Ministerul Educaţiei al Republicii Moldova Universitatea Tehnică a Moldovei Facultatea Calculatoare Informatica si Microelectronica Catedra Tehnologia Informatiei

**RAPORT**

Lucrarea de laborator nr.2

Disciplina: *Metode numerice*

Varianta 14

|  |  |  |
| --- | --- | --- |
| A efectuat: |  |  |
| st. gr. TI-195 | Rotaru Dan |  |
| A verificat: | Godonoaga Anatol |  |
|  |  |

Chisinau 2020

**Tema:** Rezolvarea numerica a sistemelor de ecuatii liniare

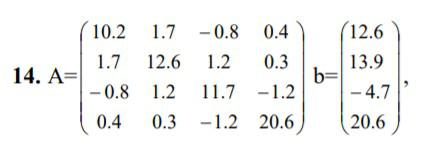
**Scopul lucrarii** :

1. Sa se rezolve sistemul de ecuatii liniare Ax = b, utilizind: Metoda lui Cholesky

Metoda iterativa a lui Jacobi cu o eroare ϵ = 10-3

Metoda iterativa a lui Gauss-Seidel cu o eroare ϵ = 10-3 si ϵ = 10-5

1. Sa se determine numarul de iteratii necesare pentru aproximarea solutiei sistemului cu eroarea data ϵ. Sa se compare rezultatele.



**Notiuni teoretice:**

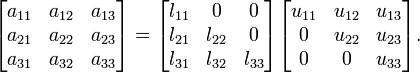
Metoda lui Cholesky. Metoda lui Cholesky de rezolvare a sistemelor de ecuaţii liniare algebrice se mai numeşte metoda rădăcinii pătratice şi constă în descompunerea sistemului Ax=b în două sisteme triunghiulare.

In aceasta metoda se presupune ca matricea A este o matrice simetrica si pozitiv definite.

Matricea L se alege astfel, incit A=LTL. Aceasta descompunere a matricei A se numeste factorizarea Choleskly.

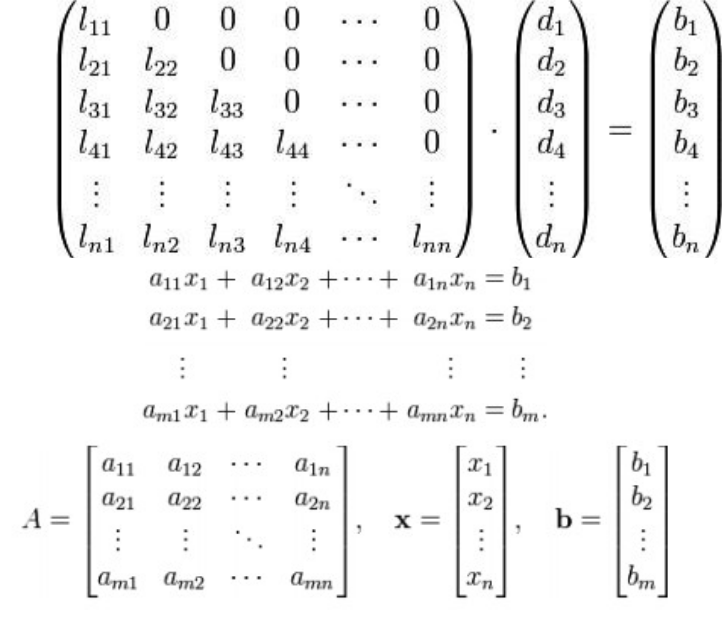
Teorema: Daca matricea A este simetrica si pozitiva definite, atunci exista o unica matrice inferior triunghiulara LT cu elementele diagonale pozitive, astfel incit A=LT .

Factorizarea LU presupune descompunerea matricei sistemului (A) intr-un produs de doua matrice L\*U. Matricea L trebuie sa fie **inferior triunghiulară**,in timp ce matricea U trebuie sa fie **superior triunghiulară**.



*Forma factorizarii LU pentru un exemplu 3x3*

Din această forma nu se pot calcula coeficienții matricelor L şi U. Pentru ca acest lucru sa fie posibil se adaugă câteva constrângeri,astfel rezultând factorizarea Cholesky. Daca sistemul **A\*x=b** devine **L\*U\*x=b**, putem nota cu **d=U\*x**. Astfel:



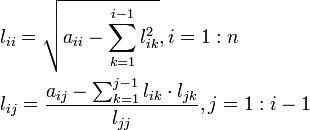
Pentru a **rezolva** sistemul de ecuații se determină mai întâi d din **L\*d=b** prin **rezolvarea unui sistem inferior triunghiular**. Ulterior se **rezolvă** şi sistemul **d=U\*x** ca un **sistem superior triunghiular.**

Dacă matricea sistemului este simetrica şi pozitiv definita, A se poate descompune astfel:

**A=L\*L’** (L’ fiind matricea L transpusa)

L este o matrice **inferior triunghiulară**.

Pentru determinarea elementelor matricei L se folosesc relațiile:



Metoda iteraţiei şi metoda Gauss-Seidel. Metoda iteraţiei reprezintă o extindere a metodei aproximaţiilor succesive folosite în cazul ecuaţiilor de o singură necunoscută şi metoda Jacobi pentru rezolvarea sistemelor de ecuaţii liniare. Acestă metodă permite rezolvarea sistemelor de ecuaţii neliniare de forma:

Plecând de la sistemului A\*x=b, descompunem A=N-P, unde N este o matrice ușor de inversat. În continuare:



Se deduce **relația de recurenţă**:



Se notează:

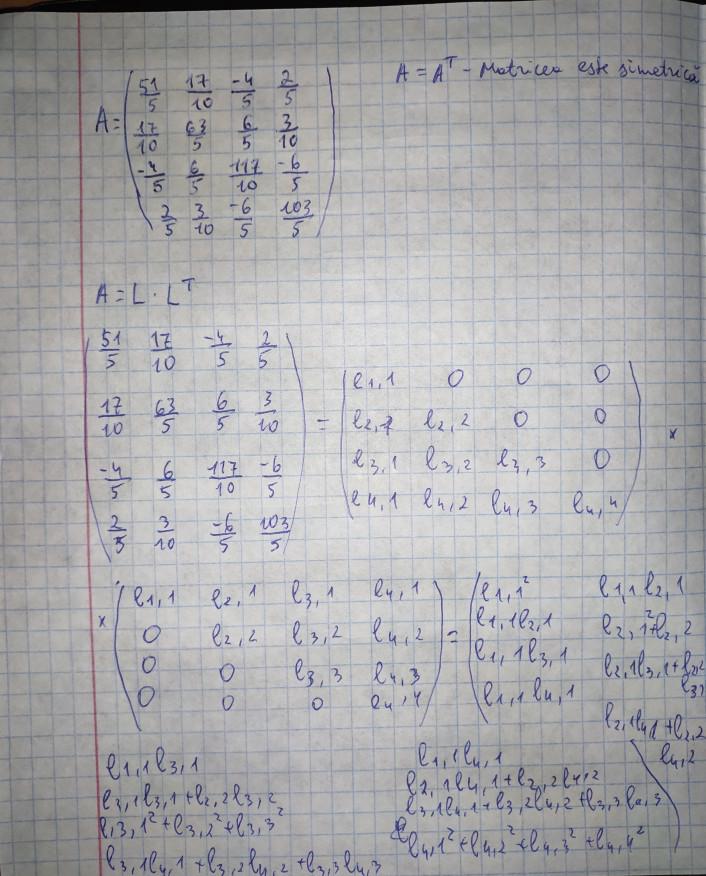


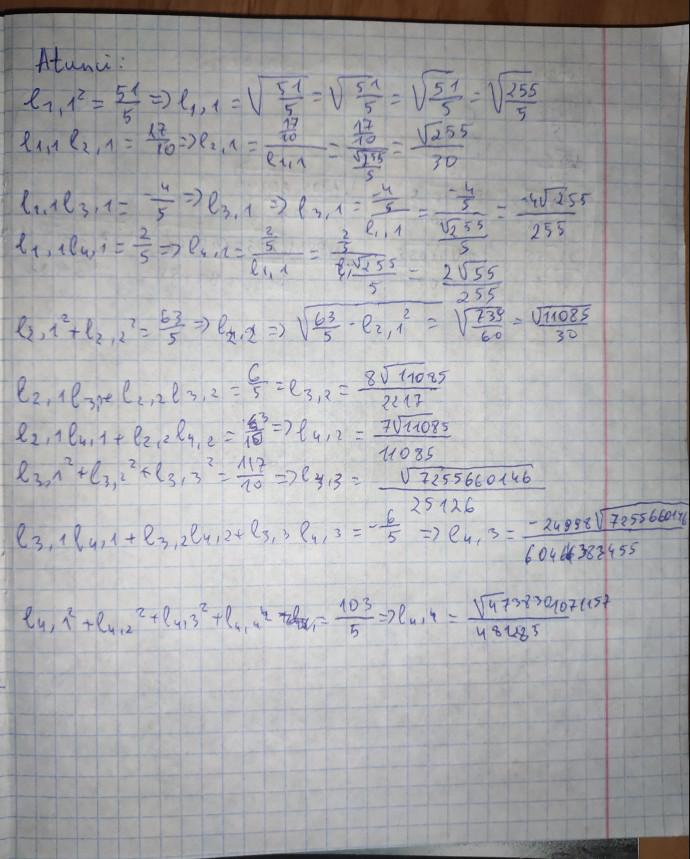
Matricea A se partiţionează ca **A=D-L-U**, unde:

*D=o matrice diagonală*

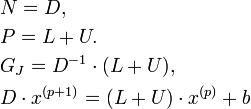
*L=o matrice strict inferior triunghiulară*

*U=o matrice superior triunghiulară*





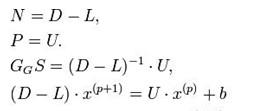
**Metoda Jacobi**



Componenta i din vectorul x la pasul (iteraţia) p+1 se calculează:



**Metoda Gauss-Seidel**



Componenta i din vectorul x la pasul (iteraţia) p+1 se calculează:



**Codul programului**



#include <iostream>

#include <cmath>

#include <conio.h>

using namespace std;

*// Functiile necesare*

float deter(float *a*[10][10], int *k*);

int Allo(int *lung*);

int simetric(int *lung*);

void factor(int *lung*);

void trans(int *lung*);

void determinare(int *lung*);

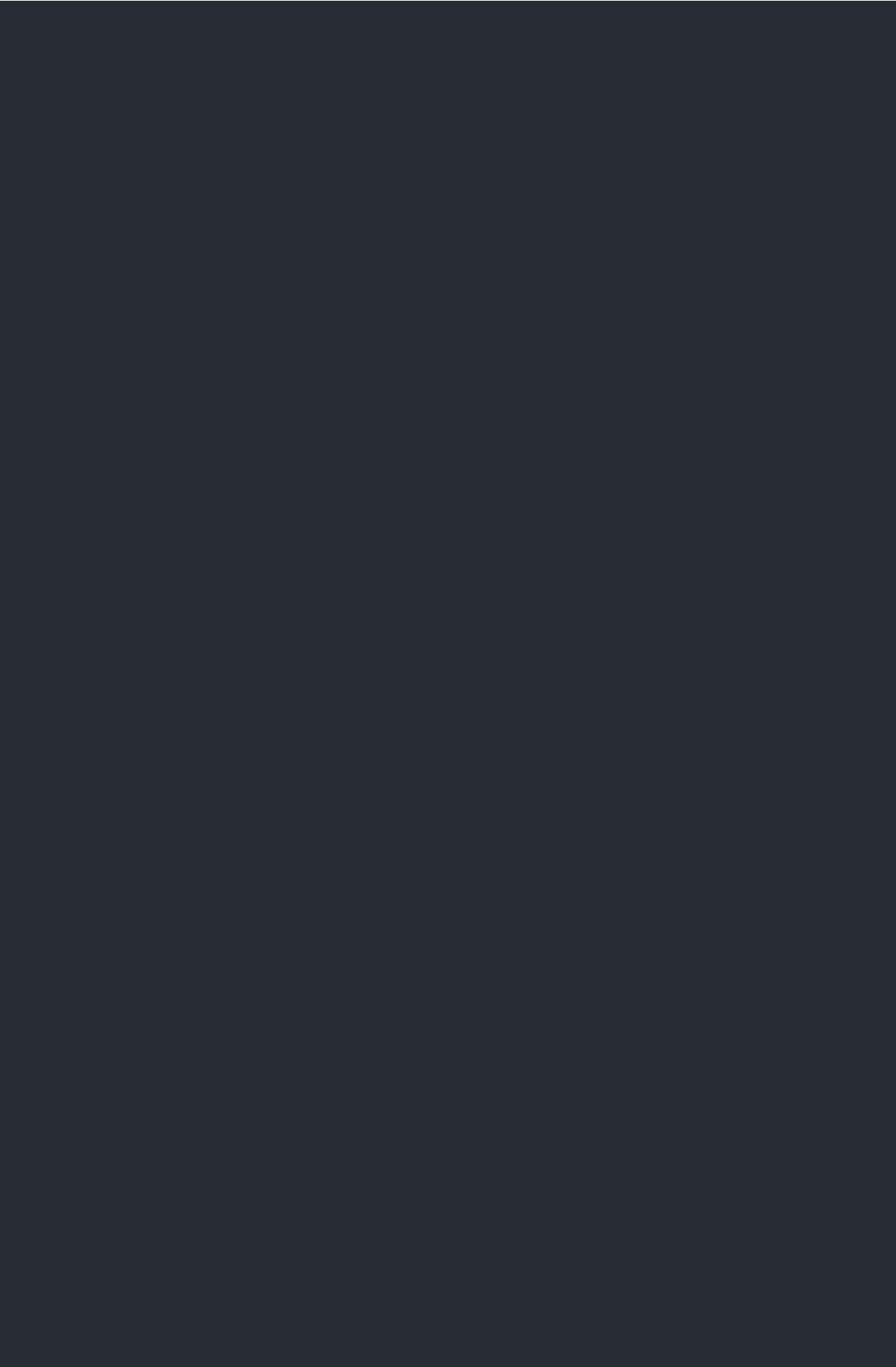
void Cholesky(int *lung*);

void Iacobi(int *lung*, float *Eps*);

void Gauss\_Seidel(int *lung*, float *Eps*);

int DiagDom(float *a*[10][10], int *lung*);

float l[10][10] = {0};



float a[10][10], b[10], c[10][10], d[10];

int lung;

int main(void) {

system("cls");

int lung, p, menu;

cout << "Selectati metoda: \n";

cout << " 1. Metoda lui Cholesky\n";

cout << " 2. Metoda Jacobi\n";

cout << " 3. Metoda Gauss-Seidel\n";

cout << " 0. Exit";

cout << endl << endl << " >>> ";

cin >> menu;

switch (menu) {

case 1:

system("cls");

cout << "Numarul liniilor: ";

cin >> lung;

cout << "\nIntroduceti tabloul A\n";

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

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

cin >> a[i][j];

c[i][j] = a[i][j];

}

cout << "\nTabloul b\n";

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

cin >> b[i];

d[i] = b[i];

}

cout << endl << endl << " Rezultatele";

cout << "\n Metoda Cholesky\n\n";

Cholesky(lung);

cout << endl << endl;

\_getch();

system("cls");

main();

exit(1);

break;

case 2:

system("cls");

cout << "Numarul liniilor: ";

cin >> lung;

cout << "\nIntroduceti tabloul A\n";

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

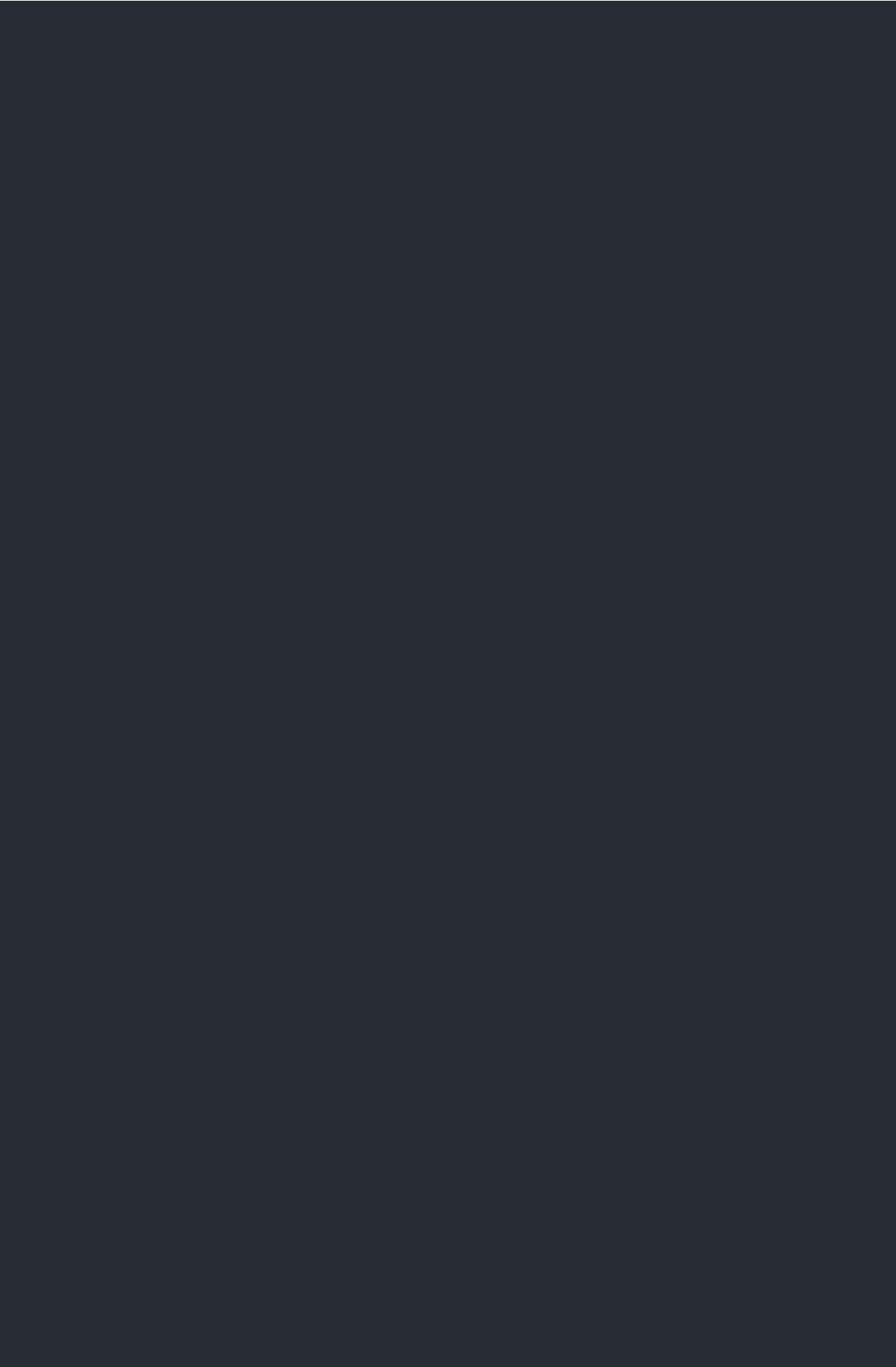
for (int j = 0; j < lung; j++) {

cin >> a[i][j];

c[i][j] = a[i][j];

}

cout << "\nTabloul b\n";



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

cin >> b[i];

d[i] = b[i];

}

cout << "\n Metoda Jacobi (Eroarea = 0.001)\n"; Iacobi(lung, 0.001);

cout << endl;

cout << "\n Metoda Jacobi (Eroarea = 0.00001)\n"; Iacobi(lung, 0.00001);

cout << endl;

\_getch();

system("cls");

main();

exit(1);

break;

case 3:

system("cls");

cout << "Numarul liniilor: ";

cin >> lung;

cout << "\nIntroduceti tabloul A\n";

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

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

cin >> a[i][j];

c[i][j] = a[i][j];

}

cout << "\nTabloul b\n";

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

cin >> b[i];

d[i] = b[i];

}

cout << "\n Metoda Gauss-Seidel (Eroarea = 0.001)\n"; Gauss\_Seidel(lung, 0.001);

cout << endl;

\_getch();

system("cls");

main();

exit(1);

break;

default:

cout << "Optiune gresita!";

break;

case 0:

return 0;

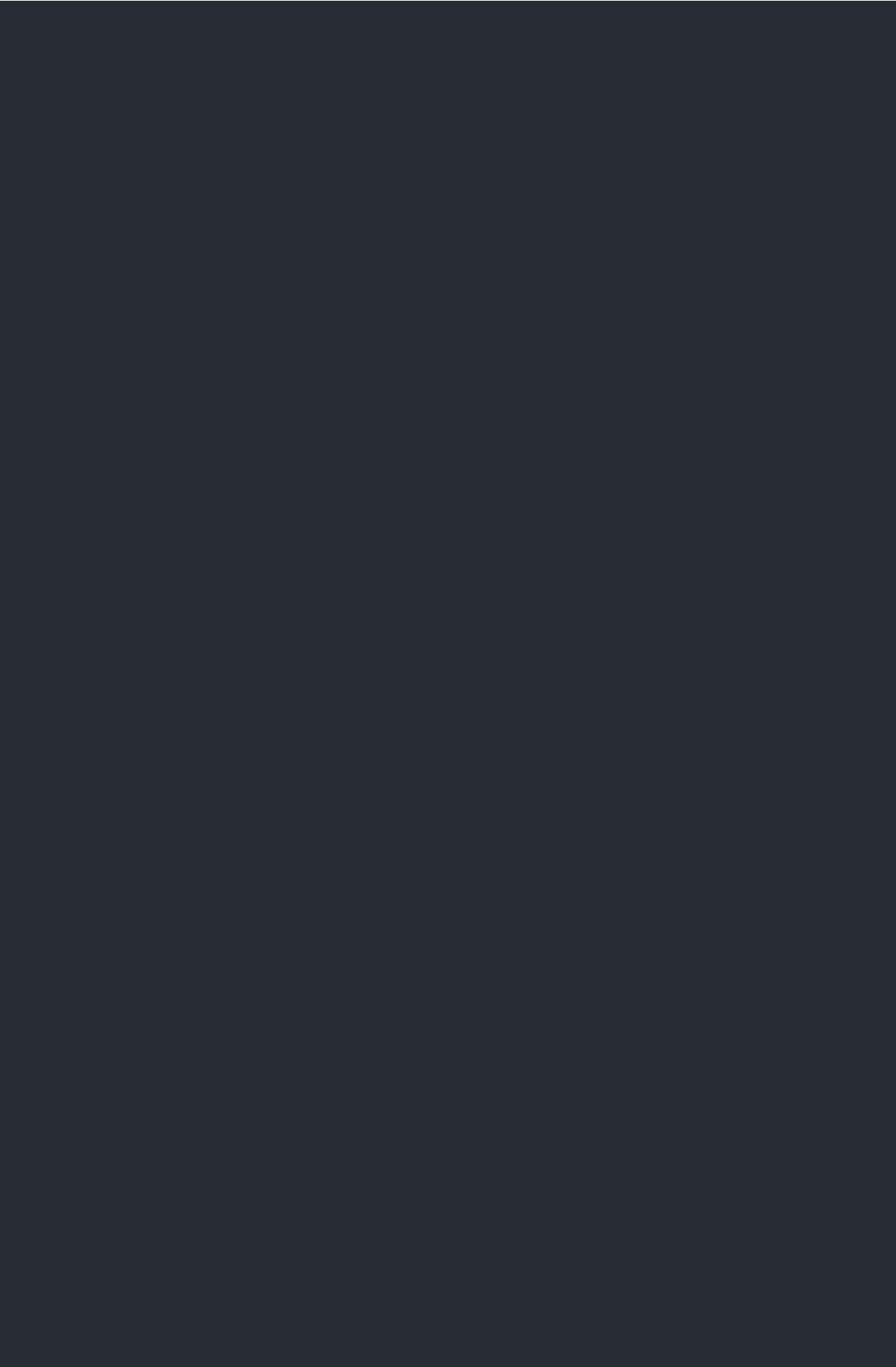
}

}

void Cholesky(int *lung*) {

if (!Allo(*lung*)) {

cout << "\nAceasta matricea nu este pozitiv definita!\n"; \_getch();



system("cls");

main();

exit(1);

goto E;

}

if (!simetric(*lung*)) {

cout << "\nAceasta matricea nu este simetrica !\n";

\_getch();

system("cls");

main();

exit(1);

}

determinare(*lung*);

\_getch();

E: ;

}

*//------------------------------------------*

float deter(float *a*[10][10], int *k*) {

if (*k* == 1) return *a*[0][0];

if (*k* == 2) return *a*[0][0] \* *a*[1][1] - *a*[0][1] \* *a*[1][0]; float s = 0;

float t[10][10] = {

0

};

for (int i = 0; i < *k*; i++) {

int x = 0;

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

if (l != i) {

for (int m = 1; m < *k*; m++)

t[x][m - 1] = *a*[l][m];\

x++;

}

}

if (i % 2 == 0) s += *a*[i][0] \* deter(t, *k* - 1); else s -= *a*[i][0] \* deter(t, *k* - 1);

}

return s;

}

*//------------------------------------------*

int Allo(int *lung*) {

for (int i = 0; i < *lung*; i++)

if (a[i][i] <= 0 || deter(a, i + 1) <= 0) return 0; return 1;

}

*//------------------------------------------*

int simetric(int *lung*) {

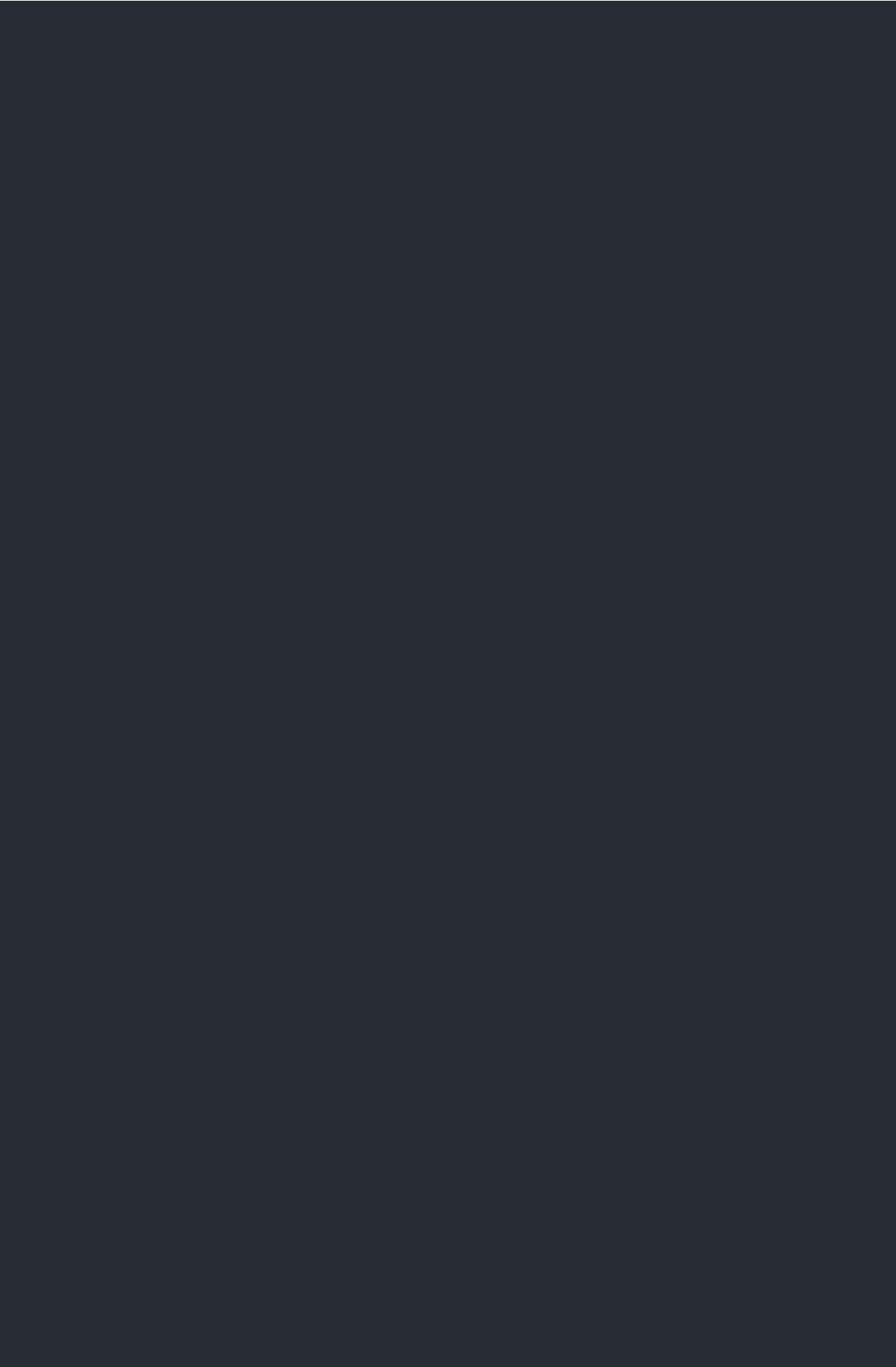
for (int i = 0; i < *lung*; i++)

for (int j = 0; j < *lung*; j++)

if (a[i][j] != a[j][i]) return 0;

return 1;

}



*//------------------------------------------*

void factor(int *lung*) {

l[0][0] = (float) sqrt(a[0][0]);

for (int i = 0; i < *lung*; i++) l[i][0] = a[i][0] / l[0][0]; for (int i = 1; i < *lung*; i++) {

float t = 0;

for (int j = 0; j < i; j++) t += l[i][j] \* l[i][j]; l[i][i] = (float) sqrt(a[i][i] - t);

for (int j = i; j < *lung*; j++) {

float t1 = 0;

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

t1 += l[j][k] \* l[i][k];

l[j][i] = (a[j][i] - t1) / l[i][i];

}

}

}

*//------------------------------------------*

void trans(int *lung*) {

float t = 0;

for (int i = 0; i < *lung*; i++)

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

* = l[i][j];

l[i][j] = l[j][i];

l[j][i] = t;

}

}

*//------------------------------------------*

void determinare(int *lung*) {

factor(*lung*);

float y[10], x[10];

y[0] = b[0] / l[0][0];

for (int i = 0; i < *lung*; i++) {

float t = 0;

for (int j = 0; j < i; j++) t += l[i][j] \* y[j]; y[i] = (b[i] - t) / l[i][i];

}

trans(*lung*);

cout << "Matricea triughiular superioara:\n"; for (int i = 0; i < *lung*; i++) {

for (int j = 0; j < *lung*; j++) {

cout << l[i][j] << "\t";

}

cout << endl;

}

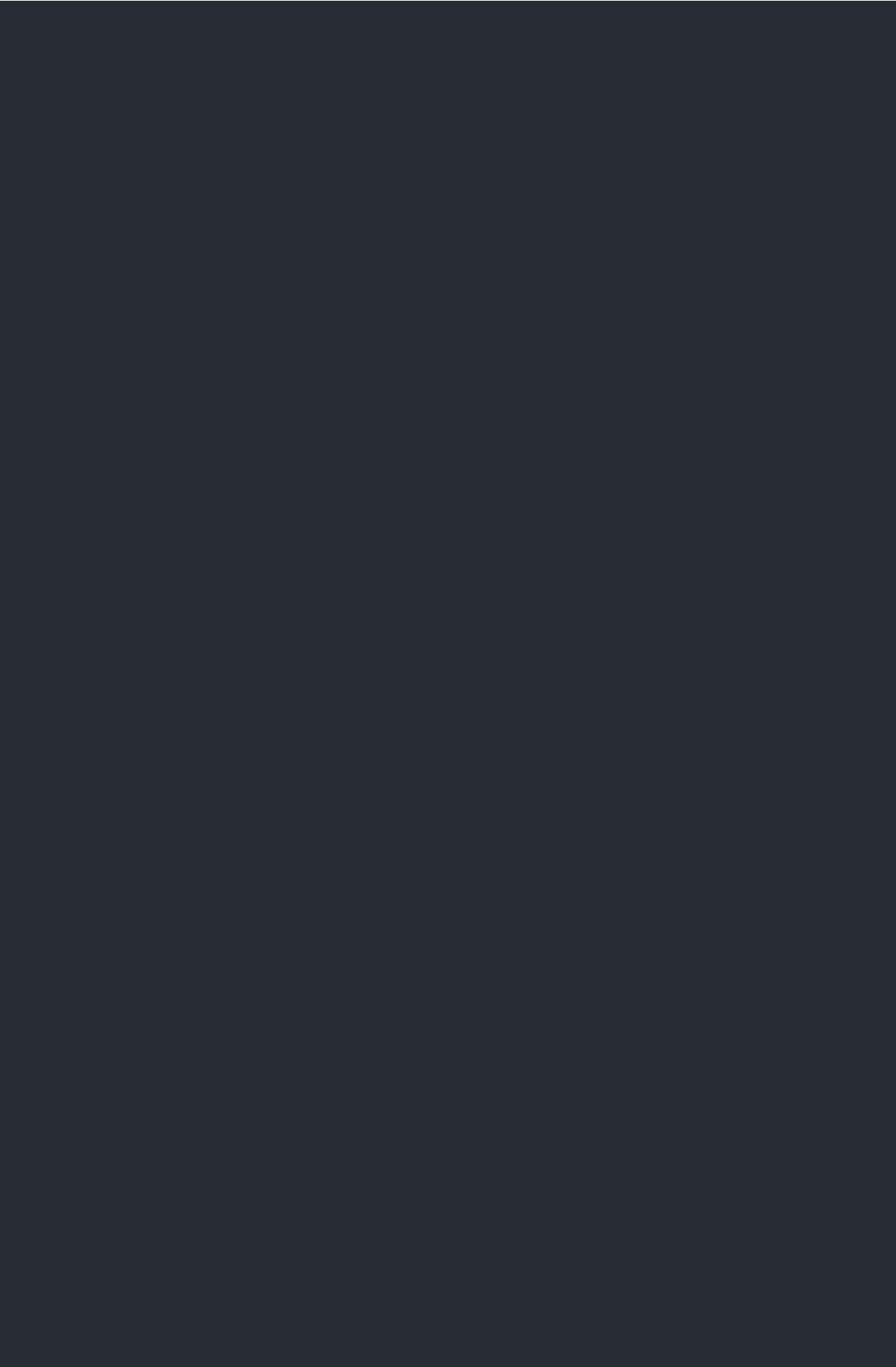
x[*lung* - 1] = y[*lung* - 1] / l[*lung* - 1][*lung* - 1]; float t;

for (int i = *lung* - 2; i >= 0; i--) {

* = 0;

for (int j = i + 1; j < *lung*; j++)

t += l[i][j] \* x[j];



x[i] = (y[i] - t) / l[i][i];

}

cout << "\nRezultatele x:\n";

for (int i = 0; i < *lung*; i++)

cout << " x" << i + 1 << " = " << x[i] << '\n';

cout << "\nRezultatele y:\n";

for (int i = 0; i < *lung*; i++)

cout << " y" << i + 1 << " = " << y[i] << "\n";

}

*//-------------------------------------*

void Iacobi(int *lung*, float *Eps*) {

float x[10], x1[10], q[10][10] = { 0}, d[10], t, max; int ni = 0;

for (int i = 0; i < *lung*; i++)

for (int j = 0; j < *lung*; j++)

if (i != j) q[i][j] = -(a[i][j] / a[i][i]); else q[i][j] = 0;

if (!DiagDom(a, *lung*)) {

cout << "Aceasta matricea nu este diagonal dominata\n";

\_getch();

system("cls");

main();

exit(1);

goto E;

}

for (int i = 0; i < *lung*; i++) d[i] = b[i] / a[i][i]; for (int i = 0; i < *lung*; i++) x[i] = d[i]; do {

for (int i = 0; i < *lung*; i++)

x1[i] = x[i];

for (int i = 0; i < *lung*; i++) {

* = 0;

for (int j = 0; j < *lung*; j++)

* += q[i][j] \* x1[j]; *// Jakobi* x[i] = t + d[i];

}

max = (float) fabs(x[0] - x1[0]);

for (int i = 1; i < *lung*; i++)

if ((float) fabs(x[i] - x1[i]) > max)

max = (float) fabs(x[i] - x1[i]);

ni++;

} while (max > *Eps*);

cout << "\nRezultatele x:\n";

for (int i = 0; i < *lung*; i++)

cout << "x" << i + 1 << " = " << x[i] << endl;

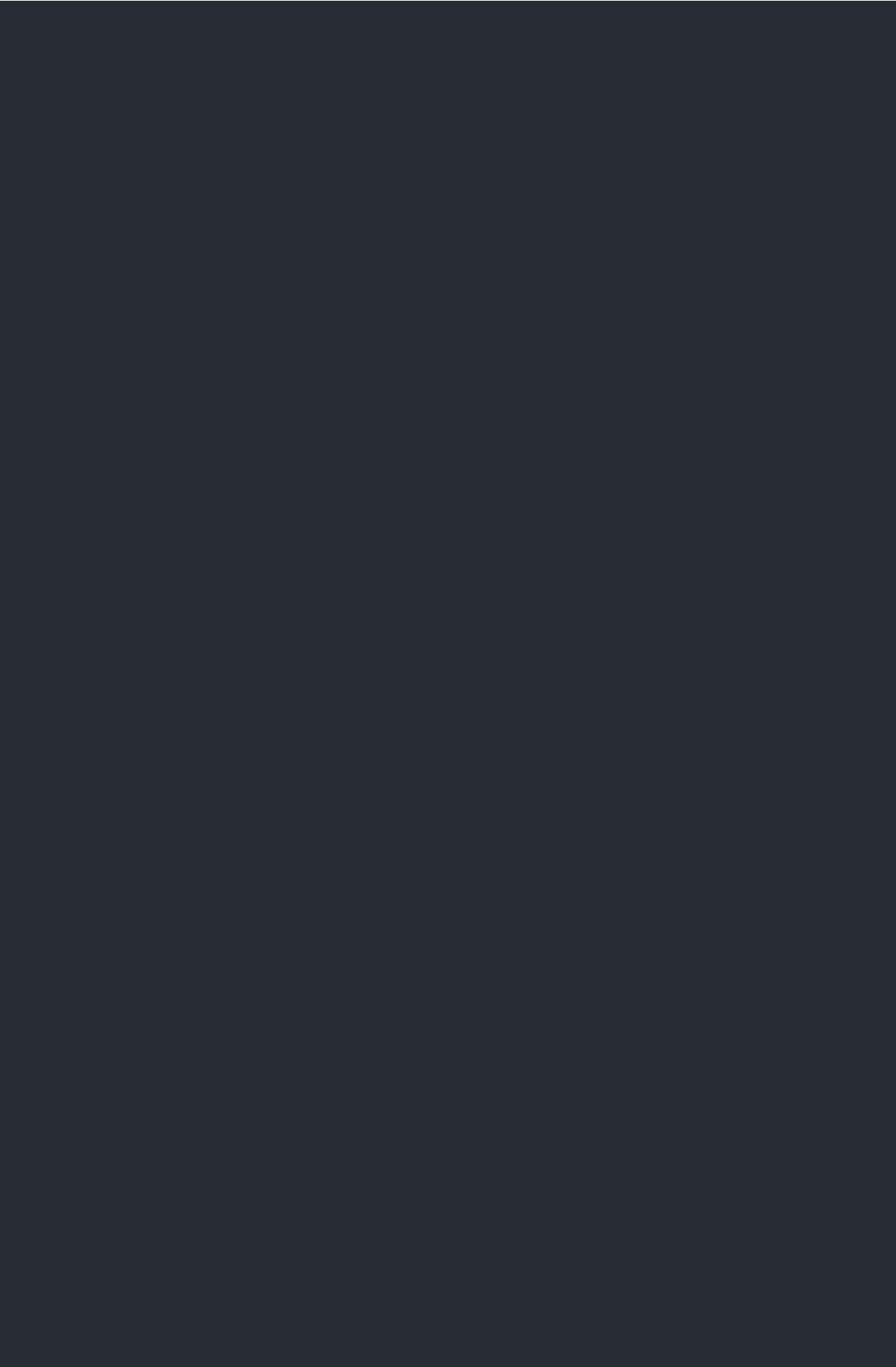
cout << "\nNumarul iteratiilor = " << ni << endl;

\_getch();

E: ;

}

*//-------------------------------------*



int DiagDom(float *a*[10][10], int *lung*) {

float s;

for (int i = 0; i < *lung*; i++) {

* = 0;

for (int j = 0; j < *lung*; j++)

if (i != j) s += *a*[i][j];

if (*a*[i][i] < s) return 0;

if (*a*[i][i] == 0) return 0;

}

return 1;

}

*//-----------------------------*

void Gauss\_Seidel(int *lung*, float *Eps*) {

float x[10], x1[10], q[10][10] = {0}, d[10], t, max; int ni = 0;

for (int i = 0; i < *lung*; i++)

for (int j = 0; j < *lung*; j++)

if (i != j) q[i][j] = -(a[i][j] / a[i][i]); else q[i][j] = 0;

if (!DiagDom(a, *lung*)) {

cout << "Aceasta matricea nu este diagonal dominata\n";

\_getch();

system("cls");

main();

exit(1);

goto E;

}

for (int i = 0; i < *lung*; i++)

d[i] = b[i] / a[i][i];

for (int i = 0; i < *lung*; i++)

x[i] = d[i];

do {

for (int i = 0; i < *lung*; i++)

x1[i] = x[i];

for (int i = 0; i < *lung*; i++) {

t = 0;

for (int j = 0; j < *lung*; j++)

t += q[i][j] \* x[j]; *// Gauss - Seidel* x[i] = t + d[i];

}

max = (float) fabs(x[0] - x1[0]);

for (int i = 1; i < *lung*; i++)

if ((float) fabs(x[i] - x1[i]) > max) max = (float) fabs(x[i] - x1[i])

;

ni++;

} while (max > *Eps*);

cout << "\nRezultatele x:\n";

for (int i = 0; i < *lung*; i++)

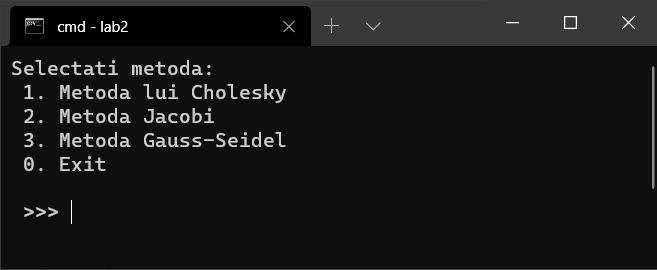
cout << "x" << i + 1 << " = " << x[i] << endl; cout << "\nNumarul iteratiilor = " << ni << endl; \_getch();

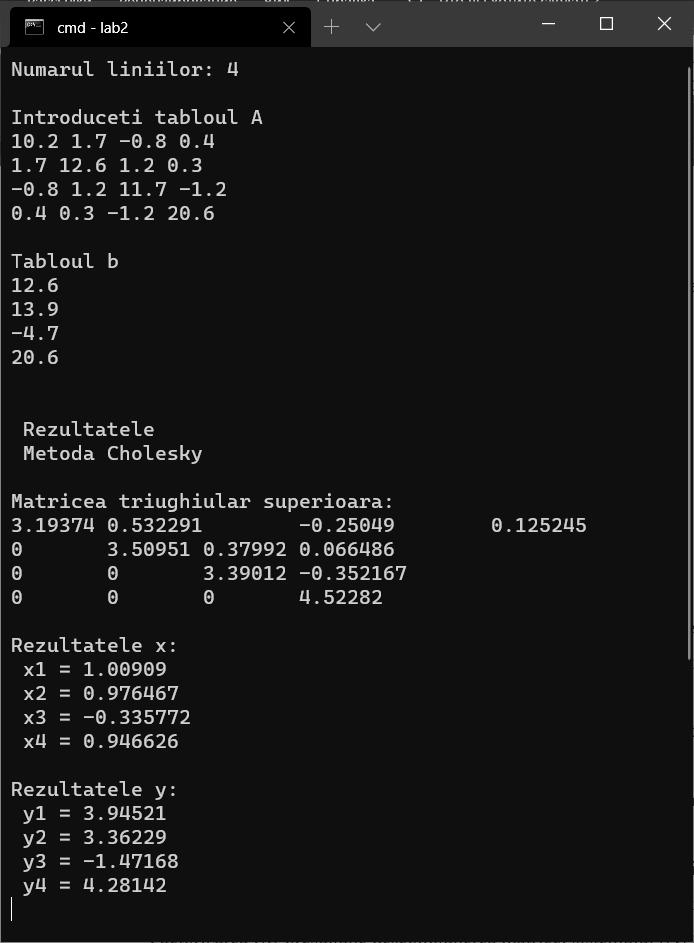


E: ;

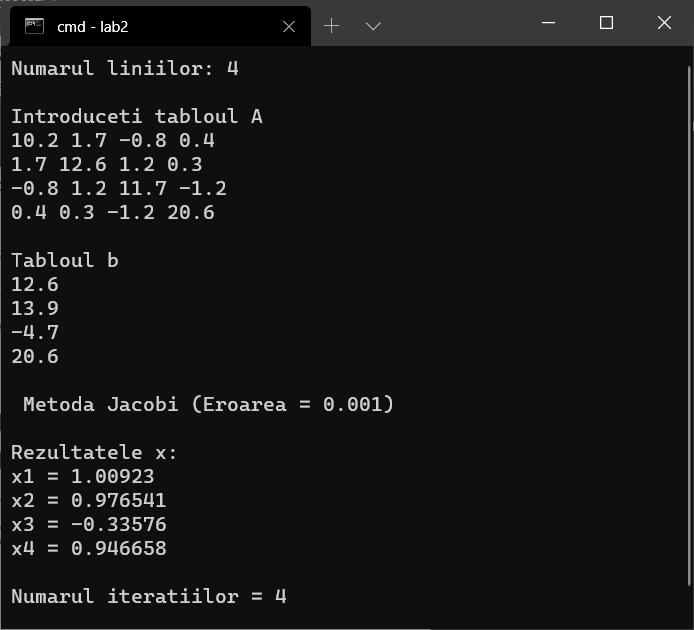
}

**Rezultatele obtinute**

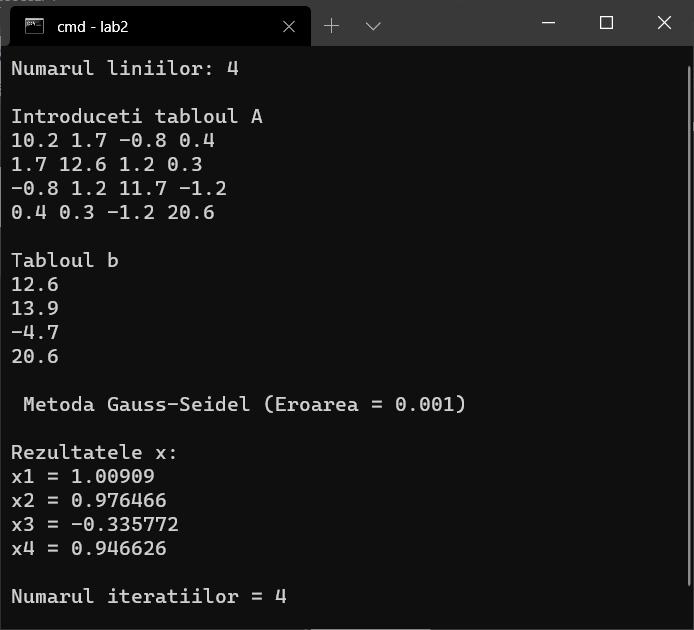




***Fig.1*** Metoda Cholesky



***Fig.2*** Metoda Jacob



***Fig.3*** Metoda Gauss – Seidel

**Compararea rezultatelor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Metoda** | **Radacina** |  | **Iteratii** | **Eroarea** |  |
|  | X | y |  |
|  |  |  |  |
|  | x1=1.00909 | y1=3.94521 |  |  |  |
| Cholesky | x2=0.976467 | y2=3.36229 | - | - |  |
|  | x3=-0.335772 | y3=-1.47168 |  |  |  |
|  | x4=0.946626 | y4=4.28142 |  |  |  |
|  | x1=1.00909 |  |  |  |  |
| Gauss -Seidel | x2=0.976466 |  | 4 | 0.001 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | x3=-0.335772 |  |  |  |  |
|  | x4=0.946626 |  |  |  |  |
|  | x1=1.00923 |  |  |  |  |
|  | x2=0.976541 |  | 4 | 0.001 |  |
|  | x3=-0.33576 |  |  |
|  |  |  |  |  |
| Jacobi | x4=0.946658 |  |  |  |  |
|  | x1=1.00909 |  | 7 | 0.00001 |  |
|  | x2=0.976466 |  |  |
|  |  |  |  |  |
|  | x3=-0.335772 |  |  |  |  |
|  | x4=0.946626 |  |  |  |  |

**Concluzii**

In aceasta lucrare de laborator am facut cunostinta cu trei metode de rezolvare numerica a sistemelor de ecuatii liniare: Metoda Cholesky, Metoda Jacobi si Metoda Gauss - Seidel .In urma rezultatelor obtinute, din exemplul dat, cea mai eficienta metoda iterativa de calculare a radacinelor sistemelor de ecuatii a fost metoda Gauss-Seidel.