Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет Программной инженерии и компьютерной техники*

**Лабораторная работа №1**

«Системы линейных алгебраических уравнений»

Вариант «Метод Гаусса-Зейделя»

Группа: P32312

Выполнил: Цю Тяньшэн

Проверил:

Перл Ольга Вычеславовна

Санкт-Петербург

2021г

Содержание

[Описание метода, расчётные формулы 3](#_Toc128480243)

[Блок-схема численного метода 4](#_Toc128480244)

[Листинг реализованного численного метода программы 5](#_Toc128480245)

[Примеры и результаты работы программы 7](#_Toc128480246)

[Пример №1 7](#_Toc128480247)

[Пример №2 7](#_Toc128480248)

[Пример №3 «Рандомно сгенерированные числа» 7](#_Toc128480249)

[Вывод 8](#_Toc128480250)

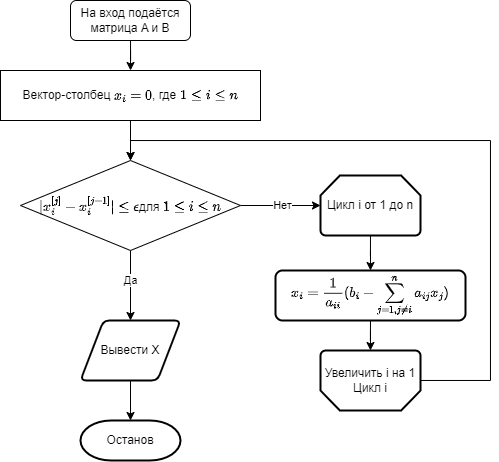
# Описание метода, расчётные формулы

Метод Гаусса-Зейделя является итерационным методом для решения СЛАУ, имеющий один параметр . Достаточное условие для сходимость метода является то, что матрица коэффициентов *A* является диагонально доминантной, т.е.:

Условие для окончания итерационного процесса:

Основная формула для вычисление новых значений на каждом шаге итерации, опирающаяся на «свежие» значения:

# Блок-схема численного метода



# Листинг реализованного численного метода программы

private int getDominantElementInRow(Matrix m, int row) {  
 double sum = 0;  
 for (int i = 0; i < m.getWidth(); i++) sum += Math.*abs*(m.get(row, i));  
  
 for (int i = 0; i < m.getWidth(); i++) {  
 if (sum <= 2 \* Math.*abs*(m.get(row, i))) return i;  
 }  
  
 return -1;  
}  
  
private boolean makeDiagonallyDominant(LinearSystem system) {  
 // Selection sort lol  
 for (int i = 0; i < system.a().getHeight(); i++) {  
 int iDom = getDominantElementInRow(system.a(), i);  
 if (iDom < 0) return false;  
  
 int minRow = i;  
 int minCol = iDom;  
  
 for (int j = i+1; j < system.a().getHeight(); j++) {  
 int jDom = getDominantElementInRow(system.a(), j);  
 if (jDom < 0) return false;  
  
 if (minCol > jDom) {  
 minRow = j;  
 minCol = jDom;  
 }  
 }  
  
 // Check if the element is on the diagonal  
 if (minCol != i) return false;  
  
 system.a().swapRows(i, minRow);  
 system.b().swapRows(i, minRow);  
 }  
  
 return true;  
}  
  
@Override  
public Matrix solve(LinearSystem system, Configuration config) throws LinearSystemSolvingException {  
 Matrix a = system.a();  
 Matrix b = system.b();  
  
 if (a.getWidth() != a.getHeight()) throw new LinearSystemSolvingException("Matrix A of the system is not square. Got " + a);  
 if (a.getHeight() != b.getHeight()) throw new LinearSystemSolvingException("Matrix A does not have the same height as matrix B, got A: " + a + ", B: " + b);  
  
 if (!makeDiagonallyDominant(system)) throw new LinearSystemSolvingException("Matrix A is not and can not be transformed to a diagonally dominant form");  
  
 boolean reachedEpsilon = false;  
 Matrix x = new Matrix(a.getHeight(), 1);  
 int iterations = 0;  
  
 while (iterations++ < *maxIterations* && !reachedEpsilon) {  
 reachedEpsilon = true;  
  
 if (config.getFlag(Configuration.***DEBUG\_FLAG***)) System.***err***.println("Iteration No." + iterations + " error = " + system.getError(x));  
  
 for (int i = 0; i < a.getHeight(); i++) {  
 double s = b.get(i, 0);  
  
 // Get right side sum  
 for (int j = 0; j < a.getWidth(); j++) {  
 // Skip diagonal elements  
 if (i == j) continue;  
  
 s -= x.get(j, 0) \* a.get(i, j);  
 }  
  
 s /= a.get(i, i);  
 if (Math.*abs*(s - x.get(i, 0)) > *eps*) reachedEpsilon = false;  
 x.set(i, 0, s);  
 }  
 }  
  
 return x;  
}

# Примеры и результаты работы программы

## Пример №1

A picture containing text

Description automatically generated

## Пример №2

A picture containing text

Description automatically generated

## Пример №3 «Рандомно сгенерированные числа»

Text

Description automatically generated

# Вывод

После тестирование можно выяснить, что на некоторых размеров данных прямые методы быстрее чем итерационные методы, это происходит, поскольку у прямых методов алгоритмическая сложность , а у итерационных методов , где l – количество итерации, так как мы не можем предсказать, сколько итерации потребует метод, время выполнения методов будет сильно различаться.

А если сравнить итерационные методы между собой, то можно сказать, что метод Гаусса-Зейделя более экономный с точки зрения памяти, так как можно в каждой итерации сразу на ходу переписывать данные в векторе столбце неизвестных, при этом метод Гаусса-Зейделя потребует меньшее количество итерации, чем метод простых итераций, поскольку сразу используются «свежие» значения. Но это одновременно обозначает, что сложнее будет реализовать параллельную обработку данных для метода Гаусса-Зейделя.

Ну и также можно отметить, что из-за строгого условия итерационных методов, они применимы для гораздо меньше случаев, чем прямые методы.