Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.04 – «Программная инженерия»

Дисциплина: «Технологии блокчейн и распределенные информационные системы»

Семестр 8

ОТЧЕТ

по лабораторной работе №3

Тема: «Решение СЛАУ с помощью OpenMP»

Выполнил: студент группы РИС-20-2б

Гулуа И.Ю. \_\_\_\_\_\_\_\_\_\_\_

Проверил: старший преподаватель кафедры ИТАС

Щапов В. А. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

Пермь, 2024

**Цель работы**

Исследовать метод Гаусса для решения систем линейных алгебраических уравнений (СЛАУ) на возможность и целесообразность распараллеливания операций.

**Ход работы**

Алгоритм Гаусса, также известный как метод исключения Гаусса, предназначен для решения систем линейных алгебраических уравнений (СЛАУ). Он состоит из двух этапов:

**1. Прямой ход (преобразование к треугольному виду):**

* **Шаг 1:** Выбираем первый ненулевой элемент в первом столбце (ведущий элемент).
* **Шаг 2:** С помощью элементарных преобразований строк (умножение строки на число, сложение строк) делаем все элементы ниже ведущего элемента равными нулю.
* **Шаг 3:** Повторяем шаги 1 и 2 для второго столбца, начиная со второй строки, затем для третьего столбца, начиная с третьей строки, и так далее, пока не получим систему уравнений в **верхнетреугольном виде**.

**2. Обратный ход (решение системы):**

* **Шаг 1:** Находим значение последней переменной из последнего уравнения.
* **Шаг 2:** Подставляем найденное значение в предпоследнее уравнение и находим значение предпоследней переменной.
* **Шаг 3:** Продолжаем этот процесс, подставляя найденные значения переменных в предыдущие уравнения, пока не найдем значения всех переменных.

// Функция для решения СЛАУ методом Гаусса

void gaussianElimination(vector<vector<double>>& matrix, vector<double>& b, vector<double>& x, int size) {

// Прямой ход метода Гаусса для получения верхней треугольной матрицы

for (int k = 0; k < size - 1; k++) {

#pragma omp parallel for

for (int i = k + 1; i < size; i++) {

double factor = matrix[i][k] / matrix[k][k];

for (int j = k; j < size; j++) {

matrix[i][j] -= factor \* matrix[k][j];

}

b[i] -= factor \* b[k];

}

}

// Обратный ход метода Гаусса для нахождения неизвестных

for (int i = size - 1; i >= 0; i--) {

x[i] = b[i];

for (int j = i + 1; j < size; j++) {

x[i] -= matrix[i][j] \* x[j];

}

x[i] /= matrix[i][i];

}

}

Листинг 1 – Прямой и обратный ход алгоритма.

В представленном коде мы распараллеливаем цикл по i с помощью #pragma omp parallel for.

Итерации этого цикла по i независимы друг от друга. Это означает, что вычисления для каждой строки i не зависят от вычислений для других строк. Такая независимость позволяет выполнять итерации параллельно без возникновения гонок данных (race conditions).

Внутренний цикл по j выполняет умножение и вычитание для каждого элемента в строке i. Это делает цикл по i наиболее вычислительно затратным, поэтому его распараллеливание может дать наибольший прирост производительности.

Остальные циклы нецелесообразно делить на несколько потоков, потому что:

* **Цикл по k:** Итерации этого цикла **не независимы**. Каждая итерация зависит от результатов предыдущей, поэтому их нельзя выполнять параллельно.
* **Цикл по j:** Хотя этот цикл итерационно независим, он выполняет сравнительно **небольшое количество вычислений**. Распараллеливание этого цикла может не дать существенного прироста производительности, а накладные расходы, связанные с созданием и управлением потоками, могут даже снизить производительность.
* **Цикл в обратном ходе:** Итерации этого цикла **зависят друг от друга**. Значение x[i] зависит от значений x[j] для j > i. Поэтому параллельное выполнение невозможно.

В итоге, распараллеливание цикла по i в прямом ходе метода Гаусса является наиболее целесообразным, так как он обладает независимыми итерациями и выполняет наибольший объем вычислений.  
Распараллеливание других циклов либо невозможно, либо не даст существенного прироста производительности.

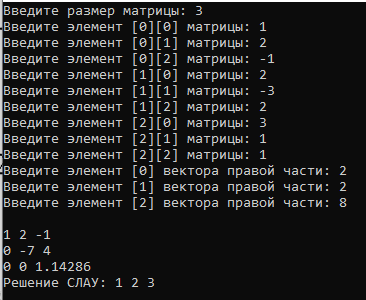


Рисунок 1 – Результат работы умножения при перестановке циклов в порядке i,j,k (классический).

**Вывод:** Распараллеливание цикла по i в прямом ходе метода Гаусса является наиболее целесообразным, так как он обладает независимыми итерациями и выполняет наибольший объем вычислений.

Распараллеливание других циклов либо невозможно, либо не даст существенного прироста производительности.