Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по «Вычислительная математика»

Выполнил:

Студент группы P3207

Разинкин А.В.

Преподаватели:

Санкт-Петербург

2024

# Цель работы

Разработать приложение для решения систем линейных алгебраических уравнений методом простых итераций.

# Задание

Должно быть реализовано:

* Ввод данных при помощи клавиатуры или файла – по усмотрению пользователя.
* Ввод размерности матрицы и точности вычислений.
* 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: ­x­1, x2, …, xn
* Вывод количества итераций, за которое было найдено решение.
* Вывод вектора погрешностей: |xi(k) – xi(k – 1)|

# Блок-схема реализованного алгоритма

A diagram of a flowchart

Description automatically generated

# Реализация (код) численного метода

// метод для приведения матрицы к матрице с диагональным преобладанием  
public void toConvergence(Data data) {  
 List<List<Double>> A = data.getA();  
 List<Double> B = data.getB();  
 int[] new\_rows\_positions = new int[A.size()];  
 Arrays.*fill*(new\_rows\_positions, -1);  
  
 for (int i = 0; i < A.size(); i++) {  
 List<Double> row = A.get(i);  
  
 double max = Math.*abs*(row.get(0));  
 int position = 0;  
 double sum = 0;  
  
 for (int j = 1; j < row.size(); j++) {  
 double current = Math.*abs*(row.get(j));  
 if (current > max) {  
 sum += max;  
 max = current;  
 position = j;  
 } else if (current == max) {  
 } else {  
 sum += current;  
 }  
 }  
  
 if (max < sum || new\_rows\_positions[position] != -1) {  
 System.*out*.print("""  
 Не удалось привести исходную матрицу к матрице с диагональным преобладанием.  
 Предупреждение: при продолжении решения конечный ответ может не сойтись.""");  
 InputReader inputReader = new InputReader();  
 data.setIterations(inputReader.readPositiveInt("Введите кол-во итераций: "));  
 return;  
 }  
  
 new\_rows\_positions[position] = i;  
 }  
  
 List<List<Double>> new\_A = new ArrayList<>();  
 List<Double> new\_B = new ArrayList<>();  
 for (int i : new\_rows\_positions) {  
 new\_A.add(A.get(i));  
 new\_B.add(B.get(i));  
 }  
  
 data.setA(new\_A);  
 data.setB(new\_B);  
}  
  
// метод поиска решения с заявленной точностью  
public void iterate(Data data) {  
 List<List<Double>> A = data.getA();  
 List<Double> B = data.getB();  
 int n = data.getA().size();  
 double[] previousApproximation = new double[n];  
 for (int i = 0; i < n; i++)  
 previousApproximation[i] = data.getB().get(i);  
  
 double[] newApproximation = new double[n];  
 int iterationCounter = 0;  
 while (true) {  
  
 for (int i = 0; i < n; i++) {  
 double newValue = B.get(i) / A.get(i).get(i);  
 for (int j = 0; j < n; j++) {  
 if (i != j) {  
 newValue -= (A.get(i).get(j) / A.get(i).get(i)) \* previousApproximation[j];  
 if (Double.*isNaN*(newValue) || Double.*isInfinite*(newValue)) {  
 System.*out*.println("Данная СЛАУ не обладает сходящимся решением.");  
 System.*exit*(1);  
 }  
 }  
 }  
 newApproximation[i] = newValue;  
 }  
 if (getMaxDeviation(previousApproximation, newApproximation) <= data.getAccuracy()  
 || (data.getIterations() != -1 && data.getIterations() == iterationCounter)) {  
 InputReader inputReader = new InputReader();  
 int c = inputReader.readPositiveInt("Введите кол-во символов после запятой: ");  
 System.*out*.println("Было проведено " + iterationCounter + " итераций.");  
 for (int i = 0; i < n; i++) {  
 System.*out*.println("x" + (i + 1) + "=" + String.*format*("%." + c + "f" , newApproximation[i]) + "; Отклонение составляет: " + String.*format*("%." + c + "f", Math.*abs*(newApproximation[i] - previousApproximation[i])));  
 }  
 break;  
 }  
 for (int i = 0; i < n; i++)  
 previousApproximation[i] = newApproximation[i];  
 iterationCounter++;  
 }  
}  
  
// метод поиска максимального отклоения следующего приближения  
private double getMaxDeviation(double[] previousApproximation, double[] newApproximation) {  
 double maxDeviation = 0;  
  
 for (int i = 0; i < previousApproximation.length; i++) {  
 double deviation = Math.*abs*(previousApproximation[i] - newApproximation[i]);  
 if (deviation > maxDeviation)  
 maxDeviation = deviation;  
 }  
  
 return maxDeviation;  
}

# Ссылка на GitHub с основной реализацией

<https://github.com/DecafMangoITMO/ITMO/tree/main/Computational%20Mathematics/MethodOfSimpleIterations>

# Пример работы программы

Для выхода из программы напишите exit.

Введите размерность матрицы: 3

Введите точность: 0.01

Введите коэффициенты построчно.

Например, если ваш имеет вид:

a11 a12 | b1

a21 a22 | b2

Ввод будет следующим:

a11 a12 b1

a21 a22 b2

2 2 10 14

10 1 1 12

2 10 1 13

Введите кол-во символов после запятой: 5

Было проведено 7 итераций.

x1=1.00041; Отклонение составляет: 0.00186

x2=1.00052; Отклонение составляет: 0.00235

x3=1.00066; Отклонение составляет: 0.00296

# Вывод

В ходе реализации данной лабораторной работы я ознакомился с работой алгоритма простых итераций, предназначенного для решения совместных определенных систем линейных алгебраических уравнений (СЛАУ). Данный алгоритм относится к виду итерационных: решение системы (если оно существует) достигается путем приближения за счет конечного числа итераций.