Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №1**

‘ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА’

Вариант №24

*Студент:*

Хоанг Ван Куан

Группа Р3266

*Преподаватель:*

Машина Екатерина Александровна

Санкт-Петербург, 2024

**Цель работы**

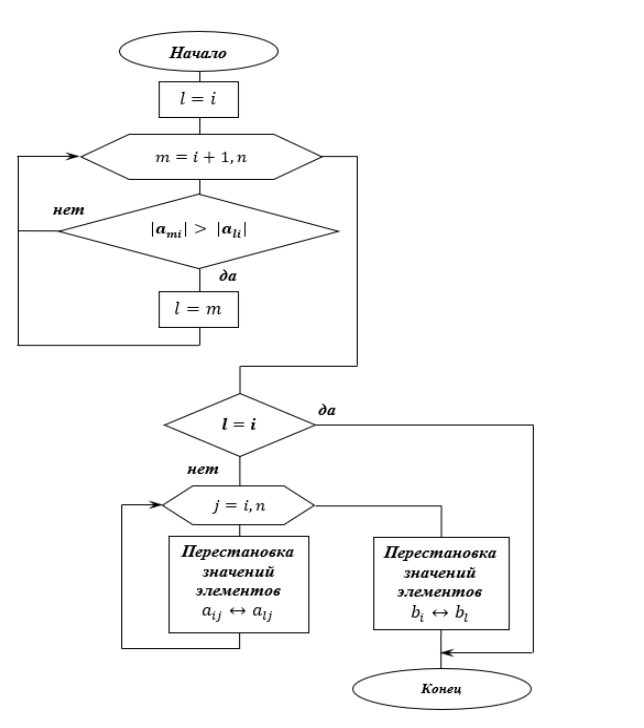
Изучить численные методы решения систем линейных алгебраических уравнений и реализовать один из них средствами программирования.

**Описание метода**

Метод Гаусса с выбором главного элемента по столбцам.

Схема с выбором главного элемента является одной из модификаций метода Гаусса. Идеей является такая перестановка уравнений, чтобы на k-ом шаге исключения ведущим элементом 𝑎𝑖𝑖 оказывался наибольший по модулю элемент k-го столбца.

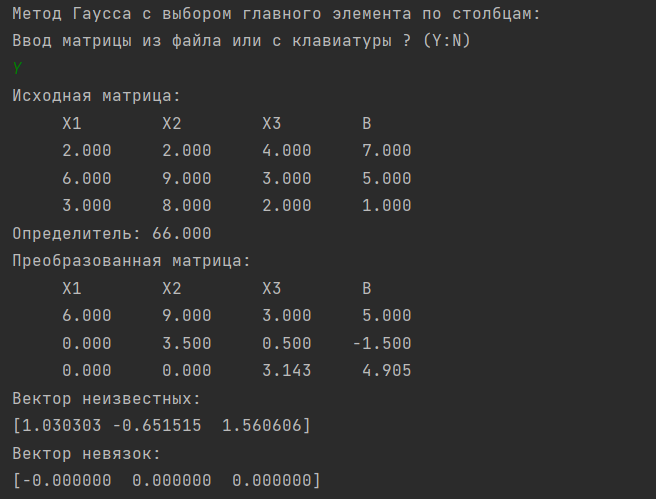
**Блок-схема метода**

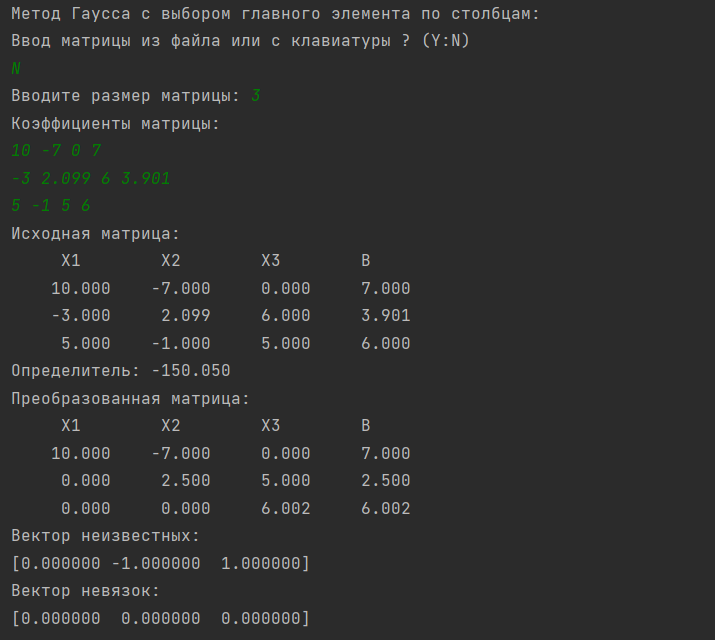
****

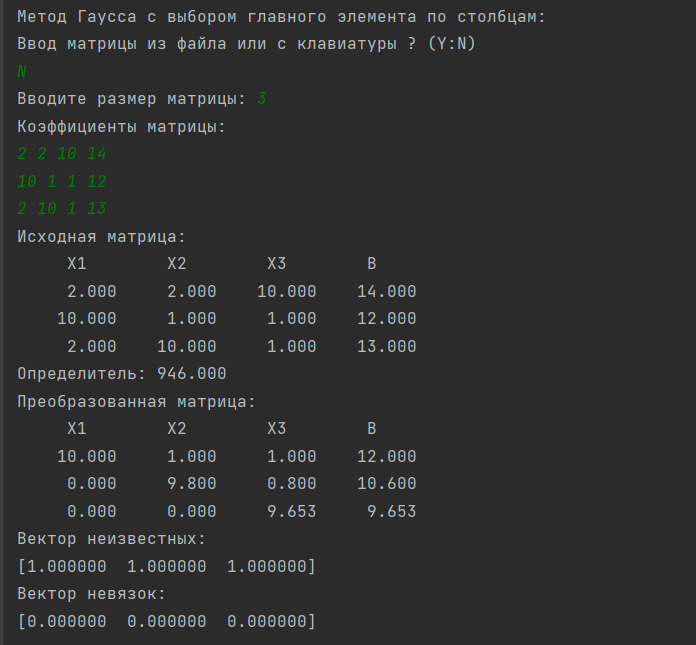
**Код**

import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.Random;  
import java.util.Scanner;  
  
public class Main {  
 static final String *FILE\_INPUT* = "src/FileInput/4.txt";  
 public static void main(String[] args) {  
 //Ввод матрицы из файла или с клавиатуры  
 double[][] matrix = *getInputMethod*();  
  
 //случай если матрица пустая  
 if(*checkNullMatrix*(matrix)) return;  
  
 //Исходная матрица  
 System.*out*.println("Исходная матрица: ");  
 *printMatrix*(matrix);  
  
 //Если определитель = 0, тогда матрица несовсестная  
 double det = *Determinant*(matrix);  
 if (det == 0) {  
 System.*out*.println("Матрица является несовместной"); return;  
 }  
 //Определитель  
 System.*out*.print("Определитель: ");  
 System.*out*.printf("%5.3f",det);  
  
 //Преобразование  
 System.*out*.println("\nПреобразованная матрица:");  
 *printMatrix*(*transpose*(matrix));  
  
 //Решение и погрешность  
 Object[] answer = *Solution*(matrix);  
 double[] roots = (double[]) answer[0];  
 double[] residuals = (double[]) answer[1];  
  
 System.*out*.println("Вектор неизвестных:");  
 *printVector*(roots);  
  
 System.*out*.println("Вектор невязок:");  
 *printVector*(residuals);  
 }  
  
 static double[][] getInputMethod(){  
 double[][] matrix;  
 System.*out*.println("Метод Гаусса с выбором главного элемента по столбцам:");  
 System.*out*.println("Ввод матрицы из файла или с клавиатуры ? (F:K:R)");  
 Scanner scanner = new Scanner(System.*in*);  
 char method = scanner.next().charAt(0);  
 while (method != 'F' && method != 'K' && method != 'R') {  
 System.*out*.println("Введите 'F' или 'K' или 'R' для выбора способа ввода.");  
 method = scanner.next().charAt(0);  
 }  
 if (method == 'F') matrix = *getMatrixFromFile*();  
 else {  
 if (method == 'K') matrix = *getMatrixFromInput*();  
 else matrix = *randomMatrix*();  
 }  
 return matrix;  
 }  
 static double[][] randomMatrix(){  
 Random r = new Random();  
 int n = 20;  
 double[][] matrix = new double[n][n+1];  
 for(int i = 0; i < n; i++){  
 for(int j = 0; j < n + 1; j++){  
 matrix[i][j] = r.nextInt(20);  
 }  
 }  
 return matrix;  
 }  
 static double[][] getMatrixFromFile() {  
 try {  
 Scanner scanner = new Scanner(new File(*FILE\_INPUT*));  
 int n = scanner.nextInt();  
 int m = n + 1; // Размер матрицы и вектора невязок  
 double[][] matrix = new double[n][m];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < m; j++)  
 matrix[i][j] = scanner.nextDouble();  
 }  
 return matrix;  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
 static double[][] getMatrixFromInput() {  
 Scanner scanner = new Scanner(System.*in*);  
 int n;  
 do {  
 System.*out*.print("Вводите размер матрицы: ");  
 while (!scanner.hasNextInt()) {  
 System.*out*.println("Размер матрицы должен быть целым числом");  
 scanner.next();  
 }  
 n = scanner.nextInt();  
 if (n <= 0) System.*out*.println("Размер матрицы должен быть положительным");  
 } while (n <= 0);  
  
 double[][] matrix = new double[n][n + 1];  
 System.*out*.println("Коэффициенты матрицы:");  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n + 1; j++)  
 matrix[i][j] = scanner.nextDouble();  
 }  
 return matrix;  
 }  
 static boolean checkNullMatrix(double[][] matrix){  
 if(matrix == null){  
 System.*out*.println("Матрица пустая");  
 return true;  
 }  
 return false;  
 }  
 static double Determinant(double[][] matrix) {  
 int n = matrix.length;  
 if (n == 1) return matrix[0][0];  
 double det = 0;  
 int sign = 1;  
  
 for (int j = 0; j < n; j++) {  
 double[][] minorMatrix = *minorMatrxix*(matrix, 0, j);  
 det += sign \* matrix[0][j] \* *Determinant*(minorMatrix);  
 sign \*= -1;  
 }  
 return det;  
 }  
 static double[][] minorMatrxix(double[][] matrix, int row, int column){  
 int newLength = matrix.length - 1;  
 double[][] minor = new double[newLength][newLength];  
 for(int i = 0, n\_row = 0; i < matrix.length; i++){  
 if(i != row) {  
 for (int j = 0, n\_column = 0; j < matrix.length; j++) {  
 if (j != column)  
 minor[n\_row][n\_column++] = matrix[i][j];  
 }  
 n\_row++;  
 }  
 }  
 return minor;  
 }  
 static double[][] transpose(double[][] matrix){  
 int n = matrix.length;  
 for (int i = 0; i < n - 1; i++) {  
 int max = i;  
 for (int m = i + 1; m < n; m++) {  
 if (Math.*abs*(matrix[m][i]) > Math.*abs*(matrix[max][i]))  
 max = m;  
 }  
 if (max != i) *swapRows*(matrix, max, i);  
  
 for (int k = i + 1; k < n; k++) {  
 double p = matrix[k][i] / matrix[i][i];  
 for (int j = i; j <= n; j++)  
 matrix[k][j] -= p \* matrix[i][j];  
 }  
 }  
 return matrix;  
 }  
 static Object[] Solution(double[][] matrix){  
 int n = matrix.length;  
 // Решение  
 double[] results = new double[n];  
 for (int i = n - 1; i >= 0; i--) {  
 double sum = 0;  
 for (int j = i + 1; j < n; j++)  
 sum += matrix[i][j] \* results[j];  
 results[i] = (matrix[i][n] - sum) / matrix[i][i];  
 }  
 // Погрешность  
 double[] residuals = new double[n];  
 for (int i = 0; i < n; i++) {  
 double sum = 0;  
 for (int j = 0; j < n; j++)  
 sum += matrix[i][j] \* results[j];  
 residuals[i] = sum - matrix[i][n];  
 }  
 return new Object[]{results, residuals};  
 }  
 static void swapRows(double[][] matrix, int row1, int row2){  
 double[] temp = matrix[row1];  
 matrix[row1] = matrix[row2];  
 matrix[row2] = temp;  
 }  
 static void printMatrix(double[][] matrix){  
 System.*out*.print(" ");  
 for (int i = 1; i < matrix[0].length; i++)  
 System.*out*.printf("X%d%8s", i, "");  
 System.*out*.println("B");  
  
 for (double[] row : matrix) {  
 for (double col : row)  
 System.*out*.printf("%10.3f", col);  
 System.*out*.println();  
 }  
 }  
 static void printVector(double[] vector){  
 System.*out*.print("[");  
 System.*out*.printf("%.6f", vector[0]);  
 for(int i = 1; i < vector.length; i++){  
 System.*out*.printf("%10.6f",vector[i]);  
 }  
 System.*out*.println("]");  
 }  
}

**Примеры работы программы**







**Вывод**

В результате выполнения данной лабораторной работой я познакомился с численными методами решения математических задач на примере систем алгебраических уравнений, реализовав метод Гаусса с выбором главного элемента по столбцам.