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 (ИУ-7)»

Лабораторная работа №3

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

Выполнил: **Варин Д.В.**

Группа: **ИУ7-46Б**

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. Таблица функции с количеством узлов N.

Задать с помощью формулы $y = x^2$ в диапазоне [0..10] с шагом 1.2.

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

Результат:

1. Значение y(x)

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

Описание алгоритма:

1. Определение диапазона для проведения интерполяции.

2. Прямой ход (вычисление прогоночных коэффициентов).

3. Обратный ход (решение системы для определения C_i с помощью вычисленных прогоночных коэффициентов.

4. Вычисление остальных трех коэффициентов коэффициентов, используя выведенные связи с вычисленным на прошлом этапе коэффициентом.

5. Вычисление значения функции.

Входные данные:

Функция вида $y = x^2$

x	y
0.00	0.00
1.00	1.00
2.00	4.00
3.00	9.00
4.00	16.00
5.00	25.00
6.00	36.00
7.00	49.00
8.00	64.00
9.00	81.00

Результаты работы:

x	Кубический сплайн	Полином Ньютона
0.5	0.3419	0.25
5.5	30.2479	30.25

Листинг кода.

Программа написана на языке Python 3.

main.py — точка входа

```python
from spline import Spline
from polynom import NewtonPolynom
from utils import *

from sys import argv


def main() -> None:
    f = argv[1]
    points = read_points(f)
    print("Table of points\n")
    print_points(points)

    print('\nEntry X for interpolate:\n')
    x = float(input())

    spline_res = Spline(points).solve(x)
    newton_res = NewtonPolynom(points).solve(x)

    print_res(spline_res, newton_res)


if __name__ == "__main__":
    main()
```

utils.py

```python
from __future__ import annotations

class Point:
    x: float
    y: float

    def __init__(self, _x: float, _y: float) -> None:
        self.x = _x
        self.y = _y

    def __lt__(self, other: Point):
        if self.x == other.x:
            return self.y < other.y
        return self.x < other.x

    def __str__(self):
        return "{:^8.3f} {:^8.3f}".format(self.x, self.y)


def read_points(fname: str) -> list[Point]:
    points = []

    with open(fname, 'r') as f:
        for line in f.readlines():
            point = Point(*list(map(float, line.split()[:2])))
            points.append(point)

    return points
def print_points(points: list[Point]) -> None:
    print("{:^8} {:^8}".format('X', 'Y'))
    for p in points:
        print(p)

def print_res(spline: Point, newton: Point) -> None:
    print(f"Spline result is {spline}")
    print(f"Newton result is {newton}")
```

spline.py

```python
from __future__ import annotations
from utils import Point


class Spline:
    points: list[Point]

    def __init__(self, _points: list[Point]):
        self.points = _points

    def get_pos(self, p: Point):
        i = 1
        while i < len(self.points) and self.points[i].x < p.x:
            i += 1

        return i - 1
```

```python
def solve(self, x: float) -> Point:
    list_x = [p.x for p in self.points]
    list_y = [p.y for p in self.points]

    a = list_y[:-1]

    c = [0 for i in range(len(self.points) - 1)]

    # Нач. данные
    ksi_k, tet_k = [0, 0], [0, 0]

    # Сначала делаем прямой проход и вычисляем прогоночные коэффициенты
    for i in range(2, len(self.points)):
        xhi_1, xhi_2 = list_x[i] - list_x[i - 1], list_x[i - 1] - list_x[i - 2]
        yhi_1, yhi_2 = list_y[i] - list_y[i - 1], list_y[i - 1] - list_y[i - 2]

        phi = 3 * (yhi_1 / xhi_1 - yhi_2 / xhi_2)

        ksi_k.append(-xhi_1 / (xhi_2 * ksi_k[i - 1] + 2 * (xhi_2 + xhi_1)))
        tet_k.append((phi - xhi_2 * tet_k[i - 1]) / (xhi_2 * ksi_k[i - 1] + 2 * (xhi_2 + xhi_1)))

    c[len(self.points) - 2] = tet_k[-1]

    # Обратный проход
    for i in range(len(self.points) - 2, 0, -1):
        c[i - 1] = ksi_k[i] * c[i] + tet_k[i]
```

```python
b, d = [], []
for i in range(1, len(self.points) - 1):
    xhi_1 = list_x[i] - list_x[i - 1]
    yhi_1 = list_y[i] - list_y[i - 1]
    b.append(yhi_1 / xhi_1 - (xhi_1 * (c[i] + 2 * c[i - 1])) / 3)
    d.append((c[i] - c[i - 1]) / (3 * xhi_1))

b.append((list_y[-1] - list_y[-2]) / (list_x[-1] - list_x[-2]) - ((list_x[-1] - list_x[-2]) * 2 * c[-1]) / 3)
d.append(-c[len(self.points) - 2] / (3 * list_x[-1] - list_x[-2]))
pos = self.get_pos(Point(x, 0))

res = a[pos] + \
      b[pos] * (x - self.points[pos].x) + \
      c[pos] * (x - self.points[pos].x) ** 2 + \
      d[pos] * (x - self.points[pos].x) ** 3


return Point(x, res)
```

polynom.py

```python
from __future__ import annotations
import numpy as np
from utils import Point

class NewtonPolynom:
    x_list: list[float]
    y_list: list[float]

    def __init__(self, points: list[Point]):
        self.x_list = [p.x for p in points]
        self.y_list = [p.y for p in points]

    def solve(self, x: float) -> Point:
        a = self.make_diff_table()
        n = len(self.x_list) - 1  # Degree of polynomial
        p = a[n]

        for k in range(1, n + 1):
            p = a[n - k] + (x - self.x_list[n - k]) * p

        return Point(x, p)

    def make_diff_table(self) -> np.ndarray[float]:
        m = len(self.x_list)
        x = np.copy(self.x_list)
        a = np.copy(self.y_list)
        for k in range(1, m):
            a[k:m] = (a[k:m] - a[k - 1]) / (x[k:m] - x[k - 1])

        return a
```

Вопросы при защите лабораторной работы.

Ответы на вопросы дать письменно в Отчете о лабораторной работе.

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

Пусть $(i-1)$ - номер первой точки, i - номер второй точки.

$1 \le i \le N$, где $1 \le i \le N$ - количество узлов.

Полином:

$$F(x) = A_i + B_i \cdot (x - x_{i-1}) + C_i \cdot (x - x_{i-1})^2 + D_i \cdot (x - x_{i-1})^3 (1)$$

Найдем коэффициенты A_i, B_i, C_i, D_i

$$f(x_{i-1}) = y_{i-1} = A_i (2)$$

Введем обозначение $h_i = x_i - x_{i-1}$

$$f(x_i) = y_i = A_i + B_i \cdot h_i + C_i \cdot h_i^2 + D_i \cdot h_i^3 (3)$$

Число таких уравнений меньше числа неизвестных в два раза.

Недостающие уравнения можно получить, приравняв во внутренних узлах первые и вторые производные, вычисляемые по коэффициентам на соседних участках:

$$f'(x_i) = y_i = B_i + 2 \cdot C_i \cdot (x_i - x_{i-1}) + 3 \cdot D_i \cdot (x_i - x_{i-1})^2 \quad (4)$$

$$x_{i-1} \le x \le x_i$$

$$f''(x_i) = 2 \cdot C_i + 6 \cdot D_i \cdot (x_i - x_{i-1}) \quad (5)$$

$$B_{i+1} = B_i + 2 \cdot C_i \cdot h_i + 3 \cdot D_i \cdot h_i^2 \quad (6)$$

$$C_{i+1} = C_i + 3 \cdot D_i \cdot h_i \quad (7)$$

Также предполагают следующие условия:

$$f''(x_0) = 0, C_1 = 0 \quad (8)$$

$$f''(x_n) = 0, C_{n+1} = C_n + 3 \cdot D_n \cdot h_n = 0 \quad (9)$$

Из (1) находятся A_i .

Из (7) и (9) следует:

$$D_i = (C_{i+1} - C_i) / (3 \cdot h_i)$$

$$D_n = -C_n / 3 \cdot h_n$$

Исключим из (3) D_i с помощью полученного выше выражения, получим:

$$B = (y_i - y_{i-1}) / h_i - h_i \cdot (C_{i+1} - 2 \cdot C_i) / 3, 1 \le i < N$$

Из (3), используя выражение для D N получим:

$$B_n = (y_n - y_{n-1}) / h_n - h_n \cdot 2 \cdot C_n / 3$$

Теперь исключим из (6) величины B_i и B_{i+1} с учетом полученных выше выражений, а также величину D_i . В результате получим систему уравнений для определения C_i :

$$C_1 = 0$$

$$h_{i-1} \cdot C_{i-1} + 2 \cdot (h_{i-1} + h_i) \cdot C_i + h_i \cdot C_i + 1 = 3 \cdot ((y_i - y_{i-1}) / h_i - (y_{i-1} - y_{i-2}) / h_i)$$

$$C_{n+1} = 0$$

После нахождения коэффициентов C_i находятся остальные коэффициенты по вычисленным ранее формулам, выраженным через C_i .

Заметим, что полученная система - СЛАУ с трехдиагональной матрицей.

Она решается методом прогонки.

В каждом уравнении содержится только два неизвестных, и при обратном ходе одно из этих неизвестных выражается через другое.

$$C_i = e_{i+1} \cdot C_{i+1} + n_{i+1}, \text{где } e_{i+1} \text{ и } n_{i+1} - \text{пока неизвестные коэффициенты}.$$

Можно записать так: $C_{i-1} = e_i \cdot C_i + n_i$

Подставляя полученное выражение во второе уравнение системы и

преобразуя, получим:

$$C_i = -h_i/(h_{i-1}\cdot e_i + 2\cdot(h_{i-1}+h\,i))\cdot C_{i+1} + (f_i - h_{i-1}\cdot n_i)/(h_{i-1}\cdot e_i + 2\cdot(h_{i-1}+h_i))$$

Где $f = 3*((y_i - y\,i-1)/h_i - (y_{i-1} - y\,i-2)/h_i)$

Сравнивая это выражение с выражением для C_i, получаем:

$$e_{i+1} = -h_i/(h_{i-1}\cdot e_i + 2\cdot(h_{i-1}+h_i))$$

$$n_{i+1} = (f_i - h_{i-1}\cdot n_i)/(h_{i-1}\cdot e_i + 2\cdot(h_{i-1}+h_i))$$

Из условия $C_1 = 0$ следует, что $e_2 = 0$ *и* $n_2 = 0$.Теперь алгоритм решения системы выглядит следующим образом.

По формулам для e_{i+1} *и* n_{i+1} при известных e_i *и* n_i вычисляют прогоночные коэффициенты (прямой ход). Затем по формуле $C_i = e_{i+1}\cdot C_{i+1} + n_{i+1}$ и при условии $C_{n+1} = 0$ определяют все C_i (обратный ход).

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

1. Необходимо определить 8N коэффициентов, следовательно, необходимо определить 8 условий:

2. Функция должна пройти через точки, ограничивающие исследуемый отрезок - 4 условия.

3. Равенство производных первой и второй степени во внутренних узлах - 2 условия.

4. Равенство нулю производных функции на концах отрезка (вводимое допущение) - 2 условия.

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

$$C_1 = e_2\cdot C_2 + n_2$$

$$C_1 = C_1 \Rightarrow C_1 = e_2\cdot C_1 + n_2$$

$$e_2\cdot C_1 - C_1 + n_2 = 0$$

$$C_1\cdot(e_2 - 1) + n_2 = 0$$

C_1 *предполагается равным* 0 *по алгоритму решения*:

$$0 + n_2 = 0$$

$$n_2 = 0$$

e_2 *может быть любым*

Если же $C_1 != 0$, *то необходимо выбрать* $e_2 = 1, n_2 = 0.$

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

$C_{N-1}=e_N\cdot C_N+n_N$
$C_{N-1}=(-m/k)\cdot C_n+(p/k)$
$e_N=(-m/k)$
$n_N=p/k$
$C_N=e_{N+1}\cdot C_{N+1}+n_{N+1}$
$C_{N+1}=0\Rightarrow C_n=n_{N+1}$
$n_{N+1}=(f_N-n_N)/(e_N+4)\,(примем\ h_i=1\ для\ удобства),\ где\ f_N=3\cdot((y_i-y_{i-1})/h_i-(y_{i-1}-y\,i-2)/h_i-1)$
$n_{N+1}=(f_N-p/k)/((-m/k)+4)$