|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA  **KHOA CÔNG NGHỆ THÔNG TIN** |

**ĐỀ THI GIỮA HỌC KỲ VÀ BÀI LÀM**

Tên học phần: Toán ứng dụng CNTT

Mã học phần: Hình thức thi: *Tự luận*

Đề số: **01** Thời gian làm bài: 90 phút *(không kể thời gian chép/phát đề)*

Được sử dụng tài liệu khi làm bài.

**Họ tên: .............Hà Đức Kiên........... Lớp**: ......23T\_DT1...... **MSSV**: ..........102230194..............

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV\_HọTên.pdf và nộp bài thông qua MSTeam

***Câu 1*** (*2 điểm*):Cho số nguyên dương N (N>1). Viết chương trình bằng C/C++ có sử dụng hàm thực hiện:

* Tìm số hoàn hảo M gần N nhất (M<N), biết rằng N=3000
* Tìm các số nguyên tố bé hơn M, liệt kê và tính tổng của chúng.

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới:  *#include* <iostream>  *#include* <cmath>  *#include*<vector>  *using* *namespace* std;  vector*<int>* *primenumbers*;  *bool* *isPerfectNumber*(*int* num) {  *int* *sum* *=* 1;  *for* (*int* *i* *=* 2; *i* *<=* *sqrt*(num); *i++*) {  *if* (num *%* *i* *==* 0) {  *sum* *+=* *i*;  *if* (*i* *!=* num */* *i*) {  *sum* *+=* num */* *i*;  }  }  }  *return* *sum* *==* num;  }  *bool* *isPrime*(*int* num) {  *if* (num *<=* 1) *return* false;  *for* (*int* *i* *=* 2; *i* *<=* *sqrt*(num); *i++*) {  *if* (num *%* *i* *==* 0) *return* false;  }  *return* true;  }  *int* *main*() {  *int* *N* *=* 3000;  *int* *M* *=* *-*1;  *int* *j* *=* 1;  *for* (*int* *i* *=* *N* *-* 1; *i* *>* 1; *i--*) {  *if* (*isPerfectNumber*(*i*)) {  *M* *=* *i*;  *break*;  }  }  *if* (*M* *!=* *-*1) {  *cout* *<<* "Perfect number lower than " *<<* *N* *<<* " is: " *<<* *M* *<<* *endl*;  *int* *sumPrimes* *=* 0;  *cout* *<<* "Primenumer lower than " *<<* *M* *<<* " are: ";  *for* (*int* *i* *=* 2; *i* *<* *M*; *i++*) {  *if* (*isPrime*(*i*)) {  *primenumbers*.*push\_back*(*i*);  *sumPrimes* *+=* *i*;  }  }  *for*(*int* *i* *=* 0; *i<primenumbers*.*size*(); *i++*){  *cout<<primenumbers[i]<<*" ";  *if*(*i%*16 *==* 0 *&&* *i!=*0)  *cout<<endl*;  }  *cout* *<<* *endl*;  *cout* *<<* "Sum of primenumber lower than " *<<* *M* *<<* " la: " *<<* *sumPrimes* *<<* *endl*;  } *else* {  *cout* *<<* "No perfect number lower than " *<<* *N* *<<* *endl*;  }  *return* 0;  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới: |

***Câu 2: (****2 điểm****)*** *Cho hệ phương trình đồng dư sau*



* Viết chương trình C/C++ có sử dụng hàm giải hệ phương trình đồng dư trên.

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới:  *#include*<iostream>  *#include*<cmath>  *#include*<vector>  *#include*<time.h>  *using* *namespace* std;  *int* *x*,*y*, *gcd*;  *void* *extendedEuclid*(*int* a, *int* b){  *if*(b *==* 0){  *x* *=* 1;  *y* *=* 0;  *gcd* *=* a;  *return*;  }  *extendedEuclid*(b, a *%* b);  *int* *x1* *=* *y*;  *int* *y1* *=* *x* *-* (a */* b) *\** *y*;  *x* *=* *x1*;  *y* *=* *y1*;  *return*;  }  *long* *long* *modunlarInverse*(*int* a, *int* m){  *extendedEuclid*(a, m);  *return* (*x+*m)*%*m;  }  *void* *input*(*int* n, vector<*int*> *&*A, vector<*int*> *&*M){  *cout<<*"Nhap cac phuong trinh (a mod b): \n";  *for*(*int* *i* *=* 0; *i<*n; *i++*){  *cout<<*"Nhap phuong trinh so: "*<<i+*1*<<*": ";  *int* *a*, *m*;  *cin>>a*;  *cin>>m*;  A.*push\_back*(*a*);  M.*push\_back*(*m*);  }  }  *long* *long* *ChineseRemainderTheorem*(vector<*int*> A, vector<*int*> M){  *long* *long* *result* *=* 0;  *long* *long* *prod* *=* 1;  *for*(*int* *i* *=* 0; *i<*A.*size*(); *i++*){  *prod* *\*=* M*[i]*;  }  *for*(*int* *i* *=* 0; *i<*A.*size*(); *i++*){  *long* *long* *p* *=* *prod* */* M*[i]*;  *result* *=* (*result* *+* A*[i]* *\** *modunlarInverse*(*p*, M*[i]*) *\** *p*) *%* *prod*;  }  *return* *result*;  }  *int* *main*(){  *clock\_t* *start*, *end*;  *start* *=* *clock*();  *int* *n*;  *cout<<*"Enter number of equations: ";  *cin>>n*;  vector*<int>* *A*, *M*;  *input*(*n*, *A*, *M*);  *long* *long* *prod* *=* 1;  *for*(*int* *i* *=* 0; *i<M*.*size*(); *i++*){  *prod* *\*=* *M[i]*;  }  *long* *long* *result* *=* *ChineseRemainderTheorem*(*A*, *M*);  *cout<<*"Result: x = "*<<result<<*" + k"*<<prod<<endl*;  *end* *=* *clock*();  *cout<<*"Time taken: "*<<*(*double*)(*end-start*)*/CLOCKS\_PER\_SEC<<*"s\n";  *return* 0;  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới: |

***Câu 3*** (*3 điểm*): Cho ma trận A. Viết chương trình bằng c/c++ có sử dụng hàm thực hiện phân rã ma trận A (có hàm kiểm tra điều kiện phân rã).

1. Phân rã ma trận A

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):  Điều kiện của ma trận A là: ma trận vuông, đối xứng qua chéo chính  *#include*<iostream>  *#include*<cmath>  *#include*<cstring>  *#include*<stdlib.h>  *#include*<iomanip>  *using* *namespace* std;  *double* *L*[4][4], *D*[4];  *void* *inputMatrix*(*double* A[4][4], *int* n, *int* m);  *bool* *checkMatrixCholesky*(*double* A[4][4], *int* n, *int* m);  *bool* *SymmetricalMatrix*(*double* A[4][4], *int* n);  *void* *CholeskyLDL\_Decomposition*(*double* A[4][4], *double* L[4][4], *double* D[4], *int* n);  *void* *printMatrix*(*double* L[4][4], *double* D[4], *int* n);  *int* *main*(){  *double* *A*[4][4];  *int* *n*, *m*;  *cout* *<<* "Enter a number n: ";  *cin* *>>* *n*;  *cout* *<<* "Enter a number m: ";  *cin* *>>* *m*;  *cout<<fixed<<setprecision*(2);  *inputMatrix*(*A*, *n*, *m*);  *if*(*!checkMatrixCholesky*(*A*, *n*, *m*)){  *cout* *<<* "Matrix is not positive definite\n";  *return* 0;  } *else* {  *CholeskyLDL\_Decomposition*(*A*, *L*, *D*, *n*);  }  *printMatrix*(*L*, *D*, *n*);  *return* 0;  }  *void* *inputMatrix*(*double* A[4][4], *int* n, *int* m){  *if*(n *!=* m){  *cout* *<<* "Matrix is not square\n";  *return*;  }  *cout* *<<* "Enter matrix A: \n";  *for*(*int* *i* *=* 0; *i* *<* n; *i++*){  *for*(*int* *j* *=* 0; *j* *<* m; *j++*){  *cout* *<<* "Enter A[" *<<* *i* *<<* "][" *<<* *j* *<<* "]: ";  *cin* *>>* A[*i*][*j*];  }  }  }  *bool* *SymmetricalMatrix*(*double* A[4][4], *int* n){  *for*(*int* *i* *=* 0; *i* *<* n; *i++*){  *for*(*int* *j* *=* 0; *j* *<* n; *j++*){  *if*(A[*i*][*j*] *!=* A[*j*][*i*]){  *return* false;  }  }  }  *return* true;  }  *bool* *checkMatrixCholesky*(*double* A[4][4], *int* n, *int* m){  *if*(n *!=* m){  *return* false;  }  *if*(*!SymmetricalMatrix*(A, n)){  *return* false;  }  *return* true;  }  *void* *CholeskyLDL\_Decomposition*(*double* A[4][4], *double* L[4][4], *double* D[4], *int* n){  *memset*(L, 0, *sizeof*(*double*) *\** 4 *\** 4);  *for* (*int* *j* *=* 0; *j* *<* n; *j++*){  *for* (*int* *k* *=* 0; *k* *<* *j*; *k++*){  A[*j*][*j*] *-=* L[*j*][*k*] *\** L[*j*][*k*] *\** D[*k*];  }  D[*j*] *=* A[*j*][*j*];  *for*(*int* *i* *=* *j* *+* 1; *i* *<* n; *i++*){  L[*i*][*j*] *=* A[*i*][*j*];  *for*(*int* *k* *=* 0; *k* *<* *j*; *k++*){  L[*i*][*j*] *-=* L[*i*][*k*] *\** L[*j*][*k*] *\** D[*k*];  }  L[*i*][*j*] */=* D[*j*];  }  L[*j*][*j*] *=* 1.0;  }  }  *void* *printMatrix*(*double* L[4][4], *double* D[4], *int* n) {  *cout* *<<* "Matrix L | D | L^T\n";  *cout* *<<* *fixed* *<<* *setprecision*(5);  *for* (*int* *i* *=* 0; *i* *<* n; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  *if* (*j* *>* *i*)  *cout* *<<* *setw*(9) *<<* "0.00" *<<* " ";  *else*  *cout* *<<* *setw*(9) *<<* L[*i*][*j*] *<<* " ";  }  *cout* *<<* " | ";  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  *if* (*i* *==* *j*)  *cout* *<<* *setw*(9) *<<* D[*i*] *<<* " ";  *else*  *cout* *<<* *setw*(9) *<<* "0.00" *<<* " ";  }  *cout* *<<* " | ";  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  *if* (*j* *<* *i*)  *cout* *<<* *setw*(9) *<<* "0.00" *<<* " ";  *else*  *cout* *<<* *setw*(9) *<<* L[*j*][*i*] *<<* " ";  }  *cout* *<<* "\n";  }  *cout* *<<* "\n";  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới với  (sai số ): |

1. Phân rã **eigendecomposition** ma trận A

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):  *#include* <iostream>  *#include* <iomanip>  *#include* <vector>  *#include*<cmath>  *#define* *PI* 3.14159265  *#define* *MAX\_SIZE* 10  *using* *namespace* std;  *typedef* *double* *matrix*[*MAX\_SIZE*][*MAX\_SIZE*];  *double* *B1*[10][10] *=* {{1,0,0},{0,1,0},{0,0,1}};  *void* *inputMatrix*(*int* n, matrix A);  *void* *outputMatrix*(*int* n, *const* matrix A);  *void* *multiMatrix*(*const* matrix A, *const* matrix B, matrix C, *int* cola, *int* rowa, *int* rowb);  *void* *Danhilepski*(matrix A, matrix M, matrix M1, matrix B, *int* n);  *void* *solution*(*double* a, *double* b, *double* c, *double* d, *double* x[]);  *double* *determinant*(*const* matrix A, *int* n);  *bool* *invertMatrix*(*const* matrix A, matrix inverse, *int* n);  *void* *inputMatrix*(*int* n, matrix A) {  *cout* *<<* "Enter matrix elements:" *<<* *endl*;  *for* (*int* *i* *=* 0; *i* *<* n; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  *cin* *>>* A[*i*][*j*];  }  }  }  *void* *outputMatrix*(*int* n, *const* matrix A) {  *cout* *<<* *fixed* *<<* *setprecision*(2);  *for* (*int* *i* *=* 0; *i* *<* n; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  *cout* *<<* A[*i*][*j*] *<<* " ";  }  *cout* *<<* *endl*;  }  }  *void* *multiMatrix*(*const* matrix A, *const* matrix B, matrix C, *int* cola, *int* rowa, *int* rowb) {  *for* (*int* *i* *=* 0; *i* *<* cola; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* rowb; *j++*) {  C[*i*][*j*] *=* 0;  *for* (*int* *k* *=* 0; *k* *<* rowa; *k++*) {  C[*i*][*j*] *+=* A[*i*][*k*] *\** B[*k*][*j*];  }  }  }  }  *void* *Danhilepski*(matrix A, matrix M, matrix M1, matrix B, *int* n) {  *for* (*int* *k* *=* n *-* 2; *k* *>=* 0; *k--*) {  *for* (*int* *i* *=* 0; *i* *<* n; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  *if* (*i* *!=* *k*) {  *if* (*i* *==* *j*) {  M[*i*][*j*] *=* 1;  M1[*i*][*j*] *=* 1;  } *else* {  M[*i*][*j*] *=* 0;  M1[*i*][*j*] *=* 0;  }  } *else* {  M1[*i*][*j*] *=* A[*k+*1][*j*];  *if* (*j* *==* *k*) {  M[*i*][*j*] *=* 1 */* A[*k+*1][*k*];  } *else* {  M[*i*][*j*] *=* *-*A[*k+*1][*j*] */* A[*k+*1][*k*];  }  }  }  }  *multiMatrix*(A, M, B, n,n,n);  *multiMatrix*(M1, B, A, n, n, n);  *multiMatrix*(*B1*, M, B, n, n, n);  *for*(*int* *i* *=* 0; *i* *<* n; *i++*){  *for*(*int* *j* *=* 0; *j* *<* n; *j++*){  *B1*[*i*][*j*] *=* B[*i*][*j*];  }  }  }  }  *double* *determinant*(*const* matrix A, *int* n) {  *double* *det* *=* 0;  *if* (n *==* 1) {  *return* A[0][0];  }  *if* (n *==* 2) {  *return* A[0][0] *\** A[1][1] *-* A[0][1] *\** A[1][0];  }  matrix *temp*;  *for* (*int* *f* *=* 0; *f* *<* n; *f++*) {  *int* *temp\_i* *=* 0;  *for* (*int* *i* *=* 1; *i* *<* n; *i++*) {  *int* *temp\_j* *=* 0;  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  *if* (*j* *==* *f*) *continue*;  *temp*[*temp\_i*][*temp\_j*] *=* A[*i*][*j*];  *temp\_j++*;  }  *temp\_i++*;  }  *det* *+=* (*f* *%* 2 *==* 0 *?* 1 *:* *-*1) *\** A[0][*f*] *\** *determinant*(*temp*, n *-* 1);  }  *return* *det*;  }  *bool* *invertMatrix*(*const* matrix A, matrix inverse, *int* n) {  *double* *det* *=* *determinant*(A, n);  *if* (*det* *==* 0) {  *cout* *<<* "Matrix is singular and cannot be inverted." *<<* *endl*;  *return* false;  }  matrix *adjoint*;  *for* (*int* *i* *=* 0; *i* *<* n; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  matrix *temp*;  *int* *temp\_i* *=* 0;  *for* (*int* *x* *=* 0; *x* *<* n; *x++*) {  *for* (*int* *y* *=* 0; *y* *<* n; *y++*) {  *if* (*x* *!=* *i* *&&* *y* *!=* *j*) {  *temp*[*temp\_i*][(*y* *<* *j*) *?* *y* *:* (*y* *-* 1)] *=* A[*x*][*y*]; *// Đảm bảo chỉ số hàng và cột đúng*  }  }  *if* (*x* *!=* *i*) *temp\_i++*; *// Chỉ tăng temp\_i khi không phải hàng i*  }  *adjoint*[*j*][*i*] *=* (*pow*(*-*1, *i* *+* *j*) *\** *determinant*(*temp*, n *-* 1)) */* *det*; *// Adjoint transpose*  }  }  *for* (*int* *i* *=* 0; *i* *<* n; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* n; *j++*) {  inverse[*i*][*j*] *=* *adjoint*[*i*][*j*];  }  }  *return* true;  }  *void* *solution*(*double* a, *double* b, *double* c, *double* d, *double* x[]) {  *double* *delta* *=* b*\**b *-* 3*\**a*\**c;  *double* *k* *=* (9*\**a*\**b*\**c *-* 2*\**b*\**b*\**b *-* 27*\**a*\**a*\**d) */* (2 *\** *sqrt*(*fabs*(*delta\*delta\*delta*)));  *if*(*delta>*0){  *if*(*-*1 *<=* *k* *&&* *k* *<=* 1){  x[0] *=* (2*\*sqrt*(*delta*)*\*cos*(*acos*(*k*)*/*3)*-*b)*/*(3*\**a);  x[1] *=* (2*\*sqrt*(*delta*)*\*cos*((*acos*(*k*)*-*2*\*PI*)*/*3)*-*b)*/*(3*\**a);  x[2] *=* (2*\*sqrt*(*delta*)*\*cos*((*acos*(*k*)*+*2*\*PI*)*/*3)*-*b)*/*(3*\**a);  }  *if*(*k* *<* *-*1 *||* *k* *>* 1){  *k* *=* (*k* *<* 0) *?* *-k* *:* *k*;  *double* *l* *=* (*sqrt*(*delta*)*\*k*) */* 3*\**a*\*k*;  *double* *j* *=* (*k+sqrt*(*k\*k-*1));  *double* *i* *=* (*k-sqrt*(*k\*k-*1));  *double* *h*;  *if*(*i<*0){  *i* *=* *-i*;  *if*(*j<*0)  *h* *=* *-pow*(*fabs*(*j*), 1.0*/*3.0) *-* *pow*(*i*, 1.0*/*3.0);  *if*(*j>=*0)  *h* *=* *pow*(*j*, 1.0*/*3.0) *-* *pow*(*i*, 1.0*/*3.0);  } *else* {  *if*(*j<*0)  *h* *=* *-pow*(*fabs*(*j*), 1.0*/*3.0) *+* *pow*(*i*, 1.0*/*3.0);  *if*(*j>=*0)  *h* *=* *pow*(*j*, 1.0*/*3.0) *+* *pow*(*i*, 1.0*/*3.0);  }  x[0] *=* 1.0*\*h-*b*/*(3*\**a);  }  }  *if*(*delta* *==* 0){  *double* *l* *=* b*\**b*\**b *-* 27*\**a*\**a*\**d;  *if*(*l<*0){  *l* *=* *-l*;  x[0] *=* (*-*b *-pow*(1, 1.0*/*3.0))*/*(3*\**a);  } *else* {  x[0] *=* (*-*b *+pow*(1, 1.0*/*3.0))*/*(3*\**a);  }  }  *if*(*delta* *<* 0){  *delta* *=* *delta* *\** (*-*1.0);  *double* *j* *=* *k* *+* *sqrt*(*k\*k+*1);  *double* *i* *=* *k* *-* *sqrt*(*k\*k+*1);  *double* *h*;  *if*(*i<*0){  *i* *=* *-i*;  *if*(*j<*0)  *h* *=* *-pow*(*fabs*(*j*), 1.0*/*3.0) *-* *pow*(*i*, 1.0*/*3.0);  *if*(*j>=*0)  *h* *=* *pow*(*j*, 1.0*/*3.0) *-* *pow*(*i*, 1.0*/*3.0);  } *else* {  *if*(*j<*0)  *h* *=* *-pow*(*fabs*(*j*), 1.0*/*3.0) *+* *pow*(*i*, 1.0*/*3.0);  *if*(*j>=*0)  *h* *=* *pow*(*j*, 1.0*/*3.0) *+* *pow*(*i*, 1.0*/*3.0);  }  x[0] *=* *sqrt*(*delta*)*\*h/*(3*\**a)*-*b*/*(3*\**a);  }  }  *int* *main*() {  matrix *A*, *M*, *M1*, *B*;  *double* *lambda*[3] *=* {0,0,0};  *int* *n*;  *cout<<*"Enter n: ";  *cin>>n*;  *inputMatrix*(*n*, *A*);  *Danhilepski*(*A*, *M*, *M1*, *B*, *n*);  *solution*(1, *-A*[0][0], *-A*[0][1], *-A*[0][2], *lambda*);  *cout<<*"Danhilepski's solution is: Eigen value:"*<<endl*;  *cout* *<<* *setprecision*(4) *<<* *fixed* *<<* *lambda*[0] *<<* *endl* *<<* *lambda*[1] *<<* *endl* *<<* *lambda*[2] *<<* *endl*;  *double* *y*[3][10], *x*[3][10];  *for* (*int* *i* *=* 0; *i* *<* 3; *i++*) {  *for* (*int* *j* *=* 0; *j* *<* 10; *j++*) {  *y*[*j*][*i*] *=* *pow*(*lambda*[*i*], *n-j-*1);  }  }  *multiMatrix*(*B*, *y*, *x*, 3, 3, 3);  *cout<<*"S: "*<<endl*;  *for* (*int* *i* *=* 0; *i* *<* 3; *i++*) {  *for*(*int* *j* *=* 0; *j* *<* 3; *j++*){  *cout<<x*[*i*][*j*]*<<*"\t";  *if*(*j* *==* 2)  *cout<<endl*;  }  }  matrix *D*;  *for*(*int* *i* *=* 0; *i<*3; *i++*){  *for*(*int* *j* *=* 0; *j<*3; *j++*){  *if*(*i* *==* *j*)  *D*[*i*][*j*] *=* *lambda*[*i*];  *else*  *D*[*i*][*j*] *=* 0;  }  }  *cout<<*"Diagonalized matrix: "*<<endl*;  *for*(*int* *i* *=* 0; *i<*3; *i++*){  *for*(*int* *j* *=* 0; *j<*3; *j++*){  *cout<<D*[*i*][*j*]*<<*"\t";  *if*(*j* *==* 2)  *cout<<endl*;  }  }  matrix *inverse*;  *invertMatrix*(*x*, *inverse*, 3);  *cout<<*"S^-1: "*<<endl*;  *for*(*int* *i* *=* 0; *i<*3; *i++*){  *for*(*int* *j* *=* 0; *j<*3; *j++*){  *cout<<inverse*[*i*][*j*]*<<*"\t";  *if*(*j* *==* 2)  *cout<<endl*;  }  }  *return* 0;  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới với  (sai số ): |

***Câu 4*** (*3 điểm*): Cho ma trận A. Viết chương trình bằng c/c++ có sử dụng hàm thực hiện phân rã ma trận A bằng phương pháp SVD.

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):  *#include* <iostream>  *#include* <Eigen/Dense>  *#include* <iomanip>  *#include* <math.h>  *using* *namespace* std;  *using* *namespace* Eigen;  *void* *input*(*double* A[][10], *int* row, *int* col);  *void* *swap*(*double* *&*a, *double* *&*b);  *void* *display*(*double* A[][10], *int* row, *int* col);  *void* *chuyenvi*(*double* A[][10], *double* At[][10], *int* rows, *int* cols);  *void* *NhanMaTran*(MatrixXd *&*S, *double* A[][10], *double* B[][10], *int* row1, *int* col1, *int* col2);  *void* *GetEigenValuesAndVector*(MatrixXd S, MatrixXd *&*lambda, MatrixXd *&*vector);  *void* *tinhMatranS*(MatrixXd lambda, *double* sigma[][10], *int* rows, *int* cols);  *void* *tinhMatranU*(MatrixXd lambda, MatrixXd vector, *double* U[][10], *double* A[][10], *int* rows, *int* cols);  *void* *tinhMatranV*(MatrixXd vector, *double* V[][10]);  *void* *gramSchmidt*(*double* U[][10], *int* rows, *int* cols);  *int* *main*() {  *int* *rows*, *cols*;  *double* *A*[10][10], *At*[10][10];  *cout* *<<* "ENTER MATRIX " *<<* *endl*;  *cout* *<<* "ROWS: "; *cin* *>>* *rows*;  *cout* *<<* "COLUMNS: "; *cin* *>>* *cols*;  *cout* *<<* "ENTER MATRIX ELEMENTS: " *<<* *endl*;  *input*(*A*, *rows*, *cols*);  *cout* *<<* "MATRIX A" *<<* *endl*;  *display*(*A*, *rows*, *cols*);  MatrixXd *S*(*cols*, cols), *lambda*(*cols*, 1), *vector*(*cols*, cols);  *chuyenvi*(*A*, *At*, *rows*, *cols*);  *NhanMaTran*(*S*, *At*, *A*, *cols*, *rows*, *cols*);  *GetEigenValuesAndVector*(*S*, *lambda*, *vector*);  *double* *sigma*[10][10], *U*[10][10], *V*[10][10];  *tinhMatranU*(*lambda*, *vector*, *U*, *A*, *rows*, *cols*);  *cout* *<<* "MATRIX U" *<<* *endl*;  *display*(*U*, *rows*, *rows*);  *tinhMatranS*(*lambda*, *sigma*, *rows*, *cols*);  *cout* *<<* "MATRIX SIGMA" *<<* *endl*;  *display*(*sigma*, *rows*, *cols*);  *tinhMatranV*(*vector*, *V*);  *cout* *<<* "MATRIX VT" *<<* *endl*;  *double* *VT*[10][10];  *chuyenvi*(*V*, *VT*, *cols*, *cols*);  *display*(*VT*, *cols*, *cols*);  *return* 0;  }  *void* *input*(*double* A[][10], *int* row, *int* col) {  *for* (*int* *i* *=* 0; *i* *<* row; *i++*)  *for* (*int* *j* *=* 0; *j* *<* col; *j++*){  *cout* *<<* "a[" *<<* *i* *+* 1 *<<* "][" *<<* *j* *+* 1 *<<* "] = ";  *cin* *>>* A[*i*][*j*];  }  }  *void* *display*(*double* A[][10], *int* row, *int* col) {  *for* (*int* *i* *=* 0; *i* *<* row; *i++*)  {  *for* (*int* *j* *=* 0; *j* *<* col; *j++*)  *cout* *<<* *setw*(9) *<<* *fixed* *<<* *setprecision*(3) *<<* A[*i*][*j*];  *cout* *<<* *endl*;  }  }  *void* *swap*(*double* *&*a, *double* *&*b) {  *double* *temp* *=* a;  a *=* b;  b *=* *temp*;  }  *void* *chuyenvi*(*double* A[][10], *double* At[][10], *int* rows, *int* cols) {  *for* (*int* *i=*0; *i<*rows; *i++*)  *for* (*int* *j=*0; *j<*cols; *j++*) {  At[*j*][*i*] *=* A[*i*][*j*];  }  }  *void* *NhanMaTran*(MatrixXd *&*S, *double* A[][10], *double* B[][10], *int* row1, *int* col1, *int* col2) {  *for* (*int* i*=*0; i*<*row1; i*++*) {  *for* (*int* j*=*0; j*<*col2; j*++*) {  *S*(i,j) *=* 0;  *for* (*int* k*=*0; k*<*col1; k*++*) {  *S*(i,j) *=* *S*(i,j) *+* *A*[i][k] *\** *B*[k][j];  }  }  }  }  *void* *GetEigenValuesAndVector*(MatrixXd S, MatrixXd *&*lambda, MatrixXd *&*vector) {  SelfAdjointEigenSolver*<*Eigen::MatrixXd*>* *eigensolver*(S);  vector *=* *eigensolver*.*eigenvectors*();  lambda *=* *eigensolver*.*eigenvalues*();  *int* k *=* *lambda*.*rows*();  *int* l *=* *vector*.*rows*();  *for* (*int* i*=*0; i*<*k; i*++*)  {  *if* (*lambda*(i, 0) *<* 0.000001)  *lambda*(i, 0) *=* 0;  }  *for* (*int* i *=* 0; i *<* k; i*++*)  *for* (*int* j *=* i *+* 1; j *<* k; j*++*)  {  *if* (*lambda*(j, 0) *>* *lambda*(i, 0))  {  *swap*(*lambda*(j, 0), *lambda*(i, 0));  *for* (*int* h *=* 0; h *<* l; h*++*)  *swap*(*vector*(h, i), *vector*(h, j));  }  }  }  *void* *tinhMatranU*(MatrixXd lambda, MatrixXd vector, *double* U[][10], *double* A[][10], *int* rows, *int* cols) {  MatrixXd *ui*(rows, 1);  *double* *Vi*[cols][10];  *for* (*int* i *=* 0; i *<* cols; i*++*) {  *for* (*int* j *=* 0; j *<* cols; j*++*) {  *Vi*[j][0] *=* *vector*(j, i);  }  *NhanMaTran*(ui, A, Vi, rows, cols, 1);  *for* (*int* k *=* 0; k *<* rows; k*++*) {  *if* (*lambda*(i, 0) *!=* 0) {  *U*[k][i] *=* (1 */* *sqrt*(*lambda*(i, 0))) *\** *ui*(k, 0);  } *else* {  *U*[k][i] *=* 0;  }  }  }  *if*(rows *>* cols) {  *if* (rows *>* cols) {  MatrixXd *U\_matrix*(rows, cols);  *for* (*int* i *=* 0; i *<* rows; i*++*) {  *for* (*int* j *=* 0; j *<* cols; j*++*) {  *U\_matrix*(i, j) *=* *U*[i][j];  }  }  MatrixXd orthogonal\_basis *=* *U\_matrix*.*householderQr*().*householderQ*();  *for* (*int* j *=* cols; j *<* rows; j*++*) {  *for* (*int* i *=* 0; i *<* rows; i*++*) {  *U*[i][j] *=* *orthogonal\_basis*(i, j);  }  }  }  } *else* {  *gramSchmidt*(U, rows, cols);  }  }  *void* *gramSchmidt*(*double* U[][10], *int* rows, *int* cols) {  *for* (*int* i *=* 0; i *<* cols; i*++*) {  *for* (*int* j *=* 0; j *<* i; j*++*) {  *double* dot\_product *=* 0;  *for* (*int* k *=* 0; k *<* rows; k*++*) {  dot\_product *+=* *U*[k][i] *\** *U*[k][j];  }  *for* (*int* k *=* 0; k *<* rows; k*++*) {  *U*[k][i] *-=* dot\_product *\** *U*[k][j];  }  }  *double* norm *=* 0;  *for* (*int* k *=* 0; k *<* rows; k*++*) {  norm *+=* *U*[k][i] *\** *U*[k][i];  }  norm *=* *sqrt*(norm);  *if* (norm *>* 1*e-*10) {  *for* (*int* k *=* 0; k *<* rows; k*++*) {  *U*[k][i] */=* norm;  }  } *else* {  MatrixXd random\_vector *=* MatrixXd::*Random*(rows, 1);  *for* (*int* j *=* 0; j *<* i; j*++*) {  *double* dot\_product *=* 0;  *for* (*int* k *=* 0; k *<* rows; k*++*) {  dot\_product *+=* *random\_vector*(k, 0) *\** *U*[k][j];  }  *for* (*int* k *=* 0; k *<* rows; k*++*) {  *random\_vector*(k, 0) *-=* dot\_product *\** *U*[k][j];  }  }  *double* random\_norm *=* *random\_vector*.*norm*();  *for* (*int* k *=* 0; k *<* rows; k*++*) {  *U*[k][i] *=* *random\_vector*(k, 0) */* random\_norm;  }  }  }  }  *void* *tinhMatranV*(MatrixXd vector, *double* V[][10]) {  *int* row *=* *vector*.*rows*(), col *=* *vector*.*cols*();  *for* (*int* i *=* 0; i *<* row; i*++*)  {  *for* (*int* j *=* 0; j *<* col; j*++*)  {  *V*[i][j] *=* *vector*(i, j);  }  }  }  *void* *tinhMatranS*(MatrixXd lambda, *double* sigma[][10], *int* rows, *int* cols){  *int* k *=* 0;  *for* (*int* i *=* 0; i *<* rows; i*++*)  *for* (*int* j *=* 0; j *<* cols; j*++*)  {  *sigma*[i][j] *=* (i *!=* j) *?* 0 *:* *sqrt*(*lambda*(k, 0));  *if* (i *==* j)  k*++*;  }  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới với  (sai số ):    Các cột vector trong ma trận U giải tìm nghiệm đảm bảo trực giao  0.3841x07194 + -0.7682x0.5967 + 0.5121x0.3555 ~ 0  0.3841x0.5787 + -0.7682x-0.2319 + 0.5121x-0.7819 ~ 0  Các cột còn lại cũng tương tự, và có kết quả gần 0. |