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»

**ОТЧЕТ**

По курсу «Вычислительные методы»

На тему:

**Интерполяция кубическими сплайнами.**

**Выполнил**:студент группы 381806-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Напылов Е.И.

Подпись

**Научный руководитель**:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Эгамов А.И.

Подпись

Нижний Новгород

2020

# Содержание

[Содержание 2](#_Toc57313371)

[Введение 3](#_Toc57313372)

[Постановка задачи 4](#_Toc57313373)

[Теоретическая часть 5](#_Toc57313374)

[Реализация 7](#_Toc57313375)

[Представление сплайна 7](#_Toc57313376)

[Алгоритм вычисления коэффициентов сплайна 7](#_Toc57313377)

[Алгоритм вычисления значения функции 8](#_Toc57313378)

[Ввод и вывод данных 9](#_Toc57313379)

[Руководство к программе 10](#_Toc57313380)

[Настройки 10](#_Toc57313381)

[Ввод данных с клавиатуры 10](#_Toc57313382)

[Использование случайных данных 11](#_Toc57313383)

[Встроенные примеры 11](#_Toc57313384)

[Просмотр результата. 11](#_Toc57313385)

[Тестирование 12](#_Toc57313386)

[Тест синуса 12](#_Toc57313387)

[Тест параболы х2 12](#_Toc57313388)

[Тест случайных данных 13](#_Toc57313389)

[Тест данных, полученных вводом с клавиатуры 13](#_Toc57313390)

[Заключение 14](#_Toc57313391)

[Список литературы 15](#_Toc57313392)

[Приложение 16](#_Toc57313393)

[main.py 16](#_Toc57313394)

[interpolationLib.py 18](#_Toc57313395)

# Введение

Одной из главных задач вычислительных методов является задача интерполяции функций. Интерполяция – вычисление значений функции в любой точке внутри области исходных данных по известному набору исходных значений этой функции. Кроме интерполяции существует термин экстраполяции. Экстраполяция – вычисление значений функции в тех точках, которые выходят за границы исходных данных.

Чаще всего интерполяция применяется в науке для установления зависимости экспериментальных данных.

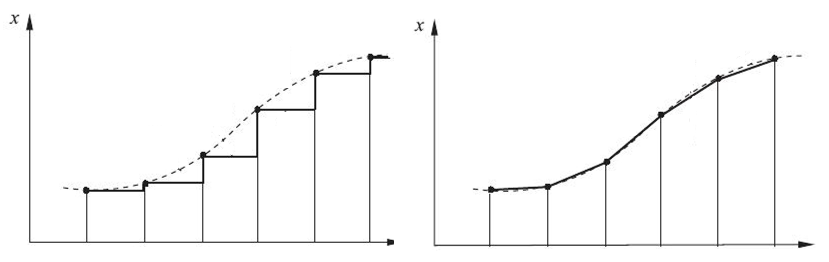
Кроме науки интерполяции активно используется в компьютерной графике и обработке изображений. Примером может быть расширение изображения.

Интерполяция может применятся для сглаживания процесса онлайн игры на клиенте. В силу ограничения пропускной способности сети, сервер не может постоянно отправлять клиенту данные, к примеру, о передвижении другого игрока. Он переделает лишь некоторые координаты. Если напрямую использовать полученные от сервера данные, то процесс игры на клиенте будет рваным и некомфортным. Для решения этой проблемы применяется интерполяция по данным сервера. Таким образом, на клиенте вычисляются промежуточные координаты игрока и его передвижение выглядит плавным.

Интерполяция может применятся для восстановления некоторых утерянных данных.

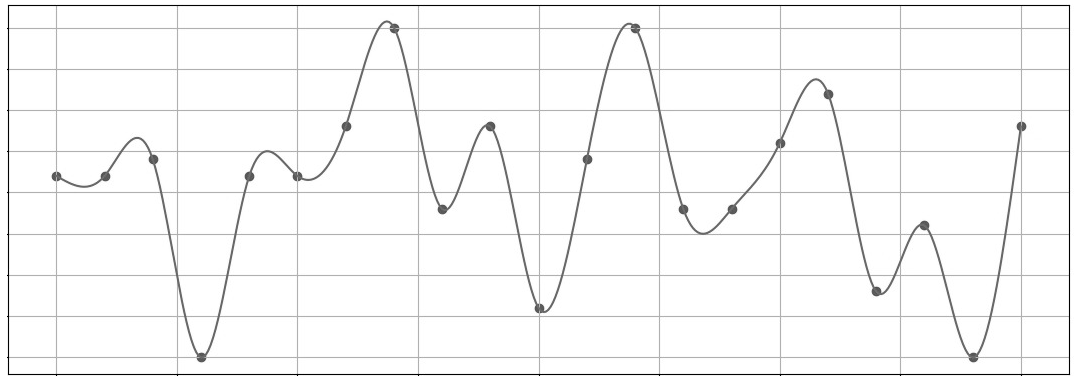
# Постановка задачи

Существует большое количество способов интерполяции. Простейшими примерами могут служить ступенчатая и линейная интерполяции.



Однако, эти способы имеют достаточно низкую точность.

Интерполяция кубическими сплайнами позволяет получить более точные значения функции в неизвестных точках, поэтому в данной работе рассмотрен этот тип.



Требуется разработать программу, которая будет производить интерполяцию кубическим сплайном с заданным шагом.

Программа должна обладать следующими функциями:

1. Ввод данных с клавиатуры
2. Генерация случайных данных
3. Построение графика с заданными границами и шагом вычисленных точек
4. Вывод промежуточных вычислений и результатов консоль
5. Возможность изменять параметры вывода и отображения на графике

# Теоретическая часть

Смысл данного метода заключается в определении коэффициентов полиномов третьей степени – составляющих сплайна.

Пусть у нас есть исходные данные, заданные таблицей точек.

Во-первых, чтобы сплайн проходил через заданные точки, должно выполнятся условие:



Кроме этого, должны выполнятся условия сопряжения в заданных точках: 

В граничных точках сплайна потребуем:



Каждый многочлен может быть представлен в виде:



Применим первое условие:



Потребуем непрерывность производных в заданных точках:





Выразим di и подставим в предыдущее выражение, исключив ai:





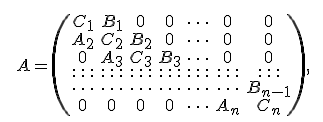
Теперь подставим полученные выражения обратно и преобразуем результат: (\*)



Этих условий недостаточно для определения коэффициентов сплайна, поэтому добавим еще 2 условия для границ:



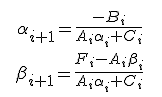
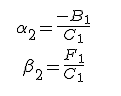
Получим систему уравнений Ax=F, которую можно решить методом прогонки.



В этой системе F – вектор равный правой части уравнений (\*),



Решим полученную систему методом прогонки.

Таким образом получили коэффициенты сплайна. Для вычисления значения в точке х0 необходимо выбрать соответствующий ей многочлен и подставить в него х0

# Реализация

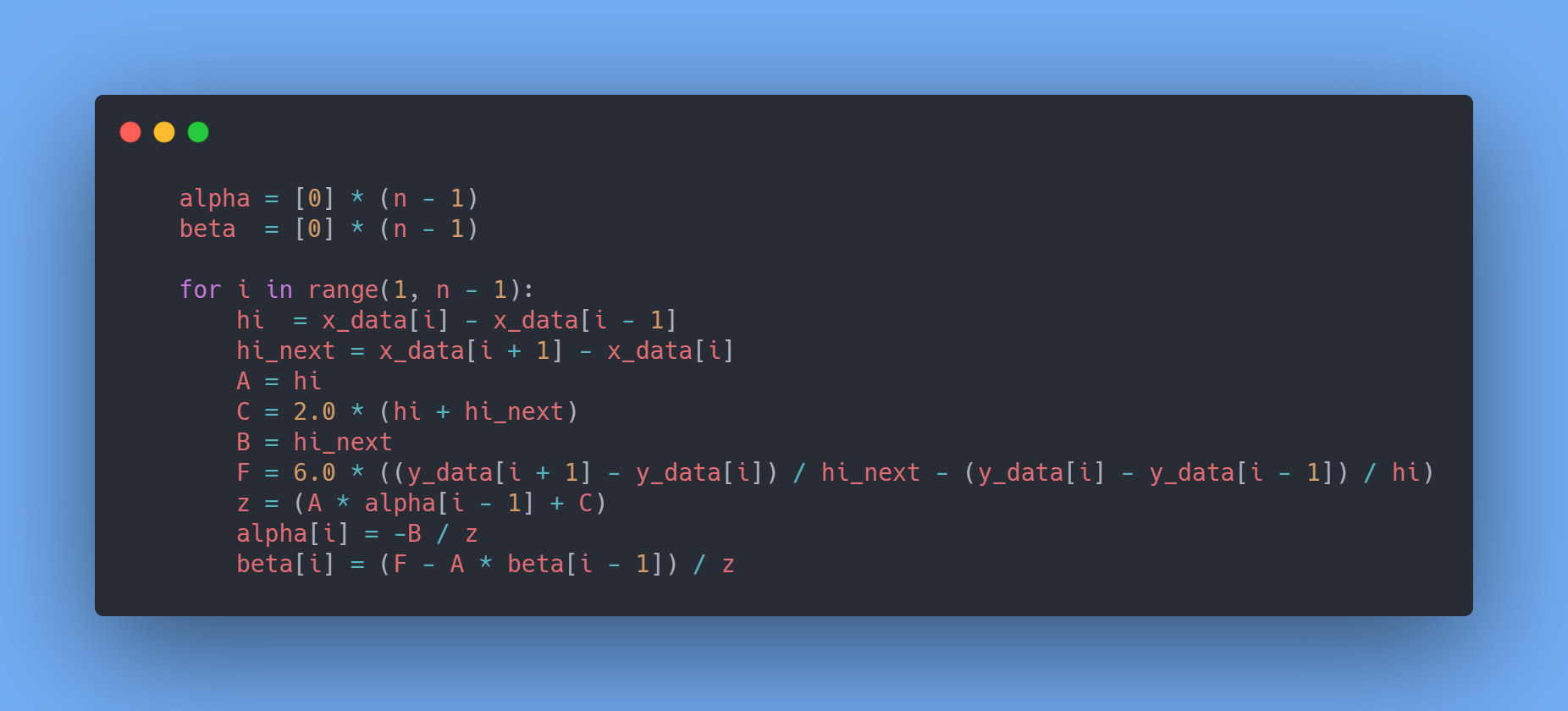
Для реализации алгоритма кубической сплайн интерполяции был выбран язык программирования Python. Это обусловлено его простотой и наличием удобной библиотеки для построения графиков – matplotlib.

## **Представление сплайна**

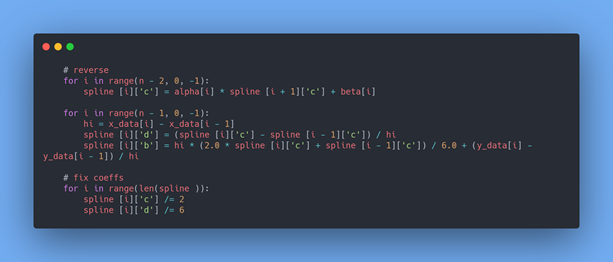
Участок сплайна хранится в виде словаря (ассоциативного массива), ключами которого являются коэффициенты многочлена третьей степени и граничное значение х, по которому из множества многочленов будет выбираться конкретный.

## Алгоритм вычисления коэффициентов сплайна

На первом этапе «составляется система уравнений», которая является трехдиагональной матрицей. Затем вычисляются прогоночные коэффициенты. Этот этап называется прямым ходом решения системы уравнений методом прогонки.



На втором этапе вычисляются неизвестные с помощью обратного хода метода прогонки.



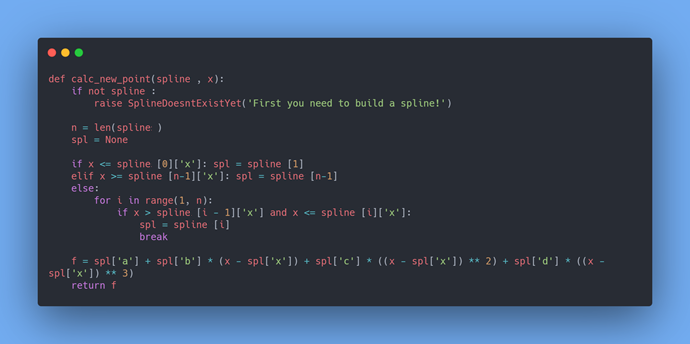
## Алгоритм вычисления значения функции

Для вычисления значения функции в точке необходимо выбрать соответствующий ей многочлен. Во-первых, если коэффициенты сплайна по какой-то причине еще не существуют, будет вызвано исключение, которое скажет об этой ошибке.

Если точка выходит за границы исходных данных, значение функции вычисляется по граничным многочленам.

Если точка попадает в интервал исходных данных, то в списке сплайна ищется конкретный многочлен, соответствующий этой точке.

И наконец, после выбора полинома происходит вычисление значения функции.



## Ввод и вывод данных

**Для ввода данных используются следующие функции:**

1. **Read\_data\_from\_keyboard()** – ввод данных с клавиатуры. При этом контролируется порядок значений Х, размеры списков Х и У, корректность параметров построения. В случае ошибки ввода нужно заново ввести данные.
2. **Random\_data()** – генерация случайных данных. Результатом является набор данных, расположенных в квадрате [-10, 10][-10, 10].

**Для отображения результатов и графика используются следующие функции:**

1. **Plot\_data(x\_data, y\_data)** – строит исходные данные (точки).
2. **Plot\_new\_points(x\_points, y\_points, flag)** – строит вычисленные точки и соединяет их отрезками (линейная интерполяция). Если flag == true точки будут дополнительно выделены цветом.
3. **Print\_data(x\_data, y\_data)** – вывод исходных данных.
4. **Print\_spline(spline)** – вывод коэффициентов сплайна в виде словарей.
5. **Print\_calc\_points(calc\_x, calc\_y)** – вывод вычисленных значений.

Построение графика реализовано на основе библиотеки matplotlib.

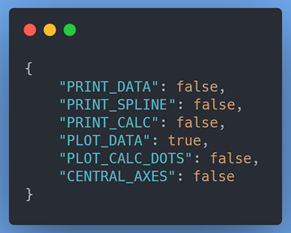
**Для работы программы также используются служебные функции:**

1. **Load\_settings()** – парсер настроек из файла «settings.json»
2. **Menu ()** – вывод списка пунктов меню, ввод выбранного пункта, проверка ввода на корректность.
3. **Main()** – управление всей программой.

# Руководство к программе

## Настройки

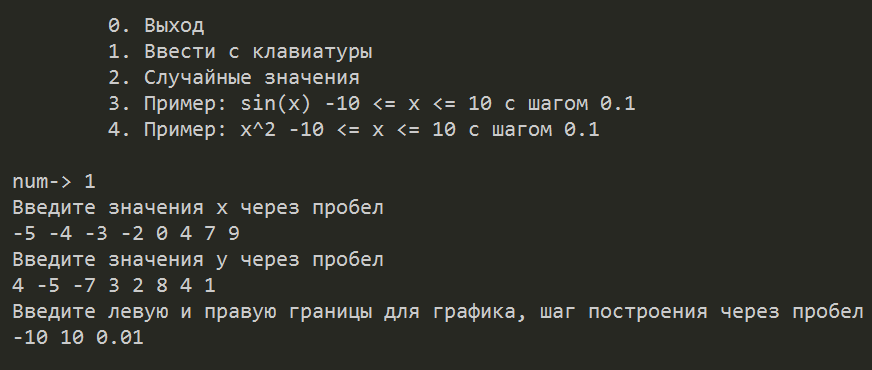
В директории программы находится файл «settings.json», с помощью которого можно настроить то, что будет выводить программа. Флаг принимает значение true или false.



1. “PRINT\_DATA” – вывод исходных данных в консоль
2. “PRINT\_SPLINE” – вывод коэффициентов полученных полиномов в консоль
3. “PRINT\_CALC” - вывод вычисленных точек в консоль
4. “PLOT\_DATA” – отображение исходных точек на графике
5. “PLOT\_CALC\_DOTS” - выделение вычисленных точек на графике
6. “CENTRAL\_AXES” – отображение осей координат в центре

## Ввод данных с клавиатуры

Для использования этой функции нужно ввести 1 и нажать Enter. Сначала нужно ввести список значений Х, отсортированных по возрастанию. После этого вводятся соответствующие им значения Y. Затем нужно ввести границы и шаг построения. Иногда границы, выходящие за данные, могут привести к проблемам с масштабом графика!



## Использование случайных данных

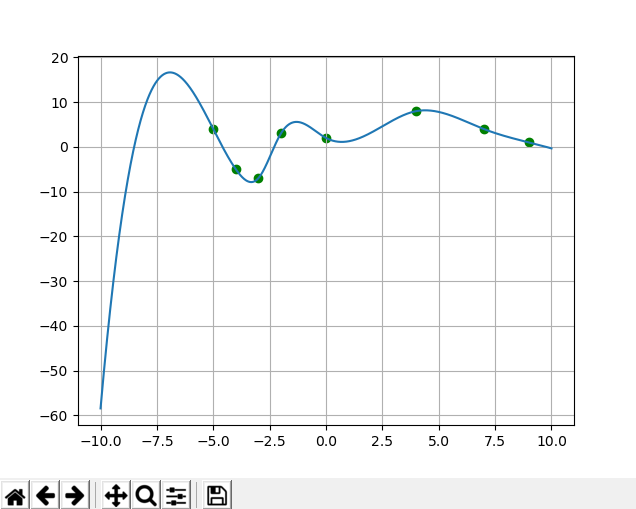
Для использования этой функции нужно ввести 2 в меню. Будет сгенерирован набор данных, расположенных в квадрате [-10; 10], [-10; 10].

## Встроенные примеры

Для удобства тестирования можно использовать встроенные примеры известных функций.

## Просмотр результата.

После ввода (выбора) данных появится окно просмотра графика. В этом окне можно настраивать размер, растяжение, масштаб, положение графика.

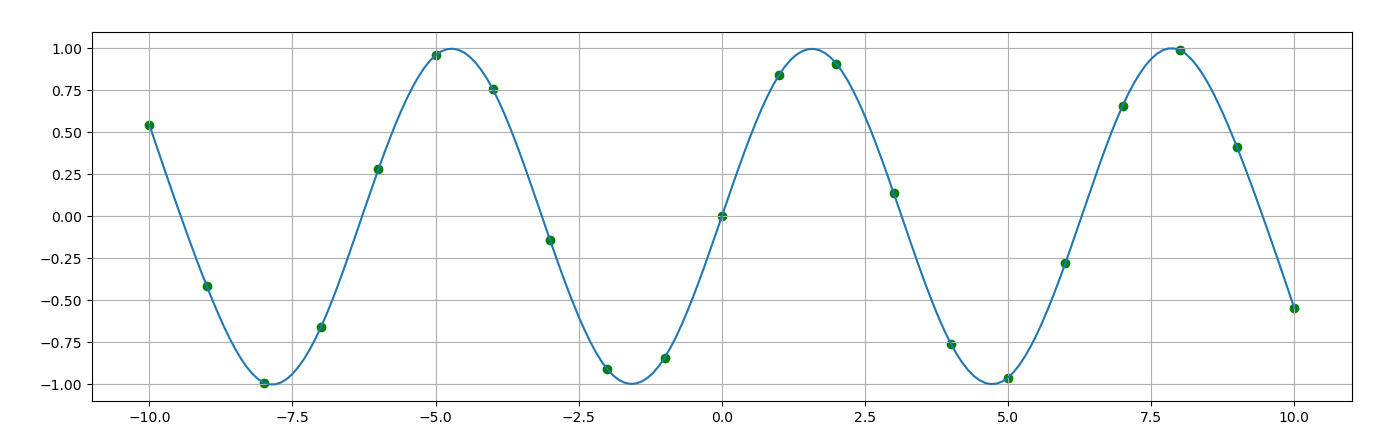


Зеленым цветом выделены исходные данные. С настройками, показанными выше, вычисленные значения не выделяются. Они соединены отрезками (линейная интерполяция). Если включить отображение вычисленных точек, то они будут выделены красным цветом.

# Тестирование

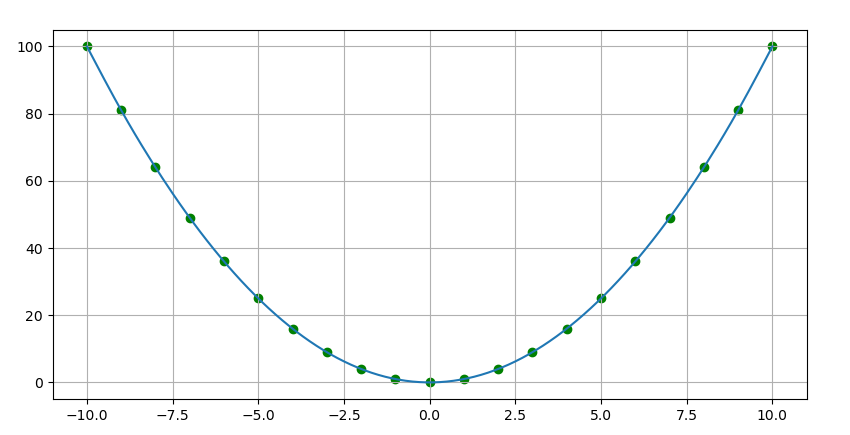
## Тест синуса

***Данные:*** значение синуса в точках [-10, -9, …, 9, 10] ***Параметры построения:*** [-10, 10], с шагом 0.1

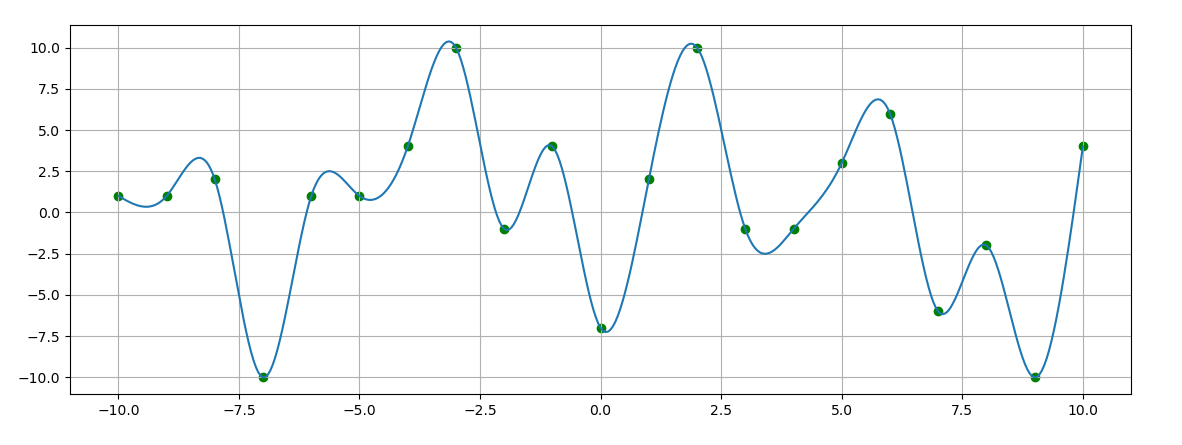


## Тест параболы х2

***Данные:*** значение параболы в точках [-10, -9, …, 9, 10] ***Параметры построения:*** [-10, 10], с шагом 0.1



## Тест случайных данных



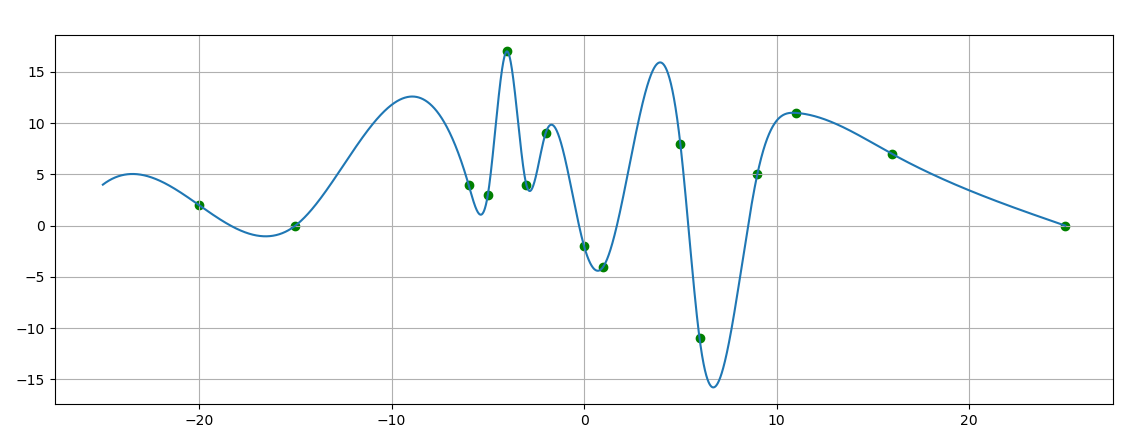
## Тест данных, полученных вводом с клавиатуры

***Данные:***

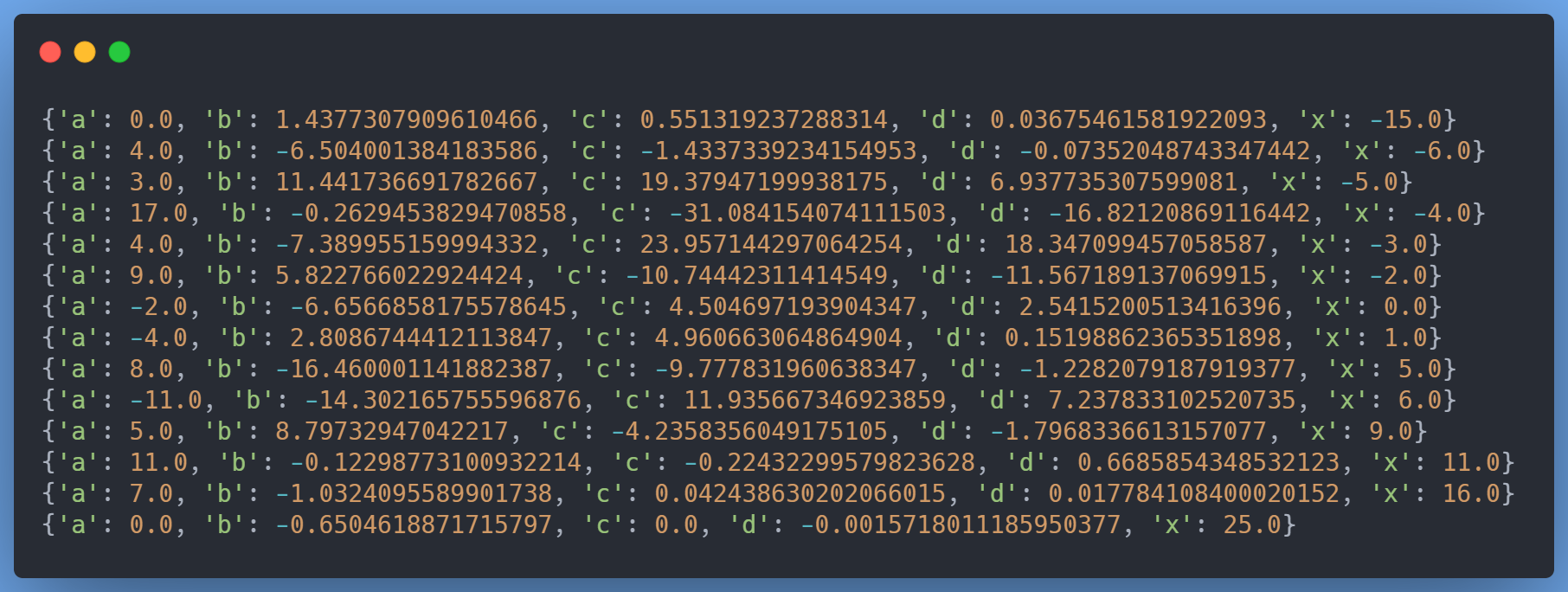
X = [-20, -15, -6, -5, -4, -3, -2, 0, 1, 5, 6, 9, 11, 16, 25],

Y = [2, 0, 4, 3, 17, 4, 9, -2, -4, 8, -11, 5, 11, 7, 0]

***Параметры построения:*** [-25, 25], с шагом 0.01



***Сплайн:***



# Заключение

В ходе работы я узнал, где может применятся интерполяция. После этого я рассмотрел основные способы интерполяции. Подробно рассмотрел способ кубической сплайн интерполяции, узнал его достоинства и недостатки. После изучения теоретической части я разработал программу, которая производит интерполяцию по исходным данным и строит недостающие точки с заданным шагом. Таким образом, я получил представление об одной из главных задач вычислительной математики и получил опыт в разработке программ на языке Python с использованием библиотеки matplotlib.

# Список **литературы**

1. Волков Е.А., «Численные методы», 2008
2. Жидков Е.Н., «Вычислительная математика», 2013
3. Лутц М., «Изучаем Python», том 1, 2019

# Приложение

## main.py

from interpolationLib import \*

from random import randint, shuffle

import json

def load\_settings():

    with open('settings.json', 'r', encoding='utf-8') as fh:

        data\_r = json.load(fh)

    fh.close()

    return data\_r

def read\_data\_from\_keyboard():

    flag = True

    x = list()

    y = list()

    params = [0.0, 0.0, 0.0]

    while (flag):

        flag = False

        x = list(map(float,input('Введите значения x через пробел\n').split()))

        if sorted(x) != x:

            print('ОШИБКА! Значения должны быть отсортированы по возрастанию!')

            flag = True

            continue

        y = list(map(float,input('Введите значения y через пробел\n').split()))

        if len(x) != len(y):

            print('ОШИБКА! Размеры списков x и y должны совпадать!')

            flag = True

            continue

        params = list(map(float,input('Введите левую и правую границы для графика, шаг построения(рекомм. <= 0.1) через пробел \n').split()))

        if params[0] > params[1] or params[2] <= 0:

            print('ОШИБКА! неверные параметры построения')

            flag = True

            continue

    return x, y, params[0], params[1], params[2]

def random\_data():

    x\_data = list(range(-10, 10+1))

    y\_data = [randint(-10, 10) for \_ in range(-10, 10+1)]

    return x\_data, y\_data

def menu():

    x = """

        0. Выход

        1. Ввести с клавиатуры

        2. Случайные значения

        3. Пример: sin(x) -10 <= x <= 10 с шагом 0.1

        4. Пример: x^2 -10 <= x <= 10 с шагом 0.1

        """

    print(x)

    pos = 0

    while True:

        pos = int(input('num-> '))

        if pos >= 0 and pos <= 4:

            break

    return pos

if \_\_name\_\_ == '\_\_main\_\_':

    settings = load\_settings()

    pos = -1

    x\_data = list()

    y\_data = list()

    left\_x = 0

    right\_x = 0

    step = 1

    while (True):

        pos = menu()

        if pos == 0:

            break

        elif pos == 1:

            x\_data, y\_data, left\_x, right\_x, step = read\_data\_from\_keyboard()

        elif pos == 2:

            x\_data, y\_data = random\_data()

            left\_x = -10

            right\_x = 10

            step = 0.05

        elif pos == 3:

            x\_data = list(range(-10, 10 + 1))

            y\_data = [math.sin(x) for x in range(-10, 10 + 1)]

            left\_x = -10

            right\_x = 10

            step = 0.1

        elif pos == 4:

            x\_data = list(range(-10, 10 + 1))

            y\_data = [x \*\* 2 for x in range(-10, 10 + 1)]

            left\_x = -10

            right\_x = 10

            step = 0.1

        spline = make\_spline(x\_data, y\_data)

        new\_x = list(np.arange(left\_x, right\_x + step, step))

        new\_y = list()

        for x in new\_x:

            new\_y.append(calc\_new\_point(spline, x))

        if settings['PRINT\_DATA']: print\_data(x\_data, y\_data)

        if settings['PRINT\_SPLINE']: print\_spline(spline)

        if settings['PRINT\_CALC']: print\_calc\_points(new\_x, new\_y)

        calc\_legend = plot\_new\_points(new\_x, new\_y, settings['PLOT\_CALC\_DOTS'])

        if settings['PLOT\_DATA']: data\_legend = plot\_data(x\_data, y\_data)

        if settings['PLOT\_CALC\_DOTS'] and settings['PLOT\_DATA']: plt.legend((data\_legend, calc\_legend), ('data', 'calc'))

        plt.gcf().canvas.set\_window\_title('Cubic Spline Interpolation')

        if settings['CENTRAL\_AXES']:

            ax = plt.gca()

            ax.spines['left'].set\_position('center')

            ax.spines['bottom'].set\_position('center')

            ax.spines['top'].set\_visible(False)

            ax.spines['right'].set\_visible(False)

        plt.grid()

        plt.show()

## interpolationLib.py

import matplotlib.pyplot as plt

import numpy as np

import math

class SplineDoesntExistYet(Exception):

    pass

def plot\_data(x\_data, y\_data):

    data\_leg = plt.scatter(x\_data, y\_data, c='green')

    return data\_leg

def plot\_new\_points(x\_points, y\_points, flag):

    if flag: calc\_leg = plt.scatter(x\_points, y\_points, c='deeppink')

    else: calc\_leg = None

    # the calculated values are connected by a line

    plt.plot(x\_points, y\_points)

    return calc\_leg

def print\_data(x\_data, y\_data):

    print('============= DATA =============')

    for i in range(len(x\_data)):

        print(x\_data[i], y\_data[i], sep='\t')

    print('================================')

def print\_spline(spline):

    print('\n\n============ SPLINE =============')

    for i in range(1, len(spline)):

        print(spline[i])

    print('=================================')

def print\_calc\_points(calc\_x, calc\_y):

    print('\n\n============= CALC =============')

    for i in range(len(calc\_x)):

        print(calc\_x[i], calc\_y[i], sep='\t')

    print('================================')

# returns coeffs for all spline - list of dictionaries(a,b,c,d,x),

# where x is needed to select the spline

def make\_spline(x\_data, y\_data):

    n = len(x\_data)

    spline = [{'a': y\_data[i], 'b': 0, 'c': 0, 'd': 0, 'x': x\_data[i]} for i in range(0, n)]

    # forward

    alpha = [0] \* (n - 1)

    beta  = [0] \* (n - 1)

    for i in range(1, n - 1):

        hi  = x\_data[i] - x\_data[i - 1]

        hi\_next = x\_data[i + 1] - x\_data[i]

        A = hi

        C = 2.0 \* (hi + hi\_next)

        B = hi\_next

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

        z = (A \* alpha[i - 1] + C)

        alpha[i] = -B / z

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

    # reverse

    for i in range(n - 2, 0, -1):

        spline[i]['c'] = alpha[i] \* spline[i + 1]['c'] + beta[i]

    for i in range(n - 1, 0, -1):

        hi = x\_data[i] - x\_data[i - 1]

        spline[i]['d'] = (spline[i]['c'] - spline[i - 1]['c']) / hi

        spline[i]['b'] = hi \* (2.0 \* spline[i]['c'] + spline[i - 1]['c']) / 6.0 + (y\_data[i] - y\_data[i - 1]) / hi

    # fix coeffs

    for i in range(len(spline)):

        spline[i]['c'] /= 2

        spline[i]['d'] /= 6

    return spline

# returns f(x), where f is selected spline

def calc\_new\_point(spline, x):

    if not spline:

        raise SplineDoesntExistYet('First you need to build a spline!')

    n = len(spline)

    spl = None

    if x <= spline[0]['x']: spl = spline[1]

    elif x >= spline[n-1]['x']: spl = spline[n-1]

    else:

        for i in range(1, n):

            if x > spline[i - 1]['x'] and x <= spline[i]['x']:

                spl = spline[i]

                break

    f = spl['a'] + spl['b'] \* (x - spl['x']) + spl['c'] \* ((x - spl['x']) \*\* 2) + spl['d'] \* ((x - spl['x']) \*\* 3)

    return f