Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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 «Теоретическая информатика и компьютерные технологии»

# Лабораторная работа № 1

## по курсу «Численные методы»

Студент группы ИУ9-62Б Нащекин Н.Д.

Преподаватель: Домрачева А.Б.

Москва, 2025

# 1 Задача

- Протабулировать функцию $f(x)$ на отрезке $[a, b]$ с шагом $h = \frac{b-a}{32}$ и распечатать таблицу $(x_i, y_i), i = 0, ...n$. Для полученных узлов $(x_i, y_i), i = 0, ...n$ построить кубический сплайн (распечатать массивы $a, b, c$ и $d$). Вычислить значения $f(x)$ в точках $x_i = a + (i - \frac{1}{2})h$, $i = 0, ...n$. Вычислить значения оригинальной функции и сплайна в произвольной точке.

## 2 Основная теория

Если задана функция $f(x)$, значения которой известны в точках $(x_i, y_i), i = 0, ...n$, то интерполяционной называется функция $y = \varphi(x)$, проходящая через эти точки (они называются узлами интерполирования). В промежуточной точке равенство $f(x) \approx \varphi(x)$ выполняется лишь с некоторой погрешностью.

Одним из способов построения интерполяционной функции является нахождение сплайнов. Сплайн $k$-го порядка с дефектом def — функция, проходящая через все узлы $(x_i, y_i), i = 0, ...n$, которая является многочленом $k$-й степени на каждом отрезке разбиения $[x_i, x_{i+1}]$ (такая функция также называется кусочно-полиномиальной) и имеет $(k - def)$ непрерывных производных на отрезке $[x_0, x_n]$.

Сплайн третьего порядка с дефектом 1 можно отыскать в виде:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3,$$
$$x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1$$

Условия на частные многочлены:

$$S_i(x_i) = y_i, \quad i = 0, \dots, n-1; \quad S_{n-1}(x_n) = y_n$$

(сплайн проходит через все узлы интерполирования). Также

$$S_{i-1}(x_i) = S_i(x_i); \quad S'_{i-1}(x_i) = S'_i(x_i);$$

$$S''_{i-1}(x_i) = S''_i(x_i), \quad i = 0, \dots, n-1$$

(непрерывность сплайна и его первых двух производных в промежуточных узлах) и

$$S''_0(x_0) = 0; \quad S''_{n-1}(x_n) = 0$$

(условия гладкости на краях).

Эти условия приводят к трехдиагональной СЛАУ относительно коэффициентов $c_i$:

$$c_{i-1} + 4c_i + c_{i+1} = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}, \quad i = 1, \ldots, n-1$$

$$c_0 = 0, \quad c_n = 0$$

где

$$h = x_{i+1} - x_i, \quad i = 0, \ldots, n-1$$

— постоянный шаг интерполирования.

Необходимо решить полученную систему (например, методом прогонки или методом Гаусса).

Остальные коэффициенты можно выразить через $c_i$ по следующим формулам:

$$\alpha_i = y_i, \quad i = 0, \ldots, n-1;$$

$$b_i = \frac{y_{i+1} - y_i}{h} - \frac{h}{3}(c_{i+1} + 2c_i), \quad i = 0, \ldots, n-2;$$

$$b_{n-1} = \frac{y_n - y_{n-1}}{h} - \frac{2}{3}hc_{n-1};$$

$$d_i = \frac{c_{i+1} - c_i}{3h}, \quad i = 0, \ldots, n-2;$$

$$d_{n-1} = -\frac{c_{n-1}}{3h}.$$

# 3 Практическая реализация

Листинг 1 — реализация программы

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


def gauss_solve(A: list[list[float]], b: list[float]) -> list[float]:
    n = len(A)

    # k-й шаг
    for k in range(n - 1):
        # Строка с максимальным по модулю ведущим элементом в k-м столбце
        maxline = k
        for i in range(k + 1, n):
            if abs(A[i][k]) > abs(A[maxline][k]):
                maxline = i

        # Перестановка ak и amaxline
        if maxline != k:
            A[k], A[maxline] = A[maxline], A[k]
        # Перестановка bk и bmaxline
        b[k], b[maxline] = b[maxline], b[k]

        # Обнуление элементов ниже главного
        for i in range(k + 1, n):
            if A[k][k] == 0:
                raise ValueError("Нулевой ведущий элемент")
            m = A[i][k] / A[k][k]
            A[i][k] = 0.0
            for j in range(k + 1, n):
                A[i][j] -= m * A[k][j]
            b[i] -= m * b[k]

    # Обратный ход
    x = [0] * n
    for i in range(n - 1, -1, -1):
        s = sum(A[i][j] * x[j] for j in range(i + 1, n))
        if A[i][i] == 0:
            raise ValueError("На главной диагонали 0")
        x[i] = (b[i] - s) / A[i][i]

    return x


def spline_coeffs_find(
    x: list[float], y: list[float]
) -> tuple[list[int], list[int], list[float], list[int]]:
```

```
    n = len(x) - 1
    # Шаг (b - a) / 32:
    h: float = (x[n] - x[0]) / 32

    # A и b для трёхдиаг. системы относительно ci
    A = [[0] * (n + 1) for _ in range(n + 1)]
    rhs = [0] * (n + 1)

    # c0 = 0; c_n = 0
    A[0][0] = 1
    A[n][n] = 1
    rhs[0] = 0
    rhs[n] = 0

    # Строки трёхдиаг. системы
    # c{i-1} + 4ci + c{i+1} = (y{i+1} - 2yi + y{i-1}) / h^2
    for i in range(1, n):
        A[i][i - 1] = 1
        A[i][i] = 4
        A[i][i + 1] = 1

        rhs[i] = 3 * (y[i + 1] - 2 * y[i] + y[i - 1]) / (h**2)

    c_result = gauss_solve(A, rhs)

    a_vals = [0] * n
    b_vals = [0] * n
    d_vals = [0] * n

    for i in range(n):
        # ai = yi:
        a_vals[i] = y[i]

        # b_i = (y{i+1} - yi)/h - h/3 * (c{i+1} + 2ci)
        # d_i = (c{i+1} - ci) / (3 * h)
        if i != n - 1:
            b_vals[i] = (y[i + 1] - y[i]) / h - (
                c_result[i + 1] + 2 * c_result[i]
            ) * h / 3
            d_vals[i] = (c_result[i + 1] - c_result[i]) / (3 * h)
        else:
            b_vals[i] = ((y[n] - y[n - 1]) / h) - 2 * h * c_result[n - 1] / 3
            d_vals[i] = -1 * (c_result[n - 1]) / (3 * h)

    return a_vals, b_vals, c_result, d_vals


def cubic_spline_eval(
    x: list[float],
    a: list[float],
```

```
    b: list[float],
    c: list[float],
    d: list[float],
    X: float,
):
    # S_i(X) = a[i] + b[i]*(X - x[i])
    #          + c[i]*(X - x[i])^2
    #          + d[i]*(X - x[i])^3
    n = len(x) - 1

    if X < x[0] or X > x[n]:
        return None

    i = 0
    while i < n - 1 and not (x[i] <= X < x[i + 1]):
        i += 1

    dx = X - x[i]
    return a[i] + b[i] * dx + c[i] * (dx**2) + d[i] * (dx**3)


def plot(xs: list[float], ys: list[float]) -> None:
    plt.figure(figsize=(8, 5))
    plt.plot(
        xs,
        ys,
        marker="o",
        linestyle="-",
        color="b",
    )

    plt.xlabel("X")
    plt.ylabel("Y")
    plt.grid(True)

    plt.show()


def make_full_graph(
    a: list[float],
    b: list[float],
    c: list[float],
    d: list[float],
    left: float,
    right: float,
    x_nodes: list[float],
    step: float,
) -> None:
    i = left
    xs = []
```

```python
    ys = []
    while i <= right:
        xs.append(i)
        ys.append(cubic_spline_eval(x_nodes, a, b, c, d, i))
        i += step
    plot(xs, ys)


def calc_vals(
    a: list[float],
    b: list[float],
    c: list[float],
    d: list[float],
    x_nodes: list[float],
) -> None:
    for i in range(1, len(x_nodes) + 1):
        point = (i - 1 / 2) * ((x_nodes[len(x_nodes) - 1] - x_nodes[0]) / 32)
        f_val = f(point)
        phi_val = cubic_spline_eval(x_nodes, a, b, c, d, point)
        print(
            f"f({point}) = {f_val}, "
            f"phi({point}) = {phi_val}, "
            f"|f({point}) - phi({point})| = {abs(phi_val - f_val)}"
        )


def f(x: float):
    return 8 * np.sin(0.5 * x) + 4 * np.cos(0.3 * x) - 0.2 * x


def main():
    x_nodes = list(range(32))
    y_nodes = [f(x) for x in x_nodes]

    # Таблица значений
    df = pd.DataFrame({"x": x_nodes, "y": y_nodes})
    print(df.to_string(index=False))

    a, b, c, d = spline_coeffs_find(x_nodes, y_nodes)

    print(
        "Массивы коэффициентов:\na: ",
        list(map(float, a)),
        "\nb: ",
        list(map(float, b)),
        "\nc: ",
        list(map(float, c)),
        "\nd: ",
        list(map(float, d)),
    )
```

```python

    # График известных значений
    plot(x_nodes, y_nodes)

    # График сплайна
    make_full_graph(a, b, c, d, 0, 32, x_nodes, 0.1)

    print("Значения в точках x_i = a + (i + 1/2)h:")
    calc_vals(a, b, c, d, x_nodes)

    X = float(input("Введите точку (от 0 до 31): "))
    S_X = cubic_spline_eval(x_nodes, a, b, c, d, X)
    print(f"f({X}) = {f(X)}, сплайн в точке {X} равен {S_X}")


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

Для тестирования программы была выбрана функция $f(x) = 8sin(\frac{x}{2}) + 4sin(0.3x) - 0.2x$. На рисунке 1 представлена часть вывода значений исходной и интерполяционной функций в промежуточных точках отрезка, а также абсолютной погрешности. Для наглядности работы, помимо пунктов, указанных в задании, с использованием библиотеки matplotlib были также выведены графики исходной функции в точках 0...31 и интерполяционной функции в тех же точках, а также в промежуточных значениях. На рисунке 2 представлен график исходной функции (для наглядности точки соединяются отрезками). На рисунке 3 представлен график интерполяционной функции.

```
Значения в точках x_i = a + (i + 1/2)h:
f(0.484375) = 5.779582677337803, phi(0.484375) = 5.8172929278032015, |f(0.484375) - phi(0.484375)| = 0.0377102504653983
f(1.453125) = 8.6497065906599, phi(1.453125) = 8.686468371581425, |f(1.453125) - phi(1.453125)| = 0.036761780921525045
f(2.421875) = 9.993046806526493, phi(2.421875) = 9.995844390196014, |f(2.421875) - phi(2.421875)| = 0.0027975836695208756
f(3.390625) = 9.362979704389376, phi(3.390625) = 9.340212051931106, |f(3.390625) - phi(3.390625)| = 0.022767652458270504
f(4.359375) = 6.730238571032246, phi(4.359375) = 6.6868429237947655, |f(4.359375) - phi(4.359375)| = 0.043395647237480794
f(5.328125) = 2.5005206105235516, phi(5.328125) = 2.4477364999550573, |f(5.328125) - phi(5.328125)| = 0.052784110568494
f(6.296875) = -2.5658144589716434, phi(6.296875) = -2.6167190780976326, |f(6.296875) - phi(6.296875)| = 0.0509046191259892
f(7.265625) = -7.514572828314041, phi(7.265625) = -7.554264027099738, |f(7.265625) - phi(7.265625)| = 0.0396911987856976
f(8.234375) = -11.403263669681365, phi(8.234375) = -11.426545561822502, |f(8.234375) - phi(8.234375)| = 0.023281892141136
```

Рис. 1 — Часть вывода значений $f(x)$, $\varphi(x)$ и $|f(x) - \varphi(x)|$ в промежуточных точках отрезка $[0, 31]$

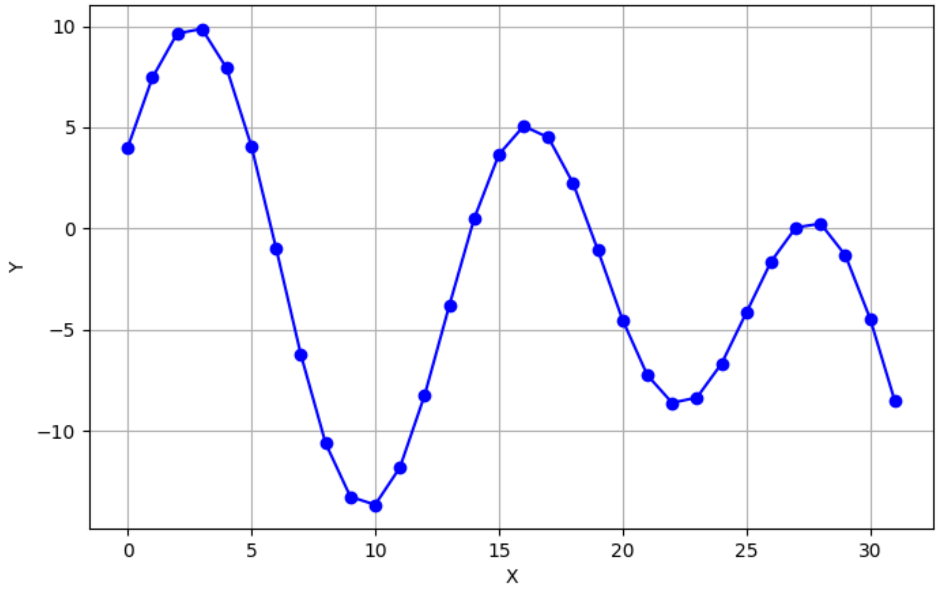


Рис. 2 — График исходной функции $f(x)$ в точках $x = 0, \ldots 31$

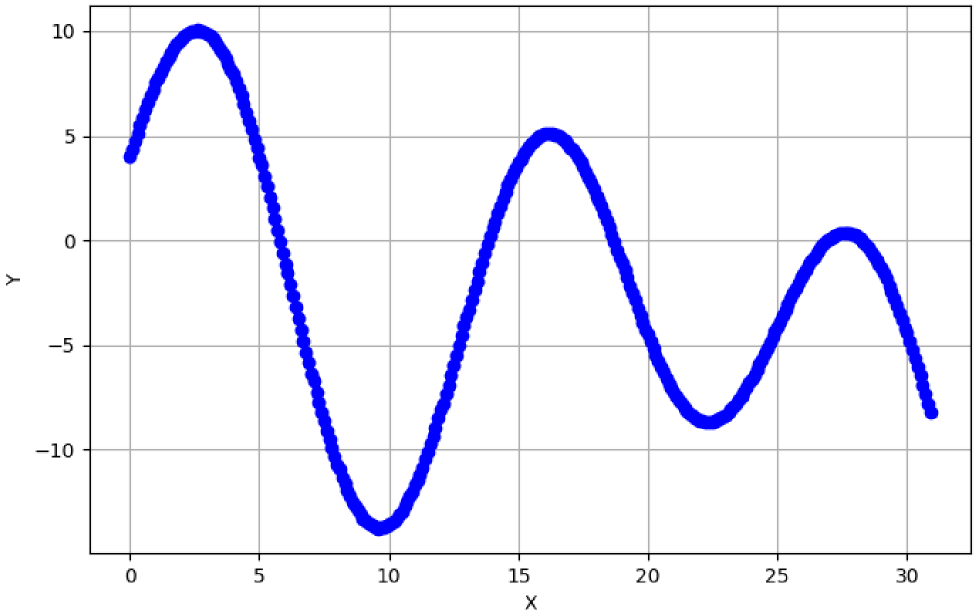


Рис. 3 — График интерполяционной функции $\varphi(x)$ в точках $x$ из промежутка $[0, 31]$ с шагом 0.1

# 4 Вывод

В данной лабораторной работе был реализован поиск интерполяционной функции (кубического сплайна). Система уравнений с неизвестными $c_i$ решается с помощью метода Гаусса, реализованного в нулевой лабораторной работе. Затем по представленным формулам вычисляются коэффициенты многочленов, после чего они выводятся на экран. После этого выводится таблица значений функции в известных точках, а также значения сплайнов в промежуточных точках. Кроме того, вычисляется абсолютная погрешность значений интерполяционной функции в промежуточных точках. Для наглядности значения функции и сплайнов были также построены на графиках. Также в программе предлагается ввести произвольную точку для вычисления в ней значений исходной и интерполяционной функций. В ходе выполнения работы была исправлена ошибка в формулах методички для $d_{n-1}$. Поскольку $c_n = 0$, при подстановке этого значения в общую формулу получается формула для границы, использующая $c_{n-1}$, а не $c_n$. Из полученных графиков наглядно видно, что сплайн построен корректно: интерполяционная функция точно равна начальной функции в узлах интерполирования, а в промежуточных узлах приближает её с некоторой погрешностью.