МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

## **ЛАБОРАТОРНАЯ РАБОТА №4**

по дисциплине

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА»

Вариант “Метод Гаусса-Зейделя”

***Выполнил:***

Студент группы P32312

Лысенко А.К.

***Преподаватель:***

Перл О.В.

Санкт-Петербург, 2023

**Описание метода, расчётные формулы**

Метод Гаусса-Зейделя предназначен для решения СЛАУ. Он является итерационным методом, что означает, что решение будет найдено с какой-то погрешностью. Этот метод подходит для решения СЛАУ с большим количеством неизвестных. Правда, данный метод способен решать далеко не все системы. Данный метод имеет условие сходимости, при невыполнении которого, метод не работает. Условие сходимости представляет собой наличие диагонально преобладания у матрицы. Диагональные преобладание – свойство матрицы, когда на главной диагонали находятся такие элементы, которые по модулю будут больше, чем сумма модулей оставшихся элементов данной строки.

Изображение выглядит как текст

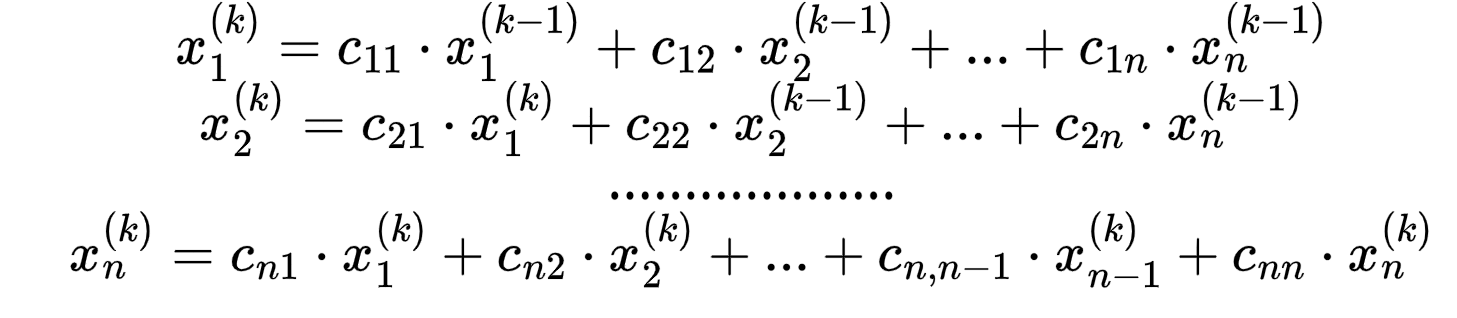
Автоматически созданное описание

Дальнейший метод – вычисление приближений. Приближение – это один из корней заданного СЛАУ, полученный с какой-то погрешностью. За нулевое приближение берутся следующие значения

Изображение выглядит как текст

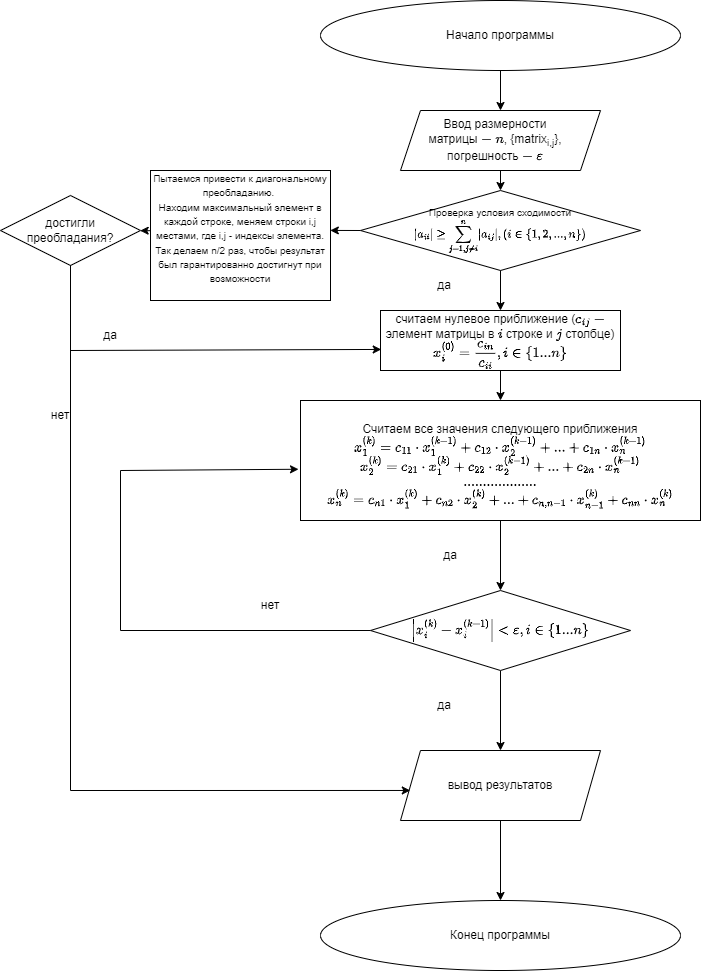
Автоматически созданное описание

– обозначает последний элемент каждой строки, то есть то число, которое стоит справа от равно. Делим мы его на диагональной элемент, при условии, что есть диагональное преобладание. Собственно, нулевое приближение мы сделали. Далее мы циклично повторяем расчёт приближений по формуле до тех пор, пока мы не добьемся нужной точности.



То есть на каждой итерации мы используем значения с прошлой.

**Блок-схема**

****

**Листинг численного метода**

**private boolean** shouldStop() {  
 **int** c = 0;  
 **for** (**int** i = 0; i < matrix.getN(); i++) {  
 stop = stop && (Math.abs(currentApproximation[i] - prevApproximation[i])) >= epsilon;  
 **if** ((Math.abs(currentApproximation[i] - prevApproximation[i])) >= epsilon) {  
 c += 1;  
 }  
 }  
 **if** (c == 0) {  
 **return false**;  
 }  
 **return** stop;  
  
}  
  
**private void** zeroApproximation() {  
 **for** (**int** i = 0; i < matrix.getN(); i++) {  
 currentApproximation[i] = matrix.getB()[i] / matrix.getArray()[i][i];  
 zeroApproximation[i] = matrix.getB()[i] / matrix.getArray()[i][i];  
 prevApproximation[i] = matrix.getB()[i] / matrix.getArray()[i][i];  
 }  
 counter++;  
}  
  
**public void** approximation() {  
 zeroApproximation();  
 **while** (stop) {  
 **double** multiplier;  
 **for** (**int** i = 0; i < matrix.getN(); i++) {  
 **for** (**int** j = 0; j < matrix.getN(); j++) {  
 **if** (i > j) {  
 multiplier = currentApproximation[j];  
 } **else** {  
 multiplier = prevApproximation[j];  
 }  
 **if** (i != j) {  
 currentApproximation[i] += ((-1) \* matrix.getArray()[i][j] \* multiplier) / matrix.getArray()[i][i];  
 }  
 }  
  
 }  
 stop = shouldStop();  
 **double**[] deltas = **new double**[matrix.getN()];  
 **for** (**int** i = 0; i < deltas.length; i++) {  
 deltas[i] = Math.abs(currentApproximation[i] - prevApproximation[i]) / Math.abs(currentApproximation[i]);  
 }  
 **if** (!stop) {  
 drawNewLine(counter, currentApproximation, deltas);  
 }  
 **if** (stop) {  
 System.arraycopy(currentApproximation, 0, prevApproximation, 0, matrix.getN());  
 System.arraycopy(zeroApproximation, 0, currentApproximation, 0, matrix.getN());  
 counter++;  
 }  
 }  
}

**Примеры и результаты работы**

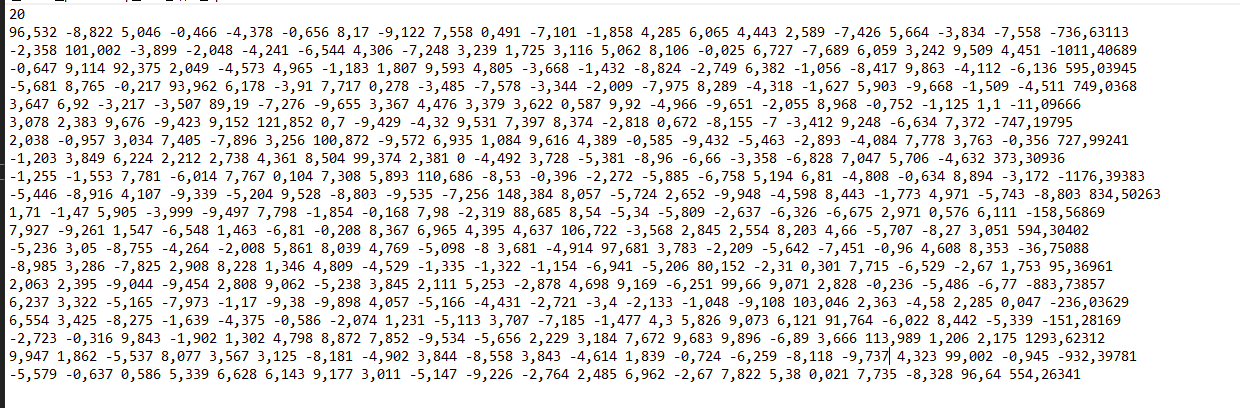
Input file

Изображение выглядит как текст, экран, снимок экрана

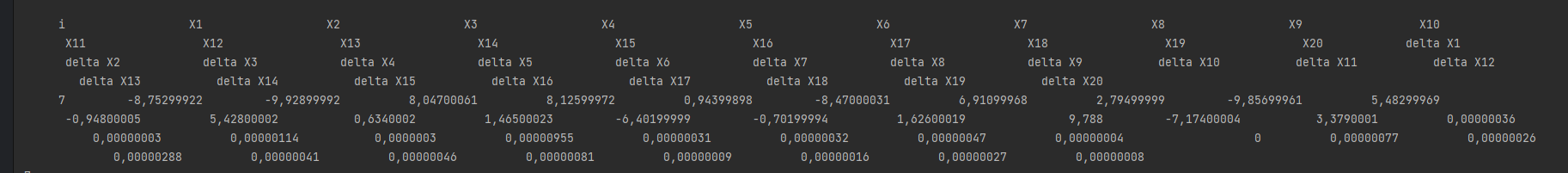
Автоматически созданное описание

Output

Input file



Output



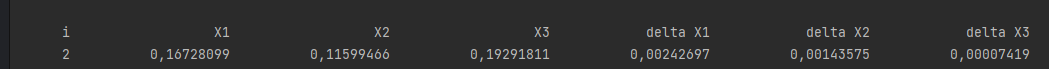
Выходные данные совпали с ожидаемыми в тесте.

Input

Изображение выглядит как текст

Автоматически созданное описание

Output



**Вывод**

Метод Гаусса-Зейделя основан на методе простых итераций. В целом, эти методы почти одинаковы, но есть одно отличие – нам нужно использовать все значения предыдущих приближений. А в методе Гаусса-Зейделя происходит комбинирование старых и новых значений, из-за чего метод Гаусса-Зейделя выигрывает по скорости.

Изображение выглядит как текст, антенна

Автоматически созданное описание

В целом, главное отличие прямых методов от итерационных – точность решения. Первые дают точное решение, а вторые – приближенные. Но эти методы применимы и не являются взаимоисключающими. В жизни достаточно много моделей, в которой нужны, как и точные значения, так и приближенные.