

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет прикладної математики та інформатики

Звіт

Лабораторна робота №3

Тема: «Розв’язування системи лінійних алгебраїчних рівнянь»
з дисципліни "Паралельні та розподільні обчислення"

Виконав студент групи ПМі-31
Яцуляк Андрій

Львів 2023 р.

Мета: Написати програми обчислення системи лінійних алгебраїчних рівнянь (послідовний та паралельний алгоритми).

Хід роботи

Завдання виконав мовою програмування Java у середовищі IntelliJ IDEA. Написав програму для роботи з системами, зокрема розв'язування методом Крамера:

```
public class Slar
{
    25 usages
    private final int dimension;
    15 usages
    private final int[][] matrix;
    2 usages
    private static int theardsNumber;
    1 usage
    public Slar(int dimension)
    {...}
    no usages
    public Slar(int[][] matrix)
    {...}
    1 usage
    public void fillMatrix()
    {...}
    no usages
    public void fillMatrix(int[][] coef, int[] terms)
    {...}
    no usages
    public void printSystem(){...}
    2 usages
    private void swapColumns(int[][] matrix, int col1, int col2)
    {...}
    1 usage
    public static int getTheardsNumber() { return Slar.theardsNumber; }
```

```

1 usage
public static void setTheardsNumber(int theardsNumber) { Slar.theardsNumber = theardsNumber; }
5 usages
public int calculateDeterminant(int[][] matrix)
{...}
// Метод для знаходження мінора (підматриці без рядка і стовпця)
1 usage
private int[][] subMatrix(int[][] matrix, int row, int col)
{...}
// Метод для обчислення коефіцієнта сполученого з мінором
1 usage
private int cofactor(int[][] matrix, int row, int col)
{...}
1 usage
public double[] Result()
{...}
1 usage
public double[] ParalelResult()
{...}
}

```

Мій метод для обчислення детермінанту працює рекурсивно, тому складність такої операції буде $n!$. Тому малої кількості змінних буде достатньо. Далі методом Fill я заповнюю рандомно мою систему значеннями від -9 до 9. Далі вводячи з консолі кількість потоків, обчислюю розв'язок послідовним та паралельним методами Result та ParalelResult відповідно.

no usages

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the number of dimension: ");
    int dimension = scanner.nextInt();

    System.out.print("Enter the number of threads: ");
    int threadsNumber = scanner.nextInt();

    Slar slar = new Slar(dimension);
    slar.Fill();
    Slar.setTheardsNumber(threadsNumber);
    long startTime, endTime, durationSeq, durationPar;
    double speedUp, efficiency;
    double[] sequence;
    double[] paralel;
    startTime = System.nanoTime();
    sequence = slar.Result();
    endTime = System.nanoTime();
    durationSeq = (endTime - startTime);
    System.out.println("Sequential time: " + durationSeq + " nanoseconds");
    startTime = System.nanoTime();
    paralel = slar.ParalelResult();
    endTime = System.nanoTime();
    durationPar = (endTime - startTime);
    System.out.println("Paralel time: " + durationSeq + " nanoseconds");
    speedUp = (double) durationSeq / durationPar;
    efficiency = speedUp / threadsNumber;
}
```

```

Slar slar = new Slar(dimension);
slar.Fill();
slar.setTheardsNumber(threadsNumber);
long startTime, endTime, durationSeq, durationPar;
double speedUp, efficiency;
double[] sequence;
double[] paralel;
startTime = System.nanoTime();
sequence = slar.Result();
endTime = System.nanoTime();
durationSeq = (endTime - startTime);
System.out.println("Sequential time: " + durationSeq + " nanoseconds");
startTime = System.nanoTime();
paralel = slar.ParalelResult();
endTime = System.nanoTime();
durationPar = (endTime - startTime);
System.out.println("Paralel time: " + durationSeq + " nanoseconds");
speedUp = (double) durationSeq / durationPar;
efficiency = speedUp / threadsNumber;
System.out.println("Speed up: " + speedUp);
System.out.println("Efficiency: " + efficiency);
}
}

```

Далі наведено результати обчислень:

```

Enter the number of dimension: 10
Enter the number of threads: 2
Sequential time: 6391783400 nanoseconds
Paralel time: 6391783400 nanoseconds
Speed up: 1.8201616501839446
Efficiency: 0.9100808250919723

```

```

Enter the number of dimension: 10
Enter the number of threads: 3
Sequential time: 6376214600 nanoseconds
Paralel time: 6376214600 nanoseconds
Speed up: 2.2059472356287513
Efficiency: 0.7353157452095838

```

```
Enter the number of dimension: 10
Enter the number of threads: 4
Sequential time: 6395444700 nanoseconds
Parallel time: 6395444700 nanoseconds
Speed up: 2.671636177111439
Efficiency: 0.6679090442778598
```

```
Enter the number of dimension: 10
Enter the number of threads: 5
Sequential time: 6369088900 nanoseconds
Parallel time: 6369088900 nanoseconds
Speed up: 3.377736808979001
Efficiency: 0.6755473617958002
```

```
Enter the number of dimension: 11
Enter the number of threads: 4
Sequential time: 77516919701 nanoseconds
Parallel time: 77516919701 nanoseconds
Speed up: 2.8693950001185673
Efficiency: 0.7173487500296418
```

```
Enter the number of dimension: 9
Enter the number of threads: 5
Sequential time: 609759300 nanoseconds
Parallel time: 609759300 nanoseconds
Speed up: 3.092973423949402
Efficiency: 0.6185946847898804
```

При даних обчисленнях найкраще прискорення при кількості невідомих 10, і кількості потоків 5. А найкраща ефективність при кількості невідомих 10, і кількості потоків 2.

Висновок. Під час виконання лабораторної роботи я написав програму для роботи з системами алгебраїчних рівнянь, зокрема метод Крамера, обчислив прискорення та ефективність для різної кількості потоків.