Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrare de laborator nr.2

la cursul „Metode Numerice”

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

Varianta 10

A efectuat: Popa Cătălin, TI-211

A verificat: Vladimir Patiuc

Chişinău 2022

**Scopul lucrării:**

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

- Metoda eliminării lui Gauss;

-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,…

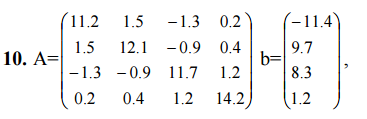
Astfel obţinem o metodă de rezolvare a sistemului liniar (1) numită metoda lui Jacobi.

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 C++**



#include <iostream>

#include <time.h>

#include <conio.h>

#include <math.h>

#define num 4

using namespace std;

long int iteratii1 = 0;

long int iteratii2 = 0;

long int iteratii3 = 0;

long int iteratii4 = 0;

float mat\_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 mat\_B[4] = {-11.4,

                  9.7,

                  8.3,

                  1.2};

void print\_matrix()

{

    cout << "\n Matricea A: \n";

    for (int i = 0; i < num; i++)

    {

        for (int j = 0; j < num; j++)

        {

            cout << "   " << mat\_A[i][j];

        }

        cout << "   \n";

    }

    cout << "\n Matricea B: \n";

    for (int i = 0; i < num; i++)

    {

        cout << " \n"

             << mat\_B[i];

    }

}

void Gauss()

{

    float t, array\_c[num], array\_a[num][num], array\_b[num];

    int i, j, l;

    for (i = 0; i < num; i++)

    {

        iteratii1 += 3;

        array\_b[i] = mat\_B[i];

        iteratii1++;

        for (j = 0; j < num; j++)

        {

            iteratii1 += 3;

            array\_a[i][j] = mat\_A[i][j];

            iteratii1++;

        }

    }

    l = 0;

    while (l < num)

    {

        iteratii1++;

        for (i = l + 1; i < num; i++)

        {

            iteratii1 += 4;

            t = array\_a[i][l] / array\_a[l][l];

            iteratii1++;

            for (j = l; j < num; j++)

            {

                iteratii1 += 3;

                array\_a[i][j] -= t \* array\_a[l][j];

                iteratii1++;

            }

            array\_b[i] -= t \* array\_b[l];

            iteratii1++;

        }

        l++;

    }

    for (i = num - 1; i >= 0; i--)

    {

        iteratii1 += 4;

        t = 0;

        iteratii1++;

        for (j = num - 1; j > i; j--)

        {

            iteratii1 += 4;

            t += array\_a[i][j] \* array\_c[j];

            iteratii1++;

        }

        t = array\_b[i] - t;

        array\_c[i] = t / array\_a[i][i];

        iteratii1 += 2;

    }

    for (i = 0; i < num; i++)

        printf("%7.3f ", array\_c[i]);

    cout << "\nNumarul de iteratii: " << iteratii1 << endl;

}

void Cholesky()

{

    float t, array\_x[num], array\_l[num][num], array\_y[num];

    int i, j, k;

    array\_l[0][0] = sqrt(mat\_A[0][0]);

    for (i = 1; i < num; i++)

    {

        array\_l[i][0] = mat\_A[i][0] / array\_l[0][0];

        iteratii3 += 5;

    }

    for (k = 1; k < num; k++)

    {

        iteratii3 += 3;

        t = 0;

        for (j = 0; j < k - 1; j++)

        {

            t += array\_l[k][j] \* array\_l[k][j];

            iteratii3 += 5;

        }

        array\_l[k][k] = sqrt(mat\_A[k][k] - t);

        iteratii3++;

        for (i = k + 1; i < num; i++)

        {

            t = 0;

            iteratii3 += 4;

            for (j = 0; j < k - 1; j++)

            {

                t += array\_l[i][j] \* array\_l[k][j];

                iteratii3 += 6;

            }

            array\_l[i][k] = (mat\_A[i][k] - t) / array\_l[k][k];

            iteratii3 += 3;

        }

    }

    for (i = 0; i < num; i++)

    {

        iteratii3 += 3;

        for (j = i + 1; j < num; j++)

        {

            array\_l[i][j] = array\_l[j][i];

            iteratii3 += 4;

        }

    }

    for (i = 0; i < num; i++)

    {

        t = 0;

        iteratii3++;

        for (j = 0; j < i; j++)

        {

            t += array\_l[i][j] \* array\_y[j];

            iteratii3 += 5;

        }

        t = mat\_B[i] - t;

        iteratii3++;

        array\_y[i] = t / array\_l[i][i];

        iteratii3 += 2;

    }

    for (i = num - 1; i >= 0; i--)

    {

        t = 0;

        iteratii3 += 5;

        for (j = num - 1; j > i; j--)

        {

            t += array\_l[i][j] \* array\_x[j];

            iteratii3 += 5;

        }

        t = array\_y[i] - t;

        array\_x[i] = t / array\_l[i][i];

        iteratii3 += 4;

    }

    for (i = 0; i < num; i++)

        printf("%7.3f ", array\_x[i]);

    cout << "\nNumarul de iteratii: " << iteratii3 << endl;

}

void Jacobi()

{

    float v, array\_x[num], array\_q[num][num], array\_d[num];

    float array\_s[num][num], array\_t[num][num], array\_x1[num], er;

    int i, j, m;

    for (i = 0; i < num; i++)

    {

        iteratii2 += 3;

        for (j = 0; j < num; j++)

        {

            iteratii2 += 3;

            if (i == j)

            {

                array\_s[i][j] = 1 / mat\_A[i][j];

                array\_t[i][j] = 0;

                iteratii2 += 4;

            }

            else

            {

                array\_s[i][j] = 0;

                array\_t[i][j] = mat\_A[i][j];

                iteratii2 += 2;

            }

        }

    }

    for (i = 0; i < num; i++)

    {

        iteratii2 += 3;

        for (j = 0; j < num; j++)

        {

            iteratii2 += 3;

            v = 0;

            for (m = 0; m < num; m++)

            {

                v += array\_s[i][m] \* array\_t[m][j];

                iteratii2 += 5;

            }

            array\_q[i][j] = -v;

            iteratii2++;

        }

    }

    for (i = 0; i < num; i++)

    {

        iteratii2 += 3;

        v = 0;

        iteratii2++;

        for (m = 0; m < num; m++)

        {

            v += array\_s[i][m] \* mat\_B[m];

            iteratii2 += 5;

        }

        array\_d[i] = v;

        iteratii2++;

    }

    for (i = 0; i < num; i++)

    {

        array\_x[i] = array\_d[i];

        iteratii2 += 4;

    }

    do

    {

        for (i = 0; i < num; i++)

        {

            array\_x1[i] = array\_x[i];

            iteratii2 += 4;

        }

        for (i = 0; i < num; i++)

        {

            iteratii2 += 3;

            v = 0;

            for (m = 0; m < num; m++)

            {

                v += array\_x1[m] \* array\_q[i][m];

                iteratii2 += 5;

            }

            array\_x[i] = v + array\_d[i];

            iteratii2++;

        }

        er = fabs(array\_x1[0] - array\_x[0]);

        iteratii2++;

        for (m = 0; m < num; m++)

        {

            iteratii2 += 3;

            if (er < fabs(array\_x1[m] - array\_x[m]))

            {

                er = fabs(array\_x1[m] - array\_x[m]);

                iteratii2 += 3;

            }

        }

    } while (er > 0.001);

    for (i = 0; i < num; i++)

        printf("%7.3f ", array\_x[i]);

    cout << "\nNumarul de iteratii: " << iteratii2 << endl;

}

void gaussseidel()

{

    int i, j, m, k1 = 0, k;

    float v, array\_x[num], array\_q[num][num], array\_d[num], array\_s[num][num];

    float array\_t[num][num], array\_x1[num], er;

    for (i = 0; i < num; i++)

    {

        iteratii4 += 3;

        for (j = 0; j < num; j++)

        {

            iteratii4 += 3;

            if (i == j)

            {

                array\_s[i][j] = 1 / mat\_A[i][j];

                array\_t[i][j] = 0;

                iteratii4 += 4;

            }

            else

            {

                array\_s[i][j] = 0;

                array\_t[i][j] = mat\_A[i][j];

                iteratii4 += 2;

            }

        }

    };

    for (i = 0; i < num; i++)

    {

        iteratii4 += 3;

        for (j = 0; j < num; j++)

        {

            iteratii4 += 3;

            v = 0;

            for (m = 0; m < num; m++)

            {

                v += array\_s[i][m] \* array\_t[m][j];

                iteratii4 += 5;

            }

            array\_q[i][j] = -v;

            iteratii4++;

        }

    }

    for (i = 0; i < num; i++)

    {

        v = 0;

        iteratii4 += 4;

        for (m = 0; m < num; m++)

        {

            v += array\_s[i][m] \* mat\_B[m];

            iteratii4 += 5;

        }

        array\_d[i] = v;

        iteratii4++;

    }

    for (i = 0; i < num; i++)

    {

        array\_x[i] = array\_d[i];

        iteratii4 += 4;

    }

    do

    {

        k1++;

        iteratii4++;

        for (i = 0; i < num; i++)

        {

            array\_x1[i] = array\_x[i];

            iteratii4 += 4;

        }

        for (i = 0; i < num; i++)

        {

            v = 0;

            iteratii4 += 4;

            for (m = 0; m < num; m++)

            {

                v += array\_x[m] \* array\_q[i][m];

                iteratii4 += 5;

            }

            array\_x[i] = v + array\_d[i];

            iteratii4 += 2;

        }

        er = fabs(array\_x1[0] - array\_x[0]);

        iteratii4++;

        for (m = 1; m < num; m++)

        {

            iteratii4 += 3;

            if (er < fabs(array\_x1[m] - array\_x[m]))

            {

                er = fabs(array\_x1[m] - array\_x[m]);

                iteratii4 + 4;

            }

        }

    } while (er > 0.001);

    for (i = 0; i < num; i++)

        printf("%7.3f ", array\_x[i]);

    printf("\n");

    cout << "\nNumarul de iteratii: " << iteratii4 << endl;

    k1 = 0;

    for (i = 0; i < num; i++)

        array\_x[i] = array\_d[i];

    do

    {

        k1++;

        for (i = 0; i < num; i++)

            array\_x1[i] = array\_x[i];

        for (i = 0; i < num; i++)

        {

            v = 0;

            for (m = 0; m < num; m++)

                v += array\_x[m] \* array\_q[i][m];

            array\_x[i] = v + array\_d[i];

        }

        er = fabs(array\_x1[0] - array\_x[0]);

        for (m = 1; m < num; m++)

            if (er < fabs(array\_x1[m] - array\_x[m]))

                er = fabs(array\_x1[m] - array\_x[m]);

    } while (er > 0.00001);

    cout << "\nMetoda prin Gauss-Seidel - eroare 0.00001\n" << endl;

    for (i = 0; i < num; i++)

        printf("%7.5f ", array\_x[i]);

    cout << endl;

}

void comparare()

{

    if (iteratii1 < iteratii2 && iteratii1 < iteratii3 && iteratii1 < iteratii4)

    {

        cout << "\n\n Cel mai eficient algoritm este dupa metoda Gauss!!!" << endl;

    }

    else if (iteratii2 < iteratii1 && iteratii2 < iteratii3 && iteratii2 < iteratii4)

    {

        cout << "\n\n Cel mai eficient algoritm este dupa metoda Jacobi!!!" << endl;

    }

    else if (iteratii3 < iteratii1 && iteratii3 < iteratii2 && iteratii3 < iteratii4)

    {

        cout << "\n\n Cel mai eficient algoritm este dupa metoda Cholesky!!!" << endl;

    }

    else if (iteratii4 < iteratii1 && iteratii4 < iteratii3 && iteratii4 < iteratii2)

    {

        cout << "\n\n Cel mai eficient algoritm este dupa metoda Gauss-Seidel!!!" << endl;

    }

    if (iteratii1 > iteratii2 && iteratii1 > iteratii3 && iteratii1 > iteratii4)

    {

        cout << "\n Cel mai neeficient algoritm este dupa metoda Gauss!!!\n" << endl;

    }

    else if (iteratii2 > iteratii1 && iteratii2 > iteratii3 && iteratii2 > iteratii4)

    {

        cout << "\n Cel mai neeficient algoritm este dupa metoda Jacobi!!!\n" << endl;

    }

    else if (iteratii3 > iteratii1 && iteratii3 > iteratii2 && iteratii3 > iteratii4)

    {

        cout << "\n Cel mai neeficient algoritm este dupa metoda Cholesky!!!\n" << endl;

    }

    else if (iteratii4 > iteratii1 && iteratii4 > iteratii3 && iteratii4 > iteratii2)

    {

        cout << "\n Cel mai neeficient algoritm este dupa metoda Gauss-Seidel!!!\n" << endl;

    }

}

int main()

{

    cout << "\n         Laborator nr2 MN         \n";

    cout << "\n           Popa Catalin           \n";

    cout << "\n              TI-211              \n";

    cout << "\nMatricele initiale       \n";

    print\_matrix();

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

    cout << "\n";

    cout << "\nMetoda prin Gauss!\n" << endl;

    Gauss();

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

    cout << "\n";

    cout << "\nMetoda prin Cholesky!\n" << endl;

    Cholesky();

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

    cout << "\n";

    cout << "\nMetoda prin Jacobi!\n" << endl;

    Jacobi();

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

    cout << "\n";

    cout << "\nMetoda prin Gauss-Seidel - eroare 0.001\n" << endl;

    gaussseidel();

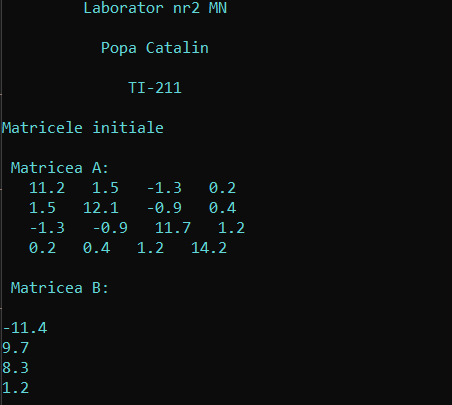
    cout << "\n----------------------------------------------\n";

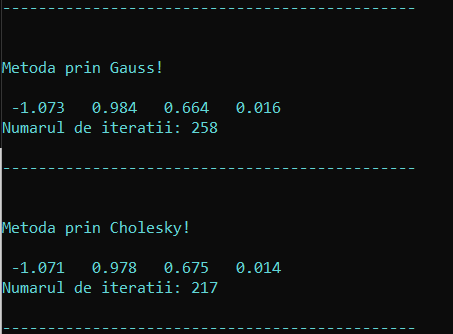
    comparare();

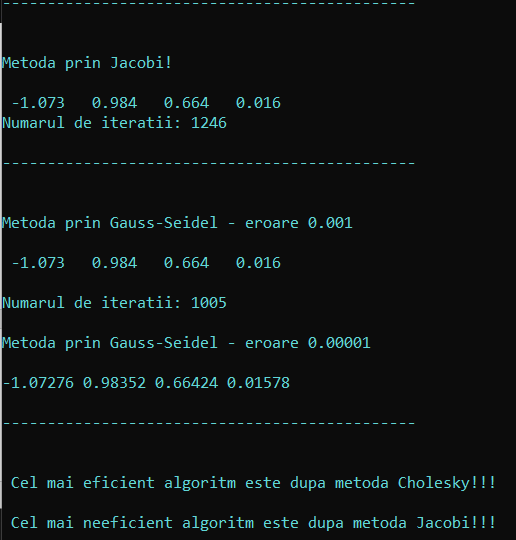
    return 0;

}

Rezultatul







**Concluzie**

In lucrarea data am studiat rezolvarea sistemelor de ecuatii lineare. In program am utilizat metodele Gauss,Cholesky, Jacobi, Gauss-Seidel. In rezultat am obtinut ca metoda Cholesky este mai eficienta deoarece efectueaza un numar de iteratii mai mic ca si metoda Gauss. Cele mai multe iteratii le-a efectuat metodele Jacobi si Gauss-Seidel, deoarece au un numar de iteratii cu mult mai mare.

In lucrarea data am invatat mai multe metode de rezolvare a sistemelor de ecuatii. Totodata am implementat aceste metode ca programe.