Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота ЛР1.4

Потоки в мові C#

з дисципліни

Програмне забезпечення високопродуктивних комп'ютерних систем

ВИКОНАЛА:
Студентка групи ІМ-21
Рабійчук Дар’я Олександрівна
№ у списку(варіант) - 18

ПЕРЕВІРИВ:
доц. Корочкін О. В.

Київ 2025

Завдання

Лабораторна робота ЛР1 виконується кожним студентом. Для виконання ЛР1 необхідно отримати варіант завдання для ЛР1, який включає номера трьох математичних функцій F1, F2, F3 з Додатку Б (1.x, 2.x, 3.x). Функції пов’язані з виконанням операцій над векторами і матрицями.

Варіант 18:

F1: 1.17 d = (A * ((B + C)*(MA*ME)))
F2: 2.21 MF = MF = MG + (MH*MK) +ML
F3: 3.25 S = (O + P + V)*(MR * MS)

Виконання ЛР

Вибір мови

Для виконання лабораторної роботи було обрано мову C#, оскільки вона забезпечує потужну підтримку паралельного програмування через вбудовані засоби (TPL, потоки), що дозволяє ефективно реалізовувати багатопотокові алгоритми.

Структура програми

Program.cs — основний файл програми, який відповідає за створення і запуск потоків. Кожен потік виконує свою конкретну функцію.

Data.cs — файл, що містить клас Data з різними методами для введення даних та виконання обчислень над ними.

Робота програми

Програма починає з ініціалізації даних (введення N, заповнення векторів та матриць). Потім створюються та запускаються три паралельні потоки, кожен із яких обчислює свою функцію (F1, F2, F3) за заданими варіантом формулами. Після завершення обчислень результати виводяться на екран. При малих значеннях (N до 4) дані вводяться вручну, а при великих (N від 1000) заповнення відбувається випадковим чином.

Лістинг програми

Program.cs

```
// ПЗ ВПКС
// ЛР1.4 ПОТОКИ В МОВІ C#
// F1: 1.17 d = (A * ((B + C)*(MA*ME)))
// F2: 2.21 MF = MF = MG + (MH*MK) +ML
// F3: 3.25 S = (O + P + V)*(MR * MS)
// Рабійчук Дар'я Олександрівна
// ІМ-21
// 19.02.2025


namespace Lab1Project
{
    // Функція F1: d = (A * ((B + C) * (MA * ME)))
    class ThreadF1
    {
        public void Run()
        {
print  СТАРТ Т1
            // Для N в малому діапазоні дані вже введені в головному потоці
            if (Data.N >= 1000)
            {
                Data.FillVectorRandomly(Data.A);
                Data.FillVectorRandomly(Data.B);
                Data.FillVectorRandomly(Data.C);
                Data.FillMatrixRandomly(Data.MA);
                Data.FillMatrixRandomly(Data.ME);
            }
            int[] sumBC = Data.AddVectors(Data.B, Data.C);
            int[][] productMA_ME = Data.MultiplyMatrices(Data.MA, Data.ME);
            int[] tempVector = Data.MultiplyVectorMatrix(sumBC, productMA_ME);
            int d = Data.DotProduct(Data.A, tempVector);

            Console.WriteLine("T1 is finished");
            Console.WriteLine("Result of F1: " + d + "\n");
print ЗАВЕРШЕННЯ Т1
        }
    }

    // Функція F2: MF = MG + (MH * MK) + ML
    class ThreadF2
    {
        public void Run()
        {
            if (Data.N >= 1000)
            {
                Data.FillMatrixRandomly(Data.MG);
                Data.FillMatrixRandomly(Data.MH);
                Data.FillMatrixRandomly(Data.MK);
                Data.FillMatrixRandomly(Data.ML);
            }
            int[][] productMH_MK = Data.MultiplyMatrices(Data.MH, Data.MK);
            int[][] tempMatrix = Data.AddMatrices(Data.MG, productMH_MK);
            int[][] MF = Data.AddMatrices(tempMatrix, Data.ML);
```

```
            Console.WriteLine("T2 is finished");
            Console.WriteLine("Result of F2:");
            for (int i = 0; i < Data.N; i++)
            {
                for (int j = 0; j < Data.N; j++)
                {
                    Console.Write(MF[i][j] + " ");
                }
                Console.WriteLine();
            }
            Console.WriteLine();
        }
    }

    // Функція F3: S = (O + P + V) * (MR * MS)
    class ThreadF3
    {
        public void Run()
        {
            if (Data.N >= 1000)
            {
                Data.FillVectorRandomly(Data.O);
                Data.FillVectorRandomly(Data.P);
                Data.FillVectorRandomly(Data.V);
                Data.FillMatrixRandomly(Data.MR);
                Data.FillMatrixRandomly(Data.MS);
            }
            int[] sumOPV = Data.AddVectors(Data.O, Data.AddVectors(Data.P,
Data.V));
            int[][] productMR_MS = Data.MultiplyMatrices(Data.MR, Data.MS);
            int[] tempVector = Data.MultiplyVectorMatrix(sumOPV, productMR_MS);
            int S = Data.DotProduct(sumOPV, tempVector);

            Console.WriteLine("T3 is finished");
            Console.WriteLine("Result of F3: " + S + "\n");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // Ініціалізація ресурсів та зчитування N
            Data.InitializeResources();

            // Після вводу N виводимо повідомлення про старт потоків
            Console.WriteLine("T1 is started");
            Console.WriteLine("T2 is started");
            Console.WriteLine("T3 is started");

            // Якщо N в малому діапазоні, дані вводяться послідовно в головному
потоці
            if (Data.N <= 4)
```

```
            {
                Console.WriteLine("\nInput data for function F1:");
                Data.FillVectorFromConsole(Data.A, "A");
                Data.FillVectorFromConsole(Data.B, "B");
                Data.FillVectorFromConsole(Data.C, "C");
                Data.FillMatrixFromConsole(Data.MA, "MA");
                Data.FillMatrixFromConsole(Data.ME, "ME");

                Console.WriteLine("Input data for function F2:");
                Data.FillMatrixFromConsole(Data.MG, "MG");
                Data.FillMatrixFromConsole(Data.MH, "MH");
                Data.FillMatrixFromConsole(Data.MK, "MK");
                Data.FillMatrixFromConsole(Data.ML, "ML");

                Console.WriteLine("Input data for function F3:");
                Data.FillVectorFromConsole(Data.O, "O");
                Data.FillVectorFromConsole(Data.P, "P");
                Data.FillVectorFromConsole(Data.V, "V");
                Data.FillMatrixFromConsole(Data.MR, "MR");
                Data.FillMatrixFromConsole(Data.MS, "MS");
            }

            // Створення потоків для обчислень
            ThreadF1 threadF1 = new ThreadF1();
            ThreadF2 threadF2 = new ThreadF2();
            ThreadF3 threadF3 = new ThreadF3();

            Thread t1 = new Thread(new ThreadStart(threadF1.Run));  T1
1            Thread t2 = new Thread(new ThreadStart(threadF2.Run));
            Thread t3 = new Thread(new ThreadStart(threadF3.Run));

            t1.Start();
            t2.Start();
            t3.Start();

            // Очікування завершення потоків
            t1.Join();
            t2.Join();
            t3.Join();
        }
    }
}
```

Data.cs

```
using System;

namespace Lab1Project
{
    public static class Data
    {
        public static int N;
```

```
public static int[] A, B, C, O, P, V;
public static int[][] MA, ME, MG, MH, MK, ML, MR, MS;

// Метод для створення квадратної матриці (jagged array)
private static int[][] CreateMatrix(int n)
{
    int[][] matrix = new int[n][];
    for (int i = 0; i < n; i++)
    {
        matrix[i] = new int[n];
    }
    return matrix;
}

public static void InitializeResources()
{
    Console.Write("Enter the value of N: ");
    N = int.Parse(Console.ReadLine());

    A = new int[N];
    B = new int[N];
    C = new int[N];
    O = new int[N];
    P = new int[N];
    V = new int[N];

    MA = CreateMatrix(N);
    ME = CreateMatrix(N);
    MG = CreateMatrix(N);
    MH = CreateMatrix(N);
    MK = CreateMatrix(N);
    ML = CreateMatrix(N);
    MR = CreateMatrix(N);
    MS = CreateMatrix(N);
}

public static void FillVectorFromConsole(int[] vector, string name)
{
    Console.WriteLine($"\nEnter elements for vector {name}:");
    for (int i = 0; i < N; i++)
    {
        Console.Write($"  {name}[{i}] = ");
        vector[i] = int.Parse(Console.ReadLine());
    }
    Console.WriteLine();
}

public static void FillMatrixFromConsole(int[][] matrix, string name)
{
    Console.WriteLine($"\nEnter elements for matrix {name}:");
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
```

```
            Console.Write($"  {name}[{i}][{j}] = ");
            matrix[i][j] = int.Parse(Console.ReadLine());
        }
        Console.WriteLine();
    }
    Console.WriteLine();
}

public static void FillVectorRandomly(int[] vector)
{
    Random random = new Random();
    for (int i = 0; i < N; i++)
    {
        vector[i] = random.Next(100);
    }
}

public static void FillMatrixRandomly(int[][] matrix)
{
    Random random = new Random();
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            matrix[i][j] = random.Next(100);
        }
    }
}

public static int[] AddVectors(int[] vector1, int[] vector2)
{
    int[] result = new int[N];
    for (int i = 0; i < N; i++)
    {
        result[i] = vector1[i] + vector2[i];
    }
    return result;
}

public static int[][] AddMatrices(int[][] matrix1, int[][] matrix2)
{
    int[][] result = CreateMatrix(N);
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
    return result;
}

// Множення вектора на матрицю
public static int[] MultiplyVectorMatrix(int[] vector, int[][] matrix)
```

```
        {
            int[] result = new int[N];
            for (int i = 0; i < N; i++)
            {
                int sum = 0;
                for (int j = 0; j < N; j++)
                {
                    sum += vector[j] * matrix[j][i];
                }
                result[i] = sum;
            }
            return result;
        }

        public static int[][] MultiplyMatrices(int[][] matrix1, int[][] matrix2)
        {
            int[][] result = CreateMatrix(N);
            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < N; j++)
                {
                    int sum = 0;
                    for (int k = 0; k < N; k++)
                    {
                        sum += matrix1[i][k] * matrix2[k][j];
                    }
                    result[i][j] = sum;
                }
            }
            return result;
        }

        // Скалярний добуток двох векторів
        public static int DotProduct(int[] vector1, int[] vector2)
        {
            int result = 0;
            for (int i = 0; i < N; i++)
            {
                result += vector1[i] * vector2[i];
            }
            return result;
        }
    }
}
```

Результат виконання програми

```
Enter the value of N: 4
T1 is started
T2 is started
T3 is started

Input data for function F1:

Enter elements for vector A:
  A[0] = 1
  A[1] = 2
  A[2] = 3
```

```
MA[2][0] = 3
MA[2][1] = 4
MA[2][2] = 5
MA[2][3] = 6

MA[3][0] = 7
MA[3][1] = 8
MA[3][2] = 9
MA[3][3] = 1
```

```
Enter elements for matrix MG:
  MG[0][0] = 9
  MG[0][1] = 1
  MG[0][2] = 2
  MG[0][3] = 3

  MG[1][0] = 4
  MG[1][1] = 5
  MG[1][2] = 6
  MG[1][3] = 7

  MG[2][0] = 8
  MG[2][1] = 9
  MG[2][2] = 1
  MG[2][3] = 2

  MG[3][0] = 3
  MG[3][1] = 4
  MG[3][2] = 5
  MG[3][3] = 6


Enter elements for matrix MH:
  MH[0][0] = 7
  MH[0][1] = 8
  MH[0][2] = 9
  MH[0][3] = 1

  MH[1][0] = 2
  MH[1][1] = 3
  MH[1][2] = 4
  MH[1][3] = 5

  MH[2][0] = 6
  MH[2][1] = 7
  MH[2][2] = 8
  MH[2][3] = 9

  MH[3][0] = 1
  MH[3][1] = 2
  MH[3][2] = 3
  MH[3][3] = 4


Enter elements for matrix MK:
  MK[0][0] = 5
  MK[0][1] = 6
  MK[0][2] = 7
  MK[0][3] = 8

  MK[1][0] = 9
  MK[1][1] = 1
  MK[1][2] = 2
  MK[1][3] = 3

  MK[2][0] = 4
  MK[2][1] = 5
  MK[2][2] = 6
  MK[2][3] = 7

  MK[3][0] = 9
  MK[3][1] = 1
  MK[3][2] = 2
  MK[3][3] = 3


Enter elements for matrix ML:
  ML[0][0] = 4
  ML[0][1] = 5
  ML[0][2] = 6
  ML[0][3] = 7

  ML[1][0] = 8
  ML[1][1] = 9
  ML[1][2] = 1
  ML[1][3] = 2

  ML[2][0] = 3
  ML[2][1] = 4
  ML[2][2] = 5
  ML[2][3] = 6

  ML[3][0] = 7
  ML[3][1] = 8
  ML[3][2] = 9
  ML[3][3] = 1


Input data for function F3:

Enter elements for vector O:
  O[0] = 2
  O[1] = 3
  O[2] = 4
  O[3] = 5


Enter elements for vector P:
  P[0] = 6
  P[1] = 7
  P[2] = 8
  P[3] = 9


Enter elements for vector V:
  V[0] = 1
  V[1] = 2
  V[2] = 3
  V[3] = 4
```

```
Enter elements for matrix MR:
  MR[0][0] = 5
  MR[0][1] = 6
  MR[0][2] = 7
  MR[0][3] = 8

  MR[1][0] = 9
  MR[1][1] = 1
  MR[1][2] = 2
  MR[1][3] = 3

  MR[2][0] = 4
  MR[2][1] = 5
  MR[2][2] = 6
  MR[2][3] = 7

  MR[3][0] = 8
  MR[3][1] = 1
  MR[3][2] = 2
  MR[3][3] = 3
```

```
Enter elements for matrix MS:
  MS[0][0] = 4
  MS[0][1] = 5
  MS[0][2] = 6
  MS[0][3] = 7

  MS[1][0] = 8
  MS[1][1] = 9
  MS[1][2] = 1
  MS[1][3] = 2

  MS[2][0] = 3
  MS[2][1] = 5
  MS[2][2] = 6
  MS[2][3] = 7

  MS[3][0] = 8
  MS[3][1] = 9
  MS[3][2] = 1
  MS[3][3] = 2
```

```
T1 is finished
T2 is finished
Result of F2:
Result of F1: 48640

T3 is finished
Result of F3: 274329

165 102 129 156
110 54 61 77
217 105 128 160
81 39 51 54
```

Зауваження до результату роботи

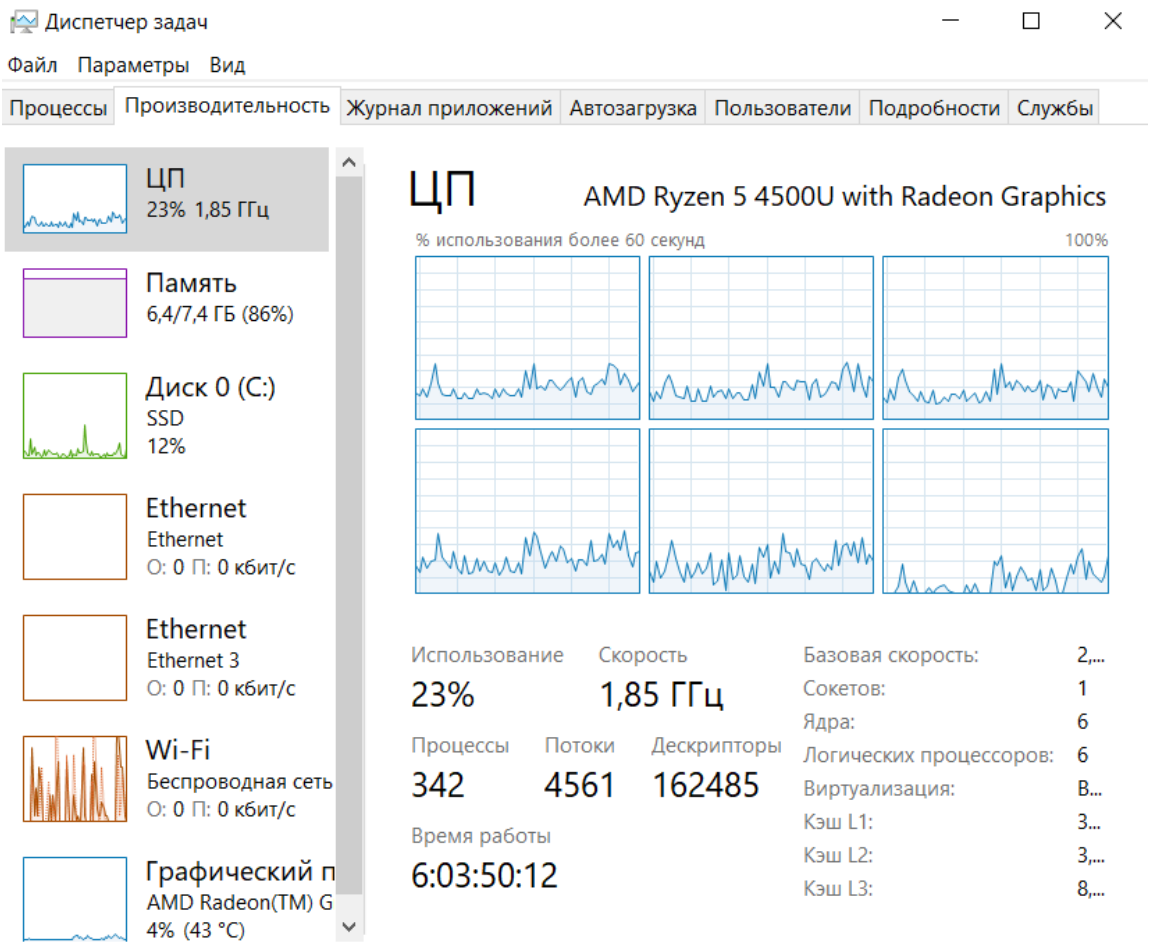
При зчитуванні даних з консолі може виникнути ситуація, коли декілька потоків одночасно намагаються зчитати дані, що призводить до некоректного вводу. Щоб уникнути цього, у C# можна використовувати механізм блокування (lock), який забезпечує ексклюзивний доступ до потоку введення, дозволяючи лише одному потоку одночасно виконувати зчитування даних.

А виведення?

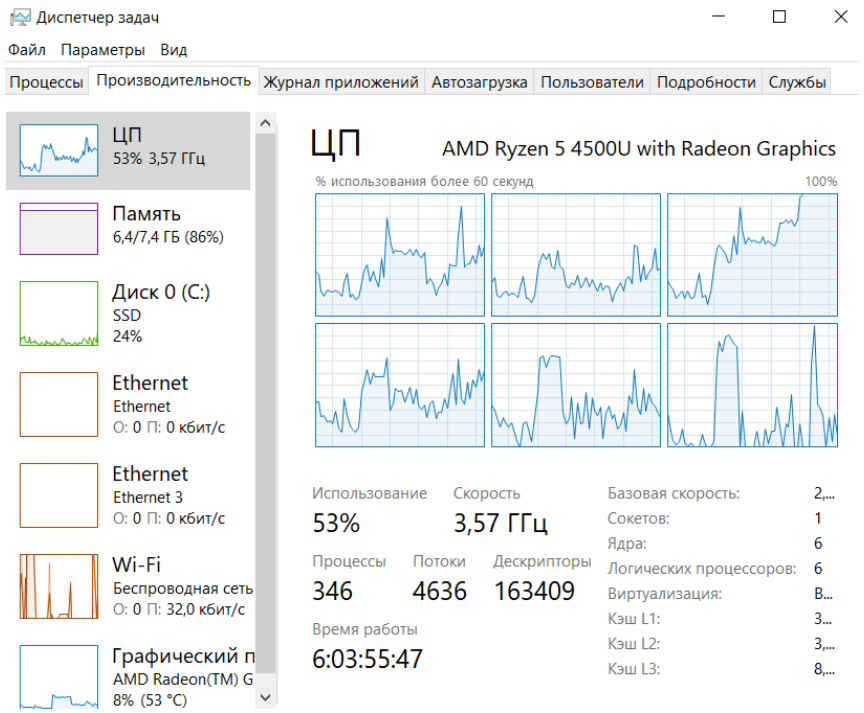
Аналіз завантаженності процесора при значенні N=1000

Тестування проводилось за режиму заповнення матриць і векторів випадковими числами. Тестування відбувалось на процесорі AMD Ryzen 5 4500U у котрого 6 ядер та 6 логічних процесорів.

Завантаженість процесорів перед тестуванням:

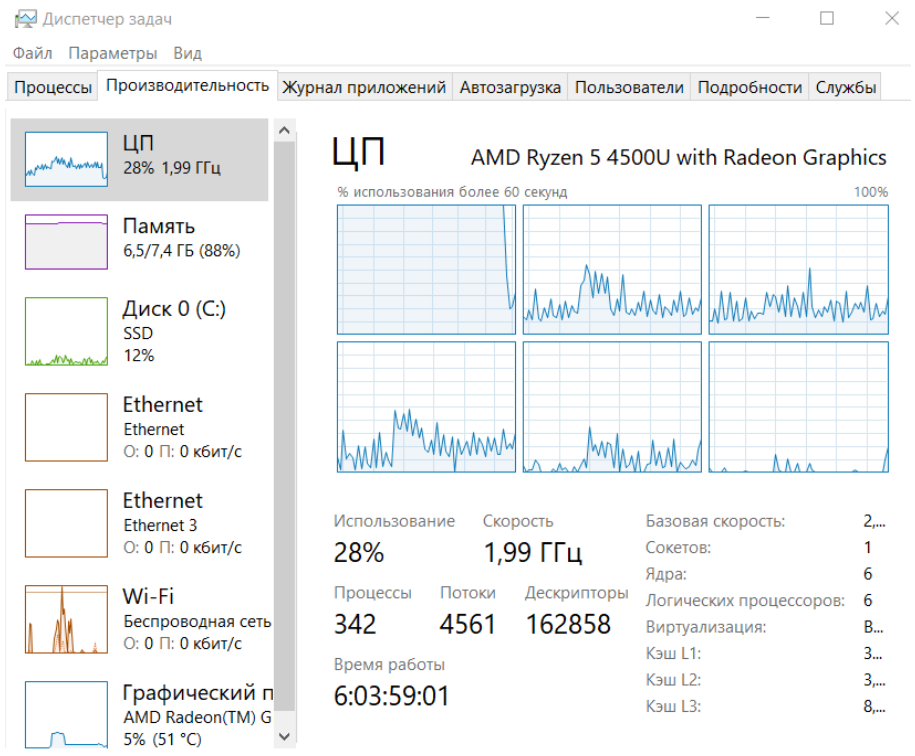


Завантаженість всіх 6 ядер під час тестування :



Згідно з результатів тестування завантаженість зросла на всіх ядрах. Це обумовлено тим, що система може автоматично розподіляти потоки між різними ядрами.

Завантаженість 1 ядра під час тестування :



При навантаженні всіх 6 ядер час виконання склав Час6 = 46 секунд. Після зменшення ядер процесору до 1 час збільшився до Час2 = 58 секунд. Тому: $K_п \approx Час1/Час6 = 1,3$.

Висновки

1 У цій лабораторній роботі було розроблено програму, яка паралельно обчислює три математичні функції (F1, F2, F3) у трьох окремих потоках (T1, T2, T3). Для реалізації багатопотоковості було використано мову програмування C# та клас Thread, який забезпечує можливість створення та управління потоками.

2 Проблеми введення ы виведення для N=4 ?

2 Результати тестування показали, що використання багатопотоковості не дало очікуваного прискорення (Кп ≈ 1.03), що може бути пов’язано з обмеженнями доступу до оперативної пам’яті (RAM) та кешу процесора, нерівномірним розподілом навантаження між потоками, недостатнім обсягом обчислень для повноцінного використання багатоядерності.
Отримані результати також показали, що ефективність багатопотоковості залежить не лише від кількості ядер, але й від правильного розподілу обчислень між ними. Для значно більших розмірів матриць можна очікувати суттєвіше зростання прискорення.