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.

«Интерполяция сплайнами»

Студент: **Ивахненко Д. А**

Группа: **ИУ7-46Б**

Оценка (баллы): _____

Преподаватель: **Градов В.М.**

Москва

2021

Цель работы

Построение и программная реализация алгоритма сплайн-интерполяции табличных функций

Исходные данные

Таблица функции с количеством узлов N. Задать с помощью формулы $y = x^2$ в диапазоне [0, 10] с шагом 1.

X	Y
0	0
1	1
…	…
9	81
10	100

Описание алгоритма

Для решения поставленной задачи используется кубический сплайн – кривая, состоящая из множества полиномов третьей степени. Для определения коэффициентов полинома необходимо решить некоторую систему алгебраических уравнений, составные уравнения которой строятся по следующим соображениям:

1) В узлах многочлена и функции должны совпадать.
2) Во внутренних узлах значения первой и второй производной должны совпадать для обеспечения «гладкого» вида кривой.
3) Во внешних узлах полагаем, что вторая производная имеет нулевое значение. Т.к. матрица данной системы трехдиагональна, то такую систему удобно решить методом прогонки.

Исходный код программы

main.py

```python
from settings import PATH_TO_TABLE
from polynomes.utils import read_table
from polynomes.splines import spline_interpolation


def main():
    x = float(input('Enter x: '))
    y = spline_interpolation(x, read_table(PATH_TO_TABLE))

    print(f'f({x}) = {y:.3f}')


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

splines.py

```python
def _get_a_coef(table: np.ndarray, index: int) -> float:
    return table[index-1, 1]


def _get_b_coef(table: np.ndarray, cl: float, cr: float, index:
int, hi: float) -> float:
    ydelta = lambda i: table[i, 1] - table[i-1, 1]
    return (ydelta(index)) / hi - hi * (cr + 2 * cl) / 3


def _get_d_coef(cl: float, cr: float, hi: float) -> float:
    return (cr - cl) / (3 * hi)


def _get_coeffs(table: np.ndarray, index: int, hi: float) ->
Tuple[float, float, float, float, float]:
    cl, cr = _get_c_coef(table, index)
    a = _get_a_coef(table, index)
    b = _get_b_coef(table, cl, cr, index, hi)
    d = _get_d_coef(cl, cr, hi)
    return a, b, cl, d


def spline_interpolation(x: float, table: np.ndarray) -> float:
    index = bisect(table[:, 0], x)
    a,b,c,d = _get_coeffs(table, index, _get_h(table, index))
    dx = x - table[index-1, 0]

    return a + b*dx + c*dx*dx + d*dx*dx*dx
```

```python
def _get_h(table: np.ndarray, index: int) -> float:
    return table[index, 0] - table[index-1, 0]


def _get_c_coef(table: np.ndarray, index: int) -> float:
    def _get_f(i: int) -> float:
        ydelta = lambda i: table[i, 1] - table[i-1, 1]
        return 3 * (ydelta(i) / _get_h(table, i) - ydelta(i-1) /
 _get_h(table, i-1))

    def _get_eta(eta_prev: float, xi_prev: float, i: int) ->
float:
        hr, hl = _get_h(table, i), _get_h(table, i-1)
        return (_get_f(i) - hl * eta_prev) / (hl * xi_prev + 2 *
 (hl + hr))

    def _get_xi(xi_prev: float, i: int) -> float:
        hr, hl = _get_h(table, i), _get_h(table, i-1)
        return -hr / (hl * xi_prev + 2 * (hl + hr))

    N: int = len(table)-1
    xi, eta = np.zeros(N+2), np.zeros(N+2)

    for i in range(2, N+1):
        xi[i+1] = _get_xi(xi[i], i)
        eta[i+1] = _get_eta(eta[i], xi[i], i)

    c: float = 0.0
    for i in range(N, index+1, -1):
        c = xi[i+1] * c + eta[i+1]

    cr = xi[index+2] * c + eta[index+2]
    cl = xi[index+1] * cr + eta[index+1]

    return cl, cr
```

Результаты работы

x	y (сплайны)	y (Ньютон 3й степени)	y (истинное)
0.5	0.342	0.25	0.25
5.5	30.25	30.25	30.25

В данном случае полином Ньютона обеспечивает лучшее приближение, однако из-за того, что коэффициенты полинома при больших степенях вычисляются с большой погрешностью, точность расчета может значительно ухудшиться. Следовательно, может быть лучше использовать интерполяцию кубическими сплайнами.

Ответы на контрольные вопросы

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

Пусть задано два узла со значениями в них $(x0, y0)$ и $(x1, y1)$ соответственно

$$\psi(x) = a_1 + b_1(x - x_0) + c_1(x - x_0)^2 + d_1(x - x_0)^3$$

$$a_1 = y_0$$

$$c_1 = \xi_2 c_2 + \eta_2 = 0$$

$$b_1 = \frac{(y_1 - y_0)}{(x_1 - x_0)} - (x_1 - x_0)\frac{2\,c_1}{3} = \frac{(y_1 - y_0)}{(x_1 - x_0)}$$

$$d_1 = -\frac{c_1}{3(x_1 - x_0)} = 0$$

$$\psi(x) = y_0 + \frac{(y_1 - y_0)}{(x_1 - x_0)}(x - x_0)$$

2. Выписать все условия для определения коэффициентов сплайна, построенного на трех точках.

Пусть задано 3 узла со значениями в них $(x0, y0)$, $(x1, y1)$ и $(x2, y2)$ соответственно.

$$\begin{cases} \psi_1(x_0) = a_1 = y_0 \\ \psi_2(x_1) = a_2 = y_1 \\ \psi_3(x_2) = a_3 = y_2 \\ \psi_1(x_1) = a_1 + b_1(x_1 - x_0) + c_1(x_1 - x_0)^2 + d_1(x_1 - x_0)^3 = y_1 \\ \psi_2(x_2) = a_2 + b_2(x_2 - x_1) + c_2(x_2 - x_1)^2 + d_2(x_2 - x_1)^3 = y_2 \\ \psi_3(x_3) = a_3 + b_3(x_3 - x_2) + c_3(x_3 - x_2)^2 + d_3(x_3 - x_2)^3 = y_3 \\ \psi_1'(x_1) = \psi'_2(x_1) \\ \psi''_1(x_1) = \psi''_2(x_1) \\ \psi_1'(x_0) = 0 \\ \psi''_1(x_0) = 0 \\ \psi_3'(x_2) = 0 \\ \psi''_3(x_2) = 0 \end{cases}$$

3. Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо $C_1 = C_2$. Какие имеются ограничения на расположение узлов при разных степенях полинома?

$$C_1 = 0 = \xi_2 C_2 + \eta_2, \xi_2 = 0, \eta_2 = 0, C_2 = 0$$

4. Написать формулу для определения последнего коэффициента сплайна C_N, чтобы можно было выполнить обратный ход метода прогонки, если в качестве граничного условия задано $kC_{N-1} + mC_N = p$, где k, m, p – заданные числа.

$$\begin{cases} C_{N-1} = \xi_N C_N + \eta_N \\ C_{N-1} = \dfrac{p - mC_N}{k} \end{cases}$$

$$C_N = \frac{p - \eta_N k}{\xi_N + m}$$