**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

Кулемин П.А.

Проверил:

Эгамов А.И.

.

Нижний Новгород

2021 г.

Содержание

[Введение 3](#_Toc58264021)

[Постановка задачи 4](#_Toc58264022)

[Руководство пользователя 5](#_Toc58264023)

[Руководство программиста 7](#_Toc58264024)

[Описание структуры программы 7](#_Toc58264025)

[Описание структур данных 7](#_Toc58264026)

[Описание алгоритмов 8](#_Toc58264027)

[Эксперименты 9](#_Toc58264028)

[Заключение 14](#_Toc58264029)

[Литература 15](#_Toc58264030)

[Приложения 15](#_Toc58264031)

# Введение

Сплайны широко используются в компьютерной графике и при решении инженерных задач. На основе сплайнов строят базисы в функциональных пространствах, в том числе ортогональные базисы с хорошей сходимостью. В форме сплайна ищут приближенное (численное) решение обыкновенных дифференциальных уравнений и уравнений в частных производных.

Итак, в большинстве практических задач желательно соединить экспериментальные точки не ломаной линией, а гладкой кривой. Лучше всего для этих целей подходит интерполяция кубическими сплайнами. Дадим определение этим понятиям. Интерполяция - это метод нахождения неизвестных промежуточных значений некоторой функции по имеющемуся дискретному набору ее известных значений. Кубическим сплайном на сетке xi , i = 0 ,.., n отрезка [a, b] называют функцию S(x), дважды непрерывно-дифференцируемую на отрезке и представляющую собой полином степени не выше 3 на каждом его участке.

В данной лабораторной работе будет реализована кубическая сплайн-интерполяция.

# Постановка задачи

Реализовать программу, выполняющую кубическую сплайн-интерполяцию.

Входные данные: координаты точек.

Выходные данные: график кубического сплайна.

# Руководство пользователя

Использование программы пользователем:

1. Скачать файл «main.exe».
2. Запустить файл «main.exe»
3. Пользователю будет предложено ввести номер последней точки (XN).

# 

Рисунок 1. Скриншот запуска программы

1. Пользователю будет предложено ввести через клавишу enter значения координаты x (не более 12 значений и в порядке возрастания), затем, после нажатия Enter, значения координаты y (их количество должно совпадать с количеством значений по x). После ввода значений по y нужно нажать Enter.

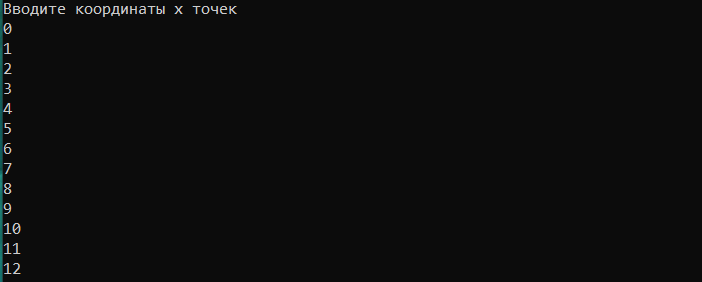


Рисунок 2. Скриншот интерфейса программы



Рисунок 3. Скриншот интерфейса программы

1. В появившемся окне будет представлен график кубического сплайна.

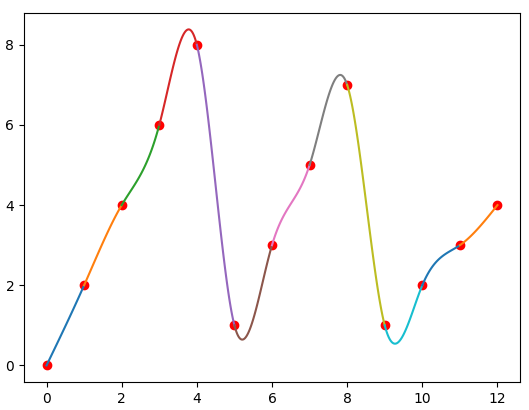


Рисунок 4. Скриншот результата выполнения программы

# Руководство программиста

## Описание структуры программы

Программа состоит из одного Python-файла. В программе определена единственная главная функция main(). В программе используются ряд вспомогательных функций:

* calc\_x – функция, возвращающая соответствующий коэффициент матрицы для подсчета ci
* calc\_y – функция, возвращающая результат ньютоновской разницы от переданных аргументов
* calc\_coefs – функция, возвращающая массив коэффициентов соответствующего Si полинома
* func – функция, подсчитывающая значение i-го полинома от переданного x
* i\_polynom\_y – функция, возвращающая массив значений Si на промежутке [xi-1, xi]

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 двумерной (2D) графикой).

## Описание структур данных

Основными структурами данных, которые я использовал в своей программе, являются списки - структуры данных, содержащие упорядоченный набор элементов.

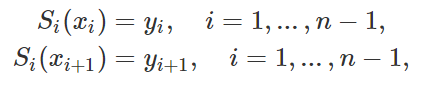
Ниже приведено описание функций, используемых мной в программе:

* solve\_banded(l\_and\_u, ab, b) -функция разрешающая матричное уравнение, где ab- определенным образом записанная матрица ленточного типа, b- результирующая матрица, l\_and\_u – кортеж из числа нулей над и под главной диагональю
* np.diagonal(X,b) – функция возвращающая диагональ матрицы X смещенную относительно ее главной диагонали на b
* np.concatenate(a,b) – функция возвращающая результат конкатенации a и b
* plt.plot(x\_new, y\_new, 'b') – функция для рисования линий между точками на графике, где x\_new, y\_new – координаты следующей точки, 'b' – цвет линии (синий).
* plt.plot(x, y, 'ro') – функция для отображения точек на графике, где x, y – координаты текущей точки, 'ro' – цвет точек (красный).
* plt.show() – функция, показывающая получившийся график.

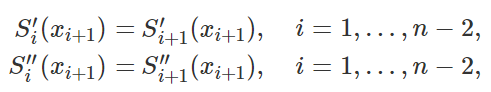
## Математическое описание алгоритмов

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:

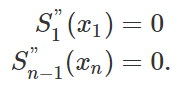


это дает нам *2(n−1)* уравнения. Далее, мы хотим, чтобы каждая кубическая функция как можно плавнее соединялась со своими «соседями», поэтому мы ограничиваем сплайны непрерывными первой и второй производными в точках при *i = 2 ,..., n−1*.



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.



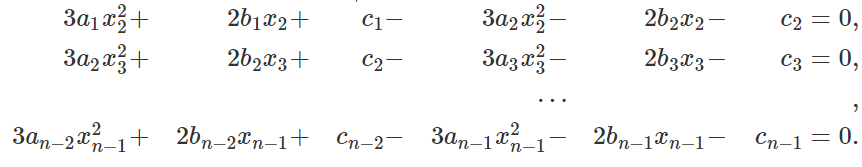
Это означает, что кривая представляет собой “прямую линию” в конечных точках.

# 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*.

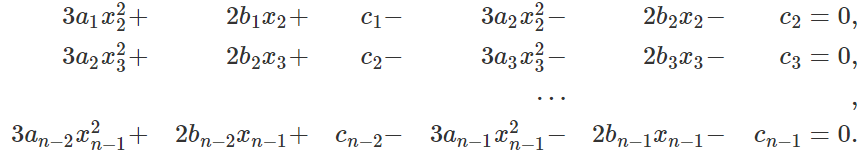
# Для ограничений *Si(xi) = yi* имеется:

# 

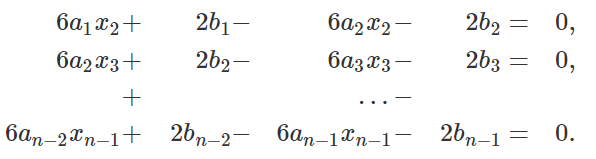
# Для ограничений *Si(xi+1) = yi+1* имеется:



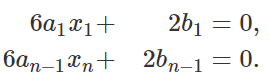
Для ограничений имеется:



 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 делением слева, при этом сама матрица должна быть квадратной и обратимой.

# Эксперименты

Проведём 3 эксперимента.

1. Возьмём следующие значения для координат x и y:

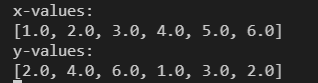


Рисунок 5. Скриншот пользовательского ввода для эксперимента №1

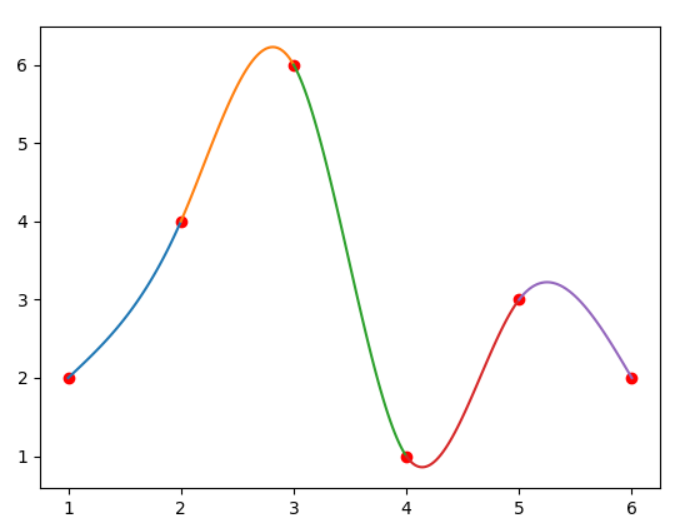


Рисунок 6. Скриншот графика, полученного в результате эксперимента №1

1. Возьмём следующие значения для координат x и y:

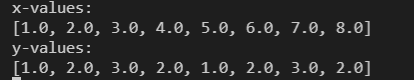


Рисунок 7. Скриншот пользовательского ввода для эксперимента №2

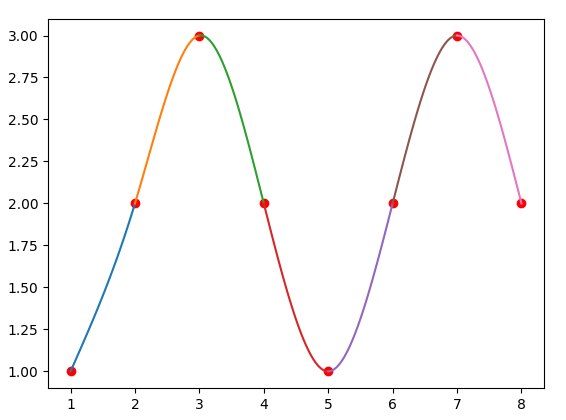


Рисунок 8. Скриншот графика, полученного в результате эксперимента №2

1. Возьмём следующие значения для координат x и y:

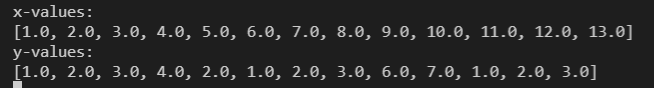


Рисунок 9. Скриншот пользовательского ввода для эксперимента №3

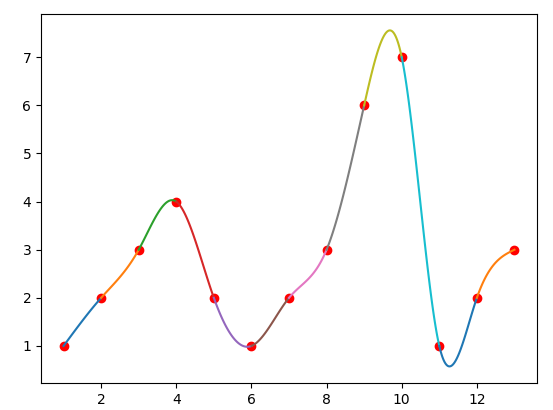


Рисунок 10. Скриншот графика, полученного в результате эксперимента №3

# 

# Заключение

Таким образом, в данной лабораторной работе была успешно 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>

**Приложение**

## Код программы

import matplotlib.pyplot as plt

import numpy as np

from scipy.linalg import solve\_banded

def calc\_x(i,j,h):

    if(i==j):

        return 2

    if(i==1+j):

        return h[j+1]/(h[j]+h[j+1])

    if(j==i+1):

        return h[j]/(h[j]+h[j+1])

    return 0

def calc\_y(i,x,y):

    return(((y[i+1]-y[i])/(x[i+1]-x[i]))-((y[i]-y[i-1])/(x[i]-x[i-1])))/(x[i+1]-x[i-1])

def calc\_coefs(y,c,h, i):

    return [y[i], (c[i]\*h[i-1]/3)+(c[i-1]\*h[i-1]/6)+((y[i]-y[i-1])/h[i-1]),c[i]/2,((c[i]-c[i-1])/(h[i-1]))/6]

def func(x,coefs,xi):

    return(coefs[0]+(coefs[1]\*(x-xi))+(coefs[2]\*((x-xi)\*\*2))+(coefs[3]\*((x-xi)\*\*3)))

def i\_polynom\_y(i,x,c\_solve,y,h):

    xlist=np.linspace(x[i-1],x[i])

    cfs=calc\_coefs(y,c\_solve,h,i)

    return [func(xt,cfs,x[i])for xt in xlist]

def main():

    print("Введите номер последней точки")

    N = int(input())

    print("Вводите координаты x точек")

    x=[float(input()) for i in range(N+1)]

    print("Вводите координаты y точек")

    y=[float(input()) for i in range(N+1)]

    h=[float(x[i]-x[i-1]) for i in range(1,N+1)]

    X=[[calc\_x(i,j,h) for i in range(N-1)] for j in range(N-1)]

    y\_fin=[6\*calc\_y(i,x,y) for i in range(1,N)]

    c\_solve=np.concatenate(([0],solve\_banded((1,1),[np.concatenate(([0],np.diagonal(X,1))),np.diagonal(X),np.concatenate((np.diagonal(X,-1),[0]))], y\_fin),[0]))

    for i in range(0,N+1):

        plt.plot(x[i],y[i],'ro')

    for i in range(1,N+1):

        plt.plot(np.linspace(x[i-1],x[i]),i\_polynom\_y(i,x,c\_solve,y,h))

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

    plt.show()

main()