МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра вычислительной математики

Отчёт

Лабораторная работа 3

“Интерполирование с помощью кубического сплайна”

Вариант 5

Благодарного Артёма Андреевича
студента 3 курса, 3 группы
специальности «Информатика»
дисциплина «Численные методы»
Преподаватель: Будник А.М

Минск, 2025

Постановка Задачи

Для функции, определенной в предыдущих лабораторных работах, произвести интерполяцию кубическими сплайнами на отрезке [a, b] (в моем случае [0,35; 1,35]) по равноотстоящим узлам с естественными граничными условиями.

Алгоритм решения

Сплайн $s \in S_{\Delta}^{m}$ называется интерполяционным для функции f, если

$$s(x_i) = f(x_i) = y_i \quad \forall\, i = 0, \dots, n$$

Построение кубического интерполяционного сплайна

Каждый «кусок» сплайна будем искать в виде:

$$s_i(x) = \alpha_i + \beta_i(x - x_i) + \frac{\gamma_i}{2} \cdot (x - x_i)^2 + \frac{\delta_i}{6} \cdot (x - x_i)^3, x \in \Delta_i = [x_{i-1}, x_i].$$

Выпишем уравнения, которым должны удовлетворять искомые коэффициенты.

Имеем $s_i(x_i) = y_i$ откуда $\alpha_i = y_i, i = 1..n$.

и $s_i(x_{i-1}) = y_{i-1}$, или $\beta_i = \frac{(y_i - y_{i-1})}{h_i} + \frac{\gamma_i}{2} \cdot h_i - \frac{\delta_i}{6} \cdot h_i^2, i = 1, \dots, n$.

Требование непрерывности второй производной $s_i''(x_{i-1}) = s_{i-1}''(x_{i-1})$ по аналогии даёт $\delta_i = \frac{(\gamma_i - \gamma_{i-1})}{h_i},\ i = 2, \dots, n.$

Для вычисления γ_i:

$$h_i\gamma_{i-1} + 2(h_i + h_{i+1})\gamma_i + h_{i+1}\gamma_{i+1} = 6(\frac{(y_{i+1} - y_i)}{h_{i+1}} - \frac{(y_i - y_{i-1})}{h_i}).$$

В нашем случае на равномерной сетке все $h_i = 0.1 = const.$

Граничные условия (естественные): $s''(a) = s''(b) = 0.$

Тогда имеем: $\gamma_0 = \gamma_n = 0.$

Полученную систему решают методом прогонки.

Общая схема вычисления интерполяционного кубического сплайна:

1. Вычисляются моменты $\{\gamma_i\}_{i=0}^{n} = 0$ как решение СЛАУ, дополненной двумя дополнительными уравнениями — граничными условиями.
2. Находятся остальные коэффициенты по формулам (1), (2), (3).

Также вычислим истинную погрешность по формуле: $r_n(x^*) = f(x^*) - P_n(x^*)$.

А для оценки погрешности будем использовать: $\|f - s\| \le h^4 \|f^4\|$.

Таким образом, для достаточно гладких f погрешность интерполирования кубическим сплайном равна $O(h^4)$.

Максимум производной функции определим аналитически: $f^{(4)} = 0.35 * e^x + 0.65 * sin(x)$ и ее максимум на отрезке [0,35; 1,35] примерно равен 1.98432.

Листинг программы

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

# Параметры
j = 5
alpha_j = 0.1 + 0.05 * j
n = 10
h = 1 / n

# Функция f(x)
def f(x):
    return alpha_j * np.exp(x) + (1 - alpha_j) * np.sin(x)

# Узлы интерполяции
x_vals = np.array([alpha_j + i * h for i in range(n + 1)])
f_vals = f(x_vals)

# Проверочные точки
x_star = x_vals[0] + (2 / 3) * h
x_star2 = x_vals[n // 2] + (1 / 2) * h
x_star3 = x_vals[-1] - (1 / 3) * h

# Производные f(x)
x = sp.Symbol('x')
f_sym = alpha_j * sp.exp(x) + (1 - alpha_j) * sp.sin(x)

# Построение сплайна
def calc_spline_coeffs():
    alpha = f_vals.copy()
    beta = [0] * (n + 1)
    gamma = [0] * (n + 1)
    delta = [0] * (n + 1)
    a_coef = [0] * n
    b_coef = [0] * n

    for i in range(1, n):
        temp1 = 6 * ((f_vals[i + 1] - f_vals[i]) - (f_vals[i] - f_vals[i - 1])) / h
        temp2 = 4 * h
        a_coef[i] = -h / temp2
        b_coef[i] = (temp1 - h * b_coef[i - 1]) / temp2

    for i in range(n - 1, 0, -1):
        gamma[i] = a_coef[i] * gamma[i + 1] + b_coef[i]

    f_derivative_2 = sp.diff(f_sym, x, 2)
    gamma[0] = f_derivative_2.subs(x, x_vals[0])
```

```
    gamma[n] = f_derivative_2.subs(x, x_vals[-1])

    for i in range(n, 0, -1):
        delta[i] = (gamma[i] - gamma[i - 1]) / h
        beta[i] = (f_vals[i] - f_vals[i - 1]) / h + (2 * gamma[i] + gamma[i - 1]) * h
/ 6

    return alpha, beta, gamma, delta

# Вычисление значения сплайна в точке x
def calc_f(x, x_i, a, b, c, d):
    dx = x - x_i
    return a + b * dx + 0.5 * c * dx ** 2 + (1 / 6) * d * dx ** 3

def calc_spline(x, nodes, P, n):
    for i in range(1, n):
        if nodes[i - 1] <= x <= nodes[i]:
            return calc_f(x, nodes[i], *P[i])
    return None


alpha, beta, gamma, delta = calc_spline_coeffs()

# Заполнение таблицы P из коэффициентов
P = [[0.0] * 4 for _ in range(n + 1)]
for i in range(n + 1):
    P[i][0] = alpha[i]  # значение функции в узле x_i
    P[i][1] = beta[i]   # первая производная в x_i
    P[i][2] = gamma[i]  # вторая производная в x_i
    P[i][3] = delta[i]  # третья производная в x_i

S_x_star = calc_spline(x_star, x_vals, P, n + 1)
S_x_star2 = calc_spline(x_star2, x_vals, P, n + 1)
S_x_star3 = calc_spline(x_star3, x_vals, P, n + 1)

df = pd.DataFrame({
    "Точка": ["x*", "x**", "x***"],
    "Значение x": [x_star, x_star2, x_star3],
    "f(x)": [f(x_star), f(x_star2), f(x_star3)],
    "S(x) (сплайн)": [S_x_star, S_x_star2, S_x_star3]
})

# Оценка погрешности
f_derivative_4 = sp.diff(f_sym, x, 4)
f_derivative_4_abs = sp.lambdify(x, sp.Abs(f_derivative_4), 'numpy')

a, b = x_vals[0], x_vals[-1]
x_test = np.linspace(a, b, 1000)
M_max = np.max(f_derivative_4_abs(x_test))
error_bound = h ** 4 * M_max
```

```
# Истинная ошибка
r_x_star = round(abs(f(x_star) - S_x_star), 6)
r_x_star2 = round(abs(f(x_star2) - S_x_star2), 6)
r_x_star3 = round(abs(f(x_star3) - S_x_star3), 6)

is_error_bound_valid = [
    r_x_star <= error_bound,
    r_x_star2 <= error_bound,
    r_x_star3 <= error_bound
]

error_table = pd.DataFrame({
    "Точка": ["x*", "x**", "x***"],
    "Значение x": [x_star, x_star2, x_star3],
    "r(ист)": [r_x_star, r_x_star2, r_x_star3],
    "Оценка погрешности (h^4 * M)": [error_bound] * 3,
    "M = max|f^(4)(x)|": [M_max] * 3,
    "Неравенство выполняется?": is_error_bound_valid
})

# Отображение таблиц
display(df)
display(error_table)
```

Результаты

- Значение функции и полинома в точках x*, x**, x***

	Точка	Значение x	f(x)	S(x) (сплайн)
0	x*	0.416667	0.793978	0.793931
1	x**	0.900000	1.370024	1.370042
2	x***	1.316667	1.934961	1.934887

- Оценка погрешности

	Точка	Значение x	r(ист)	Оценка погрешности (h^4 * M)	M = max|f^(4)(x)|	Неравенство выполняется?
0	x*	0.416667	0.000048	0.000198	1.984319	True
1	x**	0.900000	0.000018	0.000198	1.984319	True
2	x***	1.316667	0.000074	0.000198	1.984319	True

Анализ результатов

В процессе интерполяции функции с использованием кубических сплайнов наблюдается разброс погрешности, что обусловлено характеристиками аппроксимируемой функции и особенностями самого сплайна.

1. **Правая часть интервала**, где преобладает экспоненциальная составляющая функции, приводит к увеличению погрешности интерполяции, приближающейся к теоретической верхней границе. Это связано с быстрым изменением производных функции в данной области.
2. **Центральная и левая части интервала**, где функция изменяется плавно, показывают погрешность значительно ниже теоретической оценки, особенно в точке $x = 0.9$, где ошибка в 11 раза меньше прогноза. Это объясняется медленным изменением функции в этих областях.
3. **Равномерность оценки погрешности** (одинаковая для всех точек) объясняется глобальной природой теоретической оценки, основанной на максимуме четвертой производной на всем интервале, в то время как фактическая погрешность зависит от локальных свойств функции.

Таким образом, разброс погрешности связан с тем, что теоретическая оценка погрешности ориентирована на глобальное максимальное значение четвертой производной, в то время как реальная ошибка зависит от локальных изменений функции на интервале.