Міністерство освіти і науки України

Львівський національний університет Ім. І. Франка

Факультет прикладної математики та інформатики

Кафедра дискретного аналізу та інтелектуальних систем

**Паралельні та розподілені обчислення**

**Лабораторна робота №2**

**Розв’язування СЛАР методом Жордана-Гауса**

Роботу виконала:

Студентка ПМІ-33

Багінська Маргарита

Прийняв:

доц. Пасічник Т.В.

Львів 2023

**Тема**: Розв’язування СЛАР методом Жордана-Гауса.

**Мета**: Реалізувати послідовний та паралельний алгоритм розв’язування СЛАР методом Жордана-Гауса.

**Послідовний алгоритм**

public static TimeSpan SyncMethod(double[,] matrix)

{

int N = matrix.GetLength(0);

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

// Зводимо матрицю до верхньої трикутної

for (int rowNum = 0; rowNum < N; rowNum++)

{

for (int i = rowNum + 1; i < N; i++)

{

double factor = matrix[i, rowNum] / matrix[rowNum, rowNum];

for (int j = rowNum; j <= N; j++)

{

matrix[i, j] -= factor \* matrix[rowNum, j];

}

}

}

CheckZerosOnDiagonal(matrix);

double[] result = new double[N];

// Обернений хід методу Жордана-Гаусса для знаходження розв'язку

for (int rowNum = N - 1; rowNum >= 0; rowNum--)

{

result[rowNum] = matrix[rowNum, N];

for (int j = rowNum + 1; j < N; j++)

{

result[rowNum] -= matrix[rowNum, j] \* result[j];

}

result[rowNum] /= matrix[rowNum, rowNum];

}

stopWatch.Stop();

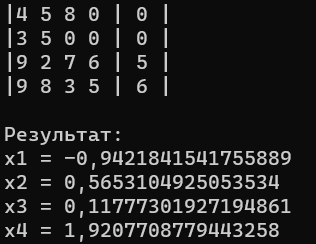
Console.WriteLine("Sync time ~ " + stopWatch.Elapsed.ToString());

return stopWatch.Elapsed;

}

Виконуємо метод Жордана-Гаусса для знаходження розв'язку СЛАР на одному потоці. Почергово виконуємо прямий хід методу для зведення матриці до діагональної форми і обернений хід для знаходження розв'язку.

Переконаємось у правильності обчислень:



**Паралельний алгоритм**

public static TimeSpan AsyncMethod(double[,] matrix, int numThreads)

{

int N = matrix.GetLength(0);

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

List<Thread> threads = new List<Thread>();

int rowsPerThread = N / numThreads;

for (int i = 0; i < numThreads; i++)

{

int startRow = i \* rowsPerThread;

int endRow = (i == numThreads - 1) ? N : startRow + rowsPerThread;

Thread thread = new Thread((object range) =>

{

int[] rangeArray = (int[])range;

int startRange = rangeArray[0];

int endRange = rangeArray[1];

for (int rowNum = startRange; rowNum < endRange; rowNum++)

{

for (int i = rowNum + 1; i < N; i++)

{

double factor = matrix[i, rowNum] / matrix[rowNum, rowNum];

for (int j = rowNum; j <= N; j++)

{

matrix[i, j] -= factor \* matrix[rowNum, j];

}

}

}

});

threads.Add(thread);

thread.Start(new int[] { startRow, endRow });

}

foreach (var thread in threads)

{

thread.Join();

}

CheckZerosOnDiagonal(matrix);

// Отримуємо результат (на жаль вже одним потоком)

double[] result = new double[N];

for (int rowNum = N - 1; rowNum >= 0; rowNum--)

{

result[rowNum] = matrix[rowNum, N];

for (int j = rowNum + 1; j < N; j++)

{

result[rowNum] -= matrix[rowNum, j] \* result[j];

}

result[rowNum] /= matrix[rowNum, rowNum];

}

stopWatch.Stop();

Console.WriteLine($"Async time ~ {stopWatch.Elapsed} with {numThreads} threads");

return stopWatch.Elapsed;

}

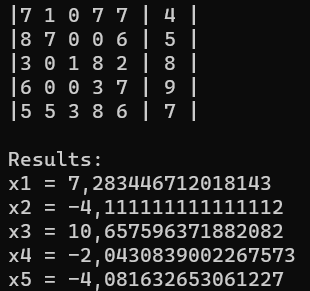
Ця функція виконує асинхронний алгоритм розв'язування системи лінійних алгебраїчних рівнянь (СЛАР) методом Гаусса з використанням паралельних потоків.

Запускаємо цикл for, в якому для кожного потоку визначаємо діапазон рядків, який цей потік буде обчислювати. Кожен потік виконує частину елімінації Гаусса для свого діапазону рядків.

Після завершення елімінації Гаусса обчислюємо розв'язок СЛАР за допомогою зворотного ходу Гауса. Ця частина, на жаль, виконується одним потоком, оскільки зворотній хід Гауса є послідовним процесом.

Ця функція демонструє спробу паралельного обчислення СЛАР методом Жордана-Гаусса, але зворотній хід виконується послідовно, оскільки він є залежним від результатів попереднього етапу.

Переконаємось у коректності роботи:



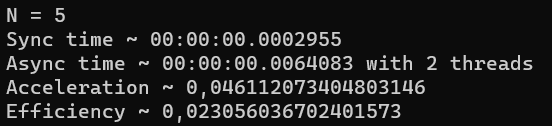
**Прискорення** Sp для паралельного алгоритму визначається відношенням часової складності послідовного T1 та паралельного алгоритмів для p процесорів Sp = T1 / Tp. (Sp > 1 Оптимально).

**Ефективність** Ep для паралельного алгоритму визначається прискоренням цього алгоритму відносно кількості процесорів: Ep= Sp/p Ідеал: Ep(n) = 1.

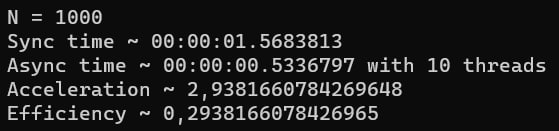
**Результати**

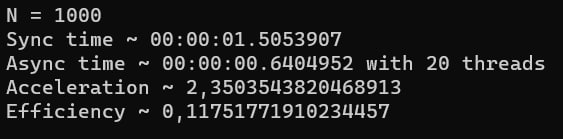
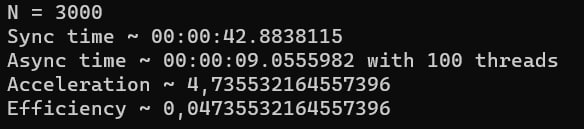
Не дивлячись на те, що зворотній хід Гауса виконується синхронно в обох варіантах, розпаралелення процесу зведення матриці до трикутного вигляду дало нам значну перевагу в часі.

На малій розмірності матриці розпаралелення не є оптимальним, як і у попередніх лабораторних роботах.

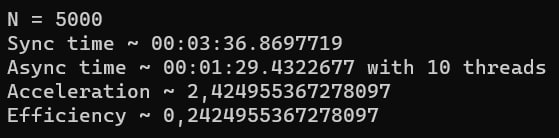


Зі збільшенням розмірності паралельність ефективніша.



Можемо спостерігати ефективність паралельного алгоритму при великих об’ємах та розумній кількості потоків.



У програмі також передбачене можливе виключення, коли після зведення матриці до верхньої трикутної, ми отримали 0 на основній діагоналі. В такому випадку можливе ділення на 0 в подальшому процесі розв’язування.



**Висновок:** У результаті виконання лабораторної роботи було реалізовано послідовний та паралельний алгоритм розв’язування СЛАР методом Жордана-Гауса мовою програмування C# та класу Thread. Переконались у ефективності розпаралелення процесу у даному випадку.