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

Кафедра мікроелектроніки

**ЛАБОРАТОРНА РОБОТА №11**

з дисципліни «Обчислювальна математика»

Варіант 3

Роботу виконав

Ст. групи ДП-01

Деркач Євген

Роботу перевірив

Татарчук Д. Д.

Київ-2021

**ЛАБОРАТОРНА РОБОТА №11**

**Тема:** Знаходження власних векторів і власних чисел симетричних матриць. Метод обертань Якобі

**Мета роботи:** вивчення алгоритму і налаштування програми для знаходження власних векторів і власних чисел симетричних матриць методом обертань Якобі.

**Що зробити:** знайти власні числа і власні вектори симетричної матриці A шляхом її декомпозиції методом обертань Якобі в добуток виду A = QDQT, де D – діагональна матриця власних чисел, а Q – ортогональна матриця власних векторів. Впевнитися у правильності результату шляхом перевірки співвідношень QTQ = E та QDQT= A. Додатково – впевнитися в інваріантності сліду матриці A та її норми Фробеніуса при послідовних ітераціях. Оцінити порядок збіжності методу обертань Якобі.

**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;  
  
public class Laba11 implements Lab {  
 @Override  
 public String getName() {  
 return null;  
 }  
  
 @Override  
 public void run() throws ForceStopException {  
 final int n = Reader.*readInt*("Введите n: ");  
 final double[][] D = Reader.*readMatrix*(n, n, "Введём матрицу СЛАР D[" + n + "," + n + "], можно написать random - для авто заполнения матрицы");  
 final double e = Reader.*readDouble*("Введите e: ");  
 final double[][] Q = new double[n][n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 if (i == j)  
 Q[i][j] = 1;  
 }  
 }  
 while (true) {  
 double Dlm = D[1][0];  
 int l = 1;  
 int m = 0;  
  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < i - 1; j++) {  
 if (Math.*abs*(D[i][j]) > Math.*abs*(Dlm)) {  
 Dlm = D[i][j];  
 l = i;  
 m = j;  
 }  
 }  
 }  
  
 if (Math.*abs*(Dlm) < e)  
 break;  
 double t = (D[0][0] - D[m][m]) / (2 \* D[l][m]);  
 double u = t / Math.*sqrt*(1 + t \* t);  
 double c = Math.*sqrt*((1 + u) / 2);  
 double s = Math.*sqrt*((1 - u) / 2);  
  
 for (int i = 0; i < n; i++) {  
 Q[i][l] = Q[i][l] \* c + Q[i][m] \* s;  
 Q[i][m] = -Q[i][l] \* s + Q[i][m] \* c;  
 }  
 double Dll = D[l][l] \* c \* c + 2 \* D[l][m] \* c \* s + D[m][m] \* s \* s;  
 double Dmm = D[l][l] \* s \* s - 2 \* D[l][m] \* c \* s + D[m][m] \* c \* c;  
 D[l][l] = Dll;  
 D[m][m] = Dmm;  
 D[l][m] = 0;  
 D[m][l] = 0;  
 for (int i = 0; i < n; i++) {  
 if (i != l && i != m) {  
 double Dil = D[i][l] \* c + D[i][m] \* s;  
 double Dim = -D[i][l] \* s + D[i][m] \* c;  
 D[i][l] = Dil;  
 D[l][i] = D[i][l];  
 D[i][m] = Dim;  
 D[m][i] = D[i][m];  
 }  
 }  
 }  
 Writer.*addMatrix*(D, "D-");  
 Writer.*addMatrix*(D, "Q-");  
 Writer.*saveResult*();  
 }  
}