Министерство науки и высшего образования Российской Федерации

Федерального государственного бюджетного образовательного учреждения высшего образования

«Казанский национальный исследовательский технологический университет»

(ФГБОУ ВО «КНИТУ»)

Кафедра интеллектуальных систем и управления информационными ресурсами

**ЛАБОРАТОРНАЯ РАБОТА №4**

по курсу  
«**Методы вычисления»**

на тему

**«Интерполяция**»

Выполнил:

студент группы 4321-23

Мвелва К.Ч.

Проверил:

доцент кафедры ИСУИР

Вотякова Л.Р.

Казань, 2024

Contents

[Задача 1: 3](#_Toc182666730)

[Решение 3](#_Toc182666731)

[Задача 2 6](#_Toc182666732)

[Решение: 6](#_Toc182666733)

[Задача 3 8](#_Toc182666734)

[Решение 8](#_Toc182666735)

[Задача 4 9](#_Toc182666736)

[Решение 10](#_Toc182666737)

[Задача 5 11](#_Toc182666738)

[Задача 6 12](#_Toc182666739)

[Задача 7 12](#_Toc182666740)

[Задача 8 12](#_Toc182666741)

**Переменные:**

β - 1 ϑ - 3

γ - 2 μ - 0

ν - 7 α – 4

# Задача 1:

Найти уравнение полинома, проходящего через точки ниже представленной таблицы и вычислить его значения в точках и .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | -1 | 1 | 3 | 4 |
| y | 8 | 9 | 6 | 7 |

## Решение

Так как у нас 4 точки данных и степень полинома определяется по выражению *n-1,* где n — количество точки данных.

Получим полином общего вида:

Подставляем соответствующие переменные x и y, получаем:

, , ;

import numpy as np

#Вводим исходные данные

x\_values = np.array([-1, 1, 3, 4])

y\_values = np.array([8, 9, 6, 7])

#Предсатавим матрицу коэффициентов для уравнения

A = np.vstack([x\_values\*\*3,x\_values\*\*2,x\_values, np.ones(len(x\_values))]).T

#Находим и печатаем решение

coeff = np.linalg.solve(A, y\_values)

print("Coefficients of the polynomials:")

print(coeff)

#Для вычисления значения полинома

def polynomial(x):

    return coeff[0]\*x\*\*3 + coeff[1]\*x\*\*2 + coeff[2]\*x + coeff[3]

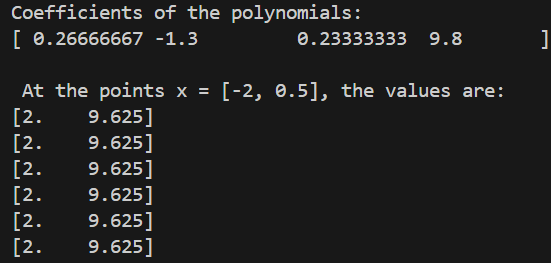
#Нахождение решения в заданных точках

x\_data = np.array([-2,0.5])

y\_data = polynomial(x\_data)

print("\n At the points x = [-2, 0.5], the values are:")

print (y\_data)



import scipy.optimize

import numpy as np

import matplotlib.pyplot as plt

x\_values = np.array ([-1, 1, 3, 4])

y\_values = np.array ([8, 9, 6, 7])

def polynomial(t, a, b, c, d):

    return a\*t\*\*3+b\*t\*\*2+c\*t+d

def graph(z):

    a = z[0]

    b = z[0]

    c = z[0]

    d = z[0]

    f = np.zeros(4)

    for i in range(len(x\_values)):

        f[i] = polynomial(x\_values[i],a,b,c,d)-y\_values[i]

        return f

z = scipy.optimize.fsolve(graph,[0.267,-1.3,0.2333,9.8])

print ("The solution to the system of equations is: a = %.6f b= %.6f, c = %.6f, d = %.6f "%

       (z[0],z[1],z[2],z[3]))

print("The values of the left part of the first equation are %.12f" %graph(z)[0])

print("The values of the left part of the second equation are %.12f" %graph(z)[1])

print("The values of the left part of the third equation are %.12f" %graph(z)[2])

print("The values of the left part of the fourth equation are %.10f" %graph(z)[3])

print("The equation of the curve Y = (%.6f)x^3 + %.6fx^2+ %.6fx + %.6g" %(z[0],z[1],z[2],z[3]))

p1 = -2; q = polynomial(p1,z[0],z[1],z[2],z[3])

p2 = 0.5; r = polynomial(p2,z[0],z[1],z[2],z[3])

print("The value of the function at point %.1f equals %.6f"% (p1,q))

print("The value of the function at point %.1f equals %.6f"% (p2,r))

fig = plt.figure()

#ax = fig.add\_subplot(111)

a1,b1 =  min(x\_values), max(x\_values)

x1 = np.linspace(a1,b1,10)

plt.grid()

plt.plot(x\_values,y\_values,'ob',label = "SOURCE DATA")

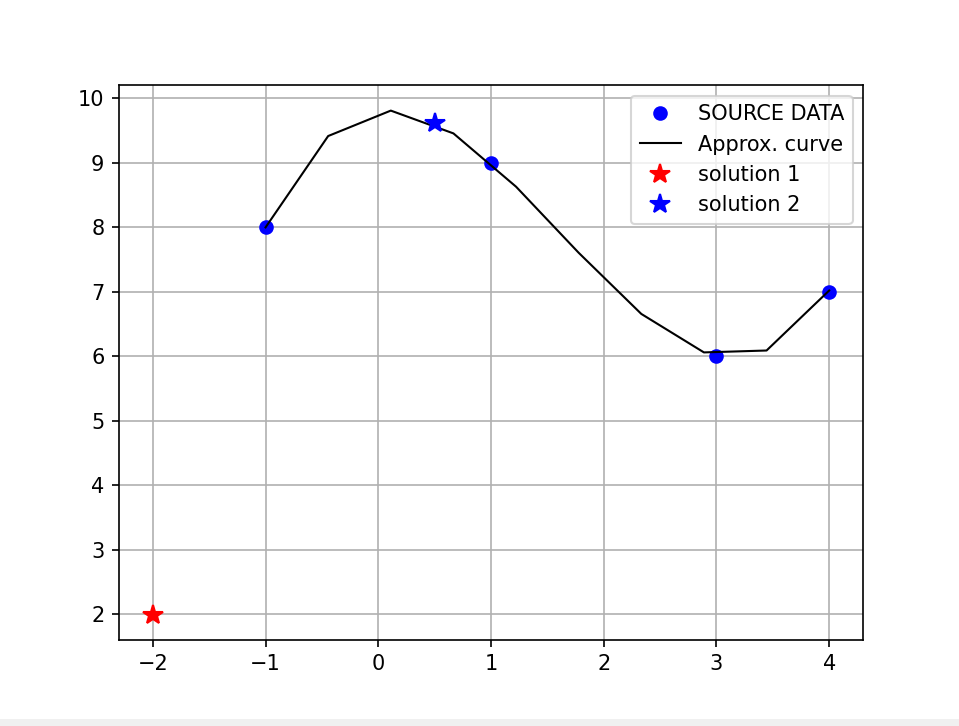
plt.plot(x1, polynomial(x1,z[0],z[1],z[2],z[3]), linewidth = 1, c = 'k', label = "Approx. curve")

plt.plot(p1,q,"\*r",ms = 10, label = "solution 1")

plt.plot(p2,r,"\*b",ms = 10, label = "solution 2")

plt.legend()

plt.show()



# Задача 2

Подобрать коэффициенты зависимости таким образом, чтобы полученная кривая прошла через точки таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| x | 2 | 3 | 12 |
| y | 4 | 0 | 24 |

## Решение:

Представим данные в систему уравнений

import numpy as np

from scipy.optimize import root

import matplotlib.pyplot as plt

def equation\_find(x):

    c1,c2,c3 = x

    eq1 = c1\*np.log(2) + 4\*c2\*np.exp(-1) + 8\*c3 - 4

    eq2 = c1\*np.log(3) + 9\*c2\*np.exp(-1.5) + 27\*c3 - 0

    eq3 = c1\*np.log(12) + 144\*c2\*np.exp(-6) + 1728\*c3 -24

    return [eq1, eq2,eq3]

x0 = np.array([1,1,1])

solution = root(equation\_find,x0)

c1,c2,c3 = solution.x

def func(x):

    return c1\*np.log(x) + c2\*x\*\*2\*np.exp(-0.5\*x) + c3\*x\*\*3

x\_values = np.array([2,3,12])

y\_values = np.array([4,0,24])

x\_plot = np.linspace(1, 20, 100)

y\_plot = func(x\_plot)

#Строим графику

plt.plot(x\_values, y\_values, 'ob', label = "Исходные данные")

plt.plot(x\_plot ,y\_plot, '-', label = "Аппроксимация")

plt.xlabel('x')

plt.ylabel('y')

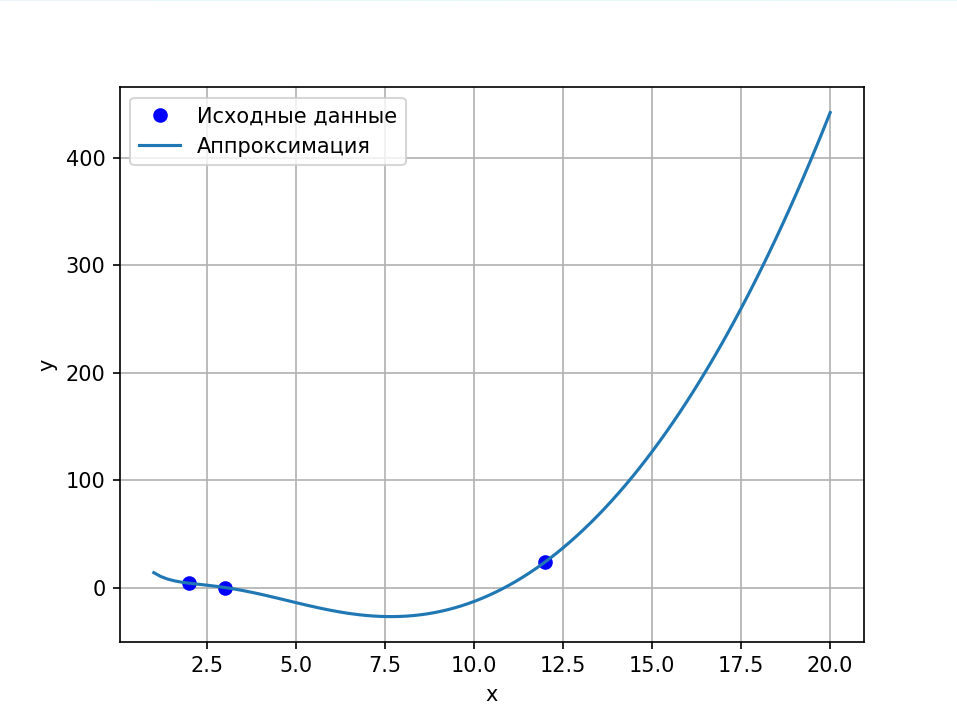
plt.legend()

plt.grid(True)

plt.show()

print ("Коэффициенты: ")

print (c1, c2, c3)





# Задача 3

Найти уравнение двух кубических сплайнов, проходящих через точки таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| x | 1 | 2 | 4 |
| y | 4 | 7 | 25 |

Значение вторых производных от каждого сплайна в первой и последней точках таблицы положить равными нулю.

## Решение

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import CubicSpline

#Вводим данные:

x = np.array([1, 2, 4])

y = np.array([4, 7, 25])

# Создание кубического сплайна с естественными граничными условиями

spline = CubicSpline(x, y, bc\_type='natural')

# Создаем массив x для построения сплайна

x\_plot = np.linspace(min(x), max(x), 100)

# Вычисляем значения сплайна

y\_plot = spline(x\_plot)

# Построение графика

plt.scatter(x, y, color='green', label='Source Data')

plt.plot(x\_plot, y\_plot, label='Cubic Spline')

plt.grid()

plt.legend()

plt.xlabel('x')

plt.ylabel('y')

plt.title('Spline')

plt.show()

# Коэффициенты сплайна на интервалах

coeffs = spline.c

# Уравнения сплайнов на интервалах [1, 2] и [2, 4]

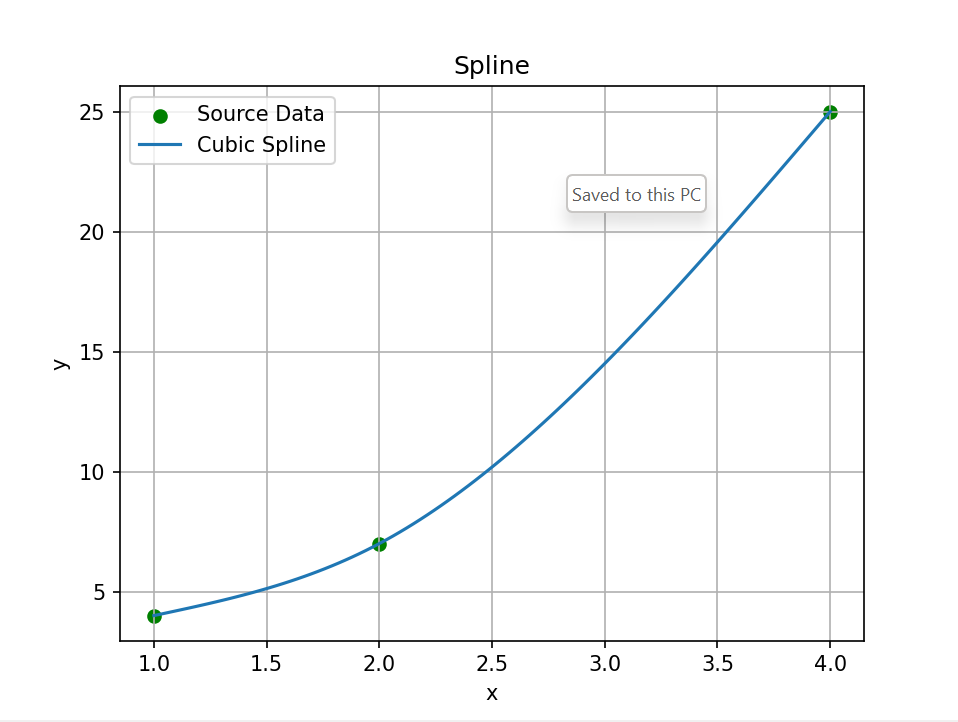
for i in range(len(x) - 1):

  a, b, c, d = coeffs[:, i]

  # Выводим коэфф.

  print(f"Уравнение сплайна на интервале [{x[i]}, {x[i+1]}]:")

  print(f"y = {a:.4f}(x - {x[i]})^3 + {b:.4f}(x - {x[i]})^2 + {c:.4f}(x - {x[i]}) + {d:.4f}")



# Задача 4

Функция задана в виде таблицы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 1 | ¼ | 6 | 7 |
| y | -1 | 6 | 18 | 0 |

Вычислить значение функции в точках x∗=3.4x и x∗=4.4. Решить задачу интерполяции с помощью функции *interp1d(*), используя квадратичную и кубическую интерполяцию. Дать графическое решение задачи.

## Решение

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import interp1d

#Вводим данных

x = np.array([1, 2, 6, 7])

y = np.array([-1, 6, 18, 0])

#Создание функций с помощью библммотеку interpld

quadratic\_interp = interp1d(x, y, kind='quadratic')

cubic\_interp = interp1d(x, y, kind='cubic')

#Вычисление значений функции в точках x=3, 4 и x=4.4

x\_new = np.array([3, 4, 4.4])

# Вычисление значений с использованием квадратичной интерполяции

y\_quadratic = quadratic\_interp(x\_new)

# Вычисление значений с использованием кубической интерполяции

y\_cubic = cubic\_interp(x\_new)

print("В точках x=3, 4, 4.4 с квадратичной интерпояции:")

print(y\_quadratic)

print("В точках x=3, 4, 4.4 с кубической интерполяции:")

print(y\_cubic)

# Создание графика

# Создание сетки для построения графика

x\_plot = np.linspace(min(x), max(x), 100)

# Вычисление значений интерполяции на сетке

y\_quadratic\_plot = quadratic\_interp(x\_plot)

y\_cubic\_plot = cubic\_interp(x\_plot)

plt.figure(figsize=(10, 6))

plt.scatter(x, y, color='red', label='Исходные данные')

plt.plot(x\_plot, y\_quadratic\_plot, label='Квадратичная интерполяция')

plt.plot(x\_plot, y\_cubic\_plot, label='Кубическая интерполяция')

#В которых вычислялись значения

plt.scatter(x\_new, y\_quadratic, color='green', label='Значения (квадратичная)')

plt.scatter(x\_new, y\_cubic, color='orange', label='Значения (кубическая)')

plt.xlabel('x')

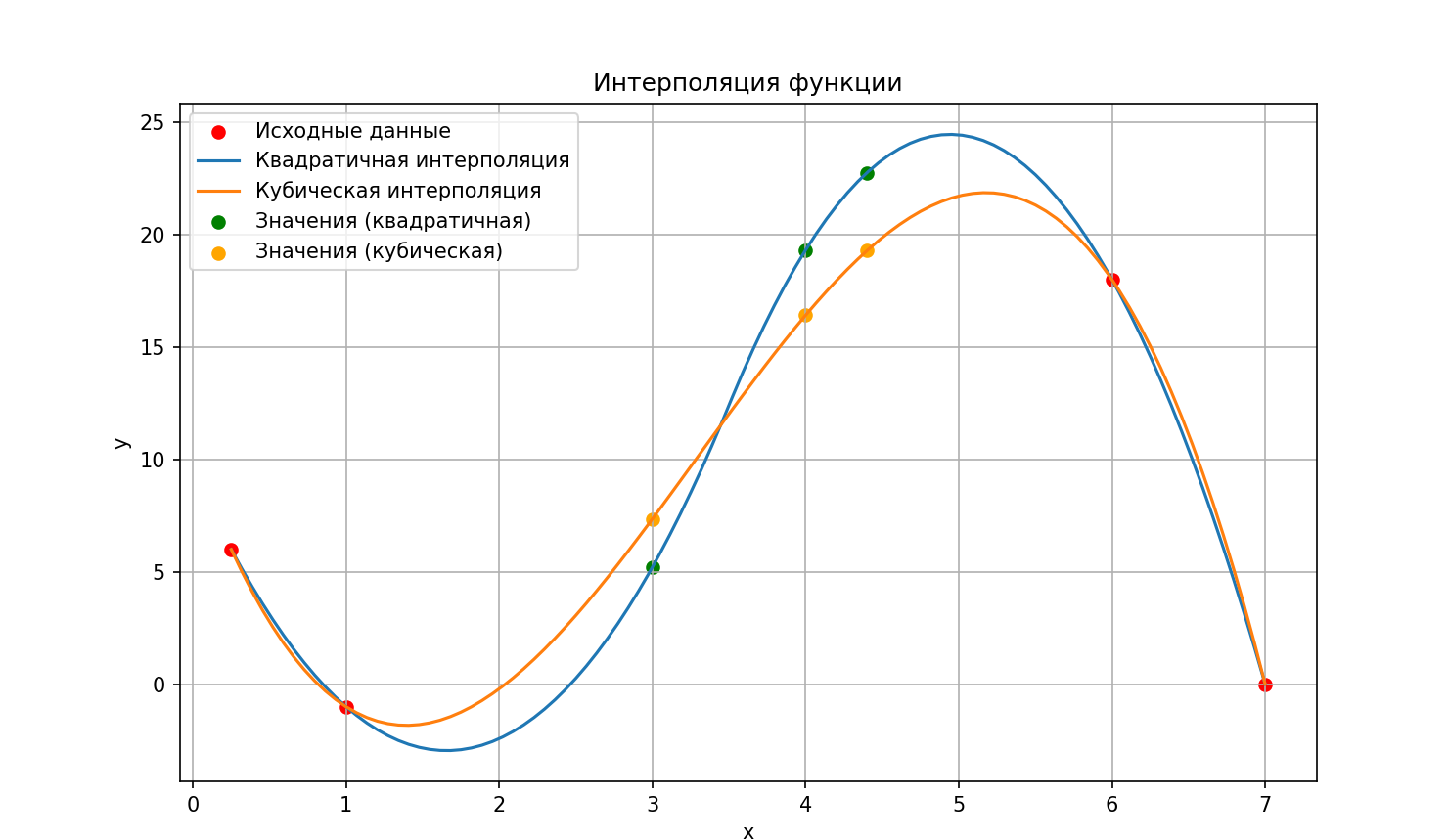
plt.ylabel('y')

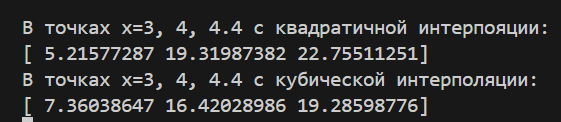
plt.title('Интерполяция функции')

plt.legend()

plt.grid(True)

plt.show()





# Задача 5

По данным таблицы построить графики *B-*сплайнов 2-й и 3-й степени на отрезке [1,5]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 1 | 2.7 | 3 | 4 | 5 |
| y | -2 | 1 | 0 | 9 | 7 |

## Решение

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import splrep, splev

# Вводим данные

x = np.array([1, 2.7, 3, 4, 5])

y = np.array([-2, 1, 0, 9, 7])

# Сортировка точек по x

sorted\_indices = np.argsort(x)

x\_sorted = x[sorted\_indices]

y\_sorted = y[sorted\_indices]

# Создание B-сплайнов 2-й и 3-й степени

tck\_2 = splrep(x\_sorted, y\_sorted, k=2)

tck\_3 = splrep(x\_sorted, y\_sorted, k=3)

# Для построения сплайнов

x\_plot = np.linspace(min(x\_sorted), max(x\_sorted), 100)

y\_plot\_2 = splev(x\_plot, tck\_2)

y\_plot\_3 = splev(x\_plot, tck\_3)

# Построение графиков

plt.scatter(x\_sorted, y\_sorted, color='blue', label='Исходные данные')

plt.plot(x\_plot, y\_plot\_2, label='B-сплайн 2-й степени')

plt.plot(x\_plot, y\_plot\_3, label='B-сплайн 3-й степени')

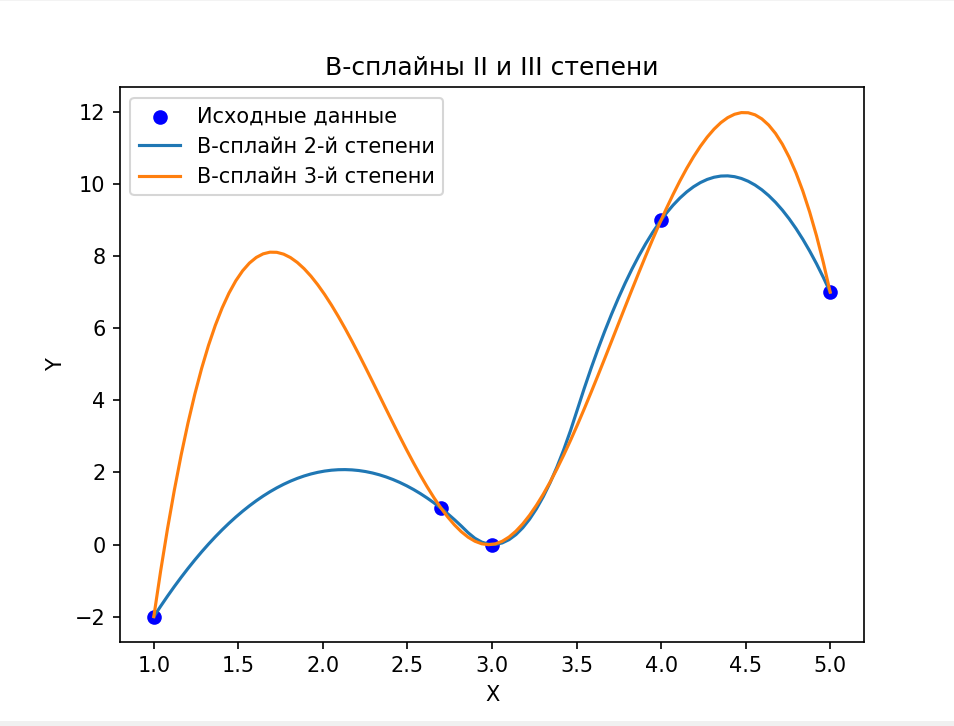
plt.legend()

plt.xlabel('X')

plt.ylabel('Y')

plt.title('B-сплайны II и III степени')

plt.show()



# Задача 6

Найти уравнения кубических B-сплайнов, полученных с помощью функции *spler()* для таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 1 | 2 | 4 | 5 | 7 |
| y | 0 | 1 | 22 | 0 | 8 |

Вычислить значения сплайнов в точках х\* = 1.5 их\*= 8.5. Дать графическое решение задачи. Вычислить значения первой и второй производных в точках х\*.

## Решение

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import splrep, splev

# Исходные данные

x = np.array([1, 2, 4, 5, 7])

y = np.array([0, 1, 22, 0, 8])

# Создание кубических B-сплайнов

tck = splrep(x, y, k=3)

# Вычисление значений сплайнов в точках x\*=1.5 и x\*=8.5

x\_val1 = 1.5

x\_val2 = 8.5

y\_val1 = splev(x\_val1, tck)

y\_val2 = splev(x\_val2, tck, ext=0)  # Используем ext=0 для продолжения сплайна за пределами интервала данных

print(f"Значение сплайна в точке x\* = {x\_val1}: y\* = {y\_val1:.2f}")

print(f"Значение сплайна в точке x\* = {x\_val2}: y\* = {y\_val2:.2f}")

# Вычисление первой производной

dy\_star\_1 = splev(x\_val1, tck, der=1)

dy\_star\_2 = splev(x\_val2, tck, der=1, ext=0)  # Используем ext=0 для продолжения сплайна за пределами интервала данных

# Вычисление второй производной

d2y\_star\_1 = splev(x\_val1, tck, der=2)

d2y\_star\_2 = splev(x\_val2, tck, der=2, ext=0)  # Используем ext=0 для продолжения сплайна за пределами интервала данных

print(f"Значение первой производной в точке x\* = {x\_val1}: dy/dx = {dy\_star\_1:.2f}")

print(f"Значение первой производной в точке x\* = {x\_val2}: dy/dx = {dy\_star\_2:.2f}")

print(f"Значение второй производной в точке x\* = {x\_val1}: d2y/dx2 = {d2y\_star\_1:.2f}")

print(f"Значение второй производной в точке x\* = {x\_val2}: d2y/dx2 = {d2y\_star\_2:.2f}")

# Создаем массив x для построения сплайна, включая точку x\_star\_2

x\_plot = np.linspace(min(x), x\_val2, 100)

# Вычисляем значения сплайна

y\_plot = splev(x\_plot, tck, ext=0)  # Используем ext=0 для продолжения сплайна за пределами интервала данных

# Построение графика

plt.scatter(x, y, color='red', label='Исходные данные')

plt.plot(x\_plot, y\_plot, label='Кубический B-сплайн')

plt.scatter([x\_val1, x\_val2], [y\_val1, y\_val2], color='green', label='Значения в x\*')

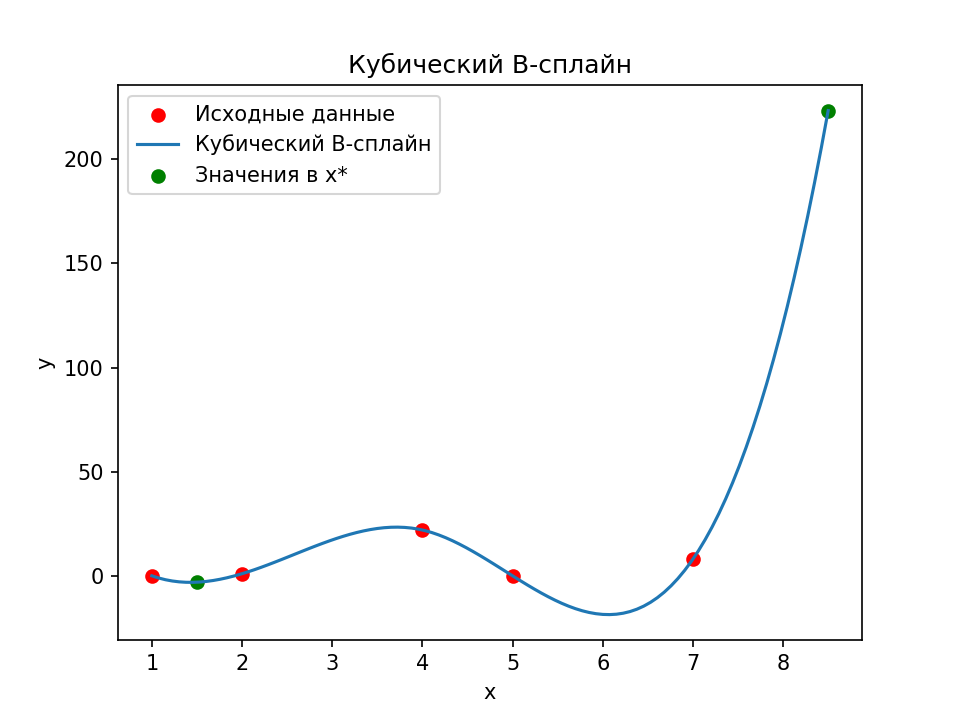
plt.legend()

plt.xlabel('x')

plt.ylabel('y')

plt.title('Кубический B-сплайн')

plt.show()



# Задача 7

Найти корни B-сплайна 3-й степени, построенного по таблице

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 1 | 2 | 3 | 5 |
| y | -6 | 0 | 7 | -6 |

## Решение

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import CubicSpline

from scipy.optimize import fsolve

# Исходные данные

x = np.array([1, 2, 3, 5])

y = np.array([-6, 0, 7, -6])

# Построение кубического сплайна

cubic\_spline = CubicSpline(x, y)

# Функция для поиска корней

def find\_roots(spline, x\_min, x\_max, num\_intervals=100):

    roots = []

    interval\_size = (x\_max - x\_min) / num\_intervals

    for i in range(num\_intervals):

        x\_start = x\_min + i \* interval\_size

        x\_end = x\_start + interval\_size

        root = fsolve(spline, (x\_start + x\_end) / 2)

        if x\_start <= root <= x\_end:

            roots.append(root[0])

    return np.unique(roots)

# Нахождение корней сплайна

roots = find\_roots(cubic\_spline, x[0], x[-1])

print("Корни B-сплайна 3-й степени:", roots)

# Создаем массив x для построения сплайна

x\_plot = np.linspace(x[0], x[-1], 100)

# Вычисляем значения сплайна

y\_plot = cubic\_spline(x\_plot)

# Построение графика

plt.scatter(x, y, color='blue', label='SOURCE DATA')

plt.grid (True)

plt.plot(x\_plot, y\_plot, label='Cubic Spline')

plt.scatter(roots, cubic\_spline(roots), color='red', label='Roots of the spline')

plt.legend()

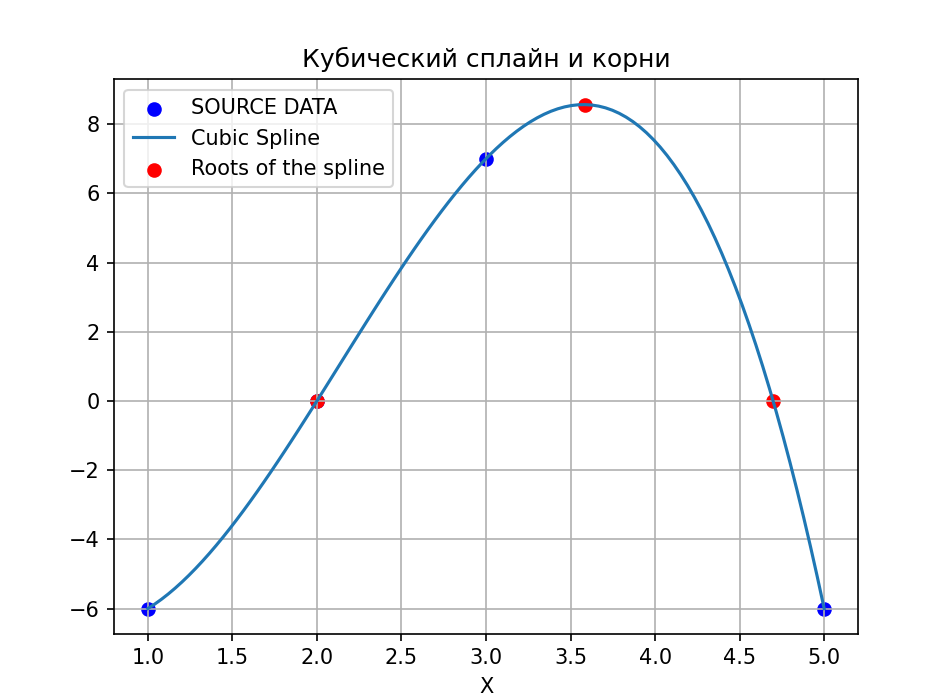
plt.xlabel('X')

plt.ylabel('')

plt.title('Кубический сплайн и корни')

plt.show()





# Задача 8

Решить задачу интерполяции для таблицы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 1 | 2 | 4 | 5 | 7 |
| y | 0 | 1 | 22 | 0 | 8 |

С помощью радиальной базисной функции. Вычислить значение функции в точке x\*=1.5. В качестве RBF выбрать функцию () Параметр взять равным 2.

## Решение

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import Rbf

# Исходные анные

X = np.array([1, 2, 4, 5, 7])

y = np.array([0, 1, 22, 0, 8])

# Определение радиуса

epsilon = 2

# Построение интерполятора с использованием RBF

rbf\_interpolator = Rbf(X, y, function='multiquadric', epsilon=epsilon)

# Вычисление значения функции в точке x\* = 1.5

x\_star = 1.5

y\_star = rbf\_interpolator(x\_star)

# Печать результата

print(f"Значение интерполированной функции в точке x\* = {x\_star}: {y\_star}")

# Построение графика интерполяции

x\_vals = np.linspace(1, 7, 100)

y\_vals = rbf\_interpolator(x\_vals)

plt.plot(x\_vals, y\_vals, label='Интерполированный RBF')

plt.grid(True)

plt.scatter(X, y, color='blue', label='Исходные значении')

plt.title('Интерполяция с помощью RBF')

plt.legend()

plt.show()

