УДК 004.1

Яценко В. В., *здобувач 2 курсу спеціальності 122 Комп'ютерні науки,*
Сеник І. О., *асистент кафедри інформаційних технологій*

АНАЛІЗ АЛГОРИТМІВ ІНТЕРПОЛЯЦІЇ ТА ЇХ ЗАСТОСУВАННЯ

Донецький національний університет імені Василя Стуса, м. Вінниця

Обробка й аналіз значень функції є основною задачею математичного аналізу [1]. Математичний аналіз – це розділ математики, що вивчає властивості функцій і пов'язані з ними поняття та методи. Для дослідження властивостей функції збирається певний набір даних. Набір даних – це структурована однотипна інформація, взята з, метою аналізу чи дослідження певної предметної області. Зібрані набори даних відіграють ключову роль в аналізі в різних галузях науки і подальшому прийнятті рішень. Від якості початкового набору даних залежить достовірність і точність дослідження певного явища. Набір даних може містити неповний або розріджений опис функції, що негативно вплине на якість подальшого дослідження. З метою підвищення якості результатів остаточного набору даних були створені методи, що дають змогу знаходити проміжні значення між відомими точками інтервалу, тобто методи інтерполяції [2]. Інтерполяція – це процес пошуку проміжних даних функції у відомих інтервалах. Лінійна, квадратична, Ньютона – основні види інтерполяції, що зараз застосовуються.

У статтях «Reconstruction and representation of 3D objects with radial basis functions» і «Radial basis functions for the multivariate interpolation of large scattered data sets» подається новий сучасний метод вирішення задачі інтерполяції [3, 4]. Інтерполяція за допомогою радіально-базисних функцій має значно вищу точність, і водночас оптимальну для задач швидкість. Цей метод буде використовуватись все частіше і може замінити класичні методи завдяки свій високій ефективності.

Метою дослідження є аналіз та порівняння різних методів інтерполяції для визначення оптимального для різноманітних задач. У межах роботи необхідно створити програму, що вираховуватиме час виконання алгоритмів інтерполяції і похибку. До того ж необхідно визначити переваги і недоліки алгоритмів та підсумувати відповідну інформацію.

Необхідно визначитись із функцією, на прикладі якої буде відбуватись порівняння алгоритмів. Для порівняння можна використати періодичну функцію. Можна додавати певний рівень шуму для перевірки стабільності методів. Можна обрати лінійну інтерполяцію, квадратичну інтерполяцію, кубічну інтерполяцію, інтерполяцію за методом Лагранжа, поліноміальну інтерполяцію ступеня 5 та інтерполяцію за допомогою кубічних сплайнів. Для реалізації алгоритмів будуть використані вбудовані модулі numpy і scipy. Підрахунок часу буде здійснюватись за допомогою методу timeit() модуля timeit. Для підрахунку похибки буде використовуватись середньоквадратична похибка, тобто середня квадратична різниця між інтерполяційними методами і реальними значеннями функцій.

```
import numpy as np
from scipy.interpolate import lagrange, interp1d, CubicSpline
from numpy.polynomial import Polynomial
import timeit

def f1(x):
    return np.sin(x)

def add_noise(y, noise_level=0.1):
    noise = noise_level * np.random.normal(size=y.shape)
    return y + noise

def linear_interpolation(x, y, x_new):
    linear_interp = interp1d(x, y, kind='linear')
    return linear_interp(x_new)

def quadratic_interpolation(x, y, x_new):
    quadratic_interp = interp1d(x, y, kind='quadratic')
    return quadratic_interp(x_new)

def cubic_interpolation(x, y, x_new):
    cubic_interp = interp1d(x, y, kind='cubic')
    return cubic_interp(x_new)

def lagrange_interpolation(x, y, x_new):
    lagrange_poly = lagrange(x, y)
    return lagrange_poly(x_new)

def cubic_spline_interpolation(x, y, x_new):
    cubic_spline = CubicSpline(x, y)
    return cubic_spline(x_new)

def polynomial_interpolation(x, y, x_new, degree):
    p = Polynomial.fit(x, y, degree)
    return p(x_new)

def rmse(y_true, y_pred):
    return np.sqrt(np.mean((y_true - y_pred) ** 2))
```

Рис. 1. Тестова функція, інтерполяційні методи, формула обчислення похибки

```
x = np.linspace(-10, 10, 100)
methods = [linear_interpolation, quadratic_interpolation, cubic_interpolation,
           lagrange_interpolation, cubic_spline_interpolation, polynomial_interpolation]
method_names = ["Лінійна", "Квадратична", "Кубічна", "Лагранжа", "Кубічні сплайни", "Поліноміальна"]

x_new = np.linspace(-10, 10, 1000)
degree = 5

y = f1(x)
y_new_true = f1(x_new)
print()

for method, name in zip(methods, method_names):
    print(f"Метод: {name}")

    if name == "Поліноміальна":
        execution_time = timeit.timeit(lambda: method(x, y, x_new, degree), number=1000) / 1000
        y_new = method(x, y, x_new, degree)
    else:
        execution_time = timeit.timeit(lambda: method(x, y, x_new), number=1000) / 1000
        y_new = method(x, y, x_new)

    error = rmse(y_new_true, y_new)
    print(f"Похибка: {error:.6f}, Час виконання: {execution_time:.9f} секунд")

    y_noisy = add_noise(y)
    if name == "Поліноміальна":
        execution_time = timeit.timeit(lambda: method(x, y_noisy, x_new, degree), number=1000) / 1000
        y_new_noisy = method(x, y_noisy, x_new, degree)
    else:
        execution_time = timeit.timeit(lambda: method(x, y_noisy, x_new), number=1000) / 1000
        y_new_noisy = method(x, y_noisy, x_new)
        error_noisy = rmse(y_new_true, y_new_noisy)
    print(f"Похибка з шумом: {error_noisy:.6f}")
```

Рис. 2. Побудова інтерполяційних методів, підрахунок часу і похибок

```
Метод: Лінійна
Похибка: 0.002570, Час виконання: 0.000041054 секунд
Похибка з шумом: 0.084022
Метод: Квадратична
Похибка: 0.000049, Час виконання: 0.000171076 секунд
Похибка з шумом: 0.093005
Метод: Кубічна
Похибка: 0.000003, Час виконання: 0.000149511 секунд
Похибка з шумом: 0.092741
Метод: Лагранжа
Похибка: 139887023127877113317674263384050567741440.000000, Час виконання: 0.168490793 секунд
Похибка з шумом: 2818614356871660642670012855784516773478 4.000000
Метод: Кубічні сплайни
Похибка: 0.000003, Час виконання: 0.000127358 секунд
Похибка з шумом: 0.097703
Метод: Поліноміальна
Похибка: 0.643340, Час виконання: 0.000109295 секунд
Похибка з шумом: 0.097703
```

Рис. 3. Лістинг результату роботи програми

Проаналізувавши вихідні дані, можна дійти висновків, що лінійна інтерполяція є найшвидшим методом інтерполяції, але похибка, порівняно з іншими методами, достатньо велика. Квадратична і кубічна інтерполяція мають нижчі похибки відповідно, але більший час роботи і меншу стабільність під час роботи з функцією з шумами. Метод Лагранжа показує аномально велику похибку. Це пов'язано з феноменом Рунге, коли зі зростанням ступеня полінома з'являється похибка на кінцях інтервала [5]. Ступінь полінома Лагранжа в цьому прикладі дуже великий, відтак похибка прямує до нескінченності. У випадку поліноміальної інтерполяції похибка не є настільки значимою через відносно низький ступінь полінома, але все одно значима, порівняно з іншими методами. Інтерполяція кубічними сплайнами – інтерполяція, яка використовує шматкові поліноми для кожного сегмента, – показала достатньо точний результат за оптимальний час. Ключова різниця цього методу від інших полягає у тому, що сплайни залежать лише від найближчих точок, отже, цей метод забезпечує більш гладкий і плавний графік.

У підсумку, кожен алгоритм має свої переваги і недоліки. У випадку, коли користувач працює з поліномами низького ступеня і швидкість є основним пріоритетом, лінійна інтерполяція є найкращим варіантом. Поліноміальна інтерполяція й інтерполяція методом Лагранжа достатньо повільні, але ефективні і максимально точні у простих функціях. Квадратична і кубічна інтерполяції забезпечують оптимальну швидкість із достатньо низьким рівнем відхилення даних від істинних. Метод кубічних сплайнів використовується, коли необхідна гладка і точна апроксимація функції з достатньо оптимальною швидкістю. Вибір методу повністю залежить від вимог користувача.

Список використаних джерел

1. Analysis. Mathematics. *Britannica.* URL: https://www.britannica.com/science/analysis-mathematics (дата звернення: 16.05.2024).

2. Інтерполяція. *Byjus.* URL: https://byjus.com/maths/interpolation/ (дата звернення: 16.05.2024).

3. Reconstruction and representation of 3D objects with radial basis functions / J. C. Carr; R. K. Beatson, J. B. Cherrie, T. J. Mitchell; W. R. Fright, B. C. McCallum, T. R. Evans. URL: http://mesh.brown.edu/DGP/pdfs/Carr-sg2001.pdf (дата звернення: 16.05.2024).

4. Lazzaro D., Montefusco L. B. Radial basis functions for the multivariate interpolation of large scattered data sets. *Journal of Computational and Applied Mathematics.* 2002. № 140. P. 521–536. URL: https://core.ac.uk/download/pdf/82502771.pdf (дата звернення: 16.05.2024).

5. Runge's phenomenon. *Wolfram Demonstrations Project.* URL: https://demonstrations.wolfram.com/RungesPhenomenon/ (дата звернення: 16.05.2024).

УДК 004.421:519.61](043.2)

Левченко М. Р., *здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,*
Сеник І. О., *асистент кафедри інформаційних технологій*

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ РОЗВ'ЯЗАННЯ СИСТЕМИ ЛІНІЙНИХ РІВНЯНЬ. МЕТОД ГАУСА ТА СІМПСОНА

Донецький національний університет імені Василя Стуса, м. Вінниця

У цьому дослідженні проводиться порівняльний аналіз ефективності і точності різних методів розв'язання систем лінійних рівнянь, зокрема методу Гауса та методу Сімпсона. Результати дослідження дають змогу виявити переваги та недоліки кожного методу залежно від характеристик системи лінійних рівнянь, як-от розмір матриці, співвідношення між числом рівнянь та невідомих, а також структура матриці. Ця робота має на меті визначити найбільш підходящий метод для конкретних умов задачі, що є важливим кроком у покращенні ефективності обчислення та точності результатів у чисельних обчисленнях.

Метод Гауса – його суть полягає в тому, що шляхом елементарних перетворень СЛАР приводять до еквівалентної системи трикутного або трапецієвидного вигляду, з якої послідовно, починаючи з останніх (за номером) змінних, знаходять усі невідомі [1].

Метод Сімпсона – це чисельний метод інтегрування, який використовується для наближеного обчислення визначених інтегралів функцій. Основна ідея полягає в апроксимації площі під кривою за допомогою квадратичних апроксимаційних функцій, відомих як параболи [2].

Порівняємо основні переваги та недоліки двох методів.

Таблиця 1 – Переваги методів

Метод Гауса	**Метод Сімпсона**
Універсальність: може застосовуватися для розв'язання широкого спектра систем лінійних рівнянь з різними характеристиками	*Висока точність:* часто дає дуже точні результати, особливо під час обчислення великих інтегралів
Стійкість: має добру стійкість до помилок округлення, що робить його надійним для використання в чисельних обчисленнях	*Простота реалізації:* має просту інтуїтивну інтерпретацію, що робить його доступним для реалізації навіть без великого досвіду в чисельних обчисленнях
Ефективність: для деяких типів систем лінійних рівнянь може бути дуже ефективним і швидким	*Можливість адаптації:* може бути легко адаптований для обчислення невизначених інтегралів та інших специфічних випадків