**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Крагель Алина Олеговна

**Приближение функции с помощью кубического сплайна**

Отчет по лабораторной работе №2

(«Методы численного анализа»)

Студента 2 курса 10 группы

Работа сдана \_\_\_апреля 2021 г. Преподаватель:

Зачтена \_\_\_\_\_\_\_\_\_ 2021 г. Никифоров Иван Васильевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Доцент кафедры

(Подпись преподавателя) вычислительной математики

Минск 2021

**Постановка задачи**

Интерполировать функцию з) на интервале [-3, 0] с шагом 0.25 с помощью кубического сплайна.

**Теория**

Пусть задана на , разбитом на части . Кубический сплайн – функция , которая на каждом отрезке – многочлен степени не больше 3; имеет непрерывные 1 и 2 производные на всём если в точках выполняется сплайн интерполирует функцию в точках

На каждом отрезке функция полином 3 степени

причём:

Чтобы построить сплайн, необходимо наложить дополнительные требования:

* ;
* ;
* ;
* ;
* ;

На кубический сплайн представим в виде:

Вычисляем коэффициенты , решив с помощью метода прогонки соответствующую СЛАУ, представленную трёхдиагональной матрицей, удовлетворяющей условию строгого диагонального преобладания:

*, ,*

а) *,*

*, ,*

б)*,*

*, ,*

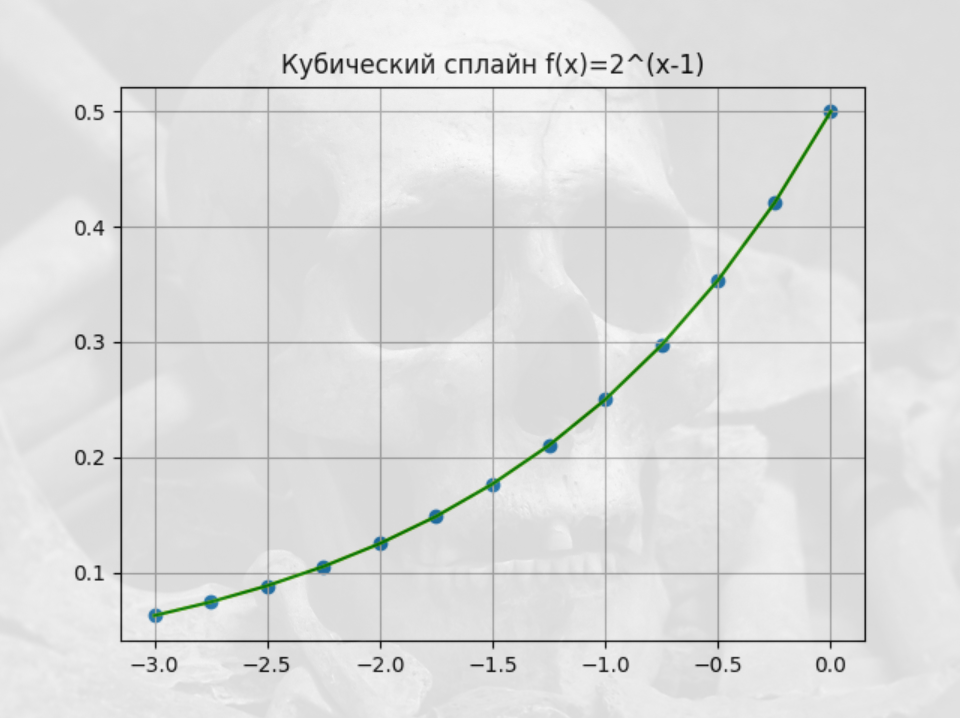
в) *,*

*, ,*

**Листинг**

import numpy as np  
from math import sqrt  
import matplotlib.pyplot as plt  
  
  
def cubic\_interp1d(x0, x, y):  
 if np.any(np.diff(x) < 0):  
 indexes = np.argsort(x)  
 x = x[indexes]  
 y = y[indexes]  
 size = len(x)  
 xdiff = np.diff(x)  
 ydiff = np.diff(y)  
 Li = np.empty(size)  
 Li\_1 = np.empty(size - 1)  
 z = np.empty(size)  
 Li[0] = sqrt(2 \* xdiff[0])  
 Li\_1[0] = 0.0  
 B0 = 0.0  
 z[0] = B0 / Li[0]  
 for i in range(1, size - 1, 1):  
 Li\_1[i] = xdiff[i - 1] / Li[i - 1]  
 Li[i] = sqrt(2 \* (xdiff[i - 1] + xdiff[i]) - Li\_1[i - 1] \* Li\_1[i - 1])  
 Bi = 6 \* (ydiff[i] / xdiff[i] - ydiff[i - 1] / xdiff[i - 1])  
 z[i] = (Bi - Li\_1[i - 1] \* z[i - 1]) / Li[i]  
 i = size - 1  
 Li\_1[i - 1] = xdiff[-1] / Li[i - 1]  
 Li[i] = sqrt(2 \* xdiff[-1] - Li\_1[i - 1] \* Li\_1[i - 1])  
 Bi = 0.0  
 z[i] = (Bi - Li\_1[i - 1] \* z[i - 1]) / Li[i]  
 i = size - 1  
 z[i] = z[i] / Li[i]  
 for i in range(size - 2, -1, -1):  
 z[i] = (z[i] - Li\_1[i - 1] \* z[i + 1]) / Li[i]  
 index = x.searchsorted(x0)  
 np.clip(index, 1, size - 1, index)  
 xi1, xi0 = x[index], x[index - 1]  
 yi1, yi0 = y[index], y[index - 1]  
 zi1, zi0 = z[index], z[index - 1]  
 hi1 = xi1 - xi0  
 f0 = zi0 / (6 \* hi1) \* (xi1 - x0) \*\* 3 + \  
 zi1 / (6 \* hi1) \* (x0 - xi0) \*\* 3 + \  
 (yi1 / hi1 - zi1 \* hi1 / 6) \* (x0 - xi0) + \  
 (yi0 / hi1 - zi0 \* hi1 / 6) \* (xi1 - x0)  
 return f0  
  
  
x = np.linspace(-3, 0, 13)  
y = np.power(2, x - 1)  
plt.scatter(x, y)  
x\_new = np.linspace(0, 10, 201)  
plt.plot(x, y, label=**"f(x)"**, color=**"pink"**, linestyle=**'-.'**)  
plt.plot(x, cubic\_interp1d(x, x, y), label=**"Ln(x)"**, color=**"green"**)  
plt.title(**"Кубический сплайн f(x)=2^(x-1)"**)  
plt.grid()  
plt.show()

**Результаты**

****

**Вывод**

Таким образом, в результате поставленного эксперимента удалось интерполировать данную на интервале с детермированным шагом функцию c помощью кубического сплайна.