**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №4

**«Итерационные методы решения**

**проблемы собственных значений»**

**Выполнил:**

Крученков Евгений Андреевич

студент 2 курса 9 группы,

специальность

“прикладная математика”

**Преподаватель:**

Ассистент кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2021

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2-3

Краткие теоретические сведения ------------------------------------------------ 3-4

Листинг программы ---------------------------------------------------------------- 5-11

Результаты --------------------------------------------------------------------------- 12-13

Выводы ------------------------------------------------------------------------------- 13

**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. Выполнить следующий вычислительный эксперимент. Сгенерировать симметрическую матрицу A порядка 10. Задать начальный ненулевой вектор и ε = .

Степенным методом найти наибольшее по модулю собственное значение и соответствующий ему собственный вектор матрицы A. В результатах необходимо привести следующую информацию:

• начальный вектор ;

• номер итерации q, при которой достигнута требуемая точность;

• приближенное наибольшее по модулю собственное значение и соответствующий ему нормированный собственный вектор.

• вектор ;

•

**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 . Выполнить следующий вычислительный эксперимент. Взять симметрическую матрицу A сгенерированную в первом задании. Задать ε = . Методом вращений найти все собственные значения и соответствующие им собственные векторы матрицы A. В результатах необходимо привести следующую информацию:

• номер итерации q, при которой достигнута требуемая точность;

• приближенные собственные значения и соответствующие им собственные векторы ;

• векторы .

**Краткие теоретические сведения**

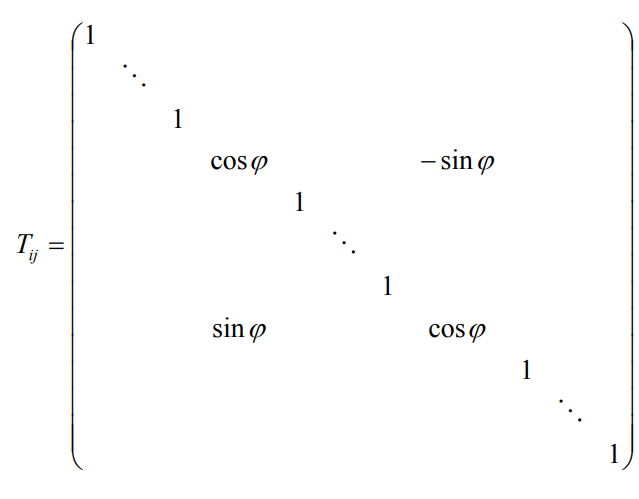
***Степенной метод (с нормировкой через скалярное произведение)***

,

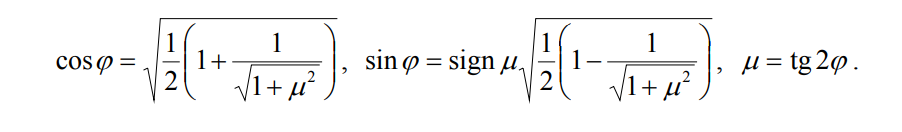
*k = 1,2,… ; r – невязка*

при k

***Метод итерационных вращений (Якоби)***

******

При :

******

При :

*k = 1,2,…;*

При kстремится к диагональной матрице с собственными значениями Ф на главной диагонали.

**Листинг программы**

**Header.h**

#pragma once

#include <iostream>

#include <cstdio>

#include <iomanip>

#include <fstream>

#include <string>

#include <ctime>

#include <cmath>

using namespace std;

class Matr

{

double\*\* M;

int n;

public:

Matr()

{

this->n = 1;

M = new double\* [n];

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

{

M[i] = new double[n];

}

srand(time(0));

M[0][0] = rand() % 200 - 100;

}

Matr(int n)

{

this->n = n;

M = new double\* [n];

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

{

M[i] = new double[i+1];

}

srand(time(0));

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

{

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

{

M[i][j]= rand() % 200 - 100;

}

}

}

Matr(const Matr& B)

{

this->n = B.n;

M = new double\* [n];

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

{

M[i] = new double[i+1];

}

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

{

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

{

M[i][j] = B.M[i][j];

}

}

}

~Matr()

{

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

{

delete[] M[i];

}

delete[] M;

}

friend ostream& operator <<(ostream& os, Matr& A)

{

os << fixed << showpoint << setprecision(3);

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

{

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

{

os << A.M[i][j] << " ";

}

for (int j = i; j < A.n; j++)

{

os << A.M[j][i] << " ";

}

os << endl;

}

return os;

}

Matr& operator = (const Matr& B)

{

if (this == &B)

return \*this;

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

{

delete[] M[i];

}

n = B.n;

delete[] M;

M = new double\* [n];

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

{

M[i] = new double[n-i];

}

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

{

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

{

M[i][j] = B.M[i][j];

}

}

return \*this;

}

Matr operator \* (const Matr& B)

{

if (n != B.n) throw "Матрицы имеют разные размеры";

Matr P(n);

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

{

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

{

P.M[i][j] = 0;

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

{

P.M[i][j] += M[i][k] \* B.M[k][j];

}

}

}

return P;

}

void StepennoiMetod()

{

double E = pow(10, -7);

double\* Y = new double[n];

double\* U = new double[n];

double norm\_y;

double norm;

double norm\_pred = 0;

double L;

int k\_max = 0;

int k = 0;

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

{

Y[i] = 1;

}

while (true)

{

norm\_y = 0;

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

{

norm\_y += Y[i] \* Y[i];

}

norm\_y = sqrt(norm\_y);

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

{

U[i] = Y[i] / norm\_y;

}

//

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

{

Y[i] = 0;

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

{

Y[i] += M[i][j] \* U[j];

}

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

{

Y[i] += M[j][i] \* U[j];

}

}

L = 0;

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

{

L += Y[i] \* U[i];

}

//

norm = 0;

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

{

norm += (Y[i] - L \* U[i]) \* (Y[i] - L \* U[i]);

}

norm = sqrt(norm);

if (norm < E) break;

else

{

if (norm > norm\_pred)

k\_max++;

else

k\_max = 0;

}

if (k\_max > 50)

{

cout << "Алгоритм расходится. Количество итераций: "<< k << endl;

break;

}

//

norm\_pred = norm;

k++;

}

cout << "Начальный вектор Y0:" << endl;

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

{

cout<<"1"<<endl;

}

cout << "Номер итерации k, при котором была достигнута требуемая точность:" <<k<< endl;

cout << "Приближенное наибольшее по модулю собственное значение:" << L << endl;

cout << "Соответствующий ему вектор U:" << endl;

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

{

cout << U[i] << endl;

}

cout << "Вектор AU-LU:" << scientific<< endl;

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

{

cout << Y[i] - L \* U[i] << endl;

}

cout << "Норма AU-LU:" << scientific << norm << endl;

}

void MetodVrashchenij()

{

bool t\_cr = true;

double E = pow(10, -7);

double cos\_f, sin\_f;

int max\_j, max\_i;

double norm;

int k = 0, k\_max = 0;

double norm\_pred = 0;

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

double\*\* A1 = new double\* [n];

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

{

A[i] = new double[n];

A1[i] = new double[n];

}

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

{

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

{

A1[i][j] = A[i][j] = M[i][j];

}

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

{

A1[i][j] = A[i][j] = M[j][i];

}

}

double\*\* T = new double\* [n];

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

{

T[i] = new double[n];

}

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

{

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

{

if(i==j)

T[i][j] = 1;

else

T[i][j] = 0;

}

}

while (true)

{

double max = 0;

max\_j = 1, max\_i = 0;

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

{

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

{

if (abs(A[i][j]) > max)

{

max = abs(A[i][j]);

max\_i = i;

max\_j = j;

}

}

}

if (A[max\_i][max\_i] == A[max\_j][max\_j])

{

cos\_f = 1 / sqrt(2);

sin\_f = -1 / sqrt(2);

}

else

{

double m = (2 \* A[max\_i][max\_j]) / (A[max\_i][max\_i] - A[max\_j][max\_j]);

cos\_f = sqrt((0.5 + (0.5 / sqrt(1 + m \* m))));

sin\_f = m \* sqrt((0.5 - (0.5 / sqrt(1 + m \* m)))) / abs(m);

}

double\* str1, \*str2;

str1 = new double[n];

str2 = new double[n];

if (t\_cr)

{

t\_cr = false;

T[max\_i][max\_i] = cos\_f;

T[max\_j][max\_j] = cos\_f;

T[max\_i][max\_j] = -sin\_f;

T[max\_j][max\_i] = sin\_f;

}

else

{

double \*mass1, \*mass2;

mass1 = new double[n];

mass2 = new double[n];

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

{

mass1[i] = (cos\_f \* T[i][max\_i] + (sin\_f)\*T[i][max\_j]);

mass2[i] = (-sin\_f) \* T[i][max\_i] + (cos\_f)\*T[i][max\_j];

}

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

{

T[i][max\_i] = mass1[i];

T[i][max\_j] = mass2[i];

}

}

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

{

str1[i] = cos\_f \* A[max\_i][i] + (sin\_f)\*A[max\_j][i];

str2[i] = (-sin\_f) \* A[max\_i][i] + (cos\_f)\*A[max\_j][i];

}

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

{

A[max\_i][i] = str1[i];

A[max\_j][i] = str2[i];

}

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

{

str1[i] = cos\_f \* A[i][max\_i] + (sin\_f)\*A[i][max\_j];

str2[i] = (-sin\_f) \* A[i][max\_i] + (cos\_f)\*A[i][max\_j];

}

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

{

A[i][max\_i] = str1[i];

A[i][max\_j] = str2[i];

}

norm = 0;

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

{

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

{

if(i!=j)

norm += A[i][j]\*A[i][j];

//cout << A[i][j] << " ";

}

//cout << endl;

}

if (norm < E) break;

else

{

if (norm > norm\_pred)

k\_max++;

else

k\_max = 0;

}

if (k\_max > 50)

{

cout << "Алгоритм расходится. Количество итераций: " << k << endl;

break;

}

//

norm\_pred = norm;

k++;

}

cout << "Номер итерации k, при котором была достигнута требуемая точность:" << k << endl;

cout << "Приближенные собственные значения:" << endl;

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

{

cout << fixed << showpoint << setprecision(3) << A[i][i] << endl;

}

cout << "Соответствующиe им собственные векторa:" << endl;

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

{

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

{

cout <<scientific<< T[i][j] << " ";

}

cout << endl;

}

cout << "Векторы AX-LX:" << endl;

double\*\* R = new double\*[n];

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

{

R[i] = new double[n];

}

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

{

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

{

R[i][j] = 0;

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

{

R[i][j] += A1[i][k] \* T[k][j];

}

R[i][j] -= T[i][j] \* A[j][j];

cout << scientific << R[i][j] << " ";

}

cout << endl;

}

cout << fixed << showpoint << setprecision(3);

}

};

**Main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "Header.h"

#include <fstream>

#include<time.h>

using namespace std;

ifstream fin;

ofstream fout;

int main()

{

srand(time(0));

setlocale(LC\_ALL, "ru");

Matr M(5);

cout << M << endl;

cout << endl << "Степенной метод:" << endl;

M.StepennoiMetod();

cout << endl << "Итерационный метод вращений:" << endl;

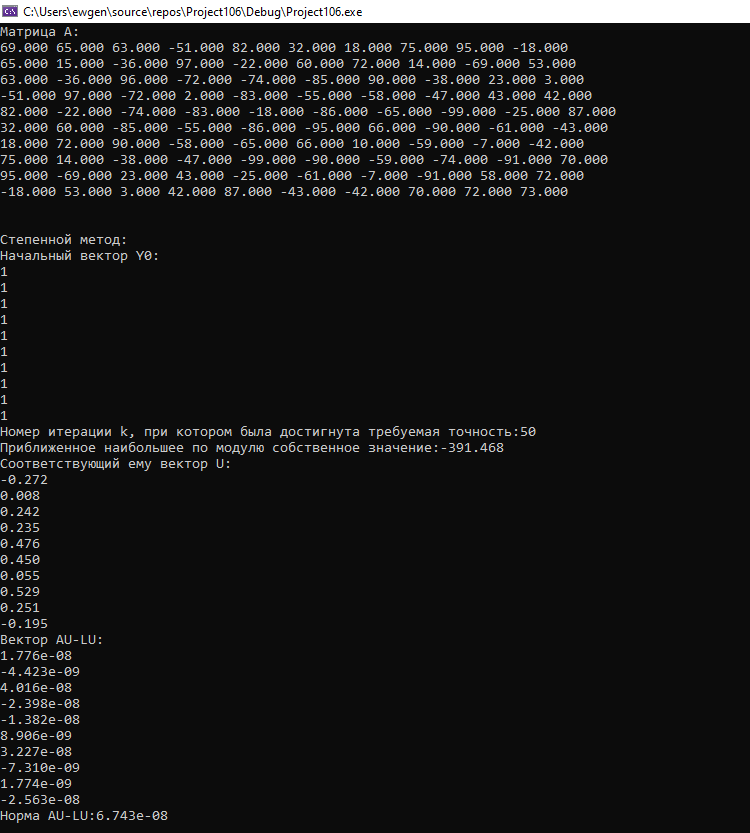
M.MetodVrashchenij();

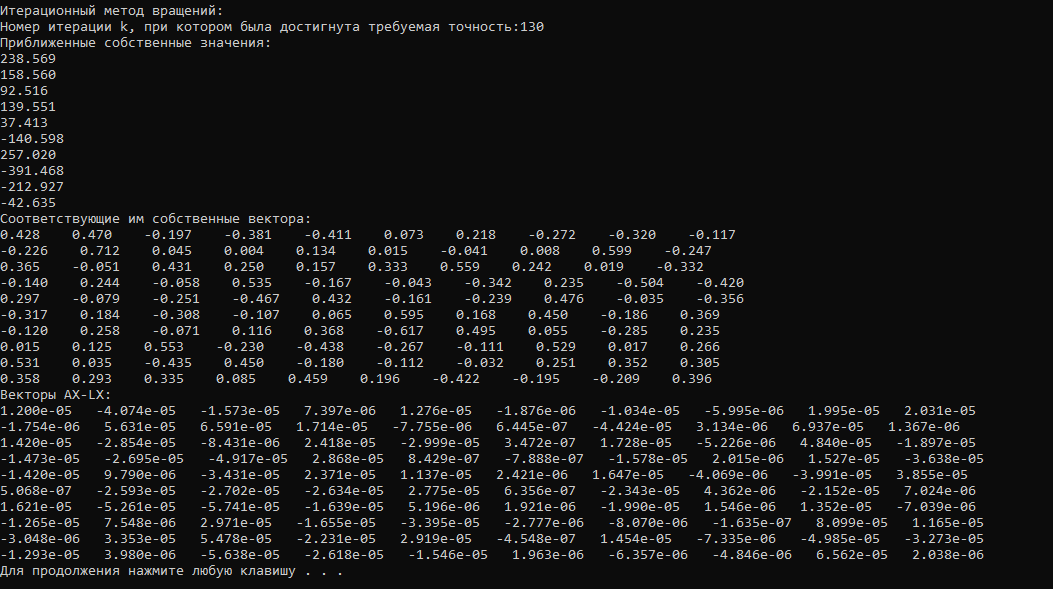
system("pause");

return 0;

}

**Результаты**

****

****

**Выводы**

1. Абсолютная погрешность при использовании итерационных методов будет зависеть от количества итераций и, следовательно, в нашем случае от наперёд заданного числа ε.
2. Для методa Якоби оптимальным элементом на k шаге является наибольший по модулю элемент выше главной диагонали.
3. Для методa Якоби при умножении справа и слева на матрицы вращений изменяются только i,j-строки и столбцы, поэтому выгоднее перемножать только их для ускорения вычислений.