Лабораторная работа 6

«Итерационный степенной метод»

Необходимый для выполнения работы теоретический материал и формулы имеются в файле «Степенной метод».

**Задание 1.** Разработать программу вычисления наибольшего и второго по величине модуля собственных значений и соответствующих им собственных векторов симметричной матрицы.

Матрицу задать следующим образом (воспользоваться программой лабораторной работы «Решение систем на основе разложения симметричных матриц»):

* недиагональные элементы *ai,j*, *i<j*, выбираются из чисел 0, –1, –2, –3, *–*4 произвольным образом; если *i>j*, то полагается *ai,j*=*aj,i*.
* *ai,i=*, 2≤*i*≤*n*;
* *a*11*=*.

Для вычислений выбрать *n* – одно из чисел в пределах от 10 до 12. В качестве языка программирования выбрать C или C++, использовать тип float.

Для вычисления наибольшего по модулю собственного значения и соответствующего собственного вектора использовать формулы из пункта Случай 1 (файл «Степенной метод»). В формуле λ1≈ (формула вспомогательная, не для счета) выбирать такое *i*, для которого достигается ). В этом случае справедливо (это уже для счета) λ1≈.

Для вычисления наибольшего по модулю собственного значения использовать также формулу λ1≈.

Вывести на печать *uk* (приближенно это собственный вектор, соответствующий собственному значению λ1) на итерациях 46–50 (*k=*46,…,50).

Для обоих случаев (общий случай и случай использования симметричности матрицы): вывести на печать приближенное λ1 для *k=*46,…,50; вычислить вектор *vk+*1–λ1*uk* (это, для проверки насколько оказались хороши вычисленные λ1 и *uk*: вектор *vk+*1–λ1*uk* равен «погрешности» *Auk*–λ1*uk*) для *k=*50, вычислить и вывести на печать кубическую норму (максимум-норму) вектора *vk+*1–λ1*uk*.

Для вычисления второго по величине модуля собственного значения использовать формулу λ2≈. Выбирать такое *i*, для которого достигается ).

Рассмотреть три случая:

1) *m=*30, λ1 берется при *k=*50 по формуле для не обязательно симметричной матрицы;

2) *m=*50, λ1 берется при *k=*50 по формуле для не обязательно симметричной матрицы;

3) *m=*50, λ1 берется при *k=*50 по формуле для симметричной матрицы.

Во всех случаях λ1 уже было вычислено ранее.

Для этих трех случаев: вывести на печать приближенное λ2, вычислить и вывести на печать собственный вектор *x* (*x*≈*vk+*1–λ1*uk*), соответствующий собственному значению λ2, вычислить вектор *Ax*–λ2*x*, вычислить и вывести на печать кубическую норму этого вектора.

Содержание работы должно включать следующие пункты.

1. Постановка задачи.
2. Расчетные формулы.
3. Входные данные.
4. Листинг программы. Комментарии обязательны.
5. Выходные данные.
6. Выводы.

**Задание 2.** (Задание дополнительное, для повышения оценки текущей успеваемости.) Вычислительные эксперименты для получения наибольшего по модулю собственного значения и соответствующего ему собственного вектора случайно заданной матрицы четвертого порядка.

Взять матрицу (именно ее), случайным образом заданную в лабораторной работе «Метод Данилевского», и попытаться получить собственное значение и соответствующий ему собственный вектор согласно случаев файла «Степенной метод».

**Расчетные формулы**

λ1≈

λ1≈

λ2≈

**Входные данные**

n = 12;

**Листинг программы**

#include <cmath>

#include <iomanip>

#include <iostream>

#include <random>

using namespace std;

//Размерность матрицы

int n = 12;

//Нахождение случайного числа

int Rand(int L, int R)

{

static random\_device rd;

static mt19937 gen(rd());

uniform\_int\_distribution<> dis(L, R);

return dis(gen);

}

//Вывод матрицы

void PrintMatrix(float\*\* A)

{

cout << "A:" << endl;

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

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

cout << fixed << setprecision(3) << setw(10) << A[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

//Вывод столбца

void PrintArray(float\* x)

{

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

cout << fixed << setprecision(15) << setw(20) << x[i] << " ";

cout << endl

<< endl;

}

//Умножение матрицы на столбец

void Multiplication(float\*\* A, float\* x, float\* ans)

{

float s;

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

s = 0;

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

s += A[i][j] \* x[j];

}

ans[i] = s;

}

}

pair<float,int> norm(float\* v) {

float ans = fabs(v[0]);

int num = 0;

for (int i = 1; i < n; ++i) {

if (fabs(v[i]) > ans) {

ans = fabs(v[i]);

num = i;

}

}

return pair<float,int>(ans,num);

}

int sign(float x) {

if (x > 0) {

return 1;

} else if (x < 0){

return -1;

} else {

return 0;

}

}

void SaveArray(float\* v, float\* x) {

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

x[i] = v[i];

}

}

void Iterations1(float\*\* A, int m) {

float\* u = new float[n];

u[0] = 1;

for (int i = 1; i < n; ++i) {

u[i] = 0;

}

float\* v = new float[n];

float\* um = new float[n];

float\* vm = new float[n];

float\* vm1 = new float[n];

float\* denom = new float[n];

float\* x = new float[n];

pair<float,int> nv;

for (int k = 1; true; ++k) {

Multiplication(A, u, v);

if (k >= 46) {

cout << "u " << k << ":" << endl;

PrintArray(u);

}

if (k == 50) {

int num = norm(v).second;

float l1 = v[num] \* sign(u[num]);

cout << "l1: " << l1 << endl;

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

denom[i] = vm[i] - l1 \* um[i];

}

int nd = norm(denom).second;

float l2 = (vm1[nd] \* norm(vm).first - l1 \* vm[nd]) / denom[nd];

cout << "l2: " << l2 << endl;

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

x[i] = v[i] - l1 \* u[i];

}

cout << "x:" << endl;

PrintArray(x);

Multiplication(A, x, v);

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

v[i] -= l2 \* x[i];

}

cout << "Ax-l2x: " << endl;

PrintArray(v);

cout << "|| Ax-l2 ||: " << norm(v).first << endl;

delete[]v;

delete[]u;

delete[]um;

delete[]vm;

delete[]vm1;

delete[]denom;

delete[]x;

break;

}

nv = norm(v);

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

u[i] = v[i] / nv.first;

}

if (k == m - 1) {

SaveArray(u, um);

}

if (k == m) {

SaveArray(v, vm);

}

if (k == m + 1) {

SaveArray(v, vm1);

}

}

}

void Iterations2(float\*\* A) {

float\* u = new float[n];

u[0] = 1;

for (int i = 1; i < n; ++i) {

u[i] = 0;

}

float\* v = new float[n];

float\* um = new float[n];

float\* vm = new float[n];

float\* vm1 = new float[n];

float\* denom = new float[n];

float\* x = new float[n];

pair<float, int> nv;

for (int k = 1; true; ++k) {

Multiplication(A, u, v);

if (k >= 46) {

cout << "u " << k << ":" << endl;

PrintArray(u);

}

if (k == 50) {

int num = norm(v).second;

float l1 = v[num] \* sign(u[num]);

cout << "l1: " << l1 << endl;

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

denom[i] = vm[i] - l1 \* um[i];

}

int nd = norm(denom).second;

float l2 = (v[nd] \* norm(vm).first - l1 \* vm[nd]) / denom[nd];

cout << "l2: " << l2 << endl;

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

x[i] = v[i] - l1 \* u[i];

}

cout << "x:" << endl;

PrintArray(x);

Multiplication(A, x, v);

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

v[i] -= l2 \* x[i];

}

cout << "Ax-l2x: " << endl;

PrintArray(v);

cout << "|| Ax-l2 ||: " << norm(v).first << endl;

delete[]v;

delete[]u;

delete[]um;

delete[]vm;

delete[]vm1;

delete[]denom;

delete[]x;

break;

}

nv = norm(v);

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

u[i] = v[i] / nv.first;

}

if (k == 48) {

SaveArray(u, um);

}

if (k == 49) {

SaveArray(v, vm);

}

}

}

void Iterations3(float\*\* A) {

float\* u = new float[n];

u[0] = 1;

for (int i = 1; i < n; ++i) {

u[i] = 0;

}

float\* v = new float[n];

float\* um = new float[n];

float\* vm = new float[n];

float\* vm1 = new float[n];

float\* denom = new float[n];

float\* x = new float[n];

pair<float, int> nv;

for (int k = 1; true; ++k) {

Multiplication(A, u, v);

if (k >= 46) {

cout << "u " << k << ":" << endl;

PrintArray(u);

}

if (k == 50) {

float num = 0, den = 0;

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

num += v[i] \* u[i];

den += u[i] \* u[i];

}

float l1 = num/den;

cout << "l1: " << l1 << endl;

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

denom[i] = vm[i] - l1 \* um[i];

}

int nd = norm(denom).second;

float l2 = (v[nd] \* norm(vm).first - l1 \* vm[nd]) / denom[nd];

cout << "l2: " << l2 << endl;

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

x[i] = v[i] - l1 \* u[i];

}

cout << "x:" << endl;

PrintArray(x);

Multiplication(A, x, v);

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

v[i] -= l2 \* x[i];

}

cout << "Ax-l2x: " << endl;

PrintArray(v);

cout << "|| Ax-l2 ||: " << norm(v).first << endl;

delete[]v;

delete[]u;

delete[]um;

delete[]vm;

delete[]vm1;

delete[]denom;

delete[]x;

break;

}

nv = norm(v);

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

u[i] = v[i] / nv.first;

}

if (k == 48) {

SaveArray(u, um);

}

if (k == 49) {

SaveArray(v, vm);

}

}

}

int main()

{

setlocale(LC\_ALL, "rus");

//k=0

//Создание матрицы n\*n

float\*\* A = new float\*[n];

float\*\* save = new float\*[n];

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

A[i] = new float[n];

save[i] = new float[n];

}

//Инициализация

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

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

if (i != j) {

if (i > j)

A[i][j] = A[j][i];

else

A[i][j] = Rand(-4, 0);

}

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

float s = 0;

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

if (i != j)

s -= A[i][j];

A[i][i] = s;

}

A[0][0]++;

//Вывод матрицы

PrintMatrix(A);

Iterations1(A, 30);

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

Iterations2(A);

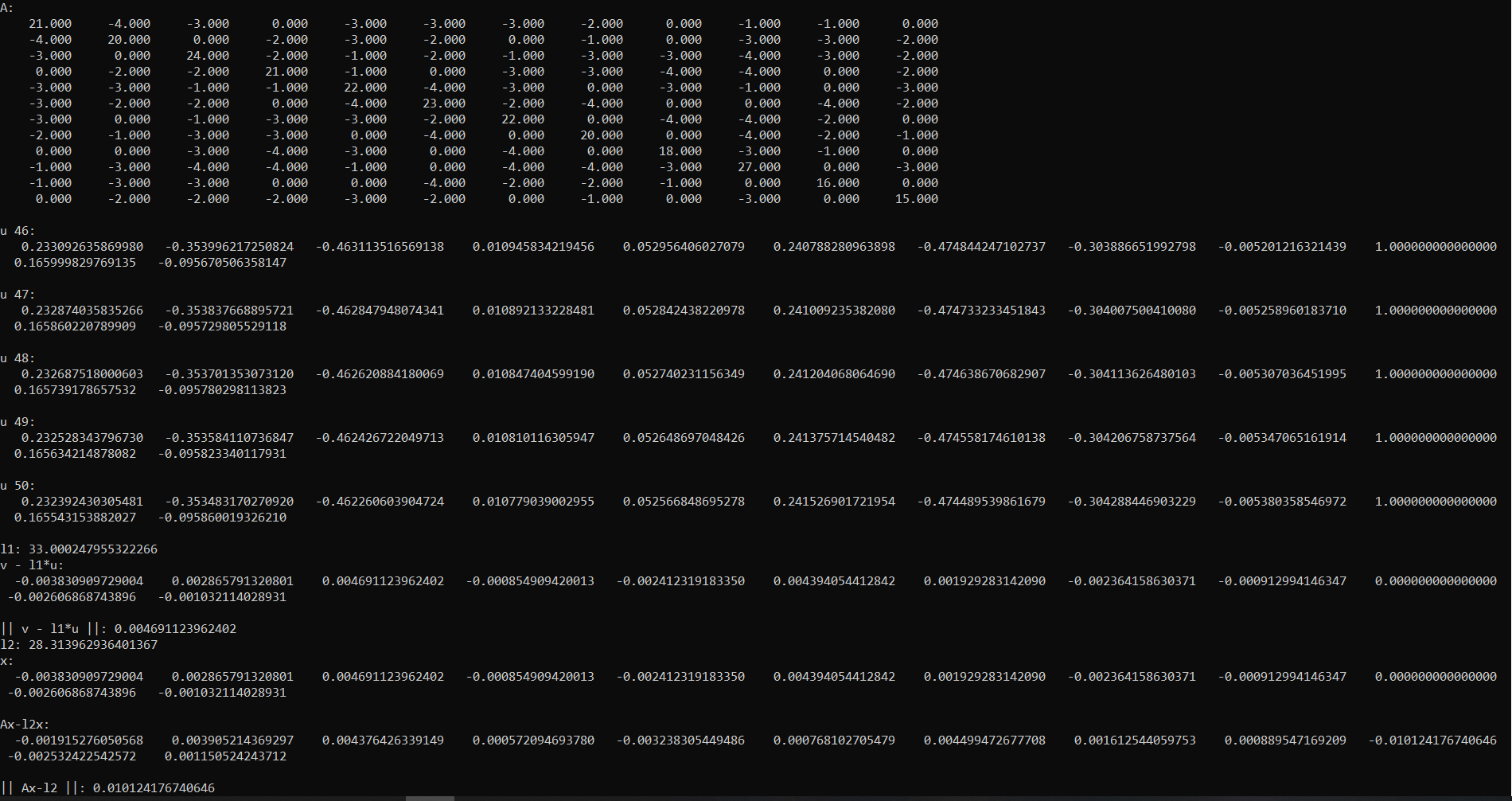
cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

Iterations3(A);

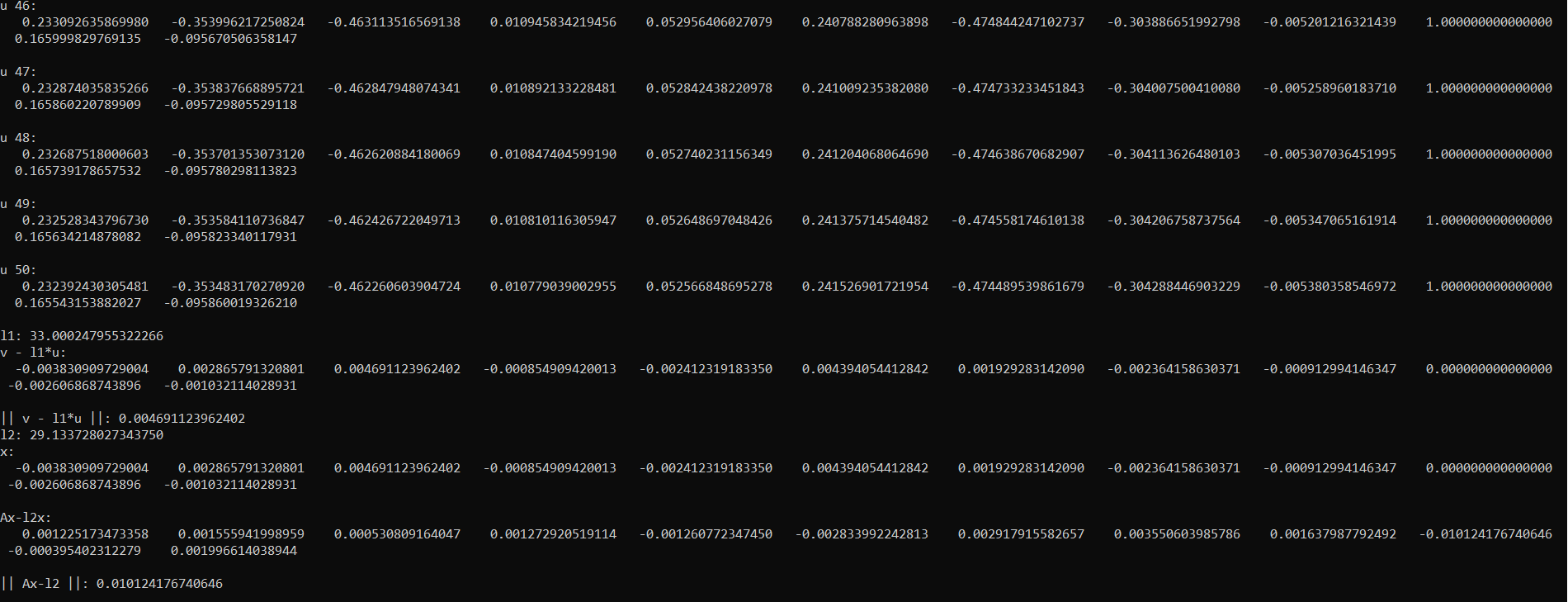
}

**Выходные данные**

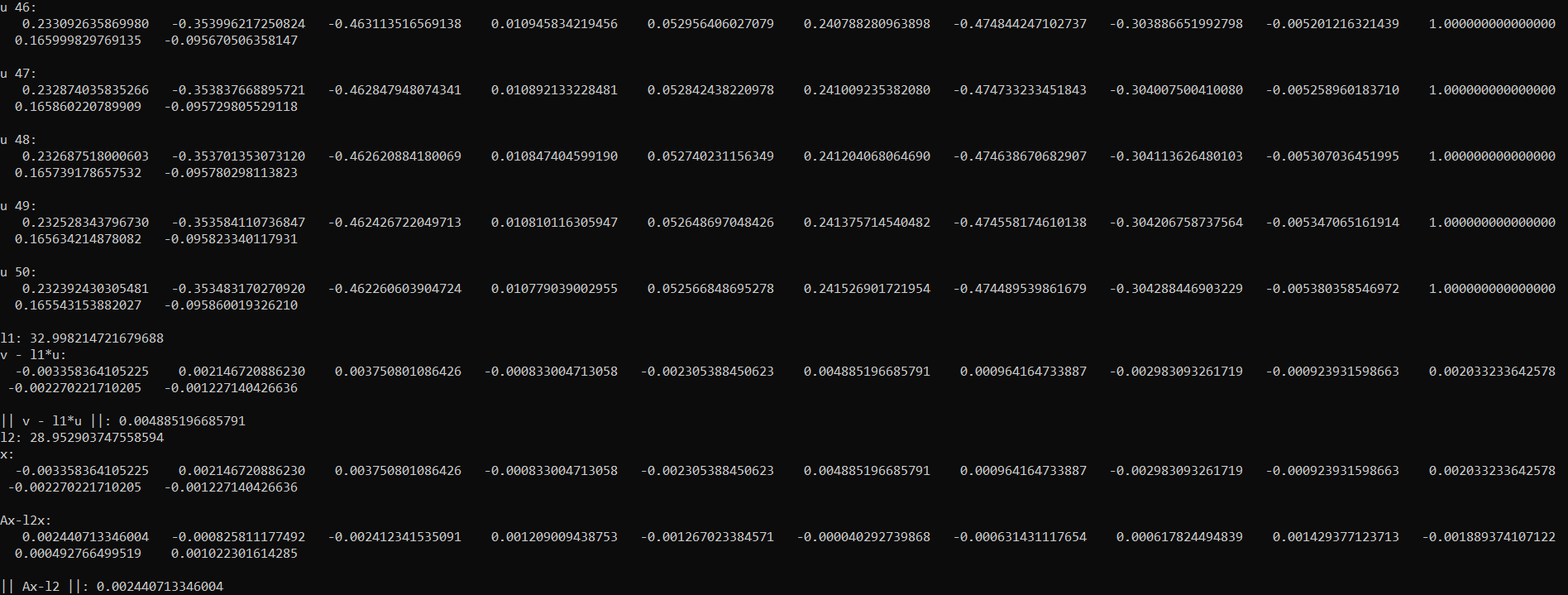
**1.**

****

**2.**

****

**3.**

****

**Вывод**

Метод, использующийся для симметричных матриц. имеет более быструю сходимость, нежели метод для обычных матриц, это означает что на ряде тестов второй метод вычисления собственных значений не даст точный результат при 50 итерациях. При вычислении второго собственного значения, лучший результат при m=30, так как при других вариантах вычитаются близкие по значению числа, что влечет потерю значащих символов, а также деление на ноль.