**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

Факультет прикладної математики та інформатики

Кафедра програмування

Звіт про виконання

лабораторної роботи №3

**Розпаралелення розв’язування систем лінійних алгебраїчних рівнянь**

з курсу “Паралельні та розподілені обчислення”

Виконав:

Студент групи ПМО-21с

Заяц Ростислав Васильович

Перевірив:

Гошко Богдан Мирославович

**Львів – 2025**

**Тема:** Написання програми розв’язування системи алгебраїчних рівнянь послідовним та паралельним алгоритмом.

**Мета:** Розвинути свої практичні навички використання паралельних алгоритмів програмування. Ширше ознайомитись з паралельним програмуванням.

**Теоретична частина**

**Прискорення (Speedup)**

Визначення:  
 Прискорення — це відношення часу виконання послідовного алгоритму до часу виконання паралельного алгоритму. Воно показує, наскільки швидше працює паралельна версія в порівнянні з послідовною.

Формула:

**Speedup = T\_посл / T\_парал**

Де:  
- T\_посл — час виконання послідовного алгоритму;  
- T\_парал — час виконання паралельного алгоритму.

**Ефективність (Efficiency)**

Визначення:  
 Ефективність показує, наскільки ефективно використовуються потоки (чи процесори) при паралельному виконанні. Це відношення прискорення до кількості потоків.

Формула:

**Efficiency = (Speedup / k) \* 100%**

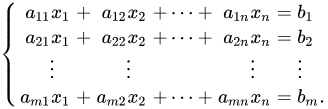
Де:  
- Speedup — прискорення;  
- k — кількість потоків (процесорів).

**СЛАР** - Це набір рівнянь, де невідомі стоять у першому степені.

**Метод Якобі** - це **ітераційний метод** розв’язання **СЛАР**.

**Хід роботи.**

Для початку написання коду я вирішив детальніше зрозуміти архітектуру будови СЛАР.



З даної ілюстрації бачимо що для реалізації СЛАР в програмуванні нам потрібна певна квадратна матриця коефіцієнтів при невідомих, а також вектор розв’язків окремих рівнянь. Також нам потрібен певний вектор початкових приближень, для реалізації ітераційного методу Якобі. Використаємо повторно готовий клас **Matrix,** написаний мною на попередніх лабораторних заняттях.

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Конструктор приймає готову матрицю коефіцієнтів, та вектор розв’язків за адресою.

Послідовний метод Якобі:

Зображення, що містить текст, знімок екрана, програмне забезпечення, Шрифт

Автоматично згенерований опис

Метод Якобі базується на покроковому оновленні значень змінних, де на кожній ітерації нове значення змінної обчислюється з використанням всіх значень попередньої ітерації.

Спочатку ініціалізуємо x\_new та x\_old значенням поточного наближення.

На кожній ітерації обчислюємо значення x\_i за формулою Якобі:

Зображення, що містить Шрифт, Вимірювальний прилад, датчик

Автоматично згенерований опис

Обчислює max\_diff між старими та новими значеннями. Зупиняється якщо досягнуто ліміт ітерацій, або якщо система збігається(max\_diff < EPSILON, EPSILON = 1e-6)

Метод Якобі гарантовано збігається, якщо матриця **A** задовольняє одну з умов:

Діагональна перевага

Симетрична та додатньо визначена матриця **A**

Паралельний метод Якобі:

Зображення, що містить текст, знімок екрана, програмне забезпечення

Автоматично згенерований опис

У методі Якобі кожне нове значення залежить лише від старих значень), тому обчислення можна виконувати незалежно.

Метод **parallelJacobi** приймає систему рівнянь і кількість потоків, ініціалізує x\_old та x\_new значенням поточного наближення, а також створює вектор потоків threads, розбиває матрицю коефіцієнтів по рядках, розділяючи кількість рядків на кількість потоків, якщо число не кратне, то залишок обробляє останній потік. Потім в вкладеному циклі кожен потік отримує команду виконувати функію JacobiWorker, вона приймає початковий, кінцевий рядки, СЛАР по адресі, а також x\_old та x\_new, і попередньо створений max\_diff. По факту потоки створюються і видаляються допоки не буде досягнуто ліміту ітерацій, або допоки система не збіжиться. Потім замінюємо старі данні оновленими і продовжуємо до збіжності системи.

Зображення, що містить текст, знімок екрана, програмне забезпечення, Шрифт

Автоматично згенерований опис

jacobiWorker це вже наш паралельний алгоритм обчислення, який майже ідентичний послідовному. Тут наші дії ідентичні як в послідовному алгоритмі, проте, тут використано local\_guard<mutex> - ця команда надає доступ лише 1 потоку який на даний момент часу працює з вектором max\_diff, це для того щоб наші данні не псувались і хаотично не змінювались.

**Висновок:** Виконавши цю лабораторну роботу, я розширив свої знання про паралельні обчислення в С++, бібліотекою mutex. Також дослідив, що розпаралелення методу Якобі дає хороший приріст до швидкості виконання, проте ефективність алгоритму дуже низька. Також алгоритм працює краще для матриць коефіцієнтів розміром >= [1000]x[1000]. Найбільш оптимальним вважаю використання 4 потоків адже співвідношення прискорення\ефективності найбільш приємне серед інших варіантів

**Результати**

**Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис**

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований описЗображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

**Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис**