Университет ИТМО

Факультет ПИиКТ

Вычислительная математика

Лабораторная работа №1

Вариант – метод Гаусса

Выполнила: Наумова Надежда

Группа P3201

Преподаватель: Перл Ольга Вячеславовна

Санкт-Петербург

2020 г.

### Цель работы

Реализовать метод Гаусса для решения СЛАУ с единственным набором решений заданной размерности.

### Описание использованного метода

Метод Гаусса является точным методом решения систем линейных алгебраических уравнений. Метод Гаусса заключается в том, чтобы привести исходную матрицу к треугольному виду таким образом, чтобы под главной диагональю были нули, а затем будем двигаться от последнего уравнения к первому и последовательно вычислять неизвестные.

Реализация метода Гаусса состоит из двух основных этапов (при условии, что мы знаем, что определитель нашей матрицы коэффициентов при неизвестных отличен от 0 – иначе мы не решим нашу систему): прямого хода и обратного хода.

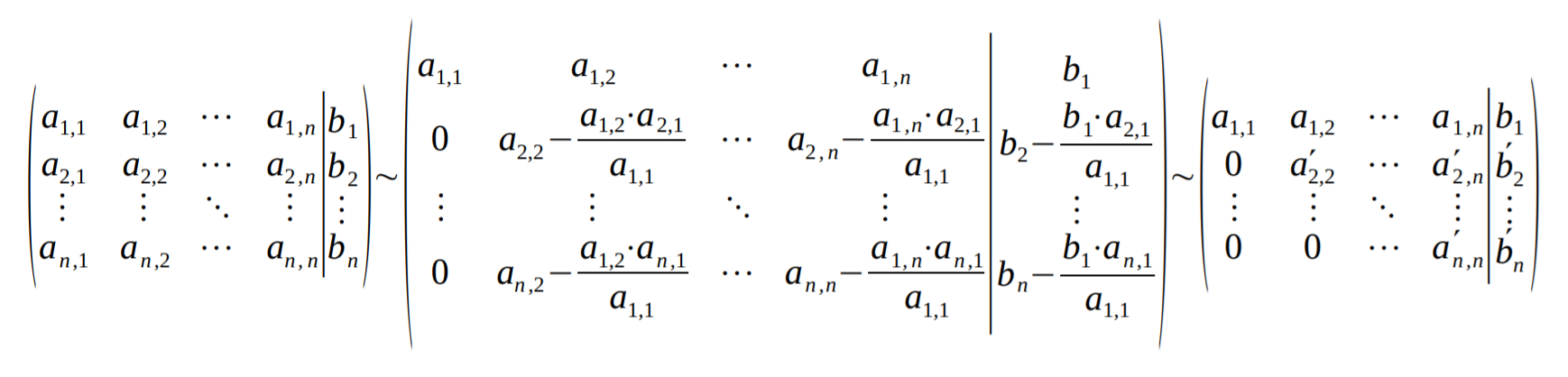
Прямой ход заключается в том, чтобы последовательно исключать неизвестные из уравнений системы. Первым делом исключаем x1 из второго, третьего, …, n-ого уравнений. Затем исключаем x2 из третьего, четвертого, .., n-ого уравнений. То есть, на i-ом шаге следует исключать xi из всех уравнений от (i+1)-го до n-го. Прямой ход производится до тех пор, пока до тех пор, пока в n-ом уравнении не останется единственный член с неизвестным xn.

Обратный ход заключается в том, что, двигаясь от n-ого уравнения к первому, будем последовательно вычислять неизвестные. Сначала вычисляем xn из последнего уравнения (n-ого) – благодаря нахождению треугольной матрицы имеем уравнение ann’ ∙ xn = b’n , откуда легко находим xn, далее подставляем найденный xn в (n – 1)-ое уравнение и находим xn-1 и так далее. Соответственно, последним находим x1 из 1-го уравнения.

Важно помнить то, что, когда в ходе реализации прямого метода мы исключаем неизвестные, используя коэффициент матрицы с индексом (i, i), но он может оказаться равным нулю. В этом случае нужно поменять местами эту строку матрицы с любой последующей строкой, в которой элемент, стоящий на другой строке, но в том же столбце, будет отличным от нуля.

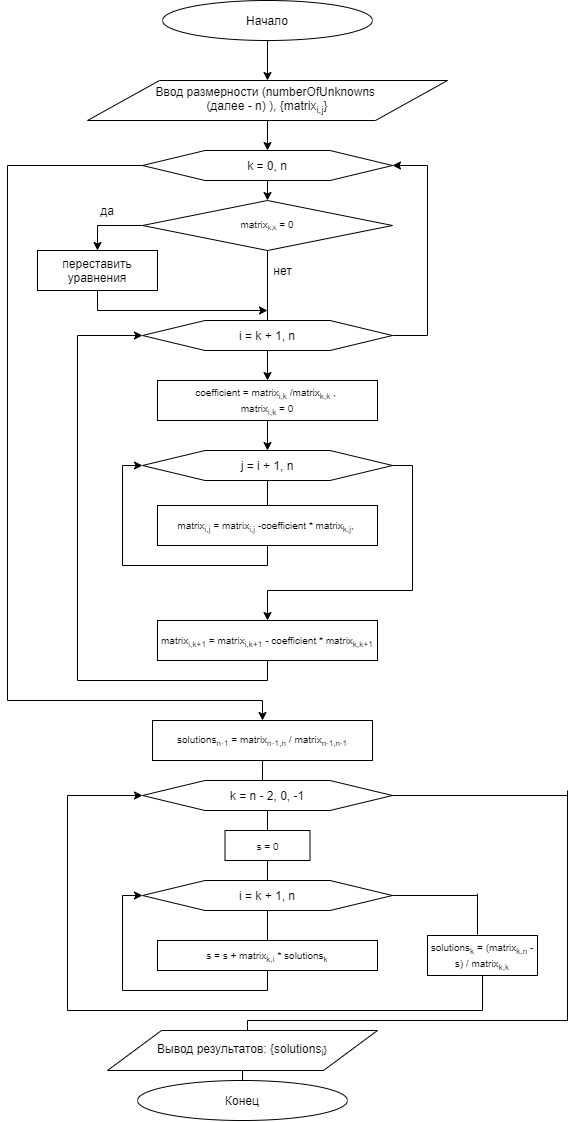
Как происходит исключение неизвестных из уравнений? На первом шаге исключаем x1 из последующих уравнений: из второго – умножаем первое уравнение на (-a21/a11) и прибавляем ко второму, из третьего - умножаем первое уравнение на (-a31/a11) и прибавляем к третьему, …, из n-ого – умножаем первое уравнение на (-an1/a11) и прибавляем к n-ому. На втором шаге исключаем x2 из последующих уравнений: из третьего - умножаем первое уравнение на (-a32/a22) и прибавляем к третьему, …, из n-ого – умножаем второе уравнение на (-an2/a22) и прибавляем к n-ому. Обобщая сказанное, на i-ом шаге производится исключение xi из уравнений, следующих в матрице за i-ым: каждое из этих уравнений (пробегаем по ним циклом по j) домножается на (-aji/aii) и прибавляется к i-ому.

Прямой ход:



Обратный ход:

### Блок-схема



### Листинг численного метода

// Algorithm.java

// straight and return run, getting solutions

private double[] solve() {  
 straightRun();  
 return returnRun();  
}  
  
private void straightRun () {  
 int count = 1;  
 boolean isColumnZero = true;  
 for (int k = 0; k < numberOfUnknowns; k ++) {  
 if (linearSystem.getEquationCoefficient(k, k) == 0) {  
 for (int i = k; i < numberOfUnknowns; i++)  
 if (linearSystem.getEquationCoefficient(i, k) != 0) {  
 swap(linearSystem, 0, i);  
 count = -count;  
 break;  
 }  
 isColumnZero = false;  
 }  
 for (int i = k + 1; i < numberOfUnknowns; i ++) {  
 double coefficient = linearSystem.findCoefficient(linearSystem.getEquationCoefficient(k, k),  
 linearSystem.getEquationCoefficient(i, k));  
 modifyRow(linearSystem, i, k, coefficient);  
 }  
 }  
 determinant = new Determinant(linearSystem).calculateDeterminant(count, isColumnZero);  
}

private double[] returnRun() {  
 double[] solutions = new double[numberOfUnknowns];  
 solutions[numberOfUnknowns - 1] =  
 linearSystem.getEquationCoefficient(numberOfUnknowns - 1, numberOfUnknowns) /  
 linearSystem.getEquationCoefficient(numberOfUnknowns - 1, numberOfUnknowns - 1);  
  
 for (int k = numberOfUnknowns - 2; k >= 0; k --) {  
 for (int i = k + 1; i < numberOfUnknowns; i ++)  
 solutions[k] += -linearSystem.getEquationCoefficient(k, i) \* solutions[i];  
 solutions[k] += linearSystem.getEquationCoefficient(k, numberOfUnknowns);  
 solutions[k] /= linearSystem.getEquationCoefficient(k, k);  
 }  
 return solutions;  
}

private void modifyRow(LinearSystem system, int strIndex1, int strIndex2, double coef) {  
 for (int i = 0; i < system.getNumberOfUnknowns() + 1; i ++) {  
 double current = system.getEquationCoefficient(strIndex1, i);  
 system.setEquationCoefficient(strIndex1, i,  
 system.getEquationCoefficient(strIndex2, i) \* coef + current);  
 }  
}

// Residual.java

// calculation of the residuals

private double[] calculateResiduals(double[][] system, double[] solutions) {  
 this.residuals = new double[solutions.length];  
 for (int i = 0; i < system.length; i++) {  
 for (int j = 0; j < system.length; j++)  
 residuals[i] += solutions[j] \* system[i][j];  
 residuals[i] -= system[i][system.length];  
 }  
 return residuals;  
}

// Determinant.java

//calculation of the determinant

double calculateDeterminant(int count, boolean isZero) {  
 if (isZero){  
 double determinant = 1;  
 for (int i = 0; i < system.getNumberOfUnknowns(); i ++) {  
 determinant \*= system.getEquationCoefficient(i, i);  
 }  
 determinant \*= count;  
 return determinant;  
 } else  
 return 0;  
}

### Примеры

Пример 1:

Enter the command >>> enter;

Enter the number of unknowns >>> 3

In each string enter the coefficients of the system >>>

2 -3 1 -1

5 2 -1 0

1 -1 2 3

Determinant: 32.0

Triangular matrix:

2,000 -3,000 1,000 -1,000

0,000 9,500 -3,500 2,500

0,000 0,000 1,684 3,368

Solutions:

x1 = 0.0

x2 = 1.0

x3 = 2.0

Residuals:

0.0

0.0

0.0

Правильные ответы:

x1 = 0.0

x2 = 1.0

x3 = 2.0

Пример 2:

Enter the command >>> enter;

Enter the number of unknowns >>> 3

In each string enter the coefficients of the system >>>

9 3 -1 -5

-2 -6 -1 24

1 -3 12 3

Determinant: -618.0

Triangular matrix:

9,000 3,000 -1,000 -5,000

0,000 -5,333 -1,222 22,889

0,000 0,000 12,875 -10,750

Solutions:

x1 = 0.718

x2 = -4.1

x3 = -0.835

Residuals:

0.0

3.552 \* 10 ^ (-15)

0.0

Правильные ответы:

x1 = 0.718446601942

x2 = -4.100323624596

x3 = -0.834951456311

Пример 3:

Enter the command >>> enter;

Enter the number of unknowns >>> 6

In each string enter the coefficients of the system >>>

100 1 -6 2 1 4 0.5

1 40 3 -8 4 3 2.542

2 7 -45 4 -7 3 1

1 3 1 25 3 -5 1

4 7 5 5 99 6 4

1 1 1 1 1 -7 -8

Determinant: 3.0875855159999995 \* 10 ^ (9)

100,000 1,000 -6,000 2,000 1,000 4,000 0,500

0,000 39,990 3,060 -8,020 3,990 2,960 2,537

0,000 0,000 -45,414 5,360 -7,716 2,403 0,547

0,000 0,000 0,000 25,678 2,550 -5,217 0,815

0,000 0,000 0,000 0,000 96,783 6,970 3,377

0,000 0,000 0,000 0,000 0,000 -6,841 -8,118

x1 = -0.042

x2 = 0.029

x3 = 0.092

x4 = 0.278

x5 = -0.051

x6 = 1.187

Residuals:

8.881 \* 10 ^ (-16)

0.0

-4.440 \* 10 ^ (-16)

8.881 \* 10 ^ (-16)

-1.332 \* 10 ^ (-15)

-1.776 \* 10 ^ (-15)

Правильные ответы:

x1 = - 0.04228277564

x2 = 0.029331777388

x3 = 0.092137496810

x4 = 0.277882688523

x5 = - 0.05056718309

x6 = 1.186643143426

Пример 4:

Enter the command >>> enter;

Enter the number of unknowns >>> 6

In each string enter the coefficients of the system >>>

-88 77 -4 -30 97 11 -43

79 -12 77 -71 -48 43 -76

30 -93 32 40 -71 -88 -34

43 -16 71 20 -85 5 16

-43 17 22 -2 -2 -1 -50

-6 -65 67 -3 -81 6 -53

Determinant: 6.591487947499998 \* 10 ^ (10)

Triangular matrix:

-88,000 77,000 -4,000 -30,000 97,000 11,000 -43,000

0,000 57,125 73,409 -97,932 39,080 52,875 -114,602

0,000 0,000 116,414 -84,660 7,732 -22,466 -182,571

0,000 0,000 0,000 72,416 -55,136 -1,679 103,073

0,000 0,000 0,000 0,000 -27,983 22,778 -11,153

0,000 0,000 0,000 0,000 0,000 55,582 87,781

Solutions:

x1 = 0.954

x2 = -0.714

x3 = 0.619

x4 = 2.742

x5 = 1.684

x6 = 1.579

Residuals:

-1.421 \* 10 ^ (-14)

2.842 \* 10 ^ (-14)

5.684 \* 10 ^ (-14)

-1.421 \* 10 ^ (-14)

7.105 \* 10 ^ (-15)

8.526 \* 10 ^ (-14)

Правильные ответы:

x1 = 0.954432831268

x2 = -0.71423282467

x3 = 0.618836076541

x4 = 2.742192852883

x5 = 1.684071015818

x6 = 1.579296182655

### Вывод:

Метод Гаусса, являющийся точным методом решения СЛАУ, использует формулы для вычисления неизвестных, а значит, позволяет получить решение за конечное число операций. Преимуществом метода Гаусса является его универсальность – этим методом можно решить подавляющее количество линейных систем. Недостатки метода Гаусса: один из них связан с тем, что требуется хранить в памяти всю матрицу, а при большой размерности это будет занимать много места, другой заключается в накапливании на каждом шаге погрешностей – вычисления на следующем шаге используют результаты предыдущих. Метод Гаусса следует применять при решении систем с определителем, не близким к нулю и матрицей, которая не будет сильно разреженной (иначе большое кол-во нулей в матрице придется хранить в памяти и производить над ними действия).

Если сравнивать метод Гаусса и метод Гаусса с выбором главного элемента, то в реализации различий будет немного: во втором методе добавится то, что на каждом шаге мы будем переставлять уравнения так, чтобы ведущим элементом aii стал максимальный по модулю элемент – по столбцу или по строке. Метод Гаусса с выбором главного элемента – это усовершенствованная версия метода Гаусса, то есть он также является прямым методом и обладает его свойствами, но, в отличие от метода Гаусса, здесь будет медленнее накапливаться погрешность: в методе Гаусса на очередном шаге может оказаться, что на месте «главного элемента» стоит близкое к нулю число, деление на которое даст большую погрешность.

Что касается сравнения метода Гаусса с методом простой итерации и методом Гаусса-Зейделя, то начать следует с того, что последние методы являются не точными, а итерационными, то есть решение ищется не по формулам, а следующим образом: задается начальное приближение, а затем производятся итерации – этапы вычисления, в результате которых получают новые приближения, и так далее до тех пор, пока не получим результат с заданной точностью. В отличии от метода Гаусса, эти методы не требуют хранения всей матрицы, а только нескольких векторов, а также в них не накапливается погрешность, т.к. точность вычислений итераций зависит только от предыдущей итерации.