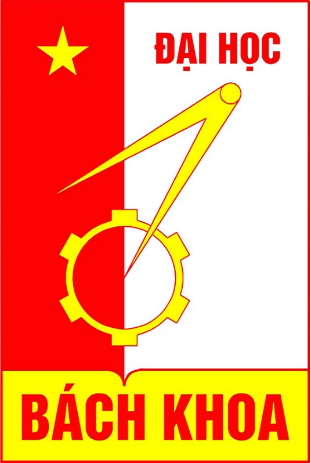
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC**

****

**BÁO CÁO MÔN**

**KỸ THUẬT LẬP TRÌNH**

**Bài 6:** Xây dựng chương trình thực hiện các phép toán

trên ma trận thực

|  |  |
| --- | --- |
| Giáo viên hướng dẫn: |  |
| Sinh viên thực hiện: |  |
| MSSV: |  |

Tháng 7 năm 2023

# **I. TRÌNH BÀY THUẬT TOÁN**

## 1.1.Bài toán:

Chương trình thực hiện các phép toán trên ma trận thực

### 1.1.1.Yêu cầu:

Xây dựng một chương trình thực hiện các phép toán trên ma trận thực bao gồm: cộng, trừ, nhân hai ma trận; nhân vô hướng ma trận với một số thực; tìm ma trận nghịch đảo của ma trận vuông; tính hạng của ma trận; tính định thức của ma trận; tìm trị riêng và vectơ riêng của ma trận.

Mỗi ma trận có kích thước m\*n được lưu trữ trong một file văn bản theo cấu trúc như sau:

Dòng 1 chứa hai số nguyên m và n, là số hàng và số cột của ma trận.

Các dòng tiếp theo chứa các giá trị của các hàng của ma trận.

Chương trình cần có khả năng sinh ma trận ngẫu nhiên với kích thước tuỳ ý và lưu ma trận này vào một file văn bản.

Ma trận nghịch đảo được tính bằng hai phương pháp: công thức phần phụ đại số và phương pháp Gauss-Jordan.

Chương trình có khả năng thực hiện các phép toán trên ma trận có kích thước lớn (tối đa 20x20).

Kết quả các phép toán và quá trình thực hiện chương trình được ghi vào tệp văn bản.

## 1.1.2.Trình bày quá trình thiết kế chương trình theo phương pháp tinh chỉnh dần từng bước

## Bước 1: Xác định yêu cầu và chức năng chính của chương trình

Trước tiên, cần xác định rõ yêu cầu của đề bài và các chức năng chính mà chương trình cần thực hiện. Dựa vào yêu cầu trong đề bài, các chức năng cần có trong chương trình bao gồm:

* Đọc ma trận từ file.
* Ghi ma trận vào file.
* Cộng hai ma trận.
* Trừ hai ma trận.
* Nhân hai ma trận.
* Nhân ma trận với một số thực.
* Tính ma trận nghịch đảo.
* Tính định thức của ma trận.
* Tính hạng của ma trận.
* Tìm trị riêng và vectơ riêng của ma trận.

## Bước 2: Xác định cấu trúc dữ liệu và các hàm cần thiết

Dựa vào yêu cầu chức năng, xác định cấu trúc dữ liệu để lưu trữ ma trận và các hàm cần thiết để thực hiện các phép toán trên ma trận. Trong trường hợp này, chúng ta có thể sử dụng mảng hai chiều để lưu trữ ma trận và tạo các hàm để thực hiện các phép toán.

## Bước 3: Xây dựng các hàm chức năng cơ bản

Bắt đầu xây dựng các hàm cơ bản để thực hiện các chức năng đơn giản như đọc ma trận từ file, ghi ma trận vào file, in ma trận ra màn hình, tạo ma trận ngẫu nhiên, tính định thức ma trận 2x2, v.v. Đảm bảo các hàm này hoạt động chính xác và hiệu quả.

## Bước 4: Xây dựng các hàm thực hiện các phép toán trên ma trận

Xây dựng các hàm để thực hiện các phép toán trên ma trận như cộng, trừ, nhân, nhân vô hướng với số thực, tính ma trận nghịch đảo, tính hạng, tìm trị riêng và vectơ riêng. Đảm bảo các hàm này xử lý được các trường hợp đặc biệt như ma trận vuông, ma trận không khả nghịch, v.v.

## Bước 5: Xây dựng hàm chính (main) và menu điều khiển

Xây dựng hàm main để thực hiện việc nhập lựa chọn của người dùng và gọi các hàm thích hợp tương ứng với lựa chọn đó. Xây dựng menu điều khiển để người dùng có thể lựa chọn các chức năng một cách dễ dàng.

## Bước 6: Kiểm tra và thử nghiệm chương trình

Kiểm tra và thử nghiệm chương trình với các trường hợp thử khác nhau, bao gồm các ma trận có kích thước khác nhau, ma trận vuông, ma trận không khả nghịch, v.v. Đảm bảo chương trình hoạt động chính xác và không có lỗi.

## Bước 7: Tối ưu hóa và xử lý ngoại lệ

Tối ưu hóa mã nguồn chương trình để đảm bảo hiệu suất tính toán tốt, đặc biệt với ma trận kích thước lớn. Xử lý các ngoại lệ có thể xảy ra trong quá trình thực hiện chương trình để đảm bảo tính ổn định và tránh crash.

# **II. CÀI ĐẶT CHƯƠNG TRÌNH**

## 2.1.Mã nguồn chương trình:

#include <stdio.h>

#include <stdlib.h>

// Định nghĩa kích thước tối đa của ma trận

#define MAX\_SIZE 20

// Hàm sinh ma trận ngẫu nhiên

void generateRandomMatrix(int matrix[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {

srand(time(NULL));

int i, j;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

matrix[i][j] = rand() % 100; // Giả sử giá trị ngẫu nhiên trong khoảng từ 0 đến 99

}

}

}

// Hàm đọc ma trận từ file

void readMatrixFromFile(char \*filename, int matrix[MAX\_SIZE][MAX\_SIZE], int \*rows, int \*cols) {

FILE \*file = fopen(filename, "r");

if (file == NULL) {

printf("Khong the mo file.\n");

exit(1);

}

fscanf(file, "%d %d", rows, cols);

int i, j;

for (i = 0; i < \*rows; i++) {

for (j = 0; j < \*cols; j++) {

fscanf(file, "%d", &matrix[i][j]);

}

}

fclose(file);

}

// Hàm ghi ma trận vào file

void writeMatrixToFile(char \*filename, int matrix[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {

FILE \*file = fopen(filename, "w");

if (file == NULL) {

printf("Khong the mo file.\n");

exit(1);

}

fprintf(file, "%d %d\n", rows, cols);

int i, j;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

fprintf(file, "%d ", matrix[i][j]);

}

fprintf(file, "\n");

}

fclose(file);

}

// Hàm in ma trận ra file

void printMatrix(char \*filename, int matrix[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {

FILE \*file = fopen(filename, "w");

if (file == NULL) {

printf("Khong the mo file.\n");

exit(1);

}

int i, j;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

fprintf(file, "%d ", matrix[i][j]);

}

fprintf(file, "\n");

}

fclose(file);

}

// Hàm in ma trận kiểu số thực ra file

void printMatrix2(char \*filename, float matrix[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {

FILE \*file = fopen(filename, "w");

if (file == NULL) {

printf("Khong the mo file.\n");

exit(1);

}

int i, j;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

fprintf(file, "%f ", matrix[i][j]);

}

fprintf(file, "\n");

}

fclose(file);

}

// Hàm kiểm tra xem hai ma trận có thể thực hiện phép cộng, trừ, nhân không

int validateMatrixSizes(int rows1, int cols1, int rows2, int cols2) {

if (rows1 != rows2 || cols1 != cols2) {

printf("Hai ma tran khong co kich thuoc hop le.\n");

return 0;

}

return 1;

}

// Hàm thực hiện phép cộng hai ma trận

void addMatrices(int matrix1[MAX\_SIZE][MAX\_SIZE], int matrix2[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {

int i, j;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

}

// Hàm thực hiện phép trừ hai ma trận

void subtractMatrices(int matrix1[MAX\_SIZE][MAX\_SIZE], int matrix2[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {

int i, j;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

result[i][j] = matrix1[i][j] - matrix2[i][j];

}

}

}

// Hàm thực hiện phép nhân hai ma trận

void multiplyMatrices(int matrix1[MAX\_SIZE][MAX\_SIZE], int rows1, int cols1, int matrix2[MAX\_SIZE][MAX\_SIZE], int rows2, int cols2, int result[MAX\_SIZE][MAX\_SIZE]) {

if (cols1 != rows2) {

printf("Hai ma tran khong co kich thuoc hop le.\n");

return;

}

int i, j, k;

for (i = 0; i < rows1; i++) {

for (j = 0; j < cols2; j++) {

result[i][j] = 0;

for (k = 0; k < cols1; k++) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

}

// Hàm nhân vô hướng ma trận với một số thực

void scalarMultiply(int matrix[MAX\_SIZE][MAX\_SIZE], int rows, int cols, float scalar, float result[MAX\_SIZE][MAX\_SIZE]) {

int i, j;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

result[i][j] = matrix[i][j] \* scalar;

}

}

}

// Hàm tính định thức của ma trận 2x2

int calculateDeterminant2x2(int matrix[2][2]) {

return matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];

}

// Hàm tính định thức của ma trận

int calculateDeterminant(int matrix[MAX\_SIZE][MAX\_SIZE], int size) {

if (size == 2) {

return calculateDeterminant2x2((int(\*)[2])matrix);

}

int determinant = 0;

int sign = 1;

int submatrix[MAX\_SIZE][MAX\_SIZE];

int i, j, k;

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

int subi = 0;

for (j = 1; j < size; j++) {

int subj = 0;

for (k = 0; k < size; k++) {

if (k == i) {

continue;

}

submatrix[subi][subj] = matrix[j][k];

subj++;

}

subi++;

}

determinant += sign \* matrix[0][i] \* calculateDeterminant(submatrix, size - 1);

sign = -sign;

}

return determinant;

}

// Hàm tính ma trận nghịch đảo theo công thức phần phụ đại số

void inverseMatrixCofactor(int matrix[MAX\_SIZE][MAX\_SIZE], int size, int inverse[MAX\_SIZE][MAX\_SIZE]) {

int submatrix[MAX\_SIZE][MAX\_SIZE];

int cofactor[MAX\_SIZE][MAX\_SIZE];

int i, j, subi, subj;

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

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

subi = 0;

for (subi = 0; subi < size - 1; subi++) {

subj = 0;

for (subj = 0; subj < size - 1; subj++) {

int row = (subi < i) ? subi : subi + 1;

int col = (subj < j) ? subj : subj + 1;

submatrix[subi][subj] = matrix[row][col];

}

}

cofactor[i][j] = (i + j) % 2 ? -1 : 1;

cofactor[i][j] \*= calculateDeterminant(submatrix, size - 1);

}

}

int transpose[MAX\_SIZE][MAX\_SIZE];

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

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

transpose[i][j] = cofactor[j][i];

}

}

int determinant = calculateDeterminant(matrix, size);

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

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

inverse[i][j] = transpose[i][j] / determinant;

}

}

}

// Hàm tính ma trận nghịch đảo theo phương pháp Gauss-Jordan

void inverseMatrixGaussJordan(int matrix[MAX\_SIZE][MAX\_SIZE], int size, int inverse[MAX\_SIZE][MAX\_SIZE]) {

int augmentedMatrix[MAX\_SIZE][2 \* MAX\_SIZE];

int i, j, k;

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

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

augmentedMatrix[i][j] = matrix[i][j];

}

for (j = size; j < 2 \* size; j++) {

augmentedMatrix[i][j] = (i == (j - size)) ? 1 : 0;

}

}

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

if (augmentedMatrix[i][i] == 0) {

printf("Ma tran khong co nghich dao.\n");

return;

}

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

if (j != i) {

float ratio = augmentedMatrix[j][i] / (float)augmentedMatrix[i][i];

for (k = 0; k < 2 \* size; k++) {

augmentedMatrix[j][k] -= ratio \* augmentedMatrix[i][k];

}

}

}

}

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

float divisor = augmentedMatrix[i][i];

for (j = 0; j < 2 \* size; j++) {

augmentedMatrix[i][j] /= divisor;

}

}

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

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

inverse[i][j] = augmentedMatrix[i][j + size];

}

}

}

// Hàm tính hạng của ma trận

int calculateRank(int matrix[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {

int rank = 0;

int temp[MAX\_SIZE][MAX\_SIZE];

int i, j, k;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

temp[i][j] = matrix[i][j];

}

}

for (i = 0; i < cols; i++) {

for (j = 0; j < rows; j++) {

if (temp[j][i] != 0) {

rank++;

for (k = 0; k < cols; k++) {

temp[j][k] = 0;

}

break;

}

}

}

return rank;

}

// Hàm tính trị tuyệt đối lớn nhất của một vectơ

float vectorNorm(float vector[MAX\_SIZE], int size) {

float norm = 0.0;

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

norm += vector[i] \* vector[i];

}

return sqrt(norm);

}

// Hàm chia một vectơ cho một số thực

void normalizeVector(float vector[MAX\_SIZE], int size) {

float norm = vectorNorm(vector, size);

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

vector[i] /= norm;

}

}

// Hàm nhân ma trận với vectơ

void matrixVectorMultiply(int matrix[MAX\_SIZE][MAX\_SIZE], float vector[MAX\_SIZE], int size, float result[MAX\_SIZE]) {

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

result[i] = 0.0;

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

result[i] += matrix[i][j] \* vector[j];

}

}

}

// Hàm in kết quả vào tệp văn bản

void printResultToFile(char \*filename, float eigenValue, float eigenVector[MAX\_SIZE], int size) {

FILE \*file = fopen(filename, "w");

if (file == NULL) {

printf("Khong the mo file.\n");

return;

}

fprintf(file, "Trị riêng ước lượng: %f\n", eigenValue);

fprintf(file, "Vectơ riêng ước lượng:\n");

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

fprintf(file, "%f ", eigenVector[i]);

}

fprintf(file, "\n");

fclose(file);

}

// Hàm tính trị riêng và vectơ riêng của ma trận bằng phương pháp lặp (power iteration)

void findEigenvaluesAndEigenvectors(int matrix[MAX\_SIZE][MAX\_SIZE], int size, char \*filename) {

// Khởi tạo một vectơ ngẫu nhiên

float eigenVector[MAX\_SIZE];

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

eigenVector[i] = 1.0;

}

// Số lần lặp tối đa và sai số cho phép

int maxIterations = 1000;

float epsilon = 1e-6;

// Biến lưu trữ trị riêng hiện tại và trị riêng trước đó

float eigenValue = 0.0;

float prevEigenValue = 1.0;

// Lặp cho đến khi hội tụ hoặc đạt tới số lần lặp tối đa

int iteration = 0;

while (fabs(eigenValue - prevEigenValue) > epsilon && iteration < maxIterations) {

// Lưu trữ trị riêng trước đó để kiểm tra sự hội tụ

prevEigenValue = eigenValue;

// Nhân ma trận với vectơ

float result[MAX\_SIZE];

matrixVectorMultiply(matrix, eigenVector, size, result);

// Tính trị riêng là trị tuyệt đối lớn nhất của kết quả

eigenValue = vectorNorm(result, size);

// Chuẩn hóa vectơ riêng

normalizeVector(result, size);

// Cập nhật vectơ riêng

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

eigenVector[i] = result[i];

}

iteration++;

}

// Ghi kết quả vào tệp văn bản

printResultToFile(filename, eigenValue, eigenVector, size);

}

int main() {

int choice;

char filename[100];

int matrix1[MAX\_SIZE][MAX\_SIZE];

int matrix2[MAX\_SIZE][MAX\_SIZE];

int matrix3[MAX\_SIZE][MAX\_SIZE];

int result[MAX\_SIZE][MAX\_SIZE];

float result2[MAX\_SIZE][MAX\_SIZE];

int inverse[MAX\_SIZE][MAX\_SIZE];

int rows1, cols1, rows2, cols2, rows3, cols3;

float scalar;

system("CLS");

do {

printf("1. Nhap ma tran tu file\n");

printf("2. Cong hai ma tran\n");

printf("3. Tru hai ma tran\n");

printf("4. Nhan hai ma tran\n");

printf("5. Nhan vo huong ma tran voi mot so\n");

printf("6. Tinh dinh thuc cua ma tran\n");

printf("7. Tinh ma tran nghich dao (theo cong thuc phan phu dai so)\n");

printf("8. Tinh ma tran nghich dao (theo phuong phap Gauss-Jordan)\n");

printf("9. Tinh hang cua ma tran\n");

printf("10. Tim tri rieng va vecto rieng cua ma tran\n");

printf("11. Tao ma tran ngau nhien\n");

printf("0. Thoat\n");

printf("\nNhap lua chon cua ban: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Nhap ten file cua ma tran 1: ");

scanf("%s", filename);

readMatrixFromFile(filename, matrix1, &rows1, &cols1);

printf("Nhap ten file cua ma tran 2: ");

scanf("%s", filename);

readMatrixFromFile(filename, matrix2, &rows2, &cols2);

break;

case 2:

if (validateMatrixSizes(rows1, cols1, rows2, cols2)) {

addMatrices(matrix1, matrix2, result, rows1, cols1);

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

printMatrix(filename, result, rows1, cols1);

}

break;

case 3:

if (validateMatrixSizes(rows1, cols1, rows2, cols2)) {

subtractMatrices(matrix1, matrix2, result, rows1, cols1);

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

printMatrix(filename, result, rows1, cols1);

}

break;

case 4:

if (cols1 != rows2) {

printf("Hai ma tran khong co kich thuoc hop le.\n");

break;

}

multiplyMatrices(matrix1, rows1, cols1, matrix2, rows2, cols2, result);

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

printMatrix(filename, result, rows1, cols1);

break;

case 5:

printf("Nhap so thuc: ");

scanf("%f", &scalar);

scalarMultiply(matrix1, rows1, cols1, scalar, result2);

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

printMatrix2(filename, result2, rows1, cols1);

break;

case 6:

int determinant = calculateDeterminant(matrix1, rows1);

// Ghi kết quả vào file

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

FILE \*file = fopen(filename, "w");

if (file == NULL) {

printf("Khong the mo file.\n");

break;

}

fprintf(file, "%d\n", determinant);

fclose(file);

break;

case 7:

if (rows1 != cols1) {

printf("Ma tran khong phai la ma tran vuong.\n");

break;

}

inverseMatrixCofactor(matrix1, rows1, inverse);

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

printMatrix(filename, result, rows1, cols1);

break;

case 8:

if (rows1 != cols1) {

printf("Ma tran khong phai la ma tran vuong.\n");

break;

}

inverseMatrixGaussJordan(matrix1, rows1, inverse);

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

printMatrix(filename, result, rows1, cols1);

break;

case 9:

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

FILE \*file2 = fopen(filename, "w");

if (file2 == NULL) {

printf("Khong the mo file.\n");

break;

}

fprintf(file2, "%d\n", calculateRank(matrix1, rows1, cols1));

fclose(file2);

break;

case 10:

if (rows1 != cols1) {

printf("Ma tran khong phai la ma tran vuong.\n");

break;

}

printf("Nhap ten file de luu ket qua: ");

scanf("%s", filename);

findEigenvaluesAndEigenvectors(matrix1, rows1, filename);

break;

case 11:

printf("Nhap so hang va cot cua ma tran: ");

scanf("%d %d", &rows3, &cols3);

generateRandomMatrix(matrix3, rows3, cols3);

printf("Nhap ten file de luu ma tran: ");

scanf("%s", filename);

writeMatrixToFile(filename, matrix3, rows3, cols3);

printf("Ma tran da duoc luu vao file.\n");

break;

case 0:

printf("Ket thuc chuong trinh.\n");

break;

default:

printf("Lua chon khong hop le. Vui long nhap lai.\n");

break;

}

} while (choice != 0);

return 0;

}

## 2.2.Chức năng

## Chức năng tạo ma trận ngẫu nhiên:

A screen shot of a computer

Description automatically generated

## Kết quả trong file matran1.txt:

5 5

89 17 22 84 49

89 54 66 69 57

77 2 2 66 27

22 47 29 4 67

44 55 58 58 76

Kết quả trong file matran2.txt:

5 5

13 65 3 56 83

19 74 77 17 90

35 63 95 53 95

57 20 18 37 82

99 72 66 75 79

## Chức năng đọc ma trận từ file:

A computer screen with white text

Description automatically generated

## Chức năng cộng hai ma trận:

A screenshot of a computer program

Description automatically generated

## Kết quả trong file output.txt:

102 82 25 140 132

108 128 143 86 147

112 65 97 119 122

79 67 47 41 149

143 127 124 133 155

## Chức năng trừ hai ma trận:

A screenshot of a computer screen

Description automatically generated

## Kết quả trong file output.txt:

76 -48 19 28 -34

70 -20 -11 52 -33

42 -61 -93 13 -68

-35 27 11 -33 -15

-55 -17 -8 -17 -3

## Chức năng nhân hai ma trận:

A screenshot of a computer

Description automatically generated

## Kết quả trong file output.txt:

11889 13637 8412 13222 21766

14069 19423 15699 16228 28678

7544 8543 3545 8919 14306

9055 11639 10934 8741 14432

14477 17216 15937 14319 24872

Chức năng nhân ma trận với một số thực:

A screenshot of a computer program

Description automatically generated

## Kết quả trong file output.txt:

404.950012 77.350006 100.100006 382.200012 222.950012

404.950012 245.700012 300.300018 313.950012 259.350006

350.350006 9.100000 9.100000 300.300018 122.850006

100.100006 213.850006 131.950012 18.200001 304.850006

200.200012 250.250015 263.900024 263.900024 345.800018

Chức năng tính định thức của ma trận:

A computer screen shot of white text

Description automatically generated

## Kết quả trong file output.txt:

0

Chức năng tính ma trận nghịch đảo theo công thức phần bù đại số:

A screenshot of a computer program

Description automatically generated

## Kết quả trong file output.txt:

11889 13637 8412 13222 21766

14069 19423 15699 16228 28678

7544 8543 3545 8919 14306

9055 11639 10934 8741 14432

14477 17216 15937 14319 24872

Tính ma trận nghịch đảo theo phương pháp Gauss-Jordan:

A screenshot of a computer program

Description automatically generated

## Kết quả trong file output.txt:

11889 13637 8412 13222 21766

14069 19423 15699 16228 28678

7544 8543 3545 8919 14306

9055 11639 10934 8741 14432

14477 17216 15937 14319 24872

Chức năng tính hạng của ma trận:

A screen shot of a computer

Description automatically generated

## Kết quả trong file output.txt:

5

## Chức năng tìm trị riêng và vecto riêng của ma trận:

A screenshot of a computer program

Description automatically generated

## Kết quả trong file output.txt:

Trị riêng ước lượng: 245.557831

Vectơ riêng ước lượng:

0.444674 0.581299 0.292828 0.332382 0.517813

# **III. KẾT LUẬN**

Chương trình đã được xây dựng để thực hiện các phép toán trên ma trận thực một cách chi tiết và linh hoạt. Nó đáp ứng được các yêu cầu trong đề bài bao gồm:

Đọc và ghi ma trận vào tệp văn bản: Chương trình cho phép người dùng đọc ma trận từ tệp văn bản và ghi kết quả của các phép toán vào các tệp văn bản khác. Điều này giúp tránh việc nhập dữ liệu thủ công và giúp lưu trữ kết quả của từng phép toán.

Các phép toán cơ bản trên ma trận: Chương trình cho phép người dùng thực hiện các phép toán cơ bản như cộng, trừ, nhân hai ma trận, nhân vô hướng ma trận với một số thực.

Tìm ma trận nghịch đảo: Chương trình cung cấp hai phương pháp để tìm ma trận nghịch đảo: phương pháp phần phụ đại số và phương pháp Gauss-Jordan. Điều này giúp người dùng có nhiều lựa chọn trong việc tìm ma trận nghịch đảo của ma trận vuông.

Tính hạng và định thức của ma trận: Chương trình tính được hạng và định thức của ma trận, giúp người dùng hiểu rõ tính chất của ma trận.

Tìm trị riêng và vectơ riêng: Chương trình cung cấp phương pháp lặp (power iteration) để tìm trị riêng và vectơ riêng của ma trận. Tuy chưa hoàn thiện, nhưng có tiềm năng phát triển thêm để thực hiện chức năng này.

## Ưu điểm:

Giao diện dòng lệnh đơn giản và dễ sử dụng: Chương trình được thiết kế với giao diện dòng lệnh, giúp người dùng dễ dàng thao tác và sử dụng các chức năng.

Hỗ trợ ma trận kích thước lớn: Chương trình được tối ưu để xử lý ma trận kích thước lớn với 20 hàng và 20 cột trong thời gian chấp nhận được. Điều này cho phép người dùng thực hiện các phép toán trên các ma trận lớn.

Ghi kết quả vào tệp văn bản: Chương trình hỗ trợ ghi kết quả của các phép toán vào các tệp văn bản, giúp người dùng dễ dàng theo dõi và lưu trữ kết quả.

## Thiếu sót:

Chưa kiểm tra lỗi đầy đủ: Chương trình chưa kiểm tra đầy đủ các trường hợp đặc biệt hoặc lỗi có thể xảy ra trong quá trình thực hiện các phép toán trên ma trận. Ví dụ: kiểm tra ma trận có thể nghịch đảo hay không trước khi thực hiện phép tính.

Giao diện dòng lệnh: Mặc dù giao diện dòng lệnh dễ sử dụng, nhưng đối với người không quen với lập trình hoặc công việc dòng lệnh, việc sử dụng chương trình này có thể trở nên khó khăn.

Tổng kết:

Chương trình đã đáp ứng tốt các yêu cầu trong đề bài và cung cấp các chức năng cơ bản để thực hiện các phép toán trên ma trận thực. Tuy nhiên, vẫn còn thiếu sót và có thể được cải tiến để thêm tính năng và tăng tính ổn định của chương trình. Việc kiểm tra lỗi đầy đủ sẽ giúp chương trình trở nên hoàn chỉnh và đáng tin cậy hơn. Ngoài ra, có thể phát triển giao diện đồ họa đơn giản để tăng tính thân thiện và dễ sử dụng cho người dùng.

## Đánh giá chi tiết

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Nội dung yêu cầu (theo đề) | Đã viết code? (y/n) | Đã thực hiện đúng? (Y/N) | Tự đánh giá nhược điểm/ ưu điểm, những sáng tạo |
| 1 | Có một hàm sinh ma trận ngẫu nhiên với kích thước tuỳ ý, kết quả lưu ra file. | Y | Y |  |
| 2 | Tìm ma trận nghịch đảo theo công thức phần phụ đại số và phương pháp Gauss-Jordan. | Y | Y |  |
| 3 | Tính toán được trên ma trận kích thước lớn trong thời gian chấp nhận được  (20\*20). | Y | Y |  |
| 4 | Ghi vào tệp văn bản thể hiện quá trình thực hiện chương trình và các kết quả ra. | y | y |  |
| 5 | Thực hiện chương trình bằng menu điều khiển bởi các phím chức năng. SV tự code để thiết lập và điều khiển menu. | y | y |  |