**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №4

**Итерационные методы решения проблемы собственных значений**

**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 ------------------------------------------------ 2-4

Листинг программы ---------------------------------------------------------------- 4-10

Результаты --------------------------------------------------------------------------- 10-11

Выводы ------------------------------------------------------------------------------- 11-12

**Постановка задачи**

Задание 1:

Написать и отладить программу нахождения степенным методом (методом скалярных произведений) наибольшего по модулю собственного значения и соответствующего ему собственного вектора вещественной диагонализируемой матрицы A порядка n. Вычислительный процесс проводить с нормировкой векторов итерационной последовательности (использовать евклидову норму). В качестве критерия остановки итерационного процесса использовать . Предусмотреть сообщение о выходе из итерационного процесса, если он расходится.

Задание 2:

Написать и отладить программу нахождения итерационным методом вращений (Якоби) всех собственных значений и соответствующих им собственных векторов вещественной симметрической матрицы A порядка n. В качестве критерия остановки итерационного процесса использовать .

**Краткие теоретические сведения**

Рассмотрим действительную матрицу Она является диагонализируемой, и все собственные значения матрицы действительны.

Метод скалярных произведений:

Рассмотрим случай , .

ненулевой вектор

k = 0, 1, …

При этом, используя в алгоритме евклидову норму , получим, что .

Также, если при выборе алгоритм не сошёлся, то можно попробовать выбрать другой начальный вектор. Если же алгоритм снова не сошёлся, то можно сделать вывод, что искомое собственное значение является комплексным или не простым.

Итерационный метод вращений (Якоби):

Если

где:

,

где:

,

Тогда в пределе по k получим диагональную матрицу, диагональные элементы которой являются её собственными значениями.

Для получения собственных векторов нужно перемножить матрицы в том порядке, в котором они строились.

**Листинг программы**

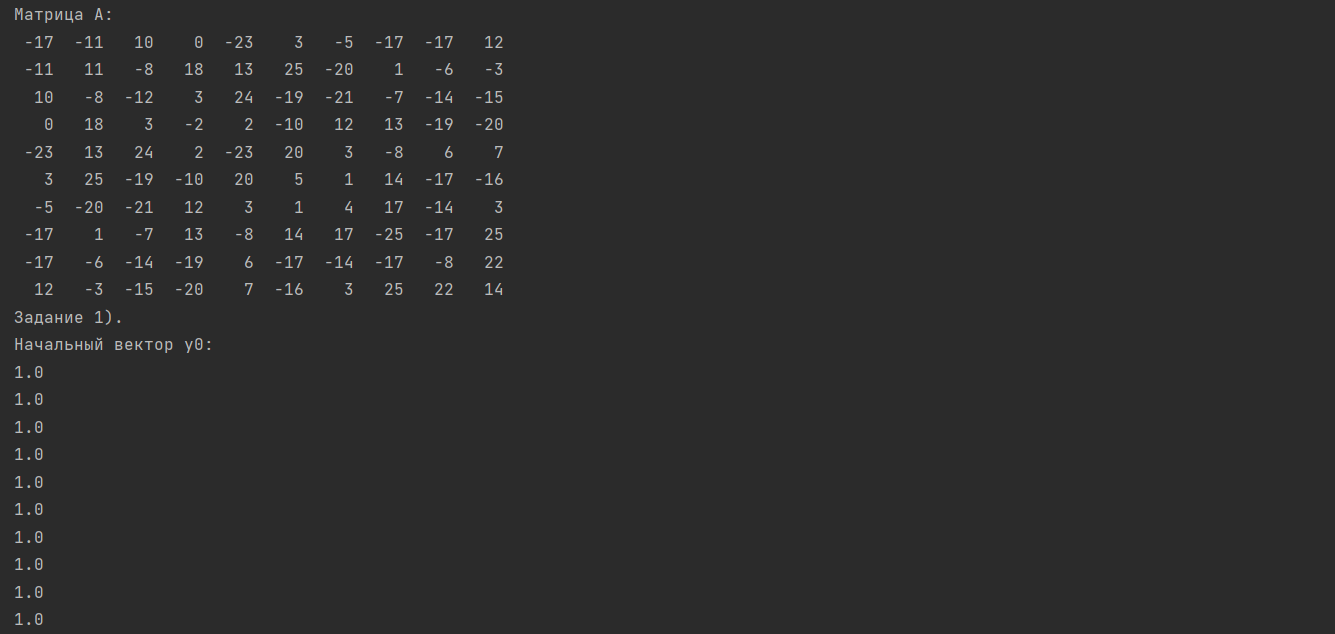
Файл Matrix.java:

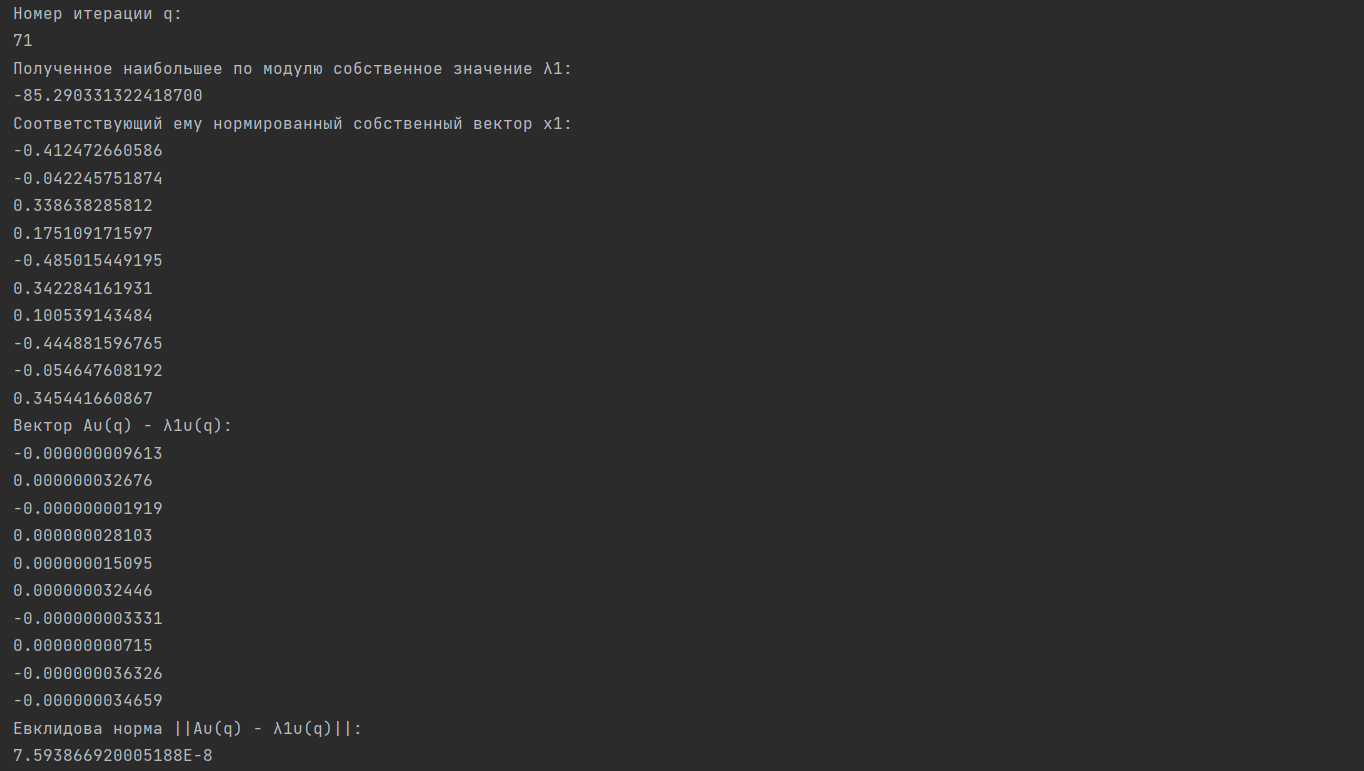
import java.util.\*;  
public class Matrix {  
 private final double[][] a;  
 private final double[] y0;  
 private double lambda1;  
 private double[] x1;  
 private double[] aul;  
 private final double[] lambda;  
 private final double[][] t;  
 private static final double *e* = 0.0000001;  
 private static final int *kMax* = 5000;  
 private static final int *kConv* = 50;  
 private boolean second = false;  
 public Matrix(int n) {  
 //Заполнение матрицы A  
 Random r = new Random();  
 a = new double[n][];  
 for (int i = 0; i < n; i++) {  
 a[i] = new double[i + 1];  
 for (int j = 0; j <= i; j++) {  
 a[i][j] = r.nextInt(51) - 25;  
 }  
 }  
  
 //Заполнение вектора y  
 y0 = new double[n];  
 for (int i = 0; i < n; i++) {  
 y0[i] = 1.;  
 }  
  
 //Вектор Au - λ1u  
 aul = new double[n];  
  
 //Матрица T и вектор из собственных значений  
 t = new double[n][n];  
 lambda = new double[n];  
 }  
 //Метод скалярных произведений  
 public int powerIt() {  
 double[] y = Arrays.*copyOf*(y0, y0.length);  
 double[] u = new double [y.length];  
 double[] newAul = new double[aul.length];  
 double norm;  
 norm = euNorm(y);  
 for (int i = 0; i < u.length; i++) {  
 u[i] = y[i] / norm;  
 }  
   
 int q = 0;  
 while (q <= *kMax*) {  
 //Подсчёт вектора y(k+1)  
 for (int i = 0; i < a.length; i++) {  
 y[i] = 0.;  
 for (int j = 0; j < a.length; j++) {  
 if (j <= i) {  
 y[i] += a[i][j] \* u[j];  
 } else {  
 y[i] += a[j][i] \* u[j];  
 }  
 }  
 }  
  
 //Подсчёт λ1  
 lambda1 = 0.;  
 for (int i = 0; i < y.length; i++) {  
 lambda1 += y[i] \* u[i];  
 }  
  
 //Подсчёт вектора u(k+1)  
 norm = euNorm(y);  
 for (int i = 0; i < u.length; i++) {  
 u[i] = y[i] / norm;  
 }  
  
 x1 = u;  
  
 //Подсчёт вектора Au(k+1) - λ1u(k+1)  
 for (int i = 0; i < newAul.length; i++) {  
 newAul[i] = 0.;  
 for (int j = 0; j < newAul.length; j++) {  
 if (j <= i) {  
 newAul[i] += a[i][j] \* u[j];  
 } else {  
 newAul[i] += a[j][i] \* u[j];  
 }  
 }  
 newAul[i] -= lambda1 \* u[i];  
 }  
  
 //Проверка на окончание процесса  
 norm = euNorm(newAul);  
 if (norm < *e*) {  
 return q + 1;  
 }  
  
 //Проверка на сходимость  
 if (q == *kConv* && norm > euNorm(aul)) {  
 if (second) {  
 return 0;  
 }  
 else {  
 for (int i = 0; i < a.length; i++) {  
 y0[i] = i + 1;  
 }  
 second = true;  
 powerIt();  
 }  
 }  
 aul = newAul;  
  
 q++;  
 }  
 return 0;  
 }  
 //Итерационный метод вращений (Якоби)  
 public int rotateIt() {  
 double max;  
 int iK;  
 int jK;  
 double m;  
 double c;  
 double s;  
 double d;  
 double sum;  
  
 //Создаём копию матрицы A  
 double[][] aC = new double[a.length][a.length];  
 for (int i = 0; i < a.length; i++) {  
 System.*arraycopy*(a[i], 0, aC[i], 0, i + 1);  
 for (int j = i + 1; j < a.length; j++) {  
 aC[i][j] = a[j][i];  
 }  
 }  
   
 int q = 0;  
 while (q <= *kMax*) {  
 //Находим максимальный по модулю недиагональный элемент  
 max = 0;  
 iK = 0;  
 jK = 0;  
 for (int i = 0; i < aC.length; i++) {  
 for (int j = i + 1; j < aC.length; j++) {  
 if (Math.*abs*(aC[i][j]) > max) {  
 max = Math.*abs*(aC[i][j]);  
 iK = i;  
 jK = j;  
 }  
 }  
 }  
  
 //Подсчёт косинуса и синуса  
 if (aC[iK][iK] == aC[jK][jK]) {  
 c = 1./Math.*sqrt*(2);  
 s = - c;  
 } else {  
 m = 2. \* aC[iK][jK] / (aC[iK][iK] - aC[jK][jK]);  
 d = 1./Math.*sqrt*(1. + Math.*pow*(m, 2.));  
 c = Math.*sqrt*((1. + d) / 2.);  
 s = Math.*signum*(m) \* Math.*sqrt*((1. - d) / 2.);  
 }  
  
 //Подсчёт новых значений матрицы A  
 for (int i = 0; i < aC.length; i++) {  
 double a1 = aC[i][iK];  
 double a2 = aC[i][jK];  
 aC[i][iK] = a1 \* c + a2 \* s;  
 aC[i][jK] = - a1 \* s + a2 \* c;  
 }  
 for (int i = 0; i < aC.length; i++) {  
 double a1 = aC[iK][i];  
 double a2 = aC[jK][i];  
 if (i <= iK) {  
 aC[iK][i] = a1 \* c + a2 \* s;  
 }  
 if (i <= jK) {  
 aC[jK][i] = - a1 \* s + a2 \* c;  
 }  
 }  
 for (int i = iK + 1; i <= jK; i++) {  
 aC[iK][i] = aC[i][iK];  
 }  
 for (int i = jK + 1; i < aC.length; i++) {  
 aC[iK][i] = aC[i][iK];  
 aC[jK][i] = aC[i][jK];  
 }  
  
 //Подсчёт значений матрицы T  
 if (q == 0) {  
 for (int i = 0; i < t.length; i++) {  
 t[i][i] = 1.;  
 }  
 t[iK][iK] = c;  
 t[iK][jK] = - s;  
 t[jK][iK] = s;  
 t[jK][jK] = c;  
 } else {  
 for (int i = 0; i < t.length; i++) {  
 double t1 = t[i][iK];  
 double t2 = t[i][jK];  
 t[i][iK] = t1 \* c + t2 \* s;  
 t[i][jK] = - t1 \* s + t2 \* c;  
 }  
 }  
  
 //Проверка на окончание процесса  
 sum = 0.;  
 for (int i = 0; i < aC.length; i++) {  
 for (int j = 0; j < i; j++) {  
 sum += 2. \* Math.*pow*(aC[i][j], 2.);  
 }  
 }  
 if (sum < *e*) {  
 for (int i = 0; i < lambda.length; i++) {  
 lambda[i] = aC[i][i];  
 }  
 return q + 1;  
 }  
  
 q++;  
 }  
 return 0;  
 }  
 //Вывод матрицы A  
 public void showA() {  
 Formatter fmt = new Formatter();  
 for (int i = 0; i < a.length; i++) {  
 for (int j = 0; j < a.length; j++) {  
 if (j <= i) {  
 fmt.format("% 4.0f ", a[i][j]);  
 } else {  
 fmt.format("% 4.0f ", a[j][i]);  
 }  
 }  
 fmt.format("\n");  
 }  
 System.*out*.print(fmt);  
 }  
 //Вывод начального вектора y0  
 public void showY0() {  
 for (double i : y0) {  
 System.*out*.println(i);  
 }  
 }  
 //Вывод полученного степенным методом значения λ1  
 public void showLambda1() {  
 Formatter fmt = new Formatter();  
 fmt.format("%.15f\n", lambda1);  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
 //Вывод собственного вектора x1, соответствующего λ1   
 public void showX1() {  
 Formatter fmt = new Formatter();  
 for (double i : x1) {  
 fmt.format("%.12f\n", i);  
 }  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
 //Вывод вектора Au(q) - λ1u(q)  
 public void showAul() {  
 Formatter fmt = new Formatter();  
 for (double i : aul) {  
 fmt.format("%.12f\n", i);  
 }  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
 //Вывод нормы вектора Au(q) - λ1u(q)  
 public void showEuAul() {  
 System.*out*.println(euNorm(aul));  
 }  
 //Вывод собственных значений, полученных методом Якоби  
 public void showLambda() {  
 Formatter fmt = new Formatter();  
 for (int i = 0; i < a.length; i++) {  
 fmt.format("% 15.8f", lambda[i]);  
 }  
 System.*out*.println(fmt);  
 fmt.close();  
 }  
 //Вывод собственных векторов, полученных методом Якоби  
 public void showT() {  
 Formatter fmt = new Formatter();  
 for (double[] row : t) {  
 for (double i : row) {  
 fmt.format("% 15.8f", i);  
 }  
 fmt.format("\n");  
 }  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
 //Вывод векторов r = Ax - λx  
 public void showResiduals() {  
 Formatter fmt = new Formatter();  
 double rJ;  
 for (int i = 0; i < a.length; i++) {  
 fmt.format("λ%-3d:", i + 1);  
 for (int j = 0; j < a.length; j++) {  
 rJ = 0.;  
 for (int k = 0; k < a.length; k++) {  
 if (k <= j) {  
 rJ += a[j][k] \* t[k][i];  
 }  
 else {  
 rJ += a[k][j] \* t[k][i];  
 }  
  
 }  
 rJ -= lambda[i] \* t[j][i];  
 fmt.format("% 14.7f", rJ);  
 }  
 fmt.format("\n");  
 }  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
 //Евклидова норма  
 private double euNorm(double[] z) {  
 double norm = 0;  
 for (double i : z) {  
 norm += Math.*pow*(i, 2);  
 }  
 norm = Math.*sqrt*(norm);  
 return norm;  
 }  
}

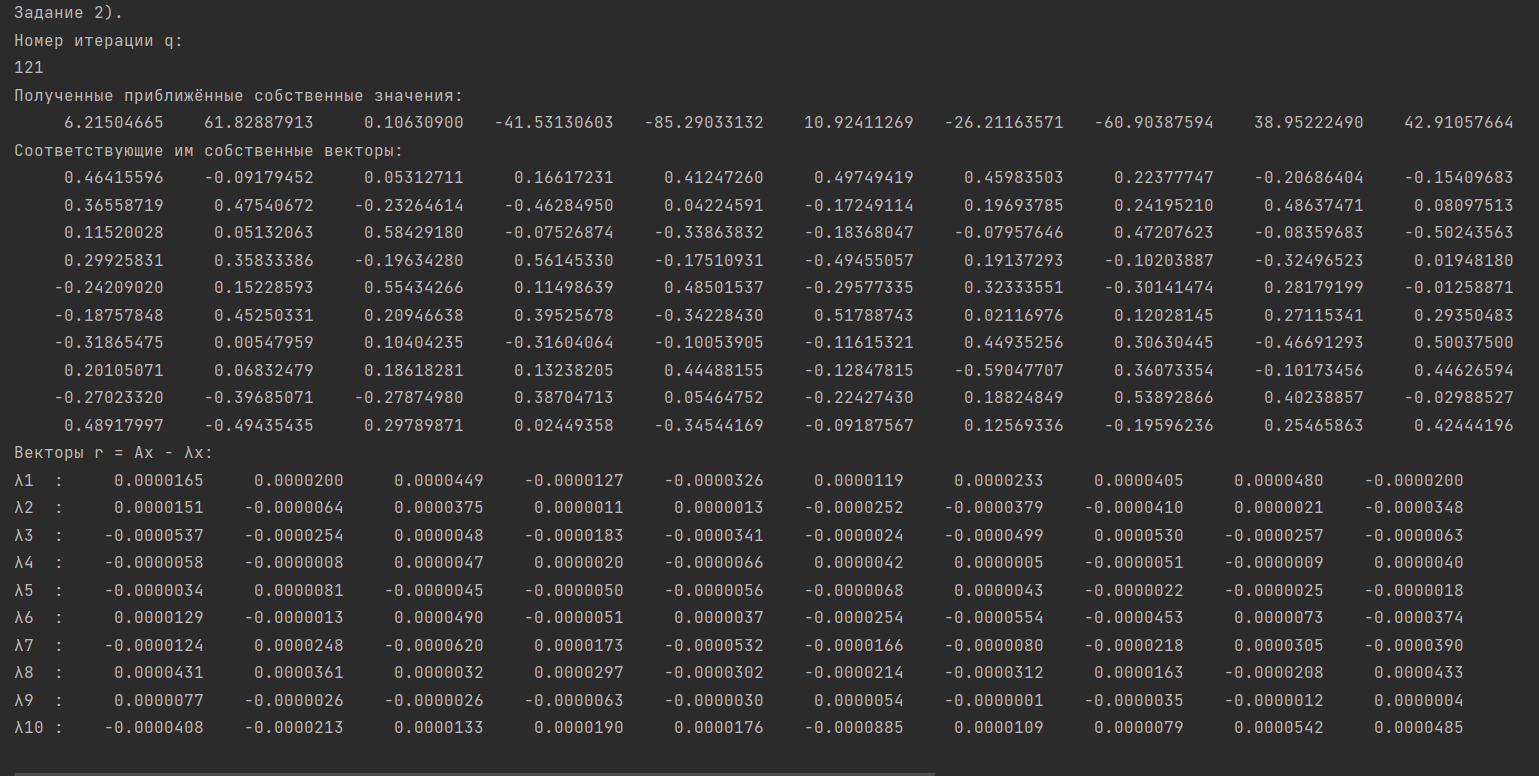
Файл Main.java:

public class Main {  
 private static final int *N* = 10;  
 public static void main(String[] args) {  
 Matrix myMatrix = new Matrix(*N*);  
 System.*out*.println("Матрица A:");  
 myMatrix.showA();  
 System.*out*.println("Задание 1).");  
 int q;  
 q = myMatrix.powerIt();  
 System.*out*.println("Начальный вектор y0:");  
 myMatrix.showY0();  
 System.*out*.println("Номер итерации q:");  
 if (q != 0) {  
 System.*out*.println(q);  
 } else {  
 System.*out*.println("Итерационный процесс расходится или было достигнуто максимальное количество итераций.");  
 }  
 System.*out*.println("Полученное наибольшее по модулю собственное значение λ1:");  
 myMatrix.showLambda1();  
 System.*out*.println("Соответствующий ему нормированный собственный вектор x1:");  
 myMatrix.showX1();  
 System.*out*.println("Вектор Au(q) - λ1u(q):");  
 myMatrix.showAul();  
 System.*out*.println("Евклидова норма ||Au(q) - λ1u(q)||:");  
 myMatrix.showEuAul();  
 System.*out*.println("Задание 2).");  
 q = myMatrix.rotateIt();  
 System.*out*.println("Номер итерации q:");  
 if (q != 0) {  
 System.*out*.println(q);  
 } else {  
 System.*out*.println("Итерационный процесс расходится.");  
 }  
 System.*out*.println("Полученные приближённые собственные значения:");  
 myMatrix.showLambda();  
 System.*out*.println("Соответствующие им собственные векторы:");  
 myMatrix.showT();  
 System.*out*.println("Векторы r = Ax - λx:");  
 myMatrix.showResiduals();  
 }  
}

**Результаты**

****

****

****

**Выводы**

1) Стандартный степенной метод сходится со скоростью геометрической прогрессии, при этом он будет сходится медленнее, если есть собственное значение, близкое по модулю к максимальному. При симметрической матрице метод скалярных произведений сходится в 2 раза быстрее стандартного степенного метода.

2) При может произойти переполнение векторов , поэтому используется нормировка. Нормировка не меняет собственных значений матрицы.

3) Из последних двух таблиц видно, что используемый вариант остановки процесса метода Якоби не гарантирует получения собственных векторов с точностью ε.