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 Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине: «Вычислительная математика»

Студент	Исраелян Ева Арамовна
Группа	РК6-54Б
Тип задания	лабораторная работа
Тема лабораторной работы	Интерполяция в условиях измерений с неопределённостью

Студент ____________________ **Исраелян Е.А.**
подпись, дата фамилия, и.о.

Преподаватель ____________________ **Першин А.Ю.**
подпись, дата фамилия, и.о.

Москва, 2021 г.

Оглавление

Задание на лабораторную работу

Задача 5 (интерполяция кубическими сплайнами)

Требуется:

1. Разработать функцию qubic_spline_coeff(x_nodes, y_nodes), которая посредством решения матричного уравнения вычисляет коэффициенты естественного кубического сплайна. Для простоты решение матричного уравнения можно производить с помощью вычисления обратной матрицы с использованием функции numpy.linalg.inv()
2. Написать функции qubic_spline(x, qs_coeff) и d_qubic_spline(x, qs_coeff), которые вычисляют соответственно значение кубического сплайна и его производной в точке x (qs_coeff обозначает матрицу коэффициентов).
3. Используя данные в таблице 1, требуется построить аппроксимацию зависимости уровня поверхности жидкости $h(x)$ от координаты x с помощью кубического сплайна и продемонстрировать её на графике вместе с исходными узлами.

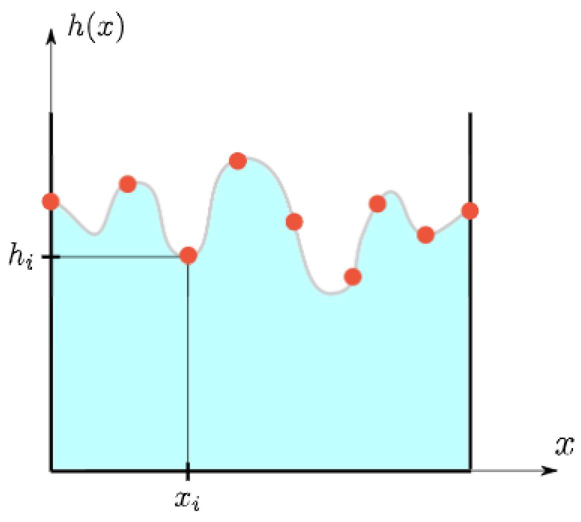

Рисунок 1 – Поверхность вязкой жидкости (серая кривая), движущейся сквозь некоторую среду (например, пористую). Её значения известны только в нескольких точках (красные узлы).

Таблица 1: Значения уровня поверхности вязкой жидкости (рисунок 1)

i	1	2	3	4	5	6	7	8	9	10	11
x_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
h_i	3.37	3.95	3.73	3.59	3.15	3.15	3.05	3.86	3.60	3.70	3.02

Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – на простейшем примере познакомиться с интерполяцией кубическими сплайнами.

Выполненные задачи

1. Разработаны функции для вычисления векторов коэффициентов кубического сплайна, которые будут использоваться непосредственно в функции qubic_spline_coeff(x_nodes, y_nodes), возвращающей матрицу коэффициентов размерности $(\mathrm{N}-1) \times 3$.
2. Написана функция, которая вычисляет промежуток, которому принадлежит точка x, после чего разработаны функции qubic_spline(x, qs_coeff, x_nodes, y_nodes) и d_qubic_spline(x, qs_coeff, x_nodes, y_nodes) для вычисления соответственно значения кубического сплайна и его производной в точке x.
3. Построена аппроксимация зависимости уровня поверхности жидкости $h(x)$ от координаты x с помощью кубического сплайна и получен её график вместе с исходными узлами.

1. Вычисление коэффициентов кубического сплайна.

Естественный кубический сплайн – это кусочно-заданная функция $S(x)$, состоящая из n-1 кубических многочленов $S_i(x)$, которая используется для интерполяции неизвестной нам функции $f(x)$, при условии, что нам известны значения этой функции в n узлах. При этом в узлах должны выполняться следующие условия:

1) $S_i(x_i) = f(x_i)$, $S_i(x_{i+1}) = f(x_{i+1})$, $i = 1, \dots, n-1$
2) Сопряжение значений смежных многочленов:
   $S_i(x_{i+1}) = S_{i+1}(x_{i+1}), \qquad i = 1, \dots, n-2$
3) Сопряжение первых и вторых производных:
   $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \qquad S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}),\ \ i = 1, \dots, n-2$
4) Граничные условия:
   $S''(x_1) = S''(x_n) = 0$
   $S'(x_1) = f'(x_1)$, $S'(x_n) = f'(x_n)$

Кубический многочлен, коэффициенты которого нам предстоит посчитать, выглядит следующим образом:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{1}$$

Из (1) очевидно, что кубический многочлен $S_i(x)$ равен a_i при $x = x_i$, то есть вектор коэффициентов a совпадает с вектором значений функции в узлах. Нам остаётся найти только коэффициенты b, c и d.

Для упрощения записи введём следующее обозначение: $h_i = x_{i+1} - x_i$. Вектор h будет генерироваться функцией create_h_nodes(x_nodes) (см. листинг 1.1).

Листинг 1.1 (код функции create_h_nodes(x_nodes))

```
def create_h_nodes(x_nodes):
    return x_nodes[1:] - x_nodes[:-1]
```

Разрешающее уравнение относительно коэффициента c, которое выглядит следующим образом:

$$h_{i-1}c_{i-1} + 2(h_i + h_{i-1})c_i + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}) \tag{2}$$

можно представить в матричном виде $A \cdot c = B$:

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ h_1 & 2(h_2 + h_1) & h_2 & 0 & \cdots & 0 \\ 0 & h_2 & 2(h_3 + h_2) & h_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} =$$

$$= \begin{bmatrix} 0 \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \frac{3}{h_3}(a_4 - a_3) - \frac{3}{h_2}(a_3 - a_2) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix}$$

Генерация матриц A и B были происходит в отдельных функциях create_matrix_A(h_nodes) и create_matrix_B(h_nodes, a_nodes) (см. листинг 1.2 и 1.3).

Листинг 1.2 (код функции create_matrix_A(h_nodes)))

```
def create_matrix_A(h_nodes):
    left_diag = []
    main_diag = [1.]
    right_diag = [0.]
    for i in range(2, len(h_nodes) + 1):
        left_diag.append(h_nodes[i-2])
        main_diag.append(2 * (h_nodes[i-1] + h_nodes[i-2]))
        right_diag.append(h_nodes[i-1])
    left_diag.append(0.)
    main_diag.append(1.)
    return np.diag(main_diag) + np.diag(left_diag, -1) +
np.diag(right_diag, 1)
```

Листинг 1.3 (код функции create_matrix_B(h_nodes, a_nodes))

```
def create_matrix_B(h_nodes, a_nodes):
    matrix_B = [0.]
    for i in range(2, len(h_nodes) + 1):
        matrix_el = 3 / h_nodes[i-1] * (a_nodes[i] - a_nodes[i-1]) - \
                    3 / h_nodes[i-2] * (a_nodes[i-1] - a_nodes[i-2])
        matrix_B.append(matrix_el)
    matrix_B.append(0.)
    return np.array(matrix_B)
```

Расчёт вектора коэффициентов c производится в отдельной функции create_c_nodes(h_nodes, a_nodes) путём умножения матрицы B на матрицу, обратную матрице A (см. листинг 1.4).

Листинг 1.4 (код функции create_c_nodes(h_nodes, a_nodes))

```
def create_c_nodes(h_nodes, a_nodes):
    A = create_matrix_A(h_nodes)
    B = create_matrix_B(h_nodes, a_nodes)
    A_inv = np.linalg.inv(A)    # обратная матрица A
    c_nodes = np.array(A_inv.dot(B))
    return c_nodes
```

Из курса лекций нам известны разрешающие уравнения для коэффициентов b и d:

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(c_{i+1} - 2c_i) \tag{3}$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i} \tag{4}$$

Выражения (3) и (4) используются в функциях create_b_nodes(h_nodes, a_nodes, c_nodes) и create_d_nodes(h_nodes, c_nodes) (см. листинг 1.5 и 1.6) для вычисления коэффициентов b и d соотвественно.

Листинг 1.5 (код функции create_b_nodes(h_nodes, a_nodes, c_nodes))

```
def create_b_nodes(h_nodes, a_nodes, c_nodes):
    b_nodes = []
    for i in range(len(c_nodes) - 1):
        b_nodes.append(1 / h_nodes[i] * (a_nodes[i+1] - a_nodes[i]) - \
                       h_nodes[i] / 3 * (c_nodes[i+1] + 2 * c_nodes[i]))
    return np.array(b_nodes)
```

Листинг 1.6 (код функции create_d_nodes(h_nodes, c_nodes))

```
def create_d_nodes(h_nodes, c_nodes):
    d_nodes = []
    for i in range(len(h_nodes)):
        d_nodes.append((c_nodes[i+1] - c_nodes[i]) / (3 * h_nodes[i]))
    return np.array(d_nodes)
```

После создания всех вспомогательных функций становится возможным написать функцию qubic_spline_coeff(x_nodes, y_nodes) (см. листинг 1.7), которая принимает на вход аргументы неизвестной функции и её значений в n узлах, а возвращает матрицу коэффициентов размерности $(\mathrm{N} - 1) \times 3$.

Листинг 1.7 (код функции create_ qubic_spline_coeff(x_nodes, y_nodes))

```
def qubic_spline_coeff(x_nodes, y_nodes):
    h_nodes = create_h_nodes(x_nodes)
    c_nodes = create_c_nodes(h_nodes, y_nodes)
    b_nodes = create_b_nodes(h_nodes, y_nodes, c_nodes)
    d_nodes = create_d_nodes(h_nodes, c_nodes)
    qubic_spline = []
    for i in range(len(x_nodes) - 1):
        qubic_spline.append([b_nodes[i], c_nodes[i], d_nodes[i]])
    return qubic_spline
```

2. Вычисление значения кубического сплайна и его первой производной в точке x.

Для вычисления значения кубического сплайна в заданной точке x необходимо определить промежуток, в котором находится точка x, и в соответствии с ним выбрать необходимые коэффициенты для полинома третьей степени. Для этого мной была разработана функция get_index(x, x_nodes) (см. листинг 2.1), которая принимает в качестве аргументов заданный x и абсциссы узлов, а возвращает номер промежутка, в котором находится заданный x.

Листинг 2.1 (код функции get_index(x, x_nodes))

```
def get_index(x, x_nodes):
    for i in range(len(x_nodes) - 1):
        if x_nodes[i] <= x <= x_nodes[i+1]:
            index = i
            break
    return index
```

Далее, используя этот номер, подбираются необходимые коэффициенты кубического сплайна, и с их помощью по формулам (1) и (5) вычисляются нужные нам значения в функциях qubic_spline(x, qs_coeff, x_nodes, y_nodes) и d_qubic_spline(x, qs_coeff, x_nodes, y_nodes) соответственно.

$$S'_i(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2 \quad (5)$$

Перечисленные функции возвращают координату y для сплайна и его производной (см. листинг 2.2 и 2.3).

Листинг 2.2 (код функции qubic_spline(x, qs_coeff, x_nodes, y_nodes))

```
    def qubic_spline(x, qs_coeff, x_nodes, y_nodes):
        index = get_index(x, x_nodes)
        a = y_nodes[index]
        b = qs_coeff[index][0]
        c = qs_coeff[index][1]
        d = qs_coeff[index][2]
        expression = a + b * (x - x_nodes[index]) + c * (x - x_nodes[index])
** 2\
                       + d * (x - x_nodes[index]) ** 3
        return expression
```

Листинг 2.3 (код функции d_qubic_spline(x, qs_coeff, x_nodes, y_nodes))

```
    def d_qubic_spline(x, qs_coeff, x_nodes, y_nodes):
        index = get_index(x, x_nodes)
        b = qs_coeff[index][0]
        c = qs_coeff[index][1]
        d = qs_coeff[index][2]
        expression = b + 2 * c * (x - x_nodes[index]) +3 * d * (x -
x_nodes[index]) ** 2
        return expression
```

3. Построение аппроксимации зависимости уровня поверхности жидкости от координаты x.

Для того чтобы построить аппроксимацию зависимости уровня жидкости $h(x)$ от координаты x была использована библиотека matplotlib, в том числе pyplot.

Используемые для аппроксимации значения представлены в таблице 1.

```
    x_nodes = np.arange(0, 1.01, 0.1)
    y_nodes = [3.37, 3.95, 3.73, 3.59, 3.15, 3.15, 3.05, 3.86, 3.60, 3.70,
3.02]
```

Внутри функции draw_spline(x_nodes, y_nodes) (см. листинг 3.1) для более гладкой отрисовки происходит генерация большого количества точек с абсциссами от 0 до 1 (т. к. абсциссы узлов принадлежат отрезку [0, 1]). Далее в цикле с помощью функции qubic_spline(x_nodes, y_nodes) рассчитывается ордината аппроксимирующей функции, после чего с помощью функций из matplotlib происходит отрисовка самой функции и известных нам узлов.

Листинг 3.1 (код функции draw_spline(x_nodes, y_nodes))

```
def draw_spline(x_nodes, y_nodes):
    fig, ax = plt.subplots(figsize=(16,4))
    qubic_x = np.linspace(0, 1, 200)
    qubic_y = []
    for i in range(len(qubic_x)):
        qubic_y.append(qubic_spline(qubic_x[i], \
                    qubic_spline_coeff(x_nodes, y_nodes), x_nodes,
y_nodes))
    ax.scatter(x_nodes, y_nodes, color='red', s=50, marker='o')
    ax.plot(qubic_x, qubic_y)
```

На рисунке 2 продемонстрирована требуемая в задании аппроксимация зависимости уровня поверхности жидкости $h(x)$ от координаты x вместе с исходными узлами.

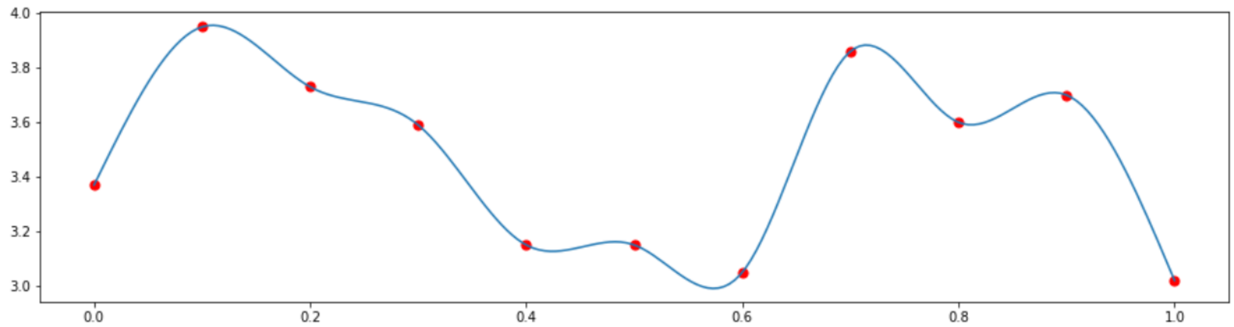

Рисунок 2 – Аппроксимация зависимости уровня поверхности жидкости $h(x)$ от координаты x с помощью кубических сплайнов.

Заключение

В ходе работы была рассмотрена интерполяция кубическими сплайнами. На языке Python была разработана функция для вычисления коэффициента c посредством решения матричного уравнения, а также функции для вычисления коэффициентов b и d по разрешающим уравнениям, выведенным нами на лекциях.

Были написаны функции, вычисляющие значение кубического сплайна и его производной в заданной точке.

С помощью кубических сплайнов была построена аппроксимация зависимости уровня поверхности жидкости $h(x)$ от координаты x.

Список использованных источников

1. **Першин А.Ю.** Лекции по вычислительной математике. [Электронный ресурс] // МГТУ им. Н. Э. Баумана, 2020. – 142с.;
2. **Першин А.Ю., Соколов А.П.** Вычислительная математика, лабораторные работы (учебное пособие) [Электронный ресурс] // МГТУ им. Баумана, Москва, 2021, 40 с.