**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**

Выполнила:

студентка 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 10](#_Toc58264028)

[Заключение 14](#_Toc58264029)

[Литература 15](#_Toc58264030)

[Приложения 15](#_Toc58264031)

# Введение

Введем понятие «интерполяция». По определению это построение функции аппроксимирующей зависимости *y(x)* в промежуточных точках (между *xi*). Поэтому иногда ее еще по-другому называют аппроксимацией. При большом количестве узлов интерполяции сильно возрастает степень интерполяционных многочленов, что делает их неудобными для вычислений. Этого можно избежать, разбив отрезок интерполяции на несколько частей, с последующим построением на каждой части самостоятельного интерполяционного многочлена. Однако такой вариант наталкивается на существенный недостаток: в точках стыка разных интерполяционных многочленов будет разрывной их первая производная.

В этом случае удобно пользоваться особым видом кусочно-полиномиальной интерполяции – интерполяцией кубическими сплайнами. Кубический сплайн — гладкая функция, область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым кубическим многочленом (полиномом).

В данной лабораторной работе будет реализована кубическая сплайн-интерполяция.

# Постановка задачи

Реализовать программу, выполняющую построение кубического сплайна.

На вход должны поступать координаты точек (не более 12).

На выходе должен быть построен 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.

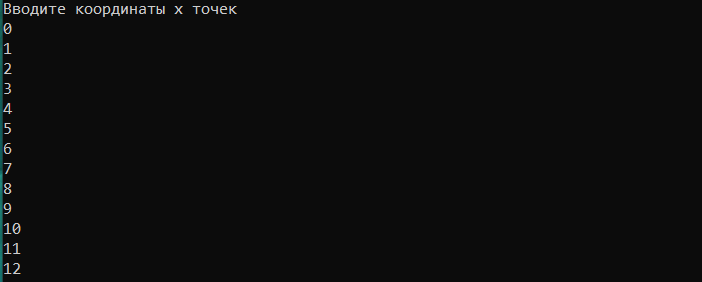


Рисунок 2. Скриншот интерфейса программы



Рисунок 3. Скриншот интерфейса программы

1. В появившемся окне будет представлен график кубического сплайна.

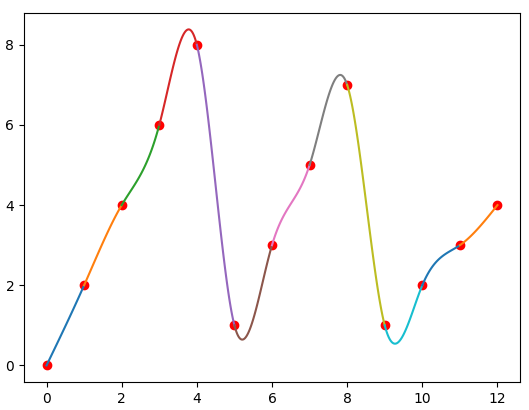


Рисунок 4. Скриншот, иллюстрирующий возможный результат выполнения программы

# Руководство программиста

## Описание структуры программы

Программа состоит из одного файла, написанного на языке программирования Python. В программе определена единственная главная функция main(). В программе используются ряд вспомогательных функций:

* calc\_x функция возвращающая соотвтствующий коэффициент матрицы для посчтета ci
* calc\_y функция возвращающая результат ньютоновской разницы от переданных аргументов
* calc\_coefs функция возвращающаф 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

Пользовательский ввод преобразуется в список - структуру данных, которая содержит упорядоченный набор элементов, т.е. хранит последовательность элементов.

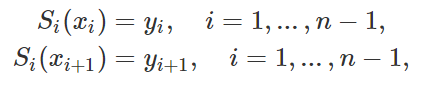
В самом коде программы как таковые структуры данных (кроме названной выше) напрямую не прописаны, поскольку я пользовалась функциями из импортированных библиотек. Однако в таком случае стоит пояснить, что именно делает каждый из использованных мной функций.

* np.linspace(min(x), max(x), 100) – данная функция создает последовательность данных, равномерно расположенных на числовой прямой в заданном интервале. Аргументами данной функции являются min(x) – начало последовательности, max(x) – конец последовательности, 100 – количество данных в выборке.
* 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.

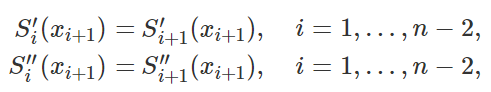
## Описание алгоритмов (математическое)

При интерполяции кубическим сплайном (см. рисунок 4) интерполирующая функция представляет собой набор кусочно-кубических функций. В частности, 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. Следовательно, у нас есть в общей сложности *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:

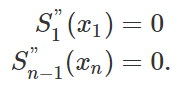


что дает нам *2(n−1)* уравнения. Далее, мы хотим, чтобы каждая кубическая функция как можно более плавно соединялась со своими «соседями», поэтому мы ограничиваем сплайны непрерывными первой и второй производными в точках при *i = 2 ,..., n−1*.



что 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.

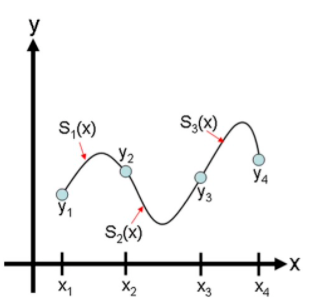


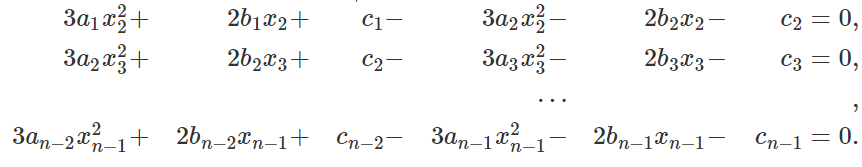
Рисунок 4. Интерполяция кубическим сплайном

# 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*.

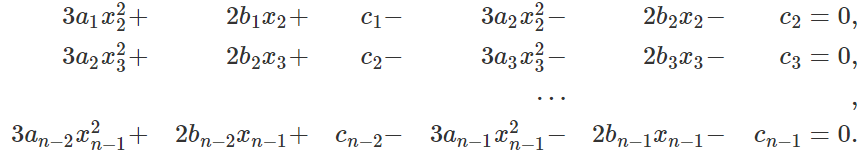
# Для ограничений *Si(xi) = yi* имеется:

# 

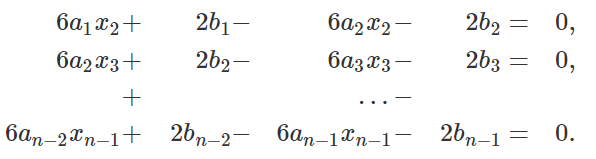
# Для ограничений *Si(xi+1) = yi+1* имеется:



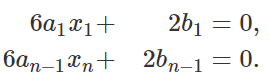
Для ограничений имеется:



 Для ограничений имеется:



 Для ограничений конечной точки и имеется:



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

Мы можем поместить их в матричную форму и решить для коэффициентов каждого сплайна делением слева. Для решения матричного уравнения *Ax = b* необходимо, чтобы матрица *A* была квадратной и обратимой. В случае нахождения уравнений кубического сплайна, матрица *A* всегда квадратна и обратима до тех пор, пока *xi* значения в наборе данных уникальны.

# Эксперименты

Для проверки работоспособности программы проведём 3 эксперимента.

1. Возьмём следующие значения для координат x и y:

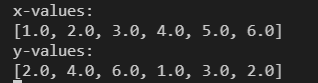


Рисунок 5. Скриншот пользовательского ввода для эксперимента №1

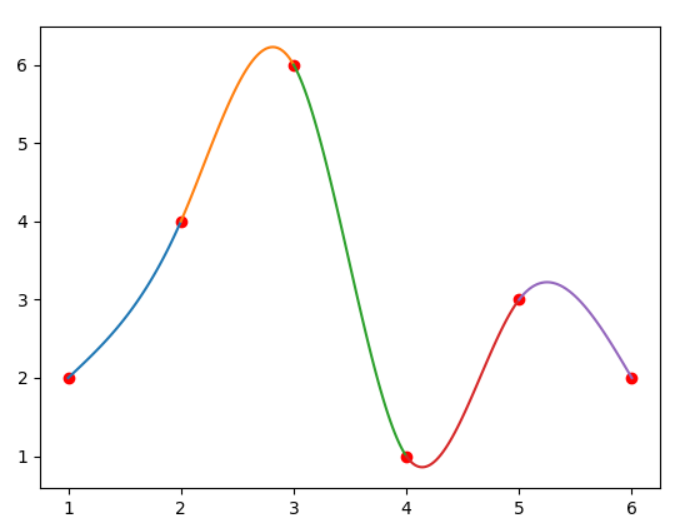


Рисунок 6. Скриншот графика, полученного в результате эксперимента №1

1. Возьмём следующие значения для координат x и y:

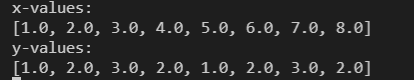


Рисунок 7. Скриншот пользовательского ввода для эксперимента №2

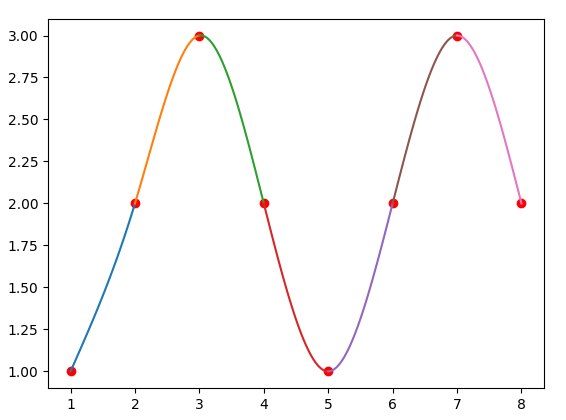


Рисунок 8. Скриншот графика, полученного в результате эксперимента №2

1. Возьмём следующие значения для координат x и y:

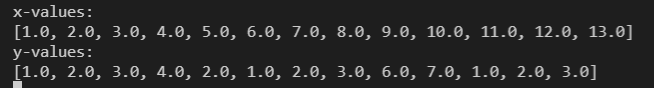


Рисунок 9. Скриншот пользовательского ввода для эксперимента №3

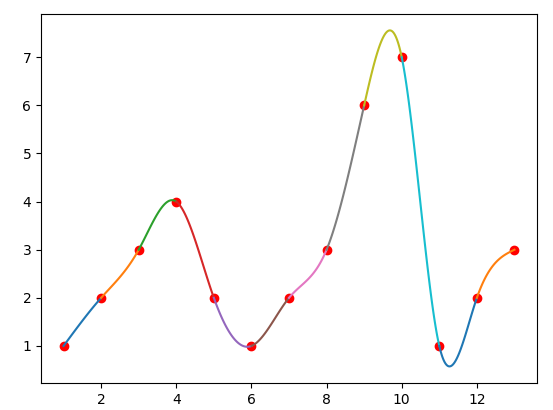


Рисунок 10. Скриншот графика, полученного в результате эксперимента №3

# 

# Заключение

Таким образом, в рамках данной лабораторной работы была успешно создана программа, которая позволяет построить кубический сплайн по заданным координатам. Программа выдержала проверку и готова к использованию.

# Литература

1. Официальный сайт PythonRu. – Режим доступа <https://pythonru.com/biblioteki/pyplot-uroki>
2. Материалы лекции - D:\\_Julia\Университет\3 курс\Вычислительные методы\ ВМ Лекция 4.docx
3. Самарский А.А., Гулин А.В. «Численные методы», 1989 г. - <http://samarskii.ru/books/book1989.pdf>
4. Qingkai Kong, Timmy Siauw, Alexandre Bayen “Python Programming and Numerical Methods” –

<https://pythonnumericalmethods.berkeley.edu/notebooks/Index.html>

**Приложения**

## 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()