## Gerb-BMSTU_01Министерство 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** Климов И.С.

**Группа** ИУ7-42Б

## Оценка (баллы)

**Преподаватель** Градов В.М.

Москва. 2021 г

**Цель работы**: Получение навыков владения методами интерполяции таблично заданных функций с помощью кубических сплайнов.

1. **Исходные данные**

1. Таблица функции с количеством узлов N. Задать с помощью формулы y = x2 в диапазоне [0..10] с шагом 1.

2. Значение аргумента x в первом интервале, например, при х=0.5 и в середине таблицы, например, при x= 5.5. Сравнить с точным значением.

1. **Код программы**

**Листинг 1. spline.py**

**def** **create\_spline**(x\_table, y\_table):

n = len(x\_table)

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

splines[**0**]['c'], splines[-**1**]['c'] = **0.0**, **0.0**

alpha = [**0.0** **for** \_ **in** range(n - **1**)]

beta = [**0.0** **for** \_ **in** range(n - **1**)]

**for** i **in** range(n - **2**):

h1 = x\_table[i + **1**] - x\_table[i]

h2 = x\_table[i + **2**] - x\_table[i + **1**]

a = h1

b = h2

c = **2.0** \* (h1 + h2)

f = (y\_table[i + **2**] - y\_table[i + **1**]) / h2 - (y\_table[i + **1**] - y\_table[i]) / h1

f \*= **6.0**

t = a \* alpha[i] + c

alpha[i + **1**] = -b / t

beta[i + **1**] = (f - a \* beta[i]) / t

**for** i **in** range(n - **2**, **0**, -**1**):

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

**for** i **in** range(n - **1**, **0**, -**1**):

h = x\_table[i] - x\_table[i - **1**]

splines[i]['d'] = (splines[i]['c'] - splines[i - **1**]['c']) / h

splines[i]['b'] = h \* (**2.0** \* splines[i]['c'] + splines[i - **1**]['c']) / **6.0**

splines[i]['b'] += (y\_table[i] - y\_table[i - **1**]) / h

**return** splines

**def** **interpolation\_spline**(splines, x):

**if** **not** splines:

**raise** **ValueError**('First param is empty')

n = len(splines)

**if** x <= splines[**0**]['x']:

spline = splines[**0**]

**elif** x >= splines[-**1**]['x']:

spline = splines[-**1**]

**else**:

left, right = **0**, n - **1**

**while** left + **1** < right:

m = left + (right - left) // **2**

**if** x <= splines[m]['x']:

right = m

**else**:

left = m

spline = splines[right]

dx = x - spline['x']

result = spline['a'] + (spline['b'] + (spline['c'] / **2.0** + spline['d'] \* dx / **6.0**) \* dx) \* dx

**return** result

**Листинг 2. newton.py**

**def** **get\_diff**(y, \*args):

**if** len(args) == **0**:

**return** None

**elif** len(args) == **1**:

**return** y[args[**0**]]

**else**:

**return** (get\_diff(y, \*args[:-**1**]) - get\_diff(y, \*args[**1**:])) / (args[**0**] - args[-**1**])

**def** **get\_xi**(nx, x):

**if** nx < **0** **or** nx > **4**:

**raise** **ValueError**('nx can**\'**t be more than maximum amount and less then zero')

first\_element = min(range(**5**), key=**lambda** value: abs(x - value)) # поиск ближайшего значения к x

xi\_array = [first\_element + step **for** step **in** range(**0**, nx // **2** + **2**) **if** first\_element + step <= **4**]

xi\_array.extend([first\_element + step **for** step **in** range(-(nx - len(xi\_array) + **1**), **0**) **if** first\_element + step >= **0**])

xi\_array.extend([first\_element + step + (nx // **2** + **2**) **for** step **in** range(nx - len(xi\_array) + **1**)])

xi\_array.sort()

**return** xi\_array

**def** **get\_polynomial**(xi, y):

polynomial = []

**for** i **in** range(len(xi)):

coefficients = xi[:i]

polynomial.append(coefficients)

diff = get\_diff(y, \*xi[:(i + **1**)])

polynomial.append(diff)

**return** polynomial

**def** **take\_x**(brackets, x):

**if** **not** brackets:

**return** **1**

result = **1**

**for** bracket **in** brackets:

result \*= (x - bracket)

**return** result

**def** **interpolation\_newton**(polynomial, x):

result = **0**

**for** i **in** range(**0**, len(polynomial), **2**):

result += take\_x(polynomial[i], x) \* polynomial[i + **1**]

**return** result

**Листинг 3. main.py**

**from** **newton** **import** get\_xi, get\_polynomial, interpolation\_newton

**from** **spline** **import** create\_spline, interpolation\_spline

**def** **f**(x):

**return** x \*\* **2**

**def** **get\_table**(n):

x, y = [], []

**for** x\_cur **in** range(n):

x.append(x\_cur)

y.append(f(x\_cur))

**return** x, y

**def** **print\_result**(y\_real, y\_spline, y\_newton, diff\_spline, diff\_newton):

**print**('-' \* **65**)

**print**(f'Значение y(x) = {y\_real:.6f}')

**print**(f'Результат интерполяции сплайном = {y\_spline:.6f}')

**print**(f'Относительная погрешность = {diff\_spline \* 100:.6f}%')

**print**(f'Результат интерполяции полиномом Ньютона 3й степени = {y\_newton:.6f}')

**print**(f'Относительная погрещность = {diff\_newton \* 100:.6f}%')

**print**('-' \* **65**)

**def** **main**():

**try**:

x = float(input('Введите x: '))

**except** **ValueError**:

**return** **print**('Вы должны ввести число')

x\_table, y\_table = get\_table(**11**)

y\_real = f(x)

y\_spline = interpolation\_spline(create\_spline(x\_table, y\_table), x)

y\_newton = interpolation\_newton(get\_polynomial(get\_xi(**3**, x), y\_table), x)

diff\_spline = abs(y\_real - y\_spline) / abs(y\_real)

diff\_newton = abs(y\_real - y\_newton) / abs(y\_real)

print\_result(y\_real, y\_spline, y\_newton, diff\_spline, diff\_newton)

**if** \_\_name\_\_ == '\_\_main\_\_':

main()

1. **Результаты работы**

1. Значение y(x).

2. Сравнить результаты интерполяции кубическим сплайном и полиномом Ньютона 3-ей степени.

Введите x: 0.5

-----------------------------------------------------------------

Значение y(x) = 0.250000

Результат интерполяции сплайном = 0.341506

Относительная погрешность = 36.602210%

Результат интерполяции полиномом Ньютона 3й степени = 0.250000

Относительная погрешность = 0.000000%

-----------------------------------------------------------------

Введите x: 5.5

-----------------------------------------------------------------

Значение y(x) = 30.250000

Результат интерполяции сплайном = 30.250345

Относительная погрешность = 0.001142%

Результат интерполяции полиномом Ньютона 3й степени = 30.250000

Относительная погрешность = 0.000000%

-----------------------------------------------------------------

1. **Вопросы при защите лабораторной работы**
2. **Получить выражения для коэффициентов кубического сплайна, построенного на двух точках.**

Для двух узлов имеет место быть 4 условия:

=>

*=>*

То есть полином вырождается в прямую

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

Для трех узлов имеет место 8 условий:

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

При C1 = C2:

,

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