**HỌ VÀ TÊN: HUỲNH HỮU HOÀNG**

**LỚP: 22T\_DT1**

**MSSV: 102220063**

#include <iostream>

#include <Eigen/Dense>

#include <vector>

using namespace std;

double a[100][100];

double transpose[100][100];

double S[100][100];

double U[100][100];

double E[100][100];

double V\_Transpose[100][100];

void nhanMaTran(double p[][100], double q[][100], int m, int l, int n, double res[][100]){

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

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

res[i][j] = 0;

for(int k = 0 ; k < l ; k++){

res[i][j] += p[i][k] \* q[k][j];

}

}

}

}

int main() {

int m, n; cin >> m >> n;

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

for(int j = 0 ; j < n ; j++) cin >> a[i][j];

}

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

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

transpose[i][j] = a[j][i];

}

}

nhanMaTran(transpose, a, n, m, n, S);

int size = m > n ? m : n;

// Nh?p kích thu?c c?a ma tr?n (c? hàng và c?t ph?i b?ng nhau)

// std::cout << "Nh?p kích thu?c c?a ma tr?n vuông (s? hàng = s? c?t): ";

// std::cin >> size;

// Khai báo ma tr?n vuông v?i kích thu?c nh?p vào

Eigen::MatrixXd A(size, size);

// Nh?p các ph?n t? c?a ma tr?n t? bàn phím

// std::cout << "Nh?p các ph?n t? c?a ma tr?n:\n";

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

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

// std::cout << "A(" << i << ", " << j << "): ";

// std::cin >> A(i, j);

A(i, j) = S[i][j];

}

}

// Tính toán eigenvalues và eigenvectors

Eigen::EigenSolver<Eigen::MatrixXd> es(A);

// Luu eigenvalues vào m?ng std::vector

std::vector<double> eigenvalues;

for (int i = 0; i < es.eigenvalues().size(); ++i) {

// eigenvalues.push\_back(abs(es.eigenvalues()[i].real()) < 1e-4 ? 0 : es.eigenvalues()[i].real());

eigenvalues.push\_back(es.eigenvalues()[i].real());

}

// Luu eigenvectors vào m?ng 2 chi?u std::vector

std::vector<std::vector<double>> eigenvectors;

for (int i = 0; i < es.eigenvectors().cols(); ++i) {

std::vector<double> vec;

for (int j = 0; j < es.eigenvectors().rows(); ++j) {

// vec.push\_back(abs(es.eigenvectors()(j, i).real()) < 1e-4 ? 0 : es.eigenvectors()(j, i).real());

vec.push\_back(es.eigenvectors()(j, i).real());

}

eigenvectors.push\_back(vec);

}

for (int i = 0; i < eigenvalues.size() - 1; i++){

int max\_idx = i;

for (int j = i + 1; j < eigenvalues.size(); j++){

if(eigenvalues[j] > eigenvalues[max\_idx]) max\_idx = j;

}

double tmp = eigenvalues[i];

eigenvalues[i] = eigenvalues[max\_idx];

eigenvalues[max\_idx] = tmp;

for(int k = 0 ; k < eigenvectors[i].size(); k++){

double temp = eigenvectors[i][k];

eigenvectors[i][k] = eigenvectors[max\_idx][k];

eigenvectors[max\_idx][k] = temp;

}

}

// In eigenvalues

// std::cout << "\nEigenvalues:\n";

// for (const auto& val : eigenvalues) {

// std::cout << val << " ";

// }

for(int i = 0 ; i < eigenvalues.size() ; i++){

E[i][i] = sqrt(eigenvalues[i]);

}

// In eigenvectors

// std::cout << "Eigenvectors:\n";

// for (const auto& vec : eigenvectors) {

// for (const auto& val : vec) {

// std::cout << val << " ";

// }

// std::cout << std::endl;

// }

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

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

V\_Transpose[i][j] = eigenvectors[i][j];

}

}

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

double tmp[100][100];

double v\_i[100][100];

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

v\_i[j][0] = eigenvectors[i][j];

// cout << v\_i[j][0] << " ";

}

// cout << endl;

nhanMaTran(a, v\_i, n, n, 1, tmp);

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

// cout << tmp[j][0] << " ";

tmp[j][0] /= sqrt(eigenvalues[i]);

U[j][i] = tmp[j][0];

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

}

// cout << endl;

}

cout << "U: " << endl;

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

for(int j = 0 ; j < m ; j++) cout << U[i][j] << " ";

cout << endl;

}

cout << "E: " << endl;

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

for(int j = 0 ; j < n ; j++) cout << E[i][j] << " ";

cout << endl;

}

cout << "VT: " << endl;

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

for(int j = 0 ; j < n ; j++) cout << V\_Transpose[i][j] << " ";

cout << endl;

}

return 0;

}

