Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 1

з дисципліни «Числові методи»

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

Виконав:

студент 3 курсу

групи КС-22

Безрук Юрій Русланович

Перевірила: доцент

Чуб Ольга Ігорівна

Харків – 2020

# ХОД РАБОТЫ

Основной целью данной работы является исследование алгоритма метода гаусса для решения систем линейных уравнений и его программная реализация.

Входными данными являются любые невырожденные системы линейных алгебраических уравнений, представленные в форме матрицы коэффициентов при переменных и вектора правых частей системы. Решение представляется в виде вектора составленного из координат-значений переменных, удовлетворяющих системе.

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.

Если рассматривать алгоритм пошагово, то он заключается в следующем: для каждой *i*-той строки матрицы на позиции элемента *k=i* необходимо получить единицу. Для этого вся строка матрицы делится на этот элемент *ai,i*. После этого, для каждой *j*-той строки такой что *j ≠ i* выполняется следующее: от этой строки отнимается *i*-тая строка, умноженная на *i*-тый коэффициент *j*-той строки, таким образом образовывая в нужной позиции ноль. Таким образом, за одну итерацию внешнего цикла алгоритм преобразует первый столбец матрицы к нужному виду. Алгоритм повторяется для каждого столбца.

Для нахождения определителя матрицы используется метод разложения матрицы на алгебраические дополнения.

Программа для реализации данного метода была написана на языке Java. Для удобства использования и архитектуры основные компоненты модели были представлены в виде классов: строка матрицы Row и система Matrix. Класс Row инкапсулирует массив коэффициентов отдельной и содержит в себе все необходимые операции для их обработки: возможности сложения и вычитания с другой строкой, а также умножение/деление строки на заданный коэффициент.

Класс Matrix содержит в себе массив строк матрицы и основные операции системы, такие как нахождение определителя. Именно в этом классе расположен метод solve(), решающий систему используя метод гаусса. Для удобства, процесс решения системы пошагово записывается в файл countingProcess.txt.

Для удобства пользования в приложении реализован графический интерфейс для ввода коэффициентов: для каждого коэффициента предусмотрено свое поле для ввода. Поскольку алгоритм способен решать квадратные системы любых размеров, в графическом интерфейсе предусмотрено поле для ввода любых размерностей системы. После ввода нужного значения размерности необходимо нажать клавишу Enter и поле для ввода коэффициентов перестроится под заданный размер. Внизу фрейма расположены две кнопки: Solve и Exit. Нажатие кнопки Exit завершает программу, а кнопки Solve – запускает процесс решения и выводит результат в поле Result. Поля для ввода коэффициентов и размерности снабжены валидацией – при вводе неправильного значения откроется всплывающее окно, уведомляющее об этом, и процесс решения / перестроения не начнется.

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

В качестве значений по умолчанию в фрейме установлены коэффициенты предоставленные преподавателем.

Программа скомпилирована в исполняемый файл lab1\_GaussMethod.exe. В процессе его выполнения в директории где он находится будет обновлен/создан файл countingProcess.txt, содержащий подробный ход решения.

# 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

Класс Row, инкапсулирующий строку матрицы:

**import** java.util.Arrays;

**public** **class** Row **implements** Cloneable{

**private** **double**[] row;

**public** Row(**double**[] arr) {

row = Arrays.*copyOf*(arr, arr.length);

}

**public** **int** size() {

**return** row.length;

}

**public** Row add(Row otherRow) {

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

row[i] += otherRow.row[i];

**return** **this**;

}

**public** Row subtract(Row otherRow) {

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

row[i] -= otherRow.row[i];

**return** **this**;

}

**public** Row multiply(**double** coeff) {

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

row[i] \*= coeff;

**return** **this**;

}

**public** Row divide(**double** coeff) {

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

row[i] /= coeff;

**return** **this**;

}

**public** **double** elem(**int** index) {

**return** row[index];

}

**public** Row subRow(**int** from, **int** to) {

**return** **new** Row(Arrays.*copyOfRange*(row, from, to));

}

@Override

**protected** Object clone() **throws** CloneNotSupportedException {

**return** **new** Row(row);

}

@Override

**public** String toString() {

String result = "";

**for** (**double** elem : row) {

result += String.*format*("%.2f ", elem);

}

**return** result + "\n";

}

}

Класс Matrix, инкапсулирующий возможности системы уравлений:

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** Matrix {

**private** Row[] system;

**private** java.io.FileWriter writer;

**public** Matrix(Row[] arr) {

system = Arrays.*copyOf*(arr, arr.length);

**try** {

writer = **new** java.io.FileWriter("countingProcess.txt");

}**catch**(java.io.IOException e) {

e.printStackTrace();

}

}

**public** **int** size() {

**return** system.length;

}

**public** Row row(**int** index) {

**return** system[index];

}

**public** **double** determinant() {

**if**(size() < 1) {

**return** 0;

}

**else** **if**(size() == 1) {

**return** row(0).elem(0);

}

**else** **if**(size() == 2) {

**return** (row(0).elem(0) \* row(1).elem(1)) - (row(0).elem(1) \* row(1).elem(0));

}

**else** {

**double** result = 0;

**for**(**int** i = 0; i < size(); i++) {

List<Row> rows = **new** ArrayList<>();

**for** (**int** j = 0; j < size(); j++) {

**if**(i != j) {

rows.add(row(j).subRow(1, row(j).size() - (row(j).size() > **this**.size() ? 1 : 0) ));

}

}

Matrix minor = **new** Matrix(rows.toArray(**new** Row[rows.size()]));

result += Math.*pow*(-1, i) \* row(i).elem(0) \* minor.determinant();

}

**return** result;

}

}

**public** **double**[] solve() {

log();

**for**(**int** i = 0; i < size(); i++) {

row(i).divide(row(i).elem(i));

log();

**for** (**int** j = 0; j < size(); j++) {

**if**(i != j) {

Row next = **null**;

**try** {

next = (Row) row(i).clone();

} **catch** (CloneNotSupportedException e) {}

row(j).subtract(next.multiply(row(j).elem(i)));

log();

}

}

}

**double**[] result = **new** **double**[size()];

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

result[i] = row(i).elem(row(i).size() - 1);

}

**try** {

writer.close();

} **catch** (IOException e) {

e.printStackTrace();

}

**return** result;

}

**private** **void** log() {

**try** {

**for** (Row row : system)

writer.write(row.toString());

writer.write("\n");

} **catch** (java.io.IOException e) {

e.printStackTrace();

}

}

}

Классы графического интерфейса реализованы на основе библиотеки swing и не обладают функционалом, необходимым к представлению в данной работе, поэтому в листинге не приводятся, но предоставляются вместе со всеми исходными и исполняемыми файлами преподавателю.