Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 1

з дисципліни «Чисельні методи»

Тема: «Чисельні методи розв’язання систем лінійних рівнянь. Метод Гаусса»

Виконала:

студентка 3 курсу

групи КС-32

Дібцева Анна Миколаївна

Перевірила: доцент

Чуб Ольга Ігорівна

Харків – 2020

# ХОД РАБОТЫ

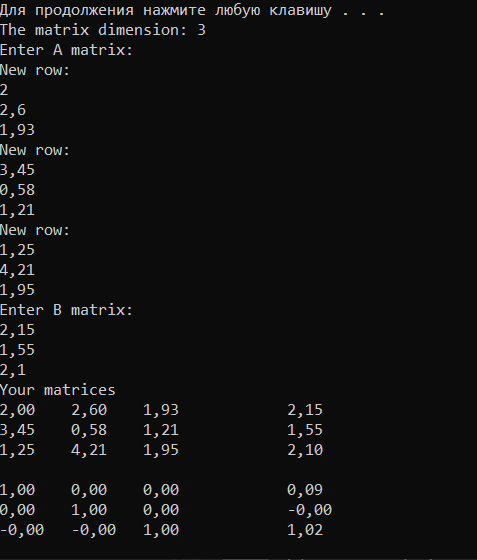
Целью данной работы является исследование алгоритма метода Гаусса для решения систем линейных уравнений и его программная реализация.

Метод решения системы заключается в последовательном преображении матрицы с дополненным столбцом правых решений путем алгебраических действий со строками к виду единичной матрицы (матрицы, у которой элементы на главной диагонали равны единице, а все остальные - нулю). В таком случае вектор правых частей уравнений будет результирующим вектором значений переменных, удовлетворяющих заданным условиям.

Перед началом вычислений программа проверяет определитель матрицы. В случае если он равен нулю, вычисление не начинается, а в строке результата выводится соответствующее сообщение.

Программа скомпилирована в исполняемый файл GaussMethod.exe.

Результат работы программы приведен ниже.



Риcунок 1 – Результат запуска GaussMethod.exe

# ВЫВОДЫ

Таким образом, в ходе выполнения данной лабораторной работы был рассмотрен алгоритм Гаусса для решения систем линейных алгебраических уравнений и написана программа, которая позволяет решать невырожденные квадратные системы уравнений использую данный метод. Листинг программы приведен далее.

# ЛИСТИНГ ПРОГРАММЫ

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws InterruptedException {

Scanner scan = new Scanner(System.*in*);

System.*out*.print("The matrix dimension: ");

int dimension;

while(true){ //ввод размера

try {

dimension = scan.nextInt();

if (dimension < 2) continue;

break;

} catch (Exception e) {

scan.next();

System.*out*.print("Input error! Try again: ");

}

}

double[][] AMatrix = new double[dimension][dimension];

System.*out*.println("Enter A matrix:");

for(int i = 0; i < dimension; i++) {

System.*out*.println("New row:");

*inputRow*(AMatrix[i], scan);

}

double[] BMatrix = new double[dimension];

System.*out*.println("Enter B matrix:");

*inputRow*(BMatrix, scan);

System.*out*.println("Your matrices");

*matrixPrint*(dimension, AMatrix, BMatrix);

*gaussMethod*(dimension, AMatrix, BMatrix);

System.*out*.println();

*matrixPrint*(dimension, AMatrix, BMatrix);

scan.close();

Thread.*sleep*(100000);

}

private static double inputValue(Scanner scan) {

double value;

while(true){

try {

value = scan.nextDouble();

break;

} catch (Exception e) {

scan.next();

System.*out*.print("Input error! Try again: ");

}

}

return value;

}

private static void inputRow(double row[], Scanner scan) {

for(int i = 0; i < row.length; i++) {

row[i] = *inputValue*(scan);

}

}

private static void matrixPrint(int dimension, double AMatrix[][], double BMatrix[]) {

for(int i = 0; i < dimension; i++) {

for(int j = 0; j < dimension; j++) {

System.*out*.printf("%4.2f", AMatrix[i][j]);

System.*out*.print("\t");

}

System.*out*.print("\t");

System.*out*.printf("%4.2f", BMatrix[i]);

System.*out*.println();

}

}

private static void gaussMethod(int dimension, double AMatrix[][], double BMatrix[]) {

//проверка, что определитель матрицы не равен 0

double determinant = *determinant*(dimension, AMatrix);

if (determinant == 0) {

System.*out*.println("The determinant = 0");

System.*exit*(-1);

}

for(int i = 0; i < dimension; i++) {

//делим строку так, чтобы по диагонали стояла единица

double diagonalVal = AMatrix[i][i];

*stringDivision*(AMatrix[i], diagonalVal);

BMatrix[i] /= diagonalVal;

for(int j = 0; j < dimension; j++) {

if (i == j) continue;

double factor = AMatrix[j][i];

for(int n = 0; n < dimension; n++) {

AMatrix[j][n] -= AMatrix[i][n] \* factor;

}

BMatrix[j] -= BMatrix[i] \* factor;

}

}

}

private static double determinant(int dimension, double matrix[][]) {

double matrixDeterminant = 0;

if(dimension > 2) {

for(int i = 0; i < dimension; i++) {

List<double[]> submatrix = new ArrayList<>();

for(int j = 0; j < dimension; j++) {

if(j != i) {

submatrix.add((double[])Arrays.*copyOfRange*(matrix[j], 1, matrix[j].length));

}

}

matrixDeterminant += Math.*pow*(-1, i) \* matrix[i][0] \*

*determinant*(dimension-1, submatrix.toArray(new double[dimension-1][dimension-1]));

}

return matrixDeterminant;

}

else {

return matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];

}

}

private static void stringDivision(double matrix[], double divider) {

for(int i = 0; i < matrix.length; i++)

matrix[i] /= divider;

}

}