МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра вычислительной математики

Отчёт

Лабораторная работа 4

“Интерполирование с помощью метода наименьших квадратов”

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 с помощью метода наименьших квадратов на отрезке [a, b] (в моем случае [0,35; 1,35]) по равноотстоящим узлам.

Алгоритм решения

Задача дискретного среднеквадратичного приближения для данного набора точек $(x_k, y_k), k = 0,..,N$, заключается в построении функции ϕ вида $\phi(x) = \sum_{i=0}^{n} \alpha_i \phi_i(x)$, где α - является решением системы линейных уравнений: $\sum_{j=0}^{n} \gamma_{lj}\alpha_j = \beta_l, l = 0, \ldots, n$.
В матричном виде $\Gamma\alpha = \beta$, где $\Gamma = (\gamma_{ij})_{i,j=0}^{n}$, $\gamma_{ij} = (\phi_i, \phi_j)$, $\beta_i = (y, \phi_i)$.

Будем строить многочлен МНК 5 степени. Положим $\varphi_i = x^i, i = \overline{0,5}$
Скалярное произведение будем вычислять по формуле: $(u, v) = \sum_{i=0}^{N} u(x)v(x)$.

Таким образом γ_{ij} будет вычисляться: $\gamma_{ij} = \sum_{k=0}^{N} \phi_i(x_k)\phi_j(x_k)$, а β_i тогда будут равны:

$\beta_i = \sum_{k=0}^{N} y_k \phi_i(x_k)$.

N возьмём равной 10.

Погрешность будем вычислять по следующей формуле:

$$\Delta = \left(\sum_{k=0}^{N}\left(y_k - \sum_{i=0}^{n} \alpha_i \varphi_i(x_k)\right)^2\right)^{1/2}$$

Листинг программы

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

# Параметры
j = 5
N = 10  # количество точек - 1
h = 1 / N
alpha_j = 0.1 + 0.05 * j
n = 6   # количество базисных функций: φ_0 до φ_5 => многочлен 5-й степени

# 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 + 0.1 * i for i in range(N + 1)])
y_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

TEST_POINTS = [x_star, x_star2, x_star3]

def phi(i, x):
    return x ** i

def calculate_gram_matrix(n, x_vals):
    A = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            A[i, j] = sum(phi(i, xk) * phi(j, xk) for xk in x_vals)
    return A

def calculate_beta(n, x_vals, y_vals):
    b = np.zeros(n)
    for i in range(n):
        b[i] = sum(fx * phi(i, xk) for fx, xk in zip(y_vals, x_vals))
    return b

def solve_system(A, b):
    return np.linalg.solve(A, b)

def approximate(alpha, x):
    return sum(alpha[i] * phi(i, x) for i in range(len(alpha)))
```

```python
def calculate_error(alpha, x_vals):
    return np.sqrt(sum((f(xk) - approximate(alpha, xk)) ** 2 for xk in x_vals))

# Основная логика
A = calculate_gram_matrix(n, x_vals)
b = calculate_beta(n, x_vals, y_vals)
alpha = solve_system(A, b)

df_alpha = pd.DataFrame([ [round(a, 6) for a in alpha] ],
                        columns=[f"α{i}" for i in range(len(alpha))])

app_x_star = approximate(alpha, x_star)
app_x_star2 = approximate(alpha, x_star2)
app_x_star3 = approximate(alpha, x_star3)

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)],
    "φ(x)": [app_x_star, app_x_star2, app_x_star3]
})

# # Истинная ошибка
r_x_star = round(abs(f(x_star) - app_x_star),10)
r_x_star2 = round(abs(f(x_star2) - app_x_star2), 10)
r_x_star3 = round(abs(f(x_star3) - app_x_star3), 10)

error_bound = calculate_error(alpha, x_vals)

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 Δ": [error_bound] * 3,
    "Неравенство выполняется?": is_error_bound_valid
})

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

Результаты

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

	Точка	Значение x	f(x)	φ(x)
0	x*	0.461111	0.844256	0.844256
1	x**	0.733333	1.163781	1.163781
2	x***	1.294444	1.902476	1.902476

- Значение коэффициентов $\phi(x)$:

	α0	α1	α2	α3	α4	α5
0	0.349912	1.000766	0.172375	-0.045436	0.010305	0.010433

- Оценка погрешности

	Точка	Значение x	r(ист)	Оценка погрешности Δ	Неравенство выполняется?
0	x*	0.461111	2.011000e-07	5.486922e-07	True
1	x**	0.733333	1.060000e-08	5.486922e-07	True
2	x***	1.294444	3.455000e-07	5.486922e-07	True

Анализ результатов

В процессе интерполяции с использованием метода наименьших квадратов (МНК) погрешность аппроксимации была вычислена для нескольких тестовых точек. Все полученные значения погрешности находятся в пределах порядка 10^{-7}, что указывает на очень высокую точность приближения, обеспечиваемую методом. Эти значения погрешности оказываются меньше теоретической оценки, что говорит о повышенной эффективности выбранного подхода и точности решения задачи.

Важно отметить, что погрешности для всех тестовых точек имеют схожий порядок величины, что свидетельствует о равномерности погрешности по всему интервалу. Это подтверждает, что метод наименьших квадратов не только эффективно аппроксимирует функцию, но и сохраняет стабильность результатов по всему отрезку.

Устойчивость метода к погрешностям также подтверждается тем, что в расчетах не возникли значительные отклонения между фактическими значениями и аппроксимированными, а значения погрешности остаются стабильно малыми.

Выводы

Полином 5-й степени, использованный для аппроксимации функции, оказался эффективным инструментом для приближения исходной функции на интервале [0.35,1.35]. Этот полином не только предоставляет хорошее приближение, но и равномерно хорошее приближение на всем заданном интервале. Это свидетельствует о том, что выбранная степень полинома достаточна для решения задачи, и использование более высокой степени не даст значительного улучшения точности.

Таким образом, метод наименьших квадратов в рамках данного эксперимента показал высокую эффективность и устойчивость при аппроксимации функции. Использование полинома 5-й степени обеспечило равномерное и точное приближение на заданном интервале, а фактическая точность значительно превысила теоретические ожидания, что подтверждает надежность и практическую применимость метода в решении задач аппроксимации.