**Ministerul Educației și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**RAPORT**

Lucrare de laborator nr.2

la cursul „Metode Numerice”

**Tema 1:**„ REZOLVAREA NUMERICĂ A SISTEMELOR DE ECUAŢII LINIARE”

Varianta 10

A efectuat : **st. gr. TI-214 Frunceac Nichita**

A verificat:  **asistent univ. Vadim Struna**

**Chișinău 2022**

**Scopul lucrării:**

1. Să se rezolve sistemul de ecuatii liniare Ax = b cu eroarea 10^-3, utilizind:

-metoda lui Cholesky

-metoda iterative lui Jacobi

-metoda iterative a lui Gauss – Seidel

2) Sa se determine numarul de iteratii necesare pentru aproximarea solutiei sistemului cu eroarea data.Sa se compare reziltatele.

**Metoda lui Cholesky** de rezolvare a sistemelor de ecuaţii liniare algebrice se mai numeşte metoda rădăcinii pătrate şi constă în descompunerea sistemului Ax=b în două sisiteme triunghiulare:

LTy=b, Lx=y. unde L e o matrice superior triunghiulară, iar LT matricea transpusă ei.

În această metodă se presupune că matricea A este o matrice simetrică şi pozitiv definită. Matricea L se alege astfel, încît A=LLT. Elementele lij ale matricei inferior triunghiulare L pot fi calculate în felul următor:

Se determină prima coloană a matricei L

L11=√α11 , li1= αi1/li1, i=2,3,…,n;

După ce s-au obţinut primele (k-1) coloane ale matricei L se calculează coloana k:

Lkk=√akk-∑l2kj ,

lik=(αik-∑lijlkj)/lkk , i=k+1,…n

**Metodelele Jacobi şi Gauss-Seidel**

# Metodele iterative se constuiesc utilizînd desfacerea matricei A definită prin *A=S-T.* Atunci sistemul Ax=b (1) e echivalent cu sistemul *Sx=Qx+d,* (2) sau *x=Qx+d,* (3)unde Q=S-1T, d=S-1b. Prin urmare putem construi şirul {x(k)}utilizînd relaţia recurentă:

*Sx(k+1)=Tx(k)+b, k=0,1,2...* (4) sau *x(k+1) =Qx(k)+d.* (5)

unde x(0) , ce aparţine Rn , e o aproximaţie iniţială a soluţiei xPentru a reduce sistemul (1) la o formă (3) sau (4), potrivită pentu iteraţie, desfacerea matricei A trebuie să satisfacă condiţiile:

Sistemul (5) are o soluţie unică x(k+1) şi se rezolvă uşor. De aceea matricea S se alege de o formă simplă şi este ireversabilă.

Ea poate fi diagonală sau triunghiulară.

1. b) Şirul {x(k) }k=1 converge către soluţia exactă x\* oricare ar fi x(0).

Avem: S-1=diag(1/α11,1/α22,…,1/αnn)

În acest caz sistemul (2) devine :xi=1/αii(bi-∑αij∙xj(k)), i=1,2,…,n.

Procesul iterativ (5) este definit prin:

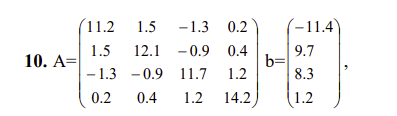
xi(k+1)= xi(k+1), i=1,2,…

Astfel obţinem o metodă de rezolvare a sistemului liniar (1) numită metoda lui Jacobi.

În **metoda lui Jacobi** e necesar de a păstra în memoria calculatorului toate componentele vectorului x(k) atît timp cît se calculează vectorul x(k+1). Putem modifica metoda lui Jacobi astfel încît la pasul (k+1) să folosim în calculul componentei xi(k+1), valorile deja calculate la aceasi pas: x1(k+1), x2(k+1),…, xi-1(k+1).

Aceasta modificare a metodei lui Jacobi se numeşte metoda lui **Gauss-Seidel**, iar şirul iterativ devine:

xi(k+1)= 1/αii(bi-∑αij∙xj(k)- ∑αij∙xj(k)), i=1,2,…,n.



**Codul**

#include <iostream>

#include <math.h>

//#include <conio.h>

using namespace std;

float deter(float a[4][4], int k);

int Allo(int lung);

int simetric(int lung);

void factor(int lung);

void trans(int lung);

void determinare(int n);

void Cholesky(int n);

void Iacobi(int n, float Eps);

void Gauss\_Seidel(int n, float Eps);

int DiagDom(float a[4][4], int n);

float l[4][4] = {0};

float a[4][4]=

{

        {11.2, 1.5, -1.3, 0.2},

        {1.5, 12.1, -0.9, 0.4},

        {-1.3, -0.9, 11.7, 1.2},

        {0.2, 0.4, 1.2, 14.2}

};

float b[4]={-11.4, 9.7, 8.3, 1.2}, c[4][4], d[4];

int n;

int main()

{

    int n=4, menu;

    cout << "\n|-------------------------|\n";

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

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

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

    cout << "| 0. Exit                 |";

    cout << "\n|-------------------------|\n";

    cout << " >>> ";

    cin >> menu;

    switch (menu)

    {

    case 1:

        for (int i = 0; i < n; i++)

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

            {

                c[i][j] = a[i][j];

            }

        cout << "\nTabloul b\n";

        for (int i = 0; i < n; i++)

        {

            d[i] = b[i];

        }

        cout << endl

             << endl

             << " Rezultatele";

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

        Cholesky(n);

        cout << endl

             << endl;

        main();

        exit(1);

        break;

    case 2:

        for (int i = 0; i < n; i++) // copiem matricea a in matricea c

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

            {

                c[i][j] = a[i][j];

            }

        cout << "\nTabloul b\n";

        for (int i = 0; i < n; i++) //copiem vectorul d in vectorul b

        {

            d[i] = b[i];

        }

        cout << "\n Metoda Jacobi (Eroarea = 10^-3)\n";

        Iacobi(n, pow(10,-3));

        cout << endl;

        main();

        exit(1);

        break;

    case 3:

        for (int i = 0; i < n; i++)

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

            {

                c[i][j] = a[i][j];

            }

        for (int i = 0; i < n; i++)

        {

            d[i] = b[i];

        }

        cout << "\n Metoda Gauss-Seidel (Eroarea = 10^-3)\n";

        Gauss\_Seidel(n, pow(10,-3));

        cout << endl;

        cout << "\n Metoda Gauss-Seidel (Eroarea = 10^-5)\n";

        Gauss\_Seidel(n, pow(10,-5));

        cout << endl;

        main();

        exit(1);

        break;

    default:

        cout << "Optiune gresita!";

        break;

    case 0:

        return 0;

    }

}

void Cholesky(int n)

{

    if (!Allo(n))

    {

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

        main();

        exit(1);

        goto E;

    }

    if (!simetric(n))

    {

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

        main();

        exit(1);

    }

    determinare(n);

E:;

}

//------------------------------------------

float deter(float a[4][4], 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[4][4] = {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;

}

//------------------------------------------

// Functia verifica daca matricea e pozitiv definita

int Allo(int n)

{

    for (int i = 0; i < n; i++)

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

            return 0;

    return 1;

}

//------------------------------------------

int simetric(int n)

{

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

            if (a[i][j] != a[j][i])

                return 0;

    return 1;

}

//------------------------------------------

void factor(int n)

{

    l[0][0] = (float)sqrt(a[0][0]);

    for (int i = 0; i < n; i++)

        l[i][0] = a[i][0] / l[0][0];

    for (int i = 1; i < n; 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 < n; 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 n)

{

    float t = 0;

    for (int i = 0; i < n; i++)

        for (int j = 0; j < i; j++)

        {

            t = l[i][j];

            l[i][j] = l[j][i];

            l[j][i] = t;

        }

}

//------------------------------------------

void determinare(int n)

{

    factor(n);

    float y[4], x[4];

    y[0] = b[0] / l[0][0];

    for (int i = 0; i < n; 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(n);

    cout << "Matricea triughiular inferioara:\n";

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < n; j++)

        {

            cout << l[j][i] << "\t";

        }

        cout << endl;

    }

    x[n - 1] = y[n - 1] / l[n - 1][n - 1];

    float t;

    for (int i = n - 2; i >= 0; i--)

    {

        t = 0;

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

            t += l[i][j] \* x[j];

        x[i] = (y[i] - t) / l[i][i];

    }

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

    for (int i = 0; i < n; i++)

        cout << " x" << i + 1 << " = " << x[i] << '\t'<< " y" << i + 1 << " = " << y[i] << "\n";

}

//-------------------------------------

void Iacobi(int n, float Eps)

{

    float x[4], x1[4], q[4][4] = {0}, d[4], t, max;

    int ni = 0;

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

            if (i != j)

                q[i][j] = -(a[i][j] / a[i][i]);

            else

                q[i][j] = 0;

    if (!DiagDom(a, n))

    {

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

        main();

        exit(1);

        goto E;

    }

    for (int i = 0; i < n; i++)

        d[i] = b[i] / a[i][i];

    for (int i = 0; i < n; i++)

        x[i] = d[i];

    do

    {

        for (int i = 0; i < n; i++)

            x1[i] = x[i];

        for (int i = 0; i < n; i++)

        {

            t = 0;

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

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

            x[i] = t + d[i];

        }

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

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

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

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

E:;

}

//-------------------------------------

int DiagDom(float a[4][4], int n)

{

    float s;

    for (int i = 0; i < n; i++)

    {

        s = 0;

        for (int j = 0; j < n; 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 n, float Eps)

{

    float x[4], x1[4], q[4][4] = {0}, d[4], t, max;

    int ni = 0;

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

            if (i != j)

                q[i][j] = -(a[i][j] / a[i][i]);

            else

                q[i][j] = 0;

    if (!DiagDom(a, n))

    {

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

        main();

        exit(1);

        goto E;

    }

    for (int i = 0; i < n; i++)

        d[i] = b[i] / a[i][i];

    for (int i = 0; i < n; i++)

        x[i] = d[i];

    do

    {

        for (int i = 0; i < n; i++)

            x1[i] = x[i];

        for (int i = 0; i < n; i++)

        {

            t = 0;

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

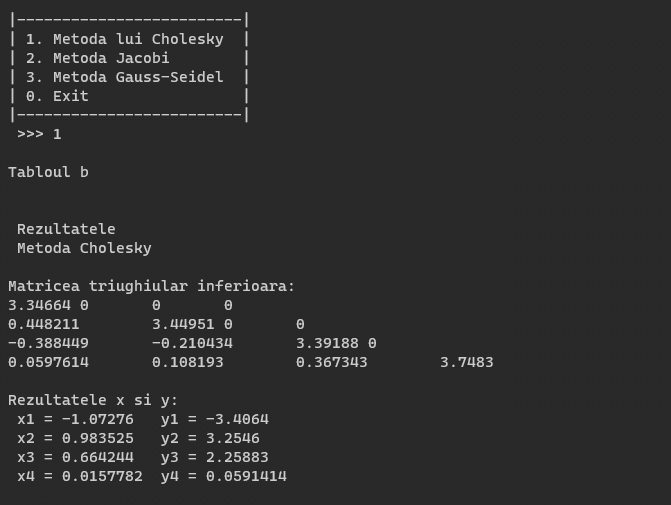
        cout << "x" << i + 1 << " = " << x[i] << endl;

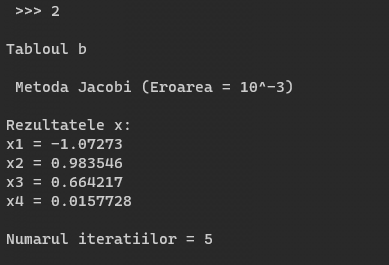
    cout << "\nNumarul iteratiilor = " << ni << endl;

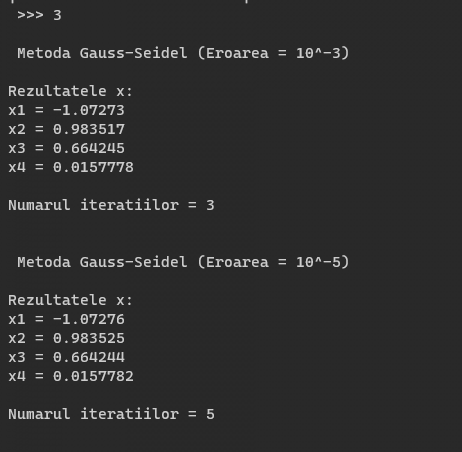
E:;

}

**Rularea codului**







**Concluzie**

In lucrarea data am studiat rezolvarea sistemelor de ecuatii lineare. In program am utilizat metodele **Cholesky, Jacobi, Gauss-Seidel**. Metoda **Cholesky** nu intotdeauna este exacta deoarece uneorie avem operatia radical. Metoda **Jacobi** necesita un numar de iteratii pentru a ajunge la solutia aproximativa, totodata este nevoie sa pastram vectorul trecut in memorie. Insa metoda **Gauss-Seidel** se dovedeste a fi mai rapida deoarece foloseste solutiile din pasul curent, dar nu din anterior, totodata nu este nevoie de pastrat in memorie vectori aditionali.Pentru epsilon mai mare metoda **Gauss-Seidel** a efectuat un numar mai mic de iteratii decit metoda **Jacobi** cu epsilon mai mic.

In lucrarea data am invatat mai multe metode de rezolvare a sistemelor de ecuatii (pe linga metodele stiute – metoda Cramer). Totodata am implementat aceste metode ca programe.