**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное   
образовательное учреждение

Высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Направление: «Программная инженерия»

Отчет по лабораторной работе №7 по дисциплине

**«Параллельное, распределенное программирование»**

Выполнил:

Студент группы 8К61 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Исламов Е.Р.

Принял:

Доцент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мокина Е.Е.

Томск 2019

**Содержание**

[Цель работы 3](#_Toc29516501)

[Ход работы 4](#_Toc29516502)

[Вывод 8](#_Toc29516503)

# Цель работы

Разработать программу, реализующую алгоритм решения линейных уравнений с использованием потоков в Java.

# Ход работы

В ходе работы была разработана программа для многопоточного решения систем линейных уравнений методом Гаусса. Исходный код предоставлен ниже.

Класс EquationSystem представляет собой систему уравнений и обладает методом GaussParallel, позволяющий решить систему методом Гаусса с вычитанием строк в параллельных потоках.

**package** Lab7;  
  
*/\*\*  
 \* Система уравнений.  
 \*/***public class** EquationSystem {  
 */\*\*  
 \* Матрица коэффициентов.  
 \*/* **double**[][] **a**;  
  
 */\*\*  
 \* Вектор решений.  
 \*/* **double**[] **b**;  
  
 */\*\*  
 \* Результат.  
 \*/* **double**[] **result**;  
  
 */\*\*  
 \* Потоки, в которых выполняются преобразования над строками матрицы.  
 \*/* **private** Thread[] **threads**;  
  
 */\*\*  
 \* Количество неизвестных.  
 \*/* **int size**;  
  
 */\*\*  
 \* Конструктор.  
 \** ***@param a*** *Матрица коэффициентов.  
 \** ***@param b*** *Вектор решений.  
 \*/* **public** EquationSystem(**double**[][] a, **double**[] b) {  
 **this**.**a** = a;  
 **this**.**b** = b;  
 **size** = a[0].**length**;  
 **threads** = **new** Thread[**size**];  
 **result** = **new double**[**size**];  
 }  
  
 */\*\*  
 \* Решить СЛУ методом Гаусса в параллельном режиме.  
 \*/* **public void** GaussParallel() {  
 *// Итерация по каждой строке.* **for** (**int** row = 0; row < **size**; row++) {  
 *// Делаем, чтобы на диагонали были только единицы.* **double** value = **a**[row][row];  
 **for** (**int** col = row + 1; col < **size**; col++) {  
 **a**[row][col] /= value;  
 }  
  
 **b**[row] /= value;  
 **a**[row][row] = 1.0;  
  
 *// Вычитаем текущую строку из следующих строк в параллельных потоках.* **for** (**int** innerRow = row + 1; innerRow < **size**; innerRow++) {  
 **threads**[innerRow] = **new** SubtractionThread(**this**, innerRow, row);  
 **threads**[innerRow].start();  
 }  
  
 *// Ожидаем выполнение потоков вычитания.* **for** (**int** innerRow = row + 1; innerRow < **size**; innerRow++) {  
 **try** {  
 **threads**[innerRow].join();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 *// Обратный ход.* **for** (**int** back = **size** - 1; back >= 0; back--) {  
 **result**[back] = **b**[back];  
 **for** (**int** i = back - 1; i >= 0; i--) {  
 **b**[i] -= **result**[back] \* **a**[i][back];  
 }  
 }  
 }  
}

**package** Lab7;  
  
*/\*\*  
 \* Поток вычитания одной строки СЛУ из другой.  
 \*/***public class** SubtractionThread **extends** Thread {  
 */\*\*  
 \* Система уравнений.  
 \*/* **private** EquationSystem **system**;  
  
 */\*\*  
 \* Строка из которой вычитать.  
 \*/* **private int innerRow**;  
  
 */\*\*  
 \* Строка которую вычитать.  
 \*/* **private int row**;  
  
 */\*\*  
 \* Конструктор.  
 \** ***@param system*** *Система уравнений.  
 \** ***@param innerRow*** *Строка из которой вычитать.  
 \** ***@param row*** *Строка которую вычитать.  
 \*/* **public** SubtractionThread(EquationSystem system, **int** innerRow, **int** row) {  
 **this**.**system** = system;  
 **this**.**innerRow** = innerRow;  
 **this**.**row** = row;  
 }  
  
 */\*\*  
 \* Выполнение вычитания.  
 \*/* @Override  
 **public void** run() {  
 **double** innerValue = **system**.**a**[**innerRow**][**row**];  
 **for** (**int** innerCol = **row** + 1; innerCol < **system**.**size**; innerCol++) {  
 **system**.**a**[**innerRow**][innerCol] -= innerValue \* **system**.**a**[**row**][innerCol];  
 }  
 **system**.**b**[**innerRow**] -= **system**.**a**[**innerRow**][**row**] \* **system**.**b**[**row**];  
 **system**.**a**[**innerRow**][**row**] = 0.0;  
 }  
}

В классе Main задается произвольная система уравнений и находится ее решение.

**package** Lab7;  
  
*/\*\*  
 \* Входная точка.  
 \*/***public class** Main {  
 */\*\*  
 \* Входная точка.  
 \** ***@param args*** *Параметры запуска.  
 \*/* **public static void** main(String[] args) {  
 **double**[][] a = {  
 {1, 3, 2},  
 {2, 7, 5},  
 {1, 4, 6}};  
 **double**[] b = {1., 18., 26.};  
 *printSLE*(a, b, **"Базовая"**);  
  
 EquationSystem system = **new** EquationSystem(a, b);  
 system.GaussParallel();  
 *printSLE*(a, b, **"Треугольная"**);  
  
 System.***out***.println(**"Ответ:"**);  
 *printVector*(system.**result**, **true**);  
 }  
  
 */\*\*  
 \* Вывести СЛУ.  
 \** ***@param a*** *Матрица коэффициентов.  
 \** ***@param b*** *Матрица решений.  
 \** ***@param name*** *Наименование матрицы.  
 \*/* **public static void** printSLE(**double**[][] a, **double**[] b, String name) {  
 System.***out***.printf(**"%s матрица:\n"**, name);  
 *printMatrix*(a);  
 System.***out***.println(**"\nВектор решений:"**);  
 *printVector*(b, **false**);  
 System.***out***.println(**"\n"**);  
 }  
  
 */\*\*  
 \* Вывести двумерный массив в виде матрицы.  
 \** ***@param matrix*** *Двумерный массив.  
 \*/* **public static void** printMatrix(**double**[][] matrix) {  
 **for** (**int** row = 0; row < matrix.**length**; row++) {  
 **for** (**int** col = 0; col < matrix[row].**length**; col++) {  
 System.***out***.printf(**"%.1f "**, matrix[row][col]);  
 }  
 System.***out***.println();  
 }  
 }  
  
 */\*\*  
 \* Вывести массив.  
 \** ***@param vector*** *Массив.  
 \** ***@param includeX*** *Включать ли x0, x1... в вывод.  
 \*/* **public static void** printVector(**double**[] vector, **boolean** includeX) {  
 **for** (**int** i = 0; i < vector.**length**; i++) {  
 **if** (includeX)  
 System.***out***.printf(**"x%d: %.1f\n"**, i, vector[i]);  
 **else** System.***out***.printf(**"%.1f "**, vector[i]);  
 }  
 }  
}

На рисунке 1 приведен результат работы программы.

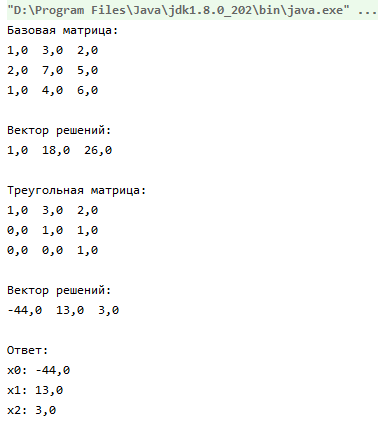


Рисунок 1 - результат работы программы

# Вывод

В результате выполненной работы были получены навыки работы с многопоточностью на примере решения СЛУ методом Гаусса.