**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №3

**Итерационные методы решения СЛАУ**

**Выполнил:**

Кендысь Алексей Максимович

студент 2 курса, 9 группы,

специальность

“прикладная математика”

**Преподавательница:**

Ассистентка кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2021

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2

Краткие теоретические сведения ------------------------------------------------ 2-3

Листинг программы ---------------------------------------------------------------- 3-7

Результаты --------------------------------------------------------------------------- 7-8

Выводы ------------------------------------------------------------------------------- 8

**Постановка задачи**

Написать и отладить программу численного решения систем линейных алгебраических уравнений с квадратной матрицей порядка n

1) методом градиентного спуска;

2) методом релаксации.

Предусмотреть сообщение о выходе из итерационного процесса из-за превышения допустимого максимального количества итераций (). В качестве критерия остановки итерационного процесса использовать

.

**Краткие теоретические сведения**

Рассмотрим систему линейных уравнений .

Метод градиентного (наискорейшего) спуска:

1)

2)

3)

k = 0, 1, …

Теорема (достаточное условие сходимости). Пусть и числа – границы спектра матрицы , т.е. метод градиентного спуска сходится при любом начальном приближении , причём со скоростью геометрической прогрессии со знаменателем .

Метод релаксации:

,

параметр релаксации

Теорема (достаточное условие сходимости). Если и , то метод релаксации является сходящимся при любом начальном приближении .

**Листинг программы**

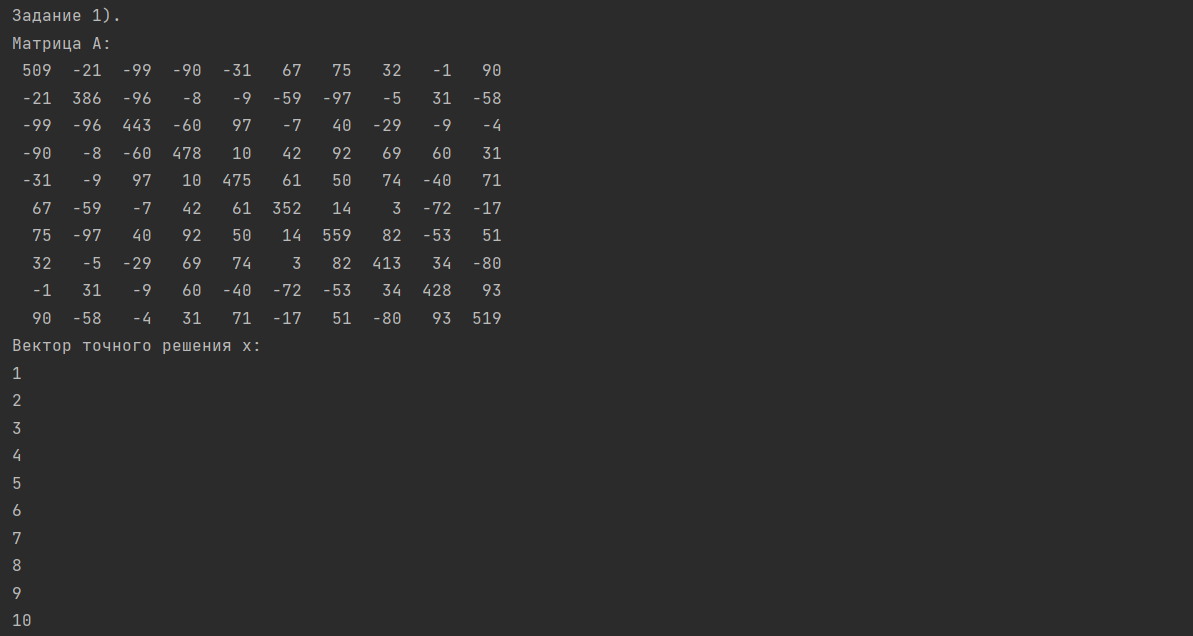
Файл Matrix.java:

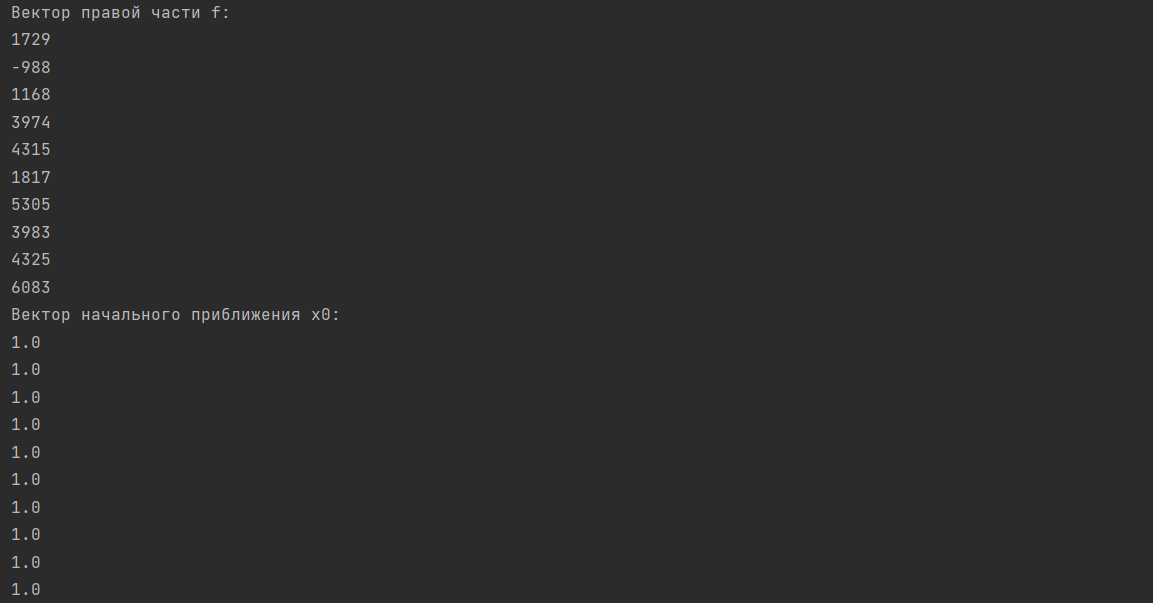
import java.util.\*;  
public class Matrix {  
 private final int[][] a;  
 private final int[] x;  
 private final int[] f;  
 private final double[] x0;  
 private double[] xK;  
 private final double e = 0.0000001;  
 private final int kMax = 5000;  
 private final static int *V* = 4;  
 Matrix(int n) {  
 //Заполнение матрицы A  
 Random r = new Random();  
 a = new int[n][n];  
 //Нижняя треугольная часть  
 for (int i = 1; i < n; i++) {  
 for (int j = 0; j < i; j++) {  
 a[i][j] = r.nextInt(201) - 100;  
 }  
 }  
 //Верхняя треугольная часть  
 for (int i = 0; i < n; i++) {  
 for (int j = i + 1; j < n; j++) {  
 a[i][j] = a[j][i];  
 }  
 }  
 //Диагональ  
 int sum = 0;  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < i; j++) {  
 sum += Math.*abs*(a[i][j]);  
 }  
 for (int j = i + 1; j < n; j++) {  
 sum += Math.*abs*(a[i][j]);  
 }  
 a[i][i] = r.nextInt(9 \* *V*) + sum;  
 sum = 0;  
 }  
  
  
 //Заполнение векторов x, f и x0  
 x = new int[n];  
 for (int i = 0; i < n; i++) {  
 x[i] = i + 1;  
 }  
 f = new int[n];  
 for (int i = 0; i < n; i++) {  
 f[i] = 0;  
 for (int j = 0; j < n; j++) {  
 f[i] += a[i][j] \* x[j];  
 }  
 }  
 x0 = new double[n];  
 for (int i = 0; i < n; i++) {  
 x0[i] = 1;  
 }  
 }  
 public int gradDescent() {  
 //Метод градиентного спуска  
 int q = 0;  
 double[] r = new double[a.length];  
 double t;  
 double tNum;  
 double tDen;  
 double aR;  
 double norm;  
  
 xK = x0.clone();  
  
 while (q <= kMax) {  
 //Невязка  
 for (int i = 0; i < a.length; i++) {  
 r[i] = 0;  
 for (int j = 0; j < a[i].length; j++) {  
 r[i] += a[i][j] \* xK[j];  
 }  
 r[i] -= f[i];  
 }  
  
 //Параметр  
 tNum = 0;  
 tDen = 0;  
 for (int i = 0; i < a.length; i++) {  
 aR = 0;  
 for (int j = 0; j < a[i].length; j++) {  
 aR += a[i][j] \* r[j];  
 }  
 tNum += r[i] \* r[i];  
 tDen += aR \* r[i];  
 }  
 t = tNum/tDen;  
  
 //Подсчёт xK  
 for (int i = 0; i < xK.length; i++) {  
 xK[i] -= t \* r[i];  
 }  
  
 q++;  
  
 //Проверка на окончание процесса  
 norm = 0;  
 for (double i : r) {  
 norm += Math.*pow*(i, 2);  
 }  
 norm = Math.*sqrt*(norm);  
 if (norm < e) {  
 return q;  
 }  
 }  
 return 0;  
 }  
 public int sor(double w) {  
 //Метод релаксации  
 int q = 0;  
 double p;  
 xK = x0.clone();  
  
 while(q <= kMax) {  
 for (int i = 0; i < xK.length; i++) {  
 xK[i] = (1 - w) \* xK[i];  
 p = f[i];  
 for (int j = 0; j < i; j++) {  
 p -= a[i][j] \* xK[j];  
 }  
 for (int j = i + 1; j < xK.length; j++) {  
 p -= a[i][j] \* xK[j];  
 }  
 xK[i] += (w/a[i][i]) \* p;  
 }  
  
 q++;  
  
 //Проверка на окончание процесса  
 if (getResidual() < e) {  
 return q;  
 }  
 }  
 return q;  
 }  
 public void outA() {  
 //Вывод матрицы A  
 Formatter fmt = new Formatter();  
 for (int[] row : a) {  
 for (int i : row) {  
 fmt.format("% 4d ", i);  
 }  
 fmt.format("\n");  
 }  
 System.*out*.print(fmt);  
 }  
 public void outX() {  
 //Вывод вектора x  
 for (int i : x) {  
 System.*out*.println(i);  
 }  
 }  
 public void outF() {  
 //Вывод вектора f  
 for (int i : f) {  
 System.*out*.println(i);  
 }  
 }  
 public void outX0() {  
 //Вывод вектора x0  
 for (double i : x0) {  
 System.*out*.println(i);  
 }  
 }  
 public void outXK() {  
 //Вывод вектора xK  
 Formatter fmt = new Formatter();  
 for (double i : xK) {  
 fmt.format("%.12f\n", i);  
 }  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
 public double getResidual() {  
 //Норма невязки  
 double norm = 0;  
 double r;  
 for (int i = 0; i < a.length; i++) {  
 r = 0;  
 for (int j = 0; j < a[i].length; j++) {  
 r += a[i][j] \* xK[j];  
 }  
 r -= f[i];  
 norm += Math.*pow*(r, 2);  
 }  
 norm = Math.*sqrt*(norm);  
 return norm;  
 }  
 public double getError() {  
 //Абсолютная погрешность  
 double norm = 0;  
 for (int i = 0; i < xK.length; i++) {  
 norm += Math.*pow*(x[i] - xK[i], 2);  
 }  
 norm = Math.*sqrt*(norm);  
 return norm;  
 }  
}

Файл Main.java:

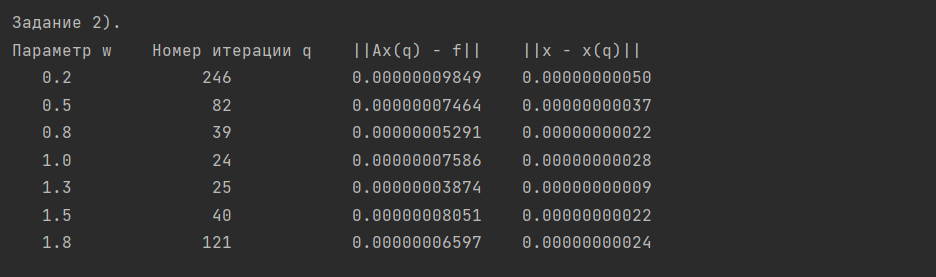
import java.util.\*;  
public class Main {  
 private static final int *N* = 10;  
 public static void main(String[] args) {  
 Formatter fmt;  
 System.*out*.println("Задание 1).");  
 Matrix myMatrix = new Matrix(*N*);  
 System.*out*.println("Матрица A:");  
 myMatrix.outA();  
 System.*out*.println("Вектор точного решения x:");  
 myMatrix.outX();  
 System.*out*.println("Вектор правой части f:");  
 myMatrix.outF();  
 System.*out*.println("Вектор начального приближения x0:");  
 myMatrix.outX0();  
 int q;  
 q = myMatrix.gradDescent();  
 System.*out*.println("Номер итерации q:");  
 if (q != 0) {  
 System.*out*.println(q);  
 } else {  
 System.*out*.println("Достигнуто максимальное количество итераций.");  
 }  
 System.*out*.println("Полученное приближённое решение xK:");  
 myMatrix.outXK();  
 fmt = new Formatter();  
 fmt.format("Норма невязки:\n%.12f\n", myMatrix.getResidual());  
 fmt.format("Абсолютная погрешность:\n%.12f", myMatrix.getError());  
 System.*out*.println(fmt);  
 fmt.close();  
 System.*out*.println("\nЗадание 2).");  
 double[] w = new double[] {0.2, 0.5, 0.8, 1.0, 1.3, 1.5, 1.8};  
 fmt = new Formatter();  
 fmt.format("Параметр w Номер итерации q ||Ax(q) - f|| ||x - x(q)||\n");  
 for (double v : w) {  
 q = myMatrix.sor(v);  
 fmt.format("%6.1f %15d %24.11f %16.11f\n", v, q, myMatrix.getResidual(), myMatrix.getError());  
 }  
 System.*out*.print(fmt);  
 }  
}

**Результаты**

****

****

****

****

**Выводы**

1) Скорость сходимости метода градиентного спуска такая же, что и у метода простой итерации с , т.е. с оптимальным параметром, что является преимуществом метода.

2) Метод Гаусса-Зейделя является частным случаем метода релаксации при .

3) Из последней таблицы видно, что чем ближе параметр релаксации к 1, тем быстрее сходится метод (для нашей матрицы). Также верхняя релаксация сходится быстрее нижней.