ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

Кафедра програмування

Паралельні та розподілені обчислення

ЛАБОРАТОРНА РОБОТА №3

Тема: «Розв'язування систем лінійних алгебраїчних рівнянь»

Виконав:
Ст. *Лук'янчук Денис*
Група ПМІ-33

2025

Тема: «Розв'язування СЛАР»

Мета роботи: Послідовно та паралельно реалізувати розв’язання СЛАР довільним із існуючих методів.

Теоретичний матеріал

Система лінійних алгебраїчних рівнянь (СЛАР) — це сукупність лінійних рівнянь, що містять одну й ту саму групу невідомих. Метою розв’язання СЛАР є визначення значень цих невідомих, які задовольняють всі рівняння системи одночасно.

Загальний вигляд СЛАР з m рівнянь та n невідомих:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

де:

- x_1, x_2,…, x_n — невідомі, які потрібно знайти,
- a_ij — коефіцієнти при невідомих,
- b_i— вільні члени.

Якщо кількість рівнянь дорівнює кількості невідомих (m = n) і матриця коефіцієнтів є невиродженою (det(A) ≠ 0), то система має унікальне рішення.

Методи розв’язання СЛАР

Методи розв’язання СЛАР поділяються на аналітичні (точні) та чисельні (приближені).

Аналітичні методи

1. Метод підстановки
    - Виражаємо одну змінну через інші та підставляємо в інші рівняння,
    - Повторюємо, поки не знайдемо всі змінні.

2. Метод Гаусса (метод додавання)

    - Рівняння системи перетворюються так, щоб поступово вилучати змінні і отримати верхню трикутну матрицю,
    - Після цього виконується зворотний хід для знаходження всіх невідомих.

3. Метод Жордана – Гауса

    - Розширення методу Гаусса, де матриця перетворюється в одиничну (1 на головній діагоналі, 0 у всіх інших елементах),
    - Після цього значення всіх змінних видно одразу, без зворотного ходу.
    - Підходить для систем будь-якого розміру, але потребує більше обчислень.

4. Метод Крамера

    - Використовує визначники:

$$x_i = \frac{\det(A_i)}{\det(A)}$$

    де A_i — матриця, утворена заміною iii-го стовпця матриці коефіцієнтів на вектор вільних членів b.

    - Застосовується тільки для невеликих систем.

5. Метод оберненої матриці

    - Якщо матриця A невироджена:

$$x = A^{-1}b$$

    - Потребує обчислення оберненої матриці.

Чисельні методи

Використовуються для великих систем або розріджених матриць, коли точне рішення аналітичними методами важко або неефективно отримати.

1. Метод Гаусса з частковим або повним вибором головного елемента

    - Покращує точність прямого методу Гаусса, зменшуючи похибки округлення.

2. Метод прогонки (для тридіагональних матриць)

- Оптимізований алгоритм для систем, де ненульові елементи знаходяться лише на головній та суміжних діагоналях.
- Швидкий і ефективний для спеціальних структур матриць.

3. Ітераційні методи
   - Метод простих ітерацій (метод Якобі)
   - Метод Гаусса-Зейделя
   - Рішення знаходиться поступово шляхом повторних наближень, поки не досягнуто заданої точності.
   - Підходять для великих або розріджених систем.

Хід роботи

Під час виконання лабораторної роботи я реалізував розв’язання системи лінійних алгебраїчних рівнянь методом Гаусса двома способами: послідовним та паралельним. У програмі передбачено самостійне введення розміру матриці (кількість рядків та стовпців) та кількості потоків, на які будуть розподілені обчислення.

Якщо кількість рядків матриці не ділиться на кількість потоків, роботу неможливо розподілити рівномірно. У цьому випадку спочатку обчислюється базова кількість рядків на потік, а залишок рядків розподіляється між першим кільком потоками, по одному рядку на потік, щоб усі рядки були оброблені.

Для перевірки ефективності я спочатку розв’язав систему розміром 100×100 з використанням 4, 8 та 16 потоків:

```
Розмір системи: 100 x 100
Кількість потоків: 4

Послідовне розв'язання: 2 мс (0,00 с)
Паралельне розв'язання: 42 мс (0,04 с)

Прискорення: 0,05
Ефективність: 1,26%
```

```
Розмір системи: 100 x 100
Кількість потоків: 8

Послідовне розв'язання: 1 мс (0,00 с)
Паралельне розв'язання: 105 мс (0,11 с)

Прискорення: 0,02
Ефективність: 0,22%
```

```
Розмір системи: 100 x 100
Кількість потоків: 16

Послідовне розв'язання: 1 мс (0,00 с)
Паралельне розв'язання: 183 мс (0,18 с)

Прискорення: 0,01
Ефективність: 0,06%
```

При малих розмірах системи швидкість виконання послідовного є швидшим ніж у паралельного. Відповідно, прискорення також не спостерігається. Проте зі збільшенням кількості потоків ефективність роботи паралельного алгоритму зменшується через додаткові витрати на керування потоками.

Далі я продовжив збільшувати розмірність матриці до 500×500:

```
Розмір системи: 500 x 500
Кількість потоків: 4

Послідовне розв'язання: 203 мс (0,20 с)
Паралельне розв'язання: 302 мс (0,30 с)

Прискорення: 0,67
Ефективність: 16,87%
```

```
Розмір системи: 500 x 500
Кількість потоків: 8

Послідовне розв'язання: 179 мс (0,18 с)
Паралельне розв'язання: 416 мс (0,42 с)

Прискорення: 0,43
Ефективність: 5,38%
```

```
Розмір системи: 500 x 500
Кількість потоків: 16

Послідовне розв'язання: 181 мс (0,18 с)
Паралельне розв'язання: 734 мс (0,73 с)

Прискорення: 0,25
Ефективність: 1,54%
```

При збільшенні розміру системи паралельне множення стає працювати краще но розмірність 500 на 500 досі є замалою. Зі збільшенням кількості потоків швидкість обчислень та прискорення зменшуються, через великі затрати на створення потоків. Ефективність знижується по аналогічним причинам. Тому найефективнішим варіантом є із меншою кількістю потоків.

У наступному прикладі з системою розмірністю 1000×100 спостерігається, що при використанні 4 та 8 потоків паралельне розв’язання стало швидшим за послідовне. Однак при 16 потоках через витрати на створення та управління потоками паралельне виконання виявилося повільнішим за послідовне. Щодо ефективності та прискорення, ситуація аналогічна попередньому прикладу: найефективнішим та з найбільшим прискоренням залишається варіант з меншою кількістю потоків.

```
Розмір системи: 1000 x 1000
Кількість потоків: 4

Послідовне розв'язання: 1226 мс (1,23 с)
Паралельне розв'язання: 867 мс (0,87 с)

Прискорення: 1,41
Ефективність: 35,32%
```

```
Розмір системи: 1000 x 1000
Кількість потоків: 8

Послідовне розв'язання: 1375 мс (1,38 с)
Паралельне розв'язання: 1140 мс (1,14 с)

Прискорення: 1,21
Ефективність: 15,08%
```

```
Розмір системи: 1000 x 1000
Кількість потоків: 16

Послідовне розв'язання: 1371 мс (1,37 с)
Паралельне розв'язання: 1725 мс (1,73 с)

Прискорення: 0,79
Ефективність: 4,97%
```

Далі я збільшив розмірність системи до 2000×2000:

```
Розмір системи: 2000 x 2000
Кількість потоків: 4

Послідовне розв'язання: 10165 мс (10,17 с)
Паралельне розв'язання: 5215 мс (5,22 с)

Прискорення: 1,95
Ефективність: 48,73%
```

```
Розмір системи: 2000 x 2000
Кількість потоків: 8

Послідовне розв'язання: 10602 мс (10,60 с)
Паралельне розв'язання: 4860 мс (4,86 с)

Прискорення: 2,18
Ефективність: 27,27%
```

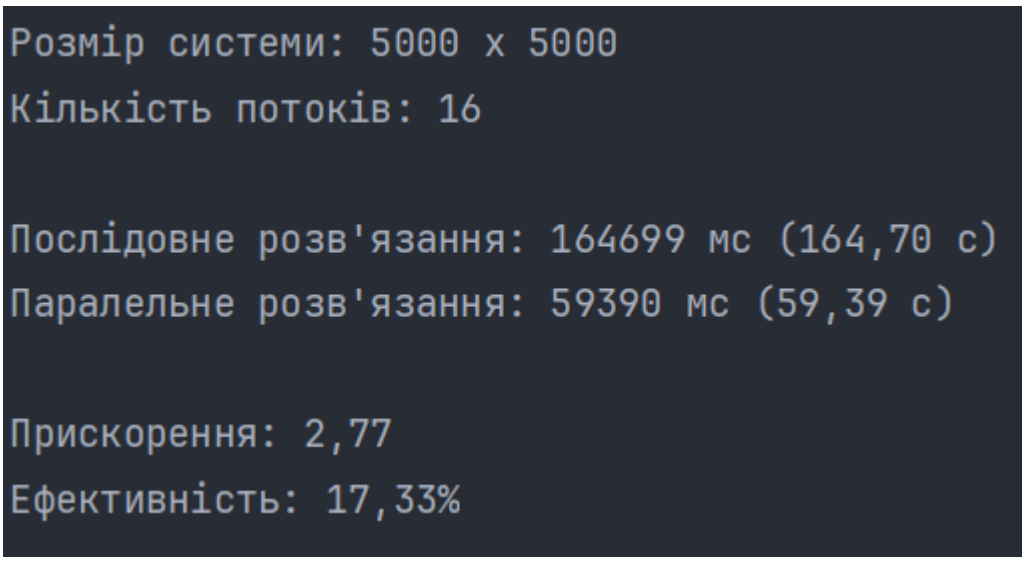
```
Розмір системи: 2000 x 2000
Кількість потоків: 16

Послідовне розв'язання: 10462 мс (10,46 с)
Паралельне розв'язання: 5679 мс (5,68 с)

Прискорення: 1,84
Ефективність: 11,51%
```

При даній розмірності системи паралельне розв’язання з використанням 8 потоків виявилось найшвидшим і дало найбільше прискорення, проте за ефективністю поступається варіанту з 4 потоками. При 16 потоках швидкість паралельного розв’язання є найнижчою серед усіх варіантів та ефективність і прискорення також є найменшими.

Для останнього прикладу взяв розмір 5000×5000

```
Розмір системи: 5000 x 5000
Кількість потоків: 4

Послідовне розв'язання: 167808 мс (167,81 с)
Паралельне розв'язання: 103530 мс (103,53 с)

Прискорення: 1,62
Ефективність: 40,52%
```

```
Розмір системи: 5000 x 5000
Кількість потоків: 8

Послідовне розв'язання: 170275 мс (170,28 с)
Паралельне розв'язання: 65102 мс (65,10 с)

Прискорення: 2,62
Ефективність: 32,69%
```



При розмірності системи 5000×5000 спостерігається, що найшвидшим за часом виконання та з найбільшим прискоренням серед паралельних розв’язків стає варіант із використанням найбільшої кількості потоків, проте за ефективністю найкращим залишається варіант з найменшою кількістю потоків.

Але якщо вказати кількість потоків більшу, ніж фізично доступно на комп’ютері, швидкість і прискорення вже не зростатимуть, а ефективність значно знизиться через додаткові накладні витрати на керування потоками.

Підсумок:

Під час роботи було реалізовано послідовне та паралельне розв’язання системи лінійних алгебраїчних рівнянь методом Гаусса з можливістю задавати розмір системи та кількість потоків.

При малих розмірах системи паралельне виконання майже не прискорює обчислення, а ефективність знижується при збільшенні потоків через накладні витрати на їх керування.

Зі збільшенням розмірності системи паралельне розв’язання стає швидшим за послідовне, і прискорення зростає, проте ефективність найкраща при меншій кількості потоків.

При дуже великих системах (наприклад 5000×5000) варіанти з більшою кількістю потоків показують найшвидший час і найбільше прискорення, але ефективність залишається вищою для варіантів з меншим числом потоків.

Таким чином, оптимальний вибір кількості потоків залежить від розміру системи: для малих систем – менше потоків, для великих – можна використовувати більше потоків для прискорення, хоча ефективність при цьому падає.

Висновок: У лабораторній роботі я навчився реалізовувати розв’язання СЛАР як послідовним, так і паралельним алгоритмами методом Гаусса. Було проведено експерименти з різними розмірами системи та кількістю потоків. Встановлено, що паралельний алгоритм суттєво прискорює розв’язання на великих системах, особливо якщо кількість потоків відповідає числу фізичних ядер процесора. При збільшенні кількості потоків ефективність зменшується через накладні витрати на керування потоками, а при перевищенні кількості фізичних потоків швидкість і прискорення перестають зростати. Отже, для оптимального використання паралельних обчислень важливо враховувати як розмір системи, так і кількість доступних апаратних потоків.