**MINISTERUL EDUCAŢIEI**
**Universitatea Tehnică a Moldovei**
**Facultatea Calculatoare Informatică și Microelectronică**
**Tehnologia Informației**

# Raport

**Disciplina:** Metode numerice

**Lucrarea de laborator nr. 2**

**Tema:** "REZOLVAREA NUMERICĂ A SISTEMELOR DE ECUAŢII LINIARE"

Varianta: 23

**Student:** _____ **Raevschi Grigore TI-231**

**Coordonator:** _____ **Conf. univ. Pațiuc Vladimir**

Chișinău 2024

Cuprins

# Scopul lucrării

1. Să se rezolve sistemul de ecuaţii lineare $Ax = b$, utilizând

- Metoda eliminării lui Gauss;
- Metoda lui Cholesky (metoda rădăcinii pătrate);
- Metoda iterativă a lui Jacobi cu o eroare $\varepsilon = 10^{-3}$;
- Metoda iterativă a lui Gauss-Seidel cu o eroare $\varepsilon = 10^{-3}$ şi $\varepsilon = 10^{-5}$

2. Să se determine numărul de iteraţii necesare pentru aproximarea soluţiei sistemului cu eroarea dată ε. Să se compare rezultatele.

# Ecuația liniară propusă spre rezolvare:

$$\textbf{23. } A=\begin{pmatrix} 15.1 & -0.9 & 1.2 & 0.4 \\ -0.9 & 14.6 & 0.8 & 0.7 \\ 1.2 & 0.8 & 17.6 & -0.6 \\ 0.4 & 0.7 & -0.6 & 21.3 \end{pmatrix} b=\begin{pmatrix} 9.2 \\ 8.7 \\ -10.6 \\ 9.7 \end{pmatrix},$$

*Figură 1: Varianta propusă spre rezolvare*

# Rezolvarea manuală ( Metoda eliminării lui Gauss )

Pentru a rezolva complet sistemul dat folosind metoda eliminării Gauss, voi parcurge pașii detaliați pentru a transforma matricea extinsă (A|b)(A|b)(A|b) într-o formă treptată și a obține soluția sistemului.

# Pasul 1: Formarea matricei extinse

## Pașii de rezolvare prin eliminare Gauss

1. **Construirea matricei extinse** $[A|b]$:

   Adăugăm vectorul $b$ ca o coloană suplimentară la matricea $A$, obținând astfel matricea extinsă:

$$\left(\begin{array}{cccc|c} 15.1 & -0.9 & 1.2 & 0.4 & 9.2 \\ -0.9 & 14.6 & 0.8 & 0.7 & 8.7 \\ 1.2 & 0.8 & 17.6 & -0.6 & -10.6 \\ 0.4 & 0.7 & -0.6 & 21.3 & 9.7 \end{array}\right)$$

## Pasul 1: Eliminarea elementelor sub diagonala principală din prima coloană

Folosim primul element al matricei, $15.1$, ca pivot pentru a elimina elementele din prima coloană aflate sub el. Vom face zero valorile $-0.9$, $1.2$ și $0.4$ din rândurile 2, 3 și 4.

1. **Rândul 2**: $R_2 = R_2 - \left(\frac{-0.9}{15.1}\right) R_1$

   Calculăm factorul de scalare:

$$\frac{-0.9}{15.1} \approx -0.0596$$

   Aplicăm această operație asupra rândului 2:

$$R_2 = \begin{pmatrix} -0.9 & 14.6 & 0.8 & 0.7 & | & 8.7 \end{pmatrix} + 0.0596 \cdot \begin{pmatrix} 15.1 & -0.9 & 1.2 & 0.4 & | & 9.2 \end{pmatrix}$$

$$R_2 = \begin{pmatrix} 0 & 14.654 & 0.7292 & 0.6762 & | & 9.249 \end{pmatrix}$$

2. **Rândul 3**: $R_3 = R_3 - \left(\frac{1.2}{15.1}\right) R_1$

   Factorul de scalare:

$$\frac{1.2}{15.1} \approx 0.0795$$

   Aplicăm această operație asupra rândului 3:

$$R_3 = \begin{pmatrix} 1.2 & 0.8 & 17.6 & -0.6 & | & -10.6 \end{pmatrix} - 0.0795 \cdot \begin{pmatrix} 15.1 & -0.9 & 1.2 & 0.4 & | & 9.2 \end{pmatrix}$$

$$R_3 = \begin{pmatrix} 0 & 0.8715 & 17.5046 & -0.6318 & | & -11.3366 \end{pmatrix}$$

3. **Rândul 4**: $R_4 = R_4 - \left(\frac{0.4}{15.1}\right) R_1$

   Factorul de scalare:

$$\frac{0.4}{15.1} \approx 0.0265$$

   Aplicăm această operație asupra rândului 4:

$$R_4 = \begin{pmatrix} 0.4 & 0.7 & -0.6 & 21.3 & | & 9.7 \end{pmatrix} - 0.0265 \cdot \begin{pmatrix} 15.1 & -0.9 & 1.2 & 0.4 & | & 9.2 \end{pmatrix}$$

$$R_4 = \begin{pmatrix} 0 & 0.7239 & -0.6318 & 21.2894 & | & 9.4558 \end{pmatrix}$$

## Matricea după Pasul 1:

$$\left(\begin{array}{cccc|c} 15.1 & -0.9 & 1.2 & 0.4 & 9.2 \\ 0 & 14.654 & 0.7292 & 0.6762 & 9.249 \\ 0 & 0.8715 & 17.5046 & -0.6318 & -11.3366 \\ 0 & 0.7239 & -0.6318 & 21.2894 & 9.4558 \end{array}\right)$$

**Pasul 2: Eliminarea elementelor sub diagonala principală din a doua coloană**

Continuăm cu pivotul $14.654$ din rândul 2 pentru a face zero elementele de sub el în coloana a doua.

1. **Rândul 3**: $R_3 = R_3 - \left(\frac{0.8715}{14.654}\right) R_2$

   Factorul de scalare:

$$\frac{0.8715}{14.654} \approx 0.0595$$

   Aplicăm această operație asupra rândului 3:

$$R_3 = \begin{pmatrix} 0 & 0.8715 & 17.5046 & -0.6318 & | & -11.3366 \end{pmatrix} - 0.0595 \cdot \begin{pmatrix} 0 & 14.654 & 0.7292 & 0.6762 & | & 9.249 \end{pmatrix}$$

$$R_3 = \begin{pmatrix} 0 & 0 & 17.4613 & -0.672 & | & -11.8855 \end{pmatrix}$$

2. **Rândul 4**: $R_4 = R_4 - \left(\frac{0.7239}{14.654}\right) R_2$

   Factorul de scalare:

$$\frac{0.7239}{14.654} \approx 0.0494$$

   Aplicăm această operație asupra rândului 4:

$$R_4 = \begin{pmatrix} 0 & 0.7239 & -0.6318 & 21.2894 & | & 9.4558 \end{pmatrix} - 0.0494 \cdot \begin{pmatrix} 0 & 14.654 & 0.7292 & 0.6762 & | & 9.249 \end{pmatrix}$$

$$R_4 = \begin{pmatrix} 0 & 0 & -0.6677 & 21.256 & | & 8.9963 \end{pmatrix}$$

**Matricea după Pasul 2:**

$$\left(\begin{array}{cccc|c} 15.1 & -0.9 & 1.2 & 0.4 & 9.2 \\ 0 & 14.654 & 0.7292 & 0.6762 & 9.249 \\ 0 & 0 & 17.4613 & -0.672 & -11.8855 \\ 0 & 0 & -0.6677 & 21.256 & 8.9963 \end{array}\right)$$

### Pasul 3: Eliminarea elementelor sub diagonala principală din a treia coloană

Acum folosim elementul de pe diagonala principală din a treia coloană, $17.4613$, ca pivot pentru a elimina elementul de sub el în această coloană, adică $-0.6677$ din rândul 4.

1. **Rândul 4**: $R_4 = R_4 - \left(\frac{-0.6677}{17.4613}\right) R_3$

   Calculăm factorul de scalare:

   $$\frac{-0.6677}{17.4613} \approx -0.0382$$

   Aplicăm această operație asupra rândului 4:

   $$R_4 = \begin{pmatrix} 0 & 0 & -0.6677 & 21.256 & | & 8.9963 \end{pmatrix} + 0.0382 \cdot \begin{pmatrix} 0 & 0 & 17.4613 & -0.672 & | & -11.8855 \end{pmatrix}$$

   $$R_4 = \begin{pmatrix} 0 & 0 & 0 & 21.2303 & | & 8.5429 \end{pmatrix}$$

### Matricea după Pasul 3:

$$\left(\begin{array}{cccc|c} 15.1 & -0.9 & 1.2 & 0.4 & 9.2 \\ 0 & 14.654 & 0.7292 & 0.6762 & 9.249 \\ 0 & 0 & 17.4613 & -0.672 & -11.8855 \\ 0 & 0 & 0 & 21.2303 & 8.5429 \end{array}\right).$$

## Substituția înapoi

Am obținut acum o matrice triunghiulară superioară, deci putem începe substituția înapoi pentru a găsi valorile necunoscutelor $x_4$, $x_3$, $x_2$ și $x_1$.

1. **Calculul lui $x_4$**

   Din ultima ecuație:

   $$21.2303 \cdot x_4 = 8.5429$$

   $$x_4 = \frac{8.5429}{21.2303} \approx 0.4025$$

2. **Calculul lui $x_3$**

   Din ecuația de pe rândul 3:

   $$17.4613 \cdot x_3 - 0.672 \cdot x_4 = -11.8855$$

   Înlocuim $x_4$ cu valoarea găsită:

   $$17.4613 \cdot x_3 - 0.672 \cdot 0.4025 = -11.8855$$

   $$17.4613 \cdot x_3 - 0.2705 = -11.8855$$

   $$17.4613 \cdot x_3 = -11.615$$

   $$x_3 = \frac{-11.615}{17.4613} \approx -0.6653$$

3. **Calculul lui $x_2$**

   Din ecuația de pe rândul 2:

$$14.654 \cdot x_2 + 0.7292 \cdot x_3 + 0.6762 \cdot x_4 = 9.249$$

   Înlocuim valorile pentru $x_3$ și $x_4$:

$$14.654 \cdot x_2 + 0.7292 \cdot (-0.6653) + 0.6762 \cdot 0.4025 = 9.249$$

$$14.654 \cdot x_2 - 0.4852 + 0.2721 = 9.249$$

$$14.654 \cdot x_2 - 0.2131 = 9.249$$

$$14.654 \cdot x_2 = 9.4621$$

$$x_2 = \frac{9.4621}{14.654} \approx 0.6457$$

4. **Calculul lui $x_1$**

   Din ecuația de pe rândul 1:

$$15.1 \cdot x_1 - 0.9 \cdot x_2 + 1.2 \cdot x_3 + 0.4 \cdot x_4 = 9.2$$

   Înlocuim valorile pentru $x_2$, $x_3$, și $x_4$:

$$15.1 \cdot x_1 - 0.9 \cdot 0.6457 + 1.2 \cdot (-0.6653) + 0.4 \cdot 0.4025 = 9.2$$

$$15.1 \cdot x_1 - 0.5811 - 0.7984 + 0.161 = 9.2$$

$$15.1 \cdot x_1 - 1.2185 = 9.2$$

$$15.1 \cdot x_1 = 10.4185$$

$$x_1 = \frac{10.4185}{15.1} \approx 0.6904$$

## Soluția finală

Am obținut soluția sistemului:

$$x_1 \approx 0.6904, \quad x_2 \approx 0.6457, \quad x_3 \approx -0.6653, \quad x_4 \approx 0.4025$$

# Codul programului(scris în limbajul Java)

```
import java.util.*;
import java.io.*;
import java.math.BigDecimal;

public class Lab2 {
    final static int lung = 4; // Updated to match the problem size (4x4 matrix)

    public static void main(String[] args) throws IOException {
        float[][] a = {
            {15.1f, -0.9f, 1.2f, 0.4f},
            {-0.9f, 14.6f, 0.8f, 0.7f},
            {1.2f, 0.8f, 17.6f, -0.6f},
            {0.4f, 0.7f, -0.6f, 21.3f}
        };

        float[] b = {9.2f, 8.7f, -10.6f, 9.7f};

        System.out.println("\nMatrix A:");
        for (float[] row : a) {
            for (float value : row) {
                System.out.print(value + "\t");
            }
            System.out.println();
        }

        System.out.println("\nVector b:");
        for (float value : b) {
            System.out.println(value);
        }

        GaussSeidel gaussSeidel = new GaussSeidel();
        Jacobi jacobi = new Jacobi();
        Cholesky cholesky = new Cholesky();
```

```
        float[][] l = new float[lung][lung]; // Initialize matrix l for Cholesky

        System.out.println("\nResults using Cholesky Method:");
        cholesky.cholesky(lung, b, l, a);

        System.out.println("\nResults using Jacobi Method (Error = 0.001):");
        jacobi.jacobi(lung, 0.001, a, b);

        System.out.println("\nResults using Gauss-Seidel Method (Error = 0.001):");
        gaussSeidel.gaussSeidel(lung, 0.001, a, b);

        System.out.println("\nResults using Gauss-Seidel Method (Error = 0.00001):");
        gaussSeidel.gaussSeidel(lung, 0.00001, a, b);

        // Menu-driven approach
        Scanner sc = new Scanner(System.in); // Scanner initialized here for menu
input
        int menu;

        while (true) {
            System.out.println("1 - Metoda lui Cholesky");
            System.out.println("2 - Metoda Jacobi");
            System.out.println("3 - Metoda Gauss-Seidel");
            System.out.println("0 - Exit");
            System.out.print(">>> ");
            menu = sc.nextInt();

            try {
                switch (menu) {
                    case 1:
                        System.out.println("Rezultatele:");
                        System.out.println("\nMetoda Cholesky:\n");
                        cholesky.cholesky(lung, b, l, a);
                        System.out.print("\nIntroduceți orice caracter pentru a
continua:\n>>> ");
                        System.in.read();
```

```
                        break;
                    case 2:
                        System.out.println("Rezultatele:");
                        System.out.println("\nMetoda Jacobi (Eroarea = 0.001):");
                        jacobi.jacobi(lung, 0.001, a, b);
                        System.out.print("\nIntroduceți orice caracter pentru a
continua:\n>>> ");
                        System.in.read();
                        break;
                    case 3:
                        System.out.println("Rezultatele");
                        System.out.println("\nMetoda Gauss-Seidel (Eroarea = 0.001)");
                        gaussSeidel.gaussSeidel(lung, 0.001, a, b);
                        System.out.println("\nMetoda Gauss-Seidel (Eroarea =
0.00001)");
                        gaussSeidel.gaussSeidel(lung, 0.00001, a, b);
                        System.out.print("\nIntroduceți orice caracter pentru a
continua:\n>>> ");
                        System.in.read();
                        break;
                    case 0:
                        System.out.println("Exiting the program.");
                        System.exit(0);
                    default:
                        System.out.println("Opțiune greșită!");
                        break;
                }
            } catch (IOException e) {
                System.out.println("An error occurred: " + e.getMessage());
            }
            System.out.println("\n\n\n");
        }
    }
}

class DiagonalaDominanta {
```

```
    protected int DiagDom(float[][] a, int lung) {
        float s;
        for (int i = 0; i < lung; i++) {
            s = 0;
            for (int j = 0; j < lung; j++) {
                if (i != j) {
                    s += a[i][j];
                }
            }
            if (a[i][i] < s) {
                return 0;
            } else if (a[i][i] == 0) {
                return 0;
            }
        }
        return 1;
    }
}


class GaussSeidel extends DiagonalaDominanta {
    public void gaussSeidel(int lung, double Eps, float[][] a, float[] b) throws
IOException {
        float[] x = new float[20], x1 = new float[20], d = new float[20];
        float[][] q = new float[20][20];
        float 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) != 1) {
```

```
                System.out.println("Aceasta matricea nu este diagonal dominata");
                return;
            }
            d[i] = b[i] / a[i][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]);
                }
                x[i] = ((float)t + d[i]);
            }
            max = (float) Math.abs(x[0] - x1[0]);
            for (int i = 1; i < lung; i++) {
                if ((float) Math.abs(x[i] - x1[i]) > max) {
                    max = (float) Math.abs(x[i] - x1[i]);
                }
            }
            ni++;
        } while (max > Eps);

        System.out.println("\nRezultatele x:");
        for (int i = 0; i < lung; i++) {
            System.out.println("x" + (i + 1) + " = " + BigDecimal.valueOf(x[i]));
        }
        System.out.println("\nNumarul iteratiilor = " + ni);
    }
}
```

```
class Jacobi extends DiagonalaDominanta {

    public void jacobi(int lung, double Eps, float[][] a, float[] b) throws
IOException {

        float[] x = new float[20], x1 = new float[20], d = new float[20];

        float[][] q = new float[20][20];

        float 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) != 1) {

            System.out.println("Aceasta matricea nu este diagonal dominata");

            return;

        }


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

            d[i] = b[i] / a[i][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] * x1[j];
                }
                x[i] = t + d[i];
            }
            max = (float) Math.abs(x[0] - x1[0]);
            for (int i = 1; i < lung; i++) {
                if ((float) Math.abs(x[i] - x1[i]) > max) {
                    max = (float) Math.abs(x[i] - x1[i]);
                }
            }
            ni++;
        } while (max > Eps);

        System.out.println("\nRezultatele x:");
        for (int i = 0; i < lung; i++) {
            System.out.println("x" + (i + 1) + " = " + BigDecimal.valueOf(x[i]));
        }
        System.out.println("\nNumarul iteratiilor = " + ni);
    }
}

class Cholesky {
    public void cholesky(int lung, float[] b, float[][] l, float[][] a) throws
IOException {
        if (Pozitiv(lung, a) == 0) {
            System.out.println("\nAceasta matricea nu este pozitiv definita!");
            return;
        } else if (Simetric(lung, a) == 0) {
            System.out.println("\nAceasta matricea nu este simetrica!");
            return;
        }
        determinare(lung, b, l, a);
    }

    private void determinare(int lung, float[] b, float[][] l, float[][] a) {
```

```
        factor(lung, l, a);
        float[] y = new float[10], x = new float[10];
        float t;
        y[0] = b[0] / l[0][0];
        for (int i = 1; i < lung; i++) {
            t = 0;
            for (int j = 0; j < i; j++) {
                t += l[i][j] * y[j];
            }
            y[i] = (b[i] - t) / l[i][i];
        }
        x[lung - 1] = y[lung - 1] / l[lung - 1][lung - 1];
        for (int i = lung - 2; i >= 0; i--) {
            t = 0;
            for (int j = lung - 1; j > i; j--) {
                t += l[j][i] * x[j];
            }
            x[i] = (y[i] - t) / l[i][i];
        }

        for (int i = 0; i < lung; i++) {
            System.out.println("\nx" + (i + 1) + " = " + x[i]);
        }
    }

    private void factor(int lung, float[][] l, float[][] a) {
        float s;
        for (int i = 0; i < lung; i++) {
            for (int j = 0; j < i; j++) {
                s = 0;
                for (int k = 0; k < j; k++) {
                    s += l[i][k] * l[j][k];
                }
                l[i][j] = (a[i][j] - s) / l[j][j];
            }
```

```
            s = 0;
            for (int k = 0; k < i; k++) {
                s += l[i][k] * l[i][k];
            }
            l[i][i] = (float) Math.sqrt(a[i][i] - s);
        }
    }

    private int Pozitiv(int lung, float[][] a) {
        float d;
        for (int i = 0; i < lung; i++) {
            d = a[i][i];
            if (d <= 0) {
                return 0;
            }
        }
        return 1;
    }

    private int Simetric(int lung, float[][] a) {
        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;
    }
}
```

## Rezultatul programului prin metoda lui Cholesky

```
Results using Cholesky Method:

Intermediate results (y):
y1 = 2,3676
y2 = 2,4249
y3 = -2,8450
y4 = 1,8528

Final results (x):
x1 = 0,6906
x2 = 0,6556
x3 = -0,6655
x4 = 0,4021
```

## Rezultatul programului prin metoda Jacobi

```
Results using Jacobi Method (Error = 0.001):

Rezultatele x:
x1 = 0.6905485391616821
x2 = 0.6556130051612854
x3 = -0.6654222011566162
x4 = 0.4021551311016083

Numarul iteratiilor = 4
```

## Rezultatul programului prin metoda Gauss-Seidel

```
Results using Gauss-Seidel Method (Error = 0.001):

Rezultatele x:
x1 = 0.6905614137649536
x2 = 0.6556369066238403
x3 = -0.6654481887817383
x4 = 0.40213900804519653

Numarul iteratiilor = 3
```

## Rezultatul programului prin metoda Gauss-Seidel

```
Results using Gauss-Seidel Method (Error = 0.00001):

Rezultatele x:
x1 = 0.6905803084373474
x2 = 0.6556428670883179
x3 = -0.6654504537582397
x4 = 0.4021383821964264

Numarul iteratiilor = 5
```

# Compararea rezultatelor

| Metoda | Radacina | | Iterații | Eroarea |
|---|---|---|---|---|
| | x | y | | |
| Cholesy | x1 = 0,6906<br>x2 = 0,6556<br>x3 = -0,6655<br>x4 = 0,4021 | y1 = 2,3676<br>y2 = 2,4249<br>y3 = -2,8450<br>y4 = 1,8528 | - | = |
| Jacobi | x1 = 0.69054853916<br>x2 = 0.65561300516<br>x3 = -0.6654222011<br>x4 = 0.40215513110 | | 4 | 0.001 |
| Gauss-Seidel | x1 = 0.690561413764<br>x2 = 0.655636906623<br>x3 = -0.66544818878<br>x4 = 0.402139008045 | | 3 | 0.001 |
| | x1 = 0.690580308437<br>x2 = 0.655642867088<br>x3 = -0.66545045375<br>x4 = 0.402138382196 | | 5 | 0.00001 |

# Concluzie

În aceasta lucrare de laborator am facut cunostinta cu trei metode de rezolvare numerica a sistemelor de ecuatii liniare: Metoda Cholesky, Metoda Jacobi și Metoda Gauss - Seidel. În urma rezultatelor obținute din exemplul dat, cea mai eficientă metodă iterativă de calculare a rădăcinelor sistemelor de ecuații a fost metoda Gauss-Seidel.