КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»

Кафедра мікроелектроніки

**ЛАБОРАТОРНА РОБОТА №8**

з дисципліни «Обчислювальна математика»

Варіант 3

Роботу виконав

Ст. групи ДП-01

Деркач Євген

Роботу перевірив

Татарчук Д. Д.

Київ-2021

**ЛАБОРАТОРНА РОБОТА №8**

**Тема:** Розв’язання систем лінійних алгебраїчних рівнянь. Метод Гауса

**Мета роботи:** вивчення алгоритмів і налаштування програм розв’язання систем лінійних алгебраїчних рівнянь (СЛАР) методом Гауса.

**Що зробити:** скласти процедуру для розв’язання СЛАР методом Гауса, яка б у випадку невиродженої системи знаходила її розв’язок, а для виродженої системи видавала відповідне попередження. Впевнитися в коректності роботи процедури, підставляючи в СЛАР отримані розв’язки і обраховуючи нев’язки. Додатково – передбачити оцінку числа обумовленості матриці системи.

**Код програми:**

package com.Jeka8833.LabsKPITwo.lab.labs;  
  
import com.Jeka8833.LabsKPITwo.ForceStopException;  
import com.Jeka8833.LabsKPITwo.Reader;  
import com.Jeka8833.LabsKPITwo.Writer;  
import com.Jeka8833.LabsKPITwo.lab.Lab;  
import org.apache.commons.math3.linear.\*;  
import org.jetbrains.annotations.Nullable;  
  
public class Laba8 implements Lab {  
  
 @Override  
 public String getName() {  
 return "No 8. Розв’язання систем лінійних алгебраїчних рівнянь. Метод Гауса";  
 }  
  
 @Override  
 public void run() throws ForceStopException {  
 final int n = Reader.readInt("Введите n: ");  
 final double[][] a = Reader.readMatrix(n, n, "Введём матрицу СЛАР A[" + n + "," + n + "], можно написать random - для авто заполнения матрицы");  
 final double[] b = Reader.readMatrix(n, "Вектор B[" + n + "], можно написать random - для авто заполнения матрицы");  
  
 Writer.addMatrix(a, "a-");  
 Writer.addMatrix(b, "b-");  
  
 try {  
 RealMatrix coefficients = new Array2DRowRealMatrix(a, true);  
 DecompositionSolver solver = new LUDecomposition(coefficients).getSolver();  
 RealVector constants = new ArrayRealVector(b, true);  
 RealVector solution = solver.solve(constants);  
 Writer.addMatrix(solution.toArray(), "Lib out x-");  
 } catch (Exception exception) {  
 System.out.println("Lib error");  
 }  
 Writer.addMatrix(a, "a-");  
 Writer.addMatrix(b, "b-");  
  
 final double[] x = calc(n, a, b);  
 if (x == null)  
 return;  
 Writer.addMatrix(x, "My out x-");  
 Writer.saveResult();  
 }  
  
 public static double @Nullable [] calc(final int n, final double[][] a, final double[] b) {  
 for (int i = 0; i < n; i++) {  
 int max = i;  
 for (int j = i + 1; j < n; j++) {  
 if (Math.abs(a[j][i]) > Math.abs(a[max][i])) {  
 max = j;  
 }  
 }  
 double[] temp = a[i];  
 a[i] = a[max];  
 a[max] = temp;  
 double t = b[i];  
 b[i] = b[max];  
 b[max] = t;  
  
 if (Math.abs(a[i][i]) <= 1e-10) {  
 System.out.println("ErrCode: -" + i);  
 return null;  
 }  
  
 for (int k = i + 1; k < n; k++) {  
 final double p = a[k][i] / a[i][i];  
 for (int j = i; j < n; j++) {  
 a[k][j] -= p \* a[i][j];  
 }  
 b[k] -= p \* b[i];  
 }  
 }  
 final double[] x = new double[n];  
 for (int i = n - 1; i >= 0; i--) {  
 double sum = 0;  
 for (int j = i + 1; j < n; j++) {  
 sum += a[i][j] \* x[j];  
 }  
 x[i] = (b[i] - sum) / a[i][i];  
 }  
 return x;  
 }  
}