Лабораторная работа

«Метод Данилевского»

Арсений Анищенко

2 группа

**1. Постановка задачи**

Разработать программу приведения матрицы к канонической

форме Фробениуса методом Данилевского (регулярный случай);

сохранить матрицы, используемые для получения собственных векторов.

Для вычислений:

* N = 4 – порядок матрицы;
* при заполнении матрицы A использовать случайные числа из диапазона от −100 до 100. Для получения случайных чисел использовать библиотечную функцию rand(), подключив хедер stdlib.h;
* использовать тип float;
* «ведущий элемент» на (n–k+1)-м шаге, k=n, n–1,... , 2, (=) должен быть отличным от нуля (рассматривается только такой регулярный случай);

Программно реализовать для рассматриваемого примера алгоритм приведения матрицы к канонической форме Фробениуса. Порядок матрицы n

должен быть в коде параметром (это требование не является обязательным).

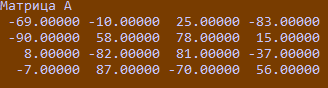
Сохранить матрицы Mn–1, Mn–2,... , M1, используемые для получения

собственных векторов (понадобятся для другой работы). Вывести на печать

матрицу A (входные данные), полученную каноническую форму Фробениуса

Φ, матрицы Mn–1, Mn–2, ... , M1, коэффициент p1 (полученный из формы Фробениуса), след матрицы Sp A (для контроля вычислений: должно приближенно выполняться равенство p1=Sp A=a1,1+a2,2+... +an,n).

**2. Входные данные.**



**3. Листинг программы.**

#include <iostream>

#include <ctime>

#include <cstdio>

#include <cstdlib>

#include <iomanip>

#include <fstream>

using namespace std;

const int N = 4;

int i, k;

float trace = 0;

void danilevskiy(float \*\*a) {

float \*\*m = new float\*[N];

float \*\*m\_rev = new float\*[N];

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

m[i] = new float[N];

m\_rev[i] = new float[N];

}

//Перебор итераций

for (int iter = N - 1; iter >= 1; --iter) {

//Заполнение матрицы M и обратной к ней на iй итерации

for (int i = 0; i < iter - 1; ++i) {

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

if (i == j) {

m[i][j] = m\_rev[i][j] = 1;

}

else {

m[i][j] = m\_rev[i][j] = 0;

}

}

}

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

if (j == iter - 1) {

m[iter - 1][j] = 1 / a[iter][iter - 1];

} else {

m[iter - 1][j] = -a[iter][j] / a[iter][iter - 1];

}

m\_rev[iter - 1][j] = a[iter][j];

}

for (int i = iter; i < N; ++i) {

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

if (i == j) {

m[i][j] = m\_rev[i][j] = 1;

}

else {

m[i][j] = m\_rev[i][j] = 0;

}

}

}

//Вывод матрицы M с индексом iter

cout << endl << "Матрица M с индексом " << iter << endl;

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

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

cout << fixed << setprecision(4) << setw(8) << m[i][j];

}

cout << endl;

}

//Выполнения преобразования A \* M

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

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

if (j != iter - 1) {

a[i][j] = m[j][j] \* a[i][j] + m[iter - 1][j] \* a[i][iter - 1];

}

}

a[i][iter - 1] \*= m[iter - 1][iter - 1];

}

//Умножение на обратную к M слева

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

float sum = 0;

for (int z = 0; z < N; ++z) {

sum += m\_rev[iter - 1][z] \* a[z][j];

}

a[iter - 1][j] = sum;

}

}

//Вывод канонической формы Фробениуса

cout << endl << "Каноническая форма Фробениуса" << endl;

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

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

cout << fixed << setprecision(4) << setw(14) << a[i][j];

}

cout << endl;

}

cout << endl << "p1 = " << a[0][0] << endl;

cout << "След = " << trace << endl;

}

int main() {

setlocale(LC\_ALL, ".1251");

srand(time(0));

float \*\*a = new float\*[N];

//Заполнение и вывод матрицы А

cout << "Матрица А" << endl;

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

a[i] = new float[N];

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

a[i][j] = rand() % 201 - 100;

if (i == j) trace += a[i][j];

cout << fixed << setprecision(5) << setw(10)<< a[i][j];

}

cout << endl;

}

danilevskiy(a);

return 0;

}

**4. Выходные данные.**

