|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

***Лабораторная работа № 3***

**Тема:** Построение и программная реализация алгоритма сплайн-интерполяции табличных функций.

**Студент:** Артемьев И.О.

**Группа:** ИУ7-43Б

**Оценка (баллы):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Преподаватель:** Градов В.М.

*Москва*

*2021 г*

**Цель работы**

Получение навыков владения методами интерполяции таблично заданных функций с помощью кубических сплайнов.

**Исходные данные**

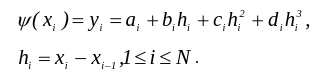
1. Значение аргумента x
2. Таблица функции с количеством узлов N, которая задается с помощью формулы y = x\*x в диапазоне [0..10] с шагом 1

**Выходные данные**

1. Значение y(x)
2. Сравнить результат интерполяции кубического сплайна с полиномом Ньютона 3-ей степени

**Анализ алгоритма**

Для кубического сплайна значение y вычисляется следующим образом



Прежде необходимо найти все коэффиценты:

1. Коэффициент a:

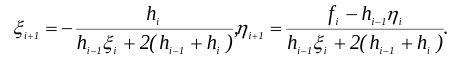


1. Коэффициент c (вычисляется раньше, поскольку участвует в вычислении коэффициентов b и d):

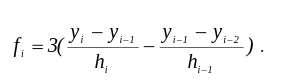
С помощью метода Гаусса получаем, что



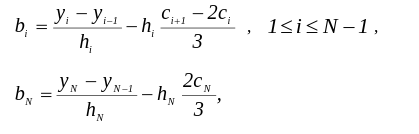
Затем сами прогоночные коэффициенты должны вычисляться следующим образом



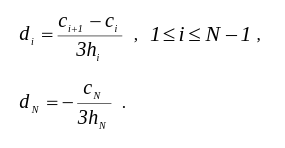
Используя дополнительное обозначение:



1. Коэффициент b:



1. Коэффициент d:



**Код программы**

"""

main.py

"""

from process\_funcs import get\_val\_pol # С прошлых лабораторных

from myio import read\_data

from spline import Spline

*def* main():

table = read\_data("test.txt")

if table is None:

exit(0)

newton\_table = []

for i in range (len(table[0])):

newton\_table.append([table[0][i], table[1][i]])

x = *float*(input("Input x: "))

print("Spline: ", "{:.5f}".format(Spline(table).get\_spline(x)))

print("The Newton polynomial of the third degree: ", get\_val\_pol(newton\_table, 3, x))

if \_\_name\_\_ == "\_\_main\_\_":

main()

"""

spline.py

"""

*class* Spline:

*def* \_\_init\_\_(*self*, *table*):

*self*.table = table

*self*.x\_arr = table[0]

*self*.y\_arr = table[1]

*self*.size = len(*self*.y\_arr)

*def* \_\_get\_a\_coeff(*self*):

return *self*.y\_arr[:-1]

*def* \_\_get\_b\_coeff(*self*, *c\_coeff*):

b\_coeff = []

for i in range(1, *self*.size - 1):

h = *self*.x\_arr[i] - *self*.x\_arr[i - 1]

b\_cur = (*self*.y\_arr[i] - *self*.y\_arr[i - 1]) / h - (h \* (c\_coeff[i] + 2 \* c\_coeff[i - 1])) / 3

b\_coeff.append(b\_cur)

h = *self*.x\_arr[*self*.size - 1] - *self*.x\_arr[*self*.size - 2]

b\_coeff.append((*self*.y\_arr[*self*.size - 1] - *self*.y\_arr[*self*.size - 2]) / h - (h \* 2 \* c\_coeff[i]) / 3)

return b\_coeff

*def* \_\_get\_c\_coeff(*self*):

c\_coeff = [0] \* (*self*.size - 1)

ksi\_arr = [0, 0]

teta\_arr = [0, 0]

for i in range(2, *self*.size):

h1 = *self*.x\_arr[i] - *self*.x\_arr[i - 1]

h2 = *self*.x\_arr[i - 1] - *self*.x\_arr[i - 2]

fi = 3 \* ((*self*.y\_arr[i] - *self*.y\_arr[i - 1]) / h1 - (*self*.y\_arr[i - 1] - *self*.y\_arr[i - 2]) / h2)

ksi\_cur = - h1 / (h2 \* ksi\_arr[i - 1] + 2 \* (h2 + h1))

teta\_cur = (fi - h1 \* teta\_arr[i - 1]) / (h1 \* ksi\_arr[i - 1] + 2 \* (h2 + h1))

ksi\_arr.append(ksi\_cur)

teta\_arr.append(teta\_cur)

c\_coeff[*self*.size - 2] = teta\_arr[len(teta\_arr) - 1]

for i in range(*self*.size - 2, 0, -1):

c\_coeff[i - 1] = ksi\_arr[i] \* c\_coeff[i] + teta\_arr[i]

return c\_coeff

*def* \_\_get\_d\_coeff(*self*, *c\_coeff*):

d\_coeff = []

for i in range(1, len(*self*.x\_arr) - 1):

h = *self*.x\_arr[i] - *self*.x\_arr[i - 1]

d\_cur = (c\_coeff[i] - c\_coeff[i - 1]) / (3 \* h)

d\_coeff.append(d\_cur)

h = *self*.x\_arr[*self*.size - 1] - *self*.x\_arr[*self*.size - 2]

d\_coeff.append((- c\_coeff[i]) / (3 \* h))

return d\_coeff

*def* get\_spline\_coeffs(*self*):

a\_coeff = *self*.\_\_get\_a\_coeff()

c\_coeff = *self*.\_\_get\_c\_coeff()

b\_coeff = *self*.\_\_get\_b\_coeff(c\_coeff)

d\_coeff = *self*.\_\_get\_d\_coeff(c\_coeff)

return a\_coeff, b\_coeff, c\_coeff, d\_coeff

*def* get\_spline(*self*, *x*):

coeffs = *self*.get\_spline\_coeffs()

pos = 1

while (pos < len(*self*.x\_arr) and *self*.x\_arr[pos] < x):

pos += 1

pos -= 1

h = x - *self*.x\_arr[pos]

result = 0

for i in range(len(coeffs)):

result += coeffs[i][pos] \* pow(h, i)

return result

"""

myio.py

"""

*def* read\_data(*filename*):

try:

file = open(filename, "r")

except:

print("Invalid filename")

return

table = []

x\_arr = []

y\_arr = []

for line in file:

try:

arr = [*float*(numb) for numb in line.split()]

x\_arr.append(arr[0])

y\_arr.append(arr[1])

except:

print("Invalid line data")

file.close()

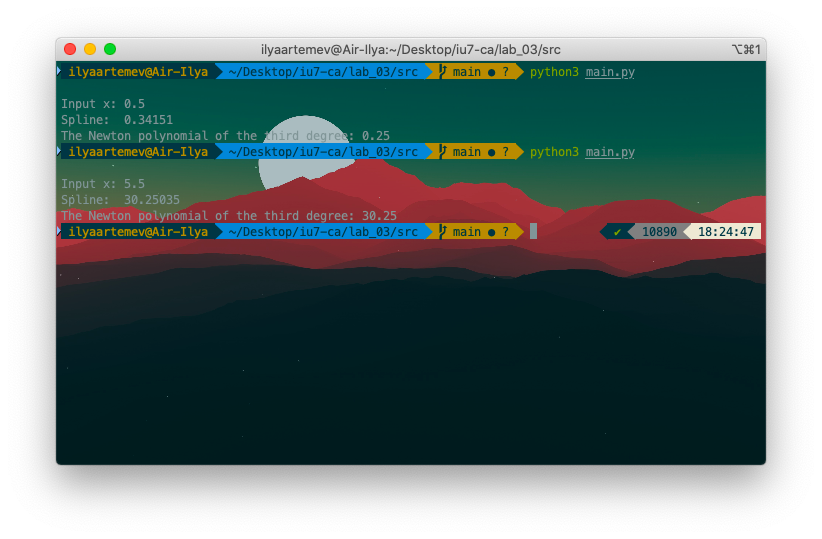
return

table.append(x\_arr)

table.append(y\_arr)

file.close()

return table

**Пример работы программы**  


**Результаты**

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **y** | **Кубический сплайн** | **Полином Ньютона 3-ей степени** |
| 0.5 | 0.25 | 0.34151 | 0.25 |
| 5.5 | 30.25 | 30.25035 | 30.25 |

**Вопросы при защите лабораторной работы**

**1. Получить выражения для коэффициентов кубического сплайна, построенного на двух точках**

Пусть заданы две точки: (x1, y1), (x2, y2)

Тогда для кубического сплайна, который имеет вид:  


Коэффициент ***A*** = y1

Поскольку ***C***(N) = 0 и сплайн строится всего по двум точкам, получаем, что ***C***(N) = ***C***, значит ***C*** = 0

Также через коэффициент ***C*** находим:

коэффициент ***D*** = - ***C***(N) / 3h(N) = 0

коэффициент ***B*** = = (y2 - y1) / (x2 - x1)

Ответ: ***A*** = y1, ***B*** = (y2 - y1) / (x2 - x1), ***C*** = 0, ***D*** = 0

**2. Выписать все условия для определения коэффициентов сплайна, построенного на 3-х точках.**

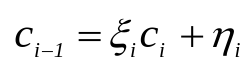
Пусть дано три точки : (x1, y1), (x2, y2), (x3, y3)

Будет 8 коэффициентов, которые будут находиться следующим образом:

1. Через значения узлов: w1(x1) = y1, w2(x2) = y2, w1(x2) = y2, w2(x3) = y3, то есть:
   * a1 = y1
   * a2 = y2
   * a1 + b1\*(x2 - x1) + c1\*(x2 - x1)^2 + d1\*(x2 - x1)^3 = y2
   * a2 + b2\*(x3 - x2) + c2\*(x3 - x2)^2 + d2\*(x3 - x2)^3 = y3
2. Через производные:

* w1’(x2) = w2’(x2) => b1+ 2\*c1\*(x2 - x1) + 3\*d1\*(x2 - x1)^2 = b2
* w1’’(x2) = w2’’(x2) => c1 + 3\*d1\*(x2 - x1) = c2
* w1’’(x1) = 0
* w2’’(x3) = 0

**3. Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо C1=C2.**



Тогда, если C1 = C2, то: = 1, = 0

**4. Написать формулу для определения последнего коэффициента сплайна СN , чтобы можно было выполнить обратный ход метода прогонки, если в качестве граничного условия задано kCN-1+mCN=p, где k,m и p - заданные числа.**

С(N - 1) = e(N) \* С(N) + n(N)

Тогда, если k\*C(N-1) + m\*C(N) = p, имеем:

*C(N) = (p - k\*n(N)) / (k\*e(N) + m)*