**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

**Кафедра вычислительной математики**

Лабораторная работа № 3

«Нахождение с помощью степенного метода минимального и максимального по модулю собственного значения»

Выполнила: Шелег Владислава Михайловна

2 курс, 9 группа

Преподаватель: Никифоров И. В

МИНСК

2016

**Постановка задачи.**

Найти максимальное и минимальное по модулю собственное значение данной матрицы с помощью степенного метода.

**Алгоритм решения**

Пусть матрица A имеет n линейно независимых собственных векторов , , и собственные значения матрицы A таковы, что .

* + - 1. Выбираем произвольное начальное (нулевое) приближение собственного вектора . Положить *k=*0.
      2. Найти , где *i* – любой номер , и положить *k=*1.
      3. Вычислить .
      4. Найти , где и – соответствующие координаты векторов и . При этом может быть любая координата с номером *i*, .
      5. Если , процесс завершить и положить . Если , положить *k=k*+1 и перейти к пункту 3.

Для того, чтобы избежать неограниченного роста координаты вектора , на каждой итерации следует применить нормировку:

Тогда с ростом итерации, а (единичному собственному вектору).

Наименьшее по модулю собственное значение матрицы A является максимальным собственным значением матрицы . Поэтому для нахождения минимального собственного значения матрицы А можно также применить степенной метод.

Это и называется методом обратной итерации. Обратные итерации сходятся к собственному вектору, соответствующему минимальному по модулю собственному значению матрицы А.

**Листинг программы**

(Программа реализована на языке Java)

public class Test {  
 public static final int N = 5;  
 public static double[] startingVector = new double[N];  
  
 public static void main(String[] args) throws IOException {  
 double eigenValueMax, eigenValueMin;  
 double[][] matrix = {{7, 2.5, 2, 1.5, 1},  
 {2.5, 8, 2.5, 2, 1.5},  
 {2, 2.5, 9, 2.5, 2},  
 {1.5, 2, 2.5, 10, 2.5},  
 {1, 1.5, 2, 2.5, 11}};  
  
 int lambda = 5;  
 for (int k = 0; k < N; ++k) {  
 matrix[k][k] += lambda;  
 }  
  
 eigenValueMax = powerMethod(matrix);  
 System.out.println("Eigen value max: " + eigenValueMax);  
  
 eigenValueMin = 1 / powerMethod(invert(matrix));  
 System.out.print("Eigen value min: " + eigenValueMin);  
 }

public static double[][] invert(double a[][]) {  
 int n = a.length;  
 double x[][] = new double[n][n];  
 double b[][] = new double[n][n];  
 int index[] = new int[n];  
 for (int i = 0; i < n; ++i)  
 b[i][i] = 1;  
  
 gaussian(a, index);  
  
 for (int i = 0; i < n - 1; ++i)  
 for (int j = i + 1; j < n; ++j)  
 for (int k = 0; k < n; ++k)  
 b[index[j]][k]  
 -= a[index[j]][i] \* b[index[i]][k];  
  
 for (int i = 0; i < n; ++i) {  
 x[n - 1][i] = b[index[n - 1]][i] / a[index[n - 1]][n - 1];  
 for (int j = n - 2; j >= 0; --j) {  
 x[j][i] = b[index[j]][i];  
 for (int k = j + 1; k < n; ++k) {  
 x[j][i] -= a[index[j]][k] \* x[k][i];  
 }  
 x[j][i] /= a[index[j]][j];  
 }  
 }  
 return x;  
 }  
  
 public static void gaussian(double a[][], int index[]) {  
 int n = index.length;  
 double c[] = new double[n];  
  
 for (int i = 0; i < n; ++i)  
 index[i] = i;  
  
 for (int i = 0; i < n; ++i) {  
 double c1 = 0;  
 for (int j = 0; j < n; ++j) {  
 double c0 = Math.abs(a[i][j]);  
 if (c0 > c1) c1 = c0;  
 }  
 c[i] = c1;  
 }  
  
 int k = 0;  
 for (int j = 0; j < n - 1; ++j) {  
 double pi1 = 0;  
 for (int i = j; i < n; ++i) {  
 double pi0 = Math.abs(a[index[i]][j]);  
 pi0 /= c[index[i]];  
 if (pi0 > pi1) {  
 pi1 = pi0;  
 k = i;  
 }  
 }  
  
 int itmp = index[j];  
 index[j] = index[k];  
 index[k] = itmp;  
 for (int i = j + 1; i < n; ++i) {  
 double pj = a[index[i]][j] / a[index[j]][j];  
 a[index[i]][j] = pj;  
 for (int l = j + 1; l < n; ++l)  
 a[index[i]][l] -= pj \* a[index[j]][l];  
 }  
 }  
 }  
  
 private static double powerMethod(double[][] matrix) throws IOException {  
 int n = matrix.length;  
 double eigenValue = 0, temp = 0;  
  
 Scanner sc = new Scanner(System.in);  
 System.out.println("Enter your starting vector ");  
 for (int i = 0; i < N; i++) {  
 startingVector[i] = sc.nextDouble();  
 }  
  
 double[] tempArray = new double[n];  
  
 do {  
 for (int i = 0; i < n; i++) {  
 tempArray[i] = 0;  
 for (int j = 0; j < n; j++)  
 tempArray[i] += matrix[i][j] \* startingVector[j];  
 }  
  
 for (int i = 0; i < n; i++)  
 startingVector[i] = tempArray[i];  
  
 temp = eigenValue;  
 eigenValue = 0;  
  
 for (int i = 0; i < n; i++) {  
 if (abs(startingVector[i]) > abs(eigenValue))  
 eigenValue = startingVector[i];  
 }  
 for (int i = 0; i < n; i++)  
 startingVector[i] /= eigenValue;  
  
 } while (abs(eigenValue - temp) > 0.00001);  
  
 return eigenValue;  
 }  
}

**Результат**

