Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

          \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A picture containing logo

Description automatically generated

Лабораторная работа №1 по дисциплине  
«Вычислительная математика»

"Решение системы линейных алгебраических уравнений СЛАУ"

Выполнил: Дау Конг Туан Ань

Группа: P32151

Преподаватель: Машина Е.А

г. Санкт-Петербург

2023

**Цель работы:**

Научиться искать решение СЛАУ при помощи численных методов, написать программу, которая будет совершать приближенные вычисления и находить решение, получая на вход матрицу из файла или консоли.

**Задание лабораторной работы:**

**Вариант 9** => метода Гаусса-Зейделя

1. № варианта определяется как номер в списке группы согласно ИСУ. (**9**)
2. В программе численный метод должен быть реализован в виде отдельной подпрограммы или класса, в который входные/выходные данные передаются в качестве параметров.
3. Размерность матрицы n <=20 (задается из файла или с клавиатуры - по выбору конечного пользователя).
4. Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

**Для итерационных методов должно быть реализовано:**

* 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: x1,  x2,  …, xn
* Вывод количества итераций, за которое было найдено решение.
* Вывод вектора погрешностей: |xi(k)-xik-1|

**Описание метода, расчетные формулы:**

Метод Гаусса-Зейделя является итерационным методом и находит конечные     значение переменных последовательным приближением после каждой итерации. Так, имея в исходную матрицу A\*X = B строится матрица с вынесенными переменными в каждой строчке (так в первой строке x1 = f(x2, x3…) + b1/a11, для второй строчки x2 = f(x1, x3, …) + b2/a22 и т.д.)

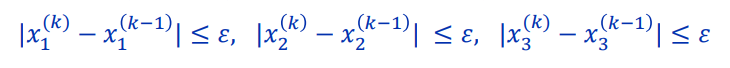
 Отсюда конечные формулы для итераций:

* Начальное приближение берется как свободные коэффициенты правой части, то есть  
  x10 = d1 = b1 / a11, x20 = d2 = b2 / a22,  
  …
* На каждой итерации вычисляются новые значения для всех переменных x1, x2 … методом подстановки в уравнение для соответствующей переменной вместо других переменных их последние значения (так для x11будут использоваться значения x20, x30, … xn0, когда как для вычисления x21 вместо x10 будет подставляться x11 найденное на прошлом шаге этой итерации и так далее – это и есть отличие данного метода от метода простых итераций).

Изображение выглядит как текст

Автоматически созданное описание

* Сам итерационный процесс продолжается до тех пор, пока значения разниц вычисленных переменных от их значений на прошлом шаге не будет меньше, чем заданное заранее значение погрешности ε



**Программа (часть, реализующая вычисления):**

public static double[][] seidelMethod(Matrix matrix, double epsilon) {  
 int dimension = matrix.getDimension();  
 Matrix copyMatrix = matrix.cloneMatrix();  
 copyMatrix.toTriangular(0);  
 copyMatrix.printMatrix();  
 if(copyMatrix.getDeterminate() == 0) {  
 return null;  
 } else {  
 System.*out*.println("Determinate of matrix: " + copyMatrix.getDeterminate());  
 }  
 double[][] C = new double[dimension][dimension];  
 double[] D = new double[dimension];  
 *rearrangeMatrix*(matrix);  
 double[][] matrix\_1 = matrix.getMatrix();  
 for(int i = 0 ;i < dimension; ++i) {  
 for(int j = 0 ; j < dimension; ++j) {  
 if(i != j) {  
 C[i][j] = -matrix\_1[i][j] /matrix\_1[i][i];  
 }  
 }  
 D[i] = matrix\_1[i][dimension] /matrix\_1[i][i];  
 C[i][i] = 0;  
 }  
 System.*out*.println("C matrix: ");  
 for(int i = 0 ;i < dimension; ++i) {  
 for(int j = 0 ; j < dimension; ++j) {  
 System.*out*.print(C[i][j] + " ");  
 }  
  
 System.*out*.println();  
 }  
  
 System.*out*.println("D vector: ");  
 for(int i = 0 ; i < dimension; ++i) {  
 System.*out*.print(D[i] + " ");  
 }  
 System.*out*.println();  
 double[] newX = new double[dimension];  
 double[] oldX = new double[dimension];  
 for(int i = 0 ;i < dimension; ++i) {  
 oldX[i] = 0;  
 newX[i] = D[i];  
 }  
 double[] temp = new double[dimension];  
 while(!*checkIfLessThanEpsilon*(dimension, newX, oldX, epsilon)) {  
 for(int i = 0; i < dimension; ++i) {  
 temp[i] = newX[i];  
 newX[i] = 0;  
 }  
  
 for(int i = 0 ;i < dimension; ++i) {  
 for(int j = 0 ;j < i; ++j) {  
 newX[i] += C[i][j]\* newX[j];  
 }  
  
 for(int j = i + 1; j < dimension; ++j) {  
 newX[i] += C[i][j] \* oldX[j];  
 }  
  
 newX[i] += D[i];  
 }  
 for(int i = 0 ;i < dimension; ++i) {  
 oldX[i] = temp[i];  
 }  
 }  
  
 double[][] result = new double[2][dimension];  
 for(int i = 0 ; i < dimension; ++i) {  
 result[0][i] = newX[i];  
 result[1][i] = D[i];  
 }  
  
 return result;  
 }

\*Epsilon: default equals to 0.1

Link github: [Click here](https://github.com/andrey551/Computation-Mathematics/tree/main/Lab1)

**Примеры и результат работы программы:**

Text

Description automatically generated

Text

Description automatically generated

**Выводы по работе:**

При помощи вычислительных устройств можно вычислять приближённые решения различных математических задач различными способами, которые сам человек вычислял бы достаточно долго. Реализован метод Гаусса-Зейделя для решения СЛАУ.