Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №1**

по «Вычислительной математике»

Решение СЛАУ

Вариант 4

Выполнил:

Студент группы P3213

Ваховский П.А.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2022

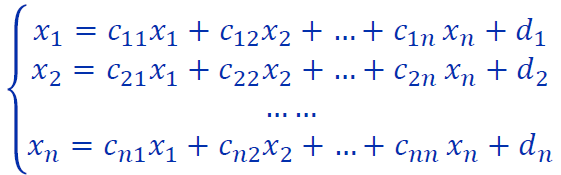
**Цель работы**: написать программу, решающую системы линейных уравнений методом простых итераций

**Задание**

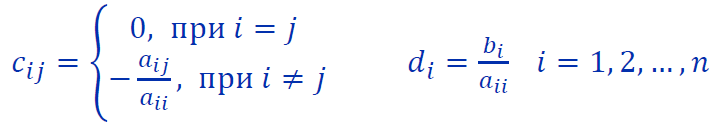
* Точность задается с клавиатуры/файла
* Проверка диагонального преобладания (в случае, если диагональное преобладание в исходной матрице отсутствует, сделать перестановку строк/столбцов до тех пор, пока преобладание не будет достигнуто). В случае невозможности достижения диагонального преобладания - выводить соответствующее сообщение.
* Вывод вектора неизвестных:
* Вывод количества итераций, за которое было найдено решение.
* Вывод вектора погрешностей:

**Описание метода, рабочие формулы:**

Общий вид системы:

**,**

где

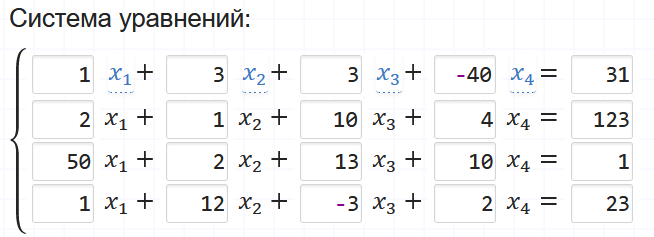




**Листинг класса SimpleIteration:**

import java.util.\*;  
  
public class SimpleIteration {  
 private int N;  
 private List<Double> matrix[], coefficients[], freeMember, prevX, nextX;  
 private List<Integer> betterPlace, checkPlace;  
  
 public SimpleIteration(int N, List<Double> matrix[]){  
 this.N = N;  
 this.matrix = matrix;  
 freeMember = new ArrayList<>();  
 coefficients = new ArrayList[N];  
 betterPlace = new ArrayList<>();  
 checkPlace = new ArrayList<>();  
 for(int i = 0; i < N; i++){  
 double temp = matrix[i].remove(N);  
 freeMember.add(temp);  
 coefficients[i] = new ArrayList<>(matrix[i]);  
 betterPlace.add(-1);  
 checkPlace.add(i);  
 this.matrix[i].add(temp);  
 }  
 }  
 public boolean diagonal(){  
 System.*out*.println("Выполняется проверка диагонального преобладания...");  
 for(int i = 0; i < N; i++){  
 //индекс максимального по модулю коэфициента в строке  
 int maxAbsId = (Collections.*max*(coefficients[i]) > -Collections.*min*(coefficients[i]) ?  
 coefficients[i].indexOf(Collections.*max*(coefficients[i])) :  
 coefficients[i].indexOf(Collections.*min*(coefficients[i])) );  
 //проверка потенциального соответвия строки условию диагонального преобладания  
 if(Math.*abs*(coefficients[i].get(maxAbsId)) \* 2 >=  
 coefficients[i].stream().reduce(0d, (a, b) -> Math.*abs*(a)+Math.*abs*(b))){  
 betterPlace.set(i, maxAbsId);  
 checkPlace.set(i, -1);  
 }  
 }  
 if(Collections.*max*(checkPlace) != -1){  
 System.*out*.println("Диагональное преобладание не может быть достигнуто");  
 return false;  
 }  
 List<Double> copy[] = new ArrayList[N];  
 for(int i = 0; i < N; i++){  
 // copy[i] = new ArrayList<Double>(matrix[i].size());  
 copy[i] = new ArrayList<>(matrix[i]);  
 System.*out*.println(matrix[i]);  
 }  
 for(int i = 0; i < N; i++){  
 matrix[betterPlace.get(i)] = copy[i];  
 freeMember.set(betterPlace.get(i), copy[i].remove(N));  
 coefficients[betterPlace.get(i)] = copy[i];  
 }  
 System.*out*.println("Условие диагонального преобладания выполнено. Финальный вид системы:");  
 for(int i = 0; i < N; i++){  
 for(Object u : coefficients[i]){  
 System.*out*.print(u + " ");  
 }  
 System.*out*.println(" " + freeMember.get(i));  
 }  
 return true;  
 }  
 public void form(){  
 System.*out*.println("Векторы коэффициентов и правых частей преобразованной системы:");  
 for(int i = 0; i < N; i++){  
 if(coefficients[i].get(i) == 0)  
 continue;  
 for(int j = 0; j < N; j++){  
 if(i == j){  
 System.*out*.print("0 ");  
 continue;  
 }  
 coefficients[i].set(j, -coefficients[i].get(j)/coefficients[i].get(i));  
 System.*out*.print(coefficients[i].get(j) + " ");  
 }  
 freeMember.set(i, freeMember.get(i)/coefficients[i].get(i));  
 System.*out*.println("\t" + freeMember.get(i));  
 coefficients[i].set(i, 0d);  
 }  
 prevX = new ArrayList<>(freeMember);  
 nextX = new ArrayList<>(freeMember);  
 }  
  
 private double eps;  
  
 public void setEps(double eps) {  
  
 this.eps = eps;   
  
 }  
  
 void iteration(){  
 double maxDelta = 0;  
 int count = 0;  
 while(true){  
 for(int i = 0; i < N; i++){  
 double xi = freeMember.get(i);  
 for(int j = 0; j < N; j++){  
 xi += prevX.get(j)\*coefficients[i].get(j);  
 }  
 nextX.set(i, xi);  
 maxDelta = Double.*max*(Math.*abs*(nextX.get(i)-prevX.get(i)), maxDelta);  
 }  
 for(int i = 0; i < N; i++){  
 prevX.set(i, nextX.get(i));  
 }  
 count++;  
 System.*out*.println("Максимальная погрешность на шаге " + count + " : " + maxDelta);  
 if(eps >= maxDelta){  
 System.*out*.println("Найдено решение за "+ count + " операций, удовлетворяющее требуемой погрешности:");  
 System.*out*.println("delta = " + maxDelta);  
 for(int i = 0; i < N; i++){  
 System.*out*.print("x" + (i+1));  
 System.*out*.printf(" = %.18f", prevX.get(i));  
 System.*out*.println();  
 }  
 break;  
 }  
 if(count > 10000){  
 System.*out*.println("Решение не может быть найдено с заданной погрешностью, попробуйте меньшую");  
 break;  
 }  
 maxDelta = 0;  
 }  
 }  
}

**Примеры:**



Изображение выглядит как стол

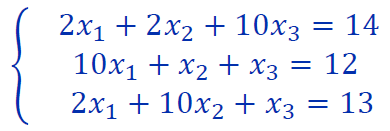
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



**input.txt**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Вывод: написал программу, которая решает СЛАУ итерационным методом с хорошей точностью, особенно, если выполнено условие диагонального преобладания.**