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 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 $y = x^2$ в диапазоне [0..10] с шагом 1.

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.

2. 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 погрешность = {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()
```

3. 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%
----------------------------------------------------------------
```

4. Вопросы при защите лабораторной работы

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

Для двух узлов имеет место быть 4 условия:

$\psi''(x_1) = c_1 = 0$

$\psi''(x_0) = c_1 - d_1 h_1 = 0 \Rightarrow d_1 = 0$

$\psi(x_0) = a_1 - b_1 h_1$

$\psi(x_1) = a_1 \Rightarrow b_1 = \frac{\psi(x_1) - \psi(x_0)}{x_1 - x_0}$

То есть полином вырождается в прямую

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

Для трех узлов имеет место 8 условий:

$\psi(x_0) = a_1 - b_1 h_1 + \frac{c_1}{2} h_1^2 - \frac{d_1}{6} h_1^3$

$\psi(x_1) = a_1$

$\psi(x_2) = a_2$

$a_1 = a_2 - b_2 h_2 + \frac{c_2}{2} h_2^2 + \frac{d_2}{6} h_2^3$

$$b_1 = b_2 - c_2 h_2 + \frac{d_2}{2} h_2^2$$

$$c_1 = c_2 - d_2 h_2$$

$$c_2 = 0$$

$$c_1 - d_1 b_1 = 0$$

3) **Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо $C_1 = C_2$.**

$$c_{i-1} = \xi_i c_i + \eta_i$$

При $C_1 = C_2$:

$$c_1 = \xi c_1 + \eta$$

$$c_1(1 - \xi) = \eta$$

$$\eta = 0, \xi = 1$$

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

$$c_{i-1} = \xi_i c_i + \eta_i$$

$$k c_{N-1} + m c_N = p$$

$$c_{N-1} = \frac{p - m c_N}{k}$$

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

$$k \xi_N c_N = p - m c_N - k \eta_N$$

$$c_N (k \xi_N + m) = p - k \eta_N$$

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