НИУ ИТМО

ITMO UNIVERSITY

Лабораторная работа №1
Метод Гаусса

Преподаватель:
Перл О.В.

Выполнил:
Куприянов А.А.

Описание метода

Метод Гаусса состоит из двух основных этапов: прямой ход и обратный.
Прямой ход - это процесс нахождения коэффициентов треугольной системы, а обратный - это процесс получения неизвестных значений переменных.

Прямой ход

1. Допустим, у нас есть система трёх уравнений с тремя неизвестными

$$(1)\quad \begin{array}{rcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & a_{14} \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & = & a_{24} \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & = & a_{34} \end{array}$$

Пусть $a_{11} != 0$ (ведущий элемент). Исключим неизвестную x_1 из системы.

Разделив коэффициенты первого уравнения сиситемы (1) на a_{11}, мы получим:

$$(2)\quad x_1 + b_{12}x_2 + b_{13}x_4 = b_{15} \text{ , где } b_{1j} = \frac{a_{1j}}{a_{11}}$$

Пользуясь этим уравнением (2) мы можем исключить из системы (1) неизвестную x_1. Нужно из второго уравнения системы (1) вычесть уравнение (2), умноженное на a_{21}, из третьего уравнения системы вычесть уравнение (2), умноженое на a_{31}

В результате получим систему уже двух уравнений:

$$\begin{array}{rcrcl} a^{(1)}_{22}x_2 & + & a^{(1)}_{23}x_3 & = & a^{(1)}_{24} \\ a^{(1)}_{32}x_2 & + & a^{(1)}_{33}x_3 & = & a^{(1)}_{34} \end{array}$$

Выполнив такую операцию еще раз, мы придем к уравнению:

$$x_3 = \frac{a^{(2)}_{34}}{a^{(2)}_{33}} = b^3_{34}$$

Обратный ход

2. Далее, мы можем последовательно определить остальные неизвестные из уравнений:

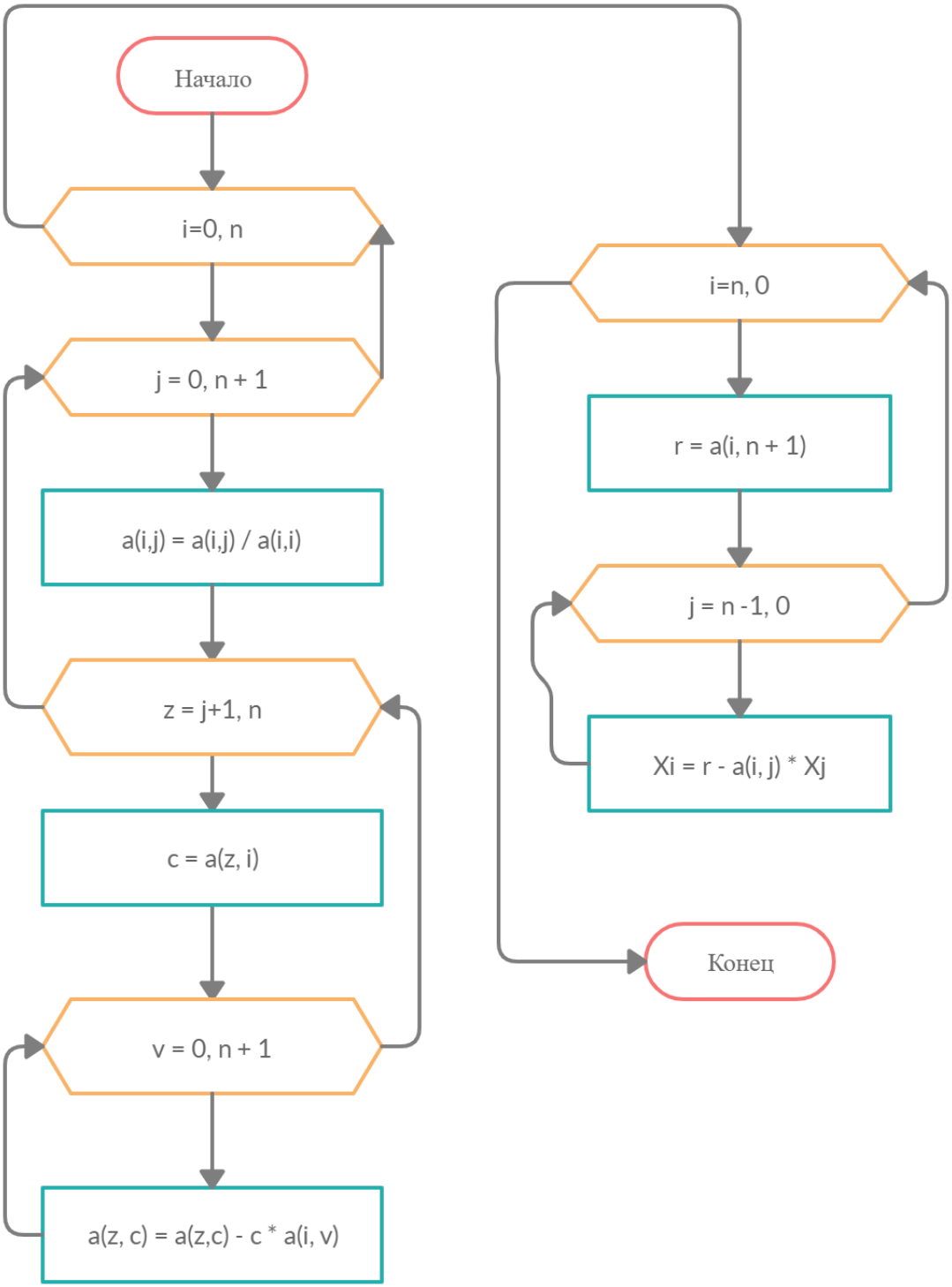
$x_2 = b^{(1)}_{24} - b^{(1)}_{23}x_3$
$x_1 = b_{14} - b_{13}x_3 - b_{12}x_2$

Таким образом, процесс решения линейной системы по методу Гаусса - это построение эквивалентной системы, имеющей треугольную матрицу.

Необходимое и достаточное условие

Необходимым и достаточным условием применимости метода является, то что ведущие элементы не равны нулю

Блок схема


Начало
i=0, n
j = 0, n + 1
a(i,j) = a(i,j) / a(i,i)
z = j+1, n
c = a(z, i)
v = 0, n + 1
a(z, c) = a(z,c) - c * a(i, v)
i=n, 0
r = a(i, n + 1)
j = n -1, 0
Xi = r - a(i, j) * Xj
Конец

Исходный код

1. Прямой ход

```
public Matrix decompose(Matrix matrix) {
    for (int vIndex = 0; vIndex < matrix.getYSize(); vIndex++){
        float leadElement = matrix.getElement(vIndex, vIndex);
        float[] equation = makeEquation(matrix, leadElement, vIndex);
        bottomSubtract(matrix, vIndex, equation);
    }
    return matrix;
}

private float[] makeEquation(Matrix matrix, float leadElement, int vIndex){
    float[] equation = new float[matrix.getXSize()];
    for (int hIndex = 0; hIndex < matrix.getXSize(); hIndex++) {
        equation[hIndex] = matrix.getElement(vIndex, hIndex)/leadElement;
        matrix.setElement(vIndex, hIndex, equation[hIndex]);
    }
    return equation;
}

private void bottomSubtract(Matrix matrix, int vIndex, float[] equation){
  for (int z = vIndex + 1; z < matrix.getYSize(); z++) {
   float nextLeadElem = matrix.getElement(z, vIndex);
   for (int hIndex = 0; hIndex < matrix.getXSize(); hIndex++) {
     float value = matrix.getElement(z, hIndex)-nextLeadElem*equation[hIndex];
     matrix.setElement(z, hIndex,value);
   }
  }
}
```

Так как определитель матрицы при делении ее строки на ведующий элемент тоже делится на этот элемент и учитывая, то что при вычитании из строк строки он не меняется - мы можем найти определитель матрицы во время прямого прохода:

$$det(a_{ij}^0) = a_{11}^{(0)} a_{22}^{(1)} ... a_{nn}^{(n-1)}$$

2. Обратный ход

```
private float[] backSubstitution(Matrix matrix){
  float[] values = new float[matrix.getYSize()];
  for (int hIndex = matrix.getYSize() - 1; hIndex >= 0; hIndex--) {
      float rightValue = matrix.getElement(hIndex, matrix.getXSize() - 1);
      values[hIndex] = findVariable(matrix, hIndex, rightValue, values);
  }

  return values;
}

private float findVariable(
Matrix matrix,int hIndex,float initialValue,float[] previousVariables
) {
  float value = initialValue;
  for (int vIndex = matrix.getXSize() - 2; vIndex >= 0; vIndex--) {
    value =value-matrix.getElement(hIndex, vIndex)*previousVariables[vIndex];
  }
  return value;
}
```

3. Пример применения

$$
\begin{cases}
13x_1 + 15x_2 + 13x_3 + 18x_4 + 13x_5 = 16 \\
12x_1 + 12x_2 + 16x_3 + 17x_4 + 13x_5 = 15 \\
14x_1 + 18x_2 + 16x_3 + 19x_4 + 17x_5 = 14 \\
18x_1 + 16x_2 + 18x_3 + 20x_4 + 18x_5 = 16 \\
19x_1 + 15x_2 + 18x_3 + 13x_4 + 15x_5 = 18
\end{cases}
$$

Решение:
x1: 0.789790, x2: 0.696697, x3: 0.818318, x4: 0.575075, x5: -1.977477,

Невязка:
2.384186e-07, 3.725290e-08, 0.000000e+00, 0.000000e+00, 0.000000e+00

Вывод

Как видно из реализации, в этом методе, если в прямом ходе окажется ведущий элемент равный нулю, то этот вариант метода Гаусса не пригоден. Альтернативным вариантом является метод Гаусса с выбором главного элемента. Это более универсальный метод, но медленее. По сути, метод Гаусса, является частным случаем метода главных элементов

Рассмотренный метод также имеет недостаток в тех случаях, когда ведущие элементы будут иметь маленькую величину. Из-за специального представления чисел в ЭВМ существует машинная погрешность, что катастрафически может увеличить погрешность при очень малых значениях элементов. Так, например, в этой лабораторной работе была использована JDK 1.8 HotSpot, которая в своей виртуальной машине поддерживала float value set: 24 бит под мантиссу и еще 8 под знак и смещенную экспоненту, итого 4 байта