«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет електроніки

Кафедра мікроелектроніки

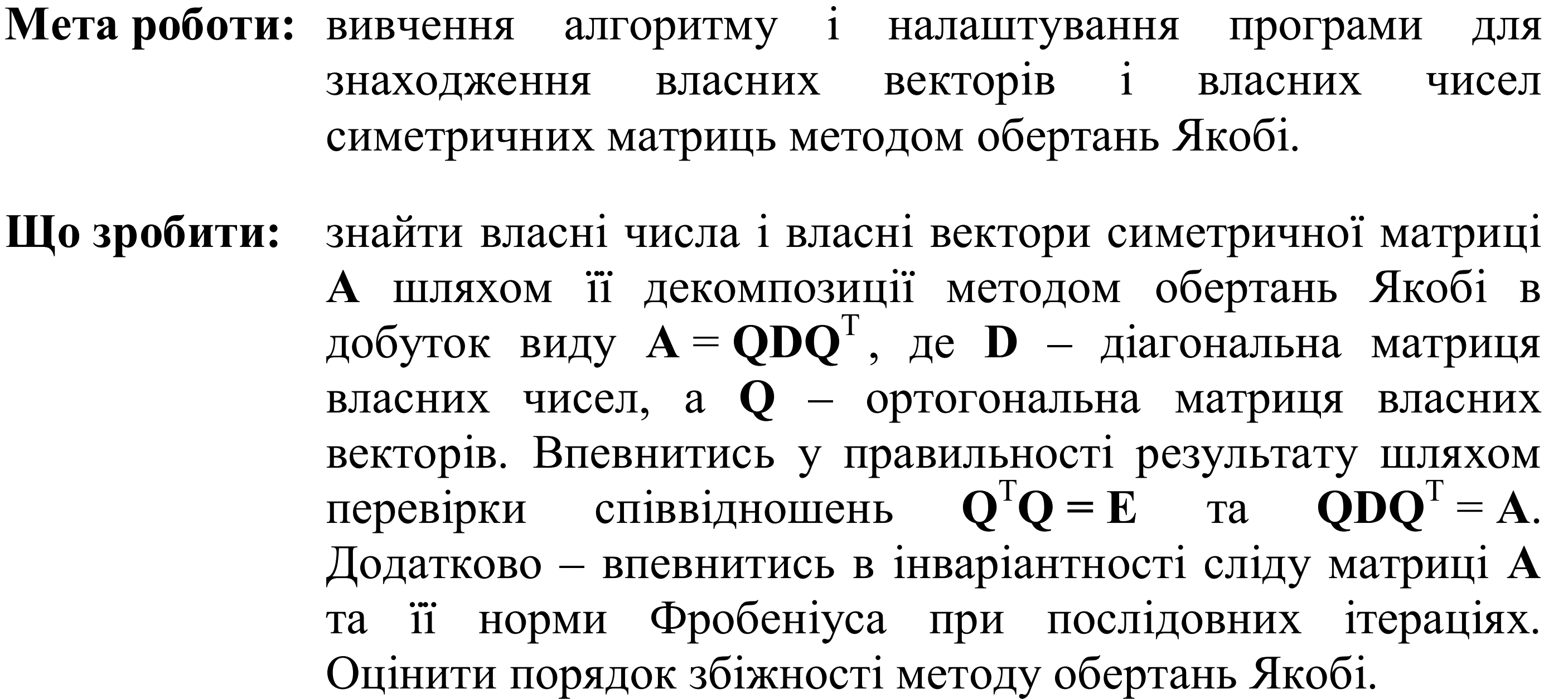
Лабораторна робота №11

Варіант №21

Виконав: студент групи ДП-82

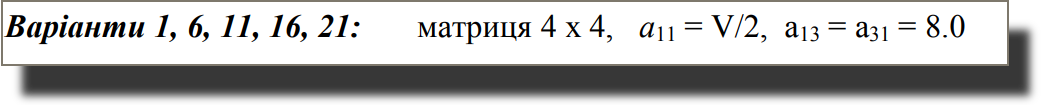
Мнацаканов Антон

Перевірив: Домбругов М.Р.



Київ-2020

Хід роботи



Фрагмент коду на С:

#include<stdio.h>

#include <stdlib.h>

#include <math.h>

#define EPS 1e-8

int read\_n(FILE\* f){

int nn;

fscanf(f,"%d", &nn);

return nn;

}

double\*\* init\_matrix(int n){

double \*\* res;

res = malloc(n\*sizeof(double\*));

if(!res){

return NULL;

}

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

res[i] = malloc(n\*sizeof(double));

if(!res[i]){

return NULL;

}

}

return res;

}

double\*\* init\_one\_matrix(int n){

double\*\* res = init\_matrix(n);

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

{

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

{

if(i==j)

{

res[i][j]=1;

}

else {

res[i][j]=0;

}

}

}

return res;

}

double\*\* read\_symm\_matrix(FILE\* f, int n){

double \*\* res = init\_matrix(n);

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

{

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

{

fscanf(f,"%lf", &res[i][j]);

}

}

return res;

}

double\*\* transpon\_matrix(int n, double\*\* matrix){

double\*\* res = init\_matrix(n);

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

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

res[j][i]=matrix[i][j];

}

}

return res;

}

double\*\* mul\_matrix(int n, double\*\* m1, double\*\* m2){

double\*\* res = init\_matrix(n);

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

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

res[i][j]=0;

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

res[i][j]+=m1[i][k]\*m2[k][j];

}

}

}

return res;

}

int print\_matrix(int i, double\*\* m);

int jacobi(int n, double\*\* d, double\*\* q, double eps)

{

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

{

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

{

if(i==j)

{

q[i][j]=1;

}

else {

q[i][j]=0;

}

}

}

do {

double dlm=d[1][0];

int l=1;

int m=0;

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

{

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

{

if(fabs(d[i][j]) > fabs(dlm))

{

dlm=d[i][j];

l=i;

m=j;

}

}

}

if (fabs(dlm)<eps) break;

double t=(d[l][l]-d[m][m])/(2.0f\*d[l][m]);

double u=t/sqrt(1.0f+t\*t) ;

double c=sqrt((1.0f+u)/2.0f);

double s=sqrt((1.0f-u)/2.0f);

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

{

double qil = q[i][l]\*c+q[i][m]\*s;

double qim = -q[i][l]\*s+q[i][m]\*c;

q[i][l]=qil;

q[i][m]=qim;

}

double dll=d[l][l]\*c\*c+2\*d[l][m]\*c\*s+d[m][m]\*s\*s;

double dmm=d[l][l]\*s\*s-2\*d[l][m]\*c\*s+d[m][m]\*c\*c;

d[l][l]=dll;

d[m][m]=dmm;

d[l][m]=0;

d[m][l]=0;

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

{

if(i!=l && i!=m)

{

double dil= d[i][l]\*c+d[i][m]\*s;

double dim=-d[i][l]\*s+d[i][m]\*c;

d[i][l]=dil;

d[l][i]=d[i][l];

d[i][m]=dim;

d[m][i]=d[i][m];

}

}

}while(1);

return 0;

}

int print\_matrix(int n, double\*\* m)

{

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

{

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

{

printf(" %e\t ", m[i][j]);

}

printf(" \n ");

}

printf(" \n ");

return 0;

}

int compare\_matrix(int n, double\*\* m1, double\*\* m2){

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

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

if(fabs(m1[i][j]-m2[i][j])>EPS)// сравнение с точностью

{

printf("%d %d %e %e\n",i,j, m1[i][j], m2[i][j]);

return 0;

}

}

}

return 1;

}

double\*\* copy\_matrix(int n, double\*\* m){

double\*\* res=init\_matrix(n);

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

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

res[i][j]=m[i][j];

}

}

return res;

}

int main(int argc, char\* argv[]){

if(argc != 2){

printf("Usage:\n\t %s <file with matrix>\n", argv[0]);

return 1;

}

FILE \*f = fopen(argv[1], "r");

if(!f){

printf("error: can't open file %s\n", argv[1]);

return 2;

}

int n = read\_n(f);

double\*\* a = read\_symm\_matrix(f, n);

double\*\* d = copy\_matrix(n,a);

double\*\* q = init\_matrix(n);

print\_matrix(n, d);

jacobi(n, d, q, EPS);

print\_matrix(n, q);

print\_matrix(n, d);

double\*\* qt = transpon\_matrix(n, q);

print\_matrix(n, qt);

double\*\* qqt = mul\_matrix(n, q, qt);

print\_matrix(n, qqt);

double\*\* e=init\_one\_matrix(n);

printf("Qt\*Q %s E\n", compare\_matrix(n, qqt, e)?"==":"!=");

double\*\* qd = mul\_matrix(n, q, d);

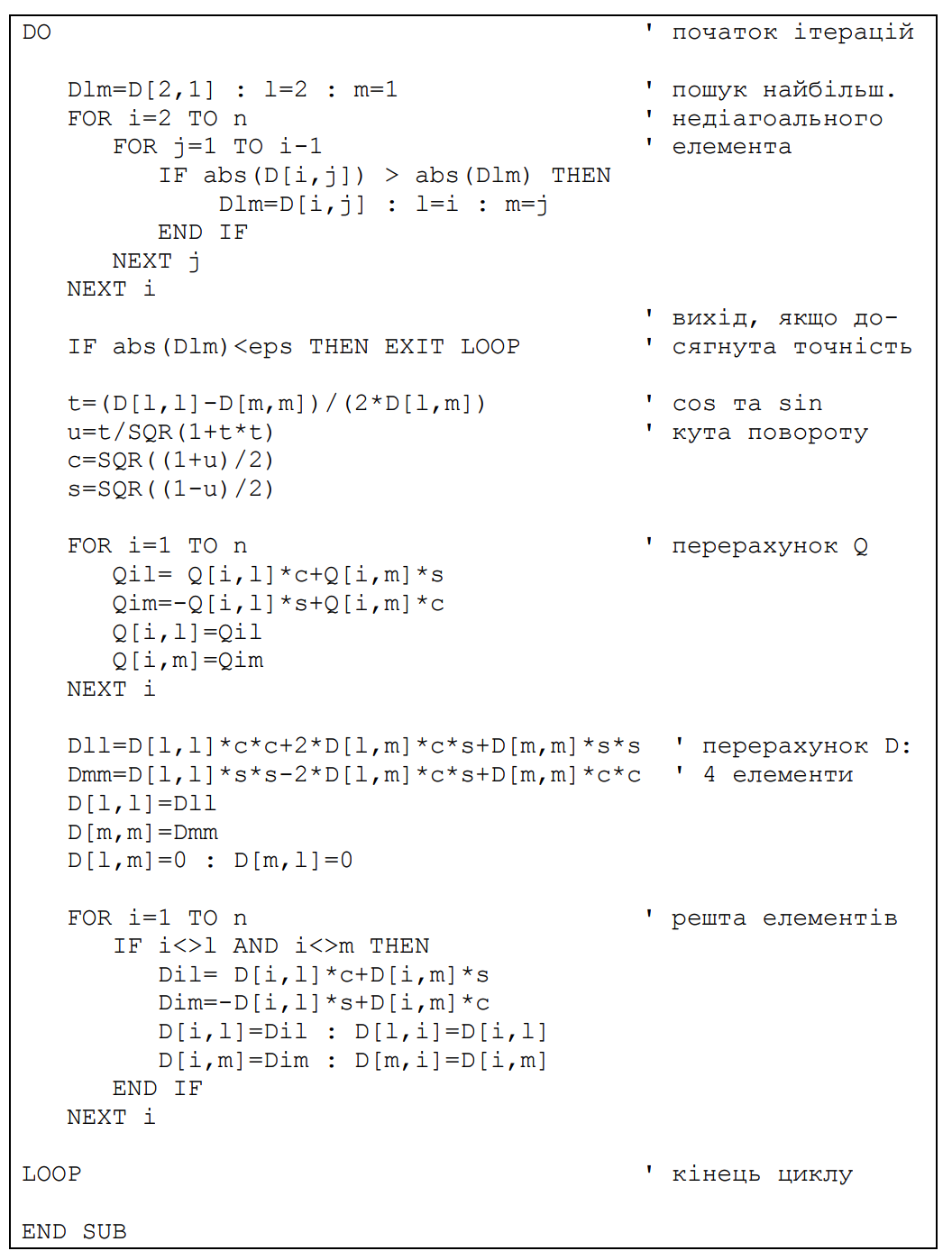
double\*\* qdqt = mul\_matrix(n, qd, qt);

print\_matrix(n, qdqt);

printf("Q\*D\*Qt %s A\n", compare\_matrix(n, qdqt, a)?"==":"!=");

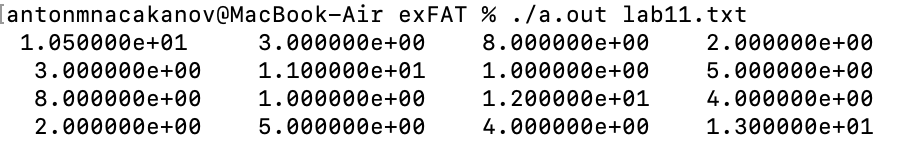
return 0;

}

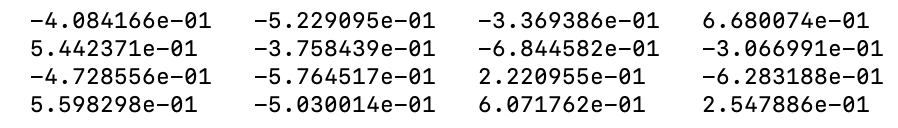
Основний код був перероблений з цього на простий С.

**Результати**

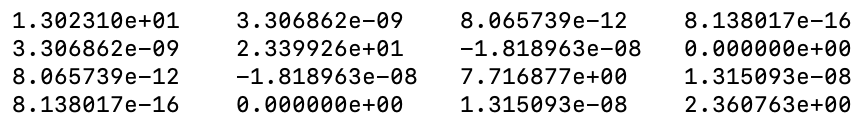
Ось це наша початкова матриця

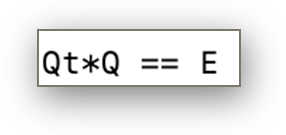


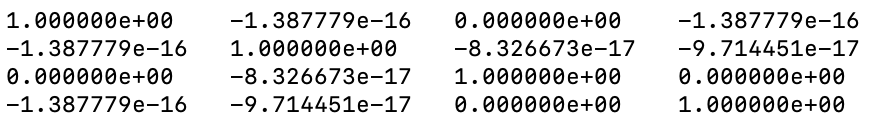
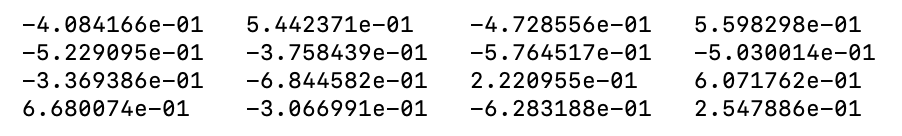
Ось це ми вивели ортогональну матрицю її власних векторів Q

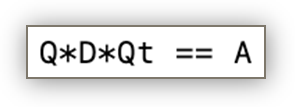


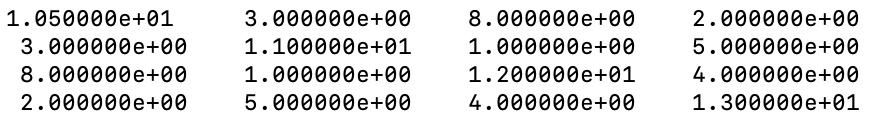
А це діагональна матриця власних чисел D



Це транспонована матриця







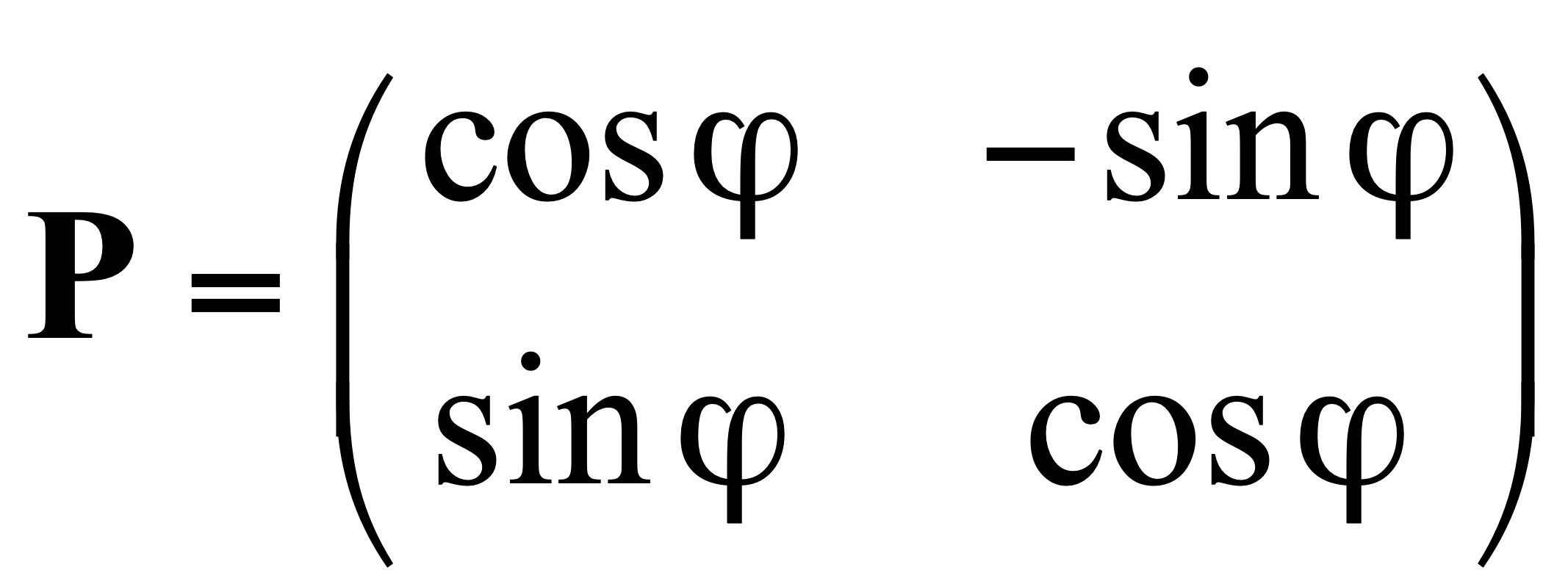
**Висновок:** в даній лабораторній роботі я написав программу знаходження власних чисел будь-якої симетричної матриці взявши за основу метод Якобі, також вивів проміжні результати в формі транспонованої матриці а також результати цих двох обчислень.



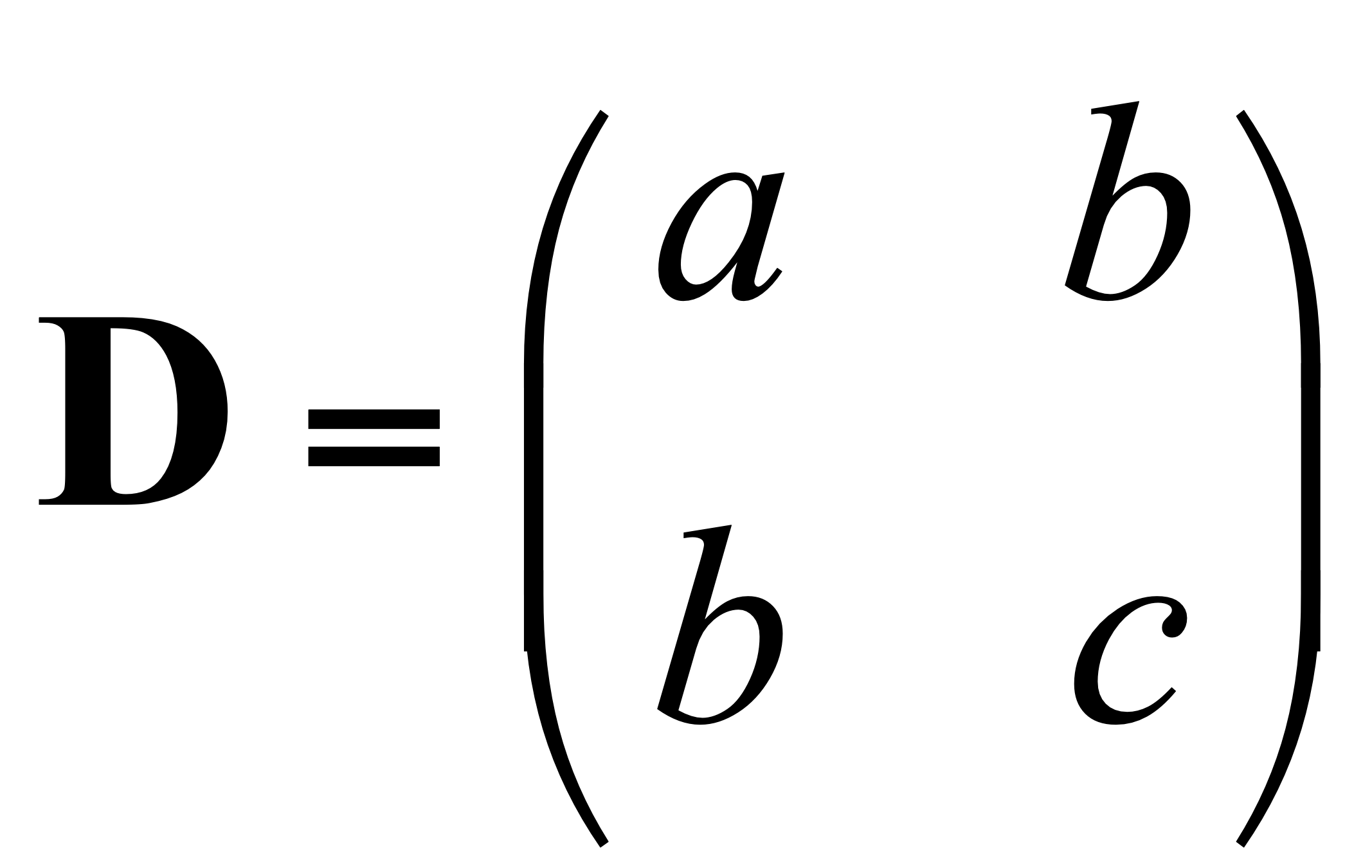
Прекрасно.

**Контрольні запитання до захисту:**

**Нехай є ортогональна матриця P розміром 2**´**2**

****

**та симетрична матриця D**

****

**Обрахуйте матрицю D' = PTDP та доведіть, що D' та D мають однакові**

1. **слід**
2. **норму Фробеніуса**
3. **слід**

|  |  |
| --- | --- |
| a\*cos^2(x)+c\*sin^2(x)+2\*b\*sin(x)\*cos(x) | sin(x)\*cos(x)(c-a)-b |
| sin(x)\*cos(x)(c-a)-b | a\*sin^2(x)+c\*cos^2(x)-2\*b\*sin(x)\*cos(x) |

Типо матрица:

**a\*cos^2(x)+c\*sin^2(x)+~~2\*b\*sin(x)\*cos(x)~~+a\*sin^2(x)+c\*cos^2(x)-~~2\*b\*sin(x)\*cos(x)~~ =a\* (cos^2(x)+sin^2(x)) +c\*(sin^2(x)+cos^2(x))**

**=>a+c - вроде так**

1. **А норму Фробеніуса мне не удалось доказать там эти косинусы и синусы в слишком больших степенях выходят и ещё и под корнем (чот сильно сложно :D ) и я запутался в конце**