УНИВЕРСИТЕТ ИТМО
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Отчёт по лабораторной работе №1
Вариант Гл

Выполнил Ощепков Артём,
группа P3202
Преподаватель Исаев И.В.

Санкт-Петербург 2017

Текст задания

Создать программу, решающую СЛАУ методом Гаусса с выбором главного элемента

Размерность n<=20 (задается из файла или с клавиатуры - по выбору конечного **пользователя**)

Должно быть предусмотрено чтение исходных данных как из файла, так и ввод с клавиатуры.

Должна быть реализована возможность ввода коэффициентов матрицы как с клавиатуры, так и из файла. Также предусмотреть случайные коэффициенты.

Должно быть реализовано:

- Вычисление определителя
- Вывод треугольной матрицы
- Столбец неизвестных
- Столбец невязок

Описание метода

Приводим основную матрицу системы к треугольному виду, производя над расширенной матрицей элементарные преобразования так, чтобы коэффициент умножения строки/столбца получались делением на большие по модулю числа (конкретная реализация ищет максимальный по модулю элемент в строке, однако существует мнение, что достаточно находить любой немаленький элемент), что позволяет сократить погрешность вычисления.

Текст разработанной программы

```
public class Matrix implements Observable {
// Использование паттерна «наблюдатель» позволяет
// Создать анимацию, не портя математический код
    private List<vt.smt.GUI.Observer.Observer> observers = new LinkedList<>();


    @Override
    public void subscribe(Observer observer) {
        observers.add(observer);
    }
```

```
public void triangulate(){
    if(isTriangle) // If the matrix is already triangle there's no reason to perform this function
        return;
    // We imitate closing strokes a Lists remembering which of its we need to skip
    Collection<Integer> strokesToSkip = new TreeSet<>();
    for(int loop = 0; loop < getY();loop++) {
        int mainPositionY = finMaxAbsInColumn(loop,strokesToSkip);
        // Сообщение для наблюдателя.
        noticeAll(new ChooseCeil(new Pair<int,int>(mainPositionY,loop), "mainElementCeil"),40);
        strokesToSkip.add(mainPositionY);// We don't need to mul the main stroke by itself
        Double main = m[mainPositionY][loop];
        for (int i = 0; i < getY(); i++) {
            if (strokesToSkip.contains(i)) continue;
            // Coefficient to mul each column by this
            Double factor = -get(i, loop) / main;
            for (int j = 0; j < getX(); j++) {
                // Прибавляем почленно главную строку, умножив её на коэффициент
                m[i][j] += get(mainPositionY, j) * factor;
                if(Math.abs(m[i][j]) < 10.440892098500626E-12 )
                    m[i][j] = 0.0;
                noticeAll(new ChangeCeil(new Pair<Integer, Integer>(i,j),
                            Double.toString(m[i][j])),50);
            }
        }
        // we need to close the column to skip it later
        //columnsToSkip.add(mainPos.getValue());
    }
    normalize();
    isTriangle = true;
  }
```

```
public void normalize(){
        // Обязательно есть колонки с количеством нулей от n до n-1
        int k = 0;
        for(int i = getX(); i >= 0;i--)
            swapStrokes(
                    findStrokeWithoutZeros(i),
                    k++
            );
    }
// Если берём расширенную матрицу
// Обратный ход
public Double[] getAnswersSLAU(){
    // Иксов ровно столько, сколько стоблцов - 1
    Double x[] = new Double[getX()-1];
    // x1 = y[i]/coeff;
    // x2 = (y[i] - x2*coeffx2) / coeff
    //          |lin comb|
    // Подсчитываем ответы,  начиная с конца
    for(int i = getY()-1;i>=0;i--){
        double linCombSum = 0.0;
        for (int j = i+1; j < x.length; j++) {
            linCombSum += m[i][j]*x[j];
        }
        x[i] = (m[i][getX()-1] - linCombSum) / m[i][i];
    }
    return x;
}

public Double det() throws IllegalArgumentException{
```

```
    if(!isSquare())
        throw new IllegalArgumentException("A matrix must be square to calculate the det of it");
    triangulate();
    Double det = 1.0;
    for(int i = 0; i < getX();i++)
        det *= m[i][i];
    return det;
}
}
```

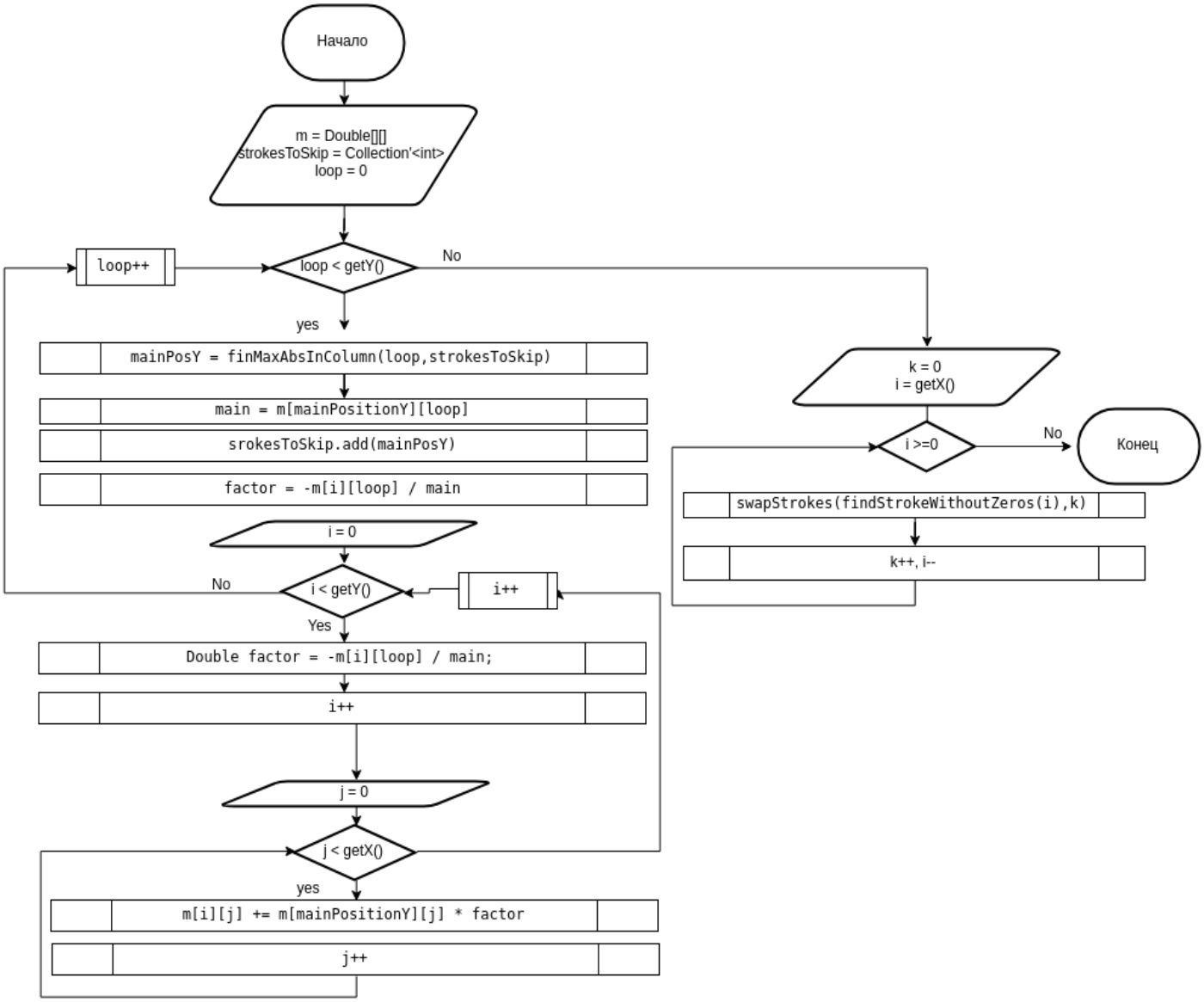
Результаты тестирования

Входные данные(4x5):

1 2 3 4 5
2 4 1 2 3
3 4 5 7 7
9 1 2 4 5
Выходные:
9.0 1.0 2.0 4.0 5.0
0.0 3.7777777777777777 0.5555555555555556 1.1111111111111112 1.8888888888888888
0.0 0.0 3.7941176470588234 4.588235294117648 3.500000000000001
0.0 0.0 0.0 -0.023255813953488857 1.1937984496124021
Det = -3.0000000000000000000626
x1 = 8.66 x2 = 62.9 x4 = -51.
Неувязки: [0.0, 3.1086244689504383E-15, 8.881784197001252E-16, 0.0]
Входные2(6x7):
1 2 3 4 5 6 7
-7 124.1 0.4 13 -14.5 6 7.994
-2 3 -15 -99.1 -0.0123 14 5
1 -2 3 3.77 5 6 6
17 -18 19 1.2 2.2 4.3 1
2 4 5 1 -24.2 222 8882
Выходные2:
17.0 -18.0 19.0 1.2 2.2 4.3 1.0
0.0 116.68823529411765 8.223529411764705 13.494117647058824 -13.594117647058823 7.770588235294118 8.405764705882353
0.0 0.0 -12.826889146544337 -99.06086101729092 0.3493171800171397 14.447124061097949 5.054085799264001
0.0 0.0 0.0 -17.87056754451811 -23.68257035571258 223.71506128977745 8882.361140591163
0.0 0.0 0.0 0.0 19.71120544283874 -132.720315492538 -5580.550226713819
0.0 0.0 0.0 0.0 0.0 9.493655119411414 367.287675142783
det = 8.50907429098247E7
x: [92.58480492115477, -0.7423232610490788, -90.71587375211557, 17.25779584433093, -22.621983343018034, 38.687699365842775]

Неувязки: [0.0, 5.329070518200751E-15, 7.105427357601002E-14, 0.0, 0.0, 0.0]

Блок-схема



Выводы

Нужно было писать математический код. А я писал гуи, дабы получить удовольствие и творчество от программирования. В итоге потратил десятки кода и в разы больше времени на выполнение лабы, но зато удовлетворён результатом. Я научился использовать CSS в JavaFx, вспомнил многие языковые конструкции языка Java, несколько паттернов ООП.

Писать математический — сложно, кропотливо. Искать ошибки - головотерзающе за волосы: её место обычно не очевидно.

Оно заработало, но, естественно, с ограничениями. Надеюсь, мелкие недостатки продукта и знания теории покроются красопетностью самой программы и творческого усердия, с которой она была изобретена.