Государственное бюджетное образовательное учреждение высшего

профессионального образования

«Поволжский государственный университет телекоммуникаций

и информатики»

Кафедра ПрИ

Лабораторная работа №2 на тему

**«Решение систем линейных алгебраических уравнений. Точные методы»**

по дисциплине «Численные методы»

Вариант №6

**Выполнил**: студент группы ПРИ-22

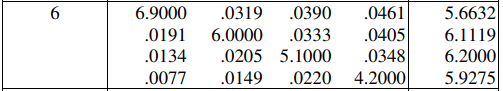
Дубинкин Данила Юрьевич

Самара 2024

**1. Цель работы:**

научится решить систему уравнений, методом Якоби и Зейделя. Разобраться в их алгоритме.

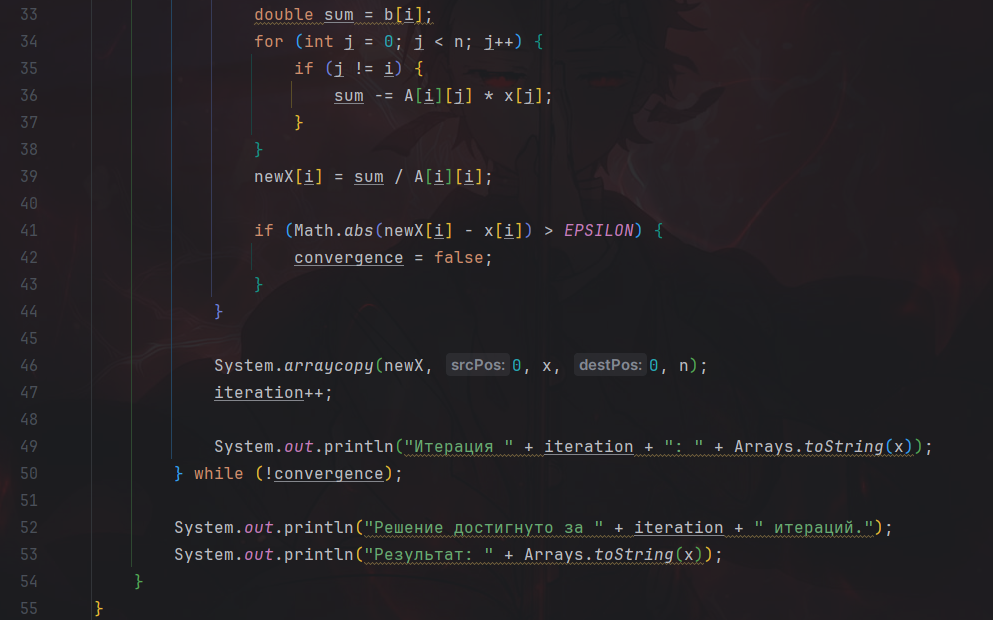
**2. Исходные Данные:**

****

**3. Листинг программы:**

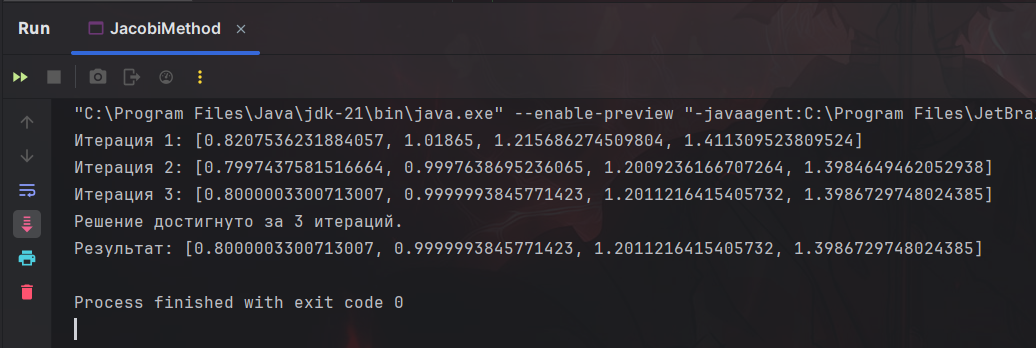
**3.1.1 Метод Якоби:**





**package src;  
import java.util.Arrays;  
public class JacobiMethod {  
 private static final double EPSILON = 0.01;  
 public static void main(String[] args) {  
 double[][] A = {  
 {6.9000, 0.0319, 0.0390, 0.0461},  
 {0.0191, 6.0000, 0.0333, 0.0405},  
 {0.0134, 0.0149, 5.1000, 0.0348},  
 {0.0077, 0.0205, 0.0220, 4.2000}  
 };  
  
 double[] b = {5.6632, 6.1119, 6.2000, 5.9275};  
  
 double[] x = new double[b.length];  
 Arrays.fill(x, 0);  
  
 jacobiMethod(A, b, x);  
 }  
 public static void jacobiMethod(double[][] A, double[] b, double[] x) {  
 int n = b.length;  
 double[] newX = new double[n];  
 boolean convergence;  
 int iteration = 0;  
 do {  
 convergence = true;  
 for (int i = 0; i < n; i++) {  
 double sum = b[i];  
 for (int j = 0; j < n; j++) {  
 if (j != i) {  
 sum -= A[i][j] \* x[j];  
 }  
 }  
 newX[i] = sum / A[i][i];  
 if (Math.abs(newX[i] - x[i]) > EPSILON) {  
 convergence = false;  
 }  
 }  
 System.arraycopy(newX, 0, x, 0, n);  
 iteration++;  
 System.out.println("Итерация " + iteration + ": " + Arrays.toString(x));  
 } while (!convergence);  
 System.out.println("Решение достигнуто за " + iteration + " итераций.");  
 System.out.println("Результат: " + Arrays.toString(x));  
 }  
}**

**3.1.2 Результат:**

****

**Итерация 1:**

[0.8207536231884057,

1.01865,

1.215686274509804,

1.411309523809524]

**Итерация 2:**

[0.7997437581516664,

0.9997638695236065,

1.2009236166707264,

1.3984649462052938]

**Итерация 3:**

[0.8000003300713007,

0.9999993845771423,

1.2011216415405732,

1.3986729748024385]

**Решение достигнуто за 3 итераций.**

**Результат:**

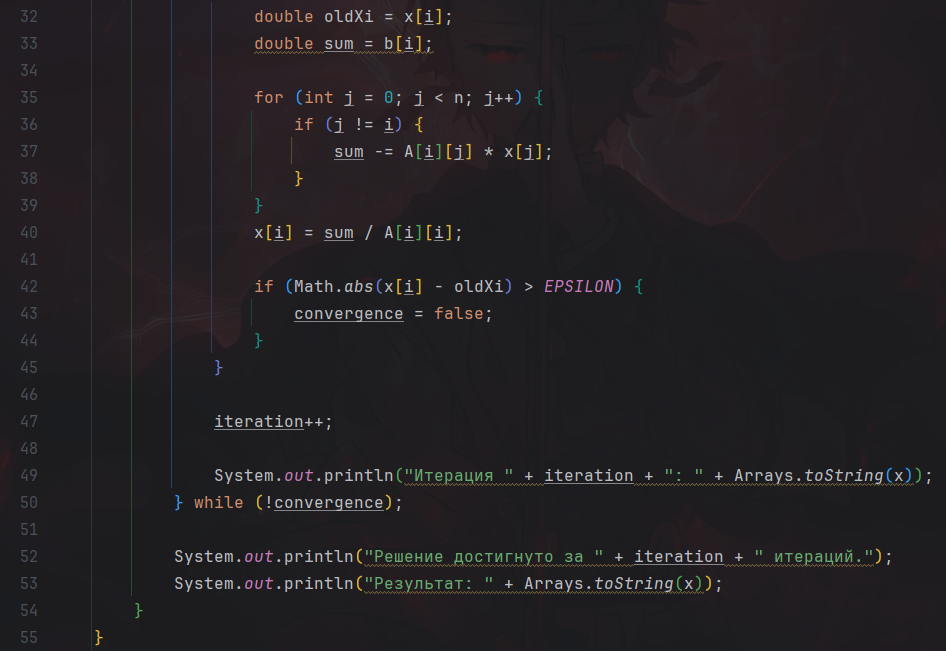
[0.8000003300713007,

0.9999993845771423,

1.2011216415405732,

1.3986729748024385]

**3.2.1 Метод Зейделя:** 



**package src;**

**import java.util.Arrays;**

**public class GaussZeidelMethod {**

**private static final double EPSILON = 0.01;**

**public static void main(String[] args) {**

**double[][] A = {**

**{6.9000, 0.0319, 0.0390, 0.0461},**

**{0.0191, 6.0000, 0.0333, 0.0405},**

**{0.0134, 0.0149, 5.1000, 0.0348},**

**{0.0077, 0.0205, 0.0220, 4.2000}**

**};**

**double[] b = {5.6632, 6.1119, 6.2000, 5.9275};**

**double[] x = new double[b.length];**

**Arrays.fill(x, 0);**

**gaussSeidelMethod(A, b, x);**

**}**

**public static void gaussSeidelMethod(double[][] A, double[] b, double[] x) {**

**int n = b.length;**

**boolean convergence;**

**int iteration = 0;**

**do {**

**convergence = true;**

**for (int i = 0; i < n; i++) {**

**double oldXi = x[i];**

**double sum = b[i];**

**for (int j = 0; j < n; j++) {**

**if (j != i) {**

**sum -= A[i][j] \* x[j];**

**}**

**}**

**x[i] = sum / A[i][i];**

**if (Math.abs(x[i] - oldXi) > EPSILON) {**

**convergence = false;**

**}**

**}**

**iteration++;**

**System.out.println("Итерация " + iteration + ": " + Arrays.toString(x));**

**} while (!convergence);**

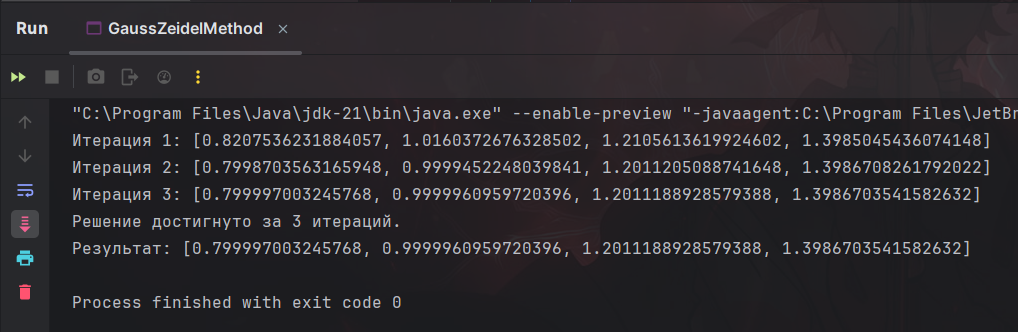
**System.out.println("Решение достигнуто за " + iteration + " итераций.");**

**System.out.println("Результат: " + Arrays.toString(x));**

**}**

**}**

**3.2.2 Результат:**

****

**Итерация 1:**

[0.8207536231884057,

1.0160372676328502,

1.2105613619924602,

1.3985045436074148]

**Итерация 2:**

[0.7998703563165948,

0.9999452248039841,

1.2011205088741648,

1.3986708261792022]

**Итерация 3:**

[0.799997003245768,

0.9999960959720396,

1.2011188928579388,

1.3986703541582632]

**Решение достигнуто за 3 итераций.**

**Результат:**

[0.799997003245768,

0.9999960959720396,

1.2011188928579388,

1.3986703541582632]