Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

Звіт
з лабораторної роботи №4
з дисципліни «Теорія паралельних обчислень»
на тему «Паралельний алгоритм Гауса-Зейделя»

Виконали ст. гр. ІПЗм-22-6:
Миронюк С.А.,
Сєнічкін І.О.

Перевірив викладач:
доц. Кобзєв В.Г.

Харків, 2023

ПАРАЛЕЛЬНИЙ АЛГОРИТМ ГАУСА-ЗЕЙДЕЛЯ

Мета роботи – навчитися створювати і аналізувати ітераційні паралельні алгоритми розв'язування систем лінійних рівнянь з щільними і розрідженими матрицями та оцінювати показники їх прискорення у порівнянні з послідовними алгоритмами.

Індивідуальне завдання:

Система лінійних алгебраїчних рівнянь (СЛАР) задана матрицею коефіцієнтів A та вектором вільних членів B, які відповідають завданню до лабораторної роботи 1.

1. Розробити послідовний та паралельний алгоритми розв'язку СЛАР:
   а) методом Якобі;
   б) методом Гауса-Зейделя.
2. Реалізувати алгоритми програмно на заповнених та розріджених матрицях.
3. Обчислити значення показників прискорення та ефективності розпаралелювання для двох варіантів розміру матриць.

Обрано варіант: а) методом Якобі.

Дані для лабораторної роботи: значення синіх пікселів зображення «Вхід до ХНУРЕ». Рядок – починаючи с 60-го пікселя, строка – з 1-го. Розміри матриць; 1000×1000, 5000×50000, 9000×9000. Стовпчик вільних членів – 6-ій з кінця рисунку 1. У випадку недостатньої інформації на фотографії, нам потрібно додати ще одну копію цієї фотографії. Зазначена фотографія відображена на рисунку 1.

Рисунок 1 – Зображення для виконання завдання

Хід роботи:

Метод Якобі – це числовий метод для розв'язання систем лінійних рівнянь, особливо у випадку великих систем. Метод полягає в ітеративному покращенні наближеного розв'язку шляхом оновлення кожної змінної залежно від її поточного значення та інших змінних.

Алгоритм методу Якобі:

1) Початкове наближення – задається початкове наближення для всіх змінних системи рівнянь.

2) Ітерації – повторюються наступні кроки до досягнення задовільного розв'язку або заданої кількості ітерацій:

- для кожної змінної обчислюється нове значення, використовуючи поточні значення інших змінних та коефіцієнти системи рівнянь.
- оновлюються значення змінної.

3) Повертаються останні обчислені значення змінних як наближений розв'язок системи.

Для квадратної системи з n лінійних рівнянь:

$$Ax = b$$

де:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \; x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ x_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Матрицю A можна розкласти на два доданки: діагональну матрицю D, та все інше R:

$$A = D + R, \qquad D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \qquad R = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}.$$

Систему лінійних рівнянь можна переписати в вигляді:

$$Dx = b - Rx$$

Ітераційний метод Якобі виражається формулою:

$$x^{(k+1)} = D^{-1}(b - Rx^{(k)}).$$

або

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij} x_j^{(x)}\right), i = 1, 2, \ldots, n.$$

де $x_i^{(k+1)}$ – нова апроксимація і-го компонента вектору розв'язку на *(k+1)*-й ітерації,
$x_j^{(k)}$ – *j*-й компонент вектору розв'язку на *k*-й ітерації,
a_{ij} – елемент матриці A в і-тому рядку і j-тому стовпці,
b_i – *i*-й компонент вектора правої частини,
n – розмірність системи (кількість рівнянь).

Дана програма була виконана на пристрої DESKTOP-IDO8GRU з процесором Intel(R) Core(TM) i5-7600 CPU, який включає 4 ядра, базовая тактовая частота 3500 МГц, объем кэш памяти 3 уровня (L1 – 256 КБ, L2 – 1,0 МБ, L3 – 6 МБ).

Програмна реалізація алгоритму розв’язання СЛАР методом Якобі на мові програмування Java.

Код застосовується для демонстрації та порівняння продуктивності послідовного та паралельного методів розв'язання системи лінійних рівнянь для різних розмірів матриць.

Послідовна реалізація алгоритму зображена на рисунку 2.

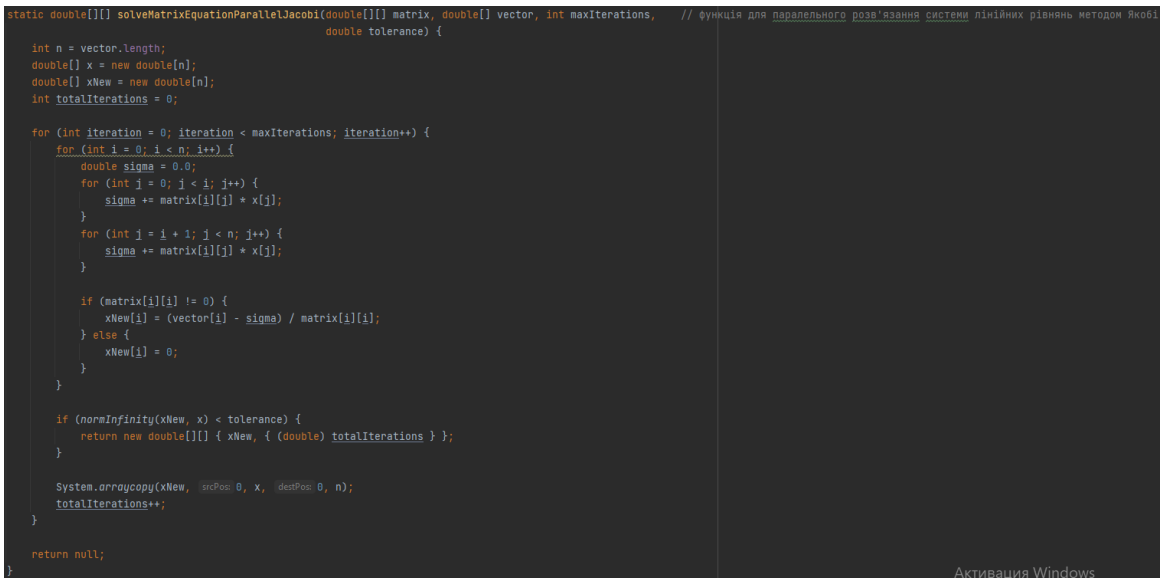
```
1 usage
static double[] solveMatrixEquationJacobi(double[][] matrix, double[] vector, int maxIterations, double tolerance) {
    int n = vector.length;
    double[] x = new double[n];
    double[] xNew = new double[n];

    for (int iteration = 0; iteration < maxIterations; iteration++) {
        for (int i = 0; i < n; i++) {
            double sigma = 0.0;
            for (int j = 0; j < i; j++) {
                sigma += matrix[i][j] * x[j];
            }
            for (int j = i + 1; j < n; j++) {
                sigma += matrix[i][j] * x[j];
            }

            if (matrix[i][i] != 0) {
                xNew[i] = (vector[i] - sigma) / matrix[i][i];
            } else {
                xNew[i] = 0;
            }
        }

        if (normInfinity(xNew, x) < tolerance) {
            return xNew;
        }

        System.arraycopy(xNew,  srcPos: 0, x,  destPos: 0, n);
    }

    return null;
}
```

```
// функція для послідовного розв'язання системи лінійних рівнянь методом Якобі
```

Рисунок 2 – Послідовна реалізація алгоритму Якобі

В даному коді використовується ExecutorService для керування пулом потоків та Future для отримання результатів завдань. Розпаралелювання відбувається навколо обробки різних матриць і векторів, що дозволяє прискорити обчислення великих даних.

Метод solveMatrixEquationParallelJacobi вирішує систему лінійних рівнянь методом Якобі паралельно (рис. 3). Розподіл ітерацій між потоками дозволяє прискорити процес розв'язання системи.



```
static double[][] solveMatrixEquationParallelJacobi(double[][] matrix, double[] vector, int maxIterations,    // функція для паралельного розв'язання системи лінійних рівнянь методом Якобі
                                                    double tolerance) {
    int n = vector.length;
    double[] x = new double[n];
    double[] xNew = new double[n];
    int totalIterations = 0;

    for (int iteration = 0; iteration < maxIterations; iteration++) {
        for (int i = 0; i < n; i++) {
            double sigma = 0.0;
            for (int j = 0; j < i; j++) {
                sigma += matrix[i][j] * x[j];
            }
            for (int j = i + 1; j < n; j++) {
                sigma += matrix[i][j] * x[j];
            }

            if (matrix[i][i] != 0) {
                xNew[i] = (vector[i] - sigma) / matrix[i][i];
            } else {
                xNew[i] = 0;
            }
        }

        if (normInfinity(xNew, x) < tolerance) {
            return new double[][] { xNew, { (double) totalIterations } };
        }

        System.arraycopy(xNew, srcPos: 0, x, destPos: 0, n);
        totalIterations++;
    }

    return null;
}
```


Рисунок 3 – Паралельна реалізація алгоритму Якобі

Результати роботи програми зображені на рисунку 4.

```
Розмір матриці 1000:
Час виконання послідовного методу: 127 мілісекунд
Час виконання паралельного методу: 111 мілісекунд
Кількість ітерацій: 98.0
Розмір матриці 5000:
Час виконання послідовного методу: 2216 мілісекунд
Час виконання паралельного методу: 2190 мілісекунд
Кількість ітерацій: 82.0
Розмір матриці 9000:
Час виконання послідовного методу: 6643 мілісекунд
Час виконання паралельного методу: 6635 мілісекунд
Кількість ітерацій: 77.0

Process finished with exit code 0
```

Рисунок 4 – Результати роботи програми

Таблиця 1 – Результати виконання алгоритмів розв’язання СЛАУ методом Якобі

Розмір матриці	Послідовний метод	Паралельний метод	Кількість ітерацій	Прискорення, $S = \frac{T_{sequential}}{T_{parallel}}$	Ефективність, $E = \frac{S}{число\ потоків}$
	Час виконання послідовного методу, мс	Час виконання паралельного методу, мс			
1000×1000	127	111	98	1,144	0,286
5000×5000	2216	2190	82	1,012	0,253
9000×9000	6643	6635	77	1,001	0,250

Графік порівняння часу виконання паралельного і послідовного алгоритмів для різних розмірів матриць зображено на рисунку 5. Ось X дорівнювати розмірності матриці B помноженої на кількість ітерацій.

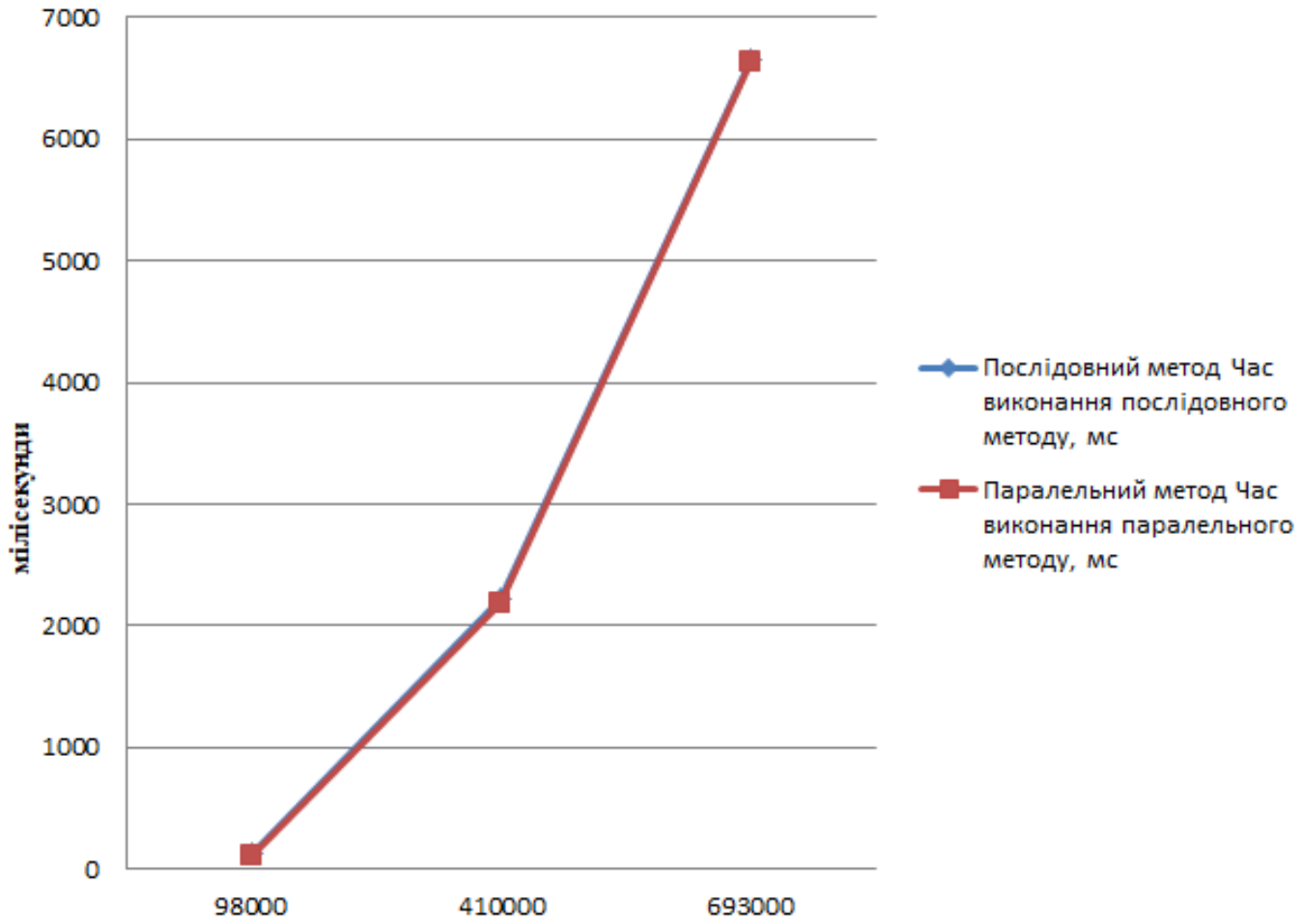


Рисунок 5 – Графік порівняння часу виконання паралельного і послідовного алгоритмів

Ці результати показують, що суттєва різниці у часі виконання між паралельним та послідовним методами відсутня. Тобто, паралельний підхід трохи перевищує послідовний. Якщо розмір матриці відносно невеликий, накладні витрати на розпаралелювання та керування потоками можуть переважити переваги паралельної обробки. Розпаралелювання ефективніше для великих обчислювальних завдань.

У нашому випадку код працює з матрицями щодо невеликого розміру, і переваги розпаралелювання можуть стати очевиднішими при використанні великих наборів даних.

Висновки: Під час виконання лабораторної роботи було реалізовано метод Якобі послідовно та паралельно. В результаті було встановлено, що паралельний алгоритм проявляє більшу ефективність при обробці великих

матриць. Також при реалізації метода Якобі величина прискорення не лише залежить від розміру матриці, але й від кількості ітерацій.

Відповіді на контрольні запитання:

3. Назвіть умови збіжності методу Гауса-Зейделя.

Метод Гауса-Зейделя збігається, якщо матриця системи має діагональне преобладання або строгу діагональну домінантність. Це означає, що модуль кожного елемента на головній діагоналі повинен перевищувати суму модулів інших елементів у цьому рівнянні:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

Для симетричних матриць метод Гауса-Зейделя збігається, якщо матриця є додатно визначеною (всі власні числа додатні). У випадку самоспряжених матриць у комплексних системах рівнянь метод також може збігатися при умові самоспряженості матриці.

Щодо обмеження на спектральний радіус, він повинен бути менший за одиницю для забезпечення збіжності. Спектральний радіус обчислюється як максимальне власне число матриці за допомогою ітераційного процесу.

Зазвичай, наявність діагонального преобладання або строгої діагональної домінантності є достатньою умовою для збіжності методу Гауса-Зейделя; інші умови можуть вважатися додатковими обмеженнями в конкретних випадках.

4. Що називається нормою вектора, які існують різновиди норми вектора?

Норма вектора – це числова характеристика вектора, яка вимірює його «розмір» чи «довжину».

Нормою вектора $\vec{u}$ називають дійсне число $\|\vec{u}\|$, яке задовольняє умови:

a) $\|\vec{0}\| = 0$;

b) $\|\vec{u}\| > 0$, якщо $\vec{u} \neq \vec{0}$;

c) $\|c\vec{u}\| = |c|\|\vec{u}\|$ для будь-якого числа c;

d) $\|\vec{u} + \vec{v}\| \leq \|\vec{u}\| + \|\vec{v}\|$

Часто використовуються такі норми вектора.

1. *Кубічна* $\|\vec{u}\|_1 = \max_i |u_i|$.

2. *Октаедрична* $\|\vec{u}\|_2 = \sum_{i=1}^{n} |u_i|$.

3. *Сферична* $\|\vec{u}\|_3 = \sqrt{\sum_{i=1}^{n} |u_i|^2}$.

Тут назви норм пов'язані з геометричним тлумаченням нерівності $\|\vec{u}\| \leq 1$.

7. У чому полягає метод Гауса-Зейделя, які умови його збіжності, яким чином його можна розпаралелити?

Метод Гауса-Зейделя – це ітераційний метод для вирішення систем лінійних рівнянь. Він застосовується до систем рівнянь виду Ax = b, де A – квадратна матриця коефіцієнтів, x – вектор невідомих, і b – вектор правих частин. Метод ґрунтується на ідеї послідовного уточнення значень невідомих у процесі ітерацій.

Принцип роботи методу Гауса-Зейделя:

1) Система рівнянь представляється як x = Cx + d, де C – матриця коефіцієнтів після поділу A на нижньотрикутну і верхньотрикутну частини, а d – вектор вільних членів.

2) Значення невідомих ітеративно уточнюються з використанням наступної формули:

$$x_i^{(k+1)} = \frac{1}{C_{ii}} \left(\partial_i - \sum_{j=1}^{i-1} C_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} C_{ij} x_j^{(k)} \right).$$

3) Процес ітерацій повторюється до досягнення заданої точності чи певної кількості ітерацій.

Умови збіжності методу Гауса-Зейделя:

- Діагональне переважання: Метод сходиться, якщо кожен елемент головної діагоналі матриці A по модулю більше суми абсолютних значень інших елементів у відповідному рядку.
- Симетричність та позитивна визначеність: Якщо матриця A симетрична та позитивно визначена, то метод Гауса-Зейделя сходиться.

Метод Гауса-Зейделя можна паралелізувати шляхом розподілу обчислень між кількома потоками чи процесами. Основна ідея у тому, щоб

розділити ітерації між різними вузлами обчислювального кластера. При цьому необхідно дотримуватись залежності між ітераціями, щоб уникнути конфліктів при оновленні значень невідомих.

Кожен потік/процес може бути відповідальним за оновлення частини вектора невідомих. Однак, через залежність між значеннями на різних ітераціях, паралелізація методу Гауса-Зейделя може вимагати введення синхронізації або інших методів керування доступом до даних для уникнення гонок даних та забезпечення коректності результату.