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.

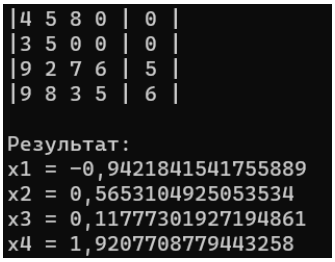
Мета: Реалізувати послідовний та паралельний алгоритм розв’язування СЛАР методом Жордана-Гауса.

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:

```
|7 1 0 7 7 | 4 |
|8 7 0 0 6 | 5 |
|3 0 1 8 2 | 8 |
|6 0 0 3 7 | 9 |
|5 5 3 8 6 | 7 |

Results:
x1 = 7,2834467120181 43
x2 = -4,111111111111112
x3 = 10,657596371882082
x4 = -2,0430839002267573
x5 = -4,081632653061227
```

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.

```
N = 5
Sync time ~ 00:00:00.0002955
Async time ~ 00:00:00.0064083 with 2 threads
Acceleration ~ 0,046112073404803146
Efficiency ~ 0,023056036702401573
```

Зі збільшенням розмірності паралельність ефективніша.

```
N = 1000
Sync time ~ 00:00:01.5683813
Async time ~ 00:00:00.5336797 with 10 threads
Acceleration ~ 2,9381660784269648
Efficiency ~ 0,2938166078426965
```

```
N = 1000
Sync time ~ 00:00:01.5053907
Async time ~ 00:00:00.6404952 with 20 threads
Acceleration ~ 2,3503543820468913
Efficiency ~ 0,117517719102344 57
```

```
N = 3000
Sync time ~ 00:00:42.8838115
Async time ~ 00:00:09.0555982 with 100 threads
Acceleration ~ 4,7355321645573 96
Efficiency ~ 0,047355321645573 96
```

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

```
N = 5000
Sync time ~ 00:03:36.8697719
Async time ~ 00:01:29.4322677 with 10 threads
Acceleration ~ 2,424955367278097
Efficiency ~ 0,2424955367278097
```

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.

```
N = 3000
Attempted to divide by zero.
```

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.