d[i]

    for i in range(n - 1, 0, -1):

        hi = x[i] - x[i - 1]

        splines[i].d = (splines[i].c - splines[i - 1].c) / hi

        splines[i].b = hi \* (2.0 \* splines[i].c + splines[i - 1].c) / 6.0 + (y[i] - y[i - 1]) / hi

    return splines

# Вычисление значения интерполированной функции в произвольной точке

def Interpolate(splines, x):

    if not splines:

        return None # Если сплайны ещё не построены - возвращаем NaN

    n = len(splines)

    s = SplineTuple(0, 0, 0, 0, 0)

    if x <= splines[0].x: # Если x меньше точки сетки x[0] - используем первый элемент массива

        s = splines[0]

    elif x >= splines[n - 1].x: # Если x больше точки сетки x[n - 1] - используем последний элемент массива

        s = splines[n - 1]

    else: # Иначе x лежит между граничными точками сетки - производим бинарный поиск нужного эл-та массива

        i = 0

        j = n - 1

        while i + 1 < j:

            k = i + (j - i) // 2

            if x <= splines[k].x:

                j = k

            else:

                i = k

        s = splines[j]

    dx = x - s.x

    return s.a + (s.b + (s.c / 2.0 + s.d \* dx / 6.0) \* dx) \* dx # Вычисление значения сплайна в точке по схеме Горнера

print("Enter the desired number of points: ")

N = int(input())

print("Enter the X values: ")

x = [float(input()) for i in range (N)]

print("Enter the Y values: ")

y = [float(input()) for i in range (N)]

spline = BuildSpline(x, y, len(x))

new\_x = np.linspace(x[0], x[-1], 201)

yi = []    # Массив значений сплайна

for i in range (0, 201):

    yi.append(Interpolate(spline, new\_x[i]))

plt.scatter(x, y)

plt.plot(new\_x, yi)

print("X values: ", x, "Y values: ", y, sep = "\n")

plt.show()