

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

|  |
| --- |
| **ШКОЛА ЕСТЕСТВЕННЫХ НАУК**  **кафедра информатики, математического и компьютерного моделирования** |

**О Т Ч Е Т**

о выполнении лабораторной работы №1 по предмету «Методы сплайн-функций»

Направление подготовки 01.03.02 «Прикладная математика и информатика»

|  |  |  |
| --- | --- | --- |
|  |  | Выполнил студент гр. Б8116-01.03.02-ПМИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Глушков В.К. |
|  |  | (*подпись*) *(Ф.И.О.)* |
|  |  |  |

г. Владивосток

2020

## Цель работы

Построить кубический сплайн класса c первым типом краевых условий по наклонам для заданных функций с параметром. Оценить погрешность при разных значениях параметра .

## Задание

Построить кубический сплайн по наклонам с граничными условиями -ого типа для функций с параметром:

|  |  |
| --- | --- |
| , где | (1) |
| , где | (2) |

## Выполнение задание

Интерполяционным кубическим сплайном называется сплайн, удовлетворяющий условиям

|  |  |
| --- | --- |
|  | (3) |

Пусть на отрезке [] в узлах сетки заданы значения некоторой функции

Необходимо определить коэффициентов. Условие выше дает равенств, а условие дает равенств, итого соотношений. Два оставшихся условия возьмем из краевых условий -ого типа:

|  |  |
| --- | --- |
|  | (4) |

Выразим интерполяционный кубический сплайн с помощью величин :

|  |  |
| --- | --- |
|  | (5) |

Выразив необходимые производные и требования непрерывности, получим следующую систему уравнений:

|  |  |
| --- | --- |
|  | (6) |

Где

|  |  |
| --- | --- |
|  | (7) |

## Код программы

|  |
| --- |
| main.py |
| import numpy as np import matplotlib.pyplot as plt  from lab1.configurations import configurations from lab1.utils import h, calc\_matrix   def t(x, x\_, i):  return (x\_ - x[i]) / h(x, i)   def s(x\_, x, m\_, y\_, n\_, diff, a):  for i in range(0, n\_):  if x[i] <= x\_ <= x[i + 1]:  a\_ = 6 \* ((y\_[i + 1] - y\_[i]) / h(x, i + 1) - (m\_[i + 1] + 2 \* m\_[i]) / 3) / h(x, i + 1)  b\_ = 12 \* ((m\_[i + 1] + m\_[i]) / 2 - (y\_[i + 1] - y\_[i]) / h(x, i + 1)) / np.power(h(x, i + 1), 2)  return y\_[i] + diff(a, x\_) \* (x\_ - x[i]) + a\_ \* np.power(x\_ - x[i], 2) / 2 + b\_ \* np.power(x\_ - x[i], 3) / 6   def main(configuration):  func = configuration["function"]  diff = configuration["diff"]  diff2 = configuration["diff2"]  a = configuration["a"]  n = configuration["n"]  x = np.linspace(configuration["x\_start"], configuration["x\_end"], num=n + 1)  y = np.array([func(a, x\_) for x\_ in x])   m\_res = calc\_matrix(n, x, y, diff2, a)   x\_dots = np.linspace(configuration["x\_start"], configuration["x\_end"], num=10000)  y\_dots = func(a, x\_dots)  s\_dots = np.array([s(x\_, x, m\_res, y, n, diff, a) for x\_ in x\_dots])  plt.plot(x\_dots, y\_dots, label="func")  plt.plot(x\_dots, s\_dots, label="spline")  plt.legend()  plt.show()   d = sum([abs(y\_dots[i] - s\_dots[i]) for i in range(0, len(y\_dots))]) / len(y\_dots)  print(np.sqrt(d))  return 0   if \_\_name\_\_ == "\_\_main\_\_":  main(configurations[0]) |
| functions.py |
| import numpy as np   def func1(a, x):  return np.sin(a \* x)   def func1dx(a, x):  return a \* np.cos(a \* x)   def func1dx2(a, x):  return (-1) \* np.power(a, 2) \* np.sin(a \* x)   def func2(a, x):  return a / (1 + 9 \* np.power(x, 2))   def func2dx(a, x):  return 18 \* a \* x / np.power(9 \* np.power(x, 2) + 1, 2)   def func2dx2(a, x):  return a \* (648 \* np.power(x, 2) / np.power(9 \* np.power(x, 2) + 1, 3) - 18 / np.power(9 \* np.power(x, 2) + 1, 2)) |
| configurations.py |
| import numpy as np  from lab1.functions import func1, func1dx, func1dx2, func2, func2dx, func2dx2  # configurations for labs 1 and 2 configurations = [  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 1, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 0  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 2, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 1  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 3, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 2  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 4, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 3  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 5, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 4  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 6, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 5  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 7, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 6  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 8, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 7  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 9, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 8  {"function": func1, "diff": func1dx, "diff2": func1dx2, "a": 10, "n": 100, "x\_start": 0, "x\_end": np.pi}, # 9   {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 1, "n": 100, "x\_start": -1, "x\_end": 1}, # 10  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 2, "n": 100, "x\_start": -1, "x\_end": 1}, # 11  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 3, "n": 100, "x\_start": -1, "x\_end": 1}, # 12  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 4, "n": 100, "x\_start": -1, "x\_end": 1}, # 13  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 5, "n": 100, "x\_start": -1, "x\_end": 1}, # 14  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 6, "n": 100, "x\_start": -1, "x\_end": 1}, # 15  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 7, "n": 100, "x\_start": -1, "x\_end": 1}, # 16  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 8, "n": 100, "x\_start": -1, "x\_end": 1}, # 17  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 9, "n": 100, "x\_start": -1, "x\_end": 1}, # 18  {"function": func2, "diff": func2dx, "diff2": func2dx2, "a": 10, "n": 100, "x\_start": -1, "x\_end": 1} # 19 ] |
| utils.py |
| import numpy as np   def h(x, i):  return x[i + 1] - x[i] if i + 1 < len(x) else x[i] - x[i - 1]   def m(x, i):  return h(x, i - 1) / (h(x, i + 1) + h(x, i))   def l(x, i):  return h(x, i) / (h(x, i + 1) + h(x, i))   # returns list of answer for matrix def calc\_matrix(n, x, y, diff2, a):  m\_a = np.zeros((n + 1) \*\* 2).reshape(n + 1, n + 1)  # m\_a[0][0] = 2  m\_a[0][0] = (y[1] - y[0]) / h(x, 0)  m\_a[0][1] = 1  for i in range(1, n):  m\_a[i][i - 1] = m(x, i)  m\_a[i][i] = 2  m\_a[i][i + 1] = l(x, i)  m\_a[n][n - 1] = 1  m\_a[n][n] = (y[n] - y[n - 1]) / h(x, 0)  # m\_a[n][n] = 2   m\_b = np.zeros(n + 1)  m\_b[0] = 3 \* (y[1] - y[0]) / h(x, 1) - h(x, 1) \* diff2(a, x[0]) / 2  for i in range(1, n):  m\_b[i] = 3 \* (m(x, i) \* (y[i + 1] - y[i]) / h(x, i + 1) + l(x, i) \* (y[i] - y[i - 1]) / h(x, i))  m\_b[n] = 3 \* ((y[n] - y[n - 1]) / h(x, n)) + h(x, n) \* diff2(a, x[n]) / 2   m\_res = np.linalg.solve(m\_a, m\_b)  return m\_res |

## Результат

|  |  |  |
| --- | --- | --- |
| Периодическая функция | | |
| Параметр | 1 | 3 |
| Ошибка | 0.02203224608042187 | 0.04620700736199554 |
| График |  |  |

|  |  |  |
| --- | --- | --- |
| Периодическая функция | | |
| Параметр | 6 | 10 |
| Ошибка | 0.09050781221109407 | 0.14887157022114095 |
| График |  |  |

|  |  |  |
| --- | --- | --- |
| Непериодическая функция | | |
| Параметр | 1 | 3 |
| Ошибка | 0.04245629716045448 | 0.07349602298591408 |
| График |  |  |

|  |  |  |
| --- | --- | --- |
| Непериодическая функция | | |
| Параметр | 6 | 10 |
| Ошибка | 0.10392685843309178 | 0.13416514581258107 |
| График |  |  |

## Вывод

Был успешно построен кубический интерполяционный сплайн класса с первым краевым условием. Установлено, что с увеличением параметра возрастает и погрешность интерполяции как в случае с периодической функцией, так и в случае с непериодической.