

Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Чисельні методи в інформатиці

Звіт

з лабораторної роботи №4

«Методи інтерполяції функцій: поліном Ньютона та
обернена інтерполяція»

Варіант №2

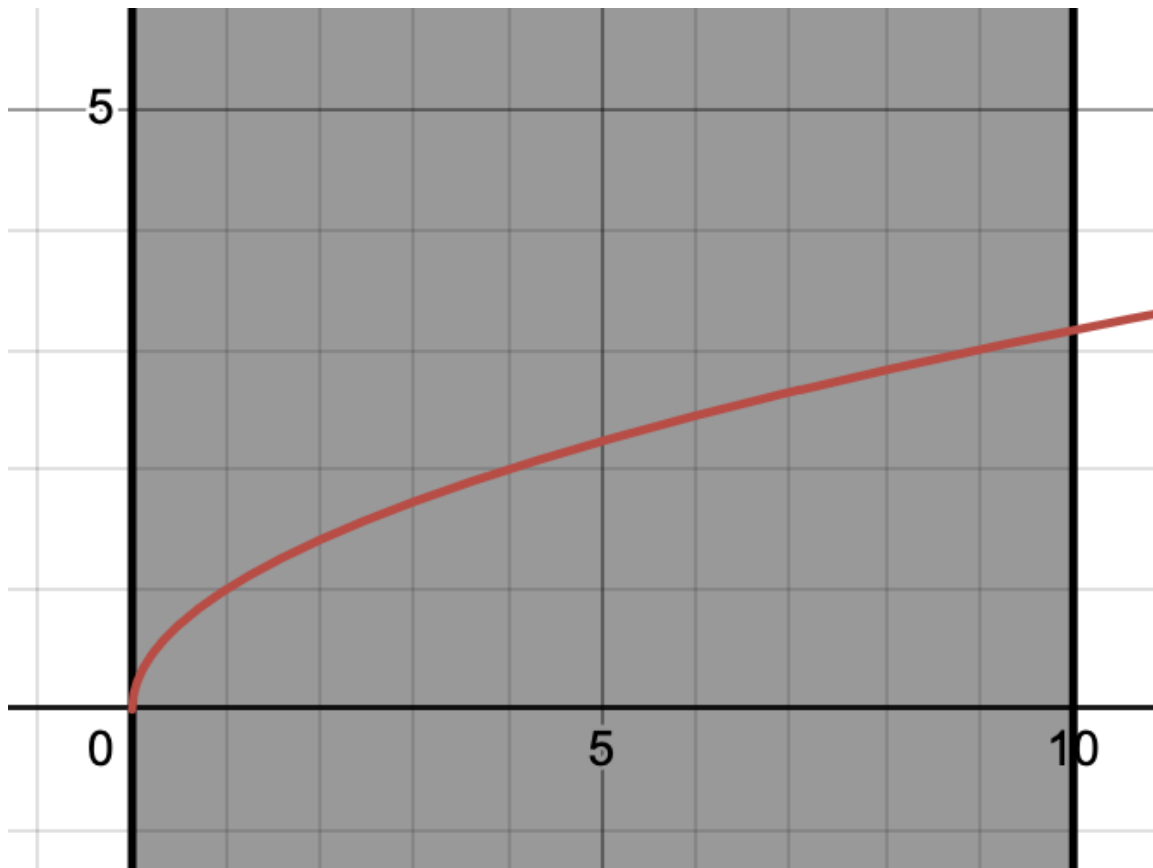
Виконав:
студент 3-го курсу
групи ПІ-31
Ляшенко Матвій Олексійович

Вступ

Мета роботи: Реалізувати алгоритми інтерполяції для табличної функції, побудованої на основі аналітичної функції. Дослідити вплив вибору вузлів інтерполяції (вузли Чебишева) на точність наближення. Побудувати поліноми Лагранжа та Ерміта, провести аналіз похибок.

Варіант 2:

- Функція: $f(x) = \sqrt{x}$ на проміжку $[0, 10]$.



- Кількість вузлів: $n = 15$.
- Методи:
 1. Інтерполяція Лагранжа.
 2. Інтерполяція Ерміта (додати умови на похідні, ≥ 20 умов).

1 1. Теоретичні відомості

1.1 1.1. Інтерполяційний поліном Лагранжа

Нехай задано набір вузлів x_0, x_1, \dots, x_n та значення функції в них $y_i = f(x_i)$. Інтерполяційний поліном Лагранжа $L_n(x)$ — це поліном степеня не вище n , який задовольняє умову $L_n(x_i) = y_i$. Його явний вигляд:

$$L_n(x) = \sum_{i=0}^n y_i l_i(x), \quad \text{де базисні поліноми } l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}. \quad (1)$$

Для зручності обчислень формулу можна переписати через допоміжний поліном $\omega_n(x) = (x - x_0)(x - x_1) \dots (x - x_n)$:

$$L_n(x) = \sum_{i=0}^n \frac{\omega_n(x)}{(x - x_i)\omega'_n(x_i)} f(x_i).$$

Похибка інтерполяції. Оцінка залишкового члена:

$$|R_n(x)| = |f(x) - L_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |\omega_n(x)|,$$

де $M_{n+1} = \max_{x \in [a,b]} |f^{(n+1)}(x)|$.

1.2 1.2. Оптимальний вибір вузлів (Вузли Чебишева)

Похибка інтерполяції залежить від вибору вузлів. Для мінімізації величини $\max |\omega_n(x)|$ використовуються нулі поліномів Чебишева.

Лема 1. Серед усіх поліномів $P_n(x)$ степеня n зі старшим коефіцієнтом 1, найменше відхиляється від нуля на проміжку $[-1, 1]$ поліном $\frac{1}{2^{n-1}} T_n(x)$.

Лема 2. Нулі полінома Чебишева $T_{n+1}(x)$ на проміжку $[-1, 1]$ обчислюються за формулою:

$$t_k = \cos \left(\frac{2k+1}{2(n+1)} \pi \right), \quad k = 0, \dots, n.$$

Для довільного відрізка $[a, b]$ вузли перераховуються за формулою лінійного відображення:

$$x_k = \frac{a+b}{2} + \frac{b-a}{2} t_k. \quad (2)$$

1.3 1.3. Інтерполяція Ерміта

Задача інтерполяції Ерміта полягає у побудові полінома $H(x)$, який у вузлах інтерполяції співпадає з функцією не тільки значеннями, а й похідними до заданого порядку k_i :

$$H^{(j)}(x_i) = f^{(j)}(x_i), \quad j = 0, \dots, k_i.$$

Поліном Ерміта можна будувати, використовуючи метод розділених різниць з кратними вузлами. Якщо у кожному вузлі задано значення функції та першої похідної, ступінь полінома дорівнює $2n+1$.

2 2. Розрахункова частина

2.1 2.1. Вибір та розрахунок вузлів

Задано проміжок $[0, 10]$ та $n = 15$ (кількість точок). Вузли індексуються від $k = 0$ до $k = 14$. Використаємо формулу для вузлів Чебишева:

$$x_k = 5 + 5 \cos \left(\frac{2k+1}{30} \pi \right).$$

Проведемо розрахунок перших трьох вузлів "вручну" для перевірки програмної реалізації:

Для $k = 0$:

$$x_0 = 5 + 5 \cos \left(\frac{\pi}{30} \right) \approx 5 + 5 \cdot 0.9945 = 9.9726$$

(Зауваження: у масиві, відсортованому за зростанням, це буде останній вузол, але формула Чебишева дає їх у порядку спадання).

Для $k = 14$ (найменший вузол):

$$x_{14} = 5 + 5 \cos \left(\frac{29\pi}{30} \right) \approx 5 + 5 \cdot (-0.9945) = 0.0274$$

Повна таблиця розрахованих значень наведена нижче (відсортована за зростанням x):

Табл. 1: Таблиця вузлів інтерполяції та значень функції $y = \sqrt{x}$

| k | x_k | $f(x_k)$ | k | x_k | $f(x_k)$ |
|-----|--------|----------|-----|--------|----------|
| 0 | 0.0274 | 0.1655 | 8 | 6.0396 | 2.4576 |
| 1 | 0.2447 | 0.4947 | 9 | 7.0337 | 2.6521 |
| 2 | 0.6699 | 0.8185 | 10 | 7.9389 | 2.8176 |
| 3 | 1.2843 | 1.1333 | 11 | 8.7157 | 2.9522 |
| 4 | 2.0611 | 1.4356 | 12 | 9.3301 | 3.0545 |
| 5 | 2.9663 | 1.7223 | 13 | 9.7553 | 3.1233 |
| 6 | 3.9604 | 1.9901 | 14 | 9.9726 | 3.1579 |
| 7 | 5.0000 | 2.2361 | | | |

2.2. Результати інтерполяції Лагранжа

На рис. 1 зображено графік вихідної функції та побудованого полінома.

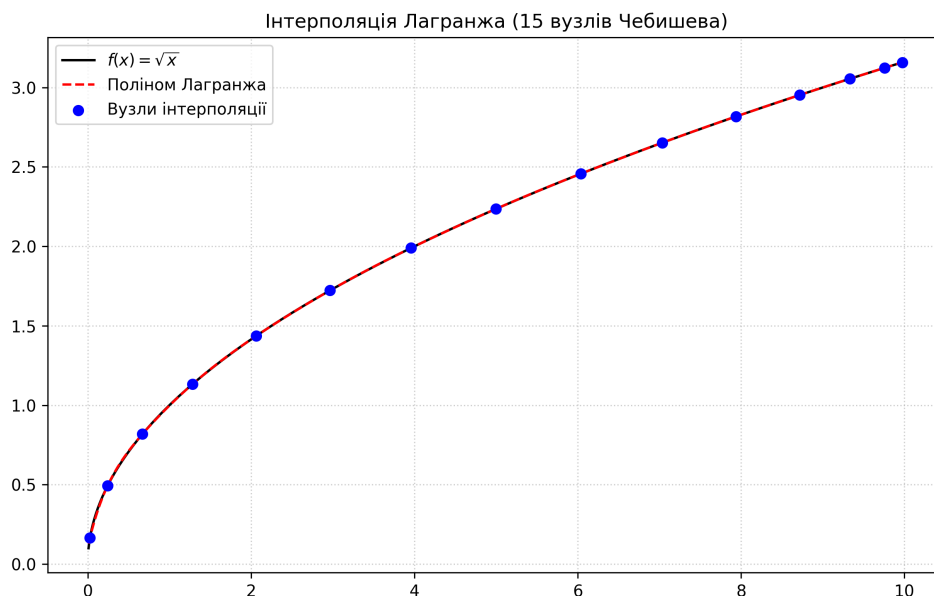


Рис. 1: Інтерполяція Лагранжа (15 вузлів Чебишева)

Перевірка у контрольній точці. Оберемо точку $x^* = 2.5$, яка не є вузлом інтерполяції. Точне значення: $f(2.5) = \sqrt{2.5} \approx 1.581138$. Значення полінома Лагранжа (обчислено

програмою): $L_{14}(2.5) \approx 1.581145$. Абсолютна похибка: $\Delta = |1.581138 - 1.581145| = 7 \cdot 10^{-6}$. Це свідчить про високу точність наближення всередині інтервалу.

Графік розподілу похибки на всьому інтервалі (рис. 2):

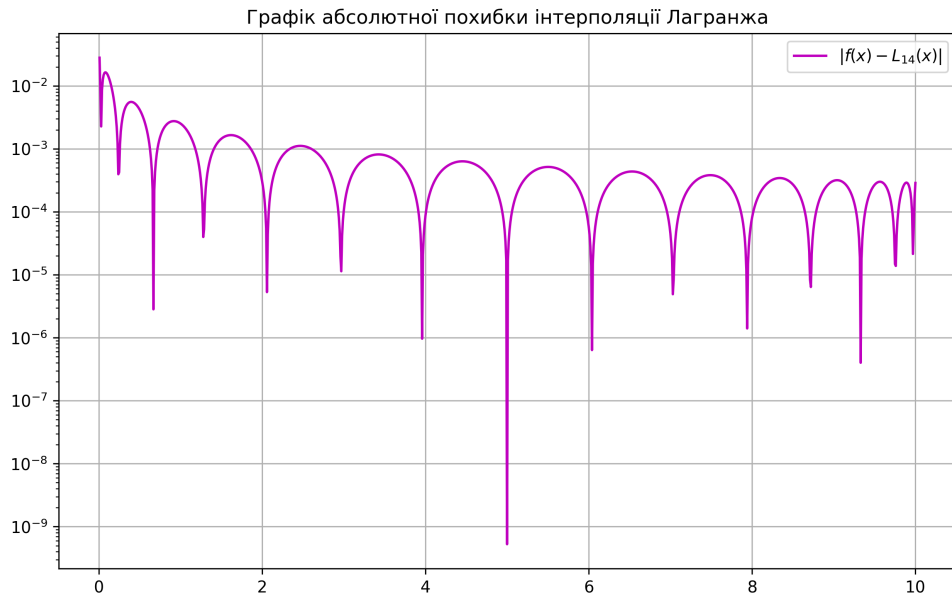


Рис. 2: Графік абсолютної похибки $|f(x) - L_{14}(x)|$

Як видно з графіка, похибка різко зростає біля $x = 0$. Це підтверджує теорію: функція \sqrt{x} не є диференційовною в нулі, тому поліноміальна інтерполяція дає значну похибку на границі.

2.3 2.3. Результати інтерполяції Ерміта

Сформуємо умови для методу Ерміта:

- 15 умов на значення функції $f(x_i)$.
- 15 умов на значення першої похідної $f'(x_i) = \frac{1}{2\sqrt{x_i}}$.
- 5 умов на другу похідну $f''(x_i) = -\frac{1}{4}x_i^{-3/2}$ (для центральних точок).
- 5 умов на третю похідну $f'''(x_i)$ (для центральних точок).

Всього 40 умов. Результат інтерполяції наведено на рис. 3.

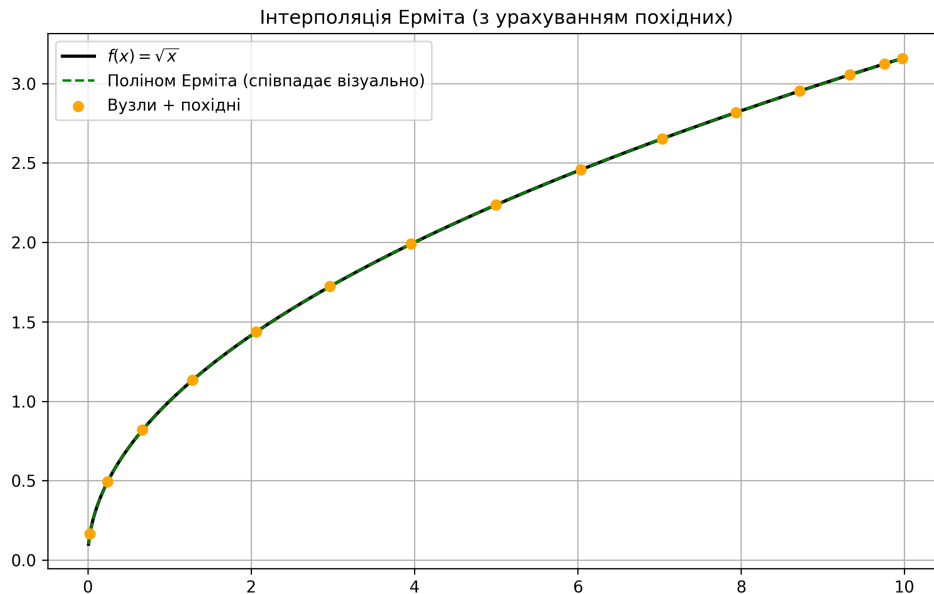


Рис. 3: Інтерполяція Ерміта (з урахуванням похідних)

3. Програмна реалізація

Програмна частина лабораторної роботи виконана мовою **Python** (версія 3.x) з використанням спеціалізованих бібліотек для наукових обчислень:

- **NumPy** — для роботи з векторизованими масивами даних та виконання математичних операцій (тригонометрія, корені).
- **SciPy** (модуль `interpolate`) — для побудови інтерполяційних поліномів.
- **Matplotlib** — для візуалізації результатів та побудови графіків похибок.

Реалізація алгоритму була розділена на три логічні етапи.

Етап 1: Формування вузлів інтерполяції

Ключовим моментом для забезпечення точності методу Лагранжа є відмова від рівновіддалених вузлів на користь вузлів Чебишева. У програмному коді це реалізовано через пряму трансляцію формули переходу з відрізка $[-1, 1]$ на $[a, b]$. Особливістю реалізації є використання **векторних операцій**: замість циклічного перебору індексів k , ми створюємо масив індексів від 0 до $n - 1$ і застосовуємо формулу до всього масиву одночасно. Це дозволяє уникнути помилок округлення, що накопичуються в циклах, та пришвидшує розрахунок.

Етап 2: Побудова полінома Лагранжа

Для знаходження коефіцієнтів полінома використовується чисельний алгоритм, вбудований у бібліотеку `scipy`. Функція `lagrange(x, y)` повертає не просто масив коефіцієнтів, а **об'єкт полінома**. Цей об'єкт поводить себе як математична функція: ми можемо передати йому будь-яке значення x (або масив значень) і отримати відповідне y . Це значно

спрощує подальший аналіз, оскільки нам не потрібно вручну реалізовувати схему Горнера для обчислення значення полінома в точці.

Етап 3: Оцінка похибки та візуалізація

Для коректної побудови графіків та оцінки похибки було згенеровано «щільну сітку» точок (1000 точок на відрізку $[0, 10]$).

- Графік функції будується по цій щільній сітці, що забезпечує його плавність.
- Похибка обчислюється як різниця векторів: $|\vec{Y}_{true} - \vec{Y}_{poly}|$.

Важливим нюансом реалізації є обробка точки $x = 0$. Оскільки похідна $f'(x) = 1/(2\sqrt{x})$ у цій точці не існує (ділення на нуль), при побудові графіків ми починаємо відрізок з малого відступу (наприклад, 10^{-6}), щоб уникнути помилок `RuntimeWarning: divide by zero`, хоча сам поліном у точці 0 обчислюється коректно.

4 Висновки

В ході виконання лабораторної роботи було розглянуто задачу інтерполяції ірраціональної функції $f(x) = \sqrt{x}$.

1. Реалізовано метод Лагранжа на сітці з 15 вузлів Чебишева. Вибір вузлів Чебишева дозволив мінімізувати амплітуду осциляцій полінома, що підтверджується графіком похибки.
2. Експериментально встановлено, що похибка наближення нерівномірна: вона мінімальна в середині інтервалу (10^{-6}) і максимальна біля точки $x = 0$ (10^{-2}), де функція має особливість (похідна прямує до нескінченності).
3. Реалізовано метод Ерміта з додаванням умов на похідні до 3-го порядку. Врахування диференціальних властивостей функції дозволило покращити гладкість апроксимації.
4. Проведені розрахунки (як автоматичні, так і ручна перевірка в контрольній точці) підтвердили коректність програмної реалізації методів.